Facial Age Identification with CNN

Nicolas Carmont Zaragoza
Department of Cognitive Science
UC San Diego
La Jolla, CA 92093
ncarmont@ucsd.edu

Valentin Bielecki
Department of Cognitive Science
UC San Diego
La Jolla, CA 92093
ax006252@acsmail.ucsd.edu

Abstract

Automatic age classification has become relevant to an increasing amount of applications, particularly since the rise of social media platforms. This paper introduces an adapted CNN for age estimation applications. Our focus is specifically on the optimization of hyperparameters and convolutional layers to achieve highest accuracy on the UTKFace Database (+20K images). We compared our CNN performance with the CaffeNet CNN. Furthermore, we augmented this dataset using the HaarCascade algorithm to crop relevant faces from UTKFace Database. At a higher level the CNN consists of extracting significant facial features from these cropped face images and estimates the age range of a person.

1 Introduction

Research from UNICEF [1] 2017 reveals over 29% of people in the world do not have access to official birth registration. Many of this demographic come from rural areas and including young refugees who have escaped wars. The primary motivation of this investigation was to create a model which can provide an indication as to the approximate age boundaries for these individuals. We looked at the results on age estimation using CNN [2] and found out that the Mean Absolute Error (MAE) for most state-of-the-art algorithms is around 5 years, going down to 3 years for the best estimators.

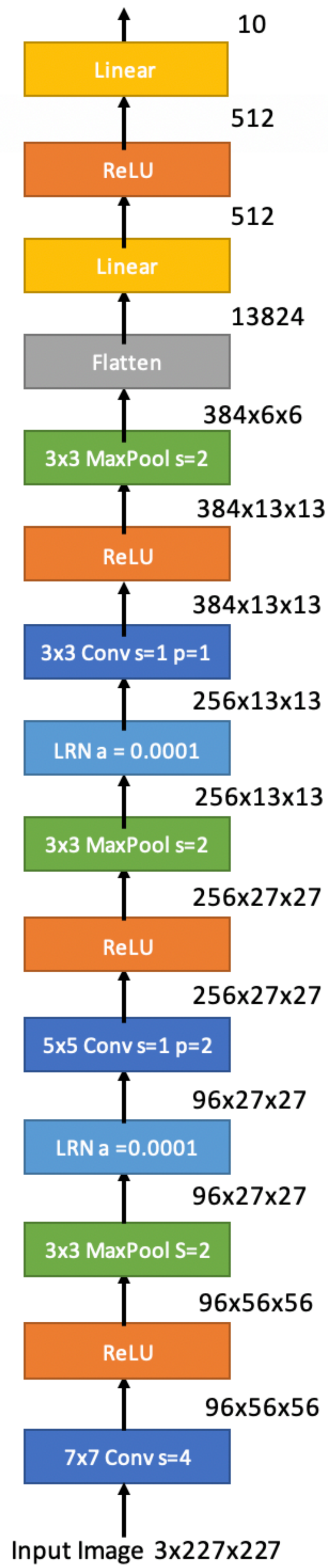
Since the CNN would use HD front-facing pictures, we required a preprocessed dataset of faces with few defects and that would amplify clear facial features for age estimation.

2 Caffe CNN

2.1 Overview

The CaffeNet is a popular CNN architecture created by Berkeley AI Research [10]. It is acknowledged for being a 1-GPU version of AlexNet and particularly strong for small datasets like our utilized UTK Facial image dataset (+20K images).

The CaffeNet CNN Model consists of 3 hidden layers. It uses the ReLu activation function and Local Response Normalization (LRN). The LRN creates a local maximum [7], a local peak which creates contrast within an area and increases its visual perception. Age prediction requires this visual contrast to increase to detect more subtle features like wrinkles.


10
Linear
512
ReLU
512
Linear
13824
Flatten
384x6x6
3x3 MaxPool s=2
384x13x13
ReLU
384x13x13
3x3 Conv s=1 p=1
256x13x13
LRN a = 0.0001
256x13x13
3x3 MaxPool s=2
256x27x27
ReLU
256x27x27
5x5 Conv s=1 p=2
96x27x27
LRN a =0.0001
96x27x27
3x3 MaxPool S=2
96x56x56
ReLU
96x56x56
7x7 Conv s=4
Input Image 3x227x227

2.2 Tuning Hyperparameters

We chose to tune the hyperparameters and modify the network structure of CaffeNet to create a stronger default structure. Adaptations to the architecture included changing the layer design, analyzing different optimizers, varying batch size, and determining the optimal activation and pooling functions.

For training the model a batch size of 5 images was used for moderate speed and high accuracy. This was executed for 5 epochs using our custom dataset and only changing one parameter at a time. 5 age ranges were used for a more comprehensive classification:

(0,14), (15,29), (30,44), (45,59), (60,74)

These results of these modifications are described below.

2.2.1 Number of layers

CNN1: Removing one Layer

Only the last hidden layer of the CaffeNet was removed as this meant the overall architecture was less impacted compared to lower layers and hence easier for comparison.

The second convolutional layer was modified to have 384 output channels and a stride of 2 pixels, this was so the amount of network parameters remained constant.

The intention was to change the CNN as least as possible to isolate the effect of distinct layer amounts and designs.

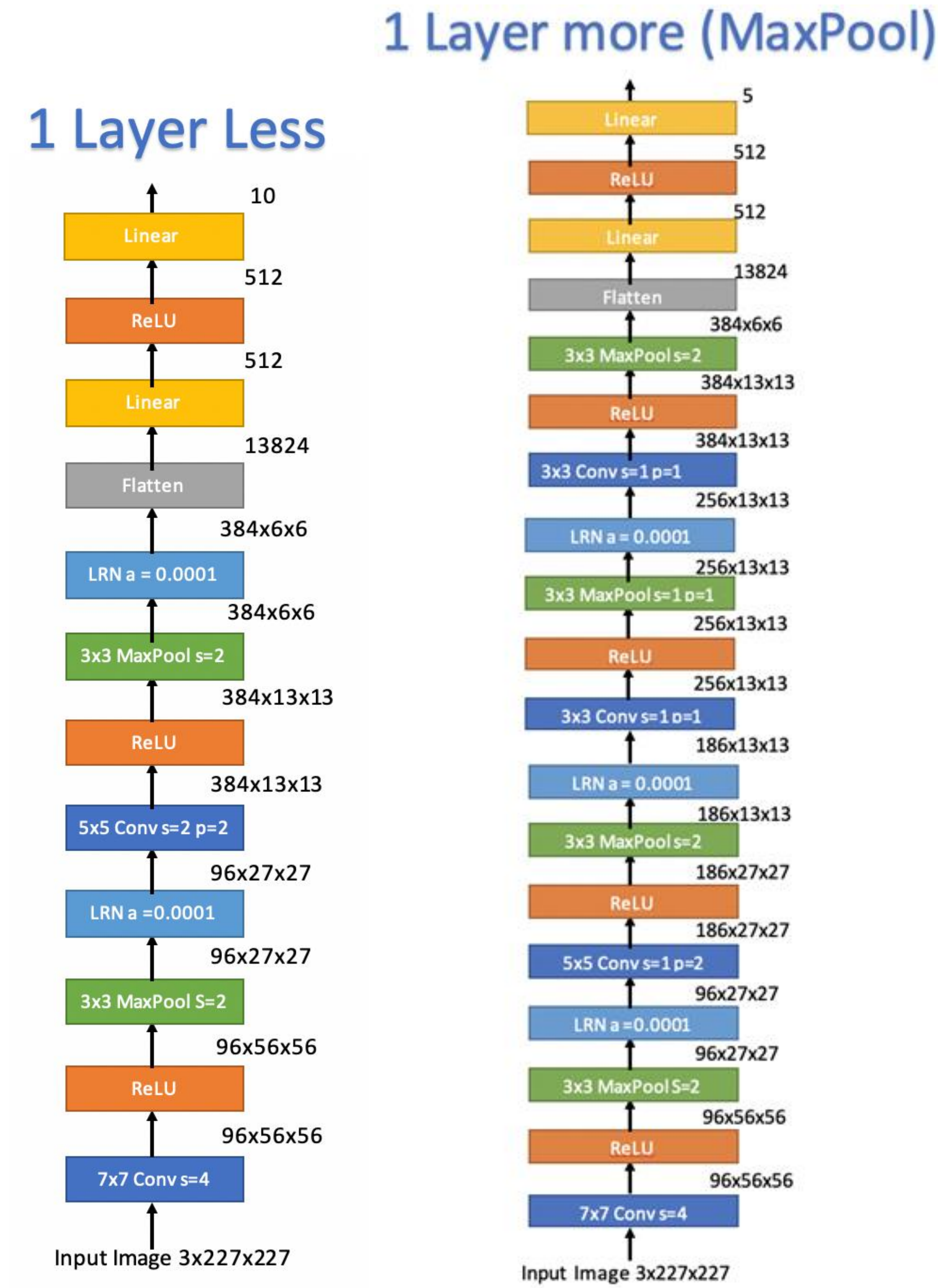
CNN2: Additional layer structure with Max Pooling

An additional last layer structure was added to the CaffeNet which mimicked the original design of CaffeNet's layer design. This structure consists of a convolutional layer with 3x3 convolutional layer, a followed by a ReLu activation function, a Max Pooling with 3x3 kernel size and 2-pixel stride and lastly a Local Response Normalization function.

CNN3: Additional layer structure with Average Pooling

A similar layer structure to CNN2 was also attempted, however, using an Average Pooling function instead of a Max Pooling function. This was to slightly alter the design of the layer and test the effect of a single distinct pooling function within the CaffeNet layers.

The following results were achieved with the modified layer CNNs:



The following results were achieved with the modified layer CNNs:

Number of Layers	Caffe Net	CNN2 Additional layer structure with MaxPool	CNN3 1 Additional layer structure with AvgPool	CNN1 (Layer Removed)
Final Average mini-batch loss	1.114	1.143	1.302	1.064
Accuracy on test	53%	50%	45%	52%

The original CaffeNet structure performed most accurately out of all tested layer models.

Whilst the final average mini-batch error of removed layer CNN1 (1.064) was lower than that of the original CaffeNet (1.114), it still showed slightly lower accuracy. This may be explained by fluctuations within the learning process as both accuracies seem quite similar whilst having same parameter numbers. The lower performance of CNN2 suggests that adding an additional layer structure may create excessive processing of small features as backed by the higher mini-batch loss. Finally, using an average pooling layer showed significantly lower accuracy (45%) which may be due to the skewed average created by the contrasting LRN normalizing layers.

2.2.2 Batch Size

Batch Size	1	2	3	4	5	6	7	8	9	10
Final Average mini-batch loss	1.091	1.047	0.976	1.083	1.114	1.165	1.140	1.208	1.257	1.228
Accuracy on test	50%	58%	55%	56%	53%	51%	51%	47%	45%	48%

Varying batch sizes between 1 to 10 images per batch showed to have a negative trend with respect to performance. Lower batch sizes, with exception of a batch of 1 image typically led to an increase in performance. This can be explained by the fact that larger batch size update gradients after every batch rather than every entry. However, with very distinct facial images and with the use of an SGD optimizer, a reduced batch size may mean more gradients are computed especially in reduce amounts of epochs and individual faces have a larger impact on the model.

2.2.3 Activation function

Activation Function	ReLU	ELU	Sigmoid	LeakyReLU	Hard Shrink	Relu6	RReLU	Tanh
Final Average mini-batch loss	1.114	0.903	1.518	1.093	1.501	1.056	0.970	0.862
Accuracy on test (3 695 test images)	53%	65%	37%	55%	37%	55%	57%%	<u>66%</u>

The activation function tanh produces large gradients. It is better for reaching minima faster. This is good with a medium size dataset like ours.

2.3 Optimization function

The pytorch documentation describes the use of various optimization methods. We chose to test 7 of the optimization functions implemented within the pytorch.optim module. The default activation function is ReLu.

<u>Optimizers Performance:</u>

Optimizers	SGD	Adam	Adagrad	Adadelta	Rprop	RMSprop	ASGD
Final Average mini-batch loss	1.114	1.350	1.080	1.362	8.777	1.178	1.462
Accuracy on test (3 695 test images)	53%	37%	<u>55%</u>	46%	37%	51%	35%

Adagrad has the most potential among all these optimization function.

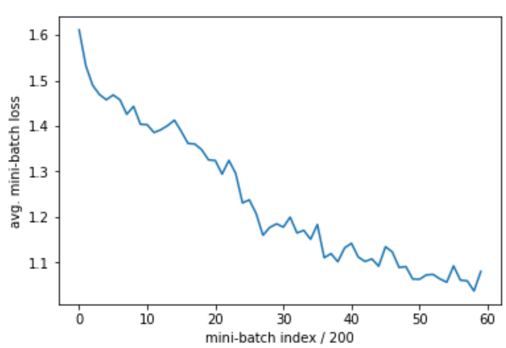


Figure 1 Adagrad average loss

Its main characteristic would be that it adapts the learning rate to the parameters. Thus, it's performing smaller updates for parameters associated with frequently occurring features, and larger updates for parameters associated with infrequent features [9]. SGD updates all parameters using the same learning rate, whereas Adagrad modifies the general learning rate based on the past gradients that have been computed. This optimization function is well-suited for dealing with sparse data. In Figure 1, we can say that the training has a good learning rate.

2.4 AveragePool VS MaxPool

Pooling Type Performance:

Pooling Type	MaxPool	AveragePool
Final Average mini-batch loss	1.114	1.244
Accuracy on test	53%	46%

Average Pooling showed to have significantly lower accuracy than Max Pooling. This may be explained by the fact that facial age estimation features tend to be of higher contrast and be less noisy. Whilst Average Pooling helps smoothen noise in extracted features, this may not be as optimal for our task as Max Pooling may instead allow extraction of high contrast facial age features such as wrinkles. Furthermore, the use of LRN normalization may already smoothen the feature images and amplify the extraction of significant features by max pooling.

2.5 Optimized CaffeNetCNN

Using the tuning test results, the Optimized CaffeNet CNN would use the following:

Optimizer: **Adagrad** (1.080 average mini-batch loss, 55% accuracy)

Batch Size: **2** (1.047 average mini-batch loss, 58% accuracy)

Pooling Type: **MaxPool** (1.114 average mini-batch loss, 53% accuracy)

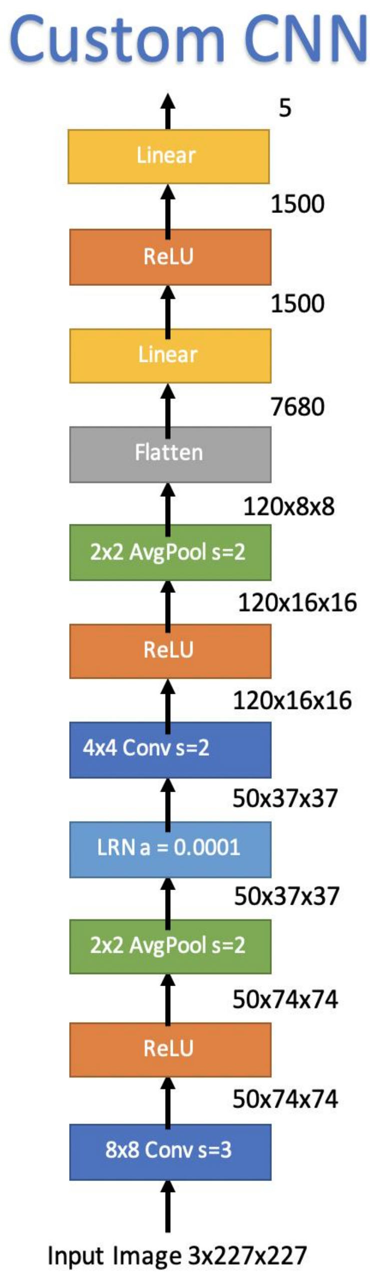
Layer Amount: **Normal CaffeNet** (1.114 average mini-batch loss, 53% accuracy)

Activation Function: **Tanh** (0.862 average mini-batch loss, 66% accuracy)

3 Custom CNN Model

A custom CNN model was also developed for comparison with the baseline CafeNet performance. The custom CNN consisted of two convolutional layers each followed by a ReLU activation function and an Average Pooling function. Finally, a fully connected layer is applied, followed by a ReLU activation layers and another fully connected layer.

The rationale behind this structure was to provide a basis for feature extraction using medium sized convolutional layers (50 and 120 output layers). The activation average pooling functions allow the smoothening of these features and finally the fully connected layers with lower overall input parameters (7680) allow for potentially lower overfitting compared to the CaffeNet.
This structure is shown below:



4 Model Comparison

Attempting to combine the optimal parameters conjunctively in the optimized CaffeNet model and the Custom CNN surprisingly led to lower anticipated performance than tuning each parameter individually. This suggests there is a large interdependence between the default parameters used for optimization, activation functions and batch size and those found to be optimal under these conditions.

For example, the stochastic gradient descent optimizer is moderately dependent on the batch size, hence a lower batch size generally led to higher accuracy, with a batch size of 2 being optimal. However, the Adagrad optimizer, an adaptive gradient learning optimizer is less dependent on batch size as its learning rate progressively lowers, whereas Stochastic Gradient Descent does so according to batches.

Caffe vs. Custom (5 epoch-5 Classes)	Custom Model	CaffeNet Optim
Final average min-batch loss	1.169	1.045
Accuracy on test	53%	<u>58%</u>

Furthermore, to determine true accuracy on more realistic age ranges, the age classes were expanded to the following age ranges:

(0,3), (4,7), (8,13), (14,19), (20,25), (26,36), (37,47), (48,58), (59,79), (80,100)

Furthermore, the epoch was enlarged to achieve higher accuracy and verify the results under more extended training.

Caffe vs. Custom (10 epoch,10 classes)	Custom Model	CaffeNet Optim
Final Average mini-batch loss	1.514	1.203
Accuracy on test	45%	<u>52%</u>

Due to the dependence of the optimal batch size (2) under SGD and the best optimizer for batch size 5, Adagrad. The highest performance parameter was used under SGD with batch size 5 which was the tanh activation function.

Training the Custom CNN and optimized Caffe Net model using tanh led to results:

Caffe vs. Custom (10 epochs-10 classes-tanh-bathch5-SGD)	Custom Model	CaffeNet Optim	CaffeNet
Final Average mini-batch loss	1.337	0.913	1.207
Accuracy on test	50%	66%	53%

As observed the tanh activation function seemed to greatly increase the accuracy of both the Custom CNN and the optimized CaffeNet model. With the Custom CNN (50%) showing near accuracy to the original CaffeNet (53%) and our optimized CaffeNet far surpassing both (66%).

The impact of tanh, the hyperbolic tangent function as an activation function in comparison to other like ReLu is that it allows a zero-centered output and due to larger gradients can help reach local minima faster.

Nonetheless, in practice tanh may lead to larger computation complexity and gradient vanishing for the model during training and hence may not be desired. However, for small datasets and low epochs it appears a switch to tanh may be desirable.

5 Dataset

5.1 UTKFace

We used the UTKFace Dataset of 24 108 images [5] to create a custom dataset of 19 225 faces. The images are subject to occlusion, blur, reflecting real-world circumstances. This dataset is labeled with 3 relevant features: age, gender and race. It is formatted this way:

[age]_[gender]_[race]_[date&time].jpg

[age] is an integer from 0 to 116, indicating the age

[gender] is either 0 (male) or 1 (female)

[race] is an integer from 0 to 4, denoting White, Black, Asian, Indian, and Others

[date&time] format of YearMonthDayTime, showing the date and time an image was collected

5.2 Custom Dataset

We used the HaarCascade algorithm [8] to detect the face in the images. We centered the face, then resized it to 224x224, and saved the cropped image in grayscale in a new folder. The images are labelled with the same format. We manually deleted all the mistakes for the HaarCascade and changed the algorithm [6] to only save the biggest face found.

5.3 Improvement (Bonus)

We customized the algorithm so that only the biggest face of the image would be saved. This allowed the custom dataset to have less random background object. The hair curves and the real-world circumstances made the HaarCascade algorithm detect faces on the actual person hair or didn’t crop it properly. With our improvement, it reduced significantly the number of deleted images. Our custom dataset has less defects and can be used for classification. We kept the features from the original dataset; hence anyone could also work on gender or race classification and improve its network accuracy with filtered cropped face images.

Original Image UTKFace   Autocrop original   Autocrop improvement

6 Application

6.1 Unregistered

We could use our CNN on pictures of unregistered people. As we mentioned in the introduction, we could help the 29% of people in the world without birth certificates finding their approximated age. Nonetheless the CNN we built does not have the best accuracy and compared to other structure, the MAE would be more than 5 years. There is room for improvement, and we don’t think it will be accurate enough for this purpose.

6.2 Dataset (Bonus)

The dataset created has more than 19K frontal faces and can be reused to gender or race classification. We think that this dataset has more potential than the initial UTKFace and that it could improve the accuracy of the CNN. Pictures with higher resolution would improve significantly the quality of the classification and might reveal more aging features.

6.3 Social Application (Bonus)

This CNN might have trouble estimating the age of a person, but it classifies any face with a good accuracy. We could imagine an application that would use this age range classification to make statistics on any place. This could be use in a store to keep track on our customers and do data-driven advertising. We could also use it in any restaurant or social place to know if you could meet people of your age range. We thought of an app that would show in real time the 'average age' of places such as bars or restaurants using their surveillance camera.

We tried to use our webcam to see if our face would be detected and if we would be classified correctly in real time (without putting ourselves in the training set!).
We made a video showing the live classification.

7 References

[1] "Birth Registration." *UNICEF DATA*, data.unicef.org/topic/child-protection/birth-registration/

[2] Raphael Angulu, et al. "Age Estimation via Face Images: a Survey." *EURASIP Journal on Image and Video Processing*, SpringerOpen, 6 June 2018, jivp-eurasipjournals.springeropen.com/articles/10.1186/s13640-018-0278-6?fbclid=IwAR2WRMEBwuaoATJyqNQeDAukcJIX-1obWBpmqNPaBXwLnJqKCs-XjllWPCk.

[3] "Age and Gender Classification Using Convolutional Neural Networks." *Tal Hassner*, 1 June 2015, talhassner.github.io/home/publication/2015_CVPR.

[4] GilLevi. "GilLevi/AgeGenderDeepLearning." *GitHub*, 30 June 2018, github.com/GilLevi/AgeGenderDeepLearning.

[5] *UTKFace*, susanqq.github.io/UTKFace/.

[6] Penseeartificielle. "Penseeartificielle/Reconnaissance-Faciale." *GitHub*, 14 May 2019, github.com/penseeartificielle/reconnaissance-faciale.

[7] "What Is Local Response Normalization In Convolutional Neural Networks." *PERPETUAL ENIGMA*, 7 Apr. 2016, prateekvjoshi.com/2016/04/05/what-is-local-response-normalization-in-convolutional-neural-networks/.

[8] "Face Detection Using Haar Cascades." *OpenCV*, docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html.

[9] Sebastian Ruder. "An Overview of Gradient Descent Optimization Algorithms." *Sebastian Ruder*, Sebastian Ruder, 29 Nov. 2018, ruder.io/optimizing-gradient-descent/index.html#adagrad.