A Deep Residual Convolutional Neural Network as Gender Classifier on an Interactive Webserver

Michele Pariani, Clemens Hutter, Jannik Steinmann, Jannis Born

Editor:

Course: **Implementing Artificial Neural Networks with Tensorflow**
Institute of Cognitive Science, University of Osnabrück, Osnabrück, Germany

Abstract

For this project we crawled pictures from ca. 400 celebrities from the Image Movie Database (IMDB). These images were face-cropped, resized and fed into a 9-layer residual convolutional neural network to learn cross-individual gender classification. We reach a performance accuracy on gender classification of 96.4% on the test dataset. On the frontend, we provide a webserver on which a user can verify the network's gender prediction of an arbitrary uploaded image. In case of wrong classification the user has the option to provide the correct label and retrain the network.

The code for this project is available at GitHub under https://github.com/dopexxx/FacePeeper

Contents

List of Figures

1. Task Description

Our original goal was to develop a deep convolutional neural network (CNN) with residual connections that is capable to recognize identities of 400 celebrities. The trained network was thought to be made publicly accessible on an interactive website where users could upload their own images of the celebrities and evaluate the network behavior. These images would firstly get face cropped and subsequently fed into the network to detect the identity. It was then thought to return the 5 most likely labels for that particular picture. When displaying this information to the user, the user was then thought to be able to retrain the network in case of misclassification, e.g. when the correct label *Bruce Willis* had only a probability of $p = 0.15$ while wrong the label *Jason Statham* had a higher value. The network then should be retrained with the correct label.
The dataset was crawled from the Image Movie Database (IMDB), preprocessed and augmented on the fly. Due to a) the complexity of this identity classification task and b) time constraints, we reduced the depth of the network and trained it on a cross-individual gender classification task using a subset of the original data. This gender classification residual CNN achieved a performance of 96.4% and can now be both tested and improved on an interactive frontend running on Amazon Web Services (AWS). Due to financial limitations, the frontend is currently not continuously accessible but could be switched on upon request on almost all times. In contrast to most projects for university courses in our field, the motivation for this project arose from the contributors' ideas to implement and solve a problem from the very scratch (data acquisition) up to a publicly accessible frontend.

2. Related Work and Similar Approaches

The concept of CNNs in combination with the rising computational power (deep learning) has achieved a lot of progress in the field of image classification in general (LeCun et al., 1998; Krizhevsky et al., 2012; He et al., 2015) but also in particular for identifying individual humans (Zhang and Zhang, 2014).

2.1 DeepFace

In 2014, a research group of Facebook developed a facial recognition system, *DeepFace*, that not only detects but also classifies faces according to the identities Taigman et al. (2014). Their model, trained on a dataset of 4 million labelled facial images, reached 97.35% accuracy, closely approaching human performance of 97.63%. This was a benchmark result in facial identity recognition, that increased the error rate by more than 50% compared to previous approaches on the same database. The core element of DeepFace is a 9-layer neural network with more than 120 million tunable parameters. However, during preprocessing a Support Vector Regressor (SVR) projects the images onto a 2D facial plane. This frontalization is achieved in two steps. First, a pretrained SVR crops the raw image based on detected, fiducial marker points. Then, a second SVR, trained on a prebuild 3D generic model that was developed upon another dataset consisting of 3D scans of human faces, frontalizes the image according to additional fiducial marker points. Via interpolation, initially occluded parts of the face (e.g. in case of slight tilt of the head with respect to the camera), are approximated on the final, frontalized representation. After this alignment is done, each facial feature can be assumed to have a particular spatial location; this greatly increases the information content provided by a single pixel.

Subsequently, the frontalized images are fed into a deep CNN whose first part follows the traditional architecture of CNNs - convolutional layers followed by max-pooling layers. Following this low-level feature extraction, a set of 3 locally connected layers make use of the spatial alignment achieved by frontalization. As this setup allows inferences from the spatial location of a pixel itself (rather than its value and the values of surrounding pixels), the inherent property of CNNs assuming a higher-level feature is of equal importance at all spatial locations falls short. In other words, the translation invariance of CNN achieved by shared weights is not necessary anymore. Therefore, a set of locally connected filters is used in the later layers to extract inter-individual differences. Finally, a set of 2 fully connected layers learns mappings between the combinations of high-level features and identities.

2.2 Deep Residual Learning

As frontalization is a tough computer vision task and rather unrelated to the concept of neural networks, we implemented a 33-layer deep residual convolutional neural network based on the procedure used by He et al. (2015). He et al. (2015), that won the classification task on the ILSVRC 2015 (ImageNet) competiton, randomly sampled a 224×224 crop from an image or its horizontal flip and demeaned it on a per-pixel base. They applied standard color augmentation and adopted batch normalization right after each convolution and before activation. They used Stochastic Gradient Descent (SGD) with a mini-batch size of 256. Moreover, the starting learning rate was 0.1 and this was divided by 10 when the error plateaued. They trained the model for up to 600,000 iterations but didn't use dropout. He et al. (2015) first built a 34-layer plain network. The first convolutional layer applies 64 kernels of size 7×7 with a stride of 2 followed by a pooling layer. Then they apply 64 kernels of size 3×3 for 6 convolutional layers, 128 kernels of size 3×3 for 8 convolutional layers (first layer has stride 2), 256 kernels of size 3×3 for 12 convolutional layers (first layer has stride 2) and 512 kernels of size 3×3 for 6 convolutional layers (first layer has stride 2). Downsampling is performed directly by convolutional layers with a stride of 2. Within this structure, the authors inserted shortcut connections between non-adjacent layers and thus made it a 34-layer residual network (more details in section 3).

3. Theoretical Basis and Used Procedures

3.1 Crawling

As we were aiming to implement the entire project by ourselves, we build a web scraper to crawl the training data rather than using existing datasets. The images were crawled from the Internet Movie Database (IMDB), that stores around 400 images per each of the 4,463,000 actors and celebrities. We built a web scraper in order to iterate through the identities, navigate through the website structure to find image links for them and download the corresponding files. For this task we made use of the Python framework scrapy. The website-specific spider that we built was used to scrape a defined list of identities and store their images in a foldered structure.

3.2 Data Preprocessing and Augmentation

First, the .jpg images crawled from IMDB were preprocessed in MATLAB by a pre-trained (built-in) vision cascade object detector, that implements the Viola-Jones algorithm to extract faces (Viola and Jones, 2001b,a). The entire dataset made use of the RGB color space and all faces were cropped to $112 \times 112 \times 3$ pixels. After visual inspection of the dataset, that removed false positives of the face detection algorithm, between 120 and 202 images per celebrity remained in the dataset. In total, the dataset consisted of 50312 images distributed across 400 classes (celebrities). Four of these images are presented in the upper row of Figure 1.

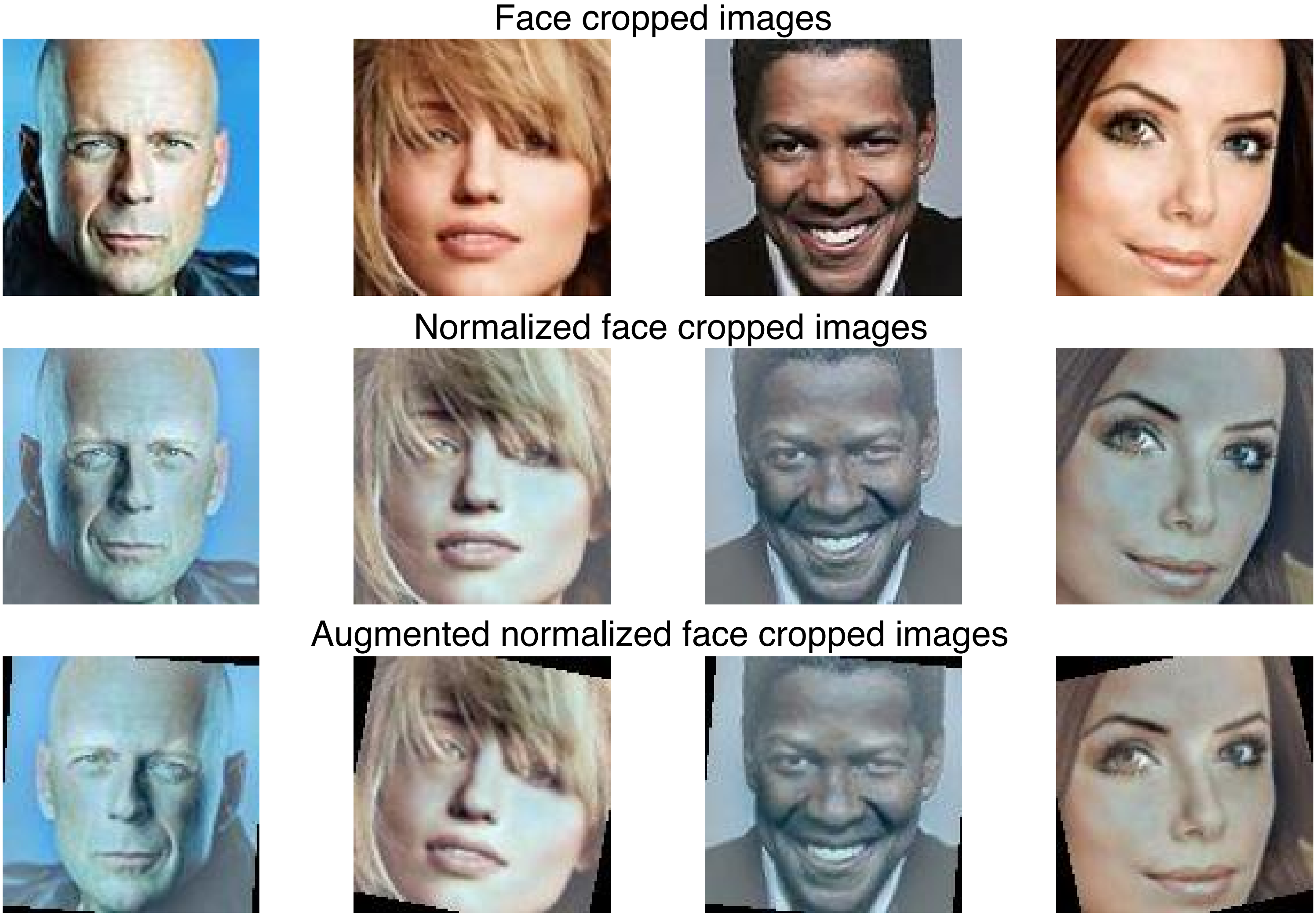

Face cropped images
Normalized face cropped images
Augmented normalized face cropped images

Figure 1: **Effect of normalizing and augmenting training data.** The upper row shows four exemplary training images that were initially crawled from IMDB, but face cropped with the Viola-Jones algorithm. The middle row displays the same four images after pixel-wise normalization over the entire dataset. The normalized images are augmented *on the fly* before fed into the network.

Subsequently, the dataset was normalized by treating every pixel of the $112 \times 112 \times 3$ input as an independent feature. This was executed in python with X -= np.mean(X,axis=0) followed by X /= np.std(X,axis=0). The normalized images are displayed in the middle row of the Figure. The dataset was split into training and testing on a 85% to 15% ratio.

The dataset was artificially enlarged according to the data augmentation used by Krizhevsky et al. (2012). Data augmentation is a common method in deep learning as it reduces the risk of overfitting and improves generalization abilities. In order to reduce memory requirements during training, data augmentation was computed on the fly (in contrast to preprocessing). A principal component analysis (PCA) was performed on the normalized dataset, with every pixel constituting a single datapoint in the RGB color space. This procedure, called *fancy PCA*, perturbs the intensities of the RGB channels in the training images along the directions of highest variance - such that they maximize information content. Thus, every pixel vector within an image (each consisting of 3 values) was altered with the same quantity consisting of noise (generated from $\mathcal{N}(0, 0.01)$) scaled by the eigenvalue of the corresponding color channel. The effect can be clearly seen on the rightmost column of Figure 1 (middle to bottom row). Thereafter, a random boolean decided whether the image gets flipped along the vertical axis (see Figure 1 bottom row, left column). Finally the images were randomly rotated within $[-15°, 15°]$.

3.3 The Concept of Residual Learning

The underlying theoretical question is the following: Does performance of neural networks gets automatically better if we stack more layers? The problem of vanishing/exploding gradients make answering this question hard in the first place, because it hinders convergence from the offset. He et al. (2015) use normalized initialization and intermediate normalization layers to solve this problem, a technique that enables deep networks to start converging for SGD with backpropagation. Another problem that hinders convergence in deep neural networks is degradation. When the network depth increases, accuracy gets saturated and then degrades rapidly. He et al. (2015) state that degradation is not caused by overfitting, but more likely arises from vanishing gradient as an inherent accompaniment of stacking layers. The authors solve this degradation problem by introducing a deep residual learning framework. An intuitive solution to the above mentioned problem is to begin with a learned shallow model and add layers to it in order to make it deeper. These added layers perform identity mappings, whereas the other layers are copied from the learned shallower network. The deep residual learning framework proposed by He et al. (2015) expands this idea. The authors explicitly let few stacked layers directly fit a desired underlying mapping. In particular, they let the stacked nonlinear layers fit the mapping $F(x) := H(x) - x$, where H(x) is the desired underlying mapping. Then they recast the original mapping into $F(x) + x$. This means that they optimize the residual mapping instead of optimizing the original, unreferenced mapping. $F(x) + x$ is implemented by a convolutional neural network with 'shortcut connections', i.e., those connections skipping one or more layers. Shortcut connections perform identity mapping, and their outputs are added to the outputs of stacked layers. These connections don't add computational complexity or extra parameters. Thus, the network can still be trained by SGD with backpropagation.

3.4 Training Environment - Amazon Web Service (AWS)

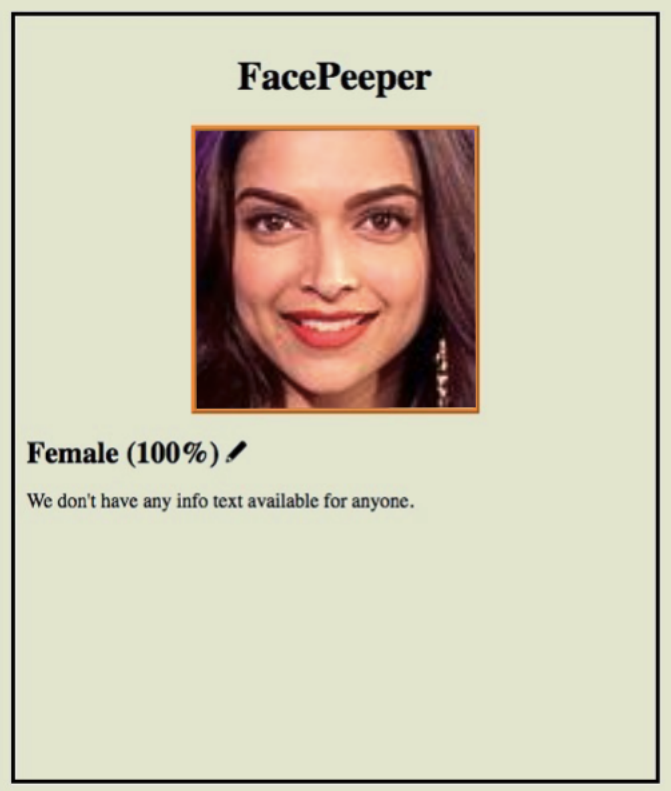
Since the university was not able to provide access to proper GPUs, we used the Amazon Elastic Compute Cloud (EC2) service of Amazon Web Services (AWS) to train our network. From our Amazon Machine Image (AMI) we launched a p2.xlarge instance that provides access to 4 vCPUs with 2.7 GHz, 61 GB memory and a NVIDIA Tesla K80 GPU. Apart from a fairly tedious setup, these provide a nice, fast and comparatively cheap way of training big networks.

3.5 Frontend

Since we want to create a usable application, we additionally set up a webserver that has a public application programming interface (API), threw which it can be reached flexibly by different frontends - with the default being a simple website.

Following the upload of an image, a preprocessing function utilizing the fontfamilycmss openCV module, detects the face within the image and crops it to size 112 × 112 × 3 in order to fit the network architecture. In case multiple or no face is detected, the API returns an error message. In case of an uniquely detected face, the user receives the classification together with a preview of the cropped face. Subsequently, the user has the opportunity to correct for misclassification. The network is then retrained with the uploaded image in conjunction with the manually set label. Figure

2 shows an exemplary shot of the model classification evaluated on an unseen image provided by a user.



FacePeeper

Female (100%)

We don't have any info text available for anyone.

Figure 2: **Web interface in action.** This figure displays generically the interaction between the user and our network within the web API. The user has just uploaded an unseen image, that the network successfully classified as female.

On the first visit to the site, the user is assigned a session ID from the server, that is used for identification. Together with each uploaded picture the user is required to provide an image ID. From the session ID (per user, server side) and the picture ID (per picture, user side) a unique hash is computed under which the image is stored on the server for a limited time. This hash is used by the API as identifier to return the cropped preview image or correct the classification. Due to this separation, users can only update the names for pictures that they actually uploaded.
The webserver is implemented in Python using the FLASK package.

4. Network Structure and Design Choices

4.1 Performance Evaluation

The network that we initially implemented is a residual deep neural network with 33 layers. In particular, 32 of these are convolutional layers and all the kernels have shape 3×3. The computational graph is a custom adaptation of He et al. (2015). First, we define a 33-layer plain network: 64 kernels are applied to 6 convolutional layers (first block), 128 kernels are applied to 8 convolutional layers (second block), 256 kernels are applied to 12 convolutional layers (third block), and 512 kernels are applied to 6 convolutional layers (fourth block). The network ends with a 400-way fully-connected layer with softmax. Each convolutional layer follows two rules: ”(i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer” He et al. (2015) (page 3). The halving of the feature map size is done in the first convolution in each block. These convolutions are applied with a stride of 2, i.e., downsampling is performed directly by convolutional layers. Moreover, in order to transform the plain network described above into a residual one, we insert shortcut connections. A single shortcut connection is implemented in the function *residualBlock*.

Unfortunately, we noticed no learning effect after 15 epochs for the classificaton of 400 classes as can be seen from Figure 3.

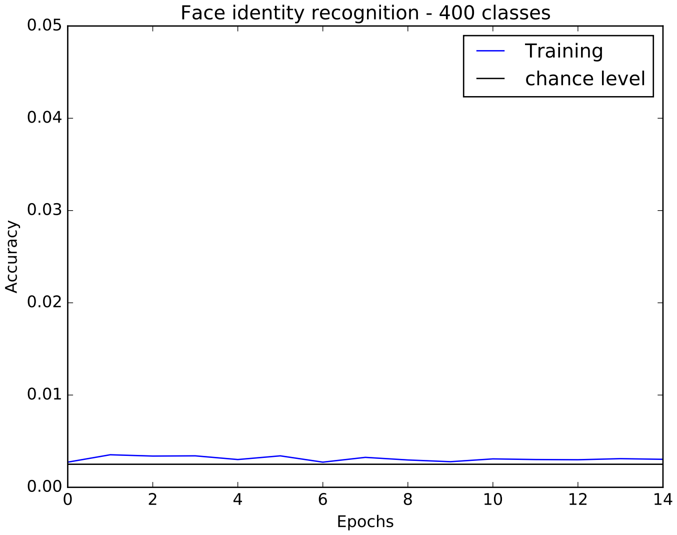


Figure 3: **Performance on classifying 400 identities.** This plot displays the accuracy of the 33-layer residual CNN in detecting the identities of 400 celebrities over the first 15 epochs. It can be seen, that no learning took place during these epochs.

Within the first 15 epochs, the network failed in diverging from the chance performance level of $\frac{1}{400}$. Please note, that the original article used 600,000 epochs on a significantly larger dataset (1,280,000 compared to 42,500 training images) to achieve good performance. This is more than 1.2 million more data then we fed into our network. Because of time contraints, we decided to simplify both, the original architecture and the task. Another reason for this decision was, that the data was self-crawled and we thus had no way of verifying whether identity classification was in principle solvable for the network. On the second glance, crawling data from IMDB might not have been the best idea. First, inter-individual differences are lowered compared to *non-celebrities*. For example, especially female celebritie's appearances follows a rather stereotypical scheme with plenty inter-individual shared features (red lips, large eyes, small noises, high cheek bones etc.). Additionally, the vast majority of celebrities within our training data are actors. Part of the job of actors is to maximize intra-individual differences, i.e. maximizing the ability to perform different roles authentically. Thus, the fact that most images show actors during movies significantly hampers the problem to solve. The simplified architecture resulted in a 9-layer residual neural network. We first built a simple convolutional neural network: we apply 6 convolutions each with 64 kernels, size: 3×3 (first block), followed by 3 convolutions each with 128 kernels, size: 3×3 (second block). The first convolution in each block has a stride of 2 (i.e., downsampling by convolution). Then we introduced three residual shortcuts: the first one after the third convolution, the second one after the sixth convolution, and the third one after the ninth convolution. Additionally, we simplified the task by classifying 10 rather than 400 identites. The result can be seen in Figure 4.

For this simulation, a significant amount of learning took place within the network. However we had doubts in tuning the model hyperparameters (learning rate, β value for ADAM optimizer). During

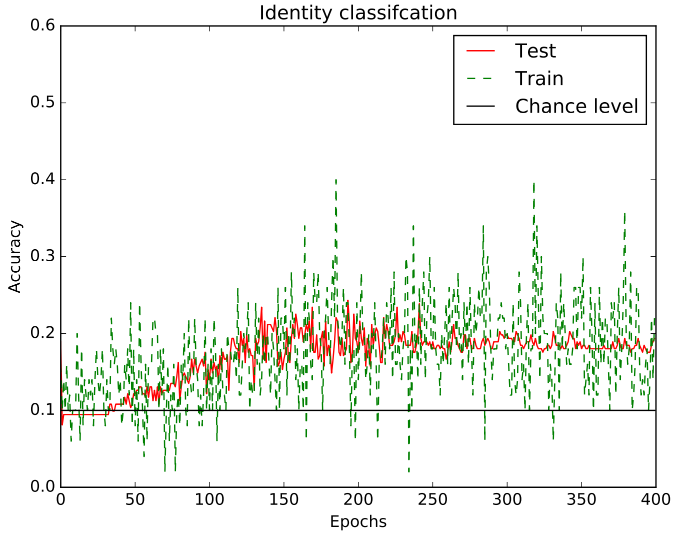


Figure 4: **Performance on classifying 10 identities.** This plot displays the accuracy of the 9-layer residual CNN in detecting the identities of 10 celebrities over the first 400 epochs. It can be seen, that considerable learning effects took place.

the entire project, we were dominantly seeking to create a usable, publicly accessible application from scratch, rather than optimizing the network's performance on a particular task.

In order to verify the network's soundness, we evaluated its performance on the MNIST dataset. This task was solved successfully and quickly (see Figure 5) with a 99.56% test accuracy performance.

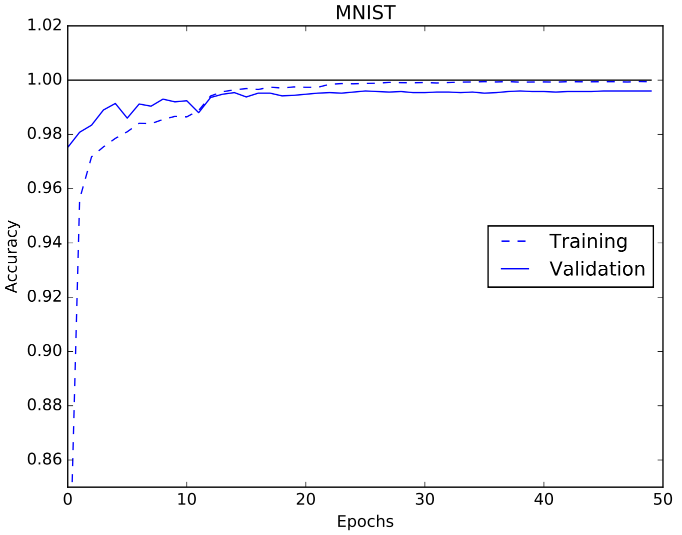


Figure 5: **Performance on MNIST classification.** This plot displays the accuracy of the 9-layer residual CNN on the benchmark MNIST task. This confirms the network's soundness.

Finally, we decided to classify the gender of the celebrities rather than the identities. As can be seen from Figure 6, this problem was solved successfully within our architecture with a final test accuray of 96.4%.

In our simulations we evaluated the effect of the data augmentation techniques presented in section 3.2. Of course, only the training data was augmented. Unsurprisingly, the accuracy on the

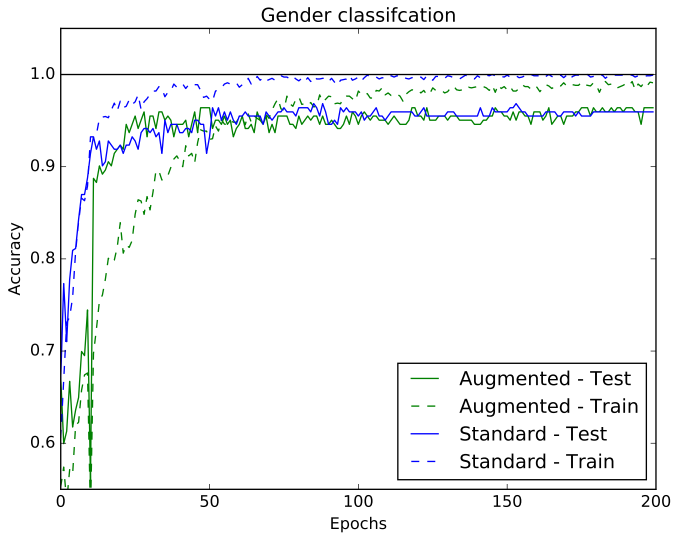


Figure 6: **Performance on gender classification.** This plot displays the accuracy of the 9-layer residual CNN on the gender classifiation task over the first 200 epochs. The plot that data augmentation had no significant effect on generalization behavior, but the network could differentiate between genders in both scenarios.

standard training data surpasses the one on the augmented dataset. However, regarding generalization abilities, we cannot clearly justify the usage of augmentation for this task - as test accuracy seems to converge against ca. 96.5% irrespective whether augmentation was used during training or not.

4.2 Implementation

Implementation was done in Python's library Tensorflow (version 1.0.0). We adopted batch normalization after each convolution and before applying the rectifier (ReLU). We initialized random weights drawn from a Gaussian distribution with fixed standard deviation of 0.1. We use ADAM optimization with a mini-batch size of 50. We implement a plateau-detecting, decaying learning rate with an initial value of is 0.01. The learning rate is halved whenever no significant improvement was made over the last 50 epochs. Because *improvement* and *plateau* refer to relative rather than absolute terms, we approximate the derivation of the accuracy function by a threshold that gradually decays as performance starts saturating.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 2015. ISSN 1664-1078.

Alex Krizhevsky, Ilya Sutskever, and Hinton Geoffrey E. Imagenet. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pages 1–9, 2012. ISSN 10495258. doi: 10.1109/5.726791. URL https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *IEEE Proceedings - Vision, Image, and Signal Processing*, page 46, 1998. ISSN 00189219. doi: 10.1109/5.726791.

Yaniv Taigman, Marc Aurelio Ranzato, Tel Aviv, and Menlo Park. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *IEEE - Computer Vision and Pattern Recognition*, 2014.

P Viola and M Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition (CVPR)*, 1:I—-511—-I—-518, 2001a. ISSN 1063-6919. doi: 10.1109/CVPR.2001.990517.

Paul Viola and Michael J Jones. Robust Real-time Object Detection. *International Journal of Computer Vision*, pages 1–30, 2001b. ISSN 09205691. doi: 10.1.1.23.2751.

Cha Zhang and Zhengyou Zhang. Improving multiview face detection with multi-task deep convolutional neural networks. *Applications of Computer Vision*, 2014.