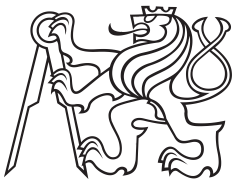
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Human face synthesis for dataset augmentation

Bc. Matěj Nikl

Supervisor: Ing. Vojtěch Franc, Ph.D.
January 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Nikl** Jméno: **Matěj** Osobní číslo: **406558**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Syntéza lidské tváře pro rozšíření trénovací množiny

Název diplomové práce anglicky:

Pokyny pro vypracování:

The current state-of-the-art methods for face recognition are based on deep Convolutional Neural Networks which rely on large databases of annotated facial images. For example, learning predictor of biological age requires large set of examples of facial images along with groundtruth age of captured subjects. Collection of such example sets is expensive. The goal of this thesis will be to develop a method augmenting the limited set of annotated real faces by generating synthetic photo-realistic faces with the same attributes. The success of the method will be evaluated via measuring accuracy improvement of a CNN learned from the generated synthetic images. In particular, the problem of age recognition will be used as a benchmark. The output will be i) a software generating faces of virtual identities with known biological age, ii) a software learning and evaluating a CNN for age prediction and iii) a statistical evaluation showing impact of the virtual faces on the prediction accuracy.

Seznam doporučené literatury:

[1] Wang Y., Liu Z., Guo B. (2011) Face Synthesis. In: Li S., Jain A. (eds) Handbook of Face Recognition. Springer, London
[2] Zhao, J. et al. Dual-Agent GANs for Photorealistic and Identity Preserving Profile Face Synthesis. NIPS 2017.
[3] Franc, V. and Cech, J. Learning CNNs from Weakly Annotated Facial Images. Journal of Image and Vision Computing, 2018.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Vojtěch Franc, Ph.D., Strojové učení FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **14.02.2019** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce: **20.09.2020**

Ing. Vojtěch Franc, Ph.D.	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce		podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Podpis studenta

Datum převzetí zadání

Acknowledgements

I gratefully thank my supervisor Ing. Vojtěch Franc, Ph.D. for his guidence and help throughout the whole process of working on this thesis. I also thank my girlfriend and my family for their endless patience and support.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, January 7, 2020

Matěj Nikl

Abstract

Training a Convolutional Neural Network (CNN) for the task of age prediction from face images requires large amounts of training data. As the freely available datasets are underrepresented for certain age categories, the CNNs cannot learn to recognize them well enough. We propose a solution to this problem by introducing a method for synthesis of new face images – of virtual identities. It creates new examples by merging appearance and shape of several compatible face parts together. Since the face parts always come from faces of the same age and gender, the synthetic example retains these attributes. We evaluate our method using a CNN and on the MORPH dataset. We try to find the best hyper-parameters of our method by performing an exhaustive search. Then, we train and evaluate the CNN in various settings, while measuring its performance using several metrics. Sadly, we observe no significant improvement using our augmentation method. We hypothesize that CNNs are able to learn themselves that the correlation between face parts can be safely ignored for the age recognition task. Hence synthetic examples obtained by permutation of face parts already contained in the training data do not improve generalization of the CNN.

Keywords: dataset augmentation, face synthesis, age prediction, CNN, poisson image editing

Supervisor: Ing. Vojtěch Franc, Ph.D.

Abstrakt

Pro trénování konvoluční neuronové sítě (CNN) pro odhad věku z obrázků tváří je potřeba velké množství trénovacích dat. Protože volně dostupné datové sady mají určité věkové kategorie nedostatečně zastoupené, konvoluční sítě se je nemohou naučit odhadovat dostatečně dobře. Jedním z řešení tohoto problému by mohla být naše metoda pro syntézu nových obrázků tváří – nových identit. Nové příklady tvoří kombinováním vzhledu a tvaru několika částí kompatibilních tváří. Vzhledem k tomu, že takovéto části vždy pochází z tváří stejného věku a pohlaví, vytvořená syntetická tvář si tyto atributy zachovává. Naši metodu hodnotíme pomocí konvoluční neuronové sítě a na datasetu MORPH. Pomocí hrubé síly se pak pokoušíme najít nejvhodnější hyperparametry naší metody. Následně sledujeme hodnoty několika metrik konvoluční sítě trénované a vyhodnocované v různých podmínkách. Naneštěstí se nám nedaří pozorovat významné zlepšení, zapříčiněné naší metodou. Naší hypotézou je, že konvoluční neuronové sítě se v úloze odhadování věku naučí ignorovat korelace mezi částmi obličejů. Z toho důvodu syntetické příklady získané permutací částí tváří z již přítomných trénovacích dat nezlepšují konvoluční síti schopnost zobecňovat na nové identity.

Klíčová slova: rozšíření datové množiny, syntéza tváře, odhad věku, CNN, poisson image editing

Překlad názvu: Syntéza lidské tváře pro rozšíření trénovací množiny

Contents

Figures

Tables

Chapter 1

Introduction

Age prediction is both theoretically interesting and practical problem with increasing number of commercial applications. For example, age-based restriction limiting virtual or even physical access to certain premises, websites, or even denying the purchase of certain goods (such as alcoholic beverages) is a one use-case. Others include customer age estimation in retail and forensic science.

The current state-of-the-art in age prediction is based on CNNs learned from vast amounts of examples. Collecting facial images with annotated age is difficult. A common approach to enlarge the training dataset is based on generating synthetic examples. In most cases, the synthetic examples are obtained by applying a set of geometrical or photometric transforms on the source annotated image. This approach helps to make the trained CNN more robust against changes in pose, rotation, scale and lighting conditions. However, it has a little impact on generalization to unseen identities. It has been observed that adding examples of new identities improves accuracy of face-verification systems more than adding new examples of identities already contained in the training set.

In this work we propose a method to generate synthetic examples of new identities. We try to accomplish this by merging appearance and shape of several indentities together. The whole process consists of several steps. First, we estimate position of the facial landmarks. From these we further estimate other important characteristics, such as the face pose and the (simplified) face expression. Then, we find tuples of compatible faces to be merged together. Finally, while controlling the shape and appearance transfer, we stitch certain face parts together using affine transformations. We use the seamless cloning algorithm [31] as a post-processing step to improve the overall image quality after stitching.

We evaluate the capabilities of the proposed method using a CNN model, which is trained and evaluated in various settings on the MORPH dataset [32]. While we verify that the generated examples are indeed different from the original ones and mostly indistinguishable from real faces, they unfortunately do not seem to provide any new useful information for the CNN to train from. Hence, we observe no significant accuracy improvement when the training dataset is extended by the synthetic examples. We hypothesize that CNNs

are able to learn that for the age recognition task, the correlation of face parts can be safely ignored. Therefore, synthetic examples obtained by permutation of face parts that have been already contained in the training set bring no new information for learning the CNN.

This thesis is structured into five chapters. Chapter 2 introduces related methods both for the age prediction task, and for the general and face-specific dataset augmentation. It also gives further motivation for this work. Chapter 3 introduces our proposed method in detail. Chapter 4 explains all the details and results of the conducted experiments. Finally, Chapter 5 concludes this thesis.

Chapter 2

Related methods

Age estimation is a long standing and still unsolved problem. Before the deep learning revolution occurred, methods based on local appearance descriptors such as HOG [7], texture descriptors such as LBP [28] and SURF [3], and others were relatively successfully applied. For example [19] examines the usage of the mentioned features together before applying the Canonical Correlation Analysis (CCA) [18] for the final age estimation. This approach yields the MAE of 4.25 on the MORPH dataset.

As deep learning became more widespread, convolutional neural networks found their way into the age estimation problem. They were useful even as a mere feature extractor for the final age predictor [36, 13], e.g. achieving MAE of 4.77 on the MORPH dataset.

Further works usually employ the end-to-end training regime of CNNs, i.e. training the CNN to predict the age directly (using some encoding) [23, 10, 19, 24, 2, 1, 27, 22, 34, 29], achieving MAE on the MORPH dataset as low as 1.96 [15]. The main differences between the referenced works consist of:

- the input structure and its preprocessing (e.g. multiple aligned patches of face as the input, registration, etc.),
- the architecture of the model, and
- the output age encoding and the training loss function.

2.1 Labeled data scarcity

One of the reasons why the deep learning approach hasn't improved upon the previous results instantly is the scarcity of labeled training data. Moreover, the labels are often noisy and heavily unevenly distributed. Figures 2.1 and 2.2 show the age distributions for various available datasets, and Table 2.1 gives a summary of age range present, the overall example counts and a measure of their non-uniformness.

The target ages are usually either manually assigned based on the appearance (so called the apparent age) as is the case for the APPA-REAL and ChaLearnAge datasets, or automatically assigned e.g. based on the known

Dataset name	age range	# examples	$D_{\mathrm{KL}}(pmf \parallel \mathcal{U}\{0, 100\})$
AgeDB [26]	1 – 100	16488	0.387
APPA-REAL [11]	1 – 100	7591	0.491
UTKFace [37]	1 – 100	23468	0.552
IMDB-WIKI [33]	0 – 100	520717	0.641
CACD2000 [6]	14 – 62	163446	0.810
ChaLearnAge [12]	2 – 87	3611	0.837
FG-NET [21]	0 – 69	978	0.900
MORPH [32]	16 – 77	55095	0.933
total	0 – 100	791394	0.628

Table 2.1: Summary of available datasets for age estimation, sorted by their *non-uniformness*

date of birth and the date of capture of the photograph (IMDB-WIKI and UTKFace datasets). In case of the automatic procedure, the error comes e.g. from incorrect identity assignment.

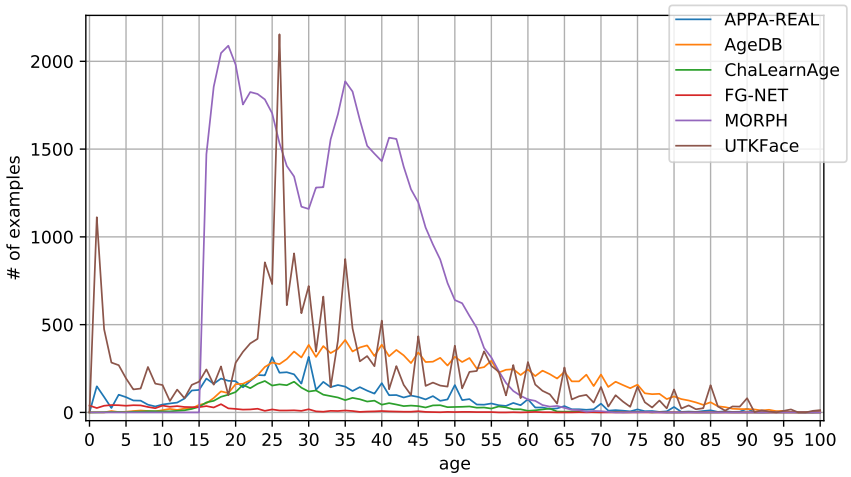


Figure 2.1: The histogram of various datasets for the age estimation task

2.2 A survey of methods used to generate synthetic examples

In the supervised learning setting, the dataset augmentation is a method of expanding the training dataset by creating new training examples with known label. When the input data happen to be images, a whole list of augmentation methods becomes available for use. Such methods simply transform the input image into a different one while preserving the label.

Modern deep learning frameworks such as PyTorch [30] often provide a way of performing such augmentations on the fly, even in parallel with the

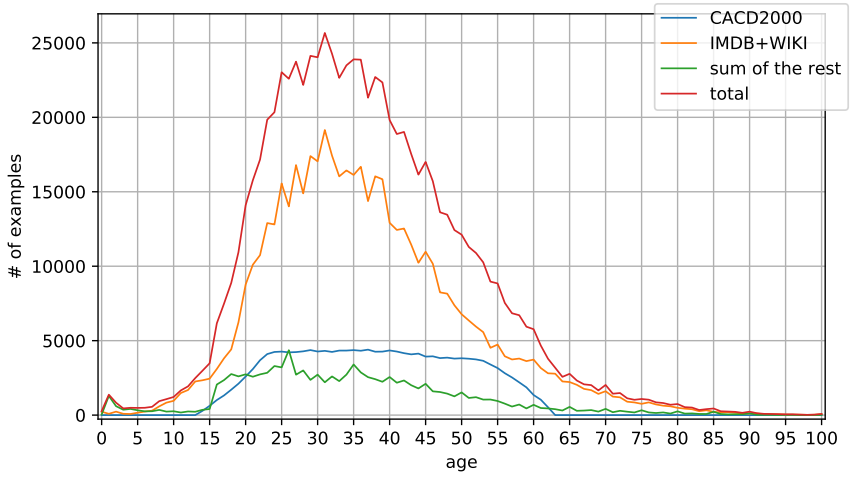


Figure 2.2: The histogram of various datasets for the age estimation task

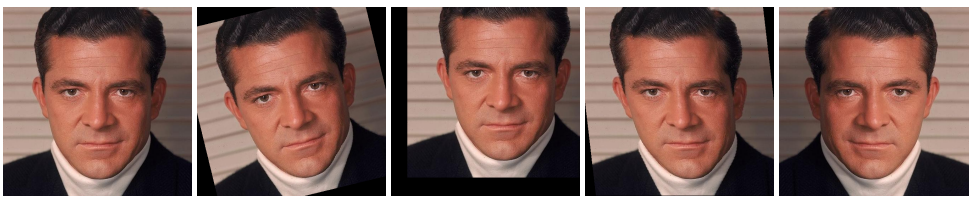
training itself, which is often done on a GPU. Therefore, it is possible to randomize the whole augmentation process each time a new training example is fetched. One can, for example, apply augmentations with some probability, alter the order of the augmentations, or even randomize their parameters (if any). This potentially leads to vast amount of variations in the augmented data.

2.2.1 Geometrical transformations

A commonly used geometrical transforms include cropping and various affine transformations, such as: rotation, translation, scaling, shearing and the vertical flip (reflection). Other, less used transformations are for example: perspective transformations and other elastic transforms.

When the transformation is used with *reasonable* parameters, the transformed image keeps its original label as demonstrated in Figure 2.3.

The reason behind using such transformation is to enhance variation in location, scale, etc. which occurs naturally in the data, but might not be captured (enough) in the training data.



(a): original **(b):** rotation **(c):** translation **(d):** shear **(e):** flip

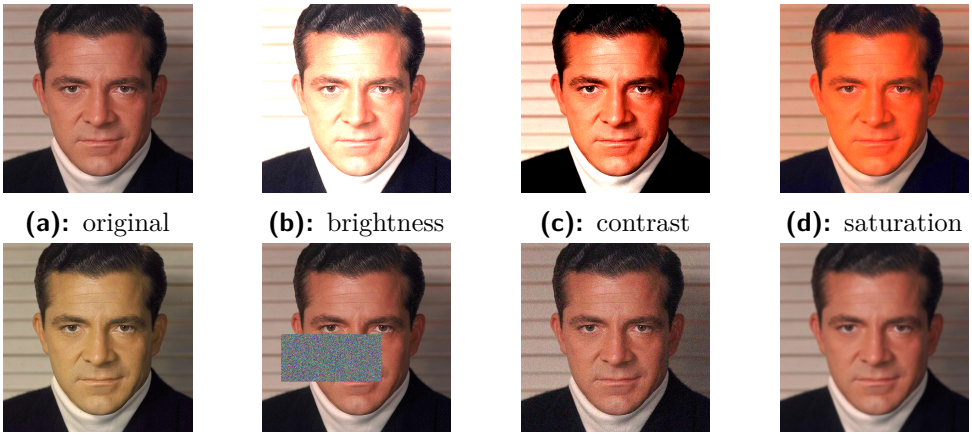
Figure 2.3: Examples of various geometrical transformations

2.2.2 Photometric transformations

The photometric transformations apply various functions to the pixel values. Color jittering is one example. One can for example alter brightness, contrast, saturation and/or even hue (Figures 2.4b–2.4e).

A different approach is taken in the random erasing [38] augmentation, which occludes random parts (rectangles) of the input image by replacing its pixel values with unit gaussian noise (Figure 2.4f).

Other approaches include adding (gaussian) noise to the whole image (Figure 2.4g), salt and pepper noise, blurring (gaussian in Figure 2.4h, motion, ...) and even applying jpeg compression.



(a): original **(b):** brightness **(c):** contrast **(d):** saturation

(e): hue **(f):** erasing **(g):** noise **(h):** blur

Figure 2.4: Examples of various photometric transformations

2.2.3 Face-specific transformations

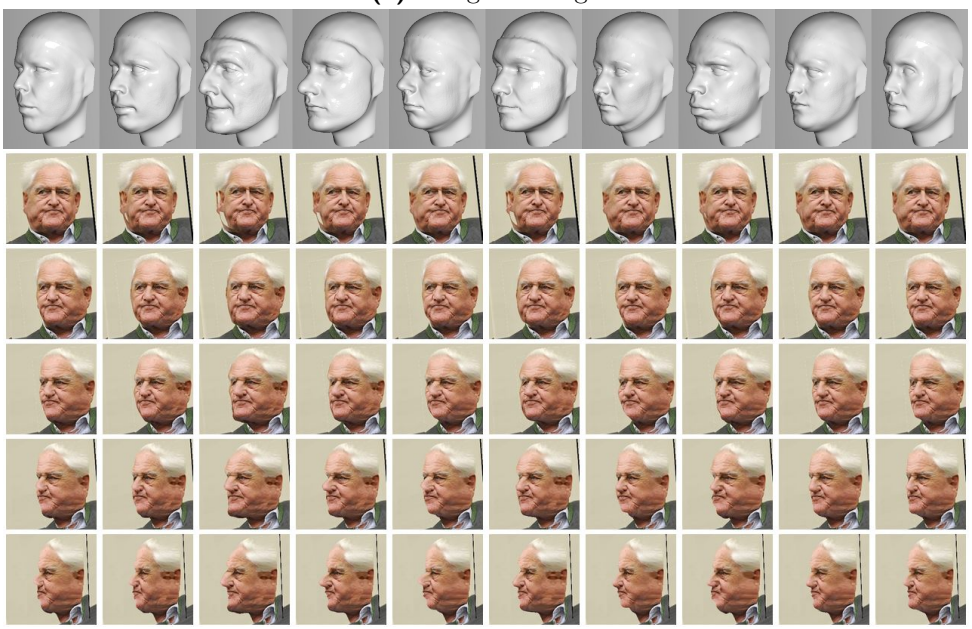
While tackling the problem of Unconstrained Face Recognition, the authors of [25] developed a face-specific augmentation, which takes into consideration the 3D shape and appearance of faces. Using this information they are able to generate multiple views on a person's face in many yaw angles and even with different 3D face shapes. See Figure 2.5 for generated examples.

Their method greatly improves the variability of the training examples, thus either improving the accuracy of the trained model or reducing the needed number of training examples to achieve the same accuracy. It is also complementary to the above presented generic image augmentations.

2.2.4 Mixup augmentation

Mixup augmentation is based on the prior knowledge that linear interpolations of the input feature vectors should lead to linear interpolations of the associated targets [17]. Therefore, it is applicable only for classification tasks. It creates training examples online, each time from exactly two existing examples by linearly interpolating them and their targets. For every pair there

(a): Original image



(b): Examples of rendered faces with different yaw angle and shape

Figure 2.5: Examples generated from a single picture [25]

is a new interpolation coefficient sampled from the Beta distribution with its parameters $\alpha = \beta$, where α is the hyperparameter. For $\alpha = 1$ the Beta distribution becomes uniform distribution $\mathcal{U}(0, 1)$, while values $\alpha \in [0.1, 0.4]$ which favour sampled values closer to 0 or 1 are suggested.

Employing this augmentation method encourages the model to behave linearly *in-between* training examples. See Figure 2.6 shows an augmented example together with its two original examples.

2.3 Comparison to our method

Our method is a face-specific method. It aims at guiding the model to be identity-invariant. It does so by stitching together several parts of people's faces, i.e. it is a many-to-one augmentation method. The following section explains it in detail.

(a): age 44: 100 %

(b): age 44: 40 %, 23: 60 %

(c): age 23: 100 %

Figure 2.6: Example of a mixup-augmented image (b), together with its two source images (a) and (c) linearly interpolated with coefficient of 0.4

Chapter 3

Proposed method

This chapter presents in detail the proposed method for human face synthesis. The goal of the method is to augment datasets of faces with known age and gender, e.g. to inflate underrepresented categories.

Each new synthetic image is created by stitching together face parts from selected annotated images. The assumption is that if the input faces are of the same age, gender, have similar pose and expression, then the resulting synthetic face will appear realistically and will preserve the age and gender attributes.

For example, the input can consist of four face images of people with different identities, all having the same age, gender and similar pose. The output will be a single face with a non-existent identity which borrowed eyes from the first input face, mouth from the second, the rest of the face from the third and its hair together with the background from the fourth.

The method works in several steps. First, we estimate location of the face together with its facial landmarks within each image. Then, from the position of the facial landmarks we estimate the pose (i.e. the yaw and the pitch angles) and the expression of the face.

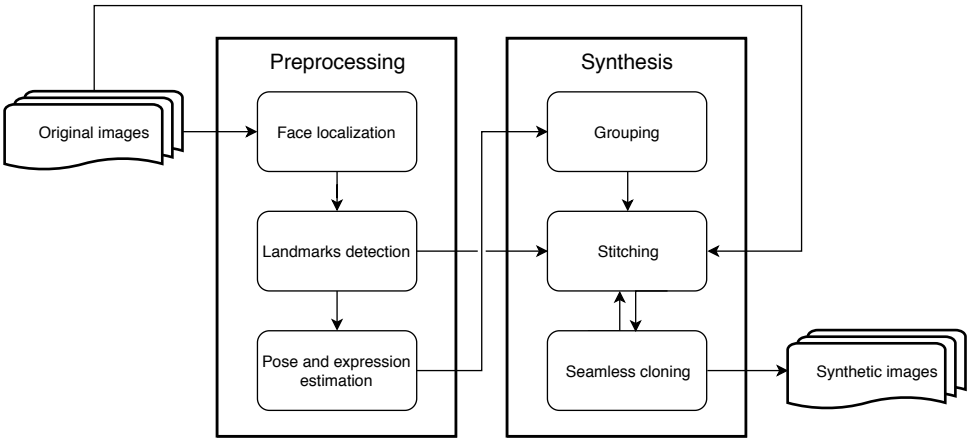


Figure 3.1: An overview of the processing pipeline

Having the preprocessing steps completed, we can proceed further to the next step – the grouping of images. In this step we create tuples of *compatible*

images, i.e. faces of the same age and gender, similar pose and expression. These tuples of images are then in the last two steps iteratively transformed into the final new, synthetic image.

Each step will be discussed in detail in the following sections, Figure 3.1 shows an overview of the method and Figure 3.7 shows a concrete example of four faces being stitched together.

3.1 Face localization

The first step of the whole pipeline is the face localization within the image at hand. Given an input image I a localization algorithm L_b will output a rectangular bounding box B within the image I, which surrounds the face present in the image:

$$L_b : I \to B, \qquad \text{where } B \text{ surrounds face in } I$$

We use a Waldboost-based commercial[1] face detector [35]. Once a rectangular bounding box is known, the next step is to locate a predefined set of facial landmarks within the face.

3.2 Facial landmarks detection

The proposed method is based on swapping semantically same parts of several face images. The face parts are defined with respect to a set of fixed facial landmarks, detection of which is described below. The detected landmarks are also used for the pose and expression estimation as described in the next section.

Given an image I and a bounding box B, a facial landmarks detection algorithm L_p returns a predefined set of facial landmarks F:

$$L_p : (I, B) \to F, \qquad \text{where } F \in \{(x_i, y_i) \mid i \in [0\,..\,67]\}$$

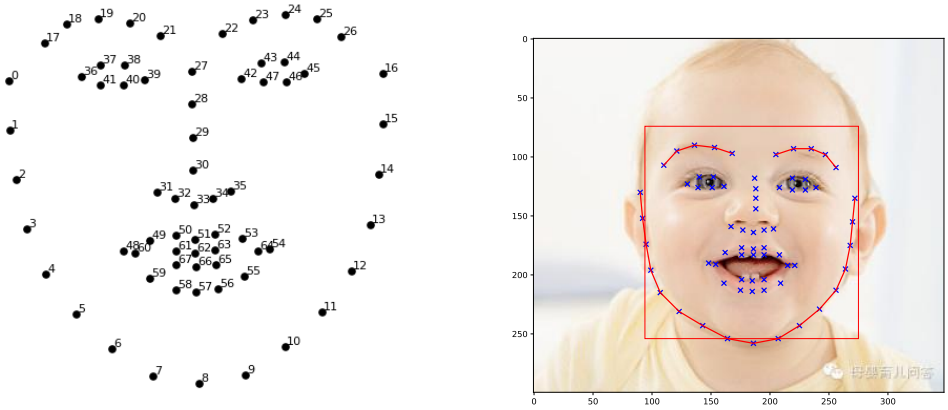
In particular, we use a set of 68 landmarks. Please refer to Figure 3.2 for an overview of facial landmarks.

We use 2D Face alignment library [5] to solve this task.

3.3 Pose and expression estimation

When creating the synthetic face, we want to stitch together parts of faces with similar pose and expression. This section describes how the pose and expression are estimated.

[1] Courtesy of Eyedea recognition, Ltd. http://www.eyedea.com



(a): The complete set of 68 facial landmarks

(b): An example of a successful landmarks detection

Figure 3.2: Facial landmarks

3.3.1 Estimating the pose

Out of the three angles (yaw, pitch and roll) which fully describe the face pose in 3D, we use only yaw and pitch as it's easy to correct for roll difference between two faces e.g. by rotating the image of one of them.

We model each face by a single 3D shape obtained by averaging 3D shapes of a large number of faces. As we have already detected the 2D landmarks (in Section 3.2) and we have a 3D average shape of them as well, we obtain the desired angles by solving the Perspective-n-Point problem [14].

The PnP problem finds a 3D pose $(\boldsymbol{p} \in \mathbb{R}^3, (\phi, \theta, \psi) \in \mathbb{R}^3)$ of a perspective camera, such that the projection of the 3D shape into 2D would return the detected 2D landmarks. See Figure 3.3 for visualization of the problem setup.

Since we know the pose of the 3D shape, the estimated angles ϕ, θ, ψ of the camera describe the pose of the face captured by the 2D landmarks.

3.3.2 Estimating the expression

The most significant difference in expression which the stitching algorithm wasn't able to correct for is the *mouth openness*. Mouth openness is calculated from detected facial points F:

$$O : F \to [0, 1] \tag{3.1}$$

$$F \mapsto \frac{\frac{\|F_{61} - F_{67}\| + \|F_{62} - F_{66}\| + \|F_{63} - F_{65}\|}{3}}{\frac{\|F_{21} - F_8\| + \|F_{22} - F_8\|}{2}} \tag{3.2}$$

It is the average distance between inner points marking the lips normalized by the height of the face, which is calculated as the average distance between the lowest chin point and innermost eyebrows points. The bigger the value, the more is the mouth opened.

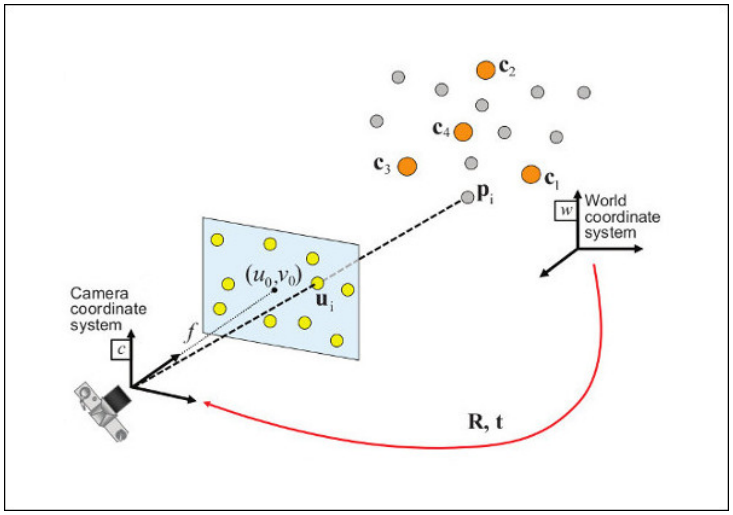


Figure 3.3: The PnP problem setup [4]

3.4 Images grouping

This preprocessing step creates **(K + 1)-tuples** of faces whose parts will be stitched together to create the new synthetic face. Each tuple consists of one *destination* face and K *source* faces. Grouping of the faces into the $(K + 1)$-tuples is driven by the following criteria/restrictions:

- Small bounding box relative size difference between each source face and the destination face.
- Similarly, the yaw, pitch and mouth openness (i.e. the pose and expression) of each source face must be close to the pose and expression of the destination face.
- Each face part can be used only limited number of times.
- No source face part can be used more than once in each destination face.
- We want to generate N new examples, which utilize the available original faces as uniformly as possible.

When creating synthetic images we use a list of K schemes which influence the grouping and govern the stitching process; i-th source face in each $(K+1)$-tuple corresponds to the i-th scheme. More specifically, each **scheme** defines the following:

- the face part (and its boundary) which will be transferred,
- whether mouth openness of the source face is checked against the destination face, and
- other stitching process parameters, which are discussed further in Section 3.6.

Our greedy algorithm for finding the $(K+1)$-tuples has two core steps. First, it creates a set of all compatible source faces for each destination face and each scheme. Then, it greedily tries to create $(K+1)$-tuples while meeting the requirements and restrictions.

Algorithm 1 shows pseudocode of our algorithm, which further call functions in Algorithm 2 and 3.

3.5 Stitching

The final step in creation of the synthetic face is the face stitching algorithm. It is responsible for transferring face parts from source images to the destination image. In short, having a $(K+1)$-tuple of images, the stitching algorithm for each source face and the destination face iteratively performs the following steps:

1. From the faces' landmarks it computes corresponding face part points.
2. On the pointwise averaged face part points it computes Delaunay triangulation [8] and imposes it back on the original points.
3. It transfers corresponding triangles from the source face to destination face by applying affine transform on each.
4. It applies seamless cloning [31] to blend the inserted part into the destination face.

The mentioned steps are further discussed below.

3.5.1 Computing face part points

A face part is described by a set of points calculated from the 68 facial landmarks. It consists of two parts:

- A set of new points, where each point is defined as a convex combination of three facial landmarks. These three facial landmarks form an enclosing triangle in the Delounay triangulation performed on an average 2D face shape.
- A subset of original landmarks contained within the convex hull of the newly added points.

Each scheme defines its own coefficients for the convex combinations and the subset of the original landmarks (i.e. defines its own face part and the boundary of the face part).

We defined the face parts manually. Figure 3.4 shows a mouth part specified within the editor.

[2]Defined as average of the width and the height of the bounding box

Algorithm 1 Grouping algorithm

function GROUPING($\boldsymbol{\delta}$, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, $\boldsymbol{\mathcal{T}}$, n, K, N)
  Input
    δ_i size of the i-th face bounding box[2]
    ϕ_i yaw of the i-th image
    θ_i pitch of the i-th image
    $\mathcal{T}_\delta$ coefficient for the bounding box size threshold
    $\mathcal{T}_\phi$ yaw difference threshold
    $\mathcal{T}_\theta$ pitch difference threshold
    $\mathcal{T}_\mu$ mouth openness difference threshold
    $\mathcal{T}_c$ usage count threshold
    n number of input images
    K number of schemes
    N number of desired generated images
  Output
    Q set of (K+1)-tuples of image indices
  $Q \leftarrow \emptyset$
  $I \leftarrow \{1\,..\,n\}$
  $C, J \leftarrow$ INIT($\boldsymbol{\delta}$, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, $\boldsymbol{\mathcal{T}}$, n, K)
  while $|I| > 0$ **do** ▷ while there is any destination face
    for $i \in I$ **do**
      $T \leftarrow$ TRYFINDSOURCEFACES(i, $\mathcal{T}_c$, C, K, J)
      if $\emptyset \notin T$ **then** ▷ all K source faces set
        $T_0 \leftarrow i$ ▷ set the destination face
        $Q \leftarrow Q \cup \{T\}$
        if $|Q| = N$ **then**
          return Q ▷ N tuples created
        end if
        for $k = 0\,..\,K$ **do**
          $C^k_{T_k} \leftarrow C^k_{T_k} + 1$ ▷ increment usage counters
          if $k > 0$ **then**
            $J^k_i \leftarrow J^k_i \setminus \{T_k\}$ ▷ forbid usage of these source face parts again for i-th destination face
          end if
        end for
      end if
      if $\emptyset \in T$ **or** $C^0_{T_0} = \mathcal{T}_c$ **then**
        $I \leftarrow I \setminus \{i\}$
      end if
    end for
  end while
  return Q ▷ less than N tuples created
end function

Algorithm 2 Init function of the grouping algorithm

function INIT($\boldsymbol{\delta}$, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, $\boldsymbol{\mathcal{T}}$, n, K)
  Input
    δ_i size of the i-th face bounding box
    ϕ_i yaw of the i-th image
    θ_i pitch of the i-th image
    $\mathcal{T}_\delta$ coefficient for the bounding box size threshold
    $\mathcal{T}_\phi$ yaw difference threshold
    $\mathcal{T}_\theta$ pitch difference threshold
    $\mathcal{T}_\mu$ mouth openness difference threshold
    $\mathcal{T}_c$ usage count threshold
    n number of input images
    K number of schemes
  Output
    C counters
    J sets of image indices for j-th image and k-th scheme
  for $i = 1\,..\,n$ **do**
    $C_i^0 \leftarrow 0$
    for $k = 1\,..\,K$ **do**
      $C_i^k \leftarrow 0$
      $J_i^k \leftarrow \left\{ j = 1\,..\,n \;\middle|\; i \neq j,\ (1-\mathcal{T}_\delta)\delta_i \leq \delta_j \leq (1+\mathcal{T}_\delta)\delta_i \right\}$
      $J_i^k \leftarrow J_i^k \setminus \left\{ j = 1\,..\,n \;\middle|\; |\phi_i - \phi_j| > \mathcal{T}_\phi \right\}$
      $J_i^k \leftarrow J_i^k \setminus \left\{ j = 1\,..\,n \;\middle|\; |\theta_i - \theta_j| > \mathcal{T}_\theta \right\}$
      if k-th scheme applies the mouth openness threshold **then**
        $J_i^k \leftarrow J_i^k \setminus \left\{ j = 1\,..\,n \;\middle|\; |\mu_i - \mu_j| > \mathcal{T}_\mu \right\}$
      end if
    end for
  end for
  return C, J
end function

Coefficients $x_1, x_2, x_3 \in \mathbb{R}$ are found for each new point $\boldsymbol{p} \in \mathbb{R}^2$ by solving the following quadratic program:

$$\begin{aligned} \underset{x_1,x_2,x_3 \in \mathbb{R}}{\arg\min} \quad & x_1^2 + x_2^2 + x_3^2 \\ \text{s.t.} \quad & x_1\boldsymbol{a} + x_2\boldsymbol{b} + x_3\boldsymbol{c} = \boldsymbol{p} \\ & x_1 + x_2 + x_3 = 1 \\ & x_1, x_2, x_3 \geq 0 \end{aligned} \tag{3.3}$$

where $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^2$ are the vertices of the enclosing triangle.

Algorithm 3 Tries to create tuple of K source faces for i-th destination face

function TryFindSourceFaces(i, $\mathcal{T}_c$, K, J)
  Input
    i the destination face index
    $\mathcal{T}_c$ usage count threshold
    C counters
    K number of schemes
    J sets of image indices for j-th image and k-th scheme
  Output
    T (K)-tuple of source face indices
  for $k = 1 \,..\, K$ **do**
    $T_k \leftarrow \emptyset$
    for $j \in J_i^k$ **do**
      if $j \notin T$ **and** $C_j^k < \mathcal{T}_c$ **then**
        $T_k \leftarrow j$ ▷ j is the k-th source face
        break
      end if
    end for
  end for
  return T
end function

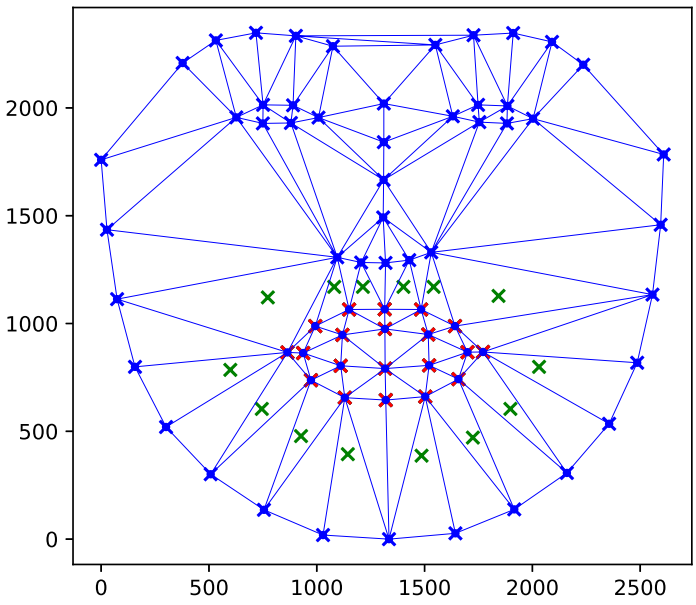


Figure 3.4: Average 2D facial landmarks with their Delaunay triangulation (in blue) with a mouth part (new points in green, subset of original landmarks in red)

3.5.2 Computing joint triangulation

We need to calculate a joint triangulation on two sets of points in a 2D-plane [9] to have a set of corresponding triangles, which can be later used for image transfer. Note that triangulation of a set of points is not unique. Note also that a triangulation found for one set of points might not be a triangulation on another set of points in general (due to overlapping triangles).

Since we are dealing only with faces with similar pose and expression, the relative location of detected landmarks is similar as well. Therefore, we simplify the problem of joint triangulation by performing the Delaunay triangulation only on the pointwise averaged face part points (see Figure 3.5c) and posing the triangulation back on the two original sets. Additionally, we remove all triangles which happen to cover area outside of the face part boundary. We find triangles outside of the boundary when the boundary happens to be non-convex, e.g. for the mouth part when a person is smiling.

See Figure 3.5 for visualization of the whole process.

3.5.3 Transferring corresponding triangles

To transfer a face part from the source face to the destination face we transfer (and transform) the corresponding triangles obtained in the previous step.

Two corresponding triangles give us 3 pairs of corresponding points, which uniquely define an affine transform. This affine transform is then used to transform the source triangle to match the shape of the destination triangle and to replace its bitmap.

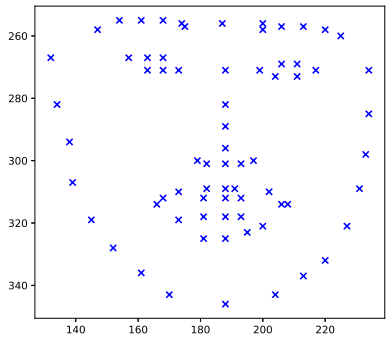
This process is repeated for all triangles of the current face part and hence transferring the bitmap of the source part to the destination part. Figure 3.6c illustrates this on a mouth part.

3.5.4 Seamless cloning

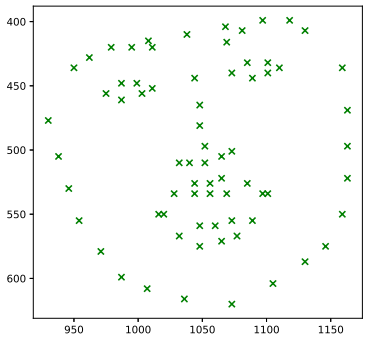
Due to various differences e.g. in the skin tone, skin texture, lighting conditions, etc., the resulting stitched face almost always actually appears stitched, i.e. the stitched parts can be easily seen and distinguished.

To alleviate this, we apply the seamless cloning, which is an algorithm based on the Poisson Image Editing [31]. It ensures the compliance of source and destination boundaries, while preserving the bitmap intensities' gradient of the inner part, thus eliminating visibility of the boundary and making the stitched part blend in.

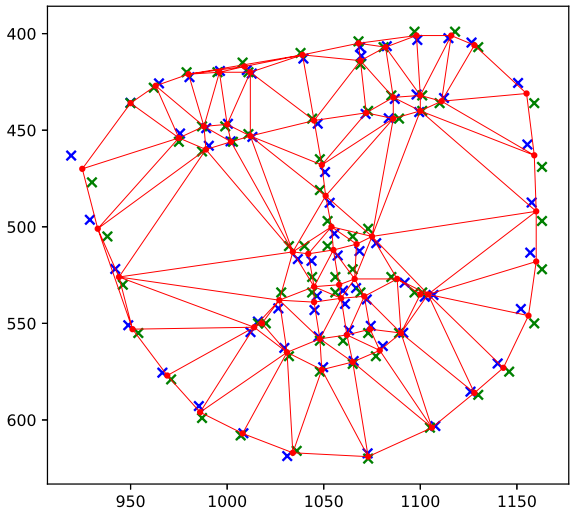
Compare Figures 3.6c and 3.6d to see the effect of seamless cloning. The boundary becomes indistinguishable and the stitched part blends with the rest of the image. However, the lighting might look unnatural in the context of the scene.



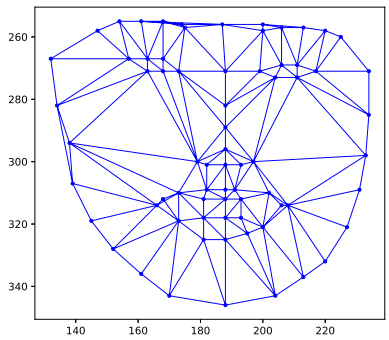
(a): Source face landmarks



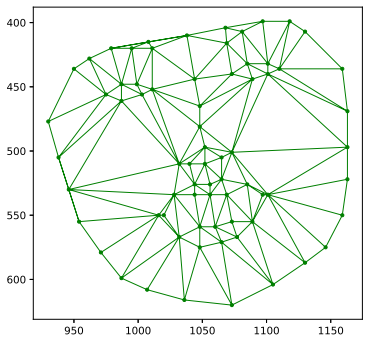
(b): Destination face landmarks



(c): Source face landmarks (blue) affinely transformed (minimizing squares error) to match the destination face landmarks (green) and their pointwise average with the corresponding Delaunay triangulation (red)



(d): Source face landmarks with imposed triangulation



(e): Destination face landmarks with imposed triangulation

Figure 3.5: The process of obtaining the joint triangulation on two corresponding sets of points. (a) and (b) show source and destination face part landmarks, (c) shows the found triangulation and (d) and (e) shows the found triangulation posed back on the original source and destination points

(a): Source **(b):** Destination

(c): After stitching **(d):** After seamless cloning

Figure 3.6: Transferring the mouth part

3.6 Additional stitching parameters

There are two additional stitching parameters specified for each scheme. They control how much of the source bitmap and the source shape is transferred to the destination. Figure 3.8 explains the effect of varying both parameter values.

3.6.1 Controlling the shape transfer

It is not necessary to preserve the shape of the destination face part. Only the boundary points must remain to ensure the transferred face part gets stitched into the right place. Otherwise, the rest of the points can potentially be altered as long as they stay within the boundary.

Therefore, the $\beta \in [0, 1]$ parameter is introduced. It controls linear interpo-

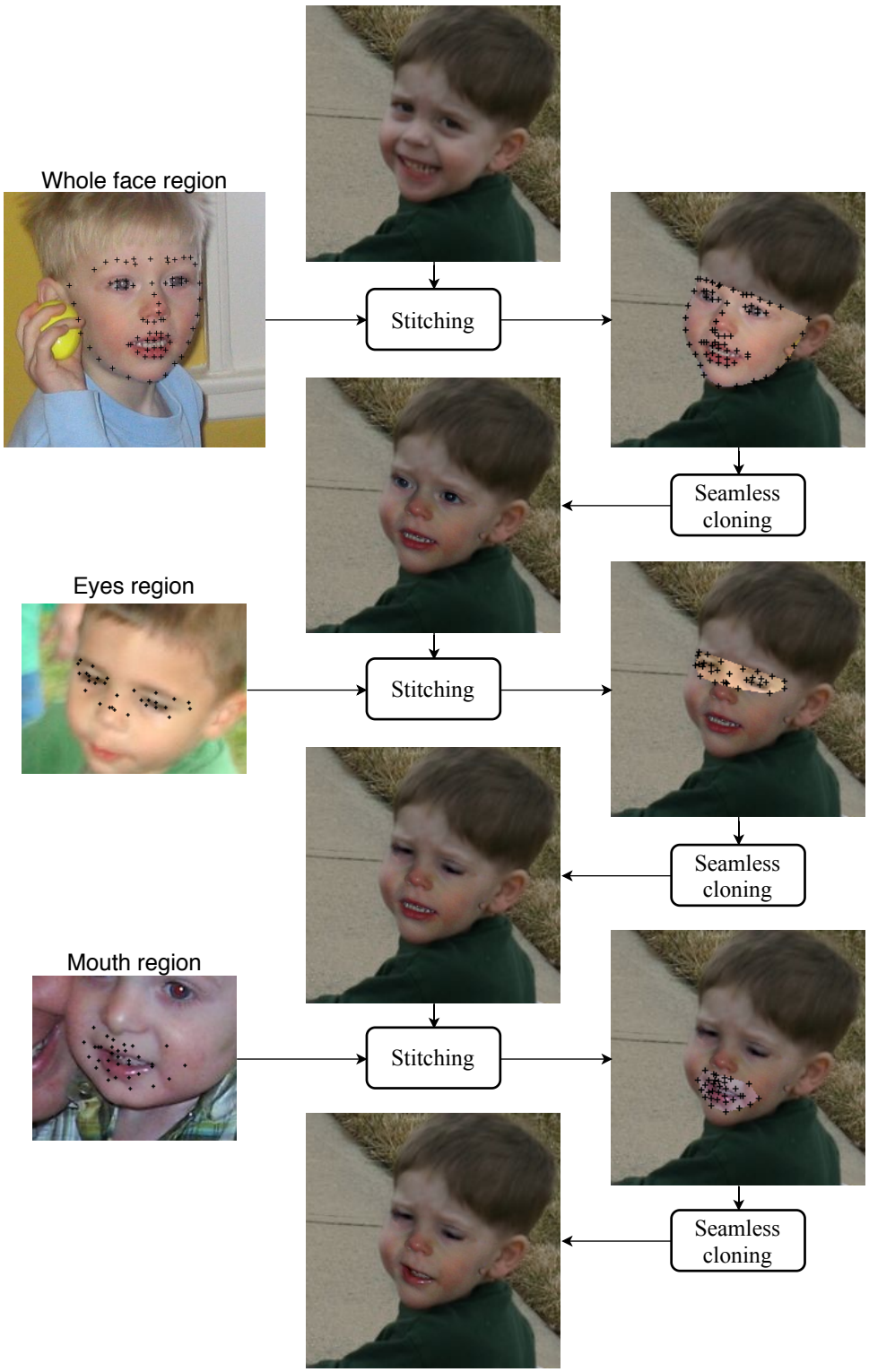


Figure 3.7: A simplified overview of the process of creating a synthetic face from three source faces (left) with the destination image at the top and the result at the bottom

lation from the original destination shape to the new source shape. I.e. after the affine transformation of the source part points to the destination face part points (minimizing squares error of the corresponding boundary points), each new destination shape point within boundary is calculated as:

$$\boldsymbol{p}_{dst\_new} = \beta \boldsymbol{p}_{src} + (1 - \beta)\boldsymbol{p}_{dst}, \quad \beta \in [0, 1]. \tag{3.4}$$

3.6.2 Controlling the bitmap transfer

Another option is to transfer the bitmap partially. The $\alpha \in [0, 1]$ controls the linear interpolation between the original destination bitmap to the new stitched bitmap. Each resulting pixel $\boldsymbol{c} \in [0\,..\,255]^3$:

$$\boldsymbol{c}_{res} = \alpha \boldsymbol{c}_{stitched} + (1 - \alpha)\boldsymbol{c}_{original}, \quad \alpha \in [0, 1]. \tag{3.5}$$

In such a case when $\alpha < 1$ and $\beta > 0$ (i.e. the bitmap is not transferred fully and the destination shape has been altered), the above Equation (3.5) cannot be applied as the altered face part points do not align. Therefore, there is an additional stitching step which alters the destination face part from its original shape to the new shape. Only after that the bitmap linear interpolation is performed.

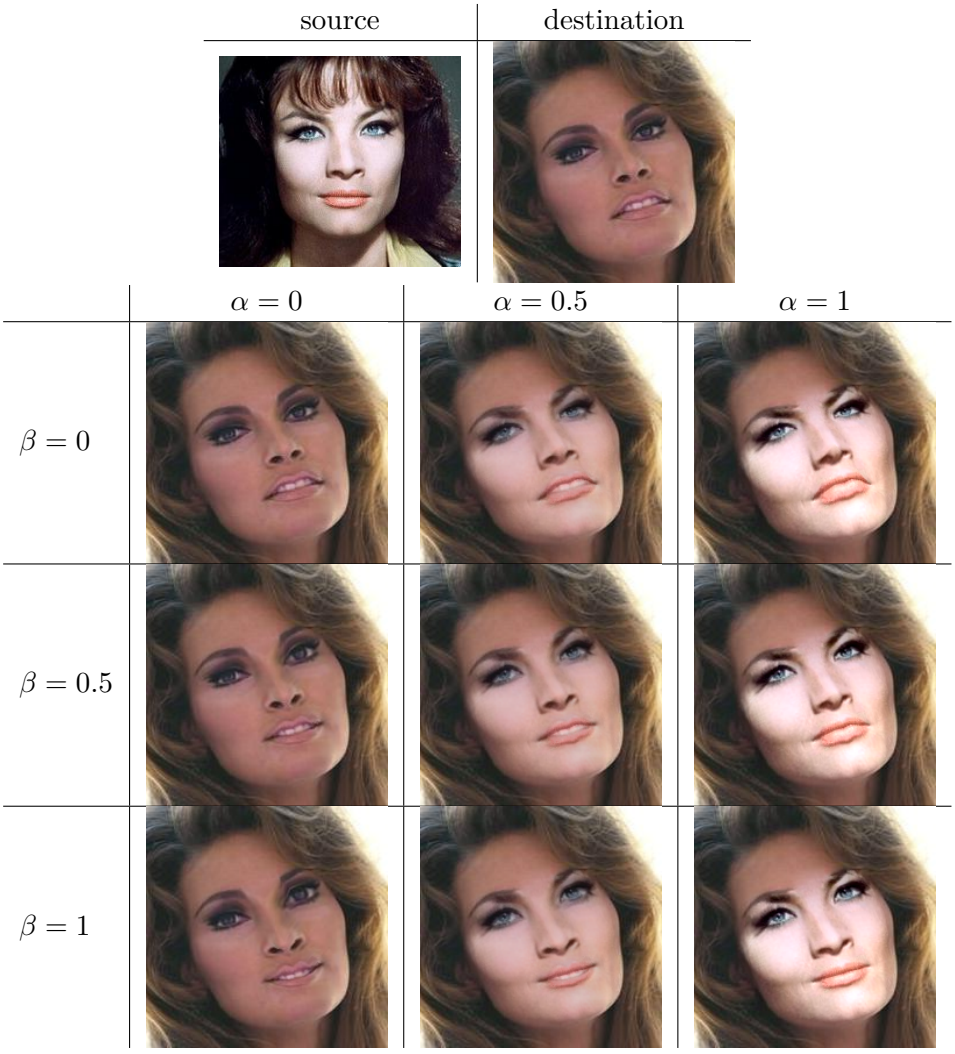


Figure 3.8: The result of transferring the whole face with different α and β parameter values

Chapter 4

Experiments

In this chapter we go over several experiments we conducted to evaluate the proposed method. First we present the used datasets in more detail, we also visually analyze the synthetized faces. Then we introduce the metrics used to evaluate the performance of the trained models, the architecture of the used model, we shortly describe how it was trained and evaluated, and finally we present the experimental results.

4.1 Datasets used

The experiments were conducted using two versions of the MORPH [32] dataset: the full one (morph) and a subsampled one (morph300). See Figure 4.1 for illustration of their respective distributions of examples for ages.

As the morph dataset in many cases contains several photos of a single individual, it is split into 10 non-overlapping folders, where no two folders contain a photo of the same individual. Therefore, one can easily split the dataset into training, validation, and testing parts without photos of the same individual leaking from the training part into the rest.

The subsampled version (morph300) contains only three folders (preserving the above mentioned properties), while adding the property that the distribution of examples across ages is uniform – exactly 300 per age category in each fold. A dataset with a uniform distribution reduces the complexity of the evaluation process.

4.2 Synthetized examples

The grouping process proposes tuples of faces which are eligible for new face synthesis under the defined conditions, such as: same age, gender, similar pose, etc. Unfortunately, there are still many reasons why the synthesis might fail and produce a new face with artifacts that are easily noticeable by a human eye. Too different skin tone/lighting, dissimilar face part shape or hair/glasses intrusion at the boundary of a face part are the most common ones. Figure 4.2 shows examples of such failures happening when synthetizing

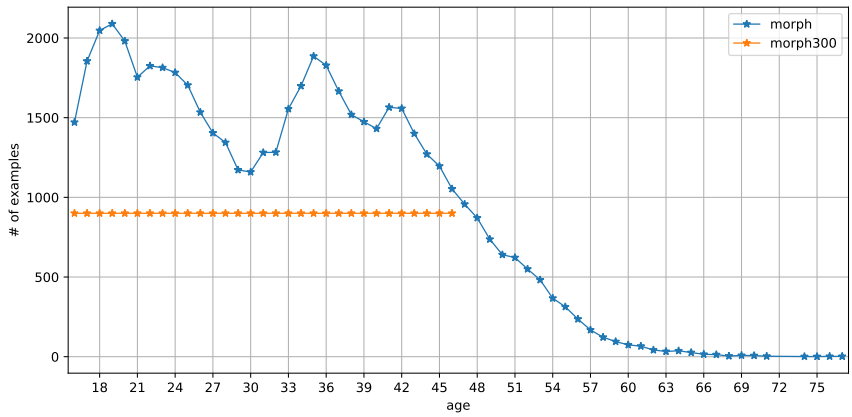


Figure 4.1: Distributions across ages for the full and subsampled morph datasets

images from the morph dataset, while Figure 4.3 shows successful examples even in more challenging situations.

4.3 Age Estimation Task

We frame the age estimation problem as a standard classification task. Given a grayscale input image $\boldsymbol{x} \in [-1, 1]^{64\times64}$ we use CNN to estimate the conditional probability distribution $\Pr(y \mid \boldsymbol{x})$, $y \in \mathcal{C}$, where $\mathcal{C}$ is a set of all possible ages. Having the conditional probability $\Pr(y \mid \boldsymbol{x})$, the age is estimated by the plug-in Bayes rule

$$\hat{y} = \arg\min_{y} \mathcal{R}(y \mid \boldsymbol{x}) \tag{4.1}$$

where

$$\mathcal{R}(y \mid \boldsymbol{x}) = \sum_{y' \in Y} \Pr(y' \mid \boldsymbol{x}) \mathcal{L}_{\text{pred}}(y', y) \tag{4.2}$$

is the conditional risk, and

$$\mathcal{L}_{\text{pred}}(y', y) = |y' - y| \tag{4.3}$$

is the target loss function, i.e. we aim to minimize the absolute error.

4.4 Evaluation metrics

In our experiments we use two evaluation metrics common in age prediction: the Mean Absolute Error (MAE), and the Cumulative Score at 5 (CS5). We also use a modification of the MAE: the Per Class Mean Absolute Error (PCMAE) which we introduce to compensate for a highly non-uniform age distribution present in the datasets. Given an evaluation set $\mathcal{S} = \{(y_i, \hat{y}_i) \mid i \in [1\,..\,n]\}$ of n pairs $(y_i, \hat{y}_i)$ of ground truth age (y_i) and the predicted age $(\hat{y}_i)$, we further describe the evaluation metrics more in detail.

(a): Skin tone/lighting mismatch

(b): Region shape misalignment (visible stretching/squeezing)

(c): Hair intrusion at the boundary of a face part

Figure 4.2: Examples of different types of failure cases. The synthetic face is generated by the proposed method from 4-tuples of images, from left to right: source images for the (whole face part, eyes part, mouth part), **the synthetized image (in GREEN)**, and the destination image

4.4.1 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) metric captures the mean absolute difference between the predicted and the ground truth age:

$$\mathrm{MAE}(\mathcal{S}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{4.4}$$

In other words, we obtain an estimate on how much off our prediction would be (on average), if we were to draw new examples from the same distribution as was used for the evaluation.

4.4.2 Cumulative Score 5 (CS5)

Another common evaluation metric is the Cumulative Score at 5 (CS5). This metric is equal to the percentage of examples whose predicted age is at most 5 years away from the ground truth age:

$$\mathrm{CS5}(\mathcal{S}) = \frac{1}{n}\sum_{i=1}^{n}[\![|y_i - \hat{y}_i| \leq 5]\!], \tag{4.5}$$

(a): Skin patch handled well

(b): Very different skin tone, hair nearby

(c): Large yaw angle, different skin tones

Figure 4.3: Success cases in more challenging scenarios, from left to right: source images for the (whole face part, eyes part, mouth part), **the synthetized image (in GREEN)**, and the destination image

where $[\![A]\!]$ is the Iverson bracket which is 1 if A is true and 0 otherwise. Intuitively, it is the percentage of faces on which the prediction is satisfactory. Similarly to the MAE, this metric also depends on the age distribution of the evaluation dataset.

4.4.3 Per Class Mean Absolute Error (PCMAE)

Since the age distributions of the examples in the datasets we experimented with are often concentrated around middle ages and other categories are poorly represented, we came up with a modification of the MAE metric which tries to compensate for that. We named it the Per Class Mean Absolute Error (PCMAE). To calculate its value, one has to partition the evaluation set into $|\mathcal{C}|$ partitions $\mathcal{S}_c$ where C is the set of all age categories:

$$\mathcal{S}_c = \{(y_i, \hat{y}_i) \mid y_i = c, i \in [1\,..\,n]\}\,. \tag{4.6}$$

Using $\mathcal{S}_c$ partitions, one can calculate the PCMAE metric:

$$\text{PCMAE}(\mathcal{S}) = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}}\text{MAE}(\mathcal{S}_c) \tag{4.7}$$

$$= \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{S}|}\frac{1}{|\mathcal{S}_{y_i}|}|y_i - \hat{y}_i|. \tag{4.8}$$

It is computed as an average of MAEs calculated for each $\mathcal{S}_c$ (eq. (4.7)), which turns out to be a weighted MAE (eq. (4.8)). When the distribution of examples among classes is uniform, i.e. $|\mathcal{S}_1| = |\mathcal{S}_2| = \cdots = |\mathcal{S}_{|\mathcal{C}|}|$, then all weights become equal and $\text{MAE}(\mathcal{S}) = \text{PCMAE}(\mathcal{S})$. Unlike MAE and CS5, the PCMAE is a metric which does not dependent on the age distribution.

4.5 Model architecture

We experimented with several CNN architectures, including fairly simple custom CNN architectures, as well as the well known ResNet [16] architectures.

The advantage of more complex and deep models is their higher expressivity. They are able to accomplish various hard, image-related tasks with great performance. However, this doesn't come for free. Such models require vast amounts of data and computation time to train, especially when trained from scratch.

As we don't have the necessary amount of data to train such models properly, they ended up overfitting the training dataset. Therefore, we also experimented with using pretrained models and only fine-tuning them, e.g. by replacing the last layer to meet our needs and training only last l layers with decreased learning rate. This approach yielded better baseline results, however it also introduced influence from the pretraining dataset and other variables which might affect our experiments.

Therefore, we decided to use only a simpler CNN architecture which empirically works well for age prediction task and is not that resource consuming to train. This simpler architecture still has over 2 million trainable parameters. Table 4.1 lists all its details.

4.6 Image preprocessing and augmentation

As described in the previous Section, our model expects a fixed-size grayscale image on its input with a centered face. To achieve this, we first fit a rectangular boundary so that it tightly encapsulates the detected facial landmarks. Then, we symetrically enlarge the shorter side to make the boundary square. Once it is square, we inflate it by 25 %, crop the image to it and resize the cropped image to the desired size (64×64). Finally, the image gets converted to grayscale and its pixel values get normalized from $[0\,..\,255]$ to $[-1, 1]$.

Layer type	Configuration
Input	64×64 grayscale image
Conv+BN+ReLU	filters: 32, kernel: 3×3, stride: 1, padding: 0
Conv+BN+ReLU	filters: 32, kernel: 3×3, stride: 1, padding: 0
MaxPool	
Conv+BN+ReLU	filters: 64, kernel: 3×3, stride: 1, padding: 0
MaxPool	
Conv+BN+ReLU	filters: 64, kernel: 3×3, stride: 1, padding: 0
Dropout	$p = 0.5$
MaxPool	
Conv+BN+ReLU	filters: 128, kernel: 3×3, stride: 1, padding: 0
Dropout	$p = 0.5$
Conv+BN+ReLU	filters: 256, kernel: 3×3, stride: 1, padding: 0
Dropout	$p = 0.5$
(Flatten)	
FC+BN+Relu	input: 256, output: 1024
FC+BN+Relu	input: 1024, output: 1024
FC+Softmax	input: 1024, output: 31 (morph300) or 62 (morph)

Table 4.1: The used CNN architecture, the output dimensionality is either 31 (for morph300 dataset) or 62 (for morph dataset)

4.7 Model training

We train parameters of the CNN using the standard cross-entropy loss which corresponds to the maximum-likelihood estimate of the conditional probabilities $\Pr(y \mid \boldsymbol{x})$. We use the Adam optimizer [20] with its default settings (i.e. learning rate $= 0.001$).

The dataset is always split into three (training, validation, and testing) parts for training. That way we can monitor the performance of the model on unseen data during training. Specifically, we calculate the PCMAE metric on the validation set after each epoch and keep the best performing model. We (early-)stop the training process when the validation PCMAE does not improve for 75 consecutive epochs.

To utilize the available data better, we apply some of the common augmentation methods. Specifically, during training each image gets randomly rotated by up to $\pm 10°$, randomly translated by up to ± 3px both horizontally and vertically, and horizontally flipped with 50 % probability.

4.8 Evaluation protocol

As described more in detail in Section 4.1, both the morph and the morph300 datasets consist of several folders. To have a better estimation of the resulting metrics, we perform a version of the K-fold cross-validation in our experiments. We perform 3-fold cross-validation on the morph300 dataset and 10-fold cross-validation on the morph dataset in all experiments. Table 4.2 summarizes this in more detail.

dataset	training folders	validation folders	testing folders	# folds
morph300	1	1	1	3
morph	7	2	1	10

Table 4.2: Cross-validation folds

4.9 Varying the number of training examples

We aimed to verify that increasing the number of training examples improves the model's performance and that it would benefit from using additional, possibly synthetically generated, training examples. We subsampled the training dataset at various fractions to simulate the effect of adding real training examples. The subsampling preserves the age distribution of the full training dataset of each folder. This experiment was performed on both datasets.

Unfortunately, the MORPH dataset in many cases contains multiple photos of the same individual. Hence, subsampling at various fractions of the number of training examples doesn't necessarily lead to the same fraction of identities present, i.e. adding plenty of training examples might lead to adding only a few new identities.

As our method aims at producing examples of new synthetically generated identities, we decided to conduct a different version of this experiment, which ensures that with adding new training examples the number of identities increases as well. Instead of subsampling, we vary the number of folders the training dataset is made of. This is not possible for the morph300 dataset, as there are not enough folders to work with (see Table 4.2).

Tables 4.3 and 4.4 summarize the average number of training examples the model was trained with when the training dataset was subsampled using the fractions and folders methods, respectively.

	10 %	15.8 %	25.1 %	39.8 %	63.1 %	100 %
morph300	930	1474	2336	3702	5868	9300
morph (±6.3 %)	3856.7	6112.3	9687.4	15353.6	24333.8	38566.5

Table 4.3: Average training dataset sizes at various fractions; size of the morph dataset varies across folds, hence the standard deviation

# of folders	morph
1	$5509.5 \pm 21.9\,\%$
2	$11019.0 \pm 18.2\,\%$
3	$16528.5 \pm 14.7\,\%$
4	$22038.0 \pm 12.3\,\%$
5	$27547.5 \pm 10.2\,\%$
6	$33057.0 \pm 8.2\,\%$
7	$38566.5 \pm 6.3\,\%$

Table 4.4: Average training dataset sizes at increasing number of folders

Both Figures 4.4 and 4.5 show improving prediction error as the number of examples increases. Moreover, the endings of the trends suggest that the model would benefit from having even more training examples. Note that the error values are incomparable between the morph300 and the morph datasets, as they have different age ranges.

We also see from Figure 4.5 that if we have two equally-sized training datasets, the model trained on the one which contains more identities will have more acurate predictions on the testing data. In other words, adding examples of novel identities helps more than adding examples of the identities already present.

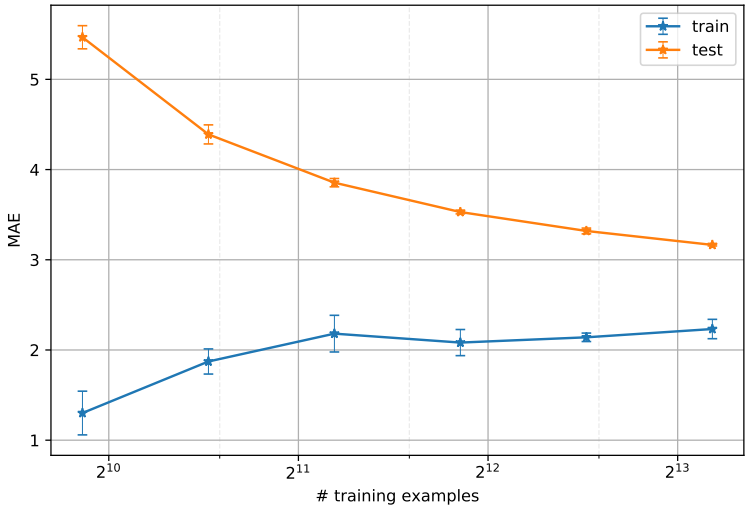


Figure 4.4: Training and testing MAE achieved on the subsampled morph300 dataset

4.10 Grid search

As discussed in Chapter 3, our method has multiple parameters which affect how new faces are synthetized – one can choose which face part(s) will be transferred, control the bitmap transfer parameter (α) and the shape transfer

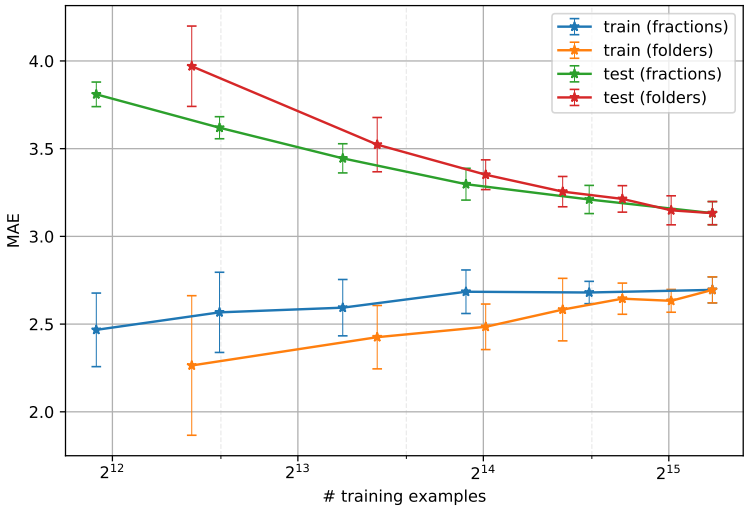


Figure 4.5: Training and testing MAE achieved on the subsampled morph dataset using two methods of subsampling (fractions and folders); *fractions* corresponds to adding new examples of identities that might have already been in the training set, and on the other hand, *folders* corresponds to adding examples of new identities, none of which have been in the training set

parameter (β). No less important variable is the number of training examples generated.

We decided to perform an exhaustive evaluation of all combinations of parameter values (i.e. a grid search) to analyze how they influence the model's performance and to ultimately find the best one for our setting.

Thanks to its uniform distribution and manageable computation demands, only the morph300 dataset was used in this experiment. In total we generated over 3 million synthetic images for this experiment.

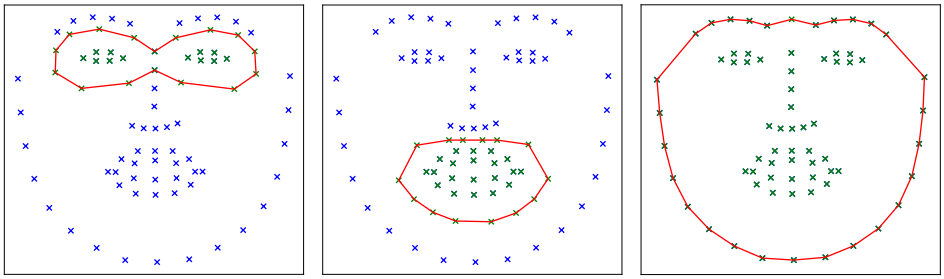
4.10.1 Face parts

We defined three face parts for this grid search: the Whole Face (W) part, the Eyes (E) part, and the Mouth (M) part. As the naming suggests, the Whole Face part transfers the whole face, i.e. the inner region defined by the outermost points. The Eyes and the Mouth part both transfer the corresponding face part extended by some margin. Figures 4.6a, 4.6b, and 4.6c show the respective face parts.

All possible combinations of the face parts were tried in the grid search.

4.10.2 Bitmap and shape transfer parameters

Both parameters have limited interval of possible values: $\alpha, \beta \in [0, 1]$, where values closer to 0 mean smaller change in the resulting image. All the combinations under and including the antidiagonal shown in Figure 3.8 were tried in the grid search.



(a): Eyes part **(b):** Mouth part **(c):** Whole Face part

Figure 4.6: All defined face parts for the grid search

4.10.3 Number of examples generated

As Figure 4.4 suggests, it is necessary to at least double the number of training examples to obtain measurable improvements. Therefore, we tried doubling and tripling the training dataset size in the grid search.

4.10.4 Results

Table 4.5 summarizes all the values tried in the grid search.

Unfortunately, the results in Table 4.6 suggest that there is no configuration of parameters which would give us a significant improvement over the baseline. In other words, no matter how we generate synthetic examples using our method, adding them to the training dataset has no or very little effect on the model's performance.

We chose one scheme (face parts = W+M, $\alpha = 1$, $\beta = 1$; in Table 4.6 highlighted in blue) which we examine further in the subsequent experiments. Extending the training dataset using synthetic examples generated using this scheme has one of the highest improvements on both the validation and on the testing MAE when the standard deviation is taken into account. More specifically, the MAE dropped on the validation set by 0.067 ± 0.034 and on the testing set by 0.061 ± 0.032, which is roughly a 2 % improvement.

As the morph300 dataset is uniform in the age distribution, the PCMAE values are equal to the MAE values and are thus omitted. We also delay showing CS5 values to further experiments.

Parameter name	possible values
Face parts	E; M; W; E+M; W+E; W+M; W+E+M
α, β	$(0, 1)$; $(1, 0)$; $(0.5, 0.5)$; $(0.5, 1)$; $(1, 0.5)$; $(1, 1)$
Amount of data generated	100 %; 200 %

Table 4.5: Grid-searched parameters together with their searched values

Face parts	α	β	%	train MAE	**val MAE**	test MAE
morph300	–	–	–	2.23 ± 0.11	3.14 ± 0.04	3.16 ± 0.01
E	0.0	1.0	100	−0.12 ± 0.20	−0.00 ± 0.06	−0.01 ± 0.02
E	0.0	1.0	200	−0.13 ± 0.26	0.02 ± 0.06	0.02 ± 0.02
E	0.5	0.5	100	−0.25 ± 0.16	−0.00 ± 0.08	−0.01 ± 0.03
E	0.5	0.5	200	−0.12 ± 0.16	−0.01 ± 0.07	0.00 ± 0.02
E	0.5	1.0	100	−0.12 ± 0.17	−0.01 ± 0.06	−0.02 ± 0.03
E	0.5	1.0	200	−0.17 ± 0.14	0.02 ± 0.07	−0.00 ± 0.04
E	1.0	0.0	100	0.01 ± 0.17	0.01 ± 0.05	−0.01 ± 0.03
E	1.0	0.0	200	−0.17 ± 0.11	0.00 ± 0.06	0.00 ± 0.06
E	1.0	0.5	100	−0.09 ± 0.12	−0.02 ± 0.06	−0.01 ± 0.04
E	1.0	0.5	200	−0.09 ± 0.22	0.01 ± 0.08	−0.00 ± 0.06
E	1.0	1.0	100	−0.03 ± 0.11	−0.01 ± 0.06	−0.02 ± 0.03
E	1.0	1.0	200	−0.10 ± 0.12	−0.01 ± 0.07	0.01 ± 0.05
M	0.0	1.0	100	−0.08 ± 0.14	−0.01 ± 0.05	−0.03 ± 0.05
M	0.0	1.0	200	−0.14 ± 0.11	−0.00 ± 0.06	−0.01 ± 0.02
M	0.5	0.5	100	−0.14 ± 0.17	−0.03 ± 0.06	−0.03 ± 0.07
M	0.5	0.5	200	−0.19 ± 0.17	−0.02 ± 0.08	−0.03 ± 0.07
M	0.5	1.0	100	−0.08 ± 0.20	−0.01 ± 0.06	−0.02 ± 0.01
M	0.5	1.0	200	−0.16 ± 0.17	−0.02 ± 0.08	−0.03 ± 0.04
M	1.0	0.0	100	−0.07 ± 0.15	−0.01 ± 0.06	−0.02 ± 0.04
M	1.0	0.0	200	−0.23 ± 0.12	−0.01 ± 0.06	−0.02 ± 0.04
M	1.0	0.5	100	−0.20 ± 0.13	−0.02 ± 0.06	−0.03 ± 0.04
M	1.0	0.5	200	−0.17 ± 0.14	−0.02 ± 0.06	−0.03 ± 0.03
M	1.0	1.0	100	−0.10 ± 0.11	−0.03 ± 0.08	−0.02 ± 0.03
M	1.0	1.0	200	−0.13 ± 0.11	−0.03 ± 0.04	−0.02 ± 0.07
W	0.0	1.0	100	−0.09 ± 0.12	−0.04 ± 0.05	−0.03 ± 0.05
W	0.0	1.0	200	−0.00 ± 0.13	0.01 ± 0.06	−0.03 ± 0.04
W	0.5	0.5	100	−0.22 ± 0.12	−0.01 ± 0.05	−0.03 ± 0.04
W	0.5	0.5	200	−0.24 ± 0.13	−0.02 ± 0.05	−0.01 ± 0.03
W	0.5	1.0	100	−0.10 ± 0.16	−0.03 ± 0.06	−0.04 ± 0.02
W	0.5	1.0	200	−0.09 ± 0.17	−0.02 ± 0.07	−0.03 ± 0.04
W	1.0	0.0	100	−0.02 ± 0.16	−0.03 ± 0.05	−0.06 ± 0.02
W	1.0	0.0	200	0.07 ± 0.11	−0.04 ± 0.06	−0.06 ± 0.02
W	1.0	0.5	100	−0.01 ± 0.16	−0.04 ± 0.06	−0.06 ± 0.03
W	1.0	0.5	200	0.04 ± 0.25	−0.05 ± 0.07	−0.06 ± 0.03
W	1.0	1.0	100	−0.01 ± 0.12	−0.05 ± 0.06	−0.06 ± 0.02
W	1.0	1.0	200	−0.01 ± 0.11	−0.07 ± 0.06	−0.09 ± 0.04
E+M	0.0	1.0	100	−0.12 ± 0.14	−0.00 ± 0.05	−0.01 ± 0.02
E+M	0.0	1.0	200	−0.14 ± 0.12	−0.01 ± 0.07	−0.03 ± 0.04
E+M	0.5	0.5	100	−0.11 ± 0.11	−0.01 ± 0.05	−0.02 ± 0.02
E+M	0.5	0.5	200	−0.23 ± 0.12	−0.02 ± 0.06	−0.04 ± 0.01
E+M	0.5	1.0	100	−0.07 ± 0.14	−0.01 ± 0.06	−0.03 ± 0.04
E+M	0.5	1.0	200	−0.17 ± 0.16	−0.01 ± 0.06	−0.03 ± 0.02
E+M	1.0	0.0	100	−0.00 ± 0.12	−0.03 ± 0.05	−0.03 ± 0.03
E+M	1.0	0.0	200	−0.10 ± 0.13	−0.03 ± 0.05	−0.03 ± 0.02
E+M	1.0	0.5	100	−0.04 ± 0.12	−0.04 ± 0.06	−0.06 ± 0.04
E+M	1.0	0.5	200	−0.04 ± 0.12	−0.02 ± 0.07	−0.04 ± 0.03
E+M	1.0	1.0	100	−0.06 ± 0.12	−0.04 ± 0.08	−0.03 ± 0.03
E+M	1.0	1.0	200	−0.13 ± 0.22	−0.02 ± 0.06	−0.04 ± 0.05
W+E	0.0	1.0	100	−0.10 ± 0.14	−0.03 ± 0.06	−0.03 ± 0.03
W+E	0.0	1.0	200	−0.14 ± 0.12	−0.03 ± 0.09	−0.03 ± 0.05
W+E	0.5	0.5	100	−0.19 ± 0.15	−0.00 ± 0.07	−0.02 ± 0.05
W+E	0.5	0.5	200	−0.21 ± 0.11	−0.00 ± 0.06	−0.00 ± 0.02

Face parts	α	β	%	train MAE	**val MAE**	test MAE
morph300	–	–	–	2.23 ± 0.11	3.14 ± 0.04	3.16 ± 0.01
W+E	0.5	1.0	100	−0.11 ± 0.17	−0.02 ± 0.06	−0.03 ± 0.02
W+E	0.5	1.0	200	−0.18 ± 0.17	−0.02 ± 0.08	−0.02 ± 0.02
W+E	1.0	0.0	100	0.02 ± 0.15	−0.04 ± 0.06	−0.05 ± 0.04
W+E	1.0	0.0	200	0.07 ± 0.19	−0.03 ± 0.06	−0.05 ± 0.04
W+E	1.0	0.5	100	−0.01 ± 0.19	−0.05 ± 0.05	−0.03 ± 0.06
W+E	1.0	0.5	200	−0.07 ± 0.16	−0.05 ± 0.07	−0.06 ± 0.05
W+E	1.0	1.0	100	−0.04 ± 0.17	−0.07 ± 0.05	−0.04 ± 0.02
W+E	1.0	1.0	200	−0.06 ± 0.29	−0.05 ± 0.06	−0.09 ± 0.02
W+E+M	0.0	100.0	1	−0.09 ± 0.12	−0.03 ± 0.07	−0.03 ± 0.03
W+E+M	0.0	100.0	2	−0.11 ± 0.12	−0.00 ± 0.07	−0.02 ± 0.04
W+E+M	0.5	000.5	1	−0.20 ± 0.14	−0.03 ± 0.05	−0.02 ± 0.05
W+E+M	0.5	000.5	2	−0.18 ± 0.12	−0.01 ± 0.07	0.00 ± 0.04
W+E+M	0.5	100.0	1	−0.14 ± 0.12	−0.02 ± 0.06	−0.03 ± 0.07
W+E+M	0.5	100.0	2	−0.22 ± 0.14	−0.01 ± 0.08	−0.05 ± 0.04
W+E+M	1.0	000.0	1	0.06 ± 0.11	−0.04 ± 0.08	−0.05 ± 0.04
W+E+M	1.0	000.0	2	−0.01 ± 0.12	−0.04 ± 0.08	−0.05 ± 0.04
W+E+M	1.0	000.5	1	−0.05 ± 0.15	−0.03 ± 0.07	−0.04 ± 0.02
W+E+M	1.0	000.5	2	−0.06 ± 0.14	−0.05 ± 0.06	−0.07 ± 0.03
W+E+M	1.0	100.0	1	0.08 ± 0.11	−0.04 ± 0.06	−0.04 ± 0.02
W+E+M	1.0	100.0	2	0.01 ± 0.22	−0.04 ± 0.07	−0.06 ± 0.06
W+M	0.0	1.0	100	−0.03 ± 0.21	−0.01 ± 0.07	−0.03 ± 0.01
W+M	0.0	1.0	200	−0.10 ± 0.19	−0.02 ± 0.06	−0.03 ± 0.04
W+M	0.5	0.5	100	−0.08 ± 0.12	−0.02 ± 0.07	−0.03 ± 0.04
W+M	0.5	0.5	200	−0.07 ± 0.13	−0.00 ± 0.06	−0.01 ± 0.02
W+M	0.5	1.0	100	−0.16 ± 0.19	−0.02 ± 0.06	−0.04 ± 0.02
W+M	0.5	1.0	200	−0.12 ± 0.17	−0.00 ± 0.07	−0.02 ± 0.03
W+M	1.0	0.0	100	0.09 ± 0.15	−0.04 ± 0.07	−0.04 ± 0.03
W+M	1.0	0.0	200	0.03 ± 0.11	−0.03 ± 0.08	−0.04 ± 0.06
W+M	1.0	0.5	100	0.05 ± 0.20	−0.03 ± 0.08	−0.06 ± 0.05
W+M	1.0	0.5	200	−0.01 ± 0.14	−0.04 ± 0.06	−0.04 ± 0.03
W+M	1.0	1.0	100	0.02 ± 0.19	−0.04 ± 0.06	−0.04 ± 0.03
W+M	1.0	1.0	200	−0.01 ± 0.16	−0.07 ± 0.05	−0.06 ± 0.03

Table 4.6: Grid search on morph300 and the resulting MAE values; E = Eyes, M = Mouth, and W = Whole face; the chosen scheme for further experiments is in blue; standard deviations of the difference rows were calculated assuming independency of the subtracted variables

4.11 Extending the morph dataset

In the previous experiment we found the most promising scheme to use when generating synthetic images from the morph300 dataset. In this experiment we take the same scheme and use it for generating synthetic examples from the morph dataset.

4.11.1 Training data

Again, we tried to double and triple the amount of training data. Figure 4.7 shows the average age distribution of the training datasets. Note we were not able to exactly triple the amount of training data as there were not enough suitable 3-tuples in some age categories.

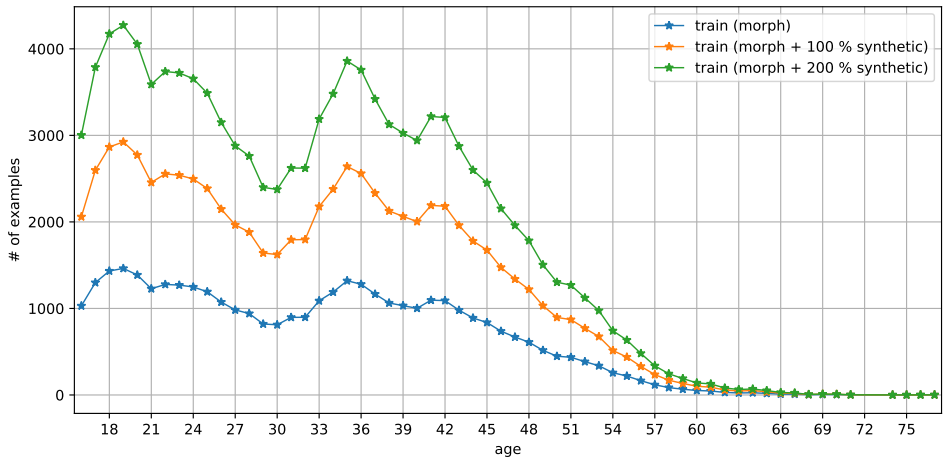


Figure 4.7: Age distributions of the (extended) morph dataset

4.11.2 Results

To assess the achieved model's performance, we plot all three measured metrics (MAE, CS5, and PCMAE) with varying number of training examples (shown in Figures 4.8a, 4.8b, and 4.8c, respectively). There, each curve starts off as a solid line. In this part the training dataset consists entirely of real examples. Once all of the available real training data is used up, the curve switches to a dashed style to signify the training datataset is extended by synthetic examples. This way, the Figures compare the model's performance on the extended datasets against the original morph dataset in the context of how the values developed when real training examples were added.

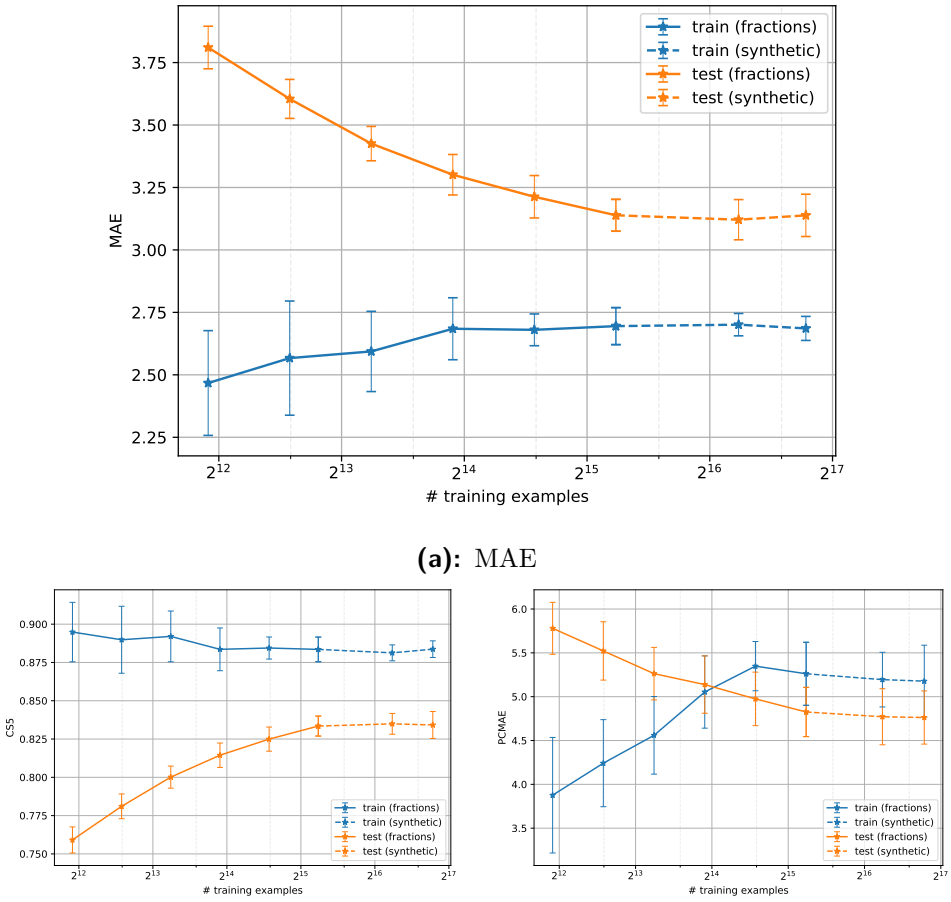
Unfortunately, the results are very similar to what the exhaustive search already suggested. All the curves of metrics evaluated on the testing datasets flatten once the dataset is extended with synthetic examples. Put differently, the model's overall performance seems unchanged by adding synthetic examples generated by our method.

4.11.3 Evaluation of MAE per age category

The measured metrics suggest the model does not change its performance overall. However, evaluation of a model's performance based on just a few numbers hides a lot of details and it still might be the case that the models behave differently e.g. for certain age categories, and that such differences cancel out when calculating the metrics.

Therefore, we decompose the overall (PC)MAE value by plotting MAE per each ground truth age category in Figure 4.9 to examine this.

Once again, even such fine-grained analysis tells the same story – the results of a model trained on the morph dataset is indistinguishable from the models trained on the extended ones.



(a): MAE

(b): CS5

(c): PCMAE

Figure 4.8: Training and testing performance achieved by extending the morph dataset using synthetic training examples (in dashed lines)

4.12 Examining the differences between real and synthetic examples

It seems to be the case that adding synthetic training examples generated by our method does not bring any new useful information to train from. To determine whether there is actually any difference from the model's perspective between real and synthetic examples, we performed the following experiment: instead of mixing together real and synthetic examples, we generated a full counterpart to the morph dataset made entirely of synthetic

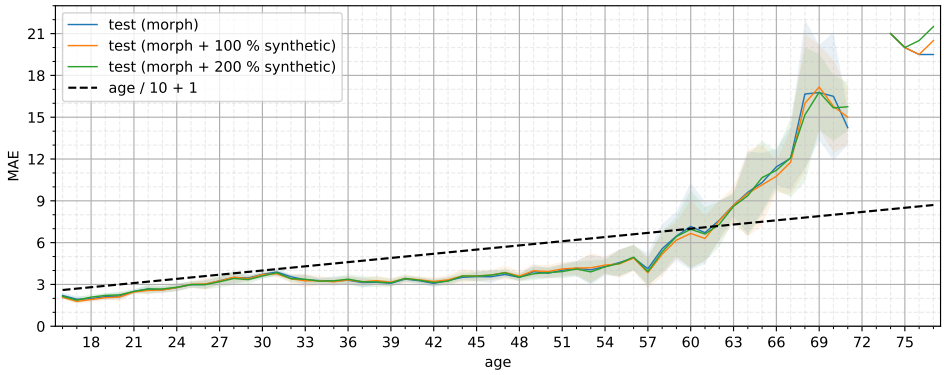


Figure 4.9: Comparison of MAE decomposition of baseline morph against morph with 100 % or 200 % synthetic training examples added

examples. Then, we trained our model the same way as before, both on the morph and this new, synthetic dataset. Once trained, we evaluated each model on both datasets again, thus obtaining four different evaluations to compare.

It is important to realize that while the synthetic dataset is different from the morph dataset, per se, examples present in it are still just combinations of the examples from the morph dataset. So, even though the model trained on the morph training dataset has never seen any of the synthetic training data, it has seen all the images the synthetic ones are stitched from. This is probably the reason, why the training and testing MAE values differ in row two of Table 4.7.

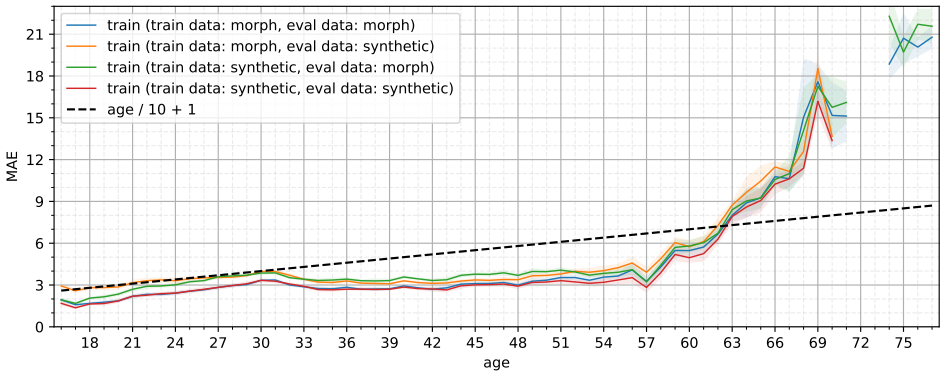
This is not always true in the opposite direction. It may happen that there are same images, which are not used as the source image (for some of the face parts) or as the destination image. In fact, in our setting it happens that there is only about 66 % of the morph images used as the source image for a face part, and about 97.5 % of the morph images used as the destination image. Therefore, there is always approximately 34 % of morph faces, which the model trained on the synthetic examples has never seen.

Having all the described steps performed, we can evaluate (through the model's performance) how the synthetic examples compare to the original ones. Table 4.7 summarizes the evaluation results. This time we also show the Mean Conditional Risk (MCR), which is just the conditional risk defined in eq. (4.2) averaged over examples from given dataset. The MCR can be interpreted as the model's prediction of the MAE it will achieve. Figures 4.10a, and 4.10b show the MAE decomposition for training and testing data, respectively.

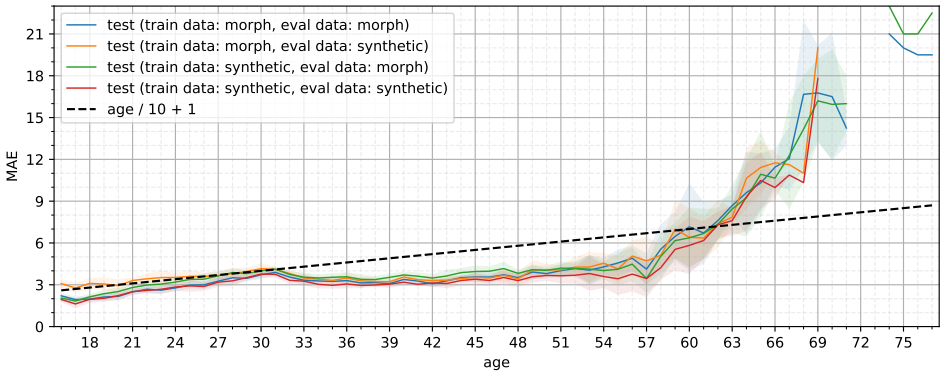
In the following subsections we discuss the results of the described experiment.

train	eval	train MAE	test MAE	train MCR	test MCR
morph	morph	2.69 ± 0.07	3.13 ± 0.07	3.04 ± 0.07	3.07 ± 0.06
morph	synthetic	3.33 ± 0.05	3.51 ± 0.08	3.21 ± 0.06	3.22 ± 0.06
synthetic	morph	3.24 ± 0.08	3.39 ± 0.08	2.97 ± 0.10	2.99 ± 0.12
synthetic	synthetic	2.63 ± 0.08	2.97 ± 0.07	2.99 ± 0.10	3.01 ± 0.11

Table 4.7: Evaluation results on four combinations of training and evaluation datasets; for example, the second row means the model was trained using the training (and validation) sets of the morph data, while the evaluation of the training and testing metrics was done using the respective training and testing sets of the synthetic data



(a): train



(b): test

Figure 4.10: Comparison of MAE decomposition of the four combinations of training and evaluation datasets; the solid lines correspond to row in Table 4.7

4.12.1 Scenarios where training and evaluation datasets are the same

These are the two scenarios where the model is trained and than evaluated on data coming from the same distribution, i.e. the usual way how model evaluation is done. More concretely, we are comparing the first and the last rows from Table 4.7.

Comparable, but still consistently lower MAE and MCR values achieved on the synthetic dataset tell us that the model is able to predict the synthetic data a bit more accurately and is ever so slightly more sure about its predictions. This in turn suggests that the synthetic data are somehow slightly easier to fit and to predict.

We supposed the reason for this might be lower variation of face poses in the synthetic dataset, caused by the necessity to find tuples of images with similar pose (as described in Section 3.4). However, it turned out that the variation of face poses is only slightly lower and even artificially reducing the variability of face poses in the morph dataset so that it matches kept the results nearly unchanged.

4.12.2 Evaluation data change perspective

In this subsection we compare how the metrics change when the training data remains fixed and the evaluation data varies, i.e. we keep the model and vary the evaluation data.

train	eval	train MAE	test MAE	train MCR	test MCR
morph	morph	2.69 ± 0.07	3.13 ± 0.07	3.04 ± 0.07	3.07 ± 0.06
morph	synthetic	+0.64 ± 0.09	+0.38 ± 0.10	+0.17 ± 0.09	+0.15 ± 0.08
synthetic	synthetic	2.63 ± 0.08	2.97 ± 0.07	2.99 ± 0.10	3.01 ± 0.11
synthetic	morph	+0.61 ± 0.11	+0.43 ± 0.11	−0.01 ± 0.14	−0.02 ± 0.16

Table 4.8: Differences in MAE and MCR caused by changing the evaluation dataset; rows one and three come from the Table 4.7, rows two and four are recalculated to show the difference against their respective rows above; standard deviations in the difference rows were calculated assuming independency of the subtracted variables; values in green are desired, values in red are undesired (we would like them to be as close to 0 as possible)

We see in Table 4.8 a desired increase in the training MAE when the evaluation dataset is different from the one the model was trained on (in green). This suggests that the synthetic data differ from the original ones (even though they are synthetised from them).

Unfortunately, we also see an (undesired, in red) increase in the testing MAE. This suggests the synthetic data are too different from the original ones. This might by one of the reasons why adding synthetic training examples does not help.

It is also worth noting, that the advantage of seeing all the original images the synthetic images are created from of the model trained on the morph

dataset does not result in lower training MAE increase. There are probably more factors at play.

We see a significant increase of MCR values for both training and testing data of the model coming from the morph data to the synthetic data. We hypothesize that it is this case, because there is some number of synthetic images with obvious artifacts. These might make the model more unsure as it was not presented with anything alike during training.

On the contrary, the MCR values of the model coming from the synthetic data to the morph data remain practically unchanged. We hypothesize the model retains its confidence, since there exists plenty of successfully synthesized images with appearance unrecognizable from the original images.

4.12.3 Training data change perspective

For the sake of completeness, we shortly discuss also the other perspective, in which we fix the evaluation data and change the training data. Please refer to Table 4.9.

train	eval	train MAE	test MAE	train MCR	test MCR
morph	morph	2.69 ± 0.07	3.13 ± 0.07	3.04 ± 0.07	3.07 ± 0.06
synthetic	morph	+0.54 ± 0.11	+0.26 ± 0.11	−0.07 ± 0.12	−0.08 ± 0.13
synthetic	synthetic	2.63 ± 0.08	2.97 ± 0.07	2.99 ± 0.10	3.01 ± 0.11
morph	synthetic	+0.71 ± 0.09	+0.54 ± 0.10	+0.22 ± 0.11	+0.21 ± 0.12

Table 4.9: Differences in MAE and MCR caused by changing the training dataset; rows one and three come from the Table 4.7, rows two and four are recalculated to show the difference against their respective rows above; standard deviations in the difference rows were calculated assuming independency of the subtracted variables; values in red are undesired (we would like them to be as close to 0 as possible)

Most importantly, we see in Table 4.9 an undesired (in red) increase of the testing MAE when the model is trained on the synthetic data (when compared to the model trained on the morph data).

Overall we see the model trained on the synthetic data being slightly more sure about its predictions. We are not sure why this is the case.

Chapter 5

Conclusions and future work

This work was motivated by the observation that adding new identities to the training dataset improves accuracy of the face recognition systems. In particular, this phenomenon is well known from learning identity recognition systems. We have seen the same effect when training age prediction models (Figure 4.5). This led us to the idea of generating new identities by combining face parts of different individuals with the hope that the generated examples will improve accuracy of age predicting models.

To this end, we introduced a novel face-specific augmentation method. The proposed method generates face images of virtual identities by stitching together face parts from a tuple of compatible photos, while controlling the shape and the appearance transfer. The result of the stitching is post-processed by Poisson editing [31] to achieve seamless, realistically looking face images. The generated images capture faces of a virtual identities of the same age (and gender) as the identities they were created from. The method was used to generate face images of virtual identities with known age from the MORPH database. The effect of using the generated faces in training and evaluation of the age predicting CNN was assessed in a series of experiments which have led to several surprising findings.

The main finding is that enhancing training dataset by synthetically generated identities does not improve accuracy of the age prediction system. It suggests that CNNs are able to learn that for the age prediction task, the correlations between face parts can be safely ignored. Hence synthetic examples obtained by permutation of face parts that have been already contained in the training set bring no new information for learning the CNN.

Other experiments (Section 4.12) suggest that distribution of the synthetically generated and the real faces are to a large extent replaceable when the faces are used for training and evaluation of the age predicting CNN. The differences observed when swapping training and evaluation set are slightly above the measurement error.

The most interesting research question is whether the same behaviour, i.e. that distribution of synthetically generated and real faces is replaceable, will be observed in case of other face recognition tasks. In particular, in case of tasks related to identity recognition, where the permutation of face parts should play an important role.

Other output of this thesis is an open source library written in Python for for generation of synthetic faces by seamless combination of face parts of different individuals. It utilizes several libraries for its inner workings, such as the Waldboost-based face detector [35], the 2D Face alignment library [5], and the OpenCV library [4]. The proposed approach is an alternative to nowadays popular GANs based face generators and as such we envision the community will find it useful in a number of applications.

Appendix A

Contents of the CD

- `DP_MN.pdf` – a pdf version of this thesis
- `DP_MN.zip` – a zip of source files of this thesis
- `src/` – a folder containing all the source codes used in this thesis

Appendix B

Bibliography

[1] G. Antipov, M. Baccouche, S.A. Berrani, and J.L. Duglay. Apparent age estimation from face images combining general and children-specialized deep learning models. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016.

[2] 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.

[3] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pages 404–417, Berlin, Heidelberg, 2006. Springer-Verlag.

[4] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[5] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017.

[6] Bor-Chun Chen, Chu-Song Chen, and Winston H. Hsu. Face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset. *IEEE Transactions on Multimedia*, 17:1–1, 06 2015.

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[8] B. Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[9] Ajit A. Diwan, Subir Kumar Ghosh, Partha P. Goswami, and Andrzej Lingas. On joint triangulations of two sets of points in the plane. *CoRR*, abs/1102.1235, 2011.

[10] D.Yi, Z.Lei, and S.Z.Li. Age estimation by multi-scale convolutional network. In *In proc of ACCV*, 2015.

[11] S Escalera X Baro I Guyon R Rothe. E Agustsson, R Timofte. Apparent and real age estimation in still images with deep residual regressors on appa-real database. In *12th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 2017*. IEEE, 2017.

[12] S. Escalera, M. Torres, B.Martinez, X. Bar, H.J. Escalante, I.Guyon, M.Oliu, and M.A.Bagheri. Chalern looking at people and faces of the world: Face analysis workshop and challenge. In *In IEEE CVPR Workshops*, 2016.

[13] F.Gurpinar, H.Kaya, H.Dibeklioglu, and A.A.Salah. Kernel elm and cnn based facial age estimation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2016.

[14] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[15] Bin-Bin Gao, Hong-Yu Zhou, Jianxin Wu, and Xin Geng. Age estimation using expectation of label distribution learning. In *Proc. The 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 2018.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[17] Yann N. Dauphin David Lopez-Paz Hongyi Zhang, Moustapha Cisse. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[18] Harold Hotelling. Relations Between Two Sets Of Variates. *Biometrika*, 28(3-4):321–377, 12 1936.

[19] I Huerta, C Fernandez, C Segura, J Hernando, and A. Prati. A deep analysis on age estimation. *Pattern Recognition*, 2015.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[21] A. Lanitis, C.J. Taylor, and T.F. Cootes. Toward automatic simulation of aging effects on face images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):442–455, 2002.

[22] S. Lapuschkin, A. Binder, and K.-R. Muller. Understanding and comparing deep neural networks for age and gender classification. In *Proceedings of the ICCV'17 Workshop on Analysis and Modeling of Faces and Gestures (AMFG)*, 2017.

[23] G. Levi and T.Hassner. Age and gender classification using convolutional neural networks. In *IEEE Computer Vision and Pattern Recongition Workshops*, 2015.

[24] Xin Liu, Shaoxin Li, Meina Kan, Jie Zhang, Shuzhe Wu, Wenxian Liu, Hu Han, Shiguang Shan, and Xilin Chen. Agenet: Deeply learned regressor and classifier for robust apparent age estimation. In *IEEE International Conference on Computer Vision Workshop (ICCVW)*, 2015.

[25] Iacopo Masi, Anh Tuân Trân, Tal Hassner, Gozde Sahin, and Gérard Medioni. Face-specific data augmentation for unconstrained face recognition. *International Journal of Computer Vision*, Apr 2019.

[26] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: the first manually collected, in-the-wild age database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop*, volume 2, page 5, 2017.

[27] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua. Ordinal regression with multiple output cnn for age estimation. In *In proc of CVPR*, 2016.

[28] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, July 2002.

[29] Hongyu Pan, Hu Han, Shiguan Shan, and Xilin Chen. Mean-variance loss for deep age estimation from a face. In *Proceedings of CVPR*, 2018.

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[31] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, July 2003.

[32] K. Ricanek and T. Tesafaye. Morph: a longitudinal image database of normal adult age-progression. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 341–345, April 2006.

[33] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision (IJCV)*, July 2016.

[34] S.Chen, C.Zhang, M. Dong, J. Le, and M. Rao. Using ranking-cnn for age estimation. In *In proc. of CVPR*, 2017.

[35] Jan Sochman and Jiri Matas. Waldboost - learning for time constrained sequential detection. In *Computer Vision and Patter Recognition*, 2005.

[36] X.Wang, R.Guo, and C.Kambhamettu. Deeply-learned feature for age estimation. In *IEEE Winter Conference on Applications of Computer Vision*, 2015.

[37] Song Yang Zhang, Zhifei and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[38] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *CoRR*, abs/1708.04896, 2017.