UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Ablation study of the approaches for kinship verification

Author:
Petro FRANCHUK

Supervisor:
Volodymyr KARPIV

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2021

Declaration of Authorship

I, Petro FRANCHUK, declare that this thesis titled, "Ablation study of the approaches for kinship verification" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"I'm smart enough to know that I'm dumb."

Richard Feynman

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Ablation study of the approaches for kinship verification

by Petro FRANCHUK

Abstract

The Kinship Verification aims to determine blood relativeness between people from visual data. This problem gains more attention from the research community during the last few years due to low human and machine performance on this task. Multiple attempts were taken to improve the performance of the Kinship Verification, but most of them are done under diverse experimental settings, and the results are benchmarked on different datasets. In our work, we conduct an ablation study to compare these improvements and determine whether different approaches are compatible with each other or should be applied separately. This ablation study consists in testing different feature extractors, embedding combinations, neural network hyperparameters and such data preprocessing techniques as data augmentation, face alignment and image normalization. This thesis showcases the importance of reach feature representation and efficient embedding combination for the overall accuracy of Kinship Verification.

Acknowledgements

I would like to thank my supervisor, Volodymyr KARPIV, for the invaluable advice and support during work on the thesis. I wish to show my appreciation to SoftServe for the given computational resources. Thank Ukrainian Catholic University for the great community, which was inspiring me during my entire studying. Finally, I am incredibly grateful to my family for their love.

Contents

List of Figures

List of Tables

List of Abbreviations

SOTA	**S**tate **O**f **T**he **A**rt
FC	**F**ully **C**onnected
CNN	**C**onvolutional **N**eural **N**etwork
NN	**N**eural **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
ReLU	**R**ectified **L**inear **U**nit

Dedicated to my parents...

Chapter 1

Introduction

Kinship recognition is a set of problems in the Computer Vision domain that aims to determine blood relativeness between people. Given images of the persons, the task is to determine whether they are relatives or not.

Potential applications of kinship recognition are in genetics, social media, entertainment, missing children and relatives search, border control and criminal investigation.

Also, kinship recognition tasks are of interest because of relatively poor human performance. In the research [9], authors report average human performance on a subset of the Cornell Kin dataset as 67.19%, with the lowest and highest accuracy of 50% and 90%, respectively. In [29] states that machines outperformed humans on the FIW dataset. In the work [3], authors also proved that machines beat humans on such datasets as Kin-Wild [22] and Uva Nemo Smile [8].

Even considering the fact that machines beat humans on the Kinship Verification task, still, performance of the AI-based methods is not perfect. For instance, SOTA performance on the biggest nowadays kinship dataset - FIW [29], on the 1-to-1 kinship verification task is 78% accuracy [28]. Indeed, such performance is a big obstacle for the usage of such algorithms in real life, especially in some critical applications. An example of such application could be a search of potential parents of the lost child, when quite a big chance of the misclassification (more than 20%) of the current approaches could lead to a bigger set of the candidates, which will need further additional testing via other methods. However, by developing more accurate methods, we can decrease that gap between the current state and the desired one and allow the usage of kinship recognition algorithms in everyday life.

During the last decade, researchers were attempting to improve the accuracy of the Kinship Verification methods in diverse ways and on different datasets. The issue of that is that separate improvements in different works can not be compared since they were done under different experimental settings. So, the main contributions of this work are:

- benchmark of different improvements, proposed in other works, on one dataset with the same experimental settings, so that they could be compared
- study of how each of the improvements influences the overall performance and which of their combinations performs better

All our code and models wights are available in the GitHub repository[1]. The rest of the work is organized as follows: in the next part of the Chapter 1 main Kinship Recognition problems are described; Chapter 2 provides a review of the existing methods for the Kinship Verification and highlights current problems in this field; Chapter 3 describes what was done during ablation study; Chapter 4 contains

[1] https://github.com/franchukpetro/kinship_verification

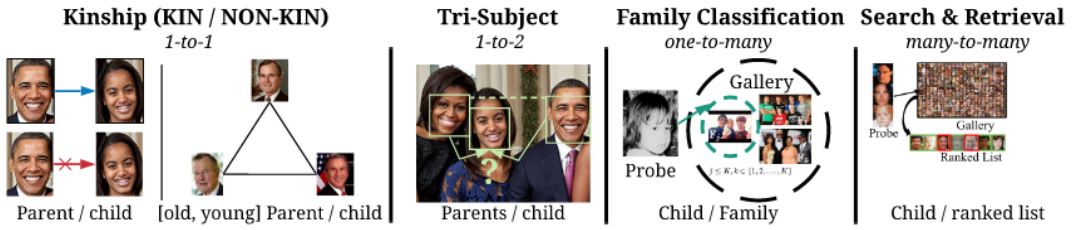

FIGURE 1.1: Kinship recognition problem types [30]

information about experimental setup and explains experiments and their results. Finally, Chapter 5 concludes our work.

1.1 Kinship recognition problems types

As already was mentioned, kinship recognition is a set of problems, each of which involves different inputs and tasks. This section briefly explains such problems as kinship verification, tri-subject verification, and search retrieval. For further research, the 1-to-1 task of kinship verification was selected, as it is most fundamental. In case of improvements in this task, the same improvements can be applied to other tasks.

1.1.1 Kinship verification

The task of kinship verification is to determine whether two persons are blood relatives. Input is defined as two images with faces, and output is a simple binary label, kin or non-kin. Besides two images, usually, kinship relation is given as prior knowledge for the task. In most datasets, there are such relation types as siblings (brother-brother (BB), sister-sister (SS) and brother-sister (SIBS)) and parent-child relations (mother-daughter (MD), mother-son (MS), father-son (FS), father-daughter (FD)), but there are also grandparent-grandchild relations (grandfather-grandson (GFGS), grandfather-granddaughter (GFGD), grandmother-grandson (GMGS) and grandmother-granddaughter(GMGD)), which are present, for instance, in FIW dataset[29][30].

1.1.2 Tri-subject verification

Tri-subject verification, instead of determining whether two persons are related, is focused on the relation of both parents and their child. As an input for this type of task, 3 images are provided: images of the father, mother and possible child (i.e. son or daughter). The task is to predict binary label whether this third one is really the child of the first two[30].

1.1.3 Search and retrieval

Search and retrieval is a many-to-many mapping task usually used for the missing children search. As input, few images of the lost child are provided, and the gallery, with pictures of all possible parents. The task is to obtain a ranked list of the persons from the gallery, where the top persons in that list will be the most possible parents of that lost child[30].

Chapter 2

Related work

As already was mentioned in the Chapter 1, this work focuses only on kinship verification, but not on the tri-subject verification or the search and retrieval tasks. The most common approaches to deal with kinship verification are metric learning, fine-tuning models, pretrained on the Face Recognition tasks, and their combination.

2.1 Metric learning

The main idea behind this method is to optimize metric between classes, in case of kinship verification - between kin and non-kin. It is quite a common approach in the Face Recognition field, where the faces of one person will create a cluster of the points in the embedding space, which can be easily separated from the cluster of another individual. On the contrary, in the kinship verification, this cluster should contain not only the faces of one person but also the faces of all family members, creating so-called family clusters, resulting in more complex task. In the last decade, researchers have made many efforts to learn such metric to create these family clusters. In [41] authors proposed discriminative multi-metric learning (DMML), where they learn multiple distance metrics from features, which they get from multiple face descriptors. In [39] was developed an approach based on the color features and extreme learning machines (ELM). In [19] authors developed a method named KVRL-fcDBN, which is currently one of the SOTA metric learning methods, resulting in 96% accuracy on the KinWild I & II datasets. Also, such methods as MHDL [24] and MvGMML [15] were proposed.

2.2 Transfer learning

Transfer learning is another group of approaches for kinship verification. As the problem setting of the kinship verification is very similar to the Face Recognition field, usually pretrained models are taken from it. There are several benchmark datasets in the FR field, such as VGGFace 1/2 [27][4], LFW [16] and MSCeleb[11], on which models were pretrained using some special loss (angular softmax in SphereFace[20], triplet loss in FaceNet[31], large margin cosine loss in CosFace[37] or additive angular margin loss in ArcFace[6]). These pretrained models are used as feature extractor, which will transfer faces from images to embedding space.

After feature extraction from the images, there are two ways to get a binary label (i.e. kin/non-kin): a) compute the distance between feature embeddings of two faces and tune threshold to produce binary label ; b) using Siamese network architecture(meaning two neural networks with shared weights), get embeddings of both of the images, fuse them and feed through few FC layers.

a) In [33] authors re-detected faces with RetinaFace[7] detector, aligned them and used the ArcFace model to produce image embeddings. The cosine distance between two face embedding was then used as a predicted probability of the kinship. Given approach results in SOTA performance in the kinship verification problem on the FIW dataset with 78% accuracy. In another work [21] was proposed to use multiple feature descriptors for different features extracted from the face image. After feature extraction, the euclidean distance between two face embeddings of the same feature descriptor is computed. All distances are then summed up and used as a final prediction score.

b) In [42] authors used the classical architecture of the Siamese network in their approach, with ResNet50[12] pretrained on VGGFace2 as a feature descriptor. Authors combined extracted embeddings in a few different ways, fused them into one embedding and then fed it into 2 fully connected layers. Approach from [23] is very similar to the previous one, except that now authors used two feature extractors - ResNet50 pretrained on the VGGFace2 and FaceNet. In another work [17], authors used ResNet50 pretrained with ArcFace loss for feature extraction and proposed to use a so-called kinship comparator - set of the local experts, each of which responsible for one kinship relation type (i.e. FS, FD, etc.).

2.3 Common issues with the kinship verification

During the literature review, we have noticed few problems with current approaches for kinship verification. Some of them are connected with data, and some with existing solutions at all.

2.3.1 Color and illumination bias

There are several benchmark datasets in the kinship recognition field now: CornellKin [9], KinFaceW I/II [22], FIW [29], UB Face [34] and many others. During data collection, a big part of the family members pairs were taken from the same photo, causing additional color and illumination bias. To prove that, researchers trained a simple CNN classifier, which was aiming to determine whether faces were cropped from the same photo [5]. They evaluated this model on different kinship recognition datasets and achieved the lowest accuracy on the FIW dataset (58.6%) and highest on the KinFaceW II (90.2%).

2.3.2 Age gap

In some works [30][40] authors mentioned that the more significant the age gap between two family members is, the less probability is that kinship will be detected. It also can be seen by a closer look at the accuracy for each type of relationship of SOTA methods - the worst accuracy is usually for the grandparent-grandchild pair type.

2.3.3 Influence of the race, gender and emotions

In the [13] was analysed the influence of gender on kinship verification accuracy. This work showed that both humans and machines perform better on the same gender pairs. It also can be confirmed by the fact that almost all SOTA kinship verification models perform better on the same gender pairs (i.e. father-son pair will be recognized better than father-daughter).

Ethnicity is another aspect that could influence the performance of kinship recognition. Different races could transmit other hereditary traits, which will result in different inherited visual traits.

Besides that, different emotions could also be an obstacle during kinship verification. For example, will the emotions be the same between relatives (even between mother and son)? Will visual features be critically influenced by totally different emotions on the images of a pair of family members (smiling mother and crying son)? That is rather a question to the classical Face Recognition field but should be considered while developing robust kinship recognition algorithms.

2.3.4 Prior knowledge of the kinship-relation types

Most current solutions assume kinship-relation type (i.e. FS, MS, BB, etc.) apriori. The reason for this is the architecture of the current solution. Previous researches proved that different relation types inherit different visual traits[34]. That is why it is common to use separate models for each kinship-relation type, resulting in a number of models same as a number of possible types (usually 11). The drawback of such an approach is applicability in real life, as it is very inefficient to have 11 separate models (in terms of the storage and resources for the training of all of them).

Besides this, many real-life applications will not satisfy such requirements and instead will require to predict the type of the kinship-relation.

Chapter 3

Problem statement

As a method for ablation study, the transfer-learning approach was selected due to the fact that all current state-of-the-art methods are based on it [28]. The most intuitive way to determine some relation between a pair of inputs is to use the Siamese network, which is why many recent papers are based on it. As was mentioned earlier, Siamese network architecture means using two sub-networks with shared weights, while each of two inputs is fed into a separate sub-network. In a kinship verification setting, those sub-networks are typically used as feature extractors [42][23][17] to encode high-dimensional image data into low-dimensional embeddings. After extraction, embeddings are combined and fused to one vector, which will then be fed into a shallow neural network to produce an output score (Figure 3.1). As a loss function to train whole method, Binary Cross-Entropy (BCE) loss was used, which is defined as

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i * \log(p(y_i)) + (1 - y_i) * \log(1 - p(y_i))$$

Existing solutions, which uses the Siamese network, differs mainly in feature extractors, feature combination methods and subsequent neural network architecture. In our work, we conduct an ablation study of these different parts of the Siamese network and propose the best choices for them.

Besides Siamese networks architecture, few works also pay attention to such data preprocessing techniques as data augmentation and face alignment. To verify whether these steps are helpful for kinship verification, some experiments with them were also conducted.

3.1 Feature extractor

To understand visual data, we should extract both low and high level features from the images. For this task, convolutional neural networks are widely used. They have a lot of capacity to encode spatial data into low-dimensional embeddings and can learn both low(e.g. lines, edges, etc.) and high (eyes, lips, nose in case of face recognition) level patterns.

For the kinship verification problem, such CNNs as ResNet50 or SeNet50 [14] are typically used as feature extractors. In [42] authors state that ResNet-50 architecture performs slightly better, while in [23], both CNN architecture got the same performance. Except for the feature extractor' architecture, dataset and loss function, which were used during model training, are important. The most common are VGGFace/VGGFace 2 dataset with triplet loss, FaceNet with its triplet loss and ArcFace's additive angular margin loss. Besides this, in [23] authors tried to use two feature extractors - VGGFace SeNet50 and FaceNet SeNet50.

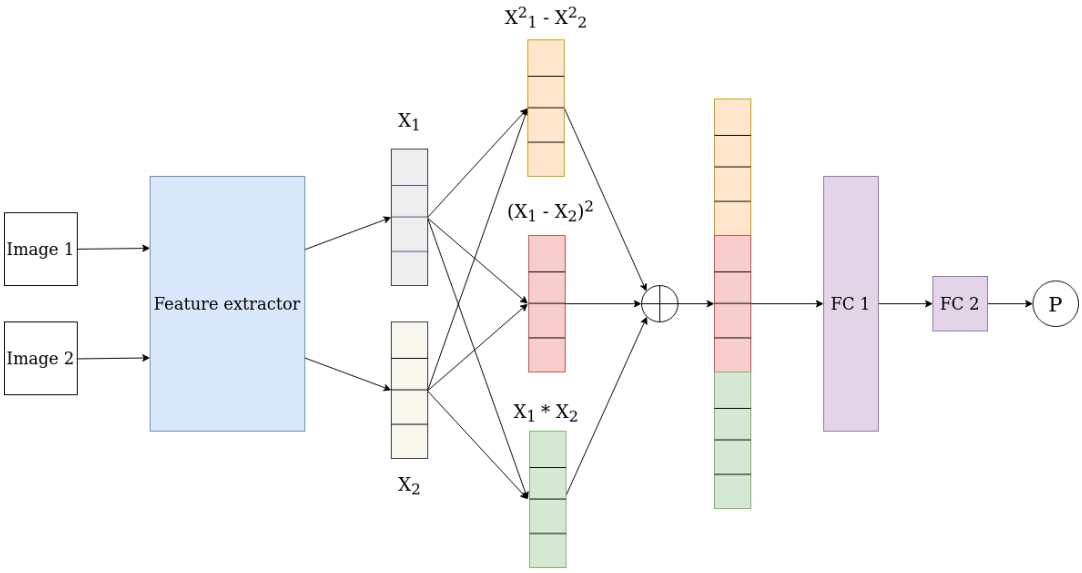

FIGURE 3.1: Example of the Siamese network architecture

Our work investigates how different feature extractors and their number influence the solution's overall performance. The idea behind this is that different models are pretrained on the different datasets and with different losses could result in quite different features extracted, each giving some more information about the image. As a baseline feature extractor, single ResNet50 pretrained on the VGGFace dataset was selected, but we have also tried feature extractors based on the FaceNet, ArcFace, OpenFace[2] and DeepID[35] models. Besides this, we also have conducted experiments with combinations of these feature extractors.

Also, it was tested whether the whole feature extractor should be fine-tuned, or just a few top layers will be sufficient. Such an experiment was done only with the VGGFace model because we assume that it will work identically for all other architectures.

3.2 Embedding combination

After extraction of the features from two input images, their embeddings are combined and fused. In [23] authors state that different combinations are needed to promote discrimitivity into the model, while in [42] authors state that fusion adds more nonlinearity and helps fully-connected layers to learn better similarity metrics. The number of combinations on this stage varies from method to method, but usually, it is from 3 to 5 different combinations. The combination itself can be any linear or nonlinear function, such as $x_1 - x_2$ or $(x_1 + x_2)^2$, where x_1 and x_2 are the embeddings of the first and second image respectively. However, in most works, authors used almost the same combinations:

- in [26] authors used such combinations as $x_1 + x_2$, $x_1 - x_2$, $x_1 * x_2$, $\sqrt{x_1} + \sqrt{x_2}$ and $x_1^2 + x_2^2$
- in [23] authors proposed $x_1^2 - x_2^2$, $(x_1 - x_2)^2$, $x_1 + x_2$, $x_1 - x_2$, $x_1 * x_2$, $\sqrt{x_1} - \sqrt{x_2}$, $\sqrt{x_1} + \sqrt{x_2}$
- in [42] $x_1 + x_2$, $x_1 - x_2$, $x_1 * x_2$, $x_1^2 - x_2^2$, $(x_1 - x_2)^2$ combinations were used

In our work, we compare the influence of different sets of these combinations on the overall solution performance.

3.3 Neural network architecture

After embeddings are extracted by feature descriptors and fused into one embedding vector, the last is fed into a Neural Network, which consists of fully connected layers. In [42] authors used two FC layers with 128 and 1 neuron respectively, while in [23] authors also used 2 FC layers but did not implicitly indicate the number of units in them.

To our knowledge, none of the existing works in the kinship verification field include some research on how NN architecture influences the accuracy of the whole system. By doing experiments with the number of fully connected layers and the number of units in each of them, we want to verify whether hyperparameters of the NN influence the overall performance.

3.4 Face alignment and cropping

The accuracy of the deep learning model highly correlates to the quality of the training data. In the case of the FIW dataset, some images are not perfectly cropped, which causes some additional noise from the background. As was already mentioned in the Chapter 2 , researchers proved the presence of the color and illumination bias caused by cropping faces of the family members from the same pictures.

In [36] authors proposed some additional image preprocessing to overcome this issue, named 'image normalization'. Given process includes face re-detection, cropping an ellipse-shaped region with the face and its further geometrical alignment and normalization. As stated in the paper, the authors used the Active Shape Models [25] technique to identify 76 facial landmarks, and an ellipse, which fit the 15 landmarks around the chin the best, was then used to crop the face (Figure 3.2).

FIGURE 3.2: Example of the image normalization from the [36]

More recently, authors of another work [33] used a similar process in their state-of-the-art solution, stating that better face detection and registration further improve model performance. Authors re-detect face by RetinaFace detector and, using obtained from its landmarks, align face (Figure 3.3). As authors released code for the detection and cropping of faces, we just used it in our experiments.

To implement 'image normalization', proposed in [36], we used HOG detector and pose estimator based on an ensemble of regression trees [18] from the dlib[1] Python library to obtain facial landmarks. The ellipse's width is defined as the distance between nose and cheek edge, while height - as the distance between nose and

[1] https://github.com/davisking/dlib

FIGURE 3.3: Example of the face alignment from the [33]

chin. Such cropping will work well on full faces (e.g. column 3 and 4 of the Figure 3.4), but for faces turned to the side, only part of the face will be cropped, while the redundant background will remain on the image (columns 1 and 2 of Figure 3.4). For such a case, more comprehensive ellipse selection is needed, which is beyond of the scope of our work.

FIGURE 3.4: Example of the implementation of the image normalization, described in the [36]

3.5 Data augmentations

The idea of data augmentation is to increase the training dataset and add some more variation to it by applying some transforms to the input data, such as rotation, flips, color jittering, etc. Although data augmentation is the default technique to improve deep learning model performance, we notice that just a few works report its usage in their solutions. In [23] was used blurring, flipping, contrast and brightness changes as data augmentation, while at [17] authors used horizontal flipping, contrast, brightness and saturation changes.

In the kinship verification problem, data augmentation can improve the performance of the deep learning models on the grandparent-grandchild pairs. This type of relationship is problematic because of two reasons: a small amount of data compared to other types of relationships and a big age gap between individuals. A

small amount of grandparent-grandchild pairs can be increased by applying different transforms to them, while to overcome the age gap problem, in [38] was proposed to use GANs [10] to rejuvenate the older face and decrease the age gap.

In our work, we could not conduct experiments with a bigger dataset or additional network for data augmentation due to computational constraints. Still, we have done experiments with different image data augmentation techniques to investigate whether they influence the overall performance.

Chapter 4

Experiments

4.1 Experimental setup

4.1.1 Dataset

During experiments, the Families in the Wild (FIW) dataset was used to train the models. To our knowledge, it is the biggest dataset today for the kinship verification task, and most of the latest works are based on it. FIW dataset contains almost 12 000 images from nearly 1 000 families.

Only part of the 5-fold variation of the FIW dataset was used due to computational constraints. Each fold contains image pairs that do not overlap with any other fold, so we used 10% of the 3rd fold (10 000 pairs) as our train set, 1% of the 2nd fold (1 000 images) as a validation set, and 10% of the 1st fold (8 700 pairs) as our test set.

4.1.2 Metrics

One of our metrics is a simple Accuracy score, defined as the ratio of correct predictions to all predictions. As a threshold value to binarize our predictions, we selected a value of 0.5.

As the accuracy score is dependent on the binarization threshold, which we assign to it manually, it can not fully represent the ability of the classification model to learn. That is why it was decided to also use AUC ROC score during our experiments to understand better the influence of the changes we made.

4.1.3 Implementation details

For neural network architecture implementation and training, we were using a Python framework named TensorFlow [1](version 1.15). NN models were trained and evaluated on the single NVIDIA T4 GPU.

Pretrained weights and architecture implementation for the VGGFace model were taken from Python package *keras-vggface*[1], while for rest models (FaceNet, ArcFace, OpenFace and DeepID) from package *deepface* [32].

For face alignment, described in [33], we used authors code, shared publically in their GitHub repository[2]. For image normalization[36] implementation Python' *dlib* package was used.

For tracking and visualization loss and metrics, we were using the TensorBoard toolkit.

[1] https://github.com/rcmalli/keras-vggface

[2] https://github.com/vuvko/fitw2020

4.2 Results

4.2.1 Feature extractor

As was already mentioned in the Chapter 3, such models as VGGFace, FaceNet, ArcFace, OpenFace, DeepID were tested in the role of the feature extractor.

Before comparing the performance of different models, an experiment to determine how models should be fine-tuned was conducted. During this test, one VGGFace feature extractor was fine-tuned completely, while in another VGGFace feature extractor, just 3 top layers were unfrozen and fine-tuned. Evaluation results (Table 4.1) showed that fine-tuning of few layers leads to worse performance, so for further experiments, it was decided to fine-tune the whole feature extractor.

Fine-tuning	Accuracy	AUC ROC
All layers	0.637	0.695
3 top layers	0.596	0.629

TABLE 4.1: Influence of the fine-tuning whole neural network and part of it

Our initial model is based on the work [42], where set of feature combination is defined as $x_1^2 - x_2^2 \bigoplus (x_1 - x_2)^2 \bigoplus x_1 * x_2$, where $\bigoplus$ denotes concatenation, x_1 and x_2 denotes features of the first and second input image respectively. Neural network consists of 2 fully connected layers: FC 1 with 128 units and ReLU activation; FC 2 with 1 unit and sigmoid activation function, which produce final probability value (Figure 3.1). During tests, nothing in the overall solution architecture, except the feature extractor, was changed. Each model was trained for the 100 epochs.

Feature extractor	Accuracy	AUC ROC
VGGFace	0.637	0.695
ArcFace	0.607	0.652
OpenFace	0.571	0.603
FaceNet	0.528	0.540
DeepID	0.488	0.488

TABLE 4.2: Influence of different single feature extractors on the whole method

Since in [23] and [26] authors used two feature extractors instead of single, it was also tested such pairs as VGGFace + FaceNet, VGGFace + ArcFace, VGGFace + OpenFace and ArcFace + OpenFace in our work. VGGFace + FaceNet pair was selected for comparison due to its usage in the [23] and [26], while the rest of the pairs were selected based on the better performance in the previous experiment. Due to computational constraints and the fact that most of these models achieved maximum accuracy from 20 to 40 epochs, it was decided to train models for this experiment only for 50 epochs.

From the results of the experiments (Tables 4.2 and 4.3) we can say that feature extractor influences a lot the performance of the whole method. Also, using two feature extractors instead of one can lead to a small boost in performance (e.g. adding FaceNet to the VGGFace increases accuracy by 3%). Still, those feature extractors should be carefully selected, as loss in the performance is also possible (e.g. adding OpenFace model to VGGFace model decreases accuracy by 1%).

Feature extractors	Accuracy	AUC ROC
VGGFace + FaceNet	0.669	0.738
VGGFace + ArcFace	0.665	0.739
VGGFace + OpenFace	0.658	0.741
ArcFace + OpenFace	0.633	0.690

TABLE 4.3: Influence of different pairs of feature extractors on the whole method

VGGFace + FaceNet and VGGFace + ArcFace feature extractors showed the best performance, for further work first pair was selected.

4.2.2 Embedding combination and fusion

During experiments, the influence of different sets of embedding combinations from such papers as [42],[23] and [26] was tested. Similarly to the experiment setting in the Section 4.2.1 section, nothing except combinations themselves was changed. Models were trained for 50 epochs.

#	Combination	Ref	Accuracy	AUC ROC
1	$(x_1^2 - x_2^2) \bigoplus (x_1 - x_2)^2 \bigoplus (x_1 * x_2)$	[42]	0.669	0.738
2	$(x_1^2 - x_2^2) \bigoplus (x_1 - x_2)^2$	[42]	0.604	0.651
3	$(x_1 + x_2) \bigoplus (x_1 - x_2) \bigoplus (x_1 * x_2)$	[42]	0.672	0.755
4	$(x_1 + x_2) \bigoplus (x_1 - x_2)$	[42]	0.621	0.678
5	$x_1 \bigoplus x_2$	[42]	0.622	0.672
6	$(x_1^2 - x_2^2) \bigoplus (x_1 - x_2)^2 \bigoplus (x_1 * x_2) \bigoplus (x_1 + x_2) \bigoplus (x_1 - x_2) \bigoplus (\sqrt{x_1} + \sqrt{x_2}) \bigoplus (\sqrt{x_1} - \sqrt{x_2})$	[23]	0.674	0.753
7	$(x_1 + x_2) \bigoplus (x_1 - x_2) \bigoplus (x_1 * x_2) \bigoplus (\sqrt{x_1} + \sqrt{x_2}) \bigoplus (x_1^2 - x_2^2)$	[26]	0.666	0.742

TABLE 4.4: Influence of different sets of feature combinations on the overall performance

After performing a comparison of different sets of the feature combination(Table 4.4), we can draw two conclusions:

- performing a combination of features via some linear or nonlinear operation almost always (except #2 combination) improves the performance of the overall solution. Experiment #5, in which extracted features were just concatenated without any linear/nonlinear operation, shows much worse accuracy (4-5% less) in comparison to other experiments
- in our case, the best combination sets were #3 and #6, showing the biggest boost in the performance (+ 5%) compare to #5 experiment

The 3 combination set was selected for further experiments since its AUC ROC score is the biggest.

4.2.3 NN architecture choices

Our baseline NN architecture is similar to one described in [42]: two FC layers, with 128 and 1 neurons respectively. The size of the embedding vector, obtained by feature embeddings fusion, which is fed as an input, is 1152.

Five experiments with different numbers of FC layers and units were conducted. In the experiment #1 number of units was decreased, while other experiments involved increasing the number of both layers and units.

According to evaluation results (Table 4.5), we can see that decreasing number of units (#1 experiment) decreases overall accuracy by 2%. However, increasing the number of layers and units in them (experiments #2 and #3) decreased accuracy as well. Experiment 5 shows the best performance and improves baseline accuracy by 1%. Therefore, it was used during further experiments.

Nonetheless, we can not state that increasing the capacity of the neural network always leads to a performance boost of the whole system since the performance boost is insignificant.

Experiment #	FC layers	Accuracy	AUC ROC
0 (Baseline)	[128, 1]	0.672	0.755
1	[64, 1]	0.650	0.711
2	[128, 64, 1]	0.663	0.725
3	[256, 128, 64, 1]	0.665	0.724
4	[512, 256, 128, 64, 1]	0.683	0.741
5	[1024, 512, 256, 128, 64, 1]	0.683	0.750

TABLE 4.5: Comparison of influence of different NN architecture on the overall performance

4.2.4 Face alignment and cropping

Both face alignment[33] and image normalization[36] were done as additional data preprocessing step, but not as an online data augmentation process. As a baseline, model #3 from the experiment with embedding combinations(Table 4.4) was used. NN consist of 2 FC layers - with 128 and 1 units, respectively.

Data preprocessing	Accuracy	AUC ROC
None	0.672	0.755
Face alignment [33]	0.670	0.740
Image normalization [36]	0.662	0.728

TABLE 4.6: Comparison of influence of data preprocessing on the overall performance

From the results in Table 4.6 we can see that additional data preprocessing such as face alignment[33] or images normalization[36] do not improve accuracy at all.

4.2.5 Data augmentation

As a baseline for this experiment, model #5 from Section 4.2.3 was used. During experiments, such data augmentation techniques as horizontal flip, change of contrast, brightness and saturation, conversion to grayscale, and blurring were compared. Also, combinations of the data augmentations which were used in the work [23] (blurring, flipping, contrast and brightness changes) and work [17] (flipping, contrast, brightness and saturation changes) were tested. All augmentations were applied with a probability of 0.5.

Evaluation results (Table 4.7) shows that from single data augmentation techniques boost in performance was gained only with horizontal flip and blurring. Sets

Augmentation	Accuracy	AUC ROC
No	0.683	0.750
Horizontal flip	0.689	0.771
Contrast	0.676	0.737
Brightness	0.673	0.742
Saturation	0.667	0.743
Gray	0.676	0.744
Blur	0.681	0.757
Blur, Flip, Contrast, Brightness [23]	0.678	0.749
Flip, Contrast, Brightness, Saturation [17]	0.666	0.731

TABLE 4.7: Comparison of influence of data augmentation techniques on the overall performance

of data augmentation techniques used in [23] and [17] did not increase the accuracy of the model.

Chapter 5

Conclusions

In our work, we have done an ablation study of the Siamese networks for the Kinship Verification problem to compare improvements proposed in other works. All improvements were tested under the same experimental settings and benchmarked on the same dataset, which allowed fair comparison. During study, we have tested diverse feature extractors, feature combinations and neural network architecture choices, along with such data preprocessing techniques as data augmentation, face alignment and cropping. Our experimental results show that:

- not all CNNs can be used as feature extractors for kinship verification, but VGGFace and ArcFace showed the best results from the tested models; using such pairs of the feature extractors as VGGFace + FaceNet and VGGFace + ArcFace improves the performance by 3%
- combination of the embeddings via some mathematical operations is needed to increase the performance by 4-5% depending on the exact combination set
- experiments showed that a neural network, which consists of two FC layers with 128 and 1 units respectively, is a good trade-off between performance and number of training parameters
- neither face alignment [33] nor image normalization[36] improves model performance
- such data augmentation techniques as horizontal flipping and blurring could improve the accuracy of the model for kinship verification

Summing up, we can say that feature extractors and embedding combinations plays key role in the model performance, while some data augmentation techniques could increase accuracy by 1-2%.

As possible next steps, we consider using the additional GAN to decrease the age gap between people (Section 3.5), try to remove the influence of the emotions (Section 2.3.3) and use advanced method for the face alignment and cropping (Section 3.4).

Bibliography

[1] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[2] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. *OpenFace: A general-purpose face recognition library with mobile applications*. Tech. rep. CMU-CS-16-118, CMU School of Computer Science, 2016.

[3] Miguel Bordallo Lopez et al. "Kinship Verification from Facial Images and Videos: Human versus Machine". In: *Mach. Vision Appl.* 29.5 (July 2018), 873–890. ISSN: 0932-8092. DOI: 10.1007/s00138-018-0943-x. URL: https://doi.org/10.1007/s00138-018-0943-x.

[4] Qiong Cao et al. "VGGFace2: A Dataset for Recognising Faces across Pose and Age". In: *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*. 2018, pp. 67–74. DOI: 10.1109/FG.2018.00020.

[5] Mitchell Dawson, Andrew Zisserman, and Christoffer Nellåker. *From Same Photo: Cheating on Visual Kinship Challenges*. 2018. arXiv: 1809.06200 [cs.CV].

[6] Jiankang Deng et al. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4685–4694. DOI: 10.1109/CVPR.2019.00482.

[7] Jiankang Deng et al. "RetinaFace: Single-Shot Multi-Level Face Localisation in the Wild". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5202–5211. DOI: 10.1109/CVPR42600.2020.00525.

[8] Hamdi Dibeklioğlu, Albert Ali Salah, and Theo Gevers. "Are You Really Smiling at Me? Spontaneous versus Posed Enjoyment Smiles". In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 525–538. ISBN: 978-3-642-33712-3.

[9] Ruogu Fang et al. "Towards computational models of kinship verification". In: *2010 IEEE International Conference on Image Processing*. 2010, pp. 1577–1580. DOI: 10.1109/ICIP.2010.5652590.

[10] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[11] Yandong Guo et al. "MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition". In: *CoRR* abs/1607.08221 (2016). arXiv: 1607.08221. URL: http://arxiv.org/abs/1607.08221.

[12] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[13] Danula Hettiachchi et al. "Augmenting automated kinship verification with targeted human input". English. In: 24th Pacific Asia Conference on Information Systems : Information Systems (IS) for the Future, PACIS 2020 ; Conference date: 20-06-2020 Through 24-06-2020. 2020.

[14] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7132–7141. DOI: 10.1109/CVPR.2018.00745.

[15] Junlin Hu et al. "Multi-View Geometric Mean Metric Learning for Kinship Verification". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 1178–1182. DOI: 10.1109/ICIP.2019.8803754.

[16] Gary B. Huang et al. "Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments". In: *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*. Erik Learned-Miller and Andras Ferencz and Frédéric Jurie. Marseille, France, Oct. 2008. URL: https://hal.inria.fr/inria-00321923.

[17] Stefan Hörmann, Martin Knoche, and Gerhard Rigoll. "A Multi-Task Comparator Framework for Kinship Verification". In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 863–867. DOI: 10.1109/FG47880.2020.00106.

[18] Vahid Kazemi and Josephine Sullivan. "One millisecond face alignment with an ensemble of regression trees". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1867–1874. DOI: 10.1109/CVPR.2014.241.

[19] Naman Kohli et al. "Hierarchical Representation Learning for Kinship Verification". In: *IEEE Transactions on Image Processing* 26.1 (2017), pp. 289–302. DOI: 10.1109/TIP.2016.2609811.

[20] Weiyang Liu et al. "SphereFace: Deep Hypersphere Embedding for Face Recognition". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6738–6746. DOI: 10.1109/CVPR.2017.713.

[21] Jiwen Lu, Junlin Hu, and Yap-Peng Tan. "Discriminative Deep Metric Learning for Face and Kinship Verification". In: *IEEE Transactions on Image Processing* 26.9 (2017), pp. 4269–4282. DOI: 10.1109/TIP.2017.2717505.

[22] Jiwen Lu et al. "The FG 2015 Kinship Verification in the Wild Evaluation". In: *2015 11th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. Vol. 1. 2015, pp. 1–7. DOI: 10.1109/FG.2015.7163159.

[23] Zhipeng Luo et al. "Challenge report Recognizing Families In the Wild Data Challenge". In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 868–871. DOI: 10.1109/FG47880.2020.00117.

[24] Shahar Mahpod and Yosi Keller. "Kinship Verification Using Multiview Hybrid Distance Learning". In: *Comput. Vis. Image Underst.* 167.C (Feb. 2018), 28–36. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2017.12.003. URL: https://doi.org/10.1016/j.cviu.2017.12.003.

[25] Stephen Milborrow and Fred Nicolls. "Locating Facial Features with an Extended Active Shape Model". In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 504–513. ISBN: 978-3-540-88693-8.

[26] Tuan-Duy H. Nguyen, Huu-Nghia H. Nguyen, and Hieu Dao. "Recognizing Families through Images with Pretrained Encoder". In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 887–891. DOI: 10.1109/FG47880.2020.00130.

[27] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition". In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Mark W. Jones Xianghua Xie and Gary K. L. Tam. BMVA Press, 2015, pp. 41.1–41.12. ISBN: 1-901725-53-7. DOI: 10.5244/C.29.41. URL: https://dx.doi.org/10.5244/C.29.41.

[28] Joseph P. Robinson et al. "Recognizing Families In the Wild (RFIW): The 4th Edition". In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 857–862. DOI: 10.1109/FG47880.2020.00138.

[29] Joseph P. Robinson et al. "Visual Kinship Recognition of Families in the Wild". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.11 (2018), pp. 2624–2637. DOI: 10.1109/TPAMI.2018.2826549.

[30] Joseph Peter Robinson, Ming Shao, and Yun Fu. "Survey on the Analysis and Modeling of Visual Kinship: A Decade in the Making". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pp. 1–1. DOI: 10.1109/TPAMI.2021.3063078.

[31] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.

[32] Sefik Ilkin Serengil and Alper Ozpinar. "LightFace: A Hybrid Deep Face Recognition Framework". In: *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE. 2020, pp. 23–27. DOI: 10.1109/ASYU50717.2020.9259802.

[33] Andrei Shadrikov. "Achieving Better Kinship Recognition Through Better Baseline". In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 872–876. DOI: 10.1109/FG47880.2020.00137.

[34] Ming Shao, Siyu Xia, and Yun Fu. "Genealogical face recognition based on UB KinFace database". In: *CVPR 2011 WORKSHOPS*. 2011, pp. 60–65. DOI: 10.1109/CVPRW.2011.5981801.

[35] Yi Sun, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Representation from Predicting 10,000 Classes". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1891–1898. DOI: 10.1109/CVPR.2014.244.

[36] Tiago F. Vieira, Andrea Bottino, and Ihtesham Ul Islam. "Automatic Verification of Parent-Child Pairs from Face Images". In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. by José Ruiz-Shulcloper and Gabriella Sanniti di Baja. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 326–333. ISBN: 978-3-642-41827-3.

[37] Hao Wang et al. "CosFace: Large Margin Cosine Loss for Deep Face Recognition". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5265–5274. DOI: 10.1109/CVPR.2018.00552.

[38] Shuyang Wang, Zhengming Ding, and Yun Fu. "Cross-Generation Kinship Verification with Sparse Discriminative Metric". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.11 (2019), pp. 2783–2790. DOI: 10.1109/TPAMI.2018.2861871.

[39] Xiaoting Wu et al. “Kinship Verification using Color Features and Extreme Learning Machine”. In: *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*. 2018, pp. 187–191. DOI: 10.1109/SIPROCESS.2018.8600423.

[40] Siyu Xia, Ming Shao, and Yun Fu. “Kinship Verification through Transfer Learning”. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*. IJCAI’11. Barcelona, Catalonia, Spain: AAAI Press, 2011, 2539–2544. ISBN: 9781577355151.

[41] Haibin Yan et al. “Discriminative Multimetric Learning for Kinship Verification”. In: *IEEE Transactions on Information Forensics and Security* 9.7 (2014), pp. 1169–1178. DOI: 10.1109/TIFS.2014.2327757.

[42] Jun Yu et al. “Deep Fusion Siamese Network for Automatic Kinship Verification”. In: *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. 2020, pp. 892–899. DOI: 10.1109/FG47880.2020.00127.