An Exploration on Face Verification with Siamese Network: Final Report

G059 (s1807464, s1829742, s1839441)

Abstract

Recently, with the fast progress on deep Convolutional Neural Networks (CNNs), many successes have been made in computer vision community, including face related tasks like face recognition. In this report, we aim to do some explorations in the domain of face verification. Due to the huge and uncertain number of human face classes as well as scarce individual data, metric learning is commonly used in this field. One of the similarity metric learning skeleton is Siamese Network, which contains two weight-shared backbone CNNs to extract human face features and uses a predefined distance metric to measures the dissimilarity between two faces. We constructed a customised Siamese Network using four built-in CNNs and four loss functions from recent research, and then evaluated all combinations on the subset of MS-Celeb-1M dataset via ROC_AUC score. We discovered that ArcFace Loss integrated with ResNet-50 renders best performance.

1. Introduction

In the last decade, much progress has been made in the domain of face related tasks such as face verification due to the rapid development of Convolutional Neural Networks (CNNs). In fact, what makes CNNs useful is their ability to extract dense features from input images like human faces. Specifically, features extracted by CNN are bottom-up, hierarchical and representing multi-stage information of inputs (Farabet et al., 2013). Moreover, recent success on deep CNNs which are able to generate more powerful and representative features indeed demonstrated impressive performances on different computer vision tasks (Chen et al., 2015), among which we are most interested in face verification.

Face verification is the task of determining whether a pair of face images comes from the same person. It has many practical applications, for example, facial payments, criminal identification, etc. In this field, metric learning methods are used heavily (Taigman et al., 2014). The main idea of metric learning is to learn a similarity measure from data, so that the model can tell how similar two inputs are, which provides a natural solution for verification tasks (Sun et al., 2014a). Besides, metric learning tries to solve problems in which the number of classes is huge and/or uncertain, while training instances for each individual class are scarce. This exactly fits the requirement of face verification tasks: in real scenarios, the amount of persons in a governmental database is usually enormous and constantly growing due to newborns and immigrants, while images for each person are relatively scarce. Therefore, traditional classification paradigms assuming fixed number of classes are inappropriate in this case (e.g. networks with softmax layers have to be retrained every time as new persons are added, which is infeasible in real applications). Instead, having a neural network learn a metric which compares two images and decides whether these two image come from the same person is more applicable.

One commonly used metric-learning-based framework is Siamese Network (will be discussed in section 3), which is initially proposed for signature verification (Bromley et al., 1994). Later Chopra et al. (2005) introduced it to face verification application. In Siamese structure, it applies an built-in customised CNN as the feature extractor of input images, and then calculates the dissimilarity of paired images via some metrics specified by the representation space. Actually, this skeleton does not define the standard structure of the CNN. Therefore, in this project we aim to embed different backbone CNNs to compare their ability of extracting human face features, meanwhile cooperating with different loss functions to discover which kind of representation space is more suitable for face verification, which will point out an avenue for future work. The backbone CNNs we explored include Chopra Net, DeepFace, DeepID and ResNet-50, while the loss functions include Constractive Loss, Triplet Loss, Logistic Loss and ArcFace Loss, all of which will be described in detail in section 3.

2. Data set and task

2.1. Dataset Source

The dataset we are applying comes from "Low Shot Learning Data" of MS-Celeb-1M (Guo et al., 2016), in which 1.15 million cropped and aligned face images of celebrities are collected. It's worth noting that only about a hundred images are collected for each person, implying that the total number of classes is roughly around $11,550$. With huge amount of labels (person identities) and relatively scarce data for each label, we presume that Siamese network is suitable for dealing with this dataset for the reasons listed in introduction. In addition, due to limited computational resources, we only take approximately one tenth (100 thousand images) of the original dataset to train and test our models.

2.2. Data Splitting and Labeling

The above dataset is divided in such a way that 70% data are for training, 20% data are for validation and the remaining 10% data are for the evaluation of generalization performance. Moreover, input images are first made into pairs (or triples for Triplet Loss in particular) before being passed into Siamese network, and the probabilities of producing positive samples (i.e. pairs of same person) and negative samples (i.e. pairs of different persons) are deliberately controlled as half and half (50%). Besides, positive pairs are labelled as 0, whereas negative pairs are labelled as 1.

2.3. Data Augmentation

Furthermore, we take advantage of face's symmetry and achieve data augmentation by randomly performing horizontal flips on input faces before feeding them into our network, which makes our networks more robust and less likely to suffer from overfitting.

2.4. Task

The primary task of our project is to construct the whole structure of Siamese network and build a model-learning pipeline within PyTorch framework. Based on the overall Siamese architecture, various backbone CNNs and loss functions that we want to explore are then separately implemented and embedded into the corresponding components the Siamese skeleton. We thereupon observe the performance of different combinations. For each backbone+loss combination, the best set of hyperparameters will be determined by selecting the set of hyperparameters with the minimum loss on the validation set, and its generalization performance will be evaluated on test set through the measure of ROC-AUC score (i.e. area under the ROC curve).

3. Methodology

3.1. Siamese Network

Siamese Network has a symmetric structure with two weight-shared sub-networks. These two backbone networks are typically CNNs used to extract features of two input face images. The outputs of sub-networks will be the vector encodings of these two images and are subsequently measured with L2 norm or space-specified metrics to identify whether two face images belong to the same person. The whole architecture is given in Figure 1.

3.2. Backbone CNNs

3.2.1. Chopra Net

Chopra Net is the initial backbone network used in the original work Chopra et al. (2005), where our inspiration came from. Basically, it mimics the structure of LeNet-5 (LeCun et al., 1998) and is composed of three convolutional layers with two interlaced subsampling layers and a final fully-connected layer. The detailed configuration is shown in Table 1.

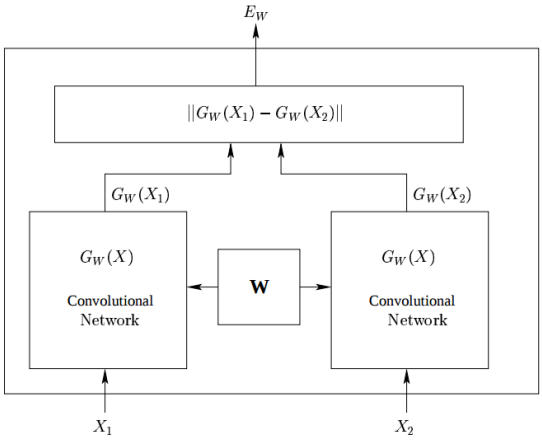

Figure 1. Architecture of the Siamese Network, which is comprised of two backbone CNNs and one similarity metric.

Layer	In. Size	Out. Size	Kernels
C1	1*56*46	15*50*40	15*7*7
S2	15*50*40	15*25*20	15*2*2 (2 strides)
C3	15*25*20	45*20*15	45*6*6
S4	45*20*15	45*5*5	45*4*3 (4*3 strides)
C5	45*5*5	250*1*1	250*5*5
F6	250	50	-

Table 1. Configuration of Chopra Net. C stands for convolutional layer, S stands for subsampling one and F for fully-connected one. The convolutions are with 1-step stride, no padding and no dilation by default.

It is worth mentioning that the subsampling layers in this network are also adopted from LeNet-5. Each one is an average pooling layer followed by multiplication with a coefficient and addition with a bias, namely convoluted by an average pooling kernel with unshared weights and bias for each channel, thereby reducing the resolution of the feature map and meanwhile alleviating the sensitivity of face shifts and distortions. After that, the result is subsequently passed through a sigmoid function. When implementing this network in PyTorch, we firstly let the input go through an average pooling layer, then decouple channels to do 1*1 convolution as well as sigmoid operation separately, and concatenate them at last to yield full output.

In addition, C3 is partially connected to S2 in order to break symmetry and thus extract more different features. However, it does not mention the details of partial connection. Initially, we have tried replacing it with dropout layer[1] for simplification. But the result turned out to be even worse than not applying the dropout layer. In the end, we chose to ignore this setup.

[1]The coefficient of the dropout layer is the division between the number of connections given in the paper and that of full connections.

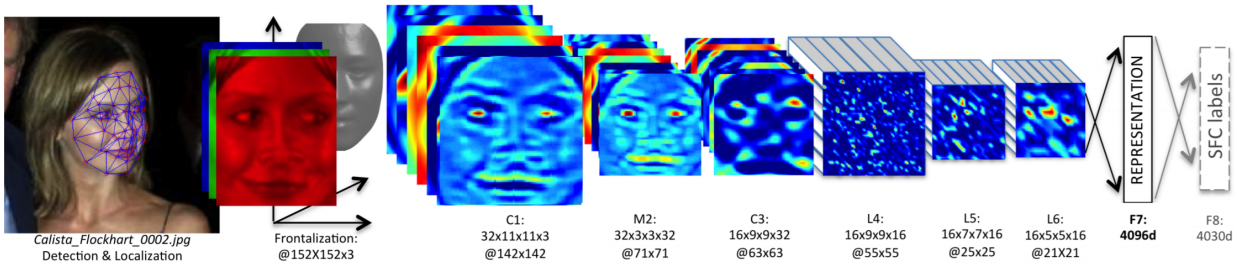

Figure 2. Outline of DeepFace architecture. It passes 3-channel RGB face images through two convolutional layers, one maxpooling layer and 3 locally-connected layers as well as a final fully-connected layer. Note that F8 is the classifying layer for face recognition which should be discarded in face verification application.

3.2.2. DeepFace

DeepFace (Taigman et al., 2014) was proposed by FaceBook AI Research. In this network, it abandons the subsampling layer, instead adopts maxpooling layer. The most important element in the model should be the locally connected layer. Unlike traditional convolutional layer where one set of kernel parameters are shared across the entire feature map of an input image, kernel parameters in locally connected layer are totally unshared between different regions of the feature map. In other words, every feature map region (each stride) has a unique weight setting. The reason to use this layer is that aligned human face contains discriminative local image pattern so there can be information loss to process the entire face with a single set of kernel parameters. Although the parameters can be enormous and cause heavy computation, the model is believed to be benefited more on performance. The locally connected layer is also introduced in some other backbone implementations such as DeepID which is explained later. We successfully implemented totally unshared layers by modifying source code of PyTorch and verified its correctness[2]. The network flowchart for DeepFace are given in Figure 2 (cited from (Taigman et al., 2014))

One thing to note is that DeepFace actually takes RGB (3-channel) images as input. Another thing to note is that the final layer F8 is used as classification layer which generates labels via softmax function. But in the scenario of face verification through Siamese Network, the backbone CNN are only used for producing vector presentations of input face images, which means the first seven layers of DeepFace are already enough.

3.2.3. DeepID

DeepID model (Sun et al., 2014b;c; 2015b;a) is another feature extraction network to be explored. Up to now, four versions of DeepID models are developed in total. However, starting from DeepID-v2, networks are inherently entangled with a highly customized cost function designed for face classification, making it difficult to decompose the independent feature extraction part from the overall structure. Therefore, our focus is on the initial version of DeepID for its compatibility to Siamese Network.

DeepID can be seen as a feature extractor that is able to obtain a low-dimensional (160 in this case) but compact feature representation for faces. The network architecture is shown in Figure 3 (cite from (Sun et al., 2014b)). It takes an one-channel greyscale image and processes it by successive Conv-ReLU-Max pooling layers. Image is shrunk in width and height but enlarged in depth to form an increasingly compact representations. Notice that kernel parameters are fully shared in the first and second convolutional layers, while the third and fourth layers contains unshared kernel parameters. However, layer 3 requires partially share kernel parameters which is beyond our scope of implementation. Therefore, only the totally unshared layer (layer 4) is implemented and partially connected layer 3 is replaced with an ordinary convolutional layer. This is by far the most significant difference compared with original implementation in the paper.

Sun et al. believes that successive down-sampling limits the information propagation in feature learning. To overcome it, features from Max pooling layer 3 and convolutional layer 4 are combined in the way shown in Equation 1. After features from both layers are fully connected to 160 dimensions separately, we sum them element-wise and then apply ReLU to the result, forming a final representation of 160 dimensions. Multi-scale features are all considered to avoid information loss, which is a popular approach in recent research such as in ResNet (He et al., 2015). Although this network is trained by softmax function for the purpose of classification, we decide to truncate the model, keeping the layer before softmax as our feature extractor in the Siamese Network for a end-to-end training, where we assume the 160-dimension features are representative enough for our verification task.

$$y_i = \max\left(0, \sum_i x_i^1 \cdot w_{i,j}^1 + \sum_i x_i^2 \cdot w_{i,j}^2 + b_j\right) \quad (1)$$

[2]Code from: https://github.com/pytorch/pytorch/pull/1583/files#diff-2

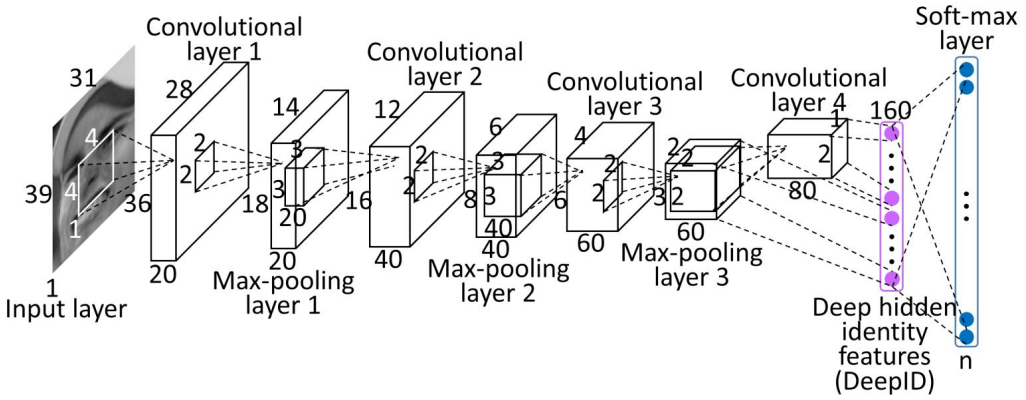

Figure 3. Illustration of DeepID network. Gray scaled image is processed through 3 Conv+MaxPool sessions with another Conv layer and formed to be a 160 dimensional representation.

3.2.4. ResNet

ResNet is a powerful neural network architecture proposed by He et al. (2015). We consider it as one of the backbone candidates for its high performance in vision tasks such as image classification and object detection, indicating its ability in learning representations. As shown in Figure 4, those two building blocks severs as the core in ResNet. The skip connections shown in blocks are believed to overcome vanishing gradient problem which happens in deep networks. Information can be largely recovered by passing through the shortcut path. In the right block, there is a down sampling process followed by up sampling with the output combined with original input, forming a more compact representation. In our experiment, we chose ResNet-50 considering it is not as deep as ResNet-101/152 while containing powerful bottleneck unit. We truncate original ResNet50 to remove its last layer which is designed for classification so we can have a higher dimensional feature for face verification purpose.

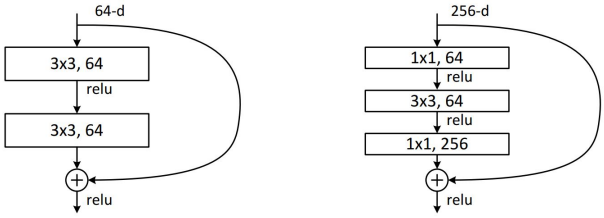

Figure 4. Core unit in ResNet architecture. Left is building block for ResNet-34. Right is the bottleneck block for ResNet-50/101/152, with down sampling and up sampling processes.

3.3. Loss Functions

3.3.1. Contrastive Loss

The loss function proposed in the original paper (Chopra et al., 2005) for Siamese Network is called "Contrastive Loss". The general form of contrastive loss is given by the following equation:

$$L_{Contrastive} = (1 - Y)L_G(E_W(X_1, X_2)^i) + YL_I(E_W(X_1, X_2)^i) \quad (2)$$

In the Equation 2, W represents the weights of the CNN and $E_W(X_1, X_2)$ calculates the relative distance between the i-th image pair X_1 and X_2 in an Euclidean space. Moreover, Y indicates whether these two images come from the same person ($Y = 0$) or different persons ($Y = 1$). The key idea behind Contrastive Loss is to ensure that L_G is monotonically increasing and L_I is monotonically decreasing, hence encouraging similar pairs to get closer and dissimilar pairs to be separated away from each other. As usual, the Contrastive Loss will be minimized through gradient descent with respect to W in order to find the best set of parameters for the CNN.

Unfortunately, the exact form of the original paper's Contrastive Loss has a value Q which is difficult to estimate. Therefore, we integrate another variant of Contrastive Loss from Hadsell et al. (2006) into our baseline.

3.3.2. Triplet Loss

Afterwards, a more recently proposed loss functions called "Triplet Loss" (Schroff et al., 2015b) has been tried. The general form of triplet loss is given by the following equation:

$$L_{Triplet} = max(\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha, 0) \quad (3)$$

The main idea behind Triplet Loss is very similar to that of Contrastive Loss, in the way that both of them try to push similar faces closer and separate dissimilar faces farther in an Euclidean space. However, the novelty that Triplet Loss brings in is that it further emphasizes the difference between positive pair and negative pair by training triple images at the same time. At every iteration, our network is given an anchor image x_i^a, a positive image x_i^p and a negative image x_i^n, in which anchor and positive images form a pair of the same person and anchor and negative images form

a pair of different persons. By considering negative and positive pairs simultaneously, networks is presumably able to learn key characteristics to identify the same person and distinguish different persons.

In the Equation 3, triplet loss makes effort to enforce the constraint that the squared distance between positive pair is strictly smaller than that of negative pair, in which α is a hyperparameter "margin" and is strictly positive.

3.3.3. Logistic Loss

The previous two losses assume fixed metric space (i.e. Euclidean space) and does not have learnable parameters. In this part, a new kind of loss which has tunable parameters and is able to learn a customized distance metric has been suggested in Taigman et al. (2014). We name it as "Logistic Loss" mainly for the reason that it essentially utilizes logistic regression layer combined with Cross Entropy Loss. The learned distance between two face vectors is as follows:

$$d(f_1, f_2) = \sum_i \alpha_i |f_1[i] - f_2[i]| \tag{4}$$

In the Equation 4, f_1 and f_2 are the vector representations of the input face pair, which are produced by Siamese Network. Additionally, α_i are learnable parameters of the logistic regression layer trained through minimizing the Cross Entropy loss. By applying this particular loss, face verification can be successfully reframed and simplified as a standard binary classification problem, with the feature vectors given by Siamese Network's backbone. The greater the distance is, the more possible these two faces come from different persons.

It is worth explaining the motivation why the absolute difference between two vectors is taken as input to the logistic regression layer. The main concern here is to ensure the symmetry of input pairs. For example, if face A and B belonging to different persons are fed as a pair into our network, it should not matter whether face A comes before or after B, as the outcome of our prediction will not be influenced by the relative positions of two images. In fact, we can interpret the result given by Equation 4 as Manhattan distance between two vectors with each axis weighted by one of the learnable parameters. Of course, absolute difference is not the only way to achieve this symmetry, but to limit the scope of our experiment, we do not explore other possibilities.

3.3.4. ArcFace Loss

Similar to the Logistic Loss, ArcFace Loss proposed in Deng et al. (2018) also has learnable parameters and take the absolute difference between two face vectors as its input. In addition, it utilizes a specially designed softmax layer along with Cross Entropy Loss. The exact form of this loss is illustrated as below:

$$L_{ArcFace} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}} \tag{5}$$
$$where\ \cos\theta_{j_i} = W_j^T x_i / (\|W_j\|\,\|x_i\|)$$

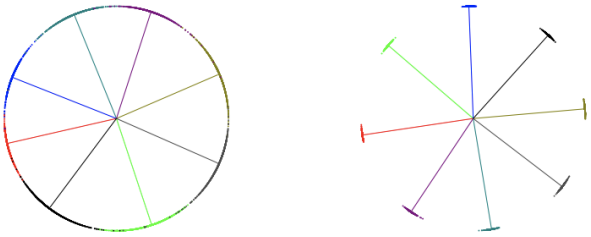

Figure 5. Decision boundaries of Softmax (left) and ArcFace (right) (Deng et al., 2018). The authors claim that ArcFace enforces a larger gap between weight vectors of different classes.

What makes it different from Logistic Loss is that it measures the angle between the input vector x_i and class vectors W_j (i.e. weight vectors corresponding to label 0 and 1 in our case) instead of distance. A hyperparameter m "margin" is added to the Equation 5 to make sure that the angle between the input vector and the class vector of its truth label is strictly less than the angle between the input vector and the class vector of its opposite label. Additionally, another hyperparameter s interpreted as a scaling factor is applied to shrink or enlarge the $\cos\theta_{j_i}$ before going through the normalization step.

4. Experiments

The experiment is carried out by trying different combinations between our aforementioned backbone networks and loss functions. The experiment aims to search for the best combination which achieves the best performance. Then, we will be able to draw conclusion on whether certain loss function or model outperform the rest.

4.1. Expeiment Settings

Consider the fact that both locally connected convolution and deeply stacked network will lead to numerous parameters in CNNs, we cannot conduct grid-search naively on huge dataset for hyperparameters like learning rate or margin in some loss functions, because it will be extremely time-consuming due to our insufficient computational resource. In this experiment, we manage to lower the cost on hyperparameter selections.

4.1.1. Learning Rate

To determine an appropriate learning rate setting, we tune the learning rate for each model on a small portion of data (10000 images) rather than the whole data (100000 images). This is reasonable because the distributions of the data in whole dataset and its small subset are similar, so that the

Model	Loss Function	Learning Rate	Valid. Loss	Test ROC_AUC	# Learnable Params
ChopraNet	Contrastive	0.003	1.6065	0.5843	0.32M
	Triplet	0.01	975.6872	0.9104	
	Logistic	0.0007	0.5524	0.7931	
	ArcFace	0.0001	3.0485	0.8308	
DeepFace	Contrastive	0.000014	1.5427	0.8650	102M
	Triplet	0.00005	15.1498	0.9732	
	Logistic	0.000032	0.2990	0.9461	
	ArcFace	0.000023	1.0576	0.9438	
DeepID	Contrastive	0.005	1.5773	0.5583	0.151M
	Triplet	0.0014	361.7614	0.9662	
	Logistic	0.0014	0.3192	0.9387	
	ArcFace	0.0032	1.8626	0.9328	
ResNet-50	Contrastive	0.0005	1.5698	0.5968	25.6M
	Triplet	0.0005	15.0407	0.9697	
	Logistic	0.00014	0.2132	0.9732	
	ArcFace	0.00014	0.8674	0.9830	

Table 2. Experiment results: verification performances of different combinations. Note that Chopra Net and DeepID models both applies batch size of 100, while DeepFace and ResNet-50 can only have batch sizes of 5 and 20 because they both require larger 3-channel input images that are memory-consuming, while only 16GB RAM is available for experiments. Hence, there will be a magnitude difference on same loss function between models.

loss space we aim to optimize will be roughly the same. This assumption is verified by conducting an experiment using DeepID and Triplet Loss on both subset and original data. The subset we used is one tenth of our whole dataset, namely 10 thousand images. In the first round we search the optimal learning rate among $1e^{-4}$, $5e^{-4}$, $1e^{-3}$, .. , $5e^{-1}$. And the result shows that both whole dataset and subset prefer the learning rate $5e^{-3}$. Then we continue searching for the second round, the result is given below:

Learning Rate	$1e^{-3}$	$2e^{-3}$	$3e^{-3}$	$4e^{-3}$	$5e^{-3}$
Loss (Subset)	48.07	39.65	**33.84**	37.90	35.55
Loss (Whole)	22.34	22.31	**20.86**	22.19	21.55

Table 3. Result of testing whether dataset and its subset would show preference on same learning rate.

Both rounds support our assumption that subset and original dataset have similar distributions, which means we can use the optimal learning rate on small subset to approximate the optimal learning rate on large dataset. Therefore, we will use the subset data with 10 thousand images to determine the optimal learning rate of the original dataset for each backbone+loss combination in further experiments.

4.1.2. Hyperparameters in Loss Function

As we mentioned before, all loss functions have hyperparameters like margin to tune except for Logistic Loss. But still, it would not be feasible to have extra loops to search for best combination of loss hyperparameters due to limited computational resource. We try to fix the hyperparameters of loss function and only leave learning rate to be searched. In fact, metric is correlated with loss function. That is to say, once loss function has determined, the metric space will also be defined, under which backbone CNN tries to extract discriminative features that suit this metric space the most. This would be more fair to compare the performances of different backbones. Yet it is undeniable this is a weak assumption but can save plenty of times. We set the hyperparameters of loss functions as follows:

Loss Function	Hyperparameter(s)
Contrastive Loss	$margin = 2.5$
Triplet Loss	$margin = 45$
Logistic Loss	-
ArcFace Loss	$margin = 0.025$, $scaler = 200$

Table 4. Hyperparameter settings of loss functions.

4.1.3. Optimizer and Learning Rate Scheduler

In the experiment, we applied Adam optimizer with weight decay coefficient of $5e^{-4}$ which follows the set in (Deng et al., 2018). Besides, we kept the other parameters β_1, β_2 and ϵ as default (i.e. 0.9, 0.999 and $1e^{-8}$, respectively). Additionally, Cosine scheduler was also adopted to anneal learning rate to achieve finer descent as the number of

iterations increase. For simplification, we set the tuning learning rate as maximum and 0 as minimum in the last iteration without restarts.

4.2. Results and Discussion

The experiment results are listed in Table 2. Besides, we also made a graph of test ROC_AUC to better visualize the outcomes.

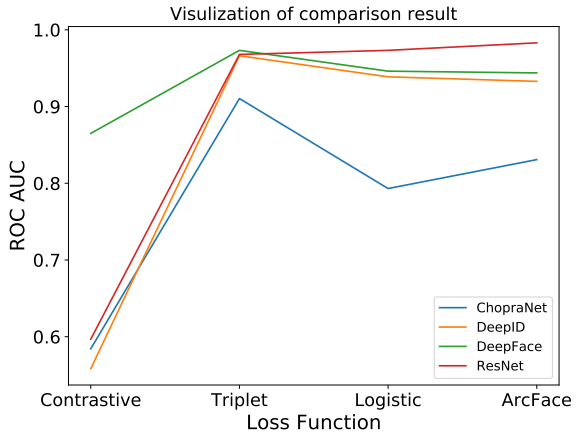

Figure 6. A visulization on the comparison of model and cost functions

4.2.1. Comparison on Models

Overall, the performance of ResNet-50 exceeds that of other backbone networks. One explanation is that the deeper network combined with bottleneck structure avoid the problem of vanishing gradient and contribute a lot to its performance. On the contrary, Chopra Net performs worst mainly due to its shallow and outdated architecture. Specifically speaking, it contains no ReLU layer at all and still adopts the old-fashioned sigmoid activation function which is prone to the problem of vanishing gradient. In addition, Chopra Net only produces 50-dimension feature vector which seems not enough to capture the complex features of human faces. What out of expectation is that DeepFace outperforms DeepID under all loss functions. There are two reasons to explain this: one is the failure on implementing the partially-shared layer in DeepID, which makes uncompleted DeepID fail to extract as much sufficient features as it is supposed to be; another reason is that we did not strictly train the model patch-by-patch as stated in the source paper (Sun et al., 2014b). However, one should be aware is DeepFace consumes far more time and has far more parameters than DeepID does given the fact that three locally-connected layers are embedded in DeepFace, whereas DeepID just has one locally-connected layer (as the number of learnable parameters suggests in Table 2). If consider the trade-off between speed and accuracy, sometimes lightweight DeepID can be more desirable.

4.2.2. Comparison on Loss Functions

Generally speaking, Triplet Loss gets far ahead ROC_AUCs among all models except for ResNet-50 due to its special characteristic of contrasting one positive and one negative example simultaneously, which greatly boosts the learning procedure. However, with Triple Loss, backbone CNNs are required to compute the vector representations of three images all at once, and thus it is rather slow than other loss functions. Regarding to ArcFace Loss, it surprisingly outperforms Triplet Loss and the rest when integrated with ResNet-50 network, while on other three CNNs it just achieves favorable but not preeminent performance. We think this phenomenon originates from the difference of metric space - arc space probably requires outstanding-quality feature extractors to well represent instances in the space, and only the deep CNN ResNets meets its requirement. As for Contrastive Loss, it is quite inferior compared with the other loss functions. In addition, although Logistic Loss cannot beat Triplet or ArcFace Loss in all cases, it still produces favorable results. The most obvious merit of it is no need for tuning hyperparameters.

4.3. Demonstration

In the last part of this section, results produced by the model with the most predictive power (i.e. Resnet-50 + ArcFace Loss) have been given as demo to demonstrate the usefulness of our model.

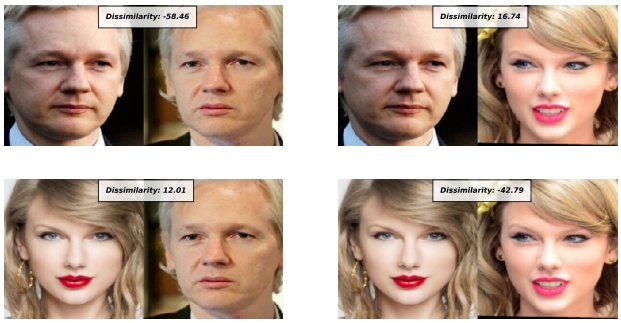

Figure 7. Measure of dissimilarity by best combination: Julian Assange vs. Taylor Swift

As shown in Figure 7, the dissimilarities or "distances" between photos of same persons (i.e. photos of Julian himself with distance of -58.46 and photos of Taylor herself with distance of -42.79) are relatively small compared to the dissimilarities between photos of different persons (i.e. photos of Julian and Taylor in pairs with distances of 12.01 and 16.74). Therefore, a predefined threshold like D = 0 can easily separate them apart without any ambiguity. Please be aware that dissimilarities or "distances" are allowed to be negative since ArcFace does not reside in Euclidean space.

5. Related Work

Face verification has always been one of the most active fields in computer vision community. In recent years, there

are mainly two approaches to improve the accuracy of face verification. At the first stage, researches almost focused on ameliorating the quality of feature extractors - CNNs. Apart from the backbone CNNs that we have explored in this project, some other powerful CNN models like FaceNet (Schroff et al., 2015a), VGGFace (Parkhi et al., 2015), light CNN (Wu et al., 2015) were proposed to obtain better representations of human face images.

However, after the advent of ResNet, it gradually dominated the community due to its superior feature extracting ability. Therefore, the concentration of research in the field has moved to the loss function which guides the learning direction of models, such as vMF Loss (Hasnat et al., 2017), Marginal Loss (Deng et al., 2017), AMS Loss (Wang et al., 2018a), etc. The latest research considered projecting features into different kinds of metric space rather than Euclidean space. In other word, compared to previous researches in which most works were done in Euclidean space, the new trend among researchers nowadays is to explore new metric space like angular one (Liu et al., 2017) or cosine one (Wang et al., 2018b). For now, even the state-of-the-art loss function (i.e. ArcFace Loss) also aims to deal with angular margin, which proves the feasibility of the idea. Consequently, the future work will most probably focus on discovering new metric spaces and objective functions to be optimized, so that feature vectors can be embedded into the space more ideally.

6. Conclusions

In this project, we conducted experiment to evaluate 16 different combinations of backbone CNNs and loss functions in customized Siamese Network on the subset of MS-Celeb-1M dataset. ResNet-50 with Arcface turned out to be the most extraordinary combination due to ResNet's powerful feature-extracting ability, as well as the innovative metric space and objective function defined by ArcFace. This points out the direction for the future work on improving performance of face verification - enhancing the feature representing capability by cooperating with existed deep CNNs or even inventing a novel one. Moreover, exploring different metric space rather than Euclidean one would probably receive improvement.

References

Bromley, Jane, Guyon, Isabelle, LeCun, Yann, Säckinger, Eduard, and Shah, Roopak. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pp. 737–744, 1994.

Chen, Jun-Cheng, Patel, Vishal M., and Chellappa, Rama. Unconstrained face verification using deep CNN features. *CoRR*, abs/1508.01722, 2015. URL http://arxiv.org/abs/1508.01722.

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pp. 539–546 vol. 1, June 2005. doi: 10.1109/CVPR.2005.202.

Deng, Jiankang, Zhou, Yuxiang, and Zafeiriou, Stefanos. Marginal loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 60–68, 2017.

Deng, Jiankang, Guo, Jia, and Zafeiriou, Stefanos. Arcface: Additive angular margin loss for deep face recognition. *CoRR*, abs/1801.07698, 2018. URL http://arxiv.org/abs/1801.07698.

Farabet, Clement, Couprie, Camille, Najman, Laurent, and LeCun, Yann. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35 (8):1915–1929, August 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2012.231. URL http://dx.doi.org/10.1109/TPAMI.2012.231.

Guo, Yandong, Zhang, Lei, Hu, Yuxiao, He, Xiaodong, and Gao, Jianfeng. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. *CoRR*, abs/1607.08221, 2016. URL http://arxiv.org/abs/1607.08221.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742, June 2006. doi: 10.1109/CVPR.2006.100.

Hasnat, Md. Abul, Bohné, Julien, Milgram, Jonathan, Gentric, Stéphane, and Chen, Liming. von mises-fisher mixture model-based deep learning: Application to face verification. *CoRR*, abs/1706.04264, 2017. URL http://arxiv.org/abs/1706.04264.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liu, Weiyang, Wen, Yandong, Yu, Zhiding, Li, Ming, Raj, Bhiksha, and Song, Le. Sphereface: Deep hypersphere embedding for face recognition. *CoRR*, abs/1704.08063, 2017. URL http://arxiv.org/abs/1704.08063.

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

Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015a. URL http://arxiv.org/abs/1503.03832.

Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015b. URL http://arxiv.org/abs/1503.03832.

Sun, Yi, Chen, Yuheng, Wang, Xiaogang, and Tang, Xiaoou. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pp. 1988–1996, 2014a.

Sun, Yi, Wang, Xiaogang, and Tang, Xiaoou. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pp. 1891–1898, Washington, DC, USA, 2014b. IEEE Computer Society. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.244. URL https://doi.org/10.1109/CVPR.2014.244.

Sun, Yi, Wang, Xiaogang, and Tang, Xiaoou. Deep learning face representation by joint identification-verification. In *NIPS*, 2014c.

Sun, Yi, Liang, Ding, Wang, Xiaogang, and Tang, Xiaoou. Deepid3: Face recognition with very deep neural networks. *CoRR*, abs/1502.00873, 2015a.

Sun, Yi, Wang, Xiaogang, and Tang, Xiaoou. Deeply learned face representations are sparse, selective, and robust. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2892–2900, 2015b.

Taigman, Yaniv, Yang, Ming, Ranzato, Marc'Aurelio, and Wolf, Lior. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708, 2014.

Wang, Feng, Liu, Weiyang, Liu, Haijun, and Cheng, Jian. Additive margin softmax for face verification. *CoRR*, abs/1801.05599, 2018a. URL http://arxiv.org/abs/1801.05599.

Wang, Hao, Wang, Yitong, Zhou, Zheng, Ji, Xing, Li, Zhifeng, Gong, Dihong, Zhou, Jingchao, and Liu, Wei. Cosface: Large margin cosine loss for deep face recognition. *CoRR*, abs/1801.09414, 2018b. URL http://arxiv.org/abs/1801.09414.

Wu, Xiang, He, Ran, and Sun, Zhenan. A lightened CNN for deep face representation. *CoRR*, abs/1511.02683, 2015. URL http://arxiv.org/abs/1511.02683.