

Table 1 Comparison of algorithms and improvements for the few-shot learning methods.

	Oriol <i>et al.</i> Matching network [11]	Chen <i>et al.</i> Baseline Plus [15]	Snell <i>et al.</i> Prototypical network [14]	Sung <i>et al.</i> Relation Network [16]	Our work CPNet
Metric function	Cosine distance	Cosine distance	Euclidean distance	Deep non-linear distance	Mahalanobis distance
Algorithm overview	<ul style="list-style-type: none"> Using CNN and LSTM with attention to access memory matrix to enable rapid learning. Trained using a meta-learning setup. Taking a softmax over the similarity score obtained from cosine distance. 	<ul style="list-style-type: none"> Also called as “Fine-tune”; using a simple pretrained deep network to extract features instead of using meta-learning setup. Performing classification based on nearest neighbor classification with cosine distance on embeddings derived from a network trained conventionally. 	<ul style="list-style-type: none"> Using a CNN to construct a feature embedding. Prototype is made for each class on the feature embedding, taken by averaging embedding vector of each class. Trained using a meta-learning setup. Taking a softmax over the negative distance from Euclidean distance. 	<ul style="list-style-type: none"> Using a relation classifier CNN. Trained using a mean square error (MSE) loss. Comparing query and sample terms using a deep nonlinear distance metric. 	<ul style="list-style-type: none"> Using a CNN to construct feature embedding from a network trained in non-meta-learning setup. Making a prototype representation for each class built from pre-trained embedding function. Taking a softmax over the negative Mahalanobis distance from the data spread of each class.
Key findings or improvements	<ul style="list-style-type: none"> LSTM makes it easier to remember and adapt to new training sets. Achieving good accuracy output especially on the one-shot settings. 	<ul style="list-style-type: none"> Using a knowledge transferred from a pretrained network trained in non-meta-learning setup also proves to show comparable and competitive results. 	<ul style="list-style-type: none"> Simple and efficient inference operation by only averaging feature vector and calculate its distance. Offering more stable results and low computational costs. 	<ul style="list-style-type: none"> The query image is compared with a small number of labeled sample images through learning. Excelling in both conventional and zero-shot settings. 	<ul style="list-style-type: none"> Combining the simplicity of using a pretrained network to build feature embeddings and the simple yet efficient operation of prototype representation distance calculation, enhanced with covariance matrix.

ber of images in a cluster and categorized the new category of images by calculating a relationship score between the query image and a small number of examples from each new category. Related works are compared in detail in Table 1.

Currently, the majority of available methods for recognition rely on a conventional learning model that requires large numbers of samples. However, the process of labeling a large number of samples requires expertise and time and also tends to introduce human error or subjective bias. Consequently, the conventional non-FSL learning methods do not work well if only limited samples are available. In addition, conventional learning requires retraining whenever new people need to be added to the system or when irrelevant personnel who have been recorded in the system need to be removed, which is inefficient.

FSL methods for face recognition have also been proposed for application to face images that are obtained under various unfavorable conditions, such as poor lighting, head rotation, and occlusion. Holkar et al. [17] presented an FSL method based on Siamese networks for multi-class face recognition from a training dataset. Three types of differences in face images were considered: low light, head rotation, and occlusion. Yang et al. [18] constructed a TwoStream Prototype Learning Network (TSPLN) scheme using learning adaptive weights for different supported images and relevance by considering the quality of the image and relevance to the query.

This study focuses on addressing specific challenges related to masked face recognition: occlusion and information loss due to masks. We present CPNet to resolve these issues. The proposed method is based on the FSL method, which allows it to handle masked face-recognition tasks with

only a few samples per identity. Furthermore, no retraining is needed whenever adding or removing an identity, thus enhancing the efficiency in real-life scenarios. The significant difference between our work and the previous studies is that we used a non-meta-learning setup for uncomplicated training. First, we applied sharpness-aware minimization (SAM) during the training stage and then incorporated a class-covariance matrix into the distance calculation process during inference time, which refined the discriminative ability of the prototype classifier. The more detailed explanation is in Sect. 2.

In short, our contributions are summarized as follows:

1. To handle the condition where only limited masked face samples are available, we propose the CPNet, a method based on few-shot learning, which fits the case where collecting numerous face examples is too laborious or impossible to accomplish.
2. To improve the distance calculation process, CPNet utilized the data spread of each identity by integrating the covariance structure of the data in the distance calculation, which improves the generalization ability.
3. To enhance the performance further, the recently invented Sharpness Aware Minimization (SAM) is integrated into the CPNet on the classifier training stage.
4. We trained and tested the CPNet with two backbones on two publicly available datasets. Out of fairness, we also self-collected and tested two other datasets, which contain multiethnic cohorts, to examine the effectiveness of our method in a diverse range of settings.
5. Our method performs remarkable results with accuracy as high as 95.3%, which is 3.4% higher than that of the

baseline prototype network used for comparison.

The rest of the paper is divided into some parts. Section 2 describes the methods. Section 3 summarizes and examines the results of our proposed method. Finally, Sect. 4 presents the conclusion of the work.

2. Proposed Method

This section describes the learning paradigm used for building the few-sample masked face recognition model, namely the CPNet, the Prototypical network, the training method, the integration of SAM, as well as the integration of the class-covariance matrix in the calculation of distance between class.

2.1 Few-Shot Learning

As inspired by humans, by being able to classify identities with undoubtedly near-perfect accuracy even only with slightest samples, Few-shot Learning (FSL) was proposed. The goal of the FSL is to overcome the problem of having limited data for the model to learn from.

The learning task of FSL involves classifiers learned from several examples of each labeled identity, as shown in the example of 2-way 3-shot classification in Fig. 1. The training of FSL can be divided into two methods: 1) episodic training and 2) conventional training. In an episodic training scenario, the learning is usually defined as n -way k -shot (or n -identity k -sample) classification. In this case, the training set contains $J = NK$ examples from N classes and K examples. In each epoch, every batch of the training process is called an *episode*, which is built as a subset generated by randomly picking n classes with k examples in each identity for the support set, as well as a few query images for evaluation purposes called the query set, which are all taken from J .

The model uses the support set to learn how to perform classification, which can be considered as “*learn to learn*.” At the evaluation stage, the classification performance is evaluated using the query images based on the knowledge learned from the support set. This training can solve the problem of data imbalance (skewed class proportions) since this method forces the training process to have the same

number of instances.

In conventional training, the training process is the same as that used in non-FSL training. Given adequate examples, this training can be advantageous as it is more adaptive to various n -way k -shot scenarios [19]. Furthermore, it is not always necessary to train the model episodically as the training process is slow (the convergence is much slower than that of a linear classifier), implementations are complex, and the performance decreases dramatically if the parameters (the number of shots and ways) of meta-training and meta-testing do not match [15], [19]–[21]. According to the study [20], fine-tuning an FSL model with a linear classifier also yields a competitive result with a considerably simpler model and fewer hyperparameters.

2.2 Embedding Learning

One of the FSL strategies is embedding learning, which could also be called “learning to compare.” Embedding learning views the classification problem as a comparison task and acquires knowledge of its embedding by comparison with or in combination with other tasks. It uses an embedding function to generate a representation using lower-dimension input data. The embedding function is a feature extractor that employs a CNN model without its final classifier layer or a LSTM model. Then, the distance distribution between samples is modeled using a similarity function to bring members of the same class together and to keep members of other classes apart [9].

A typical example of a distance function used as a similarity function is the Euclidean or cosine distance. Figure 2 shows an illustration of embedding learning. First, support and query sets are built from J . Each support class is labeled with distinct colors that each represent a particular class alongside the query example and is passed to the feature extractor, which outputs embedding vectors. The embedding vectors represent the information needed for distance computation, which is done between support and query sets. The distance from each of the support examples is then calculated with the corresponding query example, and then similarity scores that ultimately point to a particular class are output. Some embedding learning-based methods are described in detail in Table 1. This study is based on the work of Snell et al. regarding PNs [14] as the method is proven and has shown effectiveness and reliability.

2.3 Prototypical Network

The PN method is well known and effective approach to embedding learning methods. To perform classification, a PN first builds a *prototype* (C_n), which is a representation of each class obtained by averaging all feature vectors of examples in the class. The prototype is obtained from computing each example (x_{n_i}) using a previously trained feature extractor, which is a function (f_ϕ) for embedding data, where the ϕ parameters are tunable by training, as illustrated in Fig. 4. The figure shows an example of a 4-way 3-shot setting. The

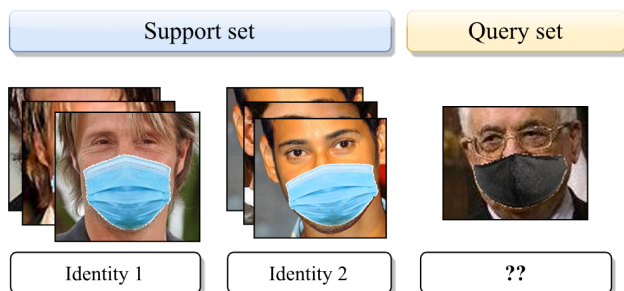


Fig. 1 Typical FSL n -way k -shot scenario with an example of $n = 2$ and $k = 3$.

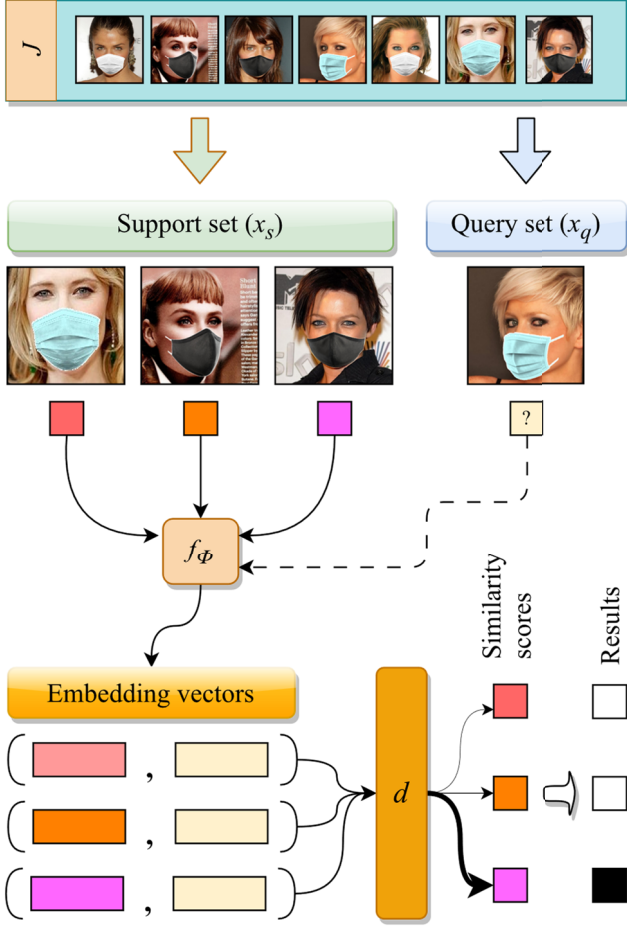


Fig. 2 Embedding learning illustration. The support and query sets built from J are used to make a representation. The support $f_\phi(x_s)$ and query vector $f_\phi(x_q)$ are both returned from the embedding function and then compared using a similarity function $d(\cdot, \cdot)$.

result is passed to an embedding function.

An embedding space is then built, and information is derived from the embedding function. Each support and query member is passed to the embedding function and the embedding space. The prototype is calculated with Eq. (1):

$$C_n = \frac{1}{k} \sum_{i=1}^k f_\phi(x_{n_i}) \quad (1)$$

where n denotes the value of ways (classes) in the support set, while k is the number of shots. After a representation of each class (prototype) is computed, the query sample (x_q) feature vector is built from the query set, and is also computed using the same embedding function. The query set comprises samples (x_q) and labels (y_q). Finally, for classification, a probability distribution derived from the negative distance of classes given a query sample can be calculated as follows:

$$p_\phi(y_q = n | x_q, C_n) = \frac{\exp(-d(f_\phi(x_q), C_n))}{\sum_{j=1}^n \exp(-d(f_\phi(x_q), C_j))} \quad (2)$$

where $d(\cdot, \cdot)$ refers to the distance function calculation. Euclidean distance is used in the PN.

2.4 Prototype Classifier Based on Fine-Tuning a Linear Classifier

By classically training the feature extractor, the training process becomes more straightforward and faster while also giving an agnostic feature extractor that shows a better result with various n -way k -shot scenarios compared to episodic training. Thus, no complicated meta-training algorithm is used in this method. The backbone used for the prototype classifier is trained using standard classification fine-tuning with cross-entropy loss. The ResNet-based backbone was chosen in this work.

Figure 5 shows the architecture of ResNet [22], which has proven effective in many image recognition applications [19]. The model contains 12 layers for ResNet-12 and 18 layers for ResNet-18. The values of A, B, C, and D are 1, 1, 2, and 1 for ResNet-12, while they are 2, 2, 2, and 2 for ResNet-18, respectively. These values indicate the number of blocks that are used to construct the model. This network is a fusion of deep architectural parsing and residual network integrations.

Due to bottleneck blocks, the training time is reasonably faster, thus increasing efficiency. A total of five convolutional blocks make up the network, and some shortcuts are inserted between the various layers. ResNet-12 and ResNet-18 were chosen after considering the tradeoffs between model depth, accuracy, and computing efficiency. These models are deep enough to extract high-level features derived from image inputs while still lightweight enough for efficient training and inference. Therefore, we chose ResNet-12 and ResNet-18 as they provide a good balance of all considered factors, making them well suited for masked face recognition.

For the training stage, we chose a backbone and trained it using training examples in training classes with a feature extractor f_ϕ from scratch. We used cross-entropy loss to minimize the loss in the training set. Hence, the model was not tied to any category. Instead, we wanted the model to generalize any new categories. Therefore, we did not use the last classification layer, which generally used for conventional neural networks.

2.5 Sharpness-Aware Minimization

To improve the model generalization and model accuracy, we used SAM [24], which has been shown to bring significant improvements in prediction performance for deep networks applied in various fields. The applications include image classification and natural language processing. SAM does not use heavy parameterization, which typically relies on decreasing the loss functions (cross-entropy loss), which guarantees only a small amount of generalization capability of the model. Unlike the Stochastic Gradient Descent (SGD) method, which relies solely on the gradient of the loss function to update the model parameters, SAM considers both the gradient of the loss function and the curvature of the loss landscape.

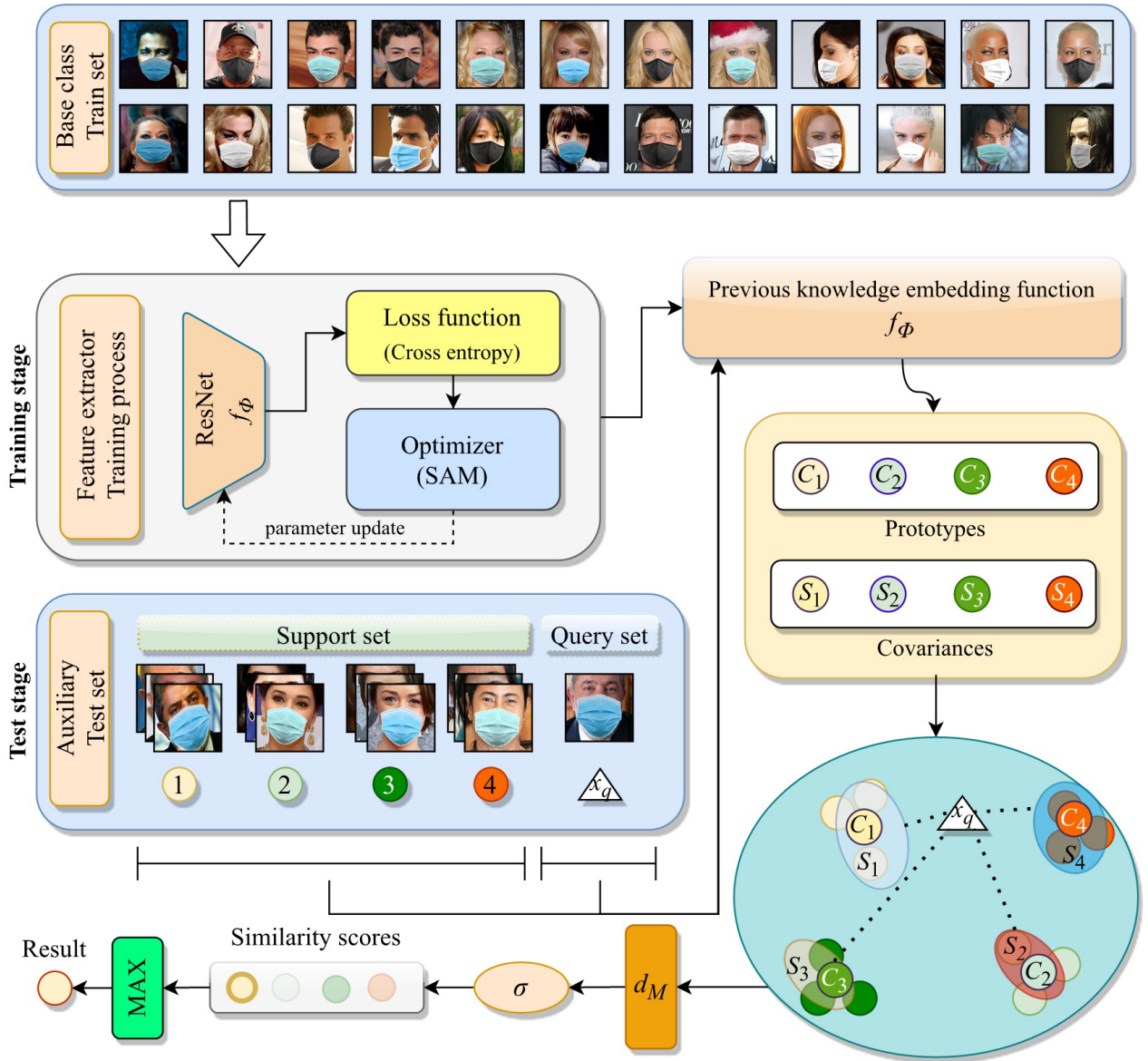


Fig. 3 Illustration showing the flow of CPNet development.

The SAM algorithm tries to find neighborhood parameters with uniformly low loss values rather than those possessing low loss values. The curvature of the loss landscape provides information about the stability of predictions of the model, which is essential for improving its generalization ability. Sharp models with high curvature are more sensitive to small changes in the input and may overfit the training data. In contrast, flat models with low curvature are more robust to changes in the input but may underfit the training data. Therefore, SAM adjusts the parameters of the model to balance the tradeoff between sharpness and stability, ensuring that the predictions of the model are both accurate and robust. Also, SAM suggests using label smoothing alongside the cross-entropy loss on top of the SGD method as a base optimizer to improve the robustness and mitigate overfitting.

2.6 Class-Covariance Matrix Improved Prototype Classifier

A key component for the classification process of the PN is the choice of distance metric. Originally, the PN employed the Euclidean distance function to calculate the distance between the query and prototype representation of each class. However, when using the Euclidean distance, it is assumed that 1) there is no correlation among dimensions in the feature vector, and 2) the dimensions have uniform variance. However, many real-life datasets have correlated features. For example, in an image of a face, the distance between the eyes is likely to be correlated with the overall size of the face. In contrast, Euclidean distance assumes that all dimensions are independent and equal and that all classes possess the

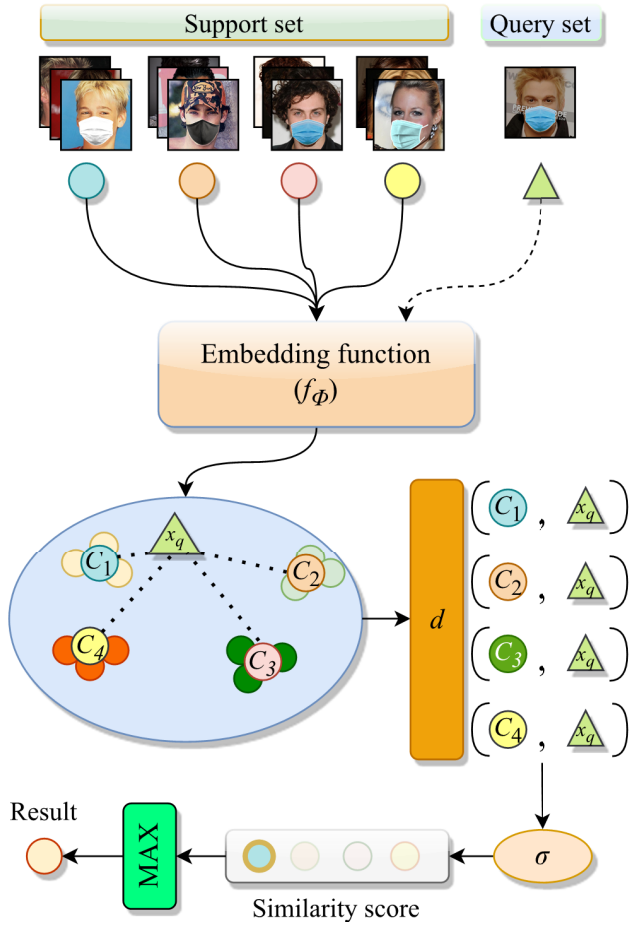


Fig. 4 Illustration of prototypical network model.

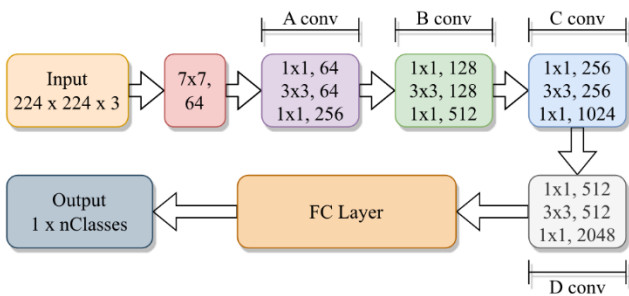


Fig. 5 ResNet architecture [21].

same distribution in the embedding space, which can cause poor outcomes when applied to data that are correlated and or may not have the same variance.

Thus, we propose using the Mahalanobis distance covariance matrix for the distance calculation to estimate and include the spread in the classification strategy. The Mahalanobis distance takes into account the correlation between features, which better captures the structure and variation of the data points. By leveraging the covariance structure, the Mahalanobis distance potentially improves the classifier's accuracy as it considers the covariance information while determining the similarity between data points. As

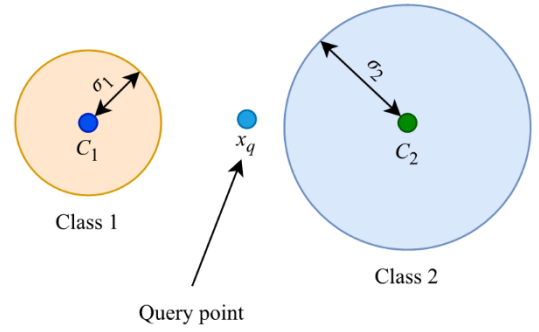


Fig. 6 An example of classes with different variances.

this study employs the correlation among dimensions in the feature vector, the choice of distance metric is very important. Rather than simply calculating the distance between the query and prototype with Euclidean distance, we consider the covariance matrix of each class. The importance of introducing the covariance matrix into the distance calculation is illustrated in Fig. 6.

We need to employ the Mahalanobis distance to use the class-covariance matrix in the distance calculation. The distance between two points, x and y , can be calculated using the Mahalanobis distance as shown in Eq. (3):

$$d_M = \sqrt{(x - y)^T S^{-1} (x - y)} \quad (3)$$

where S^{-1} denotes the inversion of the covariance matrix S and is specific to the current class and task. If the distribution D has an invertible covariance matrix S , then the distance between the distribution with a mean μ and a data point x may also be determined by Eq. (4):

$$d_M = \sqrt{(x - \mu)^T S^{-1} (x - \mu)}. \quad (4)$$

Figure 6 shows a case of classes with different variances. It can be seen that the choice of distance metric is important. The figure shows an example of two-class classification with C_1 and C_2 as prototypes of each class, and the distributions for the spread of each class are labeled as σ_1 and σ_2 , respectively. x_q is a query point. When we use the Euclidean distance as an example, the exact distance between C_1 and x_q and between C_2 and x_q outputs the same probability since the Euclidean distance does not consider the spread of the classes. If we use the Mahalanobis distance, class 2 yields a higher probability since x_q is closer to the data spread of the second class.

We calculate the Mahalanobis distance using Eq. (4). However, the value covariance matrix needs to be estimated using the feature vector from the feature embedding based on a previous study [25], which was derived from the Ledoit-Wolf regularized shrinkage estimator. The covariance matrix can be estimated as follows:

$$S = \delta_c \Sigma_c + (1 - \delta_c) \Sigma_t + I, \quad (5)$$

which is formed from a convex combination of the covariance

matrix of each class in a task Σ_c and the covariance matrix of all classes in a task Σ_t . I is the identity matrix. To weigh the covariance estimation of each class and all classes in a task, we use a simple and particular approach using the deterministic ratio δ_c , which is obtained from the number of shots:

$$\delta_c = \frac{k}{k+1} \quad (6)$$

where k denotes the number of shots or samples in each class. The precise values of the covariance matrix cannot be known in advance, and the number of examples in the support set is typically smaller than the feature space dimension. Thus, we employ Eq. (5) to estimate the covariance structure necessary for distance computation. where k denotes the number of shots or samples in each class. This simple deterministic ratio tells us that when the number of shots is high, the value of δ_c will approach 1, indicating that the estimated covariance matrix will mainly consist of the covariance matrix of each class. In other words, when the number of examples is high (in a high-shot setting), the covariance estimation is primarily influenced by the covariance of each individual class. As the number of shots increases, the quality of the covariance of each class estimation improves, leading to a closer resemblance to the actual covariance.

The overall proposed algorithm can be seen in Algorithm 1. Figure 3 illustrates the overall flow of our work, in which a classifier is trained conventionally using a base class with a ResNet backbone, where the parameter is updated using SAM. The previously learned embedding is then used as the feature embedding for the FSL task. Class representatives (prototypes) and the covariance are the outputs from the image input data and are used in the distance calculation. The distance among queries and prototypes is then negated to obtain similarity scores, which are passed to a Softmax function to output a probability that ultimately leads to a particular prediction.

2.7 Datasets

We used several datasets to validate the effectiveness of the method. The “Celebrities in Frontal” and CelebA-HQ datasets were chosen for the training stage to make the pre-trained classifier model. These two datasets are diverse and include various ethnicities, genders, and ages, making them ideal for training models to recognize faces in various settings. Moreover, due to the wide range of variation, the masked face-recognition issues are included, such as illumination, pose variation, mask model variations, and low image resolution. Table 2 describes the challenges present in each dataset to show the capability of the proposed method for specific issues.

One of the datasets is the “Celebrities in Frontal” dataset [26]. Examples are shown in Fig. 8. It is one of the most frequently used datasets for face recognition tasks and contains 500 subjects with a fixed number of 10 examples per identity. Aligned and cropped versions are also available, so

Algorithm 1: CPNet algorithm. Following the n -way k -shot, n is the number of classes in the support set, while k denotes the number of support examples for each class. COMPUTECOVARIANCE denotes covariance computation from a set of data. UNIQUE returns unique elements from a series of data.

Input: Support image (x_s), Support label (y_s), Query image (x_q), Query label (y_q)
Output: Probability of Query image class membership for each class label

- 1: Embed support image to $f_\phi(x_s) \rightarrow z_{task}$
- 2: $\Sigma_t = \text{COMPUTECOVARIANCE}(z_{task})$
- 3: $C_n = \frac{1}{k} \sum_{i=1}^k f_\phi(x_{n_i})$ ▷ Compute prototype
- 4: **for** each *class* in $\text{UNIQUE}(y_s)$ **do**
- 5: Choose k samples belongs to *class* at random and call them x_{s_class}
- 6: Embed support image belongs to *class* to $f_\phi(x_{s_class}) \rightarrow z_{class}$
- 7: $\Sigma_c = \text{COMPUTECOVARIANCE}(z_{class})$ ▷ Covariance estimation
- 8: $S = \delta_c \Sigma_c + (1-\delta_c) \Sigma_t + I$
- 9: Embed query image to $f_\phi(x_q) \rightarrow z_{query}$
- 10: difference = $C_n - z_{query}$
- 11: distance = difference $\circ S^{-1} \circ$ difference ▷ Distance calculation
- 12: probability = softmax ($-$ distance, y_q)
- 13: **end for**

Table 2 Datasets used and variations present in them.

Face Dataset	Scenarios of Variation			
	Illumination	Pose variation	Mask variation	Low resolution
“Celebrities in Frontal”			✓	✓
CelebA-HQ	✓	✓	✓	
Our dataset A	✓		✓	
Our dataset B	✓		✓	



Fig. 7 Examples of pre- and post-masked “Celebrities in Frontal” dataset. (a) Pre-masked dataset. (b) Post-masked dataset.

there is no need for preprocessing (face cropping and alignment). Out of 500 identities, 300 were randomly chosen as the training dataset, while the other 200 identities were used for the validation and testing dataset with 100 identities each.

To create the masked version of the dataset, we used the MaskTheFace tool [4] to create a simulated masked face dataset. The tool uses open-source face landmark and detection methods, finds estimated mask key positions, estimates the face tilt angle, and then applies a suitable mask image template based on the face tilt. This straightforward tool can generate random mask types and variations with various orientations and lighting conditions. The pre- and post-masked datasets are shown in Fig. 7(a) and 7(b).

The second dataset used in this experiment was the CelebA-HQ dataset [27]. The dataset has also been cropped and aligned. The dataset consists of 30,000 images. Out of more than 10,000 identities, we chose 750 random identities for training for each identity with at least 10 examples for

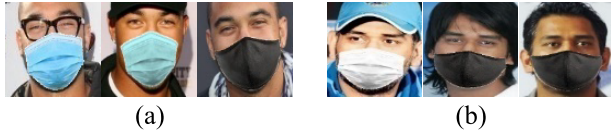


Fig. 8 Examples of dataset on each identity of masked “Celebrities in Frontal” dataset. (a) and (b) indicate two different identities.

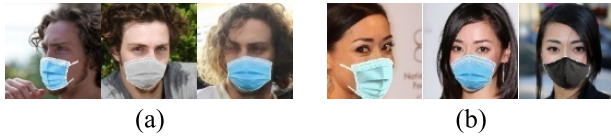


Fig. 9 Examples of dataset on each identity of masked CelebA-HQ dataset. (a) and (b) indicate two different identities.



Fig. 10 Examples of Self Dataset A. (a), (b), (c), and (d) indicate four different identities.

each identity. We then randomized each 150 examples for validation and testing. We ensured that the identities were not overlapping for training, validation, and testing. The same MaskTheFace tool was used to create a simulated masked dataset. Examples of this dataset are shown in Fig. 9.

To validate the performance of our method on a real masked face, we collected two sets of data using various smartphone cameras for testing. Moreover, other researchers have not concentrated on the problem of racial bias, even though it is present in most face recognition methods [28]. Therefore, we tried to evaluate the generalization ability of our method with a wide range of ethnic groups, particularly Asian and African subjects, since numerous face recognition algorithms perform worse on these two specific racial groups [29].

Thus, we gathered Self Dataset A (examples are shown in Fig. 10), which contains 20 Asian subjects, and Self Dataset B (examples are shown in Fig. 11), which contains 20 African subjects. These datasets contain various poses and lighting conditions that are sufficient for representing actual use conditions. Each of the identities has 10 samples, which were selected randomly when inference was performed. The gathered photos were preprocessed (cropped and aligned) using MTCNN [30].

2.8 Experimental Setup and Procedure

An experiment was carried out on a computer with an Intel i7-9700, 16 GB of RAM, an NVIDIA RTX 2080 GPU, and

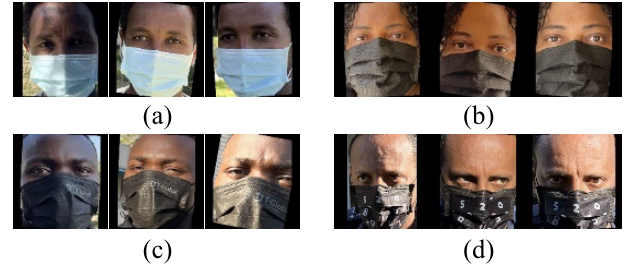


Fig. 11 Examples of Self Dataset B. (a), (b), (c), and (d) indicate four different identities.

the Windows 10 operating system. The entire code was implemented using the PyTorch package, including COMPUTE-COVARIANCE in Algorithm 1, which was used to calculate the covariance estimation of the data. The training image sets were first preprocessed, as shown in Fig. 7, to create the simulated masked face dataset. Three random colors/variations were chosen, as shown in the example in Fig. 8. The images were then resized to 224×224 as the input image size requirement for ResNet-based models.

We employed an extensive data augmentation method using various image transformations to increase the generalization ability of the model. Our augmentations included flipping, color shifting, contrast, and brightness, which were all applied randomly to the training set using Albumentations [31].

To make the classifier, we trained all the backbones for 200 epochs from scratch with a learning rate of 0.1 and a StepLR scheduler to periodically reduce the learning rate. We used a momentum of 0.9 and weight decay of $5e-4$ for both SGD and SAM experiments. The Rho value of SAM was set to 0.05 with a label smoothing value of 0.1, as recommended by the original author. To prevent overfitting, we performed the validation procedure every three epochs and monitored the validation accuracy. The training epoch with the best validation accuracy was used as the model classifier in the testing stage. To report the performance, 100 random test tasks were used, and the evaluation metric was obtained by averaging the results of all tasks. Each task included an n -way k -shot support set with a query sample for each way.

3. Experimental Results

We present the results for models with several different configurations: (1) 5-way (1-shot and 5-shot), (2) 10-way (1-shot and 5-shot), and (3) 20-way (1-shot and 5-shot). We present the results as the accuracy, which is an evaluation metric that is commonly used in evaluating biometric systems. The accuracy is defined as the degree of correctness of predictions for all query images (N_q) inferred in the testing process, as shown in Eq. (7) below:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N_q} \quad (7)$$

where TP is the number of true positive results, and TN is

Table 3 Accuracy results comparison on masked celebrities in frontal dataset.

	Backbone		5-way		10-way		20-way	
			1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Celebrities in Frontal	ResNet-12	Matching network [11]	0.854	0.924	0.760	0.857	0.667	0.820
		Baseline Plus [15]	0.820	0.942	0.722	0.862	0.643	0.846
		Prototypical network [14]	0.848	0.938	0.755	0.875	0.661	0.842
		Relation network [16]	0.852	0.946	0.762	0.864	0.665	0.854
		CPNet (our work)	0.856	0.954	0.768	0.908	0.670	0.865
	ResNet-18	Matching network [11]	0.863	0.938	0.788	0.866	0.712	0.836
		Baseline Plus [15]	0.834	0.956	0.742	0.896	0.677	0.856
		Prototypical network [14]	0.852	0.944	0.771	0.898	0.709	0.854
		Relation network [16]	0.856	0.958	0.782	0.902	0.710	0.863
		CPNet (our work)	0.867	0.972	0.799	0.923	0.718	0.886

Table 4 Accuracy results comparison on masked CelebA-HQ dataset.

	Backbone		5-way		10-way		20-way	
			1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
CelebA-HQ	ResNet-12	Matching network [11]	0.884	0.958	0.851	0.902	0.801	0.880
		Baseline Plus [15]	0.858	0.966	0.836	0.927	0.782	0.912
		Prototypical network [14]	0.876	0.960	0.843	0.926	0.790	0.895
		Relation network [16]	0.882	0.955	0.851	0.925	0.798	0.902
		CPNet (our work)	0.892	0.970	0.860	0.946	0.806	0.921
	ResNet-18	Matching network [11]	0.900	0.970	0.864	0.925	0.809	0.895
		Baseline Plus [15]	0.865	0.969	0.848	0.941	0.795	0.919
		Prototypical network [14]	0.894	0.963	0.857	0.939	0.803	0.901
		Relation network [16]	0.893	0.970	0.862	0.945	0.804	0.918
		CPNet (our work)	0.902	0.974	0.868	0.952	0.815	0.935

the number of true negative results.

3.1 In-Domain Testing

We examined the proposed method’s in-domain results (results obtained after training and testing with the same kind of dataset) and compared them with those of re-implementations of other algorithms. The in-domain test results of CPNet trained using the “Celebrities in Frontal” and CelebA-HQ datasets can be seen in Tables 3 and 4. There was improvement when using the deeper and heavier backbone, ResNet-18. Slight performance gains were seen in this in-domain testing. The in-domain testing achieved high accuracy for both datasets even in high-way scenarios. Although it is more difficult to qualitatively judge the difficulty of the dataset, the CelebA-HQ dataset has reasonably high resolution, making our model easier to represent each identity.

In contrast, the “Celebrities in Frontal” dataset presents a challenge of low resolution, making it harder to form a good representation, thus giving a slightly lower recognition accuracy. Overall, CPNet obtained reasonably good performance ranging from 67.0% accuracy (0.9% improvement over original PN) in the 20-identity 1-example scenario to 95.4% accuracy (1.6% improvement over original PN) in the 5-identity 5-example scenario. With higher ways and shots, as in the 20-way 5-shot scenario, the CPNet even achieved more than 3% accuracy improvement over the original PN for both datasets trained with the ResNet-18 backbone. One interesting result is that CPNet had more significant improvement at higher shots due to the integration of the covariance structure of the data.

3.2 Out-of-Domain Testing

We used the classifier trained using the “Celebrities in Frontal” and CelebA-HQ datasets to perform few-shot

Table 5 OUT-domain testing results using masked celebrities in frontal as training set.

	Backbone		5-way		10-way		20-way	
			1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Self-Dataset A	ResNet-12	Matching network [11]	0.888	0.920	0.782	0.885	0.672	0.849
		Baseline Plus [15]	0.866	0.942	0.761	0.919	0.664	0.870
		Prototypical network [14]	0.878	0.940	0.768	0.908	0.667	0.859
		Relation network [16]	0.882	0.942	0.762	0.912	0.668	0.864
		CPNet (our work)	0.892	0.950	0.785	0.938	0.678	0.879
	ResNet-18	Matching network [11]	0.903	0.930	0.795	0.900	0.708	0.872
		Baseline Plus [15]	0.884	0.952	0.774	0.936	0.697	0.896
		Prototypical network [14]	0.899	0.945	0.786	0.922	0.704	0.886
		Relation network [16]	0.902	0.936	0.797	0.940	0.712	0.902
		CPNet (our work)	0.912	0.966	0.805	0.954	0.719	0.924
Self-Dataset B	ResNet-12	Matching network [11]	0.798	0.902	0.705	0.865	0.601	0.842
		Baseline Plus [15]	0.794	0.924	0.695	0.875	0.590	0.864
		Prototypical network [14]	0.790	0.920	0.700	0.871	0.599	0.858
		Relation network [16]	0.792	0.922	0.696	0.875	0.602	0.866
		CPNet (our work)	0.806	0.927	0.709	0.888	0.613	0.876
	ResNet-18	Matching network [11]	0.820	0.914	0.736	0.883	0.639	0.872
		Baseline Plus [15]	0.802	0.932	0.720	0.909	0.627	0.890
		Prototypical network [14]	0.818	0.930	0.733	0.908	0.635	0.880
		Relation network [16]	0.822	0.933	0.738	0.912	0.638	0.892
		CPNet (our work)	0.826	0.944	0.740	0.926	0.648	0.902

Table 6 OUT-domain testing results using masked CelebA-HQ as training set.

	Backbone		5-way		10-way		20-way	
			1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Self-Dataset A	ResNet-12	Matching network [11]	0.868	0.944	0.795	0.905	0.708	0.884
		Baseline Plus [15]	0.858	0.967	0.788	0.929	0.694	0.906
		Prototypical network [14]	0.862	0.962	0.791	0.928	0.706	0.900
		Relation network [16]	0.854	0.968	0.795	0.932	0.702	0.904
		CPNet (our work)	0.876	0.980	0.802	0.948	0.716	0.926
	ResNet-18	Matching network [11]	0.904	0.956	0.837	0.928	0.755	0.902
		Baseline Plus [15]	0.896	0.974	0.818	0.953	0.735	0.942
		Prototypical network [14]	0.894	0.964	0.830	0.946	0.741	0.933
		Relation network [16]	0.896	0.972	0.828	0.952	0.742	0.945
		CPNet (our work)	0.908	0.986	0.841	0.965	0.760	0.953
Self-Dataset B	ResNet-12	Matching network [11]	0.820	0.911	0.737	0.884	0.640	0.842
		Baseline Plus [15]	0.800	0.932	0.719	0.915	0.630	0.876
		Prototypical network [14]	0.816	0.926	0.733	0.900	0.637	0.867
		Relation network [16]	0.822	0.910	0.736	0.895	0.642	0.876
		CPNet (our work)	0.826	0.944	0.740	0.926	0.648	0.892
	ResNet-18	Matching network [11]	0.845	0.927	0.750	0.919	0.671	0.874
		Baseline Plus [15]	0.825	0.949	0.739	0.937	0.655	0.900
		Prototypical network [14]	0.837	0.946	0.745	0.930	0.665	0.893
		Relation network [16]	0.838	0.952	0.755	0.945	0.672	0.902
		CPNet (our work)	0.849	0.958	0.764	0.950	0.679	0.927

masked face recognition with other datasets. Using the knowledge from other training set, we performed cross-domain testing on the datasets that we collected to evaluate the capability of our proposed method in different domains, especially for various ethnicities. The results of out-of-domain testing can be seen in Tables 5 and 6.

The model trained with CelebA-HQ yielded better results than the model trained with the “Celebrities in Frontal” dataset. Our model performed best when it was trained with CelebA-HQ using a ResNet-18 backbone and tested with Self Dataset A (98.6%) and Self Dataset B (95.8%) in a 5-way 5-shot setting. The proposed method CPNet outperformed other methods in terms of accuracy for all settings of ways and shots. The accuracies showed 0.6% and 1.4% improvements compared to PN, respectively.

In lower-shot settings, our proposed method was comparable to the matching network (MN), which consistently worked well in lower-shot scenarios, showing improvements of 0.8% in the 20-way 1-shot setting for both Self Datasets A and B (trained with CelebA-HQ and ResNet-12). In higher-shot settings, Baseline Plus worked well in higher-shot scenarios compared with CPNet. However, our method achieved 92.7% accuracy in a 20-way 5-shot setting for Self

Table 7 Reduced dataset scenario for CelebA-HQ dataset using ResNet-12.

Class	Method	5-way		10-way		20-way	
		1s	5s	1s	5s	1s	5s
300	PN [14]	0.788	0.924	0.702	0.876	0.653	0.825
	Our work	0.804	0.952	0.726	0.906	0.674	0.869
500	PN [14]	0.862	0.948	0.791	0.910	0.704	0.872
	Our work	0.878	0.956	0.812	0.932	0.735	0.904
744	PN [14]	0.876	0.960	0.843	0.926	0.790	0.895
	Our work	0.892	0.970	0.860	0.946	0.806	0.921

Dataset B (trained with CelebA-HQ and ResNet-18), which was a 2.7% improvement over Baseline Plus. Moreover, the domain shift verification that we conducted on all methods also showed that our method improves accuracy over the other methods in all scenarios and datasets.

3.3 Parameters Analysis

3.3.1 Effect of Reduced Number of Training Classes

We conducted several experiments to examine the response of the model to the number of classes in the training phase with a reduced number of classes, as well as the influence of the number of classes (n -way) and the size of the support set (k -shot). First, we analyzed the effect of reduced training classes on the classifier accuracy, as shown in Table 7. The table shows the effect of the number of training samples on CPNet and PN on in-domain testing accuracy.

We performed a comparison using the CelebA-HQ dataset with class reductions of 300 and 500. There was a significant accuracy improvement with the 300-class reduction, and our method performed significantly better than PN. This happened because CPNet makes a better feature space and distance judgment with a better feature extractor, thus improving the accuracy.

3.3.2 Impact of Class Size (n -Way) and Number of Supports (k -Shot)

As shown in Fig. 12, we investigated the influence of class size and support size on the accuracy by varying the values of n and k . We used PN as a natural baseline for comparison to demonstrate the enhancements and advancements achieved by our proposed approach. According to Fig. 12(a), the performance decreased as more classes were introduced, leading to increased difficulty of classification. Figure 12(b) shows that the performance grew as the value of k increased, meaning that the model made a better representation during the recognition. Furthermore, compared with PN, CPNet led to an improvement of performance. The figures show that CPNet empirically handles the recognition better than PN.

3.4 Ablation Study

We conducted ablation experiments to assess the effects of the various modules used in this study. We conducted the ablation experiment using ResNet-12 as a representative backbone. The first scenario employed SGD and the Euclidean

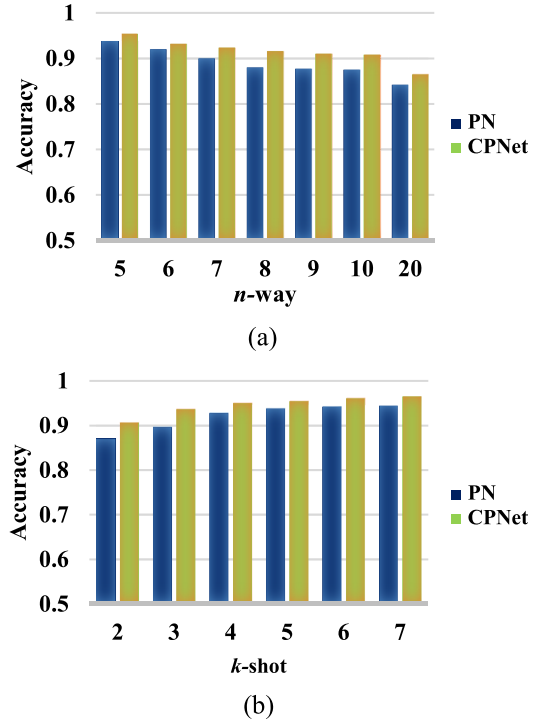


Fig. 12 Effect of n -way k -shot on PN (prototypical network) and CPNet recognition accuracy. (a) Accuracy of 5-shot with varying n -way, (b) accuracy of 5-way with varying k -shot.

Table 8 Ablation study of accuracy on ResNet-12 using “celebrities in frontal” dataset.

Module used	5-way		10-way		20-way	
	1s	5s	1s	5s	1s	5s
DA	0.848	0.938	0.755	0.875	0.661	0.842
DA + SAM	0.854	0.942	0.764	0.887	0.667	0.851
DA + SAM + CC (CPNet, our work)	0.856	0.954	0.768	0.908	0.670	0.865

Table 9 Ablation study of accuracy on ResNet-12 using CelebA-HQ dataset.

Module used	5-way		10-way		20-way	
	1s	5s	1s	5s	1s	5s
DA	0.876	0.960	0.843	0.926	0.790	0.895
DA + SAM	0.884	0.964	0.851	0.929	0.800	0.902
DA + SAM + CC (CPNet, our work)	0.892	0.974	0.860	0.946	0.806	0.921

distance metric, and we looked at the performance improvement by implementing particularly heavy data augmentation used in the training stage, which was labeled as DA. Next, we looked at the effect of integrating the SAM in the training stage in conjunction with the Euclidean distance metric, which was labeled as DA+SAM. Lastly, we looked at our full CPNet model, which combines the previous two modules alongside the class-covariance matrix in the distance calculation process. The findings of the ablation experiment are shown in Tables 8 and 9.

Tables 8 and 9 show that CPNet outperformed the baseline model for the “Celebrities in Frontal” and CelebA-HQ datasets, demonstrating the robustness of the proposed

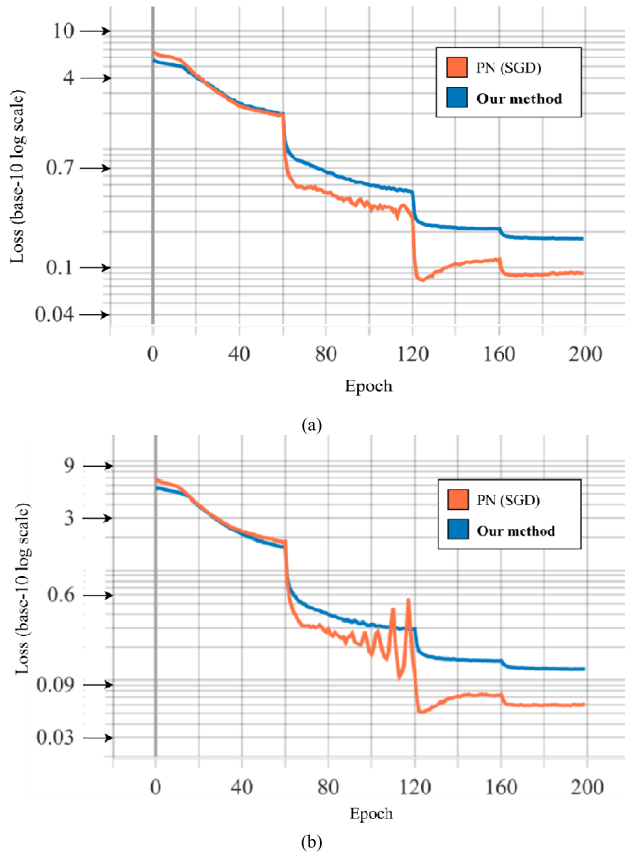


Fig. 13 Training loss with CelebA-HQ dataset using (a) ResNet-12 backbone and (b) ResNet-18 backbone.

distance calculation method, which considers the class-covariance matrix. Improvements from integrating the covariance structure of the data were more evident in 5-shot scenarios. The results showed a large accuracy increase due to the effect of the data spread in higher shots scenarios as the model can use more information from the samples and make a better hypothesis than the PN baseline.

Figure 13 shows the training loss after training with ResNet-12 and ResNet-18. With integration of SAM, CPNet (blue line) converged to a much more stable level than PN, which used SGD (orange line), as shown in the graph. The y -axis (loss) values are shown on a non-linear scale to emphasize the effect. However, after the same number of epochs (200 epochs), the SAM-based loss graph did not decrease anymore compared to the SGD as it tried to find a uniformly low loss value rather than minimizing loss values individually.

Although the loss was not lower, it is important to note that the capability of SAM in avoiding the overfitting issue has improved the performance. This outcome suggests that the integration of SAM has a positive impact across all ablation experiment scenarios. SAM not only improves the optimization process by reducing the loss, but also helps the model navigate a more favorable loss landscape for generalization. By considering both the loss value and the landscape, we ensure that this approach enhances the overall

performance of the model.

It is worth mentioning that achieving a lower loss does not necessarily guarantee good generalization ability or improved accuracy. Therefore, the focus has been on enhancing the model's ability to generalize well rather than solely the lowest possible loss value. In short, compared with PN, which uses the SGD, our method CPNet improves the overall generalization capability of the model.

4. Conclusion

We have presented CPNet using a few-shot-based method that uses an improved covariance-matrix prototype classifier and SAM to improve the classifier further for a limited sample of masked face recognition data. Our work is unique in terms of tackling the difficulties of collecting massive samples of masked face data and providing convenience when a new identity needs to be added or irrelevant ones need to be removed since no retraining is needed. We experimented with various backbones, namely, the ResNet-12 and ResNet-18, and different datasets to evaluate a wide range of settings and obtained promising results. We obtained 95.3% accuracy in the 20-way 5-shot setting with Self Dataset A and 92.7% with Self Dataset B, which showed 2.0% and 3.4% accuracy improvements over the reported baseline results, respectively.

Thus, these results show the feasibility of using our method for limited masked face recognition samples. Although we have successfully developed and experimented with CPNet, there is room for improvement. Further research should concentrate on developing a more efficient way to consider the covariance matrix in the distance calculation process, which restricts the performance in some cases, especially in higher-shot settings.

Acknowledgments

The authors would also like to thank the Associate Editor and the anonymous reviewers for their helpful comments that helped enhance the quality of this research. This work is financially supported in part by National Taiwan University of Science and Technology and MacKay Memorial Hospital, under Grants MMH-NTUST-103-06 and MMH-NTUST-104-06, and in part by the Ministry of Science and Technology (MOST), Taiwan, under Grants MOST 109-2221-E-011-074, MOST 110-2221-E-011-121, MOST 111-2221-E-011-146-MY2, and the National Science and Technology Council, Taiwan, under the Grant NSTC 113-2221-E-011-089.

Declarations

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

- [1] S. Minaee, A. Abdolrashidi, H. Su, M. Bennamoun, and D. Zhang,

- “Biometrics recognition using deep learning: A survey,” *Artif. Intell. Rev.*, vol.56, pp.8647–8695, Jan. 2023.
- [2] A. Alzu’bi, F. Albalas, T. Al-Hadhrani, L.B. Younis, and A. Bashayreh, “Masked face recognition using deep learning: A review,” *Electronics (Switzerland)*, vol.10, no.21, 2666, Oct. 2021.
- [3] L. Cao, X. Huo, Y. Guo, and K. Du, “Sketch face recognition via cascaded transformation generation network,” *IEICE Trans. Fundamentals*, vol.E104-A, no.10, pp.1403–1415, Oct. 2021.
- [4] A. Anwar and A. Raychowdhury, “Masked face recognition for secure authentication,” *arXiv:2008.11104*, Aug. 2020.
- [5] B. Mandal, A. Okeukwu, and Y. Theis, “Masked face recognition using ResNet-50,” *arXiv:2104.08997*, April 2021.
- [6] W. Hariri, “Efficient masked face recognition method during the COVID-19 pandemic,” *Signal Image Video Process*, vol.16, no.3, pp.605–612, April 2022.
- [7] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Process. Mag.*, vol.29, no.6, pp.82–97, Nov. 2012.
- [8] X. Tan, S. Chen, Z.-H. Zhou, and F. Zhang, “Face recognition from a single image per person: A survey,” *Pattern Recognit.*, vol.39, no.9, pp.1725–1745, Sept. 2006.
- [9] Y. Song, T. Wang, P. Cai, S.K. Mondal, and J.P. Sahoo, “A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities,” *ACM Comput. Surv.*, vol.55, no.13s, pp.1–40, Feb. 2023.
- [10] M. Fink, “Object classification from a single example utilizing class relevance metrics,” *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec. 2004.
- [11] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” *Advances in Neural Information Processing Systems*, vol.1, pp.3637–3645, 2016.
- [12] L. Bertinetto, J.F. Henriques, J. Valmadre, P.H. S. Torr, and A. Vedaldi, “Learning feed-forward one-shot learners,” *arXiv:1606.05233*, June 2016.
- [13] A. Brock, T. Lim, J.M. Ritchie, and N. Weston, “SMASH: one-shot model architecture search through hypernetworks,” *arXiv:1708.05344*, Aug. 2017.
- [14] J. Snell, K. Swersky, and R.S. Zemel, “Prototypical networks for few-shot learning,” *arXiv:1703.05175*, March 2017.
- [15] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C.F. Wang, and J.-B. Huang, “A closer look at few-shot classification,” *arXiv:1904.04232*, April 2019.
- [16] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H.S. Torr, and T.M. Hospedales, “Learning to compare: Relation network for few-shot learning,” *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.1199–1208, 2018.
- [17] A. Holkar, R. Walambe, and K. Kotecha, “Few-shot learning for face recognition in the presence of image discrepancies for limited multi-class datasets,” *Image Vis. Comput.*, vol.120, no.104420, 2022.
- [18] X. Yang, M. Han, Y. Luo, H. Hu, and Y. Wen, “Two-stream prototype learning network for few-shot face recognition under occlusions,” *IEEE Trans. Multimedia*, vol.25, pp.1555–1563, 2023.
- [19] M. Hou and I. Sato, “A closer look at prototype classifier for few-shot image classification,” *arXiv:2110.05076*, Oct. 2021.
- [20] S. Laenen and L. Bertinetto, “On episodes, prototypical networks, and few-shot learning,” *arXiv:2012.09831*, Dec. 2020.
- [21] G.S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, “A baseline for few-shot image classification,” *arXiv:1909.02729*, Sept. 2019.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *arXiv:1512.03385*, Dec. 2015.
- [23] S.-H. Hsiao and J.-S.R. Jang, “Improving ResNet-based feature extractor for face recognition via re-ranking and approximate nearest neighbor,” *Proc. IEEE International Conference on Advanced Video and Signal Based Surveillance*, Taipei, Taiwan, pp.1–8, Sept. 2019.
- [24] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, “Sharpness-aware minimization for efficiently improving generalization,” *arXiv:2010.01412*, Oct. 2020.
- [25] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, “Improved few-shot visual classification,” *arXiv:1912.03432*, Dec. 2019.
- [26] S. Sengupta, J.-C. Chen, C. Castillo, V.M. Patel, R. Chellappa, and D.W. Jacobs, “Frontal to profile face verification in the wild,” *Proc. IEEE Winter Conference on Applications of Computer Vision*, New York, United States of America, pp.1–9, March 2016.
- [27] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” *arXiv:1411.7766*, Nov. 2014.
- [28] M. Du, F. Yang, N. Zou, and X. Hu, “Fairness in deep learning: A computational perspective,” *IEEE Intell. Syst.*, vol.36, no.4, pp.25–34, July 2021.
- [29] P.J. Grother, M.L. Ngan, and K.K. Hanaoka, “Face recognition vendor test part 3: Demographic effects,” *NIST Interagency/Internal Report (NISTIR)*, no.8280, Dec. 2019.
- [30] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Process. Lett.*, vol.23, no.10, pp.1499–1503, Oct. 2016.
- [31] A. Buslaev, V.I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A.A. Kalinin, “Albumentations: Fast and flexible image augmentations,” *Information*, vol.11, no.2, 125, Feb. 2020.



Sendren Sheng-Dong Xu received the Ph.D. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2009. He is currently an Associate Professor at the Department of Power Mechanical Engineering, National Tsing Hua University, Hsinchu, Taiwan. His current research interests include intelligent control systems, signal processing, image processing, embedded systems, and next-generation communications. He is serving as an Associate Editor for some international journals, including *IEEE Trans. Industrial Informatics*, *ISA Transactions*, *International Journal of Control, Automation, and Systems*, *Neural Processing Letters*, and *IEICE Trans. Fundamentals of Electronics, Communications, and Computers*.



Albertus Andrie Christian received the M.S. degree at the Graduate Institute of Automation and Control, National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include image processing, computer vision, face recognition, and machine learning.



Chien-Peng Ho received the Ph.D. degree in Computer Science and Engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2012. He is currently an Associate Professor at the Department of Communication Engineering, Asia Eastern University of Science and Technology, New Taipei City, Taiwan. His current research interests include intelligent robot systems, signal processing, image processing, video coding systems, and multimedia communications. He has been involved in the MPEG-21 Testbed

project of ISO/IEC TR 21000.



Shun-Long Weng received the M.D./Ph.D. degree in Biomedical Science & Engineering from National Chiao Tung University, Hsinchu, Taiwan. He is currently a superintendent and an attending physician of the Department of Obstetrics and Gynecology at Hsinchu MacKay Memorial Hospital and Hsinchu Municipal MacKay Children's Hospital. He is also an associate professor in the Department of Medicine at MacKay Medical College, New Taipei City, Taiwan. His research interests include obstetrics/gynecology,

infertility and reproductive endocrinology, computational biology, microbiome analysis, and machine learning. In addition, He is serving as an editorial board member for the Taiwanese Journal of Obstetrics and Gynecology.