[Paper]

ICLR

HyperFace: Generating Synthetic Face Recognition Datasets by Exploring Face Embedding Hypersphere

Hatef Otroshi Shahreza[1,2], Sébastien Marcel[1,3]

[1] Idiap Research Institute, Martigny, Switzerland
[2] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[3] Université de Lausanne (UNIL), Lausanne, Switzerland

idiap RESEARCH INSTITUTE
EPFL
Unil
UNIL | Université de Lausanne

Introduction

- Training face recognition models require **large-scale** datasets.
  - different identities and several samples per identity

Introduction

- Training face recognition models require **large-scale** datasets.
    - different identities and several samples per identity
- Existing ***real* face recognition datasets** are collected by **crawling** internet without individual's consent: e.g., MS-Celeb-1M, WebFace260M, etc.
    - Therefore, existing face recognition datasets have **privacy** and **legal concerns**.
    - Given such concerns some of these datasets are **retracted**. eg: MS-Celeb-1M

Introduction

- Training face recognition models require **large-scale** datasets.
  - different identities and several samples per identity
- Existing ***real* face recognition datasets** are collected by **crawling** internet without individual's consent: e.g., MS-Celeb-1M, WebFace260M, etc.
  - Therefore, existing face recognition datasets have **privacy** and **legal concerns**.
  - Given such concerns some of these datasets are **retracted**. eg: MS-Celeb-1M

- **Solutions:**
  - Use generative models to generate **synthetic datasets** for training face recognition models

Introduction

- For **generating synthetic face recognition dataset,** we need to generate **inter-class** and **intra-class** variations

Introduction

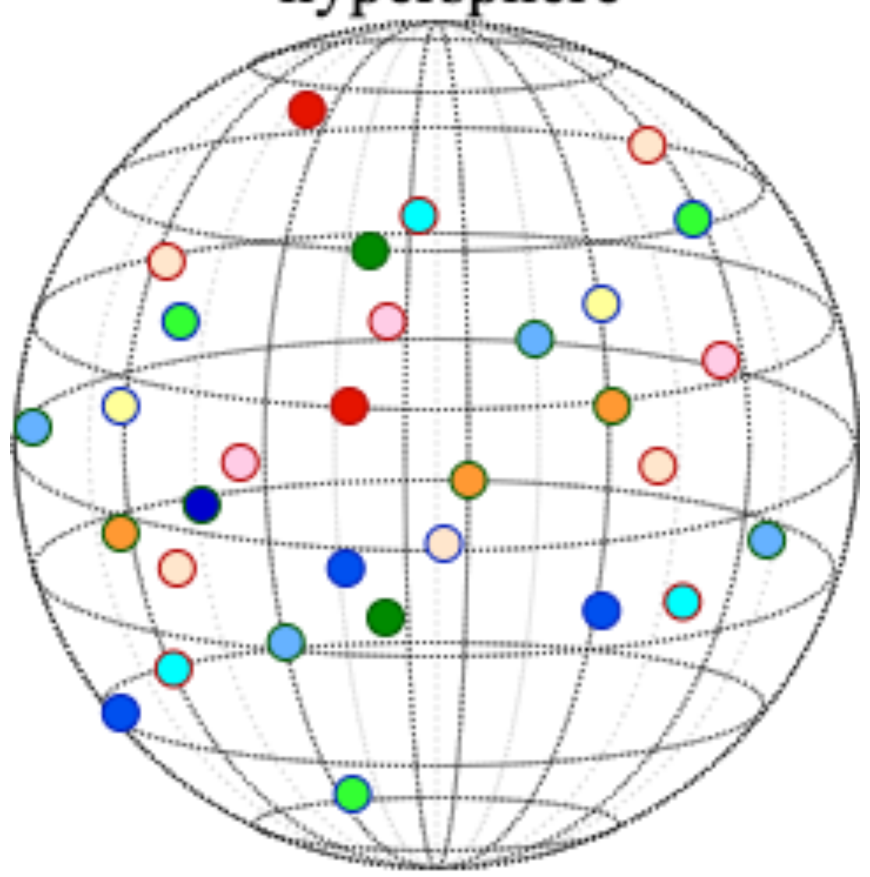
- For **generating synthetic face recognition dataset,** we need to generate **inter-class** and **intra-class** variations
  - Increasing **intra-class** variations can be achieved by conditioning generative models

Introduction

- For **generating synthetic face recognition dataset,** we need to generate **inter-class** and **intra-class** variations
  - Increasing **intra-class** variations can be achieved by conditioning generative models
  - But how we can increase **inter-class** variations?

Problem Formulation

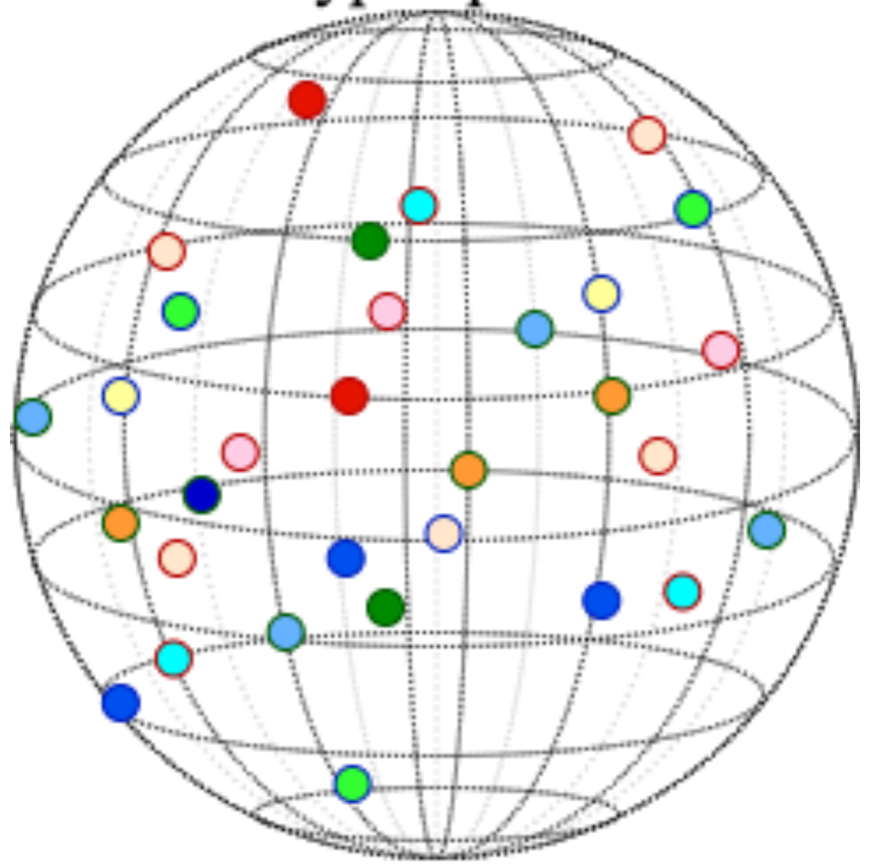
- **Identity Hypersphere**
  - Face Recognition model $F : I \rightarrow X$
  - we can assume that the extracted identity features cover a unit hypersphere (otherwise we normalise it)



Problem Formulation

- **Identity Hypersphere**
    - Face Recognition model $F : I \rightarrow X$
    - we can assume that the extracted identity features cover a unit hypersphere (otherwise we normalise it)

- **Representing Synthetic Dataset on the Identity Hypersphere**
    - we can extract embeddings from all images in the dataset and represent the dataset on the surface of a unit hypersphere $\{\boldsymbol{x}_{\mathrm{ref},i}\}_{i=1}^{n_{\mathrm{id}}}$

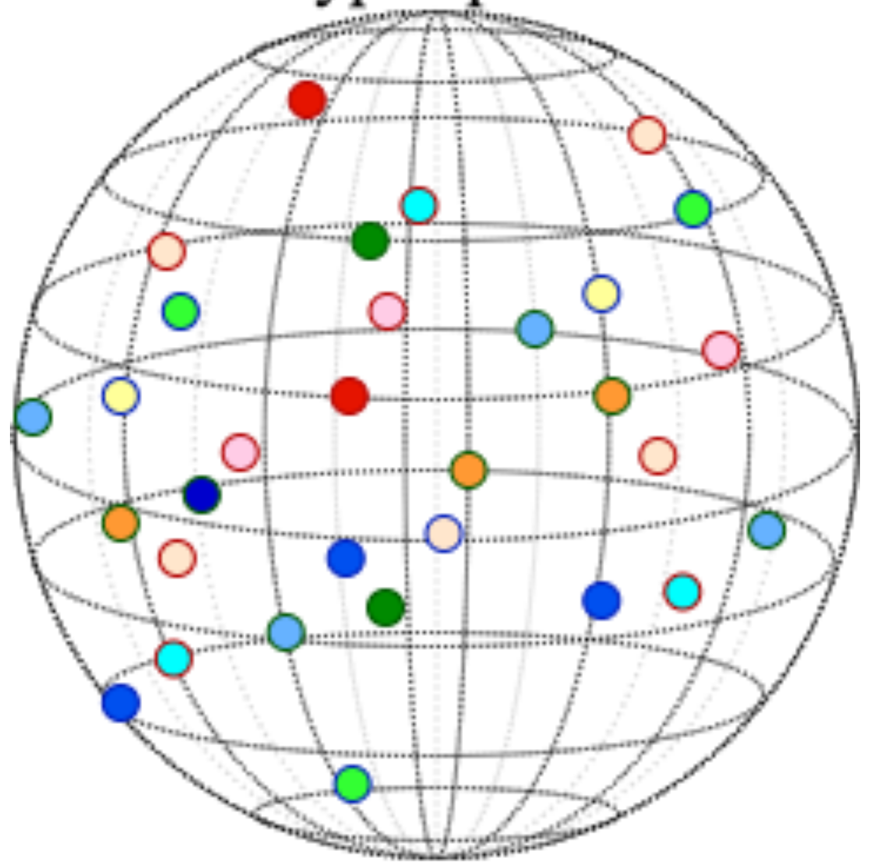


Problem Formulation

- **Identity Hypersphere**
  - Face Recognition model $F : I \rightarrow X$
  - we can assume that the extracted identity features cover a unit hypersphere (otherwise we normalise it)

- **Representing Synthetic Dataset on the Identity Hypersphere**
  - we can extract embeddings from all images in the dataset and represent the dataset on the surface of a unit hypersphere $\{\boldsymbol{x}_{\mathrm{ref},i}\}_{i=1}^{n_{\mathrm{id}}}$

- **How should reference embeddings cover the identity hypersphere?**



HyperFace Optimization

- Since we would like to have a high inter-class variation in the generated dataset, we can say that we need to **maximize** the distances between reference embeddings $\{\boldsymbol{x}_{\text{ref},i}\}_{i=1}^{n_{\text{id}}}$

- In other words, we need to solve the following optimization problem:

$$\max \quad \min_{\{\boldsymbol{x}_{\text{ref}}\}, i \neq j} d(\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j}) \qquad \text{subject to} \quad ||\boldsymbol{x}_{\text{ref},k}||_2 = 1, \forall k \in \{1, ..., n_{\text{id}}\}$$

HyperFace Optimization

- **Regularization:**
  - The **manifold of embeddings** does not necessarily **cover** the **whole surface** of the hypersphere.

HyperFace Optimization

- **Regularization:**
  - The **manifold of embeddings** does not necessarily **cover** the **whole surface** of the hypersphere.
  - We need to add a **regularization term** to our optimization problem that tends to **keep** the reference embeddings on the **manifold**

HyperFace Optimization

- **Regularization:**
  - The **manifold of embeddings** does not necessarily **cover** the **whole surface** of the hypersphere.
  - We need to add a **regularization term** to our optimization problem that tends to **keep** the reference embeddings on the **manifold**
  - We consider a set of face images (we call it a ***gallery***) and extract their embeddings. We use these embeddings to **represent the embedding manifold**. Therefore, as a **regularization** term, we try to minimize the distance of points with the gallery:

$$\min \quad \max_{\{\boldsymbol{x}_{\text{ref}}\}, i \neq j} -d(\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j}) + \alpha \underbrace{\frac{1}{n_{\text{id}}} \sum_{k=1}^{n_{\text{id}}} \min_{\{\boldsymbol{x}_g\}_{g=1}^{n_{\text{gallery}}}} d(\boldsymbol{x}_{\text{ref},k}, \boldsymbol{x}_g)}_{\text{regularization}};$$

$$\text{subject to} \quad ||\boldsymbol{x}_{\text{ref},k}||_2 = 1, \forall k \in \{1, ..., n_{\text{id}}\},$$

HyperFace Optimization

- **Regularization:**
  - The **manifold of embeddings** does not necessarily **cover** the **whole surface** of the hypersphere.
  - We need to add a **regularization term** to our optimization problem that tends to **keep** the reference embeddings on the **manifold**
  - We consider a set of face images (we call it a ***gallery***) and extract their embeddings. We use these embeddings to **represent the embedding manifold**. Therefore, as a **regularization** term, we try to minimize the distance of points with the gallery:

$$\min \quad \max_{\{\boldsymbol{x}_{\text{ref}}\}, i \neq j} -d(\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j}) + \alpha \underbrace{\frac{1}{n_{\text{id}}} \sum_{k=1}^{n_{\text{id}}} \min_{\{\boldsymbol{x}_g\}_{g=1}^{n_{\text{gallery}}}} d(\boldsymbol{x}_{\text{ref},k}, \boldsymbol{x}_g)}_{\text{regularization}};$$

$$\text{subject to} \quad ||\boldsymbol{x}_{\text{ref},k}||_2 = 1, \forall k \in \{1, ..., n_{\text{id}}\},$$

  - **Note**: The gallery can be from synthetic images.

HyperFace Optimization

Algorithm 1 HyperFace Optimization for Finding Reference Embeddings

---

Inputs: λ : learning rate, n_{itr} : number of iterations, $\{\boldsymbol{x}_g\}_{g=1}^{n_{\text{gallery}}}$: embeddings of a gallery of face images,
α : hyperparameter (contribution of regularization).
Output: $\boldsymbol{X}_{\text{ref}} = \{\boldsymbol{x}_{\text{ref},i}\}_{i=1}^{n_{\text{id}}}$: optimized reference embeddings.
Procedure:
  Initialize reference embeddings $\{\boldsymbol{x}_{\text{ref},i}\}_{i=1}^{n_{\text{id}}}$
  For $n = 1, .., n_{\text{itr}}$ **do**
    Find $\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j} \in \boldsymbol{X}_{\text{ref}}$ which have minimum distance $d(\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j})$
    Reg $\leftarrow \frac{1}{n_{\text{id}}} \sum_{k=1}^{n_{\text{id}}} \min_{\{\boldsymbol{x}_g\}_{\text{gallery}}} d(\boldsymbol{x}_{\text{ref},k}, \boldsymbol{x}_g)$ ▷ Calculate the regularization term
    cost $\leftarrow -d(\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j})$
    $\boldsymbol{X}_{\text{ref}} \leftarrow \boldsymbol{X}_{\text{ref}} - \text{Adam}(\nabla \text{cost}, \lambda)$
    $\boldsymbol{X}_{\text{ref}} \leftarrow \text{normalize}(\boldsymbol{X}_{\text{ref}})$ ▷ To ensure that resulting embeddings $\boldsymbol{X}_{\text{ref}}$ remain on the hypersphere.
  End For
End Procedure

Image Generation

- We use a (conditional) diffusion model *G* which can generate face image $\boldsymbol{I} = G(\boldsymbol{x}_{\text{ref}}, \boldsymbol{z})$ from reference embedding $\boldsymbol{x}_{\text{ref}}$

Image Generation

- We use a (conditional) diffusion model G which can generate face image $\boldsymbol{I} = G(\boldsymbol{x}_{\text{ref}}, \boldsymbol{z})$ from reference embedding $\boldsymbol{x}_{\text{ref}}$
- By changing noise $\boldsymbol{z}$, we can generate different samples per identity.

Image Generation

- We use a (conditional) diffusion model *G* which can generate face image $\boldsymbol{I} = G(\boldsymbol{x}_{\text{ref}}, \boldsymbol{z})$ from reference embedding $\boldsymbol{x}_{\text{ref}}$
- By changing noise $\boldsymbol{z}$, we can generate different samples per identity.
- Moreover, we can also add small noise on the reference embedding and generate different images:

$$\boldsymbol{I} = G(\frac{\boldsymbol{x}_{\text{ref}} + \beta \boldsymbol{v}}{||\boldsymbol{x}_{\text{ref}} + \beta \boldsymbol{v}||_2}, \boldsymbol{z}), \quad \boldsymbol{v} \sim \mathcal{N}(0, \mathbb{I}^{n_{\mathcal{X}}}), \boldsymbol{z} \sim \mathcal{N}(0, \mathbb{I}^{\text{DM}})$$

Evaluation

- To evaluate our approach, we use the generated dataset to **train a face recognition model** and compare with face recognition model trained with similar configuration using existing synthetic datasets. Then, we benchmark trained face recognition model for each synthetic dataset:

Evaluation

- To evaluate our approach, we use the generated dataset to **train a face recognition model** and compare with face recognition model trained with similar configuration using existing synthetic datasets. Then, we benchmark trained face recognition model for each synthetic dataset:

Dataset name	# IDs	# Images	LFW	CPLFW	CALFW	CFP	AgeDB
SynFace (Qiu et al., 2021)	10'000	999'994	86.57	65.10	70.08	66.79	59.13
SFace (Boutros et al., 2022)	10'572	1'885'877	93.65	74.90	80.97	75.36	70.32
Syn-Multi-PIE (Colbois et al., 2021)	10'000	180'000	78.72	60.22	61.83	60.84	54.05
IDnet (Kolf et al., 2023)	10'577	1'057'200	84.48	68.12	71.42	68.93	62.63
ExFaceGAN (Boutros et al., 2023b)	10'000	599'944	85.98	66.97	70.00	66.96	57.37
GANDiffFace (Melzi et al., 2023)	10'080	543'893	94.35	76.15	79.90	78.99	69.82
Langevin-Dispersion (Geissbühler et al., 2024)	10'000	650'000	94.38	65.75	86.03	65.51	77.30
Langevin-DisCo (Geissbühler et al., 2024)	10'000	650'000	97.07	76.73	89.05	79.56	83.38
DigiFace-1M (Bae et al., 2023)	109'999	1'219'995	90.68	72.55	73.75	79.43	68.43
IDiff-Face (Uniform) (Boutros et al., 2023a)	10'049	502'450	98.18	80.87	90.82	82.96	85.50
IDiff-Face (Two-Stage) (Boutros et al., 2023a)	10'050	502'500	98.00	77.77	88.55	82.57	82.35
DCFace (Kim et al., 2023)	10'000	500'000	98.35	83.12	**91.70**	88.43	**89.50**
HyperFace [ours]	10'000	500'000	**98.50**	**84.23**	89.40	**88.83**	86.53
CASIA-WebFace (Yi et al., 2014)	10'572	490'623	99.42	90.02	93.43	94.97	94.32

Scaling Dataset Generation

- **Complexity:** The **HyperFace optimization** (Algorithm 1) considers **all pairs of reference points** in the hypersphere and maximizes their distances. Therefore, the optimization considers all pairs of points and has **quadratic complexity !**

n_{id}	HyperFace Optimization Runtime
10k	6 hours
20k	11 hours
30k	23 hours
50k	84 hours

Scaling Dataset Generation

- **Complexity:** The **HyperFace optimization** (Algorithm 1) considers **all pairs of reference points** in the hypersphere and maximizes their distances. Therefore, the optimization considers all pairs of points and has **quadratic complexity !**

n_{id}	HyperFace Optimization Runtime
10k	6 hours
20k	11 hours
30k	23 hours
50k	84 hours

- **To reduce complexity,** we propose **HyperFace Stochastic Optimization**, where in each iteration we randomly select a mini-batch of *b* points. This will lead to **constant complexity wrt *number of identities*** and **quadratic complexity wrt mini-batch size.**

Scaling Dataset Generation

- **Complexity:** The **HyperFace optimization** (Algorithm 1) considers **all pairs of reference points** in the hypersphere and maximizes their distances. Therefore, the optimization considers all pairs of points and has **quadratic complexity !**

n_{id}	HyperFace Optimization Runtime
10k	6 hours
20k	11 hours
30k	23 hours
50k	84 hours

- **To reduce complexity,** we propose **HyperFace Stochastic Optimization**, where in each iteration we randomly select a mini-batch of *b* points. This will lead to **constant complexity wrt *number of identities*** and **quadratic complexity wrt mini-batch size.**

Batch Size (b)	# ID (n_{id})	HyperFace Stochastic Optimization Runtime
1,000	30k	0.4 hours
	50k	0.5 hours
	100k	0.5 hours
5,000	30k	2.2 hours
	50k	2.2 hours
	100k	2.2 hours

HyperFace Stochastic Optimization

Algorithm 2 HyperFace Stochastic Optimization for Finding Reference Embeddings

Inputs: λ : learning rate, n_{itr} : number of iterations, $\{\boldsymbol{x}_g\}_{g=1}^{n_{\text{gallery}}}$: embeddings of a gallery of face images,
α : hyperparameter (contribution of regularization), b : size of mini-batch.
Output: $\boldsymbol{X}_{\text{ref}} = \{\boldsymbol{x}_{\text{ref},i}\}_{i=1}^{n_{\text{id}}}$: optimized reference embeddings.
Procedure:
  Initialize reference embeddings $\boldsymbol{X}_{\text{ref}} = \{\boldsymbol{x}_{\text{ref},i}\}_{i=1}^{n_{\text{id}}}$
  For $n = 1, .., n_{\text{itr}}$ **do**
    Sample a random mini-batch $B \subset \boldsymbol{X}_{\text{ref}}$ of size b ▷ Sampling a random mini-batch
    Find $\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j} \in B$ which have minimum distance $d(\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j})$
    Reg $\leftarrow \frac{1}{b} \sum_{k=1}^{b} \min_{\{\boldsymbol{x}_g\}_{\text{gallery}}} d(\boldsymbol{x}_{\text{ref},k}, \boldsymbol{x}_g)$ ▷ Calculate the regularization term
    cost $\leftarrow -d(\boldsymbol{x}_{\text{ref},i}, \boldsymbol{x}_{\text{ref},j})$
    $B \leftarrow B -$ Adam(∇cost, λ)
    $B \leftarrow$ normalize(B) ▷ To ensure that resulting embeddings B remain on the hypersphere.
    Update B in $\boldsymbol{X}_{\text{ref}}$
  End For
End Procedure

HyperFace Stochastic Optimization

- **Ablation study** on the effect of number of batch size in HyperFace stochastic optimization (Algorithm 2)

Batch Size	LFW	CPLFW	CALFW	CFP	AgeDB
1,000 (mini-batch)	98.28	85.23	91.05	91.86	89.37
5,000 (mini-batch)	98.62	84.98	90.73	90.41	88.97
30,000 (full-batch)	98.38	85.07	90.88	91.57	89.60

Thanks for your attention!

[Paper]

[Project Page]