ABSTRACT

Title of dissertation: LEARNING EXPLAINABLE FACIAL FEATURES FROM NOISY UNCONSTRAINED VISUAL DATA

Emily M. Hand
Doctor of Philosophy, 2018

Dissertation directed by: Professor Rama Chellappa
Department of Computer Science

Attributes are semantic features of objects, people, and activities. They allow computers to describe people and things in the way humans would, which makes them very useful for recognition. Facial attributes - *gender*, *hair color*, *makeup*, *eye color*, etc. - are useful for a variety of different tasks, including face verification and recognition, user interface applications, and surveillance, to name a few. The problem of predicting facial attributes is still relatively new in computer vision. Because facial attribute recognition is not a long-studied problem, a lack of publicly available data is a major challenge. As with many problems in computer vision, a large portion of facial attribute research is dedicated to improving performance on benchmark datasets. However, it has been shown that research progress on a benchmark dataset does not necessarily translate to a genuine solution for the problem. This dissertation focuses on learning models for facial attributes that are robust to changes in data, i.e. the models perform well on unseen data. We do this by taking cues from human recognition, and translating these ideas into

deep learning techniques for robust facial attribute recognition. Towards this goal, we introduce several techniques for learning from noisy unconstrained visual data: utilizing relationships among attributes, a selective learning approach for multi-label balancing, a temporal coherence constraint and a motion-attention mechanism for recognizing attributes in video, and parsing faces according to attributes for improved localization.

We know that facial attributes are related, e.g. *heavy makeup* and *wearing lipstick* or *male* and *goatee*. Humans are capable of recognizing and taking advantage of these relationships. For example, if a face of a subject is occluded, and facial hair can be seen, then the likelihood that the subject being *male* should increase. We introduce several methods for implicitly and explicitly utilizing attribute relationships for improved prediction.

Some attributes are more common than others in the real world, e.g. *male* v. *bald*. These disparities are even more pronounced in datasets consisting of posed celebrities on the red carpet (i.e. there are very few celebrities not wearing makeup). These imbalances can cause a facial attribute model to learn the bias in the dataset, rather than a true representation for the attribute. To alleviate this problem, we introduce selective learning, a method of balancing each batch in a deep learning algorithm according to each attribute given a target distribution. Selective learning allows a deep learning algorithm to learn from a balanced set of data at each iteration during training, removing the bias from the label imbalance.

Learning a facial attribute model from image data, and testing on video data gives unexpected results (e.g. *gender* changing between frames). When working

with video, it is important to account for the temporal and motion aspects of the data. In order to stabilize attribute predictions in video, we utilized weakly-labeled data and introduced time and motion constraints in the model learning process. Introducing temporal coherence and motion-attention constraints during learning of an attribute model allows the use of weakly-labeled data, which is essential when working with video.

Framing the problem of facial attribute recognition as one of semantic segmentation, where the goal is to predict attributes at each pixel, we are able to reduce the effect of unwanted relationships between attributes (e.g. *high cheekbones* and *smiling*).

Robust facial attribute recognition algorithms are necessary for improving the applications which use these attributes. Given limited data for training, we develop several methods for learning explainable facial features from noisy unconstrained visual data, introducing several new datasets labeled with facial attributes and improving over the state-of-the-art.

LEARNING EXPLAINABLE FACIAL FEATURES FROM NOISY UNCONSTRAINED VISUAL DATA

by

Emily M. Hand

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
2018

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Dr. Carlos D. Castillo
Professor David Jacobs
Professor Donald Perlis
Professor Min Wu

Dedication

To my husband, Christopher, for pushing me to work my hardest every day, for inspiring me with your creativity and talent, and for always being incredibly supportive.

Acknowledgments

I would like to thank my advisor, Professor Rama Chellappa, for giving me the opportunity to work on interesting and important problems over the past four years. Rama was available when I needed him, but he did not hover. He let me choose my own path and gave suggestions along the way. He encouraged me to participate in research outside of UMD, giving me a well-rounded research experience. He was very supportive of my plan to join academia after graduation and routinely gave me advice about university faculty life. From him I have learned the values and challenges of research, teaching, and service as a faculty member and someday I will pass these lessons on to my students.

I would also like to thank Dr. Carlos Castillo for his invaluable help and advice. I spent many hours collaborating with Carlos, and discussing potential research directions. Carlos helped me learn how to better communicate my ideas and experiments in my technical papers. He taught me how to make the absolute best case possible for my work. With his help, I was able to greatly improve both my technical writing and my ability to work through the details of a research idea.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Facial attribute recognition is a new and interesting problem in the field of computer vision. Being able to automatically recognize semantic features of faces in the way that humans do is very useful for many applications. Facial attributes can be used for face recognition and verification, surveillance, human-computer interaction, and many other applications. Existing approaches for facial attribute prediction focus on improving performance on a benchmark dataset: CelebA [2]. However, the state-of-the-art methods for facial attribute recognition on CelebA do not take advantage of techniques from human recognition. In this dissertation, we take cues from human vision and perception and translate those into novel deep learning approaches for facial attribute recognition. We focus on building robust models for facial attributes that can be used for any of the aforementioned applications. In order for these models to perform well on unseen data, we take cues from the ways in which humans recognize and understand visual data to learn more robust models with deep learning techniques. With the methods discussed in this dissertation, we are able to accurately describe faces in images and video.

1.2 Using Attribute Relationships

When first introduced, facial attributes were treated as independent features, and separate models were used for the prediction of each attribute [3] [4] [5] [6]. Recently, attribute prediction algorithms have been taking advantage of attribute relationships [7] [8] [9] [10]. Attributes can be strongly related, such as *heavy makeup* and *wearing lipstick* or *male* and *goatee* and many others. In order to take these relationships into account during training of our model, we introduce a multi-task deep convolutional neural network (MCNN) with an auxiliary network at the top (AUX) which takes advantage of attribute relationships for improved classification. We call our final network MCNN-AUX. MCNN-AUX uses attribute relationships in three ways: by sharing the lowest layers for all attributes, by sharing the higher layers for spatially-related attributes, and by feeding the attribute scores from MCNN into the AUX network to find score-level relationships. Using MCNN-AUX rather than individual attribute classifiers, we are able to reduce the number of parameters in the network from 64 million to fewer than 16 million and reduce the training time by a factor of 16. We demonstrate the effectiveness of our method by producing results on two challenging publicly available datasets (CelebA and LFWA) achieving state-of-the-art performance on most attributes.

1.3 Selective Learning

Despite the usefulness of facial attributes, to date there is only one large-scale dataset labeled with these features, CelebA [2]. Impressive results have been achieved on this dataset, but it exhibits a variety of very significant biases. As CelebA contains mostly frontal idealized images of celebrities, it is difficult to generalize a model trained on this data for use on another dataset (of non celebrities). A typical approach to dealing with imbalanced data involves sampling the data in order to balance the positive and negative labels, however, with a multi-label problem this becomes a non-trivial task. By sampling to balance one label, we affect the distribution of other labels in the data. To address this problem, we introduce a novel selective learning method for deep networks which adaptively balances the data in each batch according to the desired distribution for each label. The bias in CelebA can be corrected for in this way, allowing the network to learn a more robust attribute model. We argue that without this multi-label balancing, the network cannot learn to accurately predict attributes that are poorly represented in CelebA. We demonstrate the effectiveness of our method on the problem of facial attribute prediction on CelebA, LFWA, and the new University of Maryland Attribute Evaluation Dataset (UMD-AED), outperforming the state-of-the-art on each dataset.

1.4 Temporal Coherence and Motion Attention

Recent research progress in facial attribute recognition has been dominated by small improvements on CelebA [2]. We extend attribute prediction research to unconstrained videos. Applying attribute models trained on CelebA – a still image dataset – to video data highlights several major problems with current models, including the lack of consideration for both time and motion. Many facial attributes (e.g. gender, hair color) should be consistent throughout a video, however, current models do not produce consistent results. We introduce two methods to increase the consistency and accuracy of attribute responses in videos: a temporal coherence constraint, and a motion-attention mechanism. Both methods work on weakly labeled data, requiring attribute labels for only one frame in a sequence, which we call the anchor frame. The temporal coherence constraint moves the network responses of non-anchor frames toward the responses of anchor frames for each sequence, resulting in more stable and accurate attribute predictions. We use the motion between anchor and non-anchor video frames as an attention mechanism, discarding the information from parts of the non-anchor frame where no motion occurred. This motion-attention focuses the network on the moving parts of the non-anchor frames (i.e. the face). Since there is no large-scale video dataset labeled with attributes, it is essential for attribute models to be able to learn from weakly labeled data. We demonstrate the effectiveness of the proposed methods by evaluating them on the challenging YouTube Faces video dataset [11]. The motion-attention and temporal coherence methods outperform attribute models trained on CelebA, as well as those

fine-tuned on video data. This work is the first to address the problem of facial attribute prediction in video.

1.5 Parsing Faces with Attributes

Many facial attributes are related (e.g. *gender* and *mustache*, *attractive* and *heavy makeup*, etc.). However, we are able to recognize these attributes independently. Humans are capable of recognizing *gender* without superficial cues from hair length or makeup. Deep learning algorithms place a significant weight (we argue too much) on these relationships when learning to recognize facial attributes. We introduce face parsing with attributes as a way to de-emphasize relationships between facial attributes, allowing the model to learn a more robust representation of the attributes. Face parsing provides an additional level of supervision in our DL facial attribute framework. Face parsing with facial attributes is similar to semantic segmentation in that we want to classify every pixel in the face as belonging to some class. However, face parsing with attributes differs from semantic segmentation in that every pixel can have multiple labels (e.g. *high cheekbones* and *rosy cheeks* come from the same area of the face and so that area would have both labels). Face parsing with attributes produces a map that indicates which attributes are present at each pixel. In face parsing with attributes, an input image results in multiple maps, where each map indicates the location of the particular attribute in the image. The proposed method, AttParseNet, combines facial attribute recognition with face parsing, out-performing the state-of-the-art on three facial attribute benchmark

datasets: CelebA, LFWA, and UMD-AED. This work was the first to address the problem of parsing faces according to semantic attributes.

1.6 Contributions

In this dissertation we make the following contributions:

1. We introduce a multitask network (MCNN-AUX) for attribute recognition utilizing implicit and explicit relationships among attributes for improved prediction.

2. We introduce selective learning, a multi-label balancing technique for deep learning allowing for learning of a more robust representation of facial attributes from severely imbalanced data.

3. We introduce two methods for transferring attribute models trained on images to video data: temporal coherence and motion-attention.

4. We frame the problem of facial attribute recognition as one of parsing faces according to attributes, achieving state-of-the-art results.

5. We released a new facial attribute recognition evaluation dataset: University of Maryland Attribute Evaluation Dataset (UMD-AED). We labeled four frames from every video of YouTube Faces with facial attributes, and made the data publicly available for future research on attribute recognition in video. We also released automatically generated facial attribute segments for CelebA, allowing for future research in the direction of parsing faces with attributes.

1.7 Organization

The remainder of this dissertation is organized as follows. Chapter 2 discusses the relevant literature in attribute recognition, multi-task learning, semantic segmentation, and multiple instance learning. Chapter 3 details our work on utilizing relationships between attributes in order to improve prediction. In chapter 4 we discuss our approach to the multi-label balancing problem in deep learning applications, which we call selective learning. Chapter 5 details our temporal coherence and motion-attention methods for adapting a model trained on still images for use on video. We re-frame the problem of facial attribute recognition as one of parsing faces according to attributes in chapter 6. Finally, we conclude and discuss future research directions in chapter 7.

Chapter 2: Related Work

There has been many years of research in gender recognition, attribute recognition, domain adaptation, video processing, face parsing, and multiple instance learning. We review the relevant literature here.

2.1 Gender Recognition

There have been decades of research on gender recognition. Humans are capable of determining the gender of a face with very high accuracy and pinpointing exactly how we do this has been of interest for many years [12].

Feature extraction has been the main source of improvement for gender recognition algorithms with SVMs and neural networks being the most common classifiers until recently. In some of the earlier gender recognition algorithms, pixel values were used directly as features for classification [13] [14]. [15] showed that a neural network could learn gender reliably from very small images - 8x6 - ones which humans could not identify as faces. Pixel values as features are very difficult to deal with as their complexity increases with the size of the image, making it infeasible to use large images. As a result, dimensionality reduction techniques gained popularity for use on pixel value features. PCA was the dimensionality reduction tool of choice in gen-

der recognition [16] as well as age and ethnicity recognition [17]. [18] used PCA and DCT for dimensionality reduction of face images. Independent component analysis was also used for gender recognition of frontal face images [19]. Rectangle features were introduced by [20] for face detection. They were useful in gender and ethnicity recognition as well with [21] and [22] extending their work.

Local Binary Pattern (LBP) features were introduced in 1994, and have been successfully used for many different computer vision problems [23]. Gender recognition was no exception, with [24] using features made up of LBP histograms extracted from small patches of the face, [25] using the distance between a reference histogram and the LBP histogram from a test image to classify the image, and [26] using Adaboost to choose the best LBP features for gender recognition. [27] used a multi-scale approach combining LBP, and pixel features. [28] combined LBP features with contrast features for their gender classifier. Many different variations on LBP have been introduced. The local directional pattern utilizes gradient magnitude to characterize the texture by edge responses in different directions [29]. Interlaced derivative patterns use a four-channel derivative image to construct a feature vector [30]. [31] introduced a new feature combining centralized binary pattern with Gabor gradient magnitude to get Centralized Gabor Gradient Histogram (CGGH).

Gabor wavelets were introduced in [32], and have been popular in gender recognition [33] [34]. [35] used a hierarchy of Gabor and Laplace features for gender recognition. Combining LBP with Gabor wavelets, [36] introduced the local Gabor binary mapping pattern feature for gender classification.

SIFT features were introduced in [37] and were the most popular features

for classification problems up until the rise of CNN-based features. [38] used SIFT features for gender classification, while [39] used boosted SIFT features, and [40] combined SIFT and Gabor features.

In addition to the appearance-based features mentioned above, there have been many geometric features proposed for gender recognition. [41] used fiducial distances as features, while [42] combined appearance-based features - such as LBP and discrete cosine transform - with their Geometrical Distance Feature (GDF).

With decades of research in automatic gender recognition, researchers have found that gender classification is affected by age and ethnicity [43] [44] [45]. [46] used a dropout-support vector machine for age and gender recognition. They later extended this work to use deep CNNs for feature extraction and classification in [47]. In recent years, the problem of gender classification has transformed into the problem of facial attribute classification.

2.2 Attribute Recognition

2.2.1 Objects

Work on attributes for objects began with [48] where the authors shifted the focus from identifying objects to describing them using attributes. Learning attributes, rather than classes, allows for describing unseen objects. Lampert et al. demonstrated this, producing results on a large-scale animal dataset annotated with attributes [49]. Following up on this, [50] took the arrangement and interactions of attributes into consideration when recognizing objects, allowing for detailed descrip-

tions of objects which have not yet been seen. Learning attributes and object classes jointly improved object recognition whether it was done iteratively or by modeling attribute correlations [51] [52]. In [53], each scene was represented as a collection of objects, so the objects were used as attributes for scene classification. Focusing on attribute correlations which are harmful for recognition, [54] introduced a method which encouraged related attributes to have similar features and unrelated attributes to have dissimilar features. [55] used a mult-task network to learn attributes for animals and clothing, and showed improved results over individual networks.

2.2.2 Activities

Utilizing both manually-specified and data-driven attributes, [56] was one of the first works to use attributes for human action recognition. [57] used a generative model to learn the distribution and dynamics of activities in an attribute space. NuActive used attributes for recognizing unseen human activities with an active-learning approach to reinforce the recognition accuracy of the algorithm [58]. [59] proposed a variation of a Conditional Random Field for activity recognition using attributes, focusing on recognizing unseen activities as well. Yao et al. jointly learned a bases of attributes and action parts for human action recognition in still images [60]. Action parts included objects and poselets related to the action. Using attributes to regularize a multi-task classifier for action recognition, [61] imposed attribute constraints on the actions according to a manually defined class-attribute matrix. [62] aimed to find the best set of attributes for action recognition by auto-

matically choosing a subset of manually specified and data driven attributes. Using a subset of attributes for action recognition boosted performance over using the entire attribute set.

2.2.3 People and Faces

Approaching the problem of person search in surveillance, Vaquero et al. used attributes rather than identity, creating a system which allowed a user to specify attributes for a search [63]. [64] used separate attribute classifiers for each pose in order to identity attributes of people in images from personal photo albums. Focusing on part extraction for attribute recognition, [65] proposed an appearance-based approach for part extraction. [66] proposed a model which recognizes both human attributes and actions using part templates. Wang et al. used a part-based model for attribute recognition, utilizing RGB-D data [67]. Using attributes for pedestrian re-identification, [68] employed a multi-label CNN to recognize attributes using overlapping body parts as input. [69] proposed jointly learning all attributes using the entire person as input, ignoring individual parts. Pose Aligned Networks for Deep Attributes (PANDA) achieved state-of-the-art performance by combining part-based models with deep learning to train pose-normalized CNNs for attribute classification [6].

Facial attributes - *gender*, *hair color*, *eye color*, etc. - have been the most popular and have been successful in face verification and recognition [3] [4]. [3] first introduced the concept of facial attributes for face verification, following up on that

work with [4]. They used 65 - and then 73 - binary attributes as face descriptors. Even before this, Kumar et al. used attributes for image search in their FaceTracer work, predicting attributes using a combination of SVMs and Adaboost [70]. In [71], rather than manually labeling attributes, they trained one SVM for each pair of people in a dataset so that each SVM could distinguish between the two people on which it was trained. The feature for each image was then constructed by classifying the image using the pair SVMs, describing each person by their likeness to the people in the training dataset. Facial attributes have also found success in image search and retrieval as they can be used to search a database of images very quickly [3] [4] [9].

With the release of two face datasets with attribute labels, great advances have been made in the recognition of facial attributes in the past few years [5]. [5] used two deep CNNs, one for localizing the face in the image (LNet), and one for attribute prediction (ANet). Their method, LNet+ANet, outperformed PANDA and FaceTracer on the CelebA dataset [70] [6]. Using wearable cameras, collecting face tracks with weather and location metadata, [72] achieved state-of-the-art results on attribute prediction by first training a verification network on the wearable camera data, then fine-tuning the network for attribute prediction. In [8], the authors try to adjust for the imbalance in CelebA by introducing a mixed objective loss, which adjusts the back-propagation weights according to a given target distribution.

Much of the past work has generally considered attributes to be independent, with [3], [6], and [5] training a separate classifier for each attribute. There have been a few exceptions, however. [9] used the correlation amongst attributes to improve image ranking and retrieval, learning pairwise correlations based on the outputs of

independently trained attribute classifiers.

2.3 Multi-Task Learning

Multi-task learning (MTL) is a way of solving several related problems simultaneously, utilizing shared information [73] [74] [75]. MTL has found success in the domains of facial landmark localization, pose estimation, action recognition, face detection, and many more [76] [77] [78] [79] [80] [81]. MTL has been applied to video data for both recognition and tracking [82] [83]. Utilizing pose as a part of MTL for face recognition creates a face recognition system which is robust to extreme changes in pose [84] [78].

In [85], [51], and [52] attributes and object classes were learned jointly to improve overall object classification performance. [51] used multiple instance learning to detect and recognize objects in images by learning attribute-object pairs. [52] used an undirected graph to model the correlation amongst attributes in order to improve object recognition. In [85], attributes and objects shared a low-dimensional representation allowing for regularization of the object classifier.

Facial attributes fit nicely into a MTL framework, as predicting each attribute is a separate problem, but each set of attributes share the same face. Combining the problem of facial landmark localization with facial pose and attribute recognition the MTL framework, [80] found that the landmark localization was much more robust. [80] used three attributes: *gender*, *smiling* and *wearing glasses* to improve facial landmark localization. [77] used MTL to jointly learn face detection, landmarks,

pose, and *gender* combining features from intermediate layers in order to learn the different tasks. Learning individual CNNs for each attribute, then performing MTL at the feature level, [55] was able to accurately predict clothing attributes. [7] used a Restricted Boltzmann Machine (RBM) for multi-task attribute learning, achieving state-of-the-art results on several large-scale attribute datasets. Addressing the problem of dataset bias in a multi-task setting, [8] introduced a mixed-objective optimization network for attribute prediction, achieving state-of-the-art results.

2.4 Domain Adaptation

There have been many different methods for domain adaptation over the years [86]. Object recognition has benefitted from domain adaptation methods, especially since the introduction of a benchmark dataset [87]. Semi-supervised approaches have dominated the field with dictionary learning methods [88] [89], and metric learning methods [87]. Many unsupervised approaches have been introduced as well, with dictionary [90] and manifold-based methods [91] [92] being the most popular. Face recognition can easily be framed as a domain adaptation problem with faces in different poses and with different illuminations and resolutions contributing to domain shifts [93] [94] [95].

[96] tackled the problem of domain adaptation of clothing attributes from ideal images to images taken in unconstrained environments. The authors used a two-stream CNN to model the two domains in separate paths, using connections between the two paths to ensure that the features are similar for both domains.

Their architecture worked in unsupervised and supervised settings.

[8] addresses the problem of dataset bias in a multi-label setting. The authors introduce a Mixed-Objective Optimization Network (MOON) for attribute recognition, by weighting the back-propagation error for each attribute according to a given target distribution.

In [97], the feature extraction portion of the network fed into two different predictor portions: the class predictor, and the domain predictor. As the domain predictor backpropagated the error, it reversed the gradient when it passed through the feature extraction portion of the network. This allowed the network to learn the class labels while keeping the feature distributions for the two domains similar.

Bootstrapping is a popular approach for unsupervised domain adaptation [98] [99] [100] [101]. Bootstrapping consists of training a model on the source data, testing it on the target data and iteratively retraining the model adding samples from the target data to the training set.

2.5 Video Processing

There have been several decades of research in automated video processing [102] [103] [104] [105]. Here we review some recent publications which are related to our work.

There are several datasets labeled with attributes for actions, however the attributes are labeled for each action, not for each video, and certainly not for each frame. For example, for the action *applying lipstick*, there is an attribute *arm*

up. If in some frame the subject's arm is not up, that frame is still labeled as such [106] [107].

The concept of temporal coherence and feature stability in video has been applied to deep networks in the past. In [108], the authors introduced the concept of steady feature analysis, which aims to learn invariant features from unlabeled data for use in recognition tasks. Rather than encouraging features between video frames to be similar, steady feature analysis encourages feature changes to be smooth, placing constraints on the higher order derivatives of the feature space. Altering stochastic gradient descent to apply a coherence constraint to unlabeled video frames while at the same time learning from images with labels, [109] learned models for different recognition problems by leveraging labeled and unlabeled information. To perform unsupervised feature learning using autoencoders, [110] used temporal coherence to leverage unlabeled data. In [111], they learned feature representations with unlabeled video using tracking as the only supervision. They enforced triplet constraints at every frame according to the query, the tracked object and a random patch from the frame that does not overlap with the tracked object.

In [112], the authors used the magnitude of the optical flow to amplify features for action recognition. The idea here is that parts of the image with more motion than others will correspond to higher weighted features for action recognition.

2.6 Face Parsing

In semantic segmentation, the goal is to assign a class label to every pixel in an image, effectively segmenting it into its parts. The problem of semantic segmentation has been studied for many years. We review the relevant literature here. Face parsing is a form of semantic segmentation where the goal is to segment the face into its parts (eyebrows, mouth, nose, etc.). For many years, Conditional Random Fields (CRFs) were used by all state-of-the-art methods for face parsing [113] [114] [115].

Like with many other fields, DL became the new state-of-the-art in face parsing and semantic segmentation. With the introduction of Fully Convolutional Networks (FCNs) in [116], DL became the go-to method for semantic segmentation. There is not as much work on face parsing as on semantic segmentation because the focus continues to be on facial landmark localization rather than segmenting faces into their parts. [117] combines CNNs with CRFs and introduces a nonparametric prior from exemplar images for improved face parsing. [118] uses a hierarchical deep learning approach focusing on parsing faces with partial occlusions. More recently, [119] combines facial alignment with segmentation, using a shared representation to improve learning of both tasks for the purpose of virtual makeup and face swapping. Attention has also been introduced as a way to improve semantic segmentation when working with multi-scale images [120].

2.7 Data

We use many of the same datasets for our work, so we outline them here.

CelebA was collected for attribute classification and was labeled with the 40 binary facial attributes [5]. The CelebA dataset consists of 200,000 images: 160,000 for training and 20,000 each for validation and testing.

The well-known LFW dataset has been used mostly for face-verification [121]. It contains images of faces with varying pose, illumination, and resolution. There are a total of 13230 face images in the dataset. Pose and illumination are the biggest challenges in this dataset. Binary labels were recently added for the 40 attributes in CelebA making it LFWA [5]. The LFWA dataset contains 13,143 images with 6,263 for training and 6,880 for testing. LFWA consists of still images of celebrities, so it is similar to CelebA, but with lower resolution images.

YouTubeFaces is a video face verification dataset. It consists of $3,425$ videos of celebrities from YouTube, with a total of roughly $620,000$ frames [122]. The data varies significantly from CelebA and LFWA in quality, resolution, lighting, and pose.

Chapter 3: Using Attribute Relationships for Improved Prediction

3.1 Overview

Improving the accuracy of attribute classifiers is an important first step in any application which uses these attributes. In most works to date, attributes have been considered independent of each other. However, attributes can be strongly related, such as *heavy makeup* and *wearing lipstick* as well as *male* and *goatee* and many others. We propose a multi-task deep convolutional neural network (MCNN) with an auxiliary network at the top (AUX) which takes advantage of attribute relationships for improved classification. We call our final network MCNN-AUX. MCNN-AUX uses attribute relationships in three ways: by sharing the lowest layers for all attributes, by sharing the higher layers for spatially-related attributes, and by feeding the attribute scores from MCNN into the AUX network to find score-level relationships. Using MCNN-AUX rather than individual attribute classifiers, we are able to reduce the number of parameters in the network from 64 million to fewer than 16 million and reduce the training time by a factor of 16. We demonstrate the effectiveness of our method by producing results on two challenging publicly available datasets achieving state-of-the-art performance on many attributes.

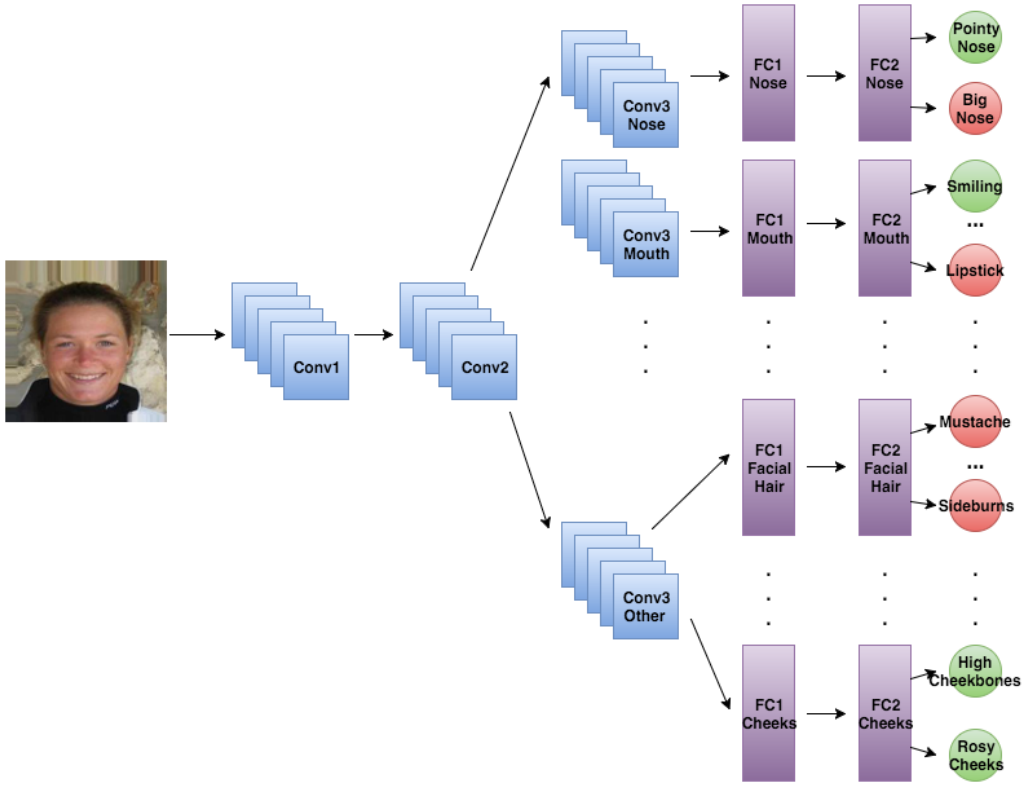


Figure 3.1: Overview of MCNN. The input image on the left is cropped to 227x227 and the training mean is subtracted. The image is then passed through the network producing attribute scores, which are then thresholded to give a positive or negative response. The red attributes indicate a lack of the attribute and the green attributes indicate a positive instance.

3.2 Proposed Approach

3.2.1 Multi-Task CNN (MCNN)

The proposed MCNN takes an image as input and outputs 40 separate attribute scores, which are then thresholded to obtain binary outputs. We describe the details of the architecture below. Figure 3.1 shows the MCNN architecture. Conv1 consists of 75 7x7 convolution filters, and it is followed by a ReLU, 3x3 Max Pooling, and 5x5 Normalization. Conv2 has 200 5x5 filters and it is also followed by a ReLU, 3x3 Max Pooling, and 5x5 Normalization. Conv1 and Conv2 are shared for all attributes. This allows for learning of implicit relationships amongst attributes at a lower level. After Conv2, groupings are used to separate the layers. We use nine groups for the MCNN: *Gender*, *Nose*, *Mouth*, *Eyes*, *Face*, *AroundHead*, *FacialHair*, *Cheeks*, and *Fat*. The attributes in each group are listed in table 3.1. There are six Conv3s: one each for *Gender*, *Nose*, *Mouth*, *Eyes*, and *Face*, and one for the remaining groups - Conv3Other. Each Conv3 has 300 3x3 filters and is followed by a ReLU, 5x5 Max Pooling and 5x5 Normalization. The Conv3s are followed by fully connected layers, FC1. There are 9 FC1s - one for each group. Each FC1 is fully connected to the corresponding previous layer, with Conv3Other connected to the FC1s for *AroundHead*, *FacialHair*, *Cheeks*, and *Fat*. Every FC1 has 512 units and is followed by a ReLU and a 50% dropout to avoid overfitting. Each FC1 is fully connected to a corresponding FC2, also with 512 units. The FC2s are followed by a ReLU and a 50% dropout. Each FC2 is fully connected to one output node for each

Group	Attributes
Gender	Male
Nose	Big Nose, Pointy Nose
Mouth	Big Lips, Lipstick, Mouth Slightly Open, Smiling
Eyes	Arched Eyebrows, Bags Under Eyes, Bushy Eyebrows, Eyeglasses, Narrow Eyes
Face	Attractive, Blurry, Heavy Makeup, Oval Face, Pale Skin, Young
AroundHead	Balding, Bangs, Black Hair, Blond Hair, Brown Hair, Earrings, Gray Hair, Hat, Necklace, Necktie, Receding Hairline, Straight Hair, Wavy Hair
FacialHair	5 o'clock Shadow, Goatee, Mustache, No Beard, Sideburns
Cheeks	High Cheekbones, Rosy Cheeks
Fat	Chubby, Double Chin

Table 3.1: Attributes and their corresponding groupings.

of the attributes in that group. For example, FC2*Nose* is connected to output nodes for *Big Nose* and *Pointy Nose*. The grouping of attributes in the Conv3, FC1, and FC2 layers allows for the learning of explicit relationships among attributes from similar locations in the face image.

The nine groups were manually chosen according to attribute location. Some groupings were separated from others and some were absorbed into others through experimentation on the validation portion of the CelebA dataset giving the groupings in table 3.1. *Male* was kept separate from all other attributes as we found that *male* classification was improved by sharing layers with other attributes, but the classification of the other attributes suffered. We found the best compromise was

to include *male* in the shared Conv1 and Conv2 layers and then to have separate Conv3, FC1, and FC2 layers.

We use Caffe [123] for our implementation, training, and testing of MCNN and MCNN-AUX. We use a sigmoid cross-entropy loss for all attribute scores to facilitate training. As preprocessing steps, the training mean is subtracted from the images and they are cropped randomly with a size of 227x227. This helps the network to be robust to shifts in the input. Unlike other attribute classification methods, we do not perform any alignment or part extraction in the preprocessing stage. Both alignment and part extraction are expensive and error-prone processes, and so we save time and avoid problems associated with poor alignment by skipping these steps. Our method is also more applicable to real-world imagery for which alignment may be challenging.

If we were to use an independent CNN for each attribute, following the architecture of one path in the MCNN - 3 convolutional layers and 3 fully connected layers - each CNN would have over 1.6 million parameters. For all 40 attributes, there would be over 64 million parameters. Using MCNN, we reduce the number of parameters to fewer than 16 million, over four times fewer.

3.2.2 MCNN-AUX

After training the MCNN, we add one fully connected layer after the output of the trained MCNN. This layer creates the two-layer AUX network. Figure 3.2 shows the connection between MCNN and AUX. The input to AUX is the attribute scores

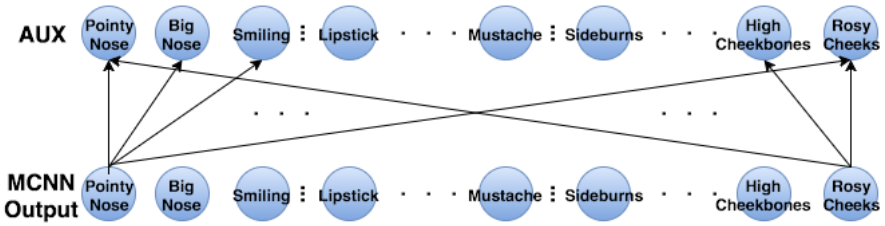


Figure 3.2: AUX network architecture. The output of the MCNN is fully connected to the final layer creating the 2-layer AUX network.

from the trained MCNN, and the output is the final attribute scores. Starting with the weights from the trained MCNN, we learn the weights for the AUX portion of the network, freezing the weights from the MCNN. The AUX network allows for learning score-level attribute relationships. The AUX network adds only 1600 parameters to the fewer than 16 million from MCNN.

3.3 Experiments

In our experiments, we used two challenging, publicly available datasets: CelebA and LFWA. Since the CelebA dataset is so large, we did not need to augment it in any way. If we did not augment the LFWA dataset, the network would severely overfit to the training data due to the large number of parameters. We augmented the LFWA dataset by jittering the original images by increments of 10 pixels. After jittering, we had over 75,000 images for training.

3.3.1 Independent CNNs

We train independent CNNs for all the 40 attributes for both datasets in order to compare these results with those from MCNN and MCNN-AUX. We use one portion of our MCNN network for this. Each independent CNN has 3 convolutional layers, and 3 fully connected layers with the parameters specified in previous sections. We train these networks for 22 epochs for both datasets and use a batch size of 100. The independent CNNs each take about an hour to train for the CelebA dataset and about 30 minutes for the LFWA dataset. For all 40 attributes, training independent CNNs takes over 40 hours for CelebA and over 20 hours for LFWA.

3.3.2 MCNN

To train MCNN, we use batches of size 100, and train for 22 epochs for both datasets. Training takes about 2.5 hours for the CelebA dataset and about 1 hour for the LFWA dataset. We see a significant reduction in training time from 40 hours to 2.5 hours for CelebA and 20 hours to 1 hour for LFWA using MCNN over independent CNNs.

3.3.3 MCNN-AUX

Taking the trained MCNN, we fix the weights for that portion of the MCNN-AUX network and only train the AUX network. This takes about 20 minutes to train for CelebA and about 10 minutes for LFWA.

3.3.4 Results

We present results for our independent CNNs, MCNN, and MCNN-AUX. We compare with the state-of-the-art, Liu et al. [5], and a baseline of always choosing the most common label for each attribute.

We see from Table 3.2 that our independent CNNs outperform Liu et al. on most attributes for CelebA. The independent CNNs improve on Liu et al. by 15% for *necklace*, 12% for *blurry*, 9% for *straight hair*, and 8% for *big nose*. MCNN makes even further improvements, and finally MCNN-AUX gives the highest accuracy for most attributes.

We see that the largest increase in performance is from the method of Liu et al. to the independent CNNs, with smaller improvements being made with MCNN and MCNN-AUX. From this, we determine that the value in MCNN and MCNN-AUX is in the reduced training time and number of parameters, which reduces the chances of overfitting. We do not expect to see an increase in performance with MCNN-AUX for every attribute, as many attributes do not have strong relationships with others. Determining which relationships to use can be done using a set of validation data, however, in this work we chose not to remove any relationships in our testing. All three of our methods outperform the baseline for every attribute in CelebA.

Figure 3.3 shows a heatmap of the weights for the AUX network on the CelebA dataset. From Figure 3.3 we can see that each attribute contributes the most to its final classifier score. This is expected as MCNN already produces strong attribute classification accuracies. Some intuitive relationships can be seen in the heatmap.

Table 3.2: Results for CelebA. The highest accuracy for each attribute is in bold.

Attribute	**Baseline**	**Liu et al.**	**Independent**	**MCNN**	**MCNN-AUX**	**Attribute**	**Baseline**	**Liu et al.**	**Independent**	**MCNN**	**MCNN-AUX**
5 o'clock Shadow	90.01	91	93.94	94.41	**94.51**	Heavy Makeup	59.50	90	90.95	91.37	**91.55**
Arched Eyebrows	71.55	79	83.16	**83.55**	83.42	High Cheekbones	51.81	**88**	87.34	87.55	87.58
Attractive	50.41	81	82.22	82.94	**83.06**	Lipstick	52.18	93	93.80	93.95	**94.11**
Bags Under Eyes	79.73	79	84.83	84.89	**84.92**	Male	61.34	98	98.02	98.16	**98.17**
Bald	97.88	98	98.85	98.87	**98.90**	Mouth Slightly Open	50.49	92	**93.99**	93.74	93.74
Bangs	84.42	95	95.99	96.04	**96.05**	Mustache	96.13	95	96.67	**96.93**	96.88
Big Lips	67.29	68	70.80	71.20	**71.47**	Narrow Eyes	85.13	81	87.22	87.16	**87.23**
Big Nose	78.79	78	84.47	84.50	**84.53**	Necklace	86.20	71	86.41	**86.82**	86.63
Black Hair	72.83	88	89.41	**89.87**	89.78	Necktie	92.99	93	**96.71**	96.53	96.51
Blond Hair	86.67	95	95.88	95.97	**96.01**	No Beard	85.36	95	95.93	**96.11**	96.05
Blurry	94.94	84	96.07	96.08	**96.17**	Oval Face	70.43	66	74.70	75.81	**75.84**
Brown Hair	82.03	80	88.75	88.99	**89.15**	Pale Skin	95.79	91	**97.07**	97.01	97.05
Bushy Eyebrows	87.04	90	**92.87**	92.80	92.84	Pointy Nose	71.42	72	**77.47**	**77.47**	**77.47**
Chubby	94.69	91	95.55	95.66	**95.67**	Receding Hairline	91.51	89	93.41	**93.81**	**93.81**
Double Chin	95.42	92	**96.43**	96.41	96.32	Rosy Cheeks	92.82	90	95.02	95.13	**95.16**
Earrings	79.33	82	90.35	90.32	**90.43**	Sideburns	95.36	96	97.77	97.82	**97.85**
Eyeglasses	93.54	99	**99.67**	99.63	99.63	Smiling	50.03	92	92.65	92.66	**92.73**
Goatee	95.41	95	97.13	**97.30**	97.24	Straight Hair	79.01	73	82.62	83.39	**83.58**
Gray Hair	96.81	97	98.07	**98.20**	**98.20**	Wavy Hair	63.59	80	83.24	**83.92**	83.91
Hat	95.79	99	98.97	99.04	**99.05**	Young	75.71	87	87.98	88.30	**88.48**

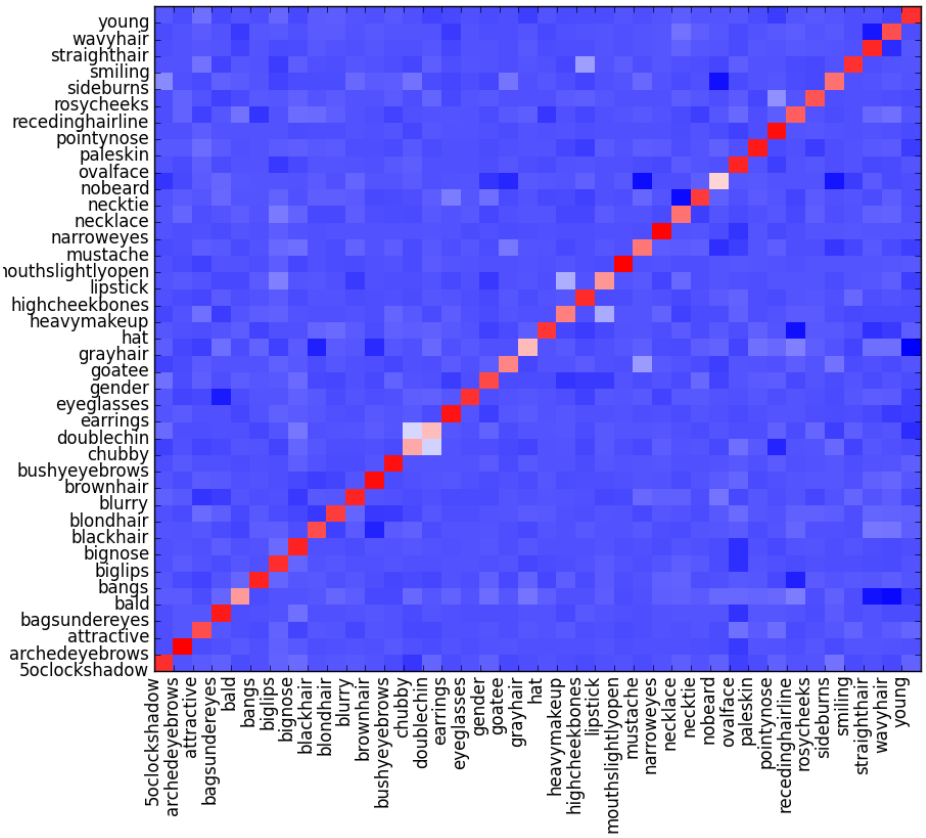


Figure 3.3: Heatmap of AUX network weights on CelebA. Along the x-axis, we have the MCNN output units and on the y-axis, the AUX units. Red indicates a strong relationship, and blue indicates a strong inverse relationship. Best viewed in color.

We see that *bald* is strongly related to *receding hairline* and has an inverse relationship with *straight hair* and *wavy hair* and that *no beard* has an inverse relationship with *5 o'clock shadow*, *mustache*, and *sideburns*. There are many more just like these. We do see some unexpected relationships as well, like *high cheekbones* and *smiling* having a strong connection. This would likely indicate that people are not very good at determining when someone has *high cheekbones* and therefore the labels for this attribute are somewhat noisy.

Table 3.3 shows the results for the LFWA dataset. We can see that the accuracies are lower for this dataset than for the CelebA dataset. This is likely due to overfitting because LFWA is much smaller than CelebA. The independent CNNs outperform Liu et al. on most attributes with an improvement of 11% for *blurry*, 11% for *rosy cheeks*, 10% improvement for *pale skin*, and 5% improvements for both *straight hair* and *wavy hair*. MCNN improved the classification accuracy of many attributes, but we see that a few, such as *blurry* and *eyeglasses*, did not improve with MCNN. For *blurry* and *eyeglasses* this makes sense, as both attributes are relatively unrelated to the other attributes, and therefore do not gain anything from shared information. We note that though MCNN-AUX does not improve the results for some attributes, we do not pre-train the networks using a larger dataset, as in Liu et al., which used a much larger dataset to initialize the weights of their networks. Pre-training on external data would likely improve the results, however that is not the focus of this work.

Figure 3.4 shows a heatmap of the weights for the AUX network on LFWA. There is much more white in this heatmap than in that of Figure 3.3 indicating

Table 3.3: Results for LFWA. The highest accuracy for each attribute is in bold.

Attribute	**Baseline**	**Liu et al.**	**Independent**	**MCNN**	**MCNN-AUX**	**Attribute**	**Baseline**	**Liu et al.**	**Independent**	**MCNN**	**MCNN-AUX**
5 o'clock Shadow	58.64	**84**	77.39	77.70	77.06	Heavy Makeup	89.20	95	95.63	95.84	**95.85**
Arched Eyebrows	74.88	82	81.4	**82.36**	81.78	High Cheekbones	67.74	88	88.02	88.25	**88.38**
Attractive	62.87	**83**	80.20	80.42	80.31	Lipstick	85.53	95	94.68	94.89	**95.04**
Bags Under Eyes	58.29	83	83.24	**83.51**	83.48	Male	78.77	94	93.27	93.66	**94.02**
Bald	89.37	88	91.51	**91.99**	91.94	Mouth Slightly Open	58.70	82	82.41	83.47	**83.51**
Bangs	83.59	88	**90.47**	89.99	90.08	Mustache	86.62	92	**93.69**	93.53	93.43
Big Lips	62.86	75	79.06	79.21	**79.24**	Narrow Eyes	65.50	81	82.48	82.73	**82.86**
Big Nose	68.59	81	84.43	84.76	**84.98**	Necklace	80.49	88	**89.98**	89.66	89.94
Black Hair	87.63	90	91.84	92.35	**92.63**	Necktie	64.09	79	80.34	80.50	**80.66**
Blond Hair	95.74	97	97.23	**97.45**	97.41	No Beard	70.05	79	81.45	82.13	**82.15**
Blurry	84.02	74	**86.71**	85.30	85.23	Oval Face	51.49	74	77.06.	77.38	**77.39**
Brown Hair	64.56	77	80.84	**80.94**	80.85	Pale Skin	52.09	84	94.31	**93.41**	93.32
Bushy Eyebrows	53.70	82	84.79	**85.11**	84.97	Pointy Nose	71.10	80	**84.41**	84.18	84.14
Chubby	63.92	73	75.85	**76.90**	76.86	Receding Hairline	59.84	85	86.00	**86.26**	86.25
Double Chin	62.44	78	**82.00**	81.17	81.52	Rosy Cheeks	79.65	78	**89.46**	87.52	87.92
Earrings	86.86	94	94.73	94.91	**94.95**	Sideburns	68.72	77	81.70	82.73	**83.13**
Eyeglasses	81.99	**95**	92.15	91.22	91.30	Smiling	60.50	91	**92.22**	91.75	91.83
Goatee	74.68	78	**83.34**	82.52	82.97	Straight Hair	64.44	76	**81.54**	78.72	78.53
Gray Hair	84.25	84	88.98	**89.04**	88.93	Wavy Hair	55.49	76	81.58	**81.96**	81.61
Hat	85.52	88	89.79	**90.20**	90.07	Young	79.60	**86**	85.11	85.37	85.84

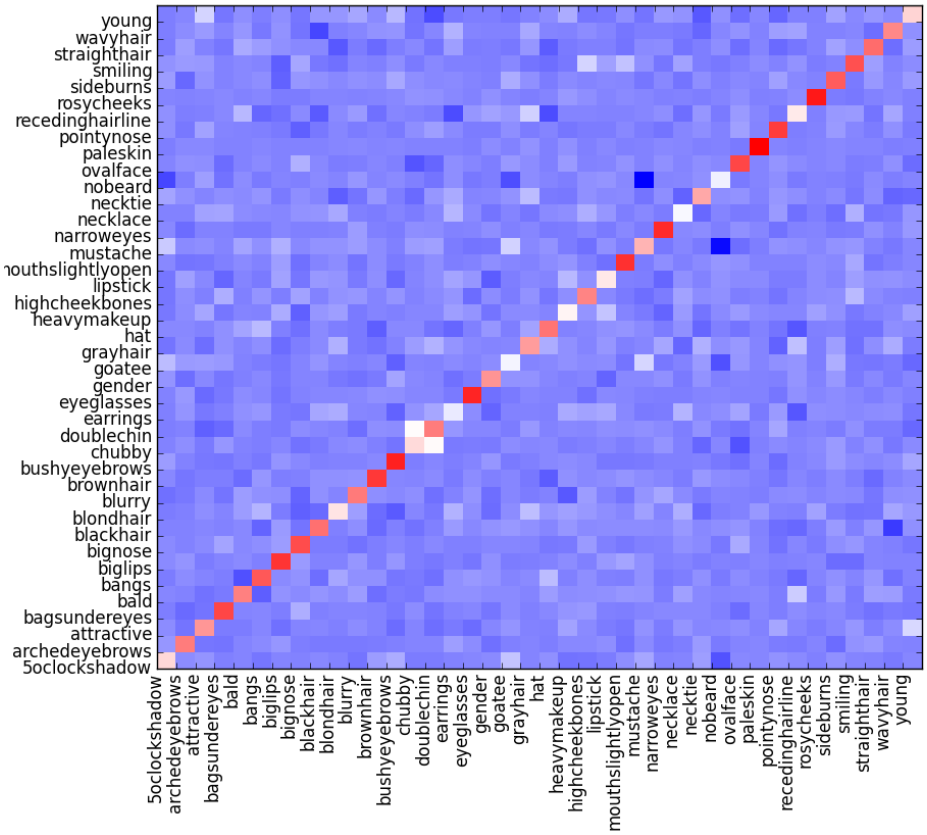


Figure 3.4: Heatmap of AUX network weights on LFWA. Along the x-axis, we have the MCNN output units and on the y-axis, the AUX units. Red indicates a strong relationship, and blue indicates a strong inverse relationship. Best viewed in color.

that there are fewer strong relationships in LFWA than in CelebA. This makes sense, as the classification accuracies for MCNN on LFWA were not as high as on CelebA. Again, we believe that this is due to the small size of the dataset. Though jittering LFWA helps, it does not compare to having a large amount of unique data as in CelebA. As with CelebA, we see that each attribute contributes most to its overall classification accuracy, though not quite as strongly. We again see promising relationships, with *bald* and *receding hairline* being strongly related as well as *heavy makeup* and *lipstick* and several others. We see that there are some noisy labels as in CelebA with *smiling* and *highcheekbones* being strongly related.

3.4 Summary

In this chapter, we detailed our MCNN-AUX network, showing that though facial attributes have been treated as independent problems in the past, there is a lot to be gained from shared information amongst attributes. Framing the attribute prediction problem as a multi-task learning problem is very natural and allows for a large reduction in training time and in the number of parameters required for the classifier. MCNN-AUX reduced the number of parameters from 64 million to fewer than 16 million, and reduced the training time by 16 times. We demonstrated the effectiveness of our independent CNN, MCNN, and MCNN-AUX classifiers on the challenging CelebA and LFWA datasets, achieving state-of-the-art performance for most attributes. Attribute relationships can be exploited in many ways and we presented three ways here: by sharing lower layers of MCNN, by grouping simi-

lar attributes in higher layers of MCNN, and by introducing an auxiliary network (AUX), which learns attribute relationships at the score level. Attribute relationships are learned implicitly at the lower levels, and explicitly in the higher grouped layers. Even without pre-training, we were able to outperform the method of [5] for many attributes. We demonstrated through experiments that a multi-task framework for attribute prediction outperforms independent classifiers. Taking advantage of implicit and explicit relationships among attributes allows for improved attribute prediction which will lead to improved facial recognition.

Chapter 4: Selective Learning

4.1 Overview

Facial attributes are intuitive descriptions of faces and have proven to be very useful in face recognition and verification. Despite their usefulness, to date there is only one large-scale facial attribute dataset, CelebA [2]. Impressive results have been achieved on this dataset, but it exhibits a variety of very significant biases. As CelebA contains mostly frontal idealized images of celebrities, it is difficult to generalize a model trained on this data for use on another dataset (of non celebrities). A typical approach to dealing with imbalanced data involves sampling the data in order to balance the positive and negative labels, however, with a multi-label problem this becomes a non-trivial task. By sampling to balance one label, we affect the distribution of other labels in the data. To address this problem, we introduce a novel selective learning method for deep networks which adaptively balances the data in each batch according to the desired distribution for each label. The bias in CelebA can be corrected for in this way, allowing the network to learn a more robust attribute model. We argue that without this multi-label balancing, the network cannot learn to accurately predict attributes that are poorly represented in CelebA. We demonstrate the effectiveness of our method on the problem of fa-

cial attribute prediction on CelebA, LFWA, and the new University of Maryland Attribute Evaluation Dataset (UMD-AED), outperforming the state-of-the-art on each dataset.

4.2 Proposed Approach

4.2.1 Multi-Task Attribute CNN

For attribute prediction, we use a multi-task deep attribute CNN (AttCNN) implemented in Caffe [124]. Table 4.1 shows the AttCNN architecture. There are three convolution layers (Conv1-3), each followed by a ReLU, max pooling and a local response normalization layer. The convolution layers are followed by three fully connected layers (FC1-3). FC1, and FC2 are both followed by a ReLU and a 50% dropout. FC3 is the output layer, with 40 nodes, one for each attribute. A sigmoid cross-entropy loss is used to facilitate training of the AttCNN. At test time, we apply the sigmoid function to FC3, taking values above 0.5 as positive instances of an attribute, and values below 0.5 as negative attribute responses.

4.2.2 Selective Learning

We introduce a novel selective learning method which adaptively balances each batch according to the desired distribution for each label in a multi-task learning framework. In other words, selective learning performs multi-label balancing of the training data. Consider, for example, the two attributes *bald*, and *male*. We intuitively know that the distribution for *male* is much more balanced than that for

Layer	**Parameters/Activation/Pooling/Norm**
Conv1	75 7x7 Filters, Stride 4 ReLU Max Pooling 3x3, Stride 2 Norm 5x5
Conv2	200 5x5 Filters ReLU Max Pooling 3x3, Stride 2 Norm 5x5
Conv3	300 3x3 Filters ReLU Max Pooling 5x5, Stride 2 Norm 5x5
FC1	512 Units ReLU Dropout 50%
FC2	512 Units ReLU Dropout 50%
FC3	40 Units

Table 4.1: AttCNN Architecture. Conv1 is the bottom layer, and FC3 is the top and final layer producing 40 outputs.

bald, and so we would expect to see more positive instances of *male* than *bald*. If we were to train a separate model for each attribute, we would be able to sample the data such that our model for *bald* could learn from a more balanced set. However, in a multi-task setting, where we learn all attributes at once, it is much more difficult to handle these imbalances. Selective learning offers a solution to this problem by adaptively balancing each label in every batch of data according to a target distribution for that label.

4.2.2.1 Batch Balancing

For each label (attribute) in each batch, if the distribution for that label does not match the desired target distribution, then we must adapt the batch accordingly. For each label, there are three cases: 1) the batch distribution is equal to the target distribution, 2) the label is over-represented, and 3) the label is under-represented. If the batch distribution for a label is equal to the target distribution, then we do nothing, and the selective learning batch (SL batch) is the same as the original batch for that label.

If a label is over-represented in a batch, that means there are more positive instances and fewer negative instances than if the batch followed the target distribution. When there are too many positive instances in the original batch, we take a random subset from the positive samples according to the target distribution and add those to the SL batch, ignoring the rest of the positive samples. For example, if we have a batch of size 100, with 70 positive instances, and a balanced target distri-

bution, then we sample 50 positive instances, ignoring the other 20. At this point, the SL batch contains a subset of the positive samples from the original batch.

We must now adjust the negative instances for the given label. Since the positive instances are over-represented, we were able to simply sample from the positive instances to meet the target distribution, but there are not enough negative instances to meet the target distribution. Instead, we weight the negative samples so they effectively match the target distribution. Using the same example from above, we have 30 negative samples in the original batch, so in the SL batch, we weight the negative samples by $\frac{5}{3}$ so that the negative samples effectively match the balanced target distribution. That is, the SL batch contains a subset of the positive samples, and all the negative samples, with an additional weight attached to them. If a label is under-represented, we reverse the above process, sampling from the negative instances and weighting the positive instances.

4.2.2.2 Implementation

Selective learning can be used with any loss function in a deep network. Here we describe the implementation details of the method.

For each label (attribute) a, we have some target distribution $P_T(a)$ and some batch distribution $P_B(a)$. If $P_T(a) = P_B(a)$, i.e. the batch distribution for a matches the target distribution, then the loss is calculated normally and the back-propagation error is unchanged. In practice, the SL batch is constructed by adding weights to every sample in the original batch.

Let $|B|$ be the size of the batch. If $P_B(a=1) > P_T(a=1)$, i.e. a is over-represented in the batch, the SL batch consists of all the samples from the original batch with weights to reduce the number of positive instances, and to increase the effective number of negative instances. Specifically, a random subset of $P_T(a=1)|B|$ positive instances are given a weight of 1, with all other positive instances given a weight of 0. The negative samples are each weighted by $\frac{P_T(a=0)}{P_B(a=0)}$ giving the negative samples the same effect as if they matched the target distribution.

Similarly, if $P_B(a=1) < P_T(a=1)$, i.e. a is under-represented in the batch, the SL batch consists of all the samples from the original batch with weights to reduce the number of negative instances, and to increase the effective number of positive instances. A random subset of $P_T(a=0)|B|$ negative instances are given a weight of 1, with all other negative instances given a weight of 0. The positive samples are each weighted by $\frac{P_T(a=1)}{P_B(a=1)}$ giving them the same effect as if they matched the target distribution.

Selective learning allows the network to learn from adapted batches for each label (or attribute), so that all labels – even under-represented and over-represented labels – are learned as if the data matched a desired target distribution. Selective learning is capable of both turning off back-propagation, and re-weighting the error for any attribute in any sample. Training of deep networks is done on a batch-by-batch basis, and so it makes sense to perform weighting and balancing at the batch-level. In MOON [8], each sample is re-weighted according to the target distribution, and so individual batch distributions are not taken into account, which leads to imbalances in training.

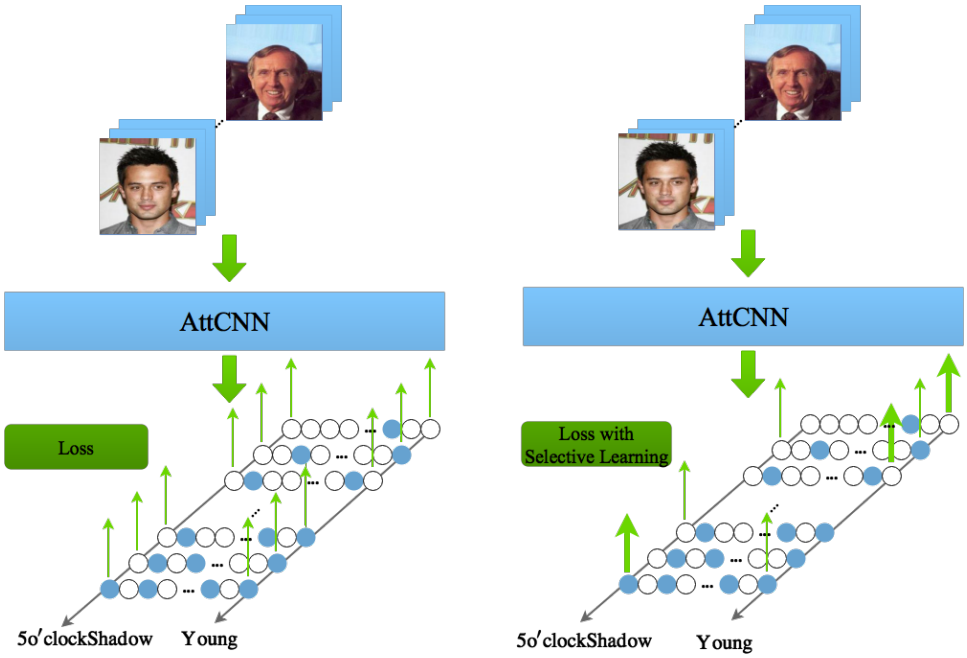


Figure 4.1: Visualization of the proposed selective learning (right) and normal learning without batch balancing (left). A blue node is a positive instance of an attribute and a white node is a negative instance of an attribute. The green upward pointing arrows indicate the back-propagation error. In a loss without selective learning (left), every attribute in every sample has the same weight, as indicated by the arrows all being the same thickness. In a loss with selective learning (right), we see that some attributes in some samples are not used for learning (they have no back-propagation arrows), and some samples have a higher weight (thicker green arrows) to account for imbalance. The two losses are demonstrated on *5o'clockShadow* and *Young*, two of the most imbalanced attributes in CelebA.

Figure 4.1 is a visualization of selective learning in comparison with normal multi-task learning. The left side shows a multi-task loss without selective learning, and the right side shows a multi-task loss with selective learning. Two attributes are highlighted: *5 o'clock shadow* and *young*. We can see that both are highly imbalanced, and with selective learning, each attribute is learned from its adapted batch, effectively removing the imbalance.

In our experiments, we apply selective learning to the proposed AttCNN, which uses a sigmoid cross-entropy loss. In the following section we demonstrate the effectiveness of selective learning on several challenging attribute datasets. We note that selective learning is extremely versatile and can be applied to any multi-label problem. It can easily be used for tasks other than facial attribute prediction, such as facial landmark detection (where nose points may be over-represented and ear points may be under-represented), body part localization (where some body parts may be occluded more than others), face verification across pose (where frontal is extremely over-represented) or any multi-task problem where the training data is imbalanced. Selective learning can also be used to combine data from several different sources, with some, or no common labels for use in training a deep network, since it adaptively balances every batch for each label.

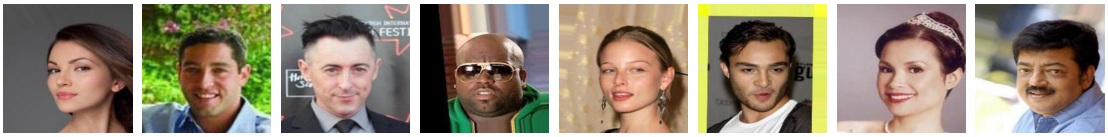


Figure 4.2: Sample images from CelebA

Figure 4.3: Sample images from LFWA

4.3 Experiments

4.3.1 Data

We use three datasets in our experiments: CelebA, LFWA, and UMD-AED - a new evaluation dataset. Sample images from CelebA can be seen in figure 4.2. Sample images from LFWA can be seen in figure 4.3.

For each attribute, the percentage of positive labels is plotted for both LFWA and the CelebA train split in figure 4.5. We can see that LFWA exhibits some of the same imbalances as CelebA, though not to the same extreme, likely due to the size of the dataset. For instance, *black hair*, *blond hair*, *heavy makeup*, and *high cheekbones* are even more under-represented in LFWA than in CelebA. So, if a model learned to prefer to output 0 for those attributes, then it would perform better on LFWA than on CelebA, without truly having learned a representation for those attributes.

Figure 4.4: Sample images from UMD-AED

4.3.1.1 University of Maryland Attribute Evaluation Dataset

In order to better evaluate an attribute model, we constructed a new evaluation dataset, UMD-AED. UMD-AED contains $2,800$ face images, each labeled with a subset of the 40 attributes from CelebA and LFWA. UMD-AED was collected in such a way that each attribute has the same number of positive and negative samples, hence why not every attribute is labeled in each image. Specifically, every attribute has 50 positive and 50 negative samples. Though UMD-AED is a small dataset, it is extremely effective at highlighting weakness in attribute models, as we will see in our experiments. With deep learning dominating almost every field in computer vision, most work is concerned with the quantity of data, rather than the quality. In our collection of UMD-AED, we focused on quality data which would effectively test the attribute representations learned by deep networks. By quality we mean that UMD-AED represents a wide variety of data, with low and high quality images, extreme lighting and poses, as well as different ages and skin tones, as can be seen in figure 4.4.

UMD-AED was constructed by performing an image search with each of the 40 attributes as search terms, running the face detector from [77], and hand-curating the resulting face images. UMD-AED is much more representative of real-world data

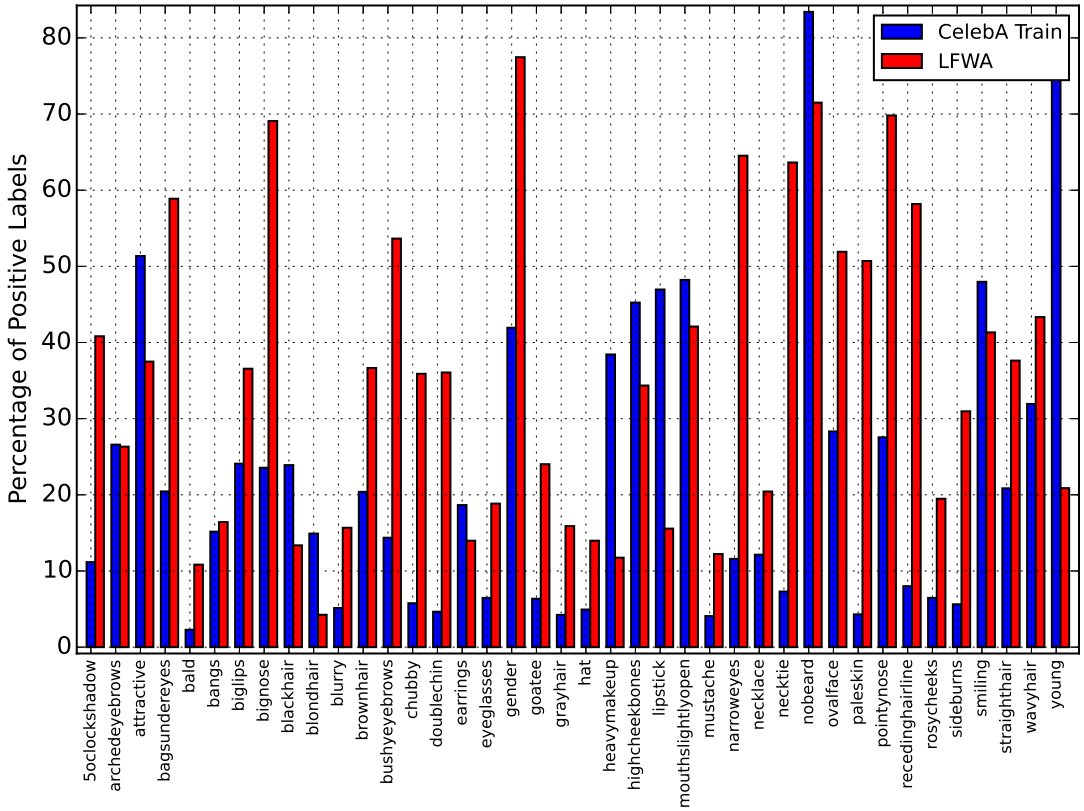


Figure 4.5: Percentage of positive attribute labels for CelebA train, and LFWA.

than CelebA or LFWA. As we will demonstrate in our experiments, to compare performance of attribute models on the test split of CelebA if they were trained on CelebA is optimistic. Evaluating models on UMD-AED will provide a much more unbiased metric for success of attribute prediction algorithms. If a model has learned a true representation for an attribute, then it can be expected to perform well on UMD-AED. We will make this dataset publicly available so that future work on attribute prediction can be evaluated on a balanced, real-world dataset.

Method	**Accuracy**
LNet+ANet [2]	87.30
Walk and Learn [72]	88.15
MOON [8]	90.94
AttCNN (Ours)	**90.97**

Table 4.2: Average attribute accuracy on the CelebA test set.

4.3.2 AttCNN

We train AttCNN directly on the CelebA training set, without any pre-training. As preprocessing steps, we subtract the training mean from each image, and take a random crop of 227x227 from the original image of size 256x256. The network weights are learned from scratch – starting with random initialization – using only the CelebA training set. AttCNN is trained for 22 epochs with batches of size 200, using a sigmoid cross-entropy loss.

We compare our AttCNN to the state of the art methods in table 4.2. AttCNN is comparable with the three previous state-of-the-art methods for attribute prediction: MOON [8], LNet+ANet [2], and Walk & Learn [72]. Table 4.2 shows that AttCNN outperforms all three methods on average. This is an impressive feat, as AttCNN has fewer than 6 million parameters, and is trained from scratch, whereas the most recent state-of-the-art, MOON, has 138 million parameters and is pre-trained on a large-scale object-recognition dataset, and both LNet+ANet and Walk & Learn are pre-trained on identification and verification data.

We argue that the success of AttCNN is due to training directly on attribute

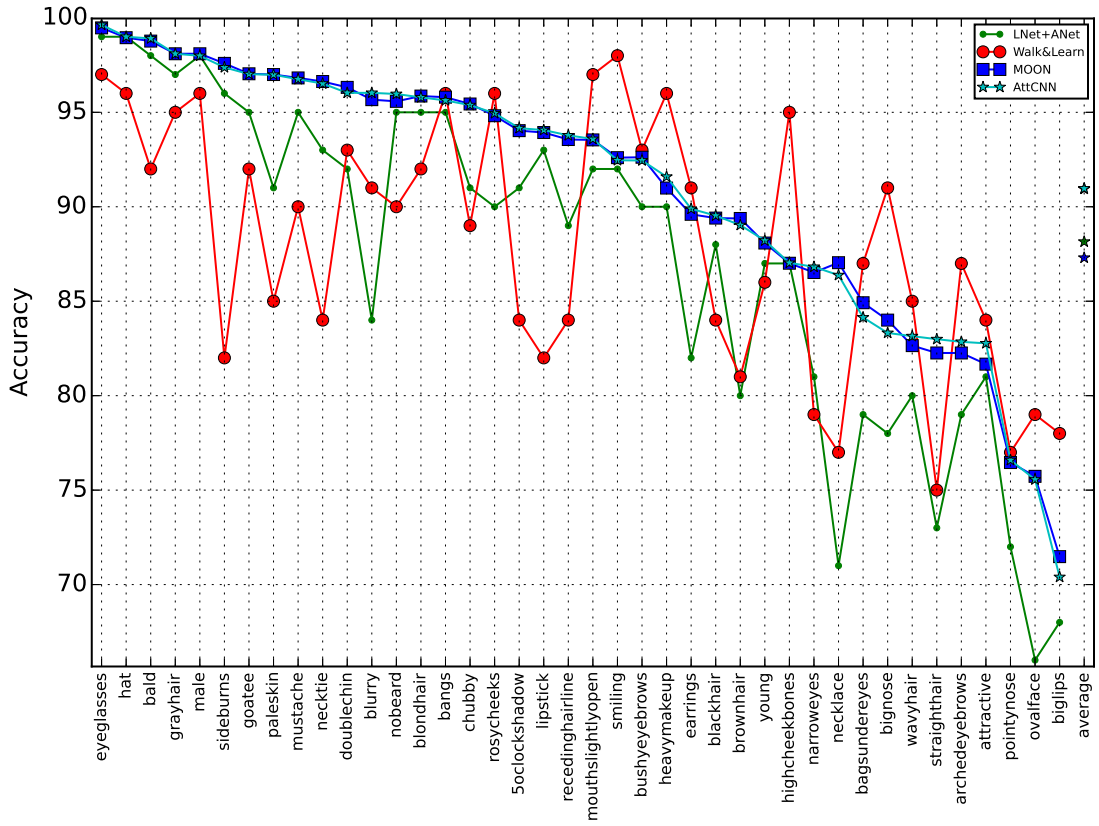


Figure 4.6: Results for AttCNN on CelebA test set along with the recent state-of-the-art methods. Best viewed in color.

data. All three of the previous state-of-the-art networks have too many parameters to train directly from the $160,000$ images in the train split of CelebA. With AttCNN as our base network, we demonstrate the effectiveness of the proposed selective learning approach in the following section.

4.3.3 Selective Learning

We test the proposed selective learning method on CelebA, LFWA, and UMD-AED, and then compare with the state-of-the-art MOON method.

Method	**Average Accuracy**
$\text{MOON}_{Balanced}$	**86.33**
$\text{AttCNN}_{Balanced}$	85.05

Table 4.3: Average attribute accuracy on the CelebA test set using the balanced networks.

For our first experiment, we train AttCNN using selective learning with a balanced target distribution. We denote this model as $\text{AttCNN}_{Balanced}$. We train $\text{AttCNN}_{Balanced}$ for 22 epochs and we use batches of size 200 just as with the original AttCNN. Table 4.3 shows that $\text{AttCNN}_{Balanced}$ performs comparably to, though not as well as the balanced MOON on the CelebA test set. However, we believe this to be an artifact of the extreme imbalance in CelebA, which is not being effectively removed by MOON, as we will demonstrate in our experiments on LFWA and UMD-AED.

We perform two experiments adapting training of AttCNN to the CelebA training distribution ($\text{AttCNN}_{P(a)=train}$) and to the CelebA test distribution ($\text{AttCNN}_{P(a)=test}$), and present the results in table 4.4. Using selective learning with P(a)=train, we improve on the state-of-the-art for the CelebA test set with 91.05% average attribute prediction accuracy. This improvement highlights the need for label balancing at the batch-level as even slight changes in distributions within batches results in decreased performance. With P(a)=test, we see a small improvement, but we normally do not have access to the distribution of the test set. We provide this result to highlight the fact that the bias in CelebA extends from the training set to the validation and test sets. If the bias was less severe in the CelebA test set, we

Method	**Average Accuracy**
AttCNN$_{P(a)=train}$	**91.05**
AttCNN$_{P(a)=test}$	91.07

Table 4.4: Average attribute accuracy on the CelebA test set using AttCNN with target distributions given by the CelebA train ($P(a) = train$) and test ($P(a) = test$) sets. AttCNN$_{P(a)=train}$ is bolded as it is the new state-of-the-art on CelebA.

Method	**Average Accuracy**
MOON$_{UnBalanced}$	68.98
MOON$_{Balanced}$	70.49
AttCNN	71.21
AttCNN$_{P(a)=train}$	71.49
AttCNN$_{Balanced}$	**73.03**

Table 4.5: Average attribute accuracy on LFWA using MOON and AttCNN.

would see a larger improvement when adjusting for the testing distribution.

We tested the balanced and unbalanced MOON, as well as AttCNN, AttCNN$_{P(a)=train}$, and AttCNN$_{Balanced}$ on LFWA. The results are reported in table 4.5. AttCNN$_{Balanced}$ outperforms MOON by over 2.5%. We see that even just adapting each batch to align with the distribution of the training data (AttCNN$_{P(a)=train}$), outperforms both the unbalanced and the balanced MOON. Figure 4.7 shows the prediction accuracy for each attribute on LFWA using the balanced MOON and AttCNN$_{Balanced}$. We can see that the two curves are very close, except for a few attributes: *hat*, *bald*, *gray hair*, *chubby*, *blurry*, and *pointy nose*. We can see from figure 4.5 that these attributes were much more under-represented in CelebA than in LFWA, and so the

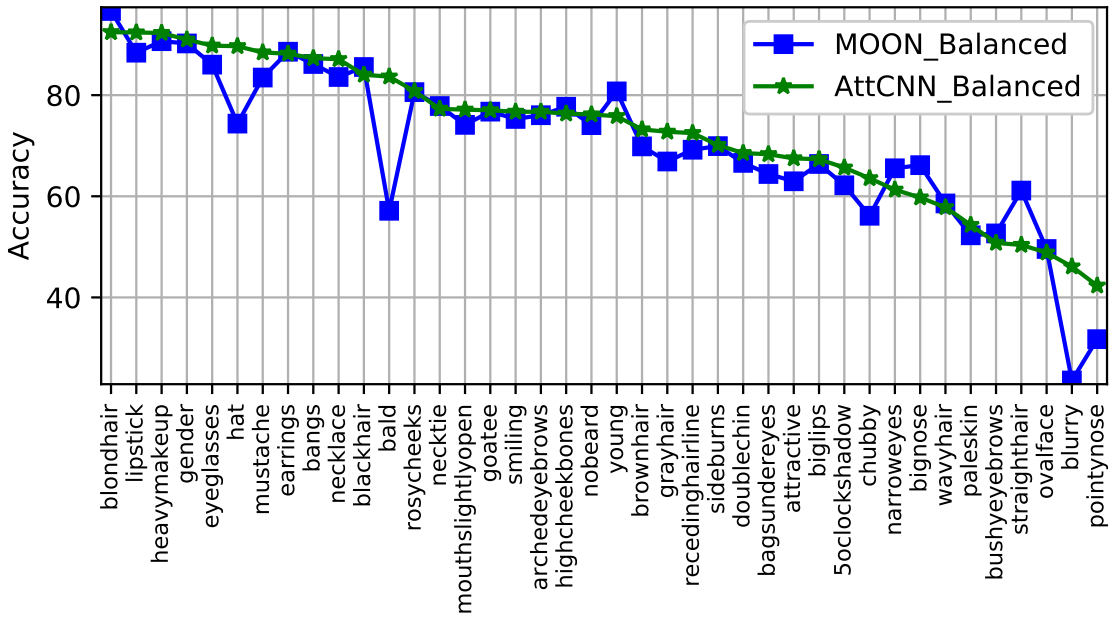


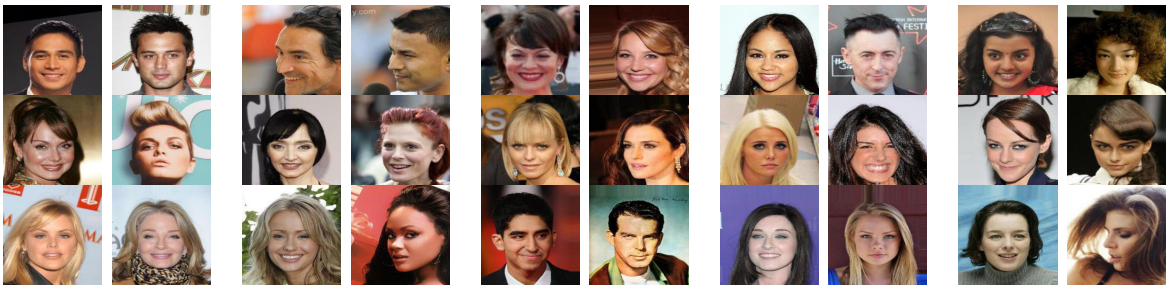
Figure 4.7: Results for AttCNN$_{Balanced}$ and MOON$_{Balanced}$ on LFWA. Best viewed in color.

bias of CelebA appears to have negatively affected the performance of MOON on LFWA. This same bias seems to have positively affected MOON on the CelebA test set, as seen in table 4.3.

For a less biased evaluation of the proposed attribute model, we test on UMD-AED, and these results are presented in table 4.6 as well as figure 4.9. We see that AttCNN$_{Balanced}$ outperforms MOON on almost every attribute. Here we truly see the effect of the extreme imbalance in CelebA on MOON, with many attributes achieving roughly 50% accuracy. In table 4.6 AttCNN$_{Balanced}$ outperforms the balanced MOON by a significant margin – over 11%, and AttCNN$_{Balanced}$ gives a 4% improvement over AttCNN. From this result, on a dataset with an even distribution for every attribute, and a better representation of real-world images, we can see that

Method	**Average Accuracy**
$\text{MOON}_{UnBalanced}$	56.36
$\text{MOON}_{Balanced}$	59.46
AttCNN	66.85
$\text{AttCNN}_{P(a)=train}$	67.40
$\text{AttCNN}_{P(a)=0.5}$	**71.11**

Table 4.6: Average attribute accuracy on UMD-AED using MOON and AttCNN.



(a) (b) (c) (d) (e)

Figure 4.8: Samples from CelebA train set with bad or ambiguous labeling for (a)*oval face*, (b)*attractive*, (c)*high cheekbones*, (d)*archedeyebrows*, and (e)*lipstick*. Positive labeled images are in the left columns, and negative labeled images are in the right columns. There is an obvious bias towards celebrities in this data. From (b), it is unclear what distinguishes an attractive person from an unattractive person.

selective learning addresses the problem of multi-label balancing for deep networks trained on imbalanced data.

Our evaluation of AttCNN$_{Balanced}$ on UMD-AED not only highlights the effectiveness of our method, but also indicates areas for improvement. Both MOON and AttCNN$_{Balanced}$ struggle with *oval face*, *attractive*, *high cheekbones*, *arched eyebrows*, and *lipstick*. All of these are very subjective attributes, with the exception of *lipstick*, and so there is likely some noise in the CelebA labels. Exploring the dataset, we find that this is exactly the case. We provide some sample images from CelebA in figure 4.8 to demonstrate the noisy labeling, showing samples with both positive and negative labels for the above attributes, with positive labels on the left and negative labels on the right. In many cases it is impossible to determine why one image has a positive label and another has a negative label. All of the subjects in figure 4.8(c) appear to have high cheekbones, but half of them are labeled as not having them. The negatively labeled samples in figure 4.8(d) have more arch in their eyebrows than the positively labeled samples. We argue that these subjective attributes should be removed from the attribute prediction task, as the goal is to accurately describe a face using attributes, and highly subjective attributes will not help with this cause.

Though *lipstick* is not a subjective attribute, we decided to perform the same analysis due to the poor performance of both MOON and AttCNN on this attribute. We found that the labels were just as noisy for *lipstick* as for the subjective attributes. Figure 4.8(e) shows samples from CelebA labeled with lipstick and not lipstick. None of the women in figure 4.8(e) are wearing lipstick, and yet half of them

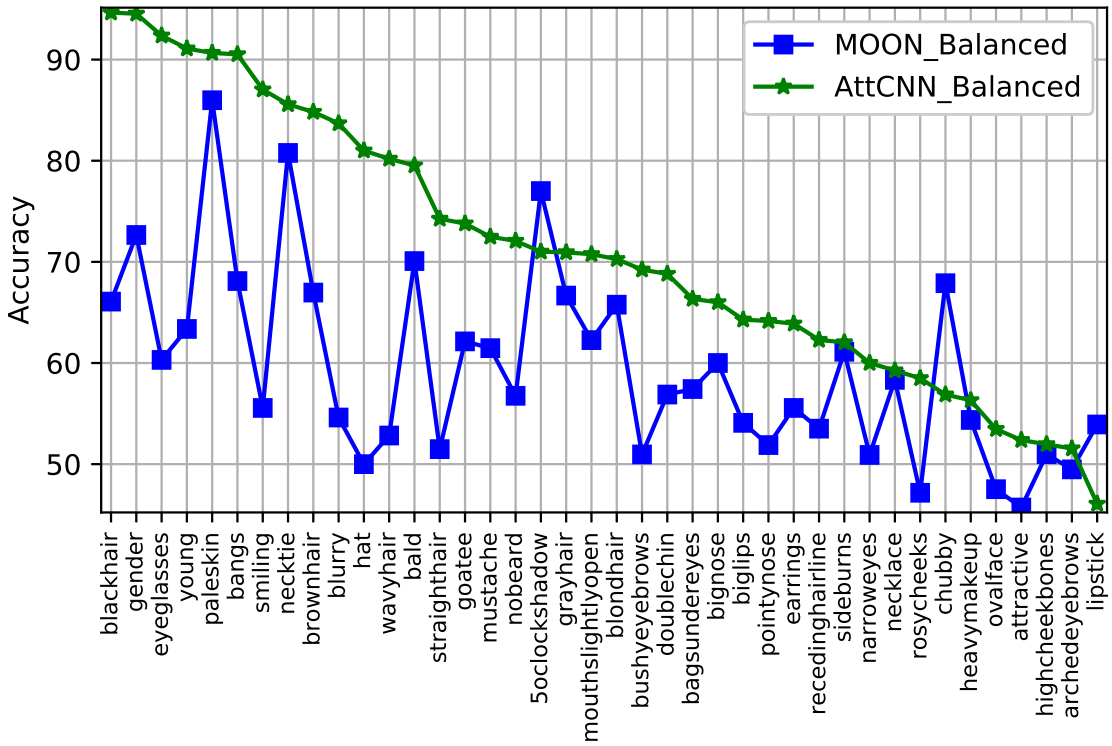


Figure 4.9: Results for AttCNN$_{Balanced}$ and MOON$_{Balanced}$ on UMD-AED. Best viewed in color.

are labeled as such. Even with multi-label balancing using selective learning, there is no way to correct for this much noise in the labels. It is clear from these analyses that the next step in attribute prediction research is to collect a new large-scale dataset with more precise labels for training.

4.4 Summary

In this chapter, we introduced a novel selective learning technique for multi-label balancing of biased training data, and demonstrated its effectiveness on the problem of facial attribute prediction, improving on the state-of-the-art. Selective learning adapts every training batch for each attribute according to a desired target distribution, allowing for balanced training with each batch. Since deep learning methods are trained on a batch-by-batch basis, it only makes sense to apply label balancing at the batch level. To test the capabilities of selective learning, we introduced a new evaluation dataset - UMD-AED. UMD-AED has an even distribution for each attribute, allowing for evaluation of attribute models in a balanced setting.

We introduced AttCNN, a deep network with fewer than 6 million parameters which is trained directly from CelebA. AttCNN outperformed the three previous state-of-the-art methods on CelebA, without pre-training on an external dataset. Training AttCNN with selective learning, we outperform the state-of-the-art on CelebA, LFWA, and UMD-AED, by 0.11%, 2.54%, and 11.65% respectively. The performance of our model on UMD-AED highlights the effectiveness of selective learning in allowing a deep network to learn a true representation of the data, rather

than just the bias of the training set. UMD-AED will be made publicly available so that future research on attribute prediction can be evaluated on a balanced dataset.

Selective learning can be applied to any multi-label problem which uses deep networks, including face verification across pose, facial landmark localization, and body part detection and localization, among many others. Though we demonstrate selective learning using a sigmoid cross-entropy loss, it can be used with any loss function. It can also be used to combine data from different sources with few or no common labels, since every batch is adapted for each label, no learning will occur for a particular label if it is not represented in the batch. Selective learning is an extremely versatile method that can be applied to many problems, and will help ease the difficulty associated with multi-label balancing in large-scale datasets, which are needed to train deep networks.

Chapter 5: Stabilizing Facial Attributes in Video

5.1 Overview

Recent research progress in facial attribute recognition has been dominated by small improvements on the only large-scale publicly available benchmark dataset, CelebA [2]. We propose to extend attribute prediction research to unconstrained videos. Applying attribute models trained on CelebA – a still image dataset – to video data highlights several major problems with current models, including the lack of consideration for both time and motion. Many facial attributes (e.g. gender, hair color) should be consistent throughout a video, however, current models do not produce consistent results. We introduce two methods to increase the consistency and accuracy of attribute responses in videos: a temporal coherence constraint, and a motion-attention mechanism. Both methods work on weakly labeled data, requiring attribute labels for only one frame in a sequence, which we call the anchor frame. The temporal coherence constraint moves the network responses of non-anchor frames toward the responses of anchor frames for each sequence, resulting in more stable and accurate attribute predictions. We use the motion between anchor and non-anchor video frames as an attention mechanism, discarding the information from parts of the non-anchor frame where no motion occurred. This

motion-attention focuses the network on the moving parts of the non-anchor frames (i.e. the face). Since there is no large-scale video dataset labeled with attributes, it is essential for attribute models to be able to learn from weakly labeled data. We demonstrate the effectiveness of the proposed methods by evaluating them on the challenging YouTube Faces video dataset [11]. The proposed motion-attention and temporal coherence methods outperform attribute models trained on CelebA, as well as those fine-tuned on video data. To the best of our knowledge, this work is the first to address the problem of facial attribute prediction in video.

5.2 Proposed Approach

5.2.1 Multi-Task Attribute CNN

We use a multi-task attribute CNN (MACNN) for feature learning and classification. All attributes are learned simultaneously in MACNN. MACNN's architecture is detailed in table 5.1. MACNN has a very small architecture, with only four convolution layers, and one fully connected layer added for classification. The network has fewer than 3 million parameters. AttCNN with Selective Learning, the state-of-the-art method for attribute prediction, has roughly 6 million parameters [1]. MACNN has fewer than half of the parameters of AttCNN. The proposed motion-attention and temporal coherence methods discussed in the following sections use MACNN as the base network.

Layer	**Parameters/Activation/Pooling/Norm**
Conv1	100 5x5 Filters ReLU Max Pooling 3x3 LRN 5x5
Conv2	200 3x3 Filters ReLU Max Pooling 3x3 LRN 5x5
Conv3	300 3x3 Filters ReLU Max Pooling 5x5 LRN 5x5
Conv4	300 5x5 Filters ReLU
FC1	40 Units

Table 5.1: MACNN Architecture. Conv1 is the bottom layer, and FC1 is the top and final layer producing 40 outputs.

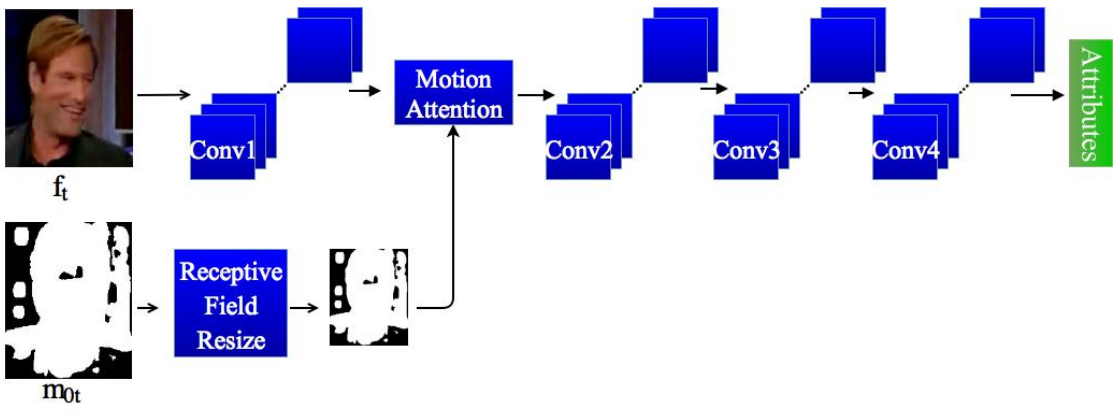


Figure 5.1: A visualization of the proposed motion-attention technique.

5.2.2 Motion-Attention

We introduce a novel attention technique based on motion in a video. Given two consecutive (or nearly consecutive) frames of a face video, we expect the attributes to remain the same, even if there is a small amount of motion. This is a reasonable assumption as most videos are captured at more than 20 frames per second and the face should not change much in $\frac{1}{20}$ of a second. In order to account for motion in a video, we introduce an attention mechanism based on motion between frames. This motion-attention mechanism is applied in a CNN, focusing the network on regions of motion, suppressing input from regions of a video frame where no motion occurred. The intuition here is that between nearly consecutive frames in a video taken from a stationary camera, the motion will occur only in the portion of the frames containing the face. Therefore, motion-attention will focus the network on the face, suppressing information from the background.

For each pair of consecutive frames (f_i, f_j), we have an associated binary op-

tical flow image m_{ij}, which is the result of thresholding the optical flow between f_i and f_j. That is, m_{ij} has a value of 1 where there is motion between f_i and f_j, and a 0 where there is no motion. Optical flow images can be combined for non-consecutive frames by taking the maximum value at each location for all flow images. For example, if we have three consecutive frames, f_i, f_j, and f_k, we can compute m_{ik} by taking the maximum of both m_{ij} and m_{jk} for every location in the image. That is, for all (x, y), $m_{ik}(x, y) = max(m_{ij}(x, y), m_{jk}(x, y))$. So, the binary flow image for non-consecutive frames has a 1 at every location where there is motion between any of the frames, and a 0 where there is no motion between any of the frames. These binary flow images are used as attention maps to focus the network on regions of motion.

For each video, we have one anchor frame f_0, and the rest are non-anchor frames f_t, $t \geq 1$. The anchor frame has attribute labels, and the non-anchor frames are unlabeled. For each pair of frames (f_0, f_t) we have the corresponding binary motion frame m_{0t}. The process of computing m_{0t} is described above. Each m_{0t} is used to turn off activations for parts of f_t where there is no motion, acting as an attention mechanism for the network. This encourages the network to focus on the parts of f_t which moved (i.e. the face), suppressing the portions of f_t which did not move (i.e. background). We refer to this method as a motion-attention mechanism. Figure 5.1 visualizes the proposed motion-attention mechanism, with two paths, one for the frame f_t, and one for the binary flow image m_{0t}.

Note that in figure 5.1, the motion-attention mechanism is being applied after the first convolution layer, but the motion-attention can be applied at any point in

the network. Since motion-attention can be applied at any point in the network, m_{0t} must first be resized to match the size of the feature maps where it will be applied. That is, if a layer L produces n_L feature maps of size w_L by h_L, then m_{0t} must be resized to w_L by h_L and then multiplied element-wise with all n_L feature maps, producing n_L focused feature maps. A receptive-field resizing is applied to m_{0t} so that each neuron in the resized m_{0t} has the same receptive field as the neuron in L where it is applied. That is if a neuron in the resized m_{0t} has a value of 1, then it means that there was motion between f_0 and f_t in the receptive field of that neuron, and similarly if it has a value of 0, then there was no motion between f_0 and f_t in the neuron's receptive field.

The motion-attention mechanism is applied during training. In the forward pass, m_{0t} is used to turn off the activation for neurons corresponding to regions without motion, and in the backward pass, no learning is performed for those neurons. We use both anchor and non-anchor frames as input to the network. For the anchor frames, f_0, we define m_{00} to be an image with all 1s, so no attention is used on the anchor frames. As noted earlier, the non-anchor frames f_t do not have attribute labels, and so the labels for the anchor frames are used as the labels for the non-anchor frames. The motion-attention works as a type of regularization deterring the network from over-fitting to the training data. By focusing the network on regions of motion in the non-anchor frame, motion-attention forces the network to explicitly account for motion and to learn how and what motion affects the labels.

Consider a video of a person talking. Let's say the anchor frame f_0 is labeled as a positive instance of *mouth slightly open*. We can assume that in f_1 the person's

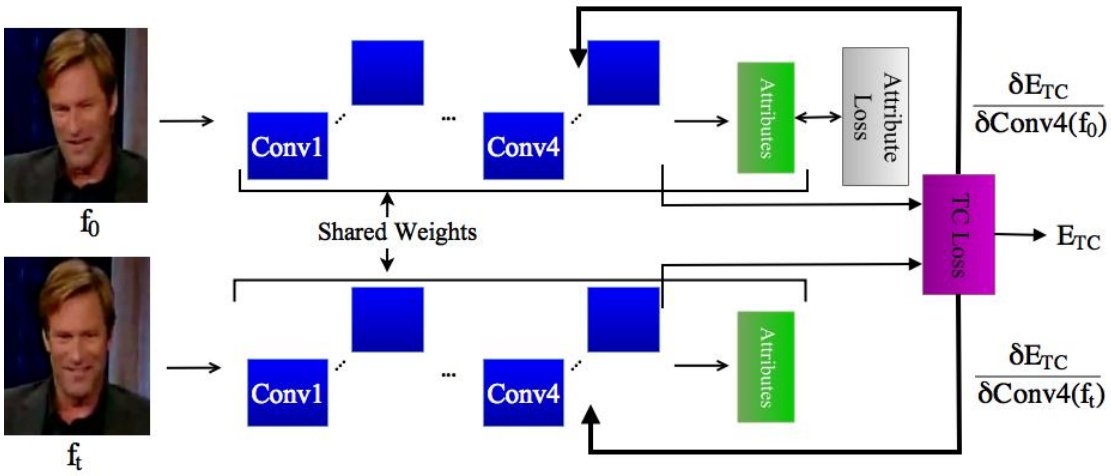


Figure 5.2: Visualization of two streams using the temporal coherence loss.

mouth will still be open, but perhaps a little more or less open, depending on what they are saying. In this instance, m_{01} may be 0 everywhere but around the mouth, where it is 1. So the motion-attention mechanism would focus the network on the mouth. Since the subject's mouth is likely still open, we use the label from f_0 for f_1, and now the network is only looking at the mouth, allowing the network to learn a more robust representation for *mouth slightly open*. The proposed motion-attention mechanism can be used alone, or with the proposed temporal coherence constraint described below.

5.2.3 Temporal Coherence

We introduce a temporal coherence (TC) constraint which works in a multi-stream network. A single-stream network is a normal network which takes an input image (or images), and outputs a label (or set of labels). A multi-stream network contains multiple copies of a single stream network, all sharing the same weights,

each with their own input and output, and the streams are connected in some way. The TC constraint is implemented as a loss that takes two layers as input, one from the first stream and the corresponding layer from another stream. The first stream of the network acts on an anchor frame of a video (f_0), and the other streams act on non-anchor frames some time t away from f_0 (f_t). Specifically, the second stream has f_1 as input, the third stream has f_2 as input and so on. The weights are shared among all the streams. The TC loss can be attached between any two streams at any point in the network, and is visualized in figure 5.2 with two streams.

Let F_i be the single stream CNN associated with input frame f_i. Since all streams in the network share the same weights, instead of referring to each stream individually, we will simplify notation by ignoring the stream, as the input indicates the stream. Let $F^l(f_0)$ and $F^l(f_t)$ be the l^{th} layer's activations for f_0 and f_t input respectively. The TC loss aims to move $F^l(f_t)$ toward $F^l(f_0)$, taking into account the distance t. That is, for a robust attribute model, we expect the activations for the anchor frame (f_0) to be similar to the activations for a non-anchor frame (f_t), assuming that not too much time has passed between f_0 and f_t. We also expect that the activations for frames that are closer together will be more similar than the activations for frames that are farther away.

Since f_0 is an anchor frame, we have labels for f_0, and so we apply a multi-label attribute loss on the first stream of the network (F_0), which takes f_0 as input. For F_t, the stream with the non-anchor frame f_t as input, we apply the TC loss in order to move the activations of F_t towards those of F_0. In other words, the error from the TC loss is only propagated through the F_t stream, not the F_0 stream, and

the multi-label loss error is only propagated through the F_0 stream since we only have labels for f_0.

More formally, the TC error for layer l is given in equation (5.1), where $\lambda(t)$ is some non-increasing positive function of t. $\lambda(t)$ is essentially the effect of the frame difference on the error, and therefore on learning. It makes sense for $\lambda(t)$ to not increase as t increases, as the farther two frames are from each other in a video sequence, the less likely they are to be similar, and so the effect of the error between their activations should be less. Equation (5.1) is used in the forward pass of the network using a TC loss, and equations (5.2) and (5.3) are used in the backward pass, propagating the error back through the network. Equation (5.2) indicates that the error only back-propagates for non-anchor frame inputs. A visualization of the proposed TC loss is shown in figure 5.2.

$$E_{TC} = \frac{\lambda(t)}{2}||F^l(f_t) - F^l(f_0)||_2^2 \tag{5.1}$$

$$\frac{\delta E_{TC}}{\delta F^l(f_0)} = 0 \tag{5.2}$$

$$\frac{\delta E_{TC}}{\delta F^l(f_t)} = \lambda(t)(F^l(f_t) - F^l(f_0)) \tag{5.3}$$

One may ask why we chose this formulation rather than assuming f_t has the same attribute labels as f_0. We illustrate this with an example: Let's say that we have two consecutive frames, f_0 and f_1. In f_0, the subject is frontal, there is no blur, and the subject does not have *arched eyebrows* (i.e. *arched eyebrows* is labeled

(a)arched eyebrows (b)bags under eyes (c)blurry (d)double chin (e)hat (f)mouth open (g)narrow eyes (h)smiling

Figure 5.3: Samples from YouTubeFaces where attributes change between frames. In (a), the top frame shows the man having *arched eyebrows*, but in the bottom frame he does not. (b) shows a frame where the woman has *bags under eyes* and then a frame where she does not. Similarly for (c)-(h)

as negative). In f_1, the subject moves, raising their eyebrows. The label for *arched eyebrows* in f_0 no longer applies. Our goal is to make the attribute model more robust using video data, and so we do not want the network to make a decision based on something it does not know. Therefore, we want to move the activations for f_1 towards not *arched eyebrows*, but not necessarily label f_1 as a negative instance of *arched eyebrows*. This gives us some intuition as to why we do not want to label f_1, or any f_t, with the same attributes as f_0. As we will see in our experiments, the proposed TC constraint improves attribute predictions over fine-tuning with f_t labeled with the attributes from f_0. The proposed temporal coherence constraint is able to utilize weakly labeled data to make a more robust attribute model without having access to labels for all frames.

In labeling four frames from every video in YouTubeFaces we found that there are eight attributes from the forty labeled in CelebA that can vary throughout a

video: *arched eyebrows*, *bags under eyes*, *blurry*, *double chin*, *hat*, *mouth slightly open*, *narroweyes*, and *smiling*. Figure 5.3 shows some examples of videos where these attributes change.

5.3 Experiments

5.3.1 YouTube Faces

We labeled four frames in YouTubeFaces [11] in every video with the 40 binary attributes from CelebA. The four frames correspond to the first frame, one from a third of the way through the video, one from two-thirds of the way through the video and the last frame: T_0, T_1, T_2, and T_3 respectively. For benchmarking purposes, there are 10 splits provided with the data for cross-validation testing. We use the anchor (attribute-labeled first frame) and non-anchor (no attribute labels) frames from the training portion of each split to fine-tune MACNN with our different methods. In all our experiments we test on the labeled frames from the test splits, and average over all 10 splits. We use the face boxes provided with the dataset, extracting each face from its original frame, and resizing it to 178×218, the same size as the aligned CelebA images. We do not perform alignment on the YouTube Faces frames, as attributes should be invariant to such mis-alignments.

5.3.2 MACNN

We use the aligned 178x218 CelebA images for training our base attribute model, MACNN. We implement and test MACNN using Caffe [124]. MACNN is

Method	**Average Accuracy**
LNet+ANet [2]	87.3%
Walk and Learn [72]	88.1%
MOON [8]	**90.90%**
AttCNN [1]	**91.05%**
MACNN (Ours)	**90.9%**

Table 5.2: Average attribute accuracy on the CelebA test set.

trained from scratch using only the aligned CelebA training images, without pre-training on an external dataset. A sigmoid cross-entropy loss is used to facilitate training with batches of size 100. As pre-processing steps, we subtract the training mean from all images and take random crops of 178x178 from each 178x218 input image. After 53 epochs, the error on the validation set no longer decreases, so training is stopped. We note that since MACNN only has 3 million parameters, it takes less than an hour to train on a single GPU.

We compare MACNN with state-of-the-art methods for attribute prediction in Table 5.2 to show that it is a good starting point for our temporal coherence and motion-attention work. From Table 5.2, we see that MACNN performs roughly as well as AttCNN – the current state-of-the-art – on average.

We use MACNN for our experiments rather than AttCNN because AttCNN has roughly 6 million parameters, while MACNN has only 3 million parameters, is very quick to train, and is less likely to over-fit. Testing MACNN on the anchor frames of YouTube Faces resulted in an average attribute accuracy of **83.80%**. The average attribute accuracy on YouTube Faces anchor frames is computed as

Model	$\mathbf{T}_0$	$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	**Average**
MACNN_0	86.55	86.57	86.44	86.23	86.44
MACNN_1	86.67	86.64	86.58	86.40	**86.57**
MACNN_2	86.55	86.61	86.56	86.20	86.48
MACNN_3	86.41	86.46	86.42	86.12	86.35
MACNN_{10}	85.68	85.75	85.58	85.53	85.64

Table 5.3: Average attribute accuracy on YouTube Faces labeled test (T_0, T_1, T_2, T_3) frames fine-tuning with anchor and non-anchor frames using the anchor labels.

follows. For each split, we compute the accuracy for each attribute, giving us 40 attribute accuracies for each split. Averaging the accuracies for all 40 attributes gives us a single average attribute accuracy for each split. We then average this average attribute accuracy for the 10 splits, giving us **83.80%** for MACNN. For all of the experiments below, we start with MACNN (trained on CelebA), and we fine-tune it on YouTube Faces using four different methods: fine-tuning with anchor and non-anchor frames, using anchor labels for both (5.3.3), using motion-attention (5.3.4), using temporal coherence (5.3.5), and using motion-attention and temporal coherence (5.3.6).

5.3.3 Fine-Tuning

Taking a model trained on still images, and fine-tuning it on labeled video data is one way to adjust the model to better handle video data. We do that here by fine-tuning MACNN on the anchor frames for each split in YouTube Faces using a sigmoid cross-entropy loss on the attributes. We call this fine-tuned network

MACNN_0, which we evaluate on the test portion for that split. We also fine-tune MACNN on non-anchor frames as well as the anchor frames. Non-anchor frames do not have labels, so we assume that the non-anchor frame has the same labels as its corresponding anchor frame. MACNN_1 is MACNN fine-tuned on f_0 and f_1 with both using the labels from f_0, MACNN_2 is MACNN fine-tuned on f_0, f_1, and f_2 all with the labels from f_0, and so on. Remember that f_0 is the anchor frame and f_1, f_2, etc. are non-anchor frames.

We evaluate our MACNN_i on the labeled test data for each split (T_0, T_1, T_2, T_3) providing the accuracies over all splits on each of the four frames as a measure of stability. Table 5.3 shows the average attribute accuracies, averaged over the 10 splits for MACNN_i for $i = 0, 1, 2, 3, 10$ on each of the four labeled frames per sequence. Fine-tuning on the anchor frames, MACNN_0, provides a 2.5% improvement over the original MACNN, which had an average accuracy of 83.80%. However, the improvements do not continue as more non-anchor frames are added to the training set for fine-tuning. In fact, MACNN_3 produces worse results than fine-tuning on the anchor frames alone. MACNN_3 is fine-tuning with f_0 and 3 non-anchor frames using the same labels as f_0. We expected the performance of fine-tuning using anchor labels on non-anchor frames would degrade as more non-anchor frames were added because as non-anchor frames move farther away from the anchor frames, it becomes less likely that they share the same label. We also note the sharp decline in performance between MACNN_3 and MACNN_{10}. We believe this to be due to two factors: one being that simply fine-tuning using anchor labels for non-anchor frames leads to overfitting of the network, and the other being that

Model	$\mathbf{T}_0$	$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	**Average**
MA_1	86.73	86.82	86.78	86.50	86.70
MA_2	86.91	86.97	86.94	85.65	86.86
MA_3	86.95	87.09	86.94	86.72	86.92
MA_{10}	87.15	87.17	87.11	86.84	**87.06**

Table 5.4: Average attribute accuracy on YouTube Faces labeled test (T_0, T_1, T_2, T_3) using MA_i.

the assumption that f_0 and f_t have the same, or similar, attribute responses breaks down as t gets larger. We see a similar, though less severe, phenomenon with TC_i and $MATC_i$ in the following sections.

5.3.4 Motion-Attention

For our motion-attention experiments, we similarly fine-tune MACNN on anchor and non-anchor frames, using the labels from anchor frames for both. The motion-attention is applied after the first pooling layer in MACNN. MA_1 is the motion-regularized model trained on f_0, and f_1 with corresponding motion-attention maps m_{00} and m_{01} respectfully. That is, m_{00} is an image of all 1s because there is no motion before the anchor frame, and m_{01} is a binary image capturing the motion between f_0 and f_1. MA_1 uses a sigmoid cross-entropy loss for both anchor and non-anchor frames using attribute labels from anchor frames (i.e. both f_0 and f_1 are labeled with the attributes from f_0). MA_2 is the motion-attention model trained on f_0, m_{00}, f_1, m_{01}, and f_2 and m_{02} (f_0, f_1, and f_2 are labeled with the attributes from f_0), and so on. MA_0 is equivalent to $MACNN_0$, since m_{00} – defined to be all

Model	$\mathbf{T}_0$	$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	**Average**
TC_1	86.52	86.63	86.60	86.40	86.53
TC_2	86.74	86.81	86.80	86.52	86.71
TC_3	86.77	86.85	86.88	86.54	**86.76**
TC_{10}	86.54	86.60	86.54	86.39	86.51

Table 5.5: Average attribute accuracy on YouTube Faces labeled test (T_0, T_1, T_2, T_3) using TC_i.

1s – provides no attention. The average attribute accuracies on anchor frames using MA_i for $i = 1, 2, 3, 4$ are reported in table 5.4. There is a consistent improvement when fine-tuning with motion-attention, and as non-anchor frames are added to the training set, the improvements continue, unlike regular fine-tuning. The motion-attention mechanism keeps the model from over-fitting since it cannot see the entire non-anchor frame, and so the motion-attention works as a kind of regularizer. We see that even MA_{10} shows improvements where $MACNN_{10}$ showed a sharp decline in performance.

5.3.5 Temporal Coherence

For the TC loss experiments, we define $\lambda(t) = e^{\frac{1}{t}-1}$ for $t \geq 1$, so the effect of each non-anchor frame on learning decreases with time. When fine-tuning MACNN using the TC loss, the loss is employed between the final convolution layers (conv4) of the two streams. A sigmoid cross-entropy loss is applied to the anchor frames – to learn the attributes in the anchor frames – in addition to the TC loss. Figure 5.2 visualizes the attribute loss on the anchor stream, and the TC loss between the two

conv4 layers. The figure only visualizes two streams, but there can be many streams depending on the number of non-anchor frames used in training. We call our models trained with the TC loss TC_1 if it fine-tunes on f_0 and f_1, TC_2 if it fine-tunes on f_0, f_1, and f_2, and so on. TC_0 is equivalent to MACNN_0, because without a non-anchor frame, there can be no TC loss, and so we end up with a single-stream network employing a sigmoid cross-entropy loss on the attribute labels of anchor frames. Table 5.5 shows the average attribute accuracy on the anchor frames over the 10 splits of YouTube Faces using the TC loss while fine-tuning. As seen with motion attention, there is a consistent improvement using the TC loss when fine-tuning, and as the number of non-anchor frames used for training increases, so does the average attribute accuracy. However, unlike MA_{10}, we do not see an improvement with TC_{10}. We believe this is due to the fact that the network is over-fitting a little, even with the TC loss. Though we do see a drop in performance with TC_{10}, it is not nearly as severe as the one we saw with MACNN_{10}, so the TC loss is helping the network to not overfit, and to account for the attributes which can change throughout a video. The TC loss on non-anchor frames improves over MACNN_i because it is less strict, and therefore allows pairs of anchor and non-anchor frames to have different attribute labels.

5.3.6 Motion-Attention with Temporal Coherence

We combine the proposed motion-attention mechanism and temporal coherence loss into one network by applying both methods on non-anchor frames. Com-

Model	$\mathbf{T}_0$	$\mathbf{T}_1$	$\mathbf{T}_2$	$\mathbf{T}_3$	**Average**
MATC_1	86.60	86.75	86.68	86.41	86.61
MATC_2	86.93	86.98	86.98	86.61	86.87
MATC_3	87.04	87.05	87.01	86.71	**86.96**
MATC_{10}	86.83	86.79	86.66	86.43	86.67

Table 5.6: Average attribute accuracy on YouTube Faces labeled test (T_0, T_1, T_2, T_3) using MATC_i.

bining both methods results in a multi-stream network where the anchor frames use a sigmoid cross-entropy loss, and the non-anchor frames each employ a TC loss and a motion-attention mechanism. We apply the motion-attention after the first pooling layer, and the TC loss at the conv4 layer. The models trained in this way are denoted MATC_1 (one non-anchor frame), MATC_2 (two non-anchor frames) and so on. Table 5.6 reports the average attribute accuracy for each of the MATC models on the labeled frames. Combining the two methods results in an improvement over using them individually, with MATC_1, MATC_2, and MATC_3 producing better results than both TC and MA individually. We do see that the dip in performance for TC_{10} carries over to MATC_{10}, reducing performance slightly. We again believe this to be due to the network slightly overfitting even with the TC loss.

5.4 Summary

In this chapter, we introduced two methods for explicitly incorporating video information into the training of attribute networks: a temporal coherence constraint and a motion-attention mechanism. Though some work has been done on adapting

attribute models trained on CelebA to better handle data from different distributions [8], we argue and show that time and motion must specifically be accounted for when training attribute models on video data. We demonstrated the effectiveness of our methods on the challenging YouTube Faces dataset, improving over the baseline of fine-tuning directly on the video data using only weakly labeled data. Our results show that when we do not account for time and motion in learning attribute models, as in MACNN_i, the model behavior is erratic. The proposed methods are able to use information provided by many non-anchor frames far away from the original anchor frames, which we cannot do when fine-tuning without these methods. We also note that there are only 3425 anchor frames, and so the fine-tuning is performed using roughly 3000 frames for each split, which is a very small amount of data for a CNN. As more labeled video data becomes available, the positive effects of the proposed temporal coherence and motion-attention methods will be even more obvious. Our results demonstrate the need to explicitly account for temporal and motion constraints when training attribute models on video data. The next step in learning robust attribute models for video data is to label a new video dataset with facial attributes, so that the effects of time and motion on attributes can be more thoroughly studied.

Chapter 6: Parsing Faces with Attributes

6.1 Overview

Many facial attributes – semantic features of faces – are related (e.g. *gender* and *mustache*, *attractive* and *heavy makeup*, etc.). However, we are able to recognize these attributes independently. Humans are capable of recognizing *gender* without superficial cues from hair length or makeup. Deep learning algorithms place a significant weight (we argue too much) on these relationships when learning to recognize facial attributes. In this chapter, we introduce face parsing with attributes to reduce the emphasis on such relationships in facial attribute recognition. Face parsing acts as an additional level of supervision in our attribute recognition model. The proposed method, AttParseNet, combines facial attribute recognition with face parsing, out-performing the state-of-the-art on three facial attribute benchmark datasets: CelebA, LFWA, and UMD-AED. To the best of our knowledge, this work is the first to address the problem of parsing faces according to semantic attributes.

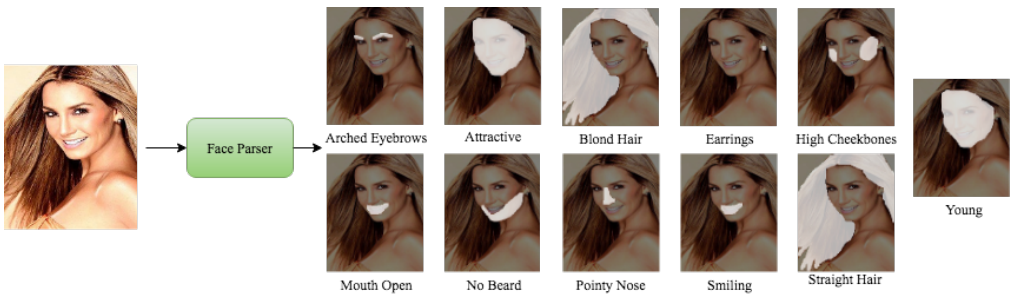


Figure 6.1: Example of the input and output of an attribute face parsing algorithm. The input is an image of a face, and the output is a set of maps, one for each attribute indicating the locations where that attribute is present in the image. The segments are displayed in different colors over the original image for readability. Only the positive attributes are shown.

6.2 Proposed Approach

The proposed approach consists of three main parts: generating attribute segments, parsing faces according to attributes, and attribute recognition. We detail these steps in the following sections.

6.2.1 Generating Segments

CelebA is a large-scale dataset labeled with semantic facial attributes. The dataset provides original images (ranging from face to full body) as well as cropped and aligned face images [2]. For this work, we use the original images. Five facial landmark points (left eye, right eye, nose, left mouth edge, and right mouth edge) are provided along with every cropped and aligned image, but none are provided for the original images. We use the Dlib facial landmark detector that gives boundaries

for the chin, mouth, eyes, nose, and eyebrows [125]. With these rough boundaries, we are able to create segment maps for each of the attributes in CelebA, using a rather simple rule-based approach, which we detail here. Figure 6.2 shows examples of how we generate segments given detected facial landmarks. In the figure, red points are those which come from the facial landmark detector and the light blue points are the ones we generate in order to create segments.

There are a total of 16 segments. Each of the attributes in CelebA comes from one or more of these segments. Six of the segments come directly from the facial landmark points, and the other 10 are generated as we describe below.

Table 6.1 details the segments and their corresponding attributes, and figure 6.3 shows an example of the segments for each attribute.

For the **face** segment, we consider the two upper-most facial landmarks from the left side of the face (let's call them p_l and p_{l-1} respectively). We want to continue the points upward and inward, as the forehead tends to be narrower than the cheekbones. If the slope of the line between p_{l-1} and p_l is negative, then the points at the top of the chin were continuing outward. Since we want them to move inward, we negate the slope and add this to p_l so that the next point (p_{l+1}) is moving inward. If the slope of the line between p_{l-1} and p_l is positive, we simply add it to p_l to get our p_{l+1}. We continue this process, adding the same slope to the most recent point until the next point will pass the top of the left eyebrow, at which point we stop adding points. We similarly repeat this process on the right side of the chin, finding the slope of the line between p_{r-1} and p_r. In this case, if the slope is negative, the points are already moving inward, and so we do not negate

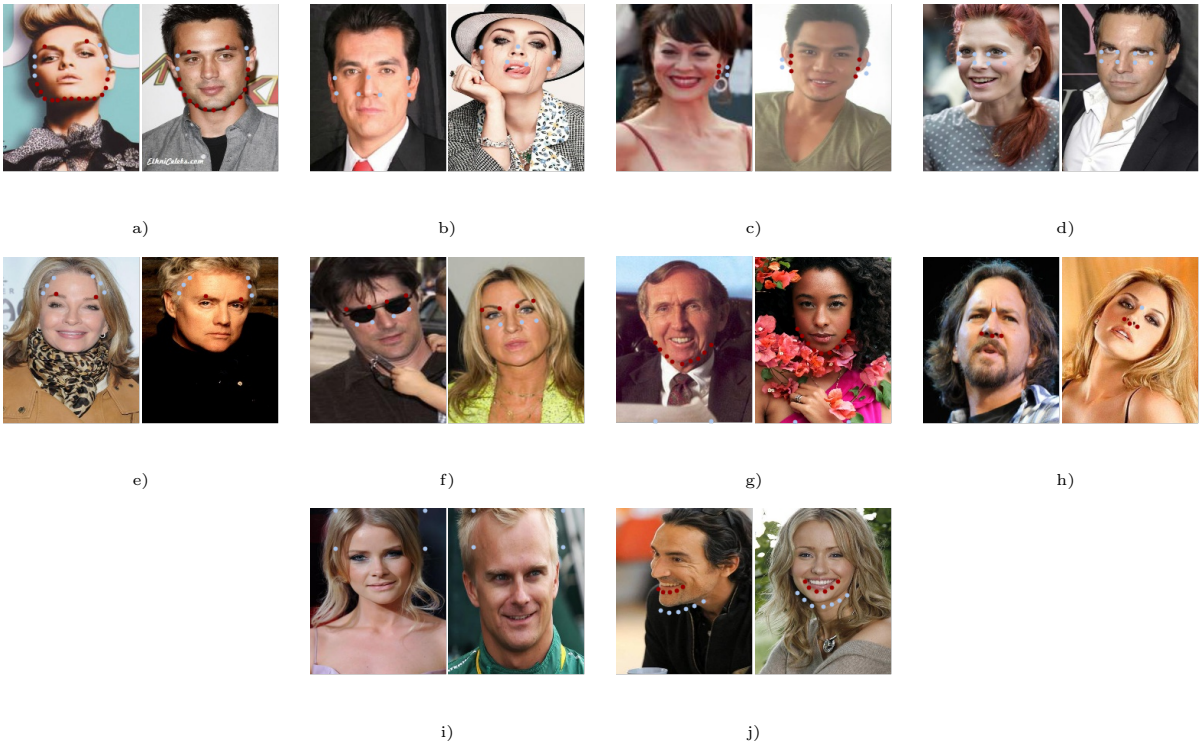


Figure 6.2: Examples of segment generation for the following segments: a)Face b)Cheek c)Earlobe d)Under Eye e)Forehead f)Glasses g)Neck h)Nose i)Top Head j)Under Chin. Red points are points provided by the facial landmark detector, and blue points are ones generated by our method as described in section 6.2.1.

the slope. If the slope is positive, we negate it before adding it to the most recent point. The **face** segment is contained by the chin points, these added points, and the tops of the eyebrows.

For the **cheek** segment, we must generate all boundaries for the cheeks because they are not provided as part of the facial landmarks. We do this by combining points from the eyes, nose and chin. The left cheek segment is generated with four points: top left, top right, bottom left, and bottom right. The top left point has the x-value from the top left chin landmark, and the y-value from the bottom of

the left eye. The top right point has the x-value from the nose bridge, which is calculated as the center of the two eyes, and the y-value from the bottom of the left eye. The bottom left point has the x-value from the third left chin point, and the y-value from the bottom of the nose. The bottom right point has the x-value from the bottom left part of the nose, and the same y-value as the bottom left point of the cheek. The points for the right cheek are similarly generated using the right side of the nose and chin and the right eye.

The **earlobe** segment is generated by the top three landmarks from the chin, and two points off of the chin. The three landmark points on the top of the chin are p_l, p_{l-1}, and p_{l-2}, for the left side of the chin, where p_l is the upper-most left point on the chin. The two points off of the chin are generated using the distances between the three chin points. The ear height is calculated as the distance between p_l and p_{l-2} and the ear width is calculated as $\frac{3}{4}$ of the ear height. The first point is located half way between p_l and p_{l-1} and a distance of ear width to the left. The second point is located half way between p_{l-1} and p_{l-2} and a distance of ear width to the left. The segment for the right earlobe is computed similarly.

The **under eye** segment is generated simply with four points: the left side of the eye with the y-coordinate of the bottom of the eye, the right side of the eye with the y-coordinate of the bottom of the eye, and both of these points with 2$*$eye height added to the y-coordinates.

The **forehead** segment is generated by adding points to the edge of the face in the same way as with the face segment. However, for the forehead segment, the addition of points only stops when the points pass the center of the eyebrow. The

Segment	Attributes
Face	attractive, chubby, male, oval face, pale skin, young
Cheek	high cheekbones, rosy cheeks
Earlobe	earrings
Under Eye	bags under eyes
Glasses	eyeglasses
Neck	necklace, necktie
Nose	big nose, pointy nose
Top Head	bald, black hair, blond hair, brown hair, gray hair, hat, receding hairline, straight hair, wavy hair
Under Chin	double chin
Chin	5 o'clock shadow, goatee, no beard, sideburns
Above Lip	5 o'clock shadow, goatee, mustache
Brow	arched eyebrows, bushy eyebrows
Mouth	big lips, heavy makeup, lipstick, mouth slightly open, smiling
Full	blurry
Eye	heavy makeup, narrow eyes
Forehead	bangs

Table 6.1: Segments and their corresponding attributes. Some attributes are covered by only one segment (e.g. earrings), while others require multiple segments (e.g. goatee).

points used to create the **forehead** segment are the tops of the eye brows as well as the points added past the face segment.

The **glasses** segment is generated using the left and right points of the eyebrows and the bottom points from the **under eye** segment.

The **neck** segment is one of the largest segments, as a necklace can be anywhere on the neck. This segment is generated from the chin points and two bottom points. The bottom points are generated from the x-coordinate of the left-most (or right-most for the right side) chin point and the y-coordinate is the bottom of the image.

The **nose** segment is computed very simply. Facial landmarks give the bottom left, bottom right and point of the nose, and we find the bridge of the nose by finding the midpoint between the eyes. If one of the eyes is not visible (the person is in a profile position), then we take the bridge to have the same y-value as the center of the visible eye, and the x-value of the bottom of the nose (left or right side depending on which is visible). The combination of the nose bridge and the facial landmark nose points gives the **nose** segment.

We have two different meanings for the **top head** segment. For the *hat* attribute, we use a fixed **top head** which contains the following points $(30, 0)$, $(147, 0)$, $(30, 50)$, $(147, 50)$ (we are using 178x218 images). For all other attributes, **top head** starts from the center of the **forehead** segment, and creates a semi-circle with a radius of half of the width of the forehead.

The **under chin** segment is generated by simply using the chin landmarks and then adding 10 to each of their y-values to get the points under the chin.

The remaining six segments are generated directly from the facial landmarks.

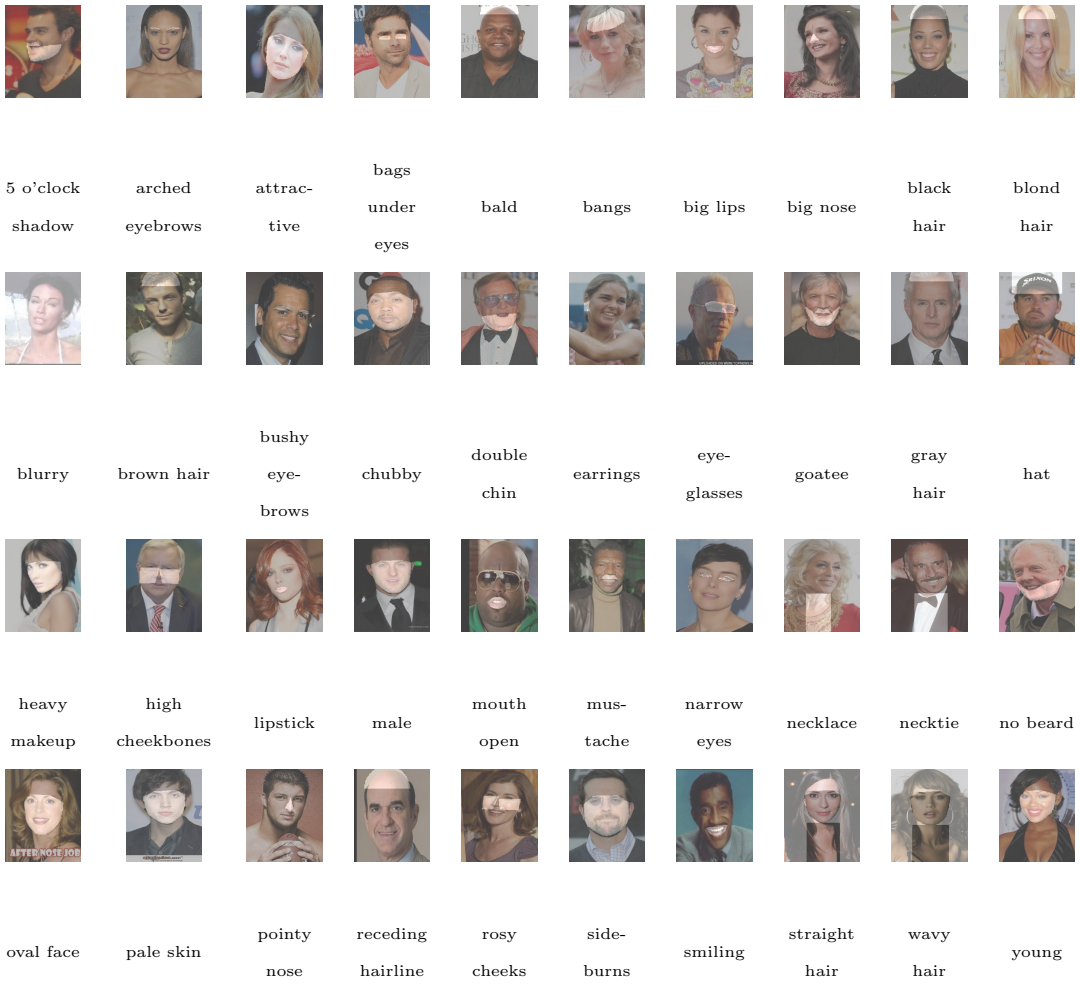


Figure 6.3: Examples of the segments generated by the rules detailed in section 6.2.1. The segments are binary images of the same size as the original face image, with 1 indicating that the attribute is present at that location, and 0 indicating that the attribute is not present at that location. For readability, the segments are presented over their corresponding face images, with white areas being positive segments.

Chin connects the chin points to the bottom of the mouth. **Above lip** connects the bottom of the nose to the top of the mouth. **Brow** uses all the eyebrow points. **Mouth** uses all the mouth points. **Full** simply uses four points from the edge of the image, and **eye** uses all eye points. Table 6.1 details which segments are used for each attribute.

As can be seen from figure 6.3, the hair attributes come from the whole image other than the **face** and the **neck** segments. In table 6.1 the hair attributes are listed in the **top head** row because there is no **other** segment.

From figure 6.3 we can see that these segments are weakly labeled. There are cases where the segments do not fully cover the area of interest (e.g. **forehead**, **top head**), segments that cover the area of interest nicely (e.g. **brow**, **mouth**), and segments that cover more areas than needed (e.g. **full**, **neck**). The goal of this work is to take advantage of these weakly labeled segments to learn a more robust attribute model. We show through our experiments that the added level of supervision from the attribute segments allows the network to learn a better attribute model, out-performing the state-of-the-art methods.

6.2.2 Parsing Faces with Attributes

Once the weakly labeled segments have been generated, the next step is to build a model which can parse faces according to their attributes. We aim to parse faces into their attribute components, much like in a semantic segmentation task. In semantic segmentation, the goal is to assign a class label to every pixel in an

image. Face parsing – a specific type of semantic segmentation – has in the past been limited to parsing the face into a handful of categories (e.g. hair, eyes, mouth, nose, etc.), none of which overlap. Instead, we want to assign a set of attribute labels to every pixel. The difference between this and semantic segmentation is that every pixel can have multiple labels (e.g. *rosy cheeks* and *high cheekbones*).

We use a deep convolutional neural network (CNN) as our segmentation model, which we call AttParseNet. The segmentation CNN has six convolution layers, with a ReLU after each convolution layer except for the last, and a pooling layer between the first and second layers. AttParseNet is shown in figure 6.4, and is an adaptation of the network from [1]. As our segmentation loss, we implement a version of selective learning, as described in [1], in a euclidean loss layer. Selective learning essentially balances each label in each batch by only learning from certain samples in each batch for each label. We use selective learning in two ways: as the authors of [1] do – to balance the positive and negative labels for each attribute in each batch – and also to balance the number of positive and negative pixels in each segmentation map, so that learning is not dominated by the background class.

We apply selective learning assuming a balanced target distribution for each attribute, and a balanced pixel distribution for positive and negative pixels in each segment map. For each face image, there are 40 segment maps, one for each attribute. For each attribute a, if a is present in the image, then its corresponding map has a value of 1 at each location in the attribute segment (see figure 6.3), otherwise, the map contains all 0 values. In each batch, the selective learning euclidean loss learns from an equal number of positive samples as negative samples for each

attribute, and for each sample, each positive pixel is weighted according to the total number of positive pixels in the map, and similarly for the negative pixels. This allows for more balanced learning, not letting attributes with large segments to be learned more quickly than attributes with small segments. This is a big problem, as there are some segments which are very small (e.g. *arched eyebrows*, *bags under eyes*), and some segments which are very large (e.g. *blurry*, *straight hair*). In addition to this, if an attribute is not present in an image then every pixel in the map for that attribute has a value of 0, further biasing learning toward the background (or negative) class. Selective learning at the pixel level helps to alleviate this problem. The euclidean selective learning loss is applied on the Conv6 layer. The Conv6 layer results in 40 26x21 feature maps, one for each attribute, as is demonstrated in the top portion of figure 6.4. Since the original image size is 178x218, we must apply max pooling on the segment maps so they match the final feature map output from Conv6.

The face parsing with attributes described above is part of the attribute recognition network, adding an additional level of supervision to training. We describe the attribute recognition portion of the network below.

6.2.3 Attribute Recognition

We use face parsing with attributes as part of our attribute recognition framework. As we see from figure 6.4, attribute classification and segmentation occur in the same network, AttParseNet. They both have their own losses, and contribute

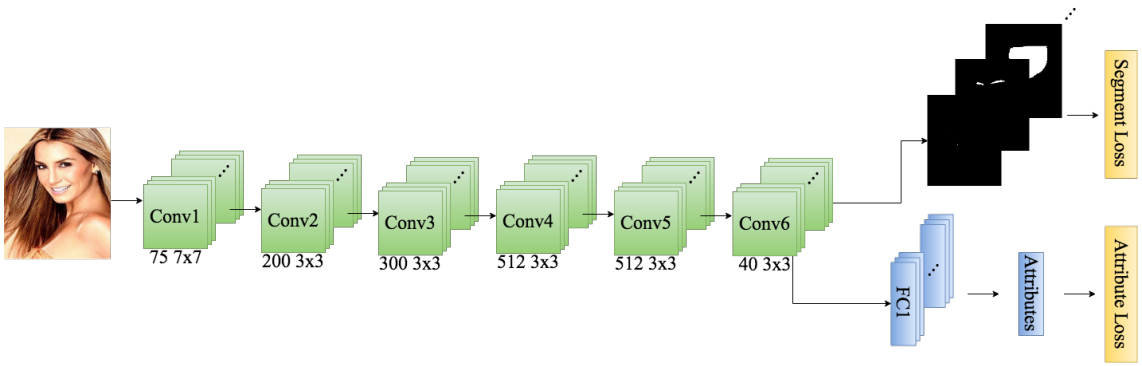


Figure 6.4: Segmentation Network

to learning of the network weights. Each of the feature maps resulting from Conv6 corresponds to an attribute segmentation, of which there are 40. Each of those maps is then connected to its own fully connected layer (with a single neuron) which classifies the map as a positive or negative instance of a particular attribute. The figure indicates that there is a segmentation loss, and an attribute loss. The segmentation loss was described in the previous section. The attribute loss is the same sigmoid cross-entropy loss with selective learning from [1] assuming a balanced target distribution for all attributes.

The face parsing with attributes is an added level of supervision in the attribute recognition framework, which leads to a more robust attribute model, as we show in our experiments.

6.3 Experiments

To demonstrate the effectiveness of the proposed methods, we perform extensive experiments on three benchmark datasets: CelebA, LFWA, and UMD-AED.

For all of our experiments, the models are trained on the training split of CelebA, and tested on the testing split of CelebA, on the full LFWA, and on UMD-AED. We compare the proposed AttParseNet method with the state-of-the-art AttCNN for facial attribute recognition: [1].

6.3.1 AttParseNet

We train AttParseNet directly on the unaligned cropped faces from the CelebA training set, resized to 178x218, without any pre-training. We implement and test our network in Caffe [124]. As we compare with the method from [1], we follow the same learning procedure: training our networks for 22 epochs. AttParseNet uses the weakly labeled segments for training only. At test time, segments are not used.

For a fair comparison, we train the method of [1] using the unaligned CelebA images as well. In order to encourage future research in this direction, we will make the attribute segments for CelebA publicly available.

6.3.2 Results

Figure 6.5 shows the average accuracy for each attribute on the testing split of CelebA for both the previous state-of-the-art, AttCNN [1] and the proposed AttParseNet. As we can see, the proposed method outperforms the state-of-the-art on all but three attributes: *lipstick*, *attractive*, and *young*. *Lipstick* is poorly labeled in CelebA, as has been previously pointed out in [1], and the difference between AttParseNet and AttCNN on this attribute is rather minor. For *attractive*

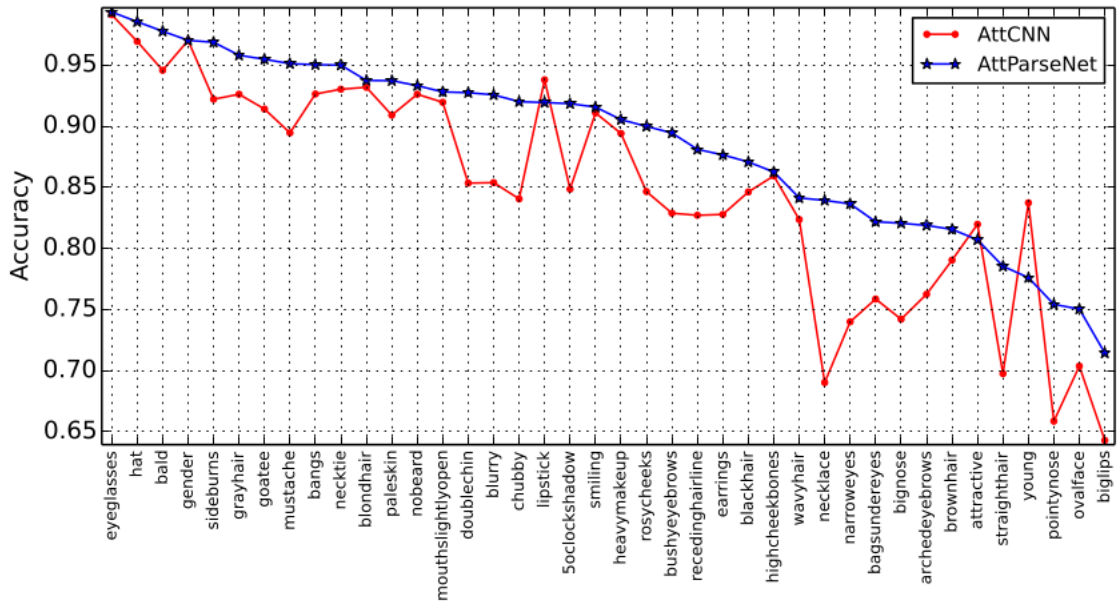


Figure 6.5: Average attribute accuracies for the test split of CelebA using AttCNN [1] and the proposed AttParseNet.

and *young* we believe the difference between AttCNN and AttParseNet to be due to the ambiguity of those attributes. The segments for *attractive* and *young* are the same, as we showed in figure 6.3. For this work, we made the assumption that *attractive* and *young* come only from the face, and nothing else. However, where exactly *attractive* and *young* manifest themselves in a face image is unclear. Further research is needed to determine where exactly certain attributes come from. We leave this for future work.

Despite the ambiguity in the segments for certain attributes, AttParseNet outperforms AttCNN by over 5% on 16 attributes. The average accuracies of AttCNN and AttParseNet on CelebA are presented in table 6.2. Adding segment supervision to the attribute recognition framework produces an almost 4% average improve-

Method	**MOON [8]**	**AttCNN [1]**	**AttParseNet (Ours)**
Accuracy	86.33%	84.80%	**88.73**%

Table 6.2: Average accuracy over all attributes on the testing split of CelebA. MOON was trained and tested on the aligned CelebA images. AttCNN and AttParseNet are trained and tested on the unaligned CelebA images.

ment on CelebA, with over half of the attributes having over 90% accuracy. This indicates that prior methods for attribute recognition were taking advantage of unwanted relationships between attributes in CelebA, and were therefore not learning a true representation for some attributes. Some examples of such attributes include: *necklace*, *straight hair*, *double chin*, and *chubby*. With the supervision from the weakly labeled segments we generate, we see significant improvements in prediction accuracy for these attributes.

We note that these results are based on AttCNN and AttParseNet trained on the unaligned CelebA training images. AttCNN trained and tested on the aligned CelebA images achieves an average accuracy of 85.05%. As the segments we generate are based on the unaligned CelebA images, we do not provide results for AttParseNet trained on the aligned CelebA images. We also note that these are results on CelebA assuming that training accounted for the imbalance in the labels for CelebA [8] [1]. When training with the unbalanced CelebA, [1] achieves an average attribute accuracy of 91.05% on the testing portion of CelebA.

We also compare AttCNN with AttParseNet on the University of Maryland Attribute Evaluation Dataset (UMD-AED). UMD-AED consists of roughly 3000 im-

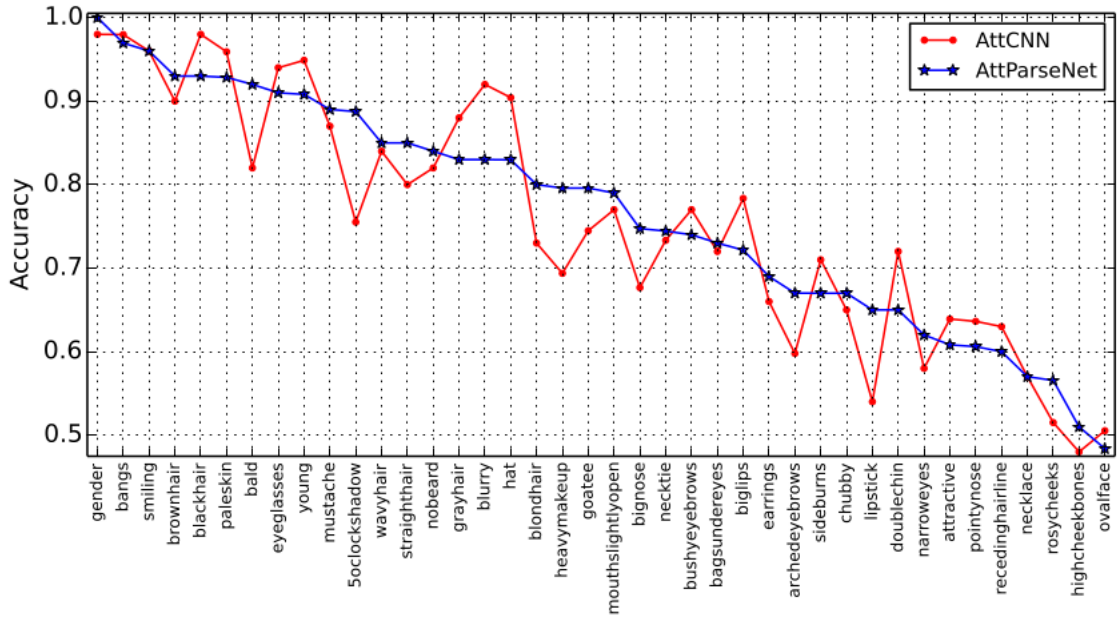


Figure 6.6: Average attribute accuracies for UMD-AED using AttCNN [1] and the proposed AttParseNet.

ages sparsely labeled with attributes. Each attribute has roughly 50 positive and 50 negative samples. This dataset was introduced in [1] as an evaluation dataset for attribute recognition methods trained on CelebA. We train AttCNN and AttParseNet on the unaligned images from the training split of CelebA and present the results on UMD-AED in figure 6.6.

As we can see, the two methods perform comparably on UMD-AED. AttCNN outperforms AttParseNet most significantly on *blurry* and *hat*. We noted in section 6.2.1 that some segments are more roughly labeled than others. The segments for both *blurry* and *hat* are very roughly labeled. The segment for *blurry* is the full image, when in reality, the face may be blurry, or the background, or both. The segment for *hat* contains the forehead as well as a constant rectangular block from

Method	**AttCNN [1]**	**AttParseNet (Ours)**
Accuracy	75.78%	**76.73**%

Table 6.3: Average accuracy over all attributes on UMD-AED. Both AttCNN and AttParseNet are trained on the unaligned CelebA images.

the top of the image, which may or may not contain the hat.

The proposed AttParseNet significantly outperforms AttCNN on *bald*, *5 o'clock shadow*, *heavy makeup*, *arched eyebrows*, and *lipstick*. The segments for all of these attributes are rather tight, even though they are weakly labeled. This indicates that for the attributes with accurate weakly labeled segments, the segment supervision improves attribute prediction in general (both on CelebA and UMD-AED). The proposed AttParseNet also achieves 100% accuracy on *gender*.

The average attribute accuracies for both AttCNN and AttParseNet on UMD-AED are presented in table 6.3. Adding weak segment supervision to the attribute recognition framework improves the average accuracy by almost 1%. We note that in [1], AttCNN is trained on the aligned CelebA images and achieves an average attribute accuracy of 71.11% on UMD-AED. An over 4% improvement is achieved by simply training on the unaligned CelebA images. Even further improvements are made by adding weak segment supervision with AttParseNet.

We also compare AttCNN and AttParseNet on LFWA [2]. Figure 6.7 shows the average prediction accuracy for each attribute on LFWA. We see that AttParseNet significantly outperforms AttCNN on *hat*, *black hair*, *gray hair*, *chubby*, and *straight hair*. The average accuracy over all attributes is presented in table 6.4. AttParseNet

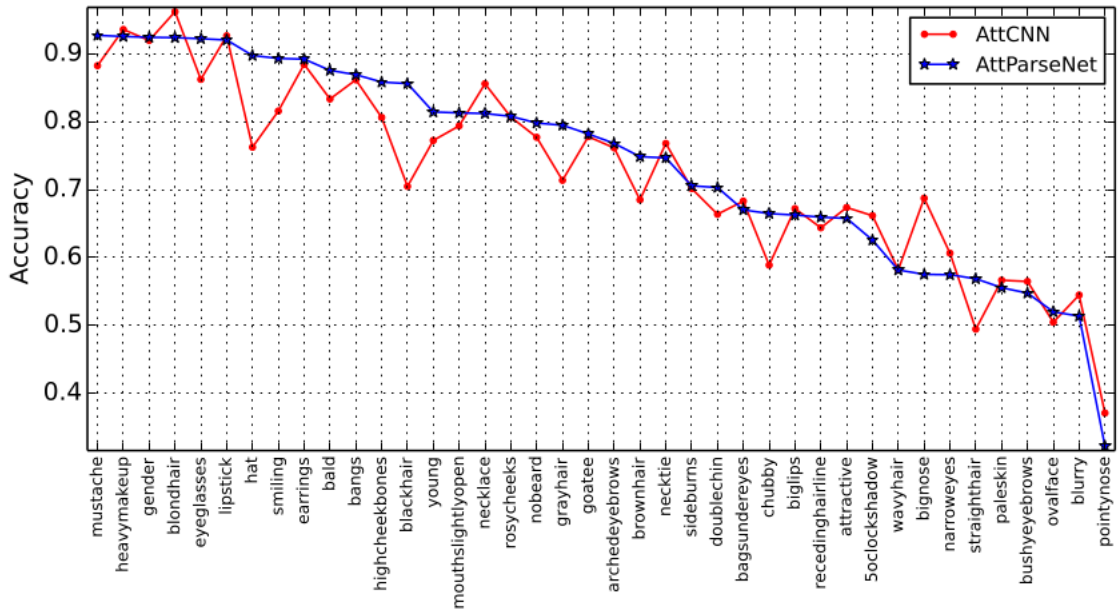


Figure 6.7: Average attribute accuracies for LFWA using AttCNN [1] and the proposed AttParseNet.

Method	**AttCNN [1]**	**AttParseNet (Ours)**
Accuracy	72.71%	**74.21**%

Table 6.4: Average accuracy over all attributes on LFWA. Both AttCNN and AttParseNet are trained on the unaligned CelebA images.

outperforms AttCNN on average by 1.5%.

Though we use face parsing as a means for improving facial attribute recognition, we do not compare our method with other face parsing or semantic segmentation methods, because our data is weakly labeled in two ways. First, the segments used in this work are automatically generated via the set of rules outlined above, and are not very precise. In some cases – e.g. straight hair – the segments cover too much area, and in some cases – e.g. blond hair – the segments do not cover the

full area. Second, it is an unanswered research question as to what area is covered by some attributes. For example, how does one determine gender? Does it comes from the entire face, as we've chosen here? As another example, smiling can be determined from the eyes, the nose, and the cheeks, as well as the mouth. This ambiguity in how many attributes manifest themselves is another reason why the segment data in this work is weakly labeled. Because of this, we do not feel that this problem is a traditional semantic segmentation problem, and so we do not compare our method to established methods in this field.

6.4 Summary

In this chapter we introduce a new method for facial attribute recognition in images, which we call AttParseNet. AttParseNet adds weakly labeled face parsing as an additional level of supervision in the attribute recognition network. We introduce a rule-based method for generating weakly labeled facial attribute segments based on landmark points. Using these weakly labeled attribute segments, we are able to add a segmentation loss to the attribute recognition framework in addition to an attribute prediction loss. Combining these two learning tasks in a single network allows for improved facial attribute recognition. We demonstrate the effectiveness of the proposed approach by comparing with the state-of-the-art method on three benchmark datasets: CelebA, LFWA, and UMD-AED. **AttParseNet outperforms the state of the art on all three datasets, by** 4%**,** 1.5%**, and** 1% **respectively, achieving** 100% **accuracy on *gender* on UMD-AED**. These

improvements indicate that previous methods were taking advantage of unwanted attribute relationships for prediction, and therefore were not learning a true representation for the attributes.

This is the first work to frame the attribute recognition problem as one of semantic segmentation. These results are very promising as we are able to achieve significant improvements over the state-of-the-art even with every rough facial attribute segment labels. As the segment labels are cleaned, the results will continue to improve. The next step in this research involves collecting ground truth segments in order to improve attribute recognition and to allow for more research in the direction of facial attribute segmentation.

Chapter 7: Conclusion and Future Work

In this dissertation we proposed several approaches to learning explainable facial features from noisy unconstrained visual data, including (1) using attribute relationships, (2) using selective learning to balance multi-label data, (3) incorporating time and motion constraints into model learning to better handle video data, and (4) re-framing the problem of attribute prediction as one of semantic segmentation. We evaluated each approach extensively on challenging datasets, including CelebA and LFWA. In addition, we introduced a new dataset, UMD-AED, for evaluating attribute recognition methods. We also labeled over 10,000 frames from the YouTube Faces dataset, and released rough attribute segmentation maps corresponding to CelebA images to enable research and development in these directions. The methods detailed in this dissertation have repeatedly pushed the state-of-the-art in attribute prediction and are useful for any future research in the area. Most of the proposed methods are not limited to attribute recognition, but are rather general purpose machine learning methods suitable for any noisy unconstrained recognition task.

7.1 Future Work

There are many avenues for future research in both attribute prediction and in recognition from noisy unconstrained data. We detail some directions for future research below.

7.1.1 Social Trait Recognition

The work on selective learning was inspired by research in developmental psychology the learning process of babies [126]. One potential area of investigation is to study the problem of social trait recognition from faces using attributes. Social traits consist of three main categories: attractiveness, competence, and trustworthiness. Humans can form a first impression of someone, based on these categories, in fewer than 33 milliseconds. People with visual impairments, or those on the autism spectrum may not be capable of making such judgments, though it may helpful for them to do so. For example, in the case of outdoor navigation, someone with a visual impairment may benefit from having a system that can identify friendly (trustworthy) people to help them when they are lost or confused. We have done some preliminary work on this topic, pulling from our past research on facial attribute recognition.

7.1.2 Micro-Expression Recognition

Expression recognition from images is a long-studied problem, mostly focused on recognizing a handful of posed expressions [14]. Facial expressions provide an

abundance of information without words. A smile can indicate that someone is having a good time, while a frown can indicate that they are uncomfortable. However, rarely do people actually frown when they are upset, and in fact they may continue to smile, but to most observers there are obvious differences between a true smile and the smile of someone who is uncomfortable. These are very subtle cues that someone with a visual impairment or someone on the autism spectrum may miss out on, resulting in difficult social exchanges. Recognizing these types of micro-expressions is key to improving social interactions for people who cannot recognize them naturally. The first step in this work will be to label data for the problem of micro-expression recognition. The best type of data for this problem will be movies and television shows, as micro-expressions will be realistic, and there are a wide range of these expressions that can be annotated.

7.1.3 Subject Clustering

Subject clustering is the problem of determining the number of unique subjects in a collection of images and building clusters for each subject. It is useful for organizing personal photo collections [127] [128], face verification and recognition [129], and image retrieval [130] [131].

This problem can be approached in the supervised and unsupervised settings. CelebA and LFWA are labeled with attributes and identities, and so it can be used for a supervised clustering approach, to determine which attributes are useful for clustering, and to evaluate the unsupervised approach. Approaches to develop and

use an attribute-based subject clustering work are as follows. First, develop two baselines for attribute-based clustering: divisive hierarchical clustering and agglomerative clustering. Second, investigate the clustering power of each facial attribute, and determine the relationship between clustering power and capacity for recognition. Third, identify redundant/overly correlated attributes based on joint clustering power. Fourth, use attributes for face recognition based on a reduced set of attributes selected based on their clustering power. And, finally determine which attributes are dynamic (i.e. change between two images of the same person) by performing attribute-based clustering and using identity information to absorb clusters together, loosening the constraint on a particular attribute. This information can be used to improve face verification, comparing with the method of [62].

Bibliography

[1] E. M. Hand, C. Castillo, and R. Chellappa. Doing the best we can with what we have: Multi-label balancing with selective learning for attribute prediction. *AAAI*, 2018.

[2] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. *ICCV*, 2015.

[3] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. *ICCV*, 2009.

[4] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Describable visual attributes for face verification and image search. *PAMI*, 2011.

[5] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. *ICCV*, 2015.

[6] N. Zhang, M. Paluri, M. A. Ranzato, T. Darrell, and L. Bourdev. Panda: Pose aligned networks for deep attribute modeling. *CVPR*, 2014.

[7] M. Ehrlich, T. J. Shields, T. Almaev, and M. R. Amer. Facial attributes classification using multi-task representation learning. *CVPR*, 2016.

[8] E. Rudd, M. Gunther, and T. Boult. Moon: A mixed objective optimization network for the recognition of facial attributes. *ECCV*, 2016.

[9] B. Siddiquie, Rogerio S. Feris, and Larry S. Davis. Image ranking and retrieval based on multi-attribute queries. *CVPR*, 2011.

[10] F. Song, X. Tan, and S. Chen. Exploiting relationships between attributes for improved face verification. *Computer Vision and Image Understanding*, 2014.

[11] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. *CVPR*, 2011.

[12] V. Bruce, A. M. Burton, E. Hanna, P. Healey, O. Mason, A. Coombes, R. Fright, and A. Linney. Sex discrimination: How do we tell the difference between male and female faces? *Perception*, 22(2):131, 1993.

[13] H. Abdi, D. Valentin, and B. Edelman. More about the difference between men and women: Evidence from linear neural network and the principal-component approach. *Perception*, 1995.

[14] B. A. Golomb, D. T. Lawrence, and T. J. Sejnowski. Sexnet: A neural network identifies sex from human faces. *NIPS*, 1991.

[15] S. Tamura, H. Kawai, and H. Mitsumoto. Male/female identification from 8x6 very low resolution faces images by neural network. *Pattern Recognition*, 1996.

[16] L. Bui, D. Tran, X. Huang, and G. Chetty. Face gender recognition based on 2d principal component analysis and support vector machine. *NSS*, 2010.

[17] S. Buchala, N. Davey, and T. Gale. Principal component analysis of gender, ethnicity, age, and identity of face images. *ICMI*, 2005.

[18] O. Smirg, J. Mikulka, M. Faundez-Zanuy, M. Grassi, and J. Mekyska. Gender recognition using pca and dct of face images. *IWANN*, 2011.

[19] A. Jain, J. Huang, and S. Fang. Gender identification using frontal face images. *ICME*, 2005.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[21] G. Shakhnarovich, P. Viola, and B Moghaddam. A unified learning framework for real time face detection and classification. *FG*, 2002.

[22] Z. Xu, L. Lu, and P. Shi. A hybrid approach to gender classification from face images. *ICPR*, 2008.

[23] T. Ojala, M. Pietikinen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. *ICPR*, 1994.

[24] H. C. Lian and B. L. Lu. Multi-view gender classification using local binary patterns. *ISNN*, 2006.

[25] Z. Yang and H. Ai. Demographic classification with local binary patterns. *Advances in Biometrics*, 2007.

[26] C. Shan. Learning local binary patterns for gender classification on real-world face images. *Pattern Recognition Letters*, 2012.

[27] L. A. Alexandre. Gender recognition: A multiscale decision fusion approach. *Pattern Recognition Letters*, 2010.

[28] J. Ylioinas, A. Hadid, and M. Pietikainen. Combining contrast information and local binary patterns for gender classification. *Image Analysis*, 2011.

[29] T. Jabid, M. H. Kabir, and O. Chae. Gender classification using local directional pattern (ldp). *ICPR*, 2010.

[30] A. Shobeirinejad and Y. Gao. Gender classification using interlaced derivative patterns. *ICPR*, 2010.

[31] X. Fu, G. Dai, C. Wang, and L. Zhang. Centralized gabor gradient histogram for facial gender recognition. *Natural Computation*, 2010.

[32] T. Lee. Image representation using 2d gabor wavelets. *PAMI*, 1996.

[33] X. M. Leng and Y. D. Wang. Improving generalization for gender classification. *ICIP*, 2008.

[34] H. Lian, B. Lu, and E. Takikawa. Gender recognition using a min-max modular support vector machine. *Advances in Natural Computation*, 2005.

[35] F. Scalzo, G. Bebis, M. Nicolescu, L. Loss, and A. Tavakkoli. Feature fusion hierarchies for gender classification. *ICPR*, 2008.

[36] B. Xia, H. Sun, and B. L. Lu. Multi-view gender classification based on local gabor binary mapping pattern and support vector machines. *Neural Networks*, 2008.

[37] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

[38] M. Demirkus, M. Toews, J. J. Clark, and T. Arbel. Gender classification from unconstrained video sequences. *CVPR*, 2010.

[39] J. G. Wang, J. Li, W. Y. Yau, and E. Sung. Boosting dense sift descriptors and shape contexts of face images for gender recognition. *CVPR*, 2010.

[40] J. G. Wang, J. Li, C. Y. Lee, and W. Y. Yau. Dense sift and gabor descriptors-based face representation with applications to gender recognition. *ICARCV*, 2010.

[41] J. M. Fellous. Gender discrimination and prediction on the basis of facial metric information. *Vision Research*, 1997.

[42] S. Mozaffari, H. Behravan, and R. Akbari. Gender classification using single frontal image per person: Combination of appearance and geometric based features. *ICPR*, 2010.

[43] G. Guo, C. R. Dyer, Y. Fu, and T. S. Huang. Is gender recognition affected by age? *ICCV*, 2009.

[44] C. Benabdelkader and P. Griffin. A local region-based approach to gender classification. *CVPR*, 2005.

[45] W. Gao and H. Ai. Face gender classification on consumer images in a multiethnic environment. *Advances in Biometrics*, 2009.

[46] E. Eidinger, R. Enbar, and T. Hassner. Age and gender estimation of unfiltered faces. *TIFS*, 2014.

[47] C. Levi and T. Hassner. Age and gender classification using convolutional neural networks. *IEEE Workshop on Analysis and Modeling of Faces and Gestures, CVPR*, 2015.

[48] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. *CVPR*, 2009.

[49] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. *CVPR*, 2009.

[50] A. Farhadi, I. Endres, and D. Hoiem. Attribute-centric recognition for cross-category generalization. *CVPR*, 2010.

[51] G. Wang and D. Forsyth. Joint learning of visual attributes, object classes and visual saliency. *CVPR*, 2009.

[52] Y. Wang and G. Mori. A discriminative latent model of object classes and attributes. *ECCV*, 2010.

[53] L. J. Li, H. Su, Y. Lim, and L. Fei-Fei. Objects as attributes for scene classification. *ECCV*, 2010.

[54] D. Jayaraman, Fei Sha, and Kristen Grauman. Decorrelating semantic visual attributes by resisting the urge to share. *CVPR*, 2014.

[55] A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia. Multi-task cnn model for attribute prediction. *arXiv preprint*, 2015.

[56] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. *CVPR*, 2011.

[57] W. Li and N. Vasconcelos. Recognizing activities by attribute dynamics. *NIPS*, 2012.

[58] H. T. Cheng, F. T. Sun, M. Griss, P. Davis, J. Li, and D. You. Nuactive: Recognizing unseen new activities using semantic attribute-based learning. *ICMSAS*, 2013.

[59] H. T. Cheng, M. Griss, P. Davis, J. Li, and D. You. Towards zero-shot learning for human activity recognition using semantic attribute sequence model. *UBICOMP*, 2013.

[60] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei. Human action recognition by learning bases of action attributes and parts. *ICCV*, 2011.

[61] Z. Zhang, C. Wang, B. Xiao, W. Zhoe, and S. Liu. Attribute regularization based human action recognition. *TIFS*, 2013.

[62] J. Zheng, Z. Jiang, R. Chellappa, and J. P. Phillips. Submodular attribute selection for action recognition in video. *NIPS*, 2014.

[63] D. A. Vaquero, R. S. Feris, D. Tran, L. Brown, A. Hampapur, and M. Turk. Attribute-based people search in surveillance environments. *WACV*, 2009.

[64] L. Bourdev, S. Maji, and J. Malik. Describing people: A poselet-based approach to attribute classification. *ICCV*, 2011.

[65] J. Joo, S. Wang, and S. C. Zhu. Human attribute recognition by rich appearance dictionary. *ICCV*, 2013.

[66] G. Sharma, F. Jurie, and C. Schmid. Expanded parts model for human attribute and action recognition in still images. *CVPR*, 2013.

[67] H. J. Wang, Y. L. Lin, C. Y. Huang, Y. L Hou, and W. Hsu. Full body human attribute detection in indoor surveillance environment using color-depth information. *AVSS*, 2013.

[68] J. Zhu, S. Liao, D. Yi, Z. Lei, and S. Z. Li. Multi-label cnn based pedestrian attribute learning for soft biometrics. *ICB*, 2015.

[69] P. Sudowe, H. Spitzer, and B. Leibe. Person attribute recognition with a jointly-trained holistic cnn model. *ICCV*, 2015.

[70] N. Kumar, P. N. Belhumeur, and S. K. Nayar. Facetracer: A search engine for large collections of images with faces. *ECCV*, 2008.

[71] T. Berg and P. N. Belhumeur. Tom-vs-pete classifiers and identity-preserving alignment for face verification. *British Machine Vision Conference*, 2012.

[72] J. Wang, Y. Cheng, and R. S. Feris. Walk and learn: Facial attribute representation learning from egocentric video and contextual data. *CVPR*, 2016.

[73] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *NIPS*, 2007.

[74] R. Caruana. Multitask learning. *Machine Learning*, 1997.

[75] S. Parameswaran and K. Weinberger. Large margin multi-task metric learning. *NIPS*, 2010.

[76] T. Devries, K. Biswaranjan, and G. W. Taylor. Multi-task learning of facial landmarks and expression. *CRV*, 2014.

[77] R. Ranjan, V. M. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *arXiv preprint*, 2015.

[78] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim. Rotating your face using multi-task deep neural network. *CVPR*, 2015.

[79] C. Zhang and Z. Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. *WACV*, 2014.

[80] Z. Zhang, P. Luo, C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. *ECCV*, 2014.

[81] Q. Zhou, G. Wang, K. Jia, and Q. Zhao. Learning to share latent tasks for action recognition. *ICCV*, 2013.

[82] X. Yuan, X. Liu, and S. Yan. Visual classification with multitask joint sparse representation. *TIP*, 2012.

[83] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *IJCV*, 2013.

[84] C. Ding, C. Xu, and D. Tao. Multi-task pose-invariant face recognition. *TIP*, 2015.

[85] S. J. Hwang, F. Sha, and K. Grauman. Sharing features between objects and their attributes. *CVPR*, 2011.

[86] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: A survey of recent advances. *Signal Processing Magazine*, 2015.

[87] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. *ECCV*, 2010.

[88] L. Bo, X. Ren, and D. Fox. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. *NIPS*, 2011.

[89] M. Yang, L. Zhang, X. Feng, and D. Zhang. Fisher discrimination dictionary learning for sparse representation. *ICCV*, 2011.

[90] B. Lu, R. Chellappa, and N. M. Nasrabadi. Incremental dictionary learning for unsupervised domain adaptation. *BMVC*, 2015.

[91] B. Gong, Y. Shi, F. Sha, and K Grauman. Geodesic flow kernel for unsupervised domain adaptation. *CVPR*, 2012.

[92] R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. *ICCV*, 2011.

[93] H. Ho and R. Gopalan. Model-driven domain adaptation on product manifolds for unconstrained face recognition. *IJCV*, 2014.

[94] Q. Qui, V. M. Patel, P. Turaga, and R. Chellappa. Domain adaptive dictionary learning. *ECCV*, 2012.

[95] S. Shekhar, V. M. Patel, H. V. Nguyen, and R. Chellappa. Generalized domain-adaptive dictionaries. *CVPR*, 2013.

[96] Q. Chen, J. Huang, R. S. Feris, L. M. Brown, J. Dong, and S. Yan. Deep domain adaptation for describing people based on fine-grained clothing attributes. *CVPR*, 2015.

[97] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint*, 2014.

[98] A. Paul, F. Rottensteiner, and C. Heipke. Iterative re-weighted instance transfer for domain adaptation. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2016.

[99] T. Tommasi and B. Caputo. Frustratingly easy nbnn domain adaptation. *ICCV*, 2013.

[100] K. Tang, V. Ramanathan, L. Fei-Fei, and D. Koller. Shifting weights: Adapting object detectors from image to video. *NIPS*, 2012.

[101] M. Chen, K. Q. Weinberger, and J. Blitzer. Co-training for domain adaptation. *NIPS*, 2011.

[102] J. R. Barr, K. W. Bowyer, P. J. Flynn, and S. Biswas. Face recognition from video: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 2012.

[103] H. S. Parekh, D. G. Thakore, and U. K. Jaliya. A survey on object detection and tracking methods. *International Journal of Innovative Research in Computer and Communication Engineering*, 2014.

[104] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 2010.

[105] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *CSUR*, 2003.

[106] J. C. Niebles, C. Wei Chen, and L. Fei-fei. Modeling temporal structure of decomposable motion segments for activity recognition. *ECCV*, 2010.

[107] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human action classes from videos in the wild. *CRCV-TR-12-01*, 2012.

[108] D. Jayaraman and K. Grauman. Slow and steady feature analysis: Higher order temporal coherence in video. *CVPR*, 2016.

[109] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. *ICML*, 2009.

[110] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherence metrics. *ICCV*, 2015.

[111] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. *ICCV*, 2015.

[112] E. Park, X. Han, T. L. Berg, and A. C. Berg. Combining multiple sources of knowledge in deep cnns for action recognition. *WACV*, 2016.

[113] J. Warrell and S. J. Prince. Labelfaces: Parsing facial features by multiclass labeling with an epitome prior. *ICIP*, 2009.

[114] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller. Augmenting crfs with boltzmann machine shape priors for image labeling. *CVPR*, 2013.

[115] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. C. Yang. Exemplar based face parsing. *CVPR*, 2013.

[116] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[117] C. H. S. F. Liu, J. M. Yang, and M. Y. Yang. Multi-object convolutional learning for face labeling. *CVPR*, 2015.

[118] P. Luo, X. G. Wang, and X. O Tang. Hierarchical face parsing via deep learning. *CVPR*, 2012.

[119] Y. Zhao, F. Tang, W. Dong, F. Huang, and X. Zhang. Joint face alignment and segmentation via deep multi-task learning. *Multimedia Tools and Applications*, 2018.

[120] L. C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. *CVPR*, 2016.

[121] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *University of Massachusetts Amherst Technical Report*, 2007.

[122] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. *CVPR*, 2011.

[123] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint*, 2014.

[124] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint*, 2014.

[125] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[126] S. Liu and E. S. Spelke. Six-month-old infants expect agents to minimize the cost of their actions. *Cognition*, 2017.

[127] E. Ardizzone, M. La Cascia, and F. Vella. Unsupervised clustering in personal photo collections. *International Workshop on Adaptive Multimedia*, 2008.

[128] M. Zhao, Y. Teo, S. Liu, T. Chua, and R. Jain. Automatic person annotation of family photo album. *Image and Video Retreival*, 2006.

[129] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CVPR*, 2015.

[130] A. Holub, P. Moreels, and P. Perona. Unsupervised clustering for google searches of celebrity images. *FG*, 2008.

[131] F. Perronin and J.L. Dugelay. Clustering face images with application to image retrieval in large databases. *Defense and Security*, 2005.