DD2427 Final Project Report Human face attributes prediction with Deep Learning

Mohamed Abdulaziz Ali Haseeb
moaah@kth.se

Abstract

We explore using deep Convolutional Neural Networks (CNN) to predict human attributes like skin tune, hair color and age from a face image. Using a dataset of face images annotated with facial attributes, we train a linear classifier for attribute prediction using representations extracted from a CNN pre-trained for a general object classification task. Furthermore, we fine-tune the pre-trained CNN using the attributes-annotated dataset and conduct a set of experiments. We report on our experiments and discuss the achieved results. When trained on 20% of CelebA dataset, our method achieves an average accuracy of $\sim 89.9\%$ predicting 40 different attributes per image.

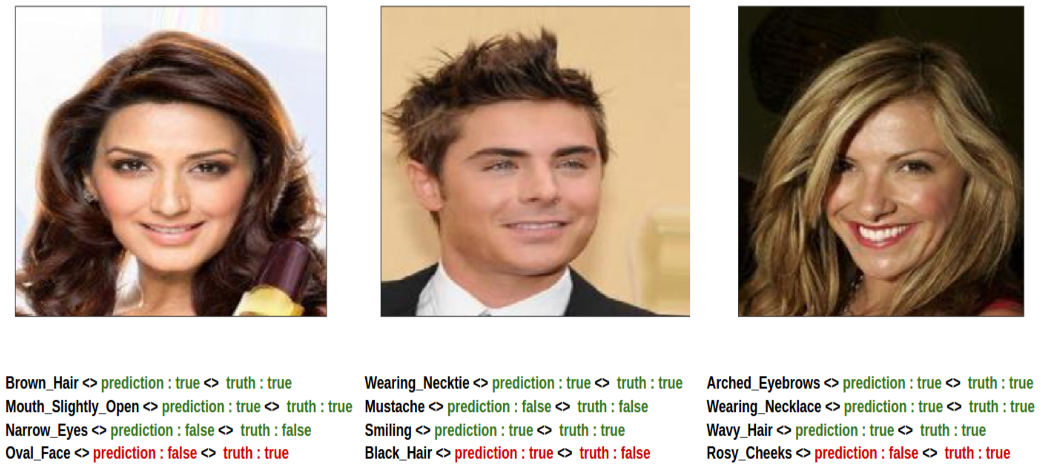

Brown_Hair <> prediction : true <> truth : true
Mouth_Slightly_Open <> prediction : true <> truth : true
Narrow_Eyes <> prediction : false <> truth : false
Oval_Face <> prediction : false <> truth : true
Wearing_Necktie <> prediction : true <> truth : true
Mustache <> prediction : false <> truth : false
Smiling <> prediction : true <> truth : true
Black_Hair <> prediction : true <> truth : false
Arched_Eyebrows <> prediction : true <> truth : true
Wearing_Necklace <> prediction : true <> truth : true
Wavy_Hair <> prediction : true <> truth : true
Rosy_Cheeks <> prediction : false <> truth : true

Figure 1: Attribute predictions for sample test images. Only 4 out 40 predicted attributes are shown per image.

1 Introduction

Predicting human face visual attributes like age, gender, hair color from face images has several applications in face verification (where the task is to compare the identify of a given image to a known identity), face identification (where a given face image identity has to be inferred from a set of known identities) and face retrieval (where face images similar to a given face are retrieved from a database) [1]. This is though, as many computer vision tasks, a challenging task because of the wide variations on images caused by problems like occlusion, clutter and lighting.

Deep learning has repeatedly been shown to produce unprecedented results on a variety of fields, and computer vision is not an exception. The success of deep learning to a large extent is accounted to the emergence of large datasets and the recent developments in processor technology (Graphical Processing Units) [2]. Training deep CNNs hence, requires a plenty of data, ample computational capacity and expertise knowledge; resources afforded by few entities. Transfer learning techniques allow reusing models trained on specific task to be used for another task with minimal effort [3]. These techniques provide a means by which the masses of organizations, scientists and individuals can leverage deep learning, by reusing pre-trained deep Neural Networks after tuning them to their specific tasks with affordable efforts.

This work explores using off-the-shelf CNN pre-trained for a general object classification task, to build a classifier that predicts facial attributes like hair color, gender, etc. (figure 1) and reports the results of experimenting with different variants of such system. Specifically, we use a subset of the face-aligned version of CelebA dataset [4], whose images are labelled with 40 image-level attributes each, to develop three attribute prediction methods:

1. Using a copy of CNN-F introduced in [5] (VGG-F), pre-trained for the general object classification task ILSVRC 2012 [6], to extract image representations and use them to train a bank of 40 simple classifiers (linear Support Vector Machines (SVM)) for attribute predictions. Each classifier is trained to predict the presence or absence of a single attribute given an input face image.

2. Fine-tuning a variant of VGG-F CNN to predict a vector of 40 binary values, from an input face image. Each value of the binary predicted vector, corresponds to one attribute from the input face image.

3. Using image representations extracted from the fine-tuned network in

2 above to train a bank of linear SVMs for attribute predictions.

The rest of this report is organized as follows. Section 2 presents the related work. This is followed by a detailed description of the developed methods in section 3. The different performed experiments and the obtained results are presented and discussed in section 4. Section 5 concludes the report by summarizing the performed work and the obtained results.

2 Background

Image representations extracted from deep CNN layers has been shown to be a very power general representations, useful for a wide range of image recognition tasks. Razavian et al [7] reported results comparable to the ones reported by what were then state of the art methods in several recognition tasks, using image representations extracted from the OverFeat CNN trained to classify objects in ILSVR13. Their setup involved using the extracted features to train a simple linear Support Vector Machine (SVM) for each recognition task. As highlighted in [7], the simplicity of this approach and its effectiveness suggest that it should be a baseline in every recognition task.

Depending on the locality of the extracted features, attribute recognition methods can be grouped into two categories [4]. 1) Global attribute recognition methods that uses global features extracted from the whole input image. These methods suffer from deformations of objects. 2) Local methods that localize object parts first (e.g. eye location, lip location, etc.), usually using hand-crafted features, and then use features extracted from these localized object parts to train classifiers for the attribute recognition task [1]. The success of these methods depends largely on its localization capability.

In [4], Liu et al proposed a global method for attribute prediction using a cascade of three deep CNNs. The first two CNNs of the cascade are used to localize the face region within the input image. The localized face region is then used as input to the third CNN, whose last fully connected (FC) layer responces are used as input features to a bank of Support Vector Machines (SVMs) trained to predict the attribute values (one SVM per attribute). The first two CNNs in the casecade were pre-trained to classify between a massive number of general objects [6] and fine-tuned using face images labeled with image-level attributes. The third CNN was pre-trained for identity classification and fine-tuned using the same attribute-labeled face images dataset.

Ranjan et al [8] built a deep CNN framework that is trained end-to-end to simultaneously perform face detection, facial landmark localization, head pose estimation and gender recognition on a given input image. Their network architecture exploited the observation that, lower network layers are good at detecting edges and corners and hence more suitable for localization tasks like pose estimation and landmark localization, while higher network layers capture higher semantic features and are therefore suited for the more complex tasks of face detection and gender recognition. The network is comprised of two parts. The first part is a CNN based on the AlexNet for image classification [9]. The second part of the network is a deep CNN that fuses the outputs of three layers from the first CNN (a low layer output, a middle layer output and a high layer output) into single *fusion* layer. From the fusion layer, the 2^{nd} network then branches into several branches of FC layers, each corresponding to one of the different tasks. The *multi-task* network is trained, end-to-end, using a total loss function compiled via the weighted sum of the task-specific loss functions.

Our method builds on the ideas of [7] and [4] of extracting image representations from pre-trained CNNs and use that to train simple linear classifiers for the prediction tasks. Further more we modify and fine-tune the pre-trained CNN as described in section 3.4.2.

3 Approach

This section starts by presenting a formulation for the attribute prediction task, followed by a description of the used dataset. The different attribute prediction methods are then explained.

3.1 Problem formulation

The problem of predicting a set of attributes given an input image can be formulated as follows:

- ***Given a dataset*** D of image-attributes pairs (x_i, y_i). $D = \{(x_1, y_1)...(x_n, y_n)\}$ where $x_i \in \mathbb{R}^{w \times h \times c}$ and $y_i \in \{-1, 1\}^d$. w, h and c are the input image width, height and number of channels. d is the number of attributes assigned to a single image. D is split into two sets: D_{train} and D_{test}.

- ***The goal is to use*** D_{train} ***to train a classifier*** f such that, $f(x; \theta) = F(g(x; \theta))$ where $f(x; \theta) \in \{-1, 1\}^d$ and θ represents the network parameters. $g(x; \theta) \in \mathbb{R}^d$, is a discriminant function with d

values, each, representing a confidence of the presence or absence of an attribute in x.

- ***That maximizes the attribute prediction accuracy*** given by:

$$Accuracy = \frac{1}{|D_{test}|} \sum_{(x_j,y_j)\in D_{test}} P[F(g(x_j;\theta)) == y_j] \tag{1}$$

Where $P \in \{1\}^d$, Note $[x == x] = 1$ and $[x == y] = 0$.

- ***Using a loss function*** $L(y, g(x;\theta))$ that measures how well y is predicted by $g(x;\theta)$.

3.2 Dataset

The face-aligned version of the publicly-available *Large-scale CelebFaces Attributes (CelebA) dataset* dataset [4] was used in the experiments. CelebA contains more than $200,000$ images, each is annotated with 40 facial attributes and 5 landmark locations. The dataset images exhibit a wide range of variations on identities, pose and clutter. Figure 1 shows some samples from the CelebA dataset.

3.3 Data preprocessing

Unless otherwise specified, before used as input to the CNN, all the dataset images were preprocessed as follows:

1. Randomly shuffle the dataset and split it into training and testing sets.
2. Calculate the training set mean image, by performing a pixel-wise average over the training set images. The mean training image of the training set used in these experiments is shown in figure 2.
3. Resize each image to the VGG-F network expected input size; $224 \times 224 \times 3$.
4. Subtract the mean image calculated in 1 above from each image in the dataset (both training and testing sets).

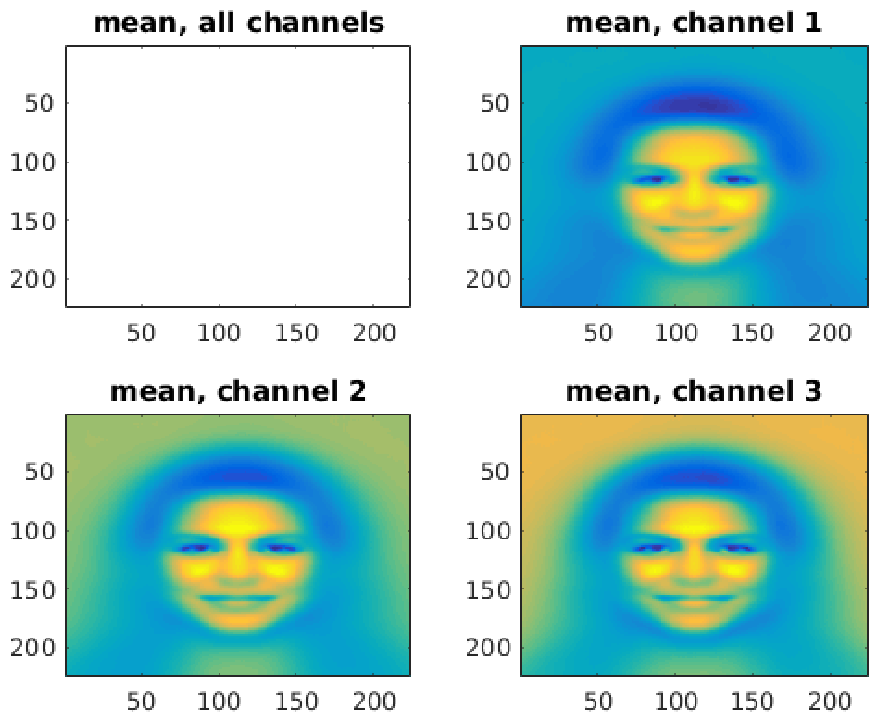


Figure 2: Training set mean image

3.4 Attributes prediction methods

For all the different prediction methods, a copy of CNN-F introduced by [5], (VGG-F), was used. VGG-F was pre-trained for the classification task of ILSVRC 2012 [6], where, for a given image, the network is trained to predict the top n objects (out of possible 1000 general objects) on the image images. Table 1 shows the VGG-F network architecture.

Table 1: Original VGG-F network architecture. Filter size is of the form (w_f, h_f, d_f, c_f), where w_f, h_f and d_f are the width, height, depth of the filter and c_f is the number of filters. Output size is of the form (w_o, h_o, c_o), where w_o, h_o are the width and height of the output and c_o is the number of output channels. Pool size is of the form (w_p, h_p), where w_p and h_p are the width and height of the pooling kernel. Stride and pad are of the shape (l, r, t, b), where l, r, t and b correspond to either stride or pad on the left, right, top or bottom directions.

Layer number/type/name	**Filter (Pool) size**	**Stride**	**Pad**	**Output size**
0/input/-	-	-	-	$(224, 224, 3)$
1/conv/conv1	$(11, 11, 3, 64)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(54, 54, 64)$
2/relu/relu1	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(54, 54, 64)$
3/lrn/norm1	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(54, 54, 64)$
4/pool1/max pool	$(3, 3)$	$(2, 2, 2, 2)$	$(0, 1, 0, 1)$	$(27, 27, 64)$
5/conv/conv2	$(5, 5, 64, 256)$	$(1, 1, 1, 1)$	$(2, 2, 2, 2)$	$(27, 27, 256)$
6/relu/relu2	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(27, 27, 256)$
7/lrn/norm2	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(27, 27, 256)$
8/pool2/max pool	$(3, 3)$	$(2, 2, 2, 2)$	$(0, 1, 0, 1)$	$(13, 13, 256)$
9/conv/conv3	$(3, 3, 256, 256)$	$(1, 1, 1, 1)$	$(1, 1, 1, 1)$	$(13, 13, 256)$
10/relu/relu3	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(13, 13, 256)$
11/conv/conv4	$(3, 3, 256, 256)$	$(1, 1, 1, 1)$	$(1, 1, 1, 1)$	$(13, 13, 256)$
12/relu/relu4	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(13, 13, 256)$
13/conv/conv5	$(3, 3, 256, 256)$	$(1, 1, 1, 1)$	$(1, 1, 1, 1)$	$(13, 13, 256)$
14/relu/relu5	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(13, 13, 256)$
15/pool3/max pool	$(3, 3)$	$(2, 2, 2, 2)$	$(0, 1, 0, 1)$	$(6, 6, 256)$
16/conv/fc1	$(6, 6, 256, 4096)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
17/relu/relu6	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
18/conv/fc2	$(1, 1, 4096, 4096)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
19/relu/relu7	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
20/conv/prediction	$(1, 1, 4096, 1000)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 1000)$

3.4.1 Pre-trained VGG-F + linear SVMs

Responses from one of the high level hidden layers of the VGG-F CNN pre-trained for the ILSVRC 2012 classification task, were collected and used as image representations(note, in the preprocessing phase, the mean image of the original training data used for VGG-F CNN pre-training was subtracted from the images before extracting the representations from the network, and not the mean of the training data). Given a set of images $x = \{x_1, x_2 ... x_n\}$, with dimensions as described in 3.1, the extracted representation from layer l will be of the form $x^l = \{x_1^l, x_2^l ... x_n^l\}$, where x_i^l is the CNN layer l response for input image x_i, $x_i^l \in \mathbb{R}^{l_s}$ and l_s is layer l response (or output) size.

The extracted representations are then used to train a group of linear SVMs, each predicting the presence or absence of a single attribute. The SVMs were trained with Stochastic Gradient Descent (SDG), by minimizing a regularized loss function of the form:

$$L(y^j, x^l; w^j, b^j) = \frac{\lambda}{2}||w^j||^2 + \sum_{i=1}^{n} max\{0, 1 - y_i^j((w^j)^T x_i^l + b^j)\} \tag{2}$$

Where w^j and b^j are the weights and bias terms for the classifier trained for attribute j. $y^j = \{y^1, y^2 ... y^n\}, y^j \in \{-1, 1\}^n$ are attribute's j values corresponding to each images x.

A total of 40 linear SVMs were trained.

3.4.2 Fine-tuned VGG-F

The VGG-F CNN is modified to output a binary vector $\hat{y}$ corresponding to the CNN predictions of the presence or absence of each one of the d attributes, where $\hat{y} = \{\hat{y_1}, \hat{y_2} ... \hat{y_n}\}$ and $\hat{y} \in \{-1, 1\}^d$. The modification was done by removing the last fully connected layer (named prediction in table 1) and adding two new layers: a *dropout* layer (dout1) of rate 0.5 and a fully connected layer (prediction) of output size $d = 40$. The modified VGG-F is shown in table 2.

The fine-tuning was done by:

1. applying the preprocessing described in 3.3 to the dataset images, and
2. loading the pre-trained VGG-F CNN weights
3. adding the new layers
4. adding a logistic loss of the form:

$$L(y; x^o) = \sum_{i=1}^{i=n} \sum_{j=1}^{j=d} L(y_{ij}; x_{ij}^o) \tag{3}$$

$$L(y_{ij}; x_{ij}^o) = -log \frac{1}{1 + e^{-y_{ij} x_{ij}^o}} \tag{4}$$

Where x^o_{ij} is the output layer's neuron j response to input image x_i, and y_{ij} is the image x_i ground truth label for attribute j.

5. and finally training the model with SGD updating all the weights of the whole CNN.

During inference phase, given an image x_i, predicting attribute's j presence ($\hat{y_{ij}} = 1$) or absence ($\hat{y_{ij}} = -1$) is performed using the below formula [10].

$$\hat{y_{ij}} = \begin{cases} 1 & \text{if } x_{ij} > 0 \\ -1 & \text{if } x_{ij} < 0 \end{cases}$$

Table 2: Modified VGG-F network architecture. Fine-tuned for 40 binary attributes prediction.

Layer number/type/name	**Filter (Pool) size**	**Stride**	**Pad**	**Output size**
0/input/-	-	-	-	$(224, 224, 3)$
1/conv/conv1	$(11, 11, 3, 64)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(54, 54, 64)$
2/relu/relu1	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(54, 54, 64)$
3/lrn/norm1	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(54, 54, 64)$
4/pool1/max pool	$(3, 3)$	$(2, 2, 2, 2)$	$(0, 1, 0, 1)$	$(27, 27, 64)$
5/conv/conv2	$(5, 5, 64, 256)$	$(1, 1, 1, 1)$	$(2, 2, 2, 2)$	$(27, 27, 256)$
6/relu/relu2	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(27, 27, 256)$
7/lrn/norm2	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(27, 27, 256)$
8/pool2/max pool	$(3, 3)$	$(2, 2, 2, 2)$	$(0, 1, 0, 1)$	$(13, 13, 256)$
9/conv/conv3	$(3, 3, 256, 256)$	$(1, 1, 1, 1)$	$(1, 1, 1, 1)$	$(13, 13, 256)$
10/relu/relu3	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(13, 13, 256)$
11/conv/conv4	$(3, 3, 256, 256)$	$(1, 1, 1, 1)$	$(1, 1, 1, 1)$	$(13, 13, 256)$
12/relu/relu4	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(13, 13, 256)$
13/conv/conv5	$(3, 3, 256, 256)$	$(1, 1, 1, 1)$	$(1, 1, 1, 1)$	$(13, 13, 256)$
14/relu/relu5	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(13, 13, 256)$
15/pool3/max pool	$(3, 3)$	$(2, 2, 2, 2)$	$(0, 1, 0, 1)$	$(6, 6, 256)$
16/conv/fc1	$(6, 6, 256, 4096)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
17/relu/relu6	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
18/conv/fc2	$(1, 1, 4096, 4096)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
19/relu/relu7	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
20/dropout/dout1	$-$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 4096)$
21/conv/prediction	$(1, 1, 4096, 1000)$	$(1, 1, 1, 1)$	$(0, 0, 0, 0)$	$(1, 1, 40)$

3.4.3 Fine-tuned VGG-F + linear SVMs

Similar to 3.4.1, image representations were extracted from the high level hidden layers of the tuned VGG-F described in section 3.4.2, and used to train a set of d linear SVMs.

4 Experiments and results

For testing the different prediction methods, a set of 36546 images from the CelebA dataset was used ($\sim$ 20% of the full CelebA dataset). 90% of the dataset was used for training and rest for testing.

For both *"Pre-trained VGG-F + linear SVMs"* method described in 3.4.1 and *"Fine-tuned VGG-F + linear SVMs"* method described in 3.4.3, two experiments were performed: 1) using image representations extracted from layer 16 (fc1) and 2) using image representations extracted from layer 18 (fc2). Additionally, the same two experiments were repeated for a version of *"Fine-tuned VGG-F + linear SVMs"* without the dropout layer (dout1). All the SVMs were trained using $\lambda = 0.0001$ and a total of 20 epocs. Figure 4 plots the prediction accuracy in the test set as a function of the layer number from which image representations were extracted. The accuracy was calculated using equation 1.

Beside verifying the *"Fine-tuned VGG-F"* described in 3.4.2, two additional variants were verified: 1) A version of *"Fine-tuned VGG-F"* without the dropout layer (dout1) and 2) a version of *"Fine-tuned VGG-F"* after subtracting both the training set mean image and image mean of the dataset used to pre-train the original VGG-F, from the dataset images. The reported fine-tuning results are for a learning rate of 0.0001, batch size of 100 and different epoc numbers (60 epocs for some CNNs).

Table 3 summarizes the results for all the experiments. Figures 1 and 4 shows samples for correctly and wrongly predicted attributes.

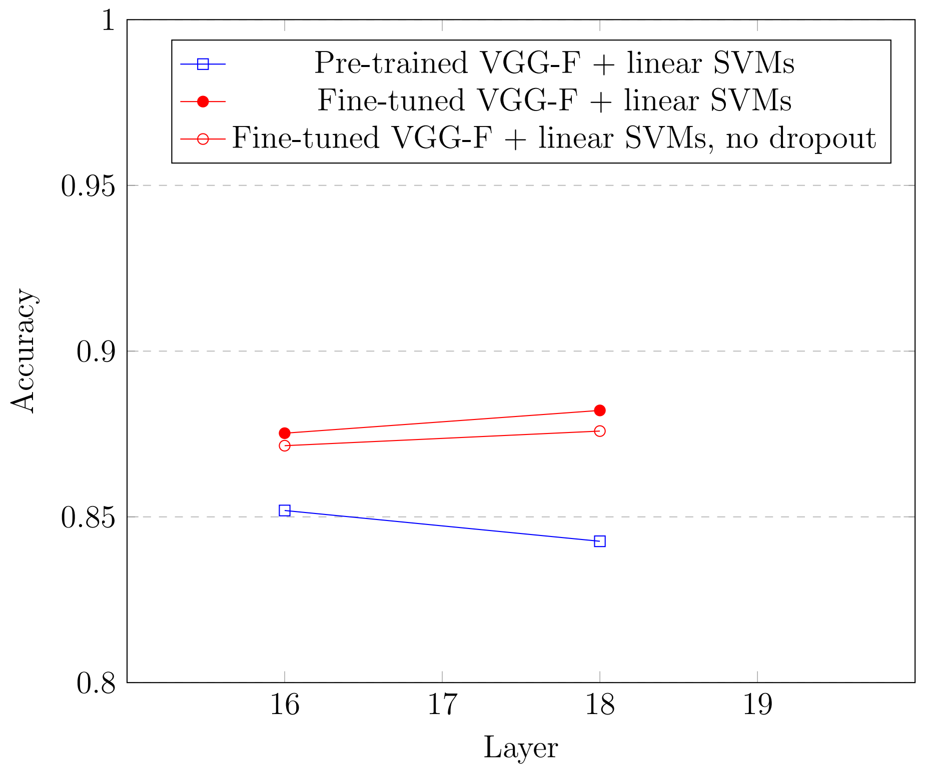
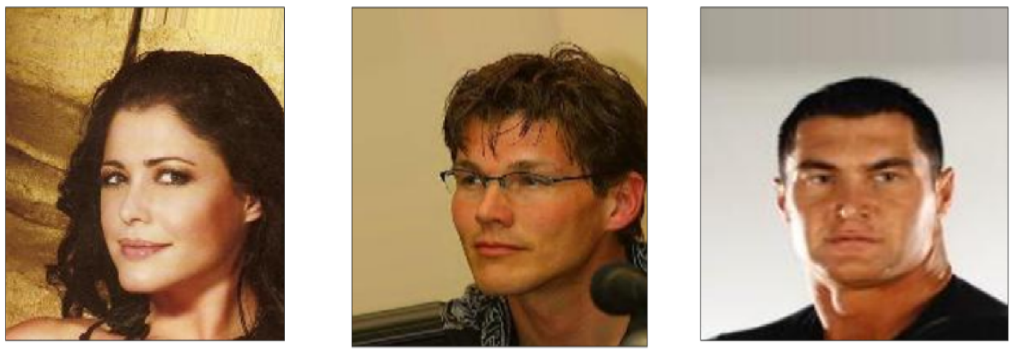


Figure 3: Prediction accuracy vs layer number



Arched_Eyebrows <> prediction : false <> truth : true	Big_Nose <> prediction : true <> truth : false	Bushy_Eyebrows <> prediction : false <> truth : true
Black_Hair <> prediction : false <> truth : true	Pointy_Nose <> prediction : false <> truth : true	High_Cheekbones <> prediction : false <> truth : true
Hig_Cheekbonesh <> prediction : true <> truth : false	Brown_Hair <> prediction : false <> truth : true	Pointy_Nose <> prediction : false <> truth : true
Heavy_Makeup <> prediction : true <> truth : true	Eyeglasses <> prediction : true <> truth : true	Black_Hair <> prediction : true <> truth : true

Figure 4: Sample test images with wrong predictions

Table 3: Experiments results summary

Method	Prediction accuracy
Pre-trained VGG-F + linear SVMs, layer 16 representations	0.85192
Pre-trained VGG-F + linear SVMs, layer 18 representations	0.84264
Fine-tuned VGG-F + linear SVMs, layer 16 representations	0.87524
Fine-tuned VGG-F + linear SVMs, layer 18 representations	0.88211
Fine-tuned VGG-F + linear SVMs, no dropout, layer 16 representations	0.87147
Fine-tuned VGG-F + linear SVMs, no dropout, layer 18 representations	0.87588
Fine-tuned VGG-F	0.89859
Fine-tuned VGG-F (subtracting combined mean)	0.89746
Fine-tuned VGG-F, no dropout	**0.8987**

4.1 Results discussion

The results of training linear SVMs for attribute prediction using image representations extracted from CNN, shows that given a CNN pre-trained on a different task (than attribute prediction), the lower the layer from which image representations are extracted the higher the prediction accuracy. While, if a CNN tuned for the same (or similar) prediction task was used, the higher the layer from which image representations are extracted the higher the prediction accuracy. This is an expected result, since, higher CNN layers learns features more specific to the task for which the CNN was trained. That is said, the results obtained using the representation from the pre-trained model and those when a fine-tuned model was used are comparable, and one with limited computational resources can avoid fine-tuning the CNN and use the representations of the pre-trained CNN with simple linear classifiers.

The *"Fine-tuned VGG-F* method yielded the best prediction accuracy of 89.9% compared to 88.2% achieved by *"Fine-tuned VGG-F + linear SVMs"* using layer 18 representations . This can be due to the fact that the network was trained to predict all the 40 attributes at the same time which allowed the network to exploit more information, while each one of the linear SVMs was trained to predict one attribute in isolation of the other attributes present on the image.

Placing the dropout layer before the last fully connect layer, to act as a regularizer didn't improve the prediction accuracy. More experiments that adds dropout layers in the other fully connected layers is needed to have better understanding of the impact.

Note, the fact that our method prediction accuracy of 89.9% exceeds that is of the state-of-art method of [4] (which achieved 87%), is due to [4] using the complete non-aligned version of CelebA dataset.

5 Conclusion

In this work, different methods of using pre-trained deep CNNs to train a facial attribute predictor were explored. Using image representations extracted from pre-trained CNNs as well as representations extracted after CNN fine-tuning were tested. Our results agrees with the related previous work, that, image representations extracted from pre-trained CNNs are powerful general representations, which when combined with simple linear classifiers yield to accuracies comparable to methods tailored to the specific task at hand. Although fine-tuning a pre-trained CNN bumps the prediction accuracy, using the pre-trained CNN representations to train a simple linear classifiers might be preferable considering the computational costs associated with fine-tuning the pre-trained CNN.

References

[1] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 365–372, Sept 2009.

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[3] Lisa Torrey and Jude Shavlik. Transfer learning. In E. Soria, J. Martin, R. Magdalena, M. Martinez, and A. Serrano, editors, *Handbook of Research on Machine Learning Applications*. IGI Global, 2009.

[4] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *2015 IEEE International Conference on*

Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, pages 3730–3738, 2015.

[5] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *CoRR*, abs/1405.3531, 2014.

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[7] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

[8] Rajeev Ranjan, Vishal M. Patel, and Rama Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *CoRR*, abs/1603.01249, 2016.

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

[10] Andrea Vedaldi and. Section 4.7.2 attribute losses of matconvnet convolutional neural networks for matlab. http://www.vlfeat.org/matconvnet/matconvnet-manual.pdf. [Online; accessed 30-May-2016].