Convolutional Neural Networks for Age and Gender Classification

Ari Ekmekji
Stanford University
aekmekji@stanford.edu

Abstract

This paper focuses on the problem of gender and age classification for an image. I build off of previous work [12] that has developed efficient, accurate architectures for these tasks and aim to extend their approaches in order to improve results. The first main area of experimentation in this project is modifying some previously published, effective architectures used for gender and age classification [12]. My attempts include reducing the number of parameters (in the style of [19]), increasing the depth of the network, and modifying the level of dropout used. These modifications actually ended up causing system performance to decrease (or at best, stay the same) as compared with the simpler architecture I began with. This verified suspicions I had that the tasks of age and gender classification are more prone to over-fitting than other types of classification.

The next facet of my project focuses on coupling the architectures for age and gender recognition to take advantage of the gender-specific age characteristics and age-specific gender characteristics inherent to images. This stemmed from the observation that gender classification is an inherently easier task than age classification, due to both the fewer number of potential classes and the more prominent intra-gender facial variations. By training different age classifiers for each gender I found that I could improve the performance of age classification, although gender classification did not see any significant gains.

1. Introduction

Over the last decade, the rate of image uploads to the Internet has grown at a nearly exponential rate. This newfound wealth of data has empowered computer scientists to tackle problems in computer vision that were previously either irrelevant or intractable. Consequently, we have witnessed the dawn of highly accurate and efficient facial detection frameworks that leverage convolutional neural networks under the hood. Applications for these systems include everything from suggesting who to "tag" in Facebook photos to pedestrian detection in self-driving cars. However the next major step to take building off of this work is to ask not only how many faces are in a picture and where they are, but also what characteristics do those faces have. The goal of this project do exactly that by attempting to classify the age and gender of the faces in an image.

Applications for this technology have a broad scope and the potential to make a large impact. For example, many languages have distinct words to be used when addressing a male versus a female or an elder versus a youth. Therefore automated translation services and other forms of speech generation can factor in gender and age classification of subjects to improve their performance. Also, having an idea about the age and gender of a subject makes the task of recognizing that subject significantly easier. This could be used to aid assisted vision devices for those with deteriorating, or lost, eyesight. Social media websites like Facebook could use the information about the age and gender of the people to better infer the context of the image. For example, if a picture contains many people studying together, Facebook might be able to caption the scene with "study session." However if it can also detect that the people are all men in their early 20s and that some are wearing shirts with the same letters, it may predict "College students in a fraternity studying."

Age and gender classification is an inherently challenging problem though, more so than many other tasks in computer vision. The main reason for this discrepancy in difficulty lies in the nature of the data that is needed to train these types of systems. While general object classification tasks can often have access to hundreds of thousands, or even millions, of images for training, datasets with age and/or gender labels are considerably smaller in size, typically numbering in the thousands or, at best, tens of thousands. The reason for this is that in order to have labels for such images we need access to the personal information of the subjects in the images. Namely we would need their date of birth and gender, and particularly the date of birth is a rarely released piece of information. Therefore, we must make do with the nature of this problem we are approaching and tailor network architectures and algorithmic approaches to cope with these limitations. These reasons are the primary motivation behind [12] choosing to implement a relatively shallow architecture for age and gender classification using convolutional neural networks, and we have followed this pattern.

The input to my algorithm is an image of a human face of size 256x256 that is then cropped to 227x227 and fed into either the age classifier, gender classifier or both. The age classifier returns a integer representing the age range of the individual. There are 8 possible age ranges (see Section 4), so the age classifier returns an integer between 0 and 7. The gender classifier returns a binary result where 0 indicates male and 1 represents female.

2. Related Work

The areas of age and gender classification have been studied for decades. Various different approaches have been taken over the years to tackle this problem, with varying levels of success. Some of the recent age classification approaches are surveyed in detail in [3]. Very early attempts [10] focused on the identification of manually tuned facial features and used differences in these features' dimensions and ratios as signs of varying age. The features of interest in this approach included the size of the eyes, mouth, ears, and the distances between them. While some early attempts have shown fairly high accuracies on constrained input images (near ideal lighting, angle, and visibility), few [2] have attempted to address the difficulties that arise from real-world variations in picture quality/clarity.

A holistic overview of methods applied to gender classification can be found in [14]. As early as 1990, neural networks were considered for the purposes of gender classification in [4] (which has perhaps one of the most interesting paper names in all of computer vision). Later on in the early 2000s, [16] used support vector machines (SVMs) and found they could achieve very low error rates on gender prediction of "thumbnail images" of subjects which were of very low resolution. Yet again though, none of these attempts seemed to acknowledge that the constrained settings of their training and test data hindered their systems from achieving equally impressive performance numbers in real-world applications where images can be subject to altered lighting, tilt, focus, occlusion, etc.

Virtually all of these papers, and their corresponding methods, tackle either age classification (in some cases regression) or gender classification, but usually not both. But in 2015, [12] broke this norm by developing one methodology and architecture to address both age and gender. Furthermore, the authors address the undeniable reality that images taken in real-world settings are not perfectly aligned, lit, or centered. To that end, they train on images from a wide range of angles, lighting conditions, etc., and they oversample the input images to the classifier to consider various regions in the image for classification.

Their focus on using deep convolutional neural networks (CNNs), follows a pattern in the computer vision community as CNNs are shown more and more to provide unparalleled performance for other types of image classification. The first application of CNNs was the LeNet-5, as described in [11]. However deeper architectures in the early 1990s were infeasible due to the state of hardware performance and cost. In recent years, with the dawn of never-before-seen fast and cheap compute, [9] revived the interest in CNNs showing that deep architectures are now both feasible and effective, and [19] continued to increase the depth of such networks to show even better performance. Therefore the authors of [12] leveraged these advances to build a powerful network that showed state-of-the-art performance. They advocate for a relatively shallow network, however, in order to prevent over-fitting the relatively small dataset they were operating on. Deeper networks, although generally more expressive, also have a greater tendency to fit noise in the data. So while [19] shows improved performance with deeper architectures training on millions of images, [12] shows improvements for shallower architectures for their use case.

I aim to show that I can build off of this prior work, particularly the work in [12], to develop a system that leverages the inherent inter-relationships between age and gender to link these architectures in such a way as to improve overall performance.

3. Methods

3.1. Network Architecture

The network architecture used throughout my project is based off of the work in [12]. As mentioned toward the end of Section 2, this network design is intended to be relatively shallow so as to prevent over-fitting the data. Figure 1 visualizes the network, which is explained below.

An RGB image being input to the network is first scaled to 3x256x256 and then cropped to 3x227x227. The types of cropping are described further in Section 3.2. There are 3 convolution layers, followed by 3 fully connected layers. The convolution layers are

1. Conv1- 96 filters of size 3x7x7 are convolved with stride 4 and padding 0, resulting in an output volume size of 96x56x56. This is followed by a ReLU, max-pooling pooling which reduces the size to 96x28x28, and a local-response normalization (LRN).

2. Conv2- 256 filters of size 96x5x5 are convolved with stride 1 and padding 2, resulting in an output volume size of 256x28x28. This is also followed by a

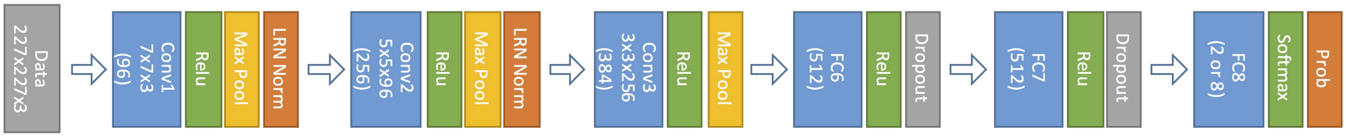

Figure 1. **Network architecture**, as described in Section 3.1

ReLU, max-pool, and LRN, reducing the output size to 256x14x14.

3. Conv3- 256 filters of size 256x3x3 are convolved with stride 1 and padding 1, followed by a ReLU and max-pool, resulting in an output volume of 256x7x7.

The fully connected layers are

1. FC6- 512 neurons fully connected to the 256x7x7 output of Conv3, followed by a ReLU layer and dropout layer.

2. FC7- 512 neurons fully connected to the 1x512 output of FC6 followed by a ReLU layer and dropout layer.

3. FC8- 2 or 8 neurons fully connected to the 1x512 output of FC7, yielding the un-normalized class scores for either gender or age, respectively.

And finally there is a softmax layer that sits on top of FC8, which gives the loss and final class probabilities.

3.2. Training and Testing

The authors of [12] split the images (as described in Section 4) into 5 folds and then perform a subject-exclusive cross-validation protocol, as first developed by [2]. The reason this type of protocol is necessary is because of the nature of the dataset being used, which contains multiple pictures of the same subjects. Therefore if the images were simply randomly shuffled and divided into fifths, the same subjects could potentially appear in both the training and test folds, thereby skewing the results to seem more promising than they are in reality. This protocol therefore ensures that all the images of a given subject appear in a single fold to avoid this issue.

For my project, I divided the dataset into 6 subject-exclusive folds, but then further divided each of those folds into males, females, and each of the 8 age groups. As described in Section 3.3, this was necessary for the types of classifiers I was aiming to build. This resulted in a total of 66 “sub-folds”, where each of the original 6 folds were broken up into 11 groups, based on the types of classifiers I would be training. All of the sub-folds coming from the 6th original fold were separated as test data, never to be trained on or validated against. Then the remaining 5 folds, and their sub-folds, were used for training, and cross-validation. So if, for example, at a given time I was training a male age classifier, I would use 4 of the male age sub-folds as the training set and the 5th male age sub-fold as the validation set. This would rotate between the first 5 sub-folds for every possible assignment of the validation fold, and the resulting classifier would be tested against the 6th (previously separated) male age sub-fold.

This altered fold distinction I implemented prevented me from being able to use the pretrained weights provided by [12] because by construction their training folds could overlap with my test folds. At first I did not see this issue and then initial experiments showed uncharacteristically high accuracies that led to further investigation and the discovery of this inherent problem.

Finally, [12] proposes 2 types of sampling of an input image when being classified. One is to simply take a center crop of 227x227 out of the 256x256 image and classify that. The other is to take 5 such crops, one from each of the corners and one from the center, classify them all, and take the majority classification from between them. While they found that the latter technique can improve accuracy slightly, for the sake of reducing testing time, I used the first approach for this project.

3.3. Goals

My first objective in this project was to determine if the proposed network architecture (see Section 3.1) was indeed optimal. Although the authors of [12] claimed that any deeper network would suffer from over-fitting, I wanted to verify this for myself. To this end I experimented with adding additional convolution layers, removing fully connected layers (in the style of [19]), and modifying the parameters used for dropout as well as LRN.

The primary goal, however, was to experiment with a new higher-level approach for composing these classifiers to improve performance. The observation I made early on was that gender classification is an inherently easier task than age classification, both due to the fewer number of classes to distinguish between and the more marked differences that exist between genders than between many age groups. This then led me to the conclusion that while it is reasonable to assume one should be able to ascertain someones gender apart from knowing their age, or vice versa, there is also some plausibility of using one of these attributes to better inform the prediction of the other. For example, the amount of hair on a man’s head can often be a useful indicator of age, but the same is not true for women.

Furthermore, separating the tasks of classifying men's age and women's age should, in theory, give the networks more expressive power by freeing them from having to learn a gender-neutral concept of age. Therefore, I proposed that training separate age classifiers for men and women could simulate the added power of deeper networks while avoiding the danger of over-fitting.

3.4. Technical Details

In this section, I elaborate on some of the technical details of the network architecture and how it is trained.

Local Response Normalization (LRN). After the first 2 pooling layers, there are local response normalization (LRN) layers. LRN is a technique that was first introduced in [9] as a way to help the generalization of deep CNNs. The idea behind it is to introduce lateral inhibition between the various filters in a given convolution by making them "compete" for large activations over a given segment of their input. Effectively this prevents repeated recording of the same information in slightly different forms between various kernels looking at the same input area and instead encourages fewer, more prominent, activations in some for a given area. If $a^i_{x,y}$ is the activation of a neuron by applying kernel i at position (x, y), then it's local response normalized activation $b^i_{x,y}$ is given by

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

where k,n,α, and β are all hyper-parameters. The parameter n is the number of "adjacent" kernel maps (filters) over which the LRN is run, and N is the total number of kernels in that given layer. The values used for these hyperparameters are the same as those used in [9].

Softmax. At the top of the proposed architecture lies a softmax layer, which computes the loss term that is optimized during training and also the class probabilities during a classification. While some loss layers like multiclass SVM loss treat the output of the final fully connected layer as the class scores, softmax (also known as multinomial logistic regression) treats these scores as the unnormalized log probabilities of the classes. That is, if we have z_i is the score assigned to class i after the final fully connected layer, then the softmax function is

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Because we want to maximize the log likelihood of the correct class, the term we want to minimize is the negative log likelihood.

$$L_i = -log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}})$$

Because the softmax function takes real-valued scores being output from f and normalizes them by their exponentiated sum, it guarantees that the sum of all softmax scores is 1, thereby allowing it to be interpreted as a true class probability. It should be noted that the softmax loss is actually a particular form of a cross-entropy loss. More specifically, the cross-entropy between an actual distribution p and an approximate distribution q is defined as

$$H(p, q) = -\sum_x p(x) log q(x)$$

From this it can be seen that the softmax classifier is really just minimizing the cross-entropy between the estimated class probabilities and the real distribution, which would look like 1 predicted for the actual class and 0 predicted for everything else.

Stochastic Gradient Descent. Now that we know how to calculate the loss, we need to know how to minimize it in order to train an accurate classifier. The type of optimization used in this experiment is stochastic gradient descent. In order to explain this, first I will elaborate on the more generalized form of gradient descent. The gradient of a function is really just its derivative, and therefore by definition it is the direction of steepest ascent (or descent if you move backwards along it). Therefore if we compute the gradient of the loss function with respect to all of the system variables/weights (in CNNs there can be up to millions of these), we will have the direction along which we can move toward our minimum loss most quickly by following the negative of the gradient. Each time we compute the gradient we take a small step (governed by a hyperparameter) in the opposite direction, and we re-evaluate the loss, re-compute the gradient, and repeat. The hope (and in fact the reality) is that by repeating this process we will iteratively decrease our loss function, which is reflective of the model becoming iteratively better at its classification task. Mathematically, we can write this as

$$\mathbf{w} = \mathbf{w} - \eta \nabla_{\mathbf{w}} L$$

where η is the learning rate, also sometimes called the step size and $\nabla_{\mathbf{w}} L$ is the gradient of the loss term with respect to the weight vector $\mathbf{w}$.

While this is theoretically great, the truth is that computing the gradient across the entire training set in order to make an incremental update to the weights is prohibitively computationally expensive. Therefore alternate approaches have been invented that evaluate the gradient of the loss function over a sample of the training data, and use that approximate gradient to make the update. The reason this gradient is approximate is that although it is the optimal direction to travel down given the sample of images it was computed over, there is no telling what kinds of images it did not look at when computing the gradient. Therefore making this form of mini-batch gradient descent, as it is called, will still usually reach the minimum loss over time (or at least a local minimum), but it will require many more iterations on average. However the time it takes to evaluate the gradient drops so dramatically when we operate on a mini-batch that it is actually significantly faster to perform many more mini-batch gradient updates than a few full gradient updates. Finally, stochastic gradient descent is a special form of gradient descent in which the mini-batch size is 1. This is extremely fast to compute since it only requires passing 1 image forward (to calculate the loss) and backward (to calculate the gradient) through the network, but the gradients are even less globally optimal than mini-batch gradient descent, therefore a smaller step size is required at each iteration and many more iterations are required.

4. Dataset

The dataset used for training and testing for this project is the Adience face dataset, which comes from the Face Image Project[12] from the Open University of Israel (OUI). This dataset contains a total of 26,580 photos of 2,284 unique subjects that are collected from Flickr. Each image is annotated with the person's gender and age-range (out of 8 possible ranges). The images are subject to various levels of occlusion, lighting, and blur, which reflects real-world circumstances. I used those images which were mostly front facing, which limited the total number of images to around 20,000. Table 1 includes details regarding the distribution of images in each gender and age range. Figure 2 shows some examples of images of both males and females in the dataset of various ages. The images were originally of size 768x768, so they were preprocessed by all being resized down to 256x256.

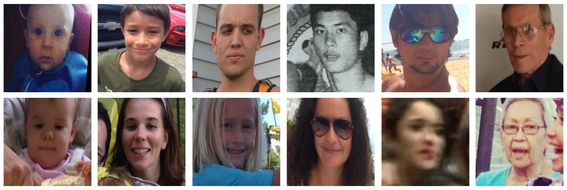

Figure 2. **Adience image dataset examples**. Top row: 6 males of various ages. Bottom row: 6 females of various ages.

5. Experiments

The training and testing for this project were done exclusively using Caffe [6], running on Amazon EC2 using between 1 and 3 instances at a time, each with 1,536 CUDA cores and 4GB of video memory. As described in Section 3.2, a 6-fold subject-exclusive cross validation protocol was used to provide added robustness to the training, while preserving the reliability of results. Although much of my network architecture was built off of the work in [12], my networks were trained from scratch without using any pretrained weights provided by that, or any other, project. The reason for this had to do with the way that I divided up the dataset for the purposes of my project (again, see Section 3.2 for more information).

	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60+	Total
Male	745	928	934	734	2308	1294	392	442	8192
Female	682	1234	1360	919	2589	1056	433	427	9411
Both	1427	2162	2294	1653	4897	2350	825	869	19487

Table 1. **Adience image dataset distribution**. Number of images for each gender and age range.

The first step I took was to attempt to reproduce the results of [12] as a baseline since I had their network architecture and training data. I attempted to replicate their experiment as closely as possible, so I also used SGD with a learning rate of 1e-3 that decays by a factor of 10 every 10,000 iterations and a batch size of 50 images. This proved quickly successful, and within a few hours of training I reached accuracies that were very close to their results for both age and gender classification. These results are recorded in Table 2. Note that their slightly higher accuracies are likely due to the oversampling they do of the input images followed by taking the majority classification of the various samples. For the sake of faster iteration in my model, I avoided this technique.

	Age	Gender
Benchmark [12]	50.7	85.9
My Baseline	**50.2**	**80.8**

Table 2. **Initial benchmark.** Classification accuracies achieved in my benchmark paper [12] as compared to my initial numbers after trying to reproduce their results.

Next, I wondered if their model could be improved upon. Namely, after reading that the Adam learning algorithm[7] can often provide improved convergence times as compared with SGD, I wondered if I could use Adam to decrease training time and increase (or at least match) performance levels. I began testing various ranges of the additional hyperparameters needed for Adam (in Caffe these are 'momentum2' and 'delta'), but after a couple days I was unable to find a setting that even matched the performance of SGD. So for the sake of time I moved on to try other potential forms of improvement.

After reading [19], it became clear that large fully connected layers at the end of a network do not necessarily contribute much to the overall performance and that depth in the convolutional layers is actually preferable. To this end I removed 1 and then 2 of the fully connected layers (out of 3) and added 1 and then 2 additional convolution layers (on top of the existing 3). I attempted 5 different combinations of these modified architectures, but like with the attempt at using Adam, after multiple days there was no clear benefit, and if anything added complexity was making the system perform the same or, sometimes, worse.

At this point I chose to focus my efforts on the main insight that motivated this project, which was that coupling the architectures for gender and age classification could give more expressive power to the classifiers, particularly for age. As a sort of "proof of concept", I attempted to train classifiers on each gender separately to see what would happen. The results pleasantly surprised me and are summarized in Figure 3. I saw that when training a classifier from the ground up only on male images, the accuracy when predicting the age of men increased. Conversely, I saw the accuracy of classifying women's ages decreases over the average (which may or may not be taken as a social commentary on how women are more effective at hiding their age).

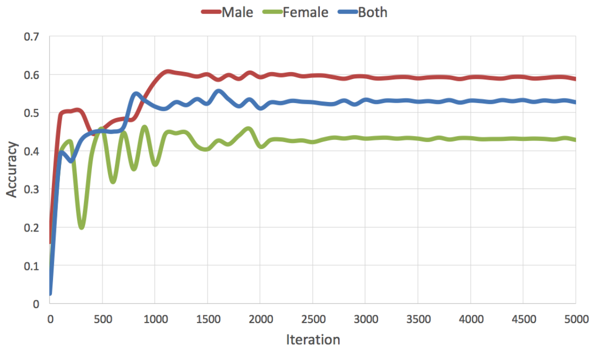

Figure 3. **Gender-specific age classification.** Accuracies observed for age classifiers trained on just men, just women, or both.

I attempted a similar approach of training separate classifiers for each age group, and those results are summarized in Table 3. These results are less striking than those of Figure 3, but there is some reassurance in how intuition lines up with the results. Namely it can be seen that the age range in which it is most difficult to predict gender, with just a 27% accuracy, is 0-2 years old. Of course though that makes perfect sense as gender-specific features are not usually present at such a young age, or at least not as much as later in life. Also the age range in which gender prediction is the best is 15-20, which also seems reasonable since that is the time when there is the most development of gender-specific features.

Age	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60+	All
Accuracy	**0.27**	0.76	0.76	**0.92**	0.78	**0.87**	0.79	0.76	0.79

Table 3. **Age-specific gender classification.** Accuracies observed for gender classifiers trained only on images in a certain age range.

Given these results, it seemed most promising to use the remaining time I had to develop and train a chained gender-age network that would first classify gender (as before), and then based on the gender classification feed the image into an age classifier trained solely on men or solely on women. It should be emphasized again that all of these networks were trained from randomly initialized weights as opposed to using any pretrained models. The measurement of interest was the final test accuracies from the 6-fold cross-validation protocol. As in [12], gender accuracies are given 2 forms, exact and 1-off. Exact accuracies are the percent of images that are classified into the correct age range (traditional definition of accuracy). The 1-off accuracies allow for a deviation of at most one bucket from the actual age range. This is used because practically there is very little to no difference in facial features between people at the top of one age range and those at the bottom of the next. Therefore this 1-off measure provides flexibility in assigning images to buckets that are "pretty close" to the actual age range.

The results of this chained net are summarized below in Table 4, and I was pleased to find that, as proposed, this chained structure allows for increased classification accuracy over the traditional approach of training age classifiers on both genders at once. Table 5 is the confusion matrix for this final chained net.

	Exact	1-off
My Baseline	50.2	78.4
Chained Net	**54.5**	**84.1**

Table 4. **Chained net accuracies.** Age prediction accuracies achieved by my chained gender-age classification model, as compared to my baseline I trained while trying to reproduce the results of [12] (see Table 2).

	0-2	4-6	8-13	15-20	25-32	38-43	48-53	60+
0-2	**0.713**	0.187	0.081	0.004	0.006	0.003	0.002	0.002
4-6	0.234	**0.473**	0.205	0.060	0.015	0.007	0.003	0.002
8-13	0.041	0.181	**0.506**	0.101	0.121	0.045	0.001	0.002
15-20	0.015	0.019	0.145	**0.308**	0.410	0.086	0.009	0.006
25-32	0.011	0.015	0.078	0.123	**0.625**	0.112	0.022	0.013
38-43	0.009	0.011	0.055	0.035	0.116	**0.306**	0.404	0.059
48-53	0.006	0.008	0.042	0.049	0.160	0.321	**0.190**	0.223
60+	0.010	0.008	0.039	0.033	0.126	0.224	0.195	**0.363**

Table 5. **Chained net confusion matrix.** Confusion matrix for age classification using my final chained gender-age model.

6. Conclusion

Although many previous methods have tackled the problem of age and gender classification of images, in this paper

I establish a benchmark for the task based on state-of-the-art network architectures and show that chaining the prediction of age with that of gender can improve overall accuracy. If there had been more time, I would have dedicated more effort towards fine-tuning the parameters and the modified architectures I experimented with. Specifically, I would have liked to get the Adam learning algorithm in place with equal or improved performance to SGD, and I would have liked to replace the multiple fully connected layers at the end of the architecture with only one and instead shifted those parameters over to additional convolutional layers.

By far the most difficult portion of this project was setting up the training infrastructure to properly divide the data into folds, train each classifier, cross-validate, and combine the resulting classifiers into a test-ready classifier. I foresee future directions building off of this work to include using gender and age classification to aid face recognition, improve experiences with photos on social media, and much more. Finally I hope that additional training data will become available with time for the task of age and gender classification which will allow successful techniques from other types of classification with huge datasets to be applied to this area as well.

References

[1] G. Antipov, S.-A. Berrani, and J.-L. Dugelay. Minimalistic cnn-based ensemble model for gender prediction from face images. *Pattern Recognition Letters*, 70:59 – 65, 2016.

[2] E. Eidinger, R. Enbar, and T. Hassner. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170–2179, Dec 2014.

[3] Y. Fu, G. Guo, and T. S. Huang. Age synthesis and estimation via faces: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11):1955–1976, Nov 2010.

[4] B. A. Golomb, D. T. Lawrence, and T. J. Sejnowski. Sexnet: A neural network identifies sex from human faces. In *Proceedings of the 1990 Conference on Advances in Neural Information Processing Systems 3*, NIPS-3, pages 572–577, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

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

[6] 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 arXiv:1408.5093*, 2014.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

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

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1106–1114. 2012.

[10] Y. H. Kwon and N. da Vitoria Lobo. Age classification from facial images. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 762–767, Jun 1994.

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

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

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

[14] E. Makinen and R. Raisamo. Evaluation of gender classification methods with automatically detected and aligned faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):541–547, March 2008.

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

[16] B. Moghaddam and M.-H. Yang. Learning gender with support faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):707–711, May 2002.

[17] D. T. Nguyen, S. R. Cho, T. D. Pham, and K. R. Park. Human age estimation method robust to camera sensor and/or face movement. *Sensors*, 15(9):21898, 2015.

[18] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.