BTech Project Report
Few-Shots Class-Incremental Learning for Face Recognition

by

Aryan Tomar

2003107

Faculty Adviser

Dr. Shitala Prasad

School of Mathematics and Computer Science

Indian Institute of Technology Goa

Contents

Chapter 1: Project Overview

Our aim was to build a Few-Shots Class-Incremental-Learning Face Recognition model. We experimented different approaches suitable for the method and provided theoretical reasoning for all our claims while formulating the adequate model suitable for the task. The training and testing strategy introduced in the work are our novelty. We used a combination of existing loss functions in a novel way and formulated an overall loss function which is also our contribution.

Chapter 2: Theoretical Review

2.1 Face Recognition

Face Recognition (FR) have recently emerged as prominent applications in artificial intelligence and computer vision domains. Particularly deep Convolutional Neural Networks have made significant advances in FR. However, all these methods presume that the training data is available to train all-at-once during the training stage, which is not the case with the real-world data that comes in continuously in incremental phases. This makes the conventional deep CNNs to forget previously learned knowledge catastrophically (1). In order to address this issue, many incremental FR techniques have been proposed which can be broadly categorized into two categories (2).

LDA Based Approach

Linear Discriminant Analysis (LDA) based incremental FR methods often involve updating scatter matrices to learn new faces through methods such as Eigendecomposition, Singular Value Decomposition (SVD), and QR decomposition. These methods adapt to new face data incrementally, but often face computational complexity challenges due to complex matrix calculations and cannot handle variations like lighting, pose, expression, or occlusion (2). All these methods are based only on incremental LDA implementation. (3,4) leverages deep neural network with discriminant analysis for incremental FR. However, (3) mentions that Deep-SRDA faces the problem of longer computation time due to its quadratic complexity and (4) assumes the training setting where numerous amounts of images will be provided for training each class in the incremental setting as pointed out in (4) in which 1000 classes with average of 494 images per person were trained in incremental fashion.

SVM based approach

Support Vector Machines (SVM) based incremental FR uses lagrangian optimization which seeks stationary points to minimize dimensionality reduction errors and update the model within KKT conditions (5). While (6) uses discrete cosine transformations to convert facial images from spatial domain to frequency domain. This reduces the space complexity. But both (5) and (6), suffers from handling face variations same as LDA based approaches. (7) leverages SVM by removing the classification layer of DNNs with linear SVMs. But (7) fails to perform well in large number of incremental states (2).

2.2 Few-Shots Learning

Few-Shots Learning (FSL) aims to mimic the learning process of human i.e. learning a new task with extremely small number of samples (8,9). When utilizing supervised learning paradigm and prior knowledge on tiny datasets, FSL may effectively generalize to new tasks and samples. Few-Shot learning typically employs Siamese networks (SNN) for learning embeddings (10). (10) use two subnetworks to extract features from two input samples and calculate the distance between these features. Matching Nets (11) is a technique that utilizes neural networks with attention and sequence models to enable memory retention and facilitate learning of embeddings and similarity computation. Generally, a Few-Shot face recognition system employs two stages to recognize faces. The first stage entails extracting features from a given face image, while the second stage involves estimating the face output or identity for that particular face based on the extracted features (12).

2.3 Continual Learning

Continual Learning (CL) works in a standard setup, which involves learning a sequence of topics one at a time and treating them as if they were witnessed simultaneously. The methods in CL can be divided into three major streams: replay-based methods (13,14), regularization-based methods (15) and architecture-based methods (16). Replay-based methods uses the memory from past data, which is either stored (13) or generated (14), to mitigate catastrophic forgetting. However, they impose strict requirements on the usage of memory, for both stored and generated. In order to balance the old and new jobs, regularization-based approaches are characterized by the addition of explicit regularization terms, which involves storing a frozen copy of the previous model for reference. Architecture-based approach focuses on the way of implementing task-specific parameters, extending the above concepts to parameter allocation, model decomposition and modular network (17).

Chapter 3: Problem Formulation

Incremental FR aims to learn L number of classes over T incremental steps. Considering constant step size, at current step t, model will learn only on classes c_{1t}, c_{2t}, ...c_{Nt}, where N = L//T is the number of classes per incremental step, without accessing old classes c_{10}, ...c_{Lt-1}. To make our model more robust, we introduce larger domain gap in the training data by training the network over a stream of datasets D = {D_1, ...D_Q} in incremental fashion. So at time q and step t, only the classes c_{q1t}, c_{q2t}, ...c_{qLt}, from the dataset D_q, will be trained on the network. The total number of continual steps for which the model will be trained will be $Q \cdot T$ and at each continual step the model cannot access the past data. Due to the scarcity of data and making the training scenario suitable for real-world situation, each class will have only k number of training samples where $k \leq 5$ (configured before the training). At the test time, the model is tested on past seen classes for its non-forgetting property and new unseen data to evaluate its generalization ability.

Chapter 4: Methodology

We propose a patch-based knowledge distillation framework for lifelong FR. It builds on an image-based base model (Section 4.1) and (Section 4.2) incorporates three new modules: a differentiable patch sampler for selecting a diverse set of patches that tend to be invariably important over streaming data and are used for piloting the distillation on the new data, patch logit distillation that encourages the current model to mimic the old one's prediction on the selected patches, and patch relation distillation that helps the model to retain various type of patch-level relational knowledge.

4.1 Base Model

We begin by introducing a base model that only uses image-level feature to address the lifelong FR problem. The base model is a mapping from input images to logits (class scores over person identities). It can be broken down into three stages as $f = \varphi \text{ o } g \text{ o } \Psi$. First, the backbone network Ψ (·) extracts a feature map from the input image, followed by the global average pooling $g(\cdot)$ that applies average pooling over the spatial dimension to generate a face recognition feature from the feature map. Finally, the classifer head φ (·) predicts logits from the face-recogniton feature, which can be converted to probabilities with an activation function σ such as softmax. Furthermore, let a superscript o be the mark for something related to the old model. For example, $f^o = \varphi^o \text{ o } g^o \text{ o } \Psi^o$ denotes the model from the last time step, which is frozen during training.

The primary goal of the base model is to extract discriminative face-oriented features based on the current identity labels. Following the common practice of conventional FR method, we introduce a cross-entropy loss that matches predicted probabilities to class labels and a triplet loss that encourages intra-class compactness and inter-class separability. Given a mini-batch of samples $\{(x_i,y_i)\}^B_{i=1}$ from the current dataset, where B is the mini-batch size, the base-model loss is defined as

$$
\begin{aligned}
&= \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}_{CE}\left(y_i, \sigma(f(x_i))\right) \\
&\quad + \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}_{tri}\left(g(\psi(x_i)), g(\psi(x_i^p)), g(\psi(x_i^n))\right)
\end{aligned}
$$

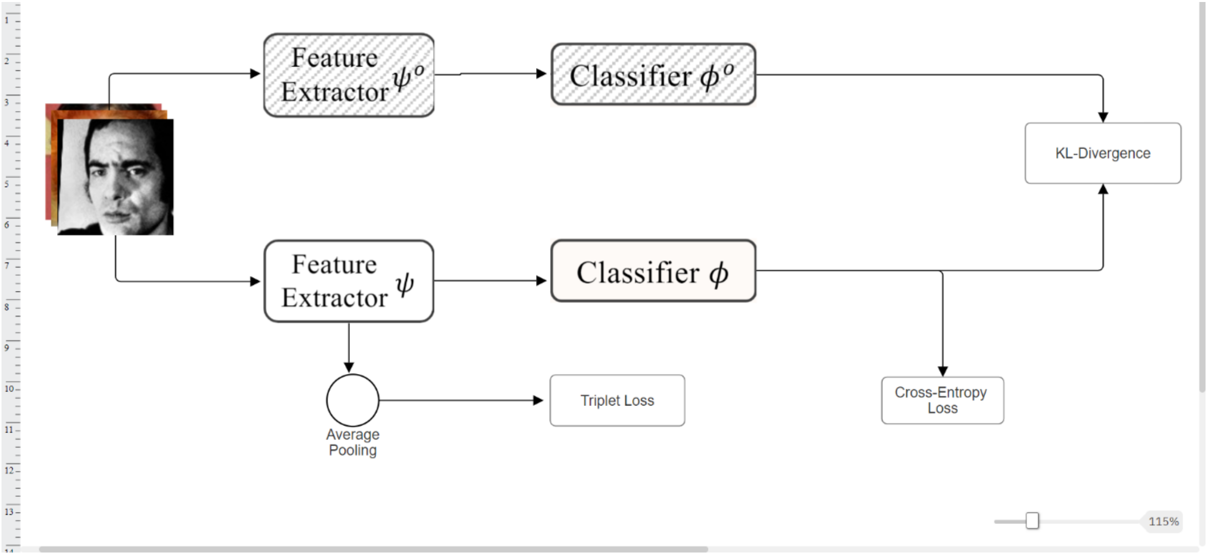
where x^p and x^n are the positive and negative samples for x_i respectively. All samples are from D only unless otherwise notified.

However, optimizing the model only by CE and triplet loss may lead to catastrophic forgetting on previously seen datasets. To preserve learned knowledge, we enforce consistency between the current model and the old model using a distillation loss that minimizes the Kullback-Leibler divergence between the logits of the two models:

$$\mathcal{L}_{KD} = \frac{1}{B}\sum_{i=1}^{B} \mathrm{KL}\left(\sigma(f^{o}(x_i)/\tau) \,\|\, \sigma(f(x_i)/\tau)\right),$$

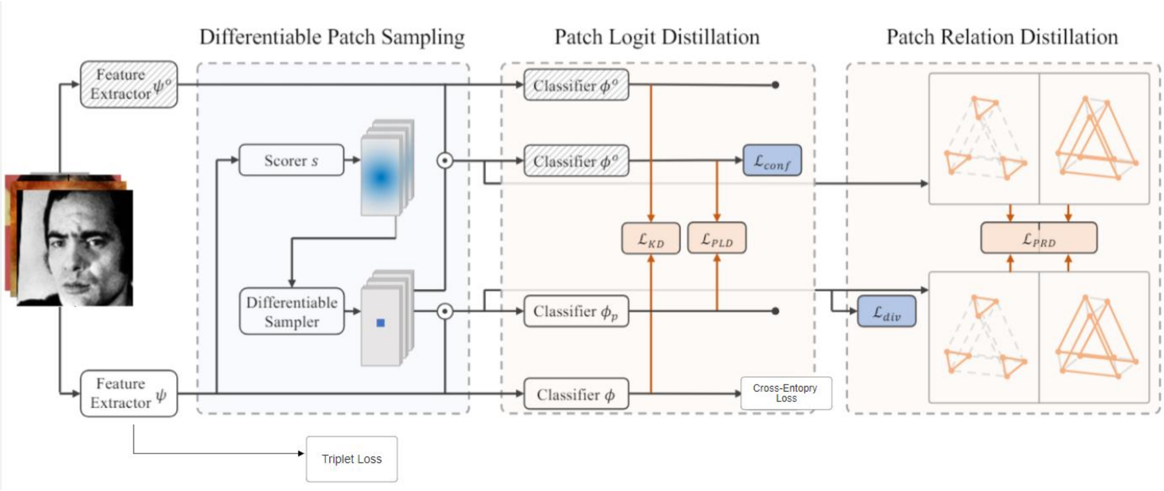
where g is a hyperparameter referred to as the temperature by Hinton et al. [21]. Then, the loss function for the base model is

$$\mathcal{L}_{base} = \mathcal{L}_{CE} + \mathcal{L}_{triplet} + \mathcal{L}_{KD}$$



In our previous approach, the efficacy of our baseline model, which leverages knowledge distillation through Learning without Forgetting (LwF), was hindered by notable distribution shifts within our training data. These shifts, stemming from the utilization of different datasets, resulted in a gradual accumulation of errors when applied to previous datasets, ultimately leading to subpar non-forgetting performance. To address this challenge, we adopted a Patch-Based Feature Learning strategy. By operating at the patch level rather than the image level, we significantly narrowed the distribution gap. This shift allowed us to mitigate the impact of distribution shifts on knowledge distillation, thereby enhancing the robustness and performance of our model.

4.2 Patch-Based Modelling



Differentiable Patch Sampler

In this section, we first explain the underlying mechanism of the proposed differentiable patch sampler in a simplified scenario, i.e., sampling a single patch, and then introduce three key designs to guide the patch sampler to select a diverse set of patches that are less affected by the distribution shift.

Sampling a single patch. For the sake of computational efficiency, we sample the patch from a relatively small feature map instead of the original image. Specifically, given an input images x, we use the last feature maps $\Psi(x)$ $\Psi^o(x)$ and treat them both as a set of candidate features. To model the sampling probability of each candidate feature, a learnable scorer is employed to predict a score vector s_i, where each value is subsequently converted to the sampling probability of the j-th candidate feature $\Psi\ (x_i)_j$ (j indexes all possible neurons or image patches) using an activation function σ. Hence, the distribution of the sampled patch feature p_i is modeled as

$$P\left(p_i = \psi(x_i)_j\right) = \sigma(s_i)_j.$$

Directly choosing a maximum from the above distribution is non-differentiable. We adopt a differentiable alternative by using the Gumbel-max trick to draw a discrete sample M_i

$$M_i = \text{one\_hot}\left(\arg\max_j (s_i + G)_j\right),$$

where G is a vector of i.i.d. Gumbel noise samples, and employing a straight-through estimator to enable differentiation of arg max in the backward pass. Here, M_i determines the location of the sampled patch, so it is used as a shared mask to extract patch features of the two models from feature maps Ψ(x) $\Psi^o(x)$:

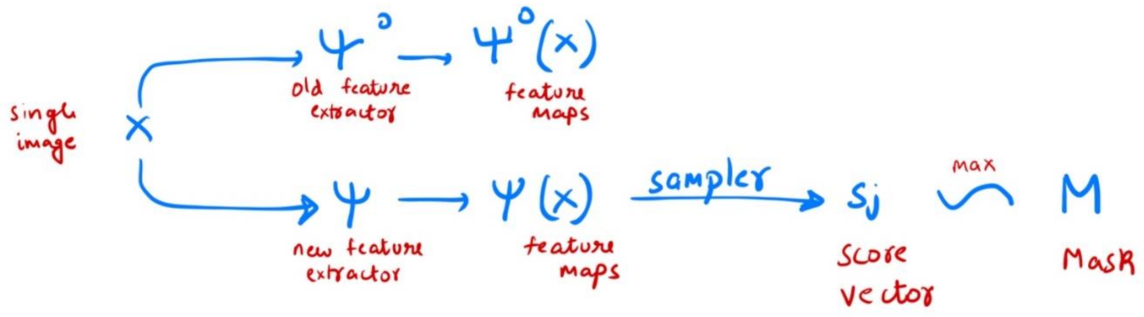
$$p_i = M_i \odot \psi(x_i), \quad p_i^o = M_i \odot \psi^o(x_i),$$

where dot in a circle denotes element-wise product. In this way, the patch sampling process becomes differentiable.

Sampling with multiple branches. A single patch is often insufficient. To draw K patches from an image, a popular solution is to sample K times from the same distribution without replacement. However, this may produce mutually similar patches. To capture diverse patterns in the feature map, we propose a patch sampler with branches. Each branch is designed to separately learn a different distribution and sample a patch from it. Let $(p_{i,r}, p^o_{i,r})$ denote the patch features sampled by the r-th branch, then our patch sampler samples K patches for an image:

$$\{(p_{i,1}, p^o_{i,1}), \ldots, (p_{i,K}, p^o_{i,K})\}.$$

This multi-branch design allows more diversity among sampled patches.



$$\Psi^o(x) \xrightarrow{M} p^o(x)$$

old patch features

$$\Psi(x) \xrightarrow{M} p(x)$$

new patch features

Repeating 3 times will give you 3 patch features

$$\{p_1(x), p_1^o(x)\} \ \{p_2(x), p_2^o(x)\} \ \{p_3(x), p_3^o(x)\}$$

Confidence loss. In order to guide the differentiable patch sampler to minimize patch-level distribution gap, the loss function has to penalize patches that are far from previous distributions. This may be seen as out-of-distribution detection at patch-level, and thus can be solved similarly. Inspired by a group of out-of-distribution detection methods that interpret inputs with low prediction confidence as out-of-distribution examples, we estimate each patch’s closeness to previous distributions with its confidence on the old model. Given a patch with features ($p_{i,r}$,$p^{o}_{i,r}$), its con!- dence on the old model can be measured by the negative entropy of the model prediction $\sigma(\varphi^{o}(p^{o}_{i,r}))$. Based on this, we define the confidence loss in the form of entropy:

$$\mathcal{L}_{conf} = \frac{1}{BK} \sum_{i=1}^{B} \sum_{i=1}^{K} \mathrm{H}\left(\sigma(\phi^{o}(p^{o}_{i,r}))\right).$$

It is easy to see that minimizing L_{conf} is equivalent to maximizing the confidence of sampled patches on the old model, thereby training the sampler to select patches that are closer to previous distributions.

Diversity loss. To avoid the scores in branches from converging to the same or similar values, resulting in poor patch diversity, we introduce a diversity loss to penalize mutually similar patch pairs, such as those with high cosine similarity

$$\mathcal{L}_{div} = \frac{1}{BK^{2}} \sum_{i=1}^{B} \sum_{r,s=1}^{K} \frac{\langle p_{i,r}, p_{i,s} \rangle}{||p_{i,r}||\,||p_{i,s}||},$$

where <.,.> is the inner product and || || is the L2 norm. However, the diversity loss may distract the patch sampler away from its original objective and define the total loss function for the patch sampler as

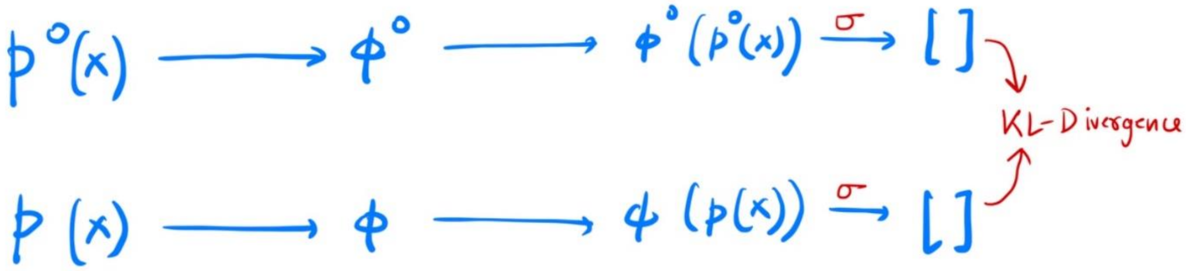
$$\mathcal{L}_{sel} = \mathcal{L}_{conf} + \mathcal{L}_{div}.$$

Patch Logit Distillation

Patch logit distillation intends to utilize selected patch features for knowledge distillation. However, directly passing patch-level information would disturb image-level feature learning due to the distribution discrepancy between images and patches. For example, patch features can interfere with the batch statics in the classifier φ and then further affect the training of the whole model. Instead of using a shared classifier φ, we employ a separate patch classifier φ_p to predict logits from patch features and distill knowledge from the old model. The distillation loss for all patches within a mini-batch is calculated as

$$\mathcal{L}_{PLD} = \frac{1}{BK}\sum_{i=1}^{B}\sum_{r=1}^{K}\mathrm{KL}\left(\sigma(\phi^{o}(p^{o}_{i,r})/\tau) \,\|\, \sigma(\phi_{p}(p_{i,r})/\tau)\right).$$

Since the patches are selected to be closer to the previous distributions, patch logit distillation is less affected by the distribution shift and thus more effective. When performing together with image-based distillation, it also helps retain more detailed knowledge about local cues.



Patch Relation Distillation

Relation distillation is also utilized to preserve high-order knowledge beyond class scores. The intuition is that knowledge can be complementarily represented by feature relations beyond individual features [42], which is in line with the task goal of FR (namely matching images for the same identity). While its power in lifelong learning has only recently been explored, we take a step further by distilling on patch relations.

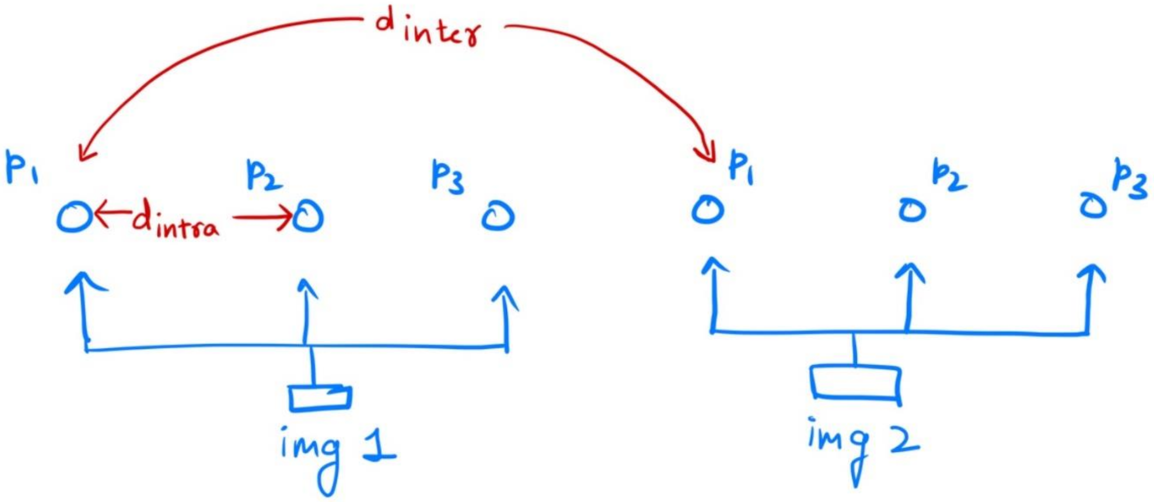
Among BK sampled patches within a mini-batch, there are a small proportion of intra-instance relations that correspond to local correlations within the image, and numerous inter-instance relations containing sparse yet valuable global identity information. Since intra-instance relations are significantly outnumbered, we propose to handle them separately. Suppose the patch relation is represented by pairwise feature distance d (·, ·), we consider the following two sets of distances, i.e., all intra-instance distances and a fraction of inter-instance distances from the same sampler branch:

$$S_{intra} = \bigcup_{i=1}^{B} \left\{ (d(p_{i,r}, p_{i,s}), d(p^o_{i,r}, p^o_{i,s})) \mid r, s \in [1..K], r \neq s \right\},$$

$$S_{inter} = \bigcup_{r=1}^{K} \left\{ (d(p_{i,r}, p_{j,r}), d(p^o_{i,r}, p^o_{j,r})) \mid i, j \in [1..B], i \neq j \right\}.$$

It is desirable for the current model to be consistent with the old one in these distances, so we adopt a Huber loss l_δ to penalize the difference between each distance generated by the two models:

$$\mathcal{L}_{PRD} = \frac{\sum l_\delta(d_{intra} - d^o_{intra})}{|S_{intra}|} + \frac{\sum l_\delta(d_{inter} - d^o_{inter})}{|S_{inter}|},$$



Since the intra-instance and inter-instance terms contribute equally to the distillation loss, both intra-instance and inter-instance relational knowledge are preserved during training. The final loss function for our framework is

$$\mathcal{L} = \mathcal{L}_{base} + \mathcal{L}_{sel} + \mathcal{L}_{PLD} + \mathcal{L}_{PRD},$$

Remarks. The patch sampler, the patch classifier and the patch relation distillation module are employed only to regularize the training of the model and do not participate in the model inference during testing. Therefore, the extra computational overhead brought by our framework stays in the training stage and does not affect testing.

4.3 Drawback of Patch Logit Distillation

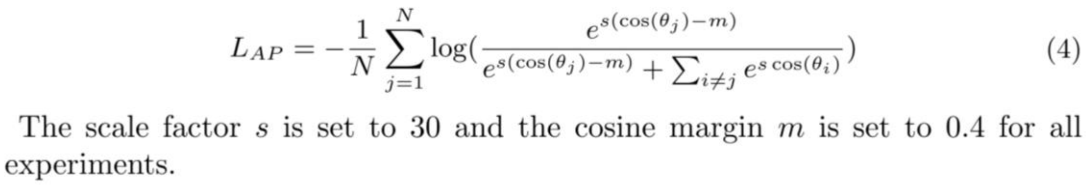
PLD typically involves directly applying knowledge distillation techniques at the level of individual patches. It often uses loss functions such as Kullback-Leibler divergence or mean squared error between the teacher model's outputs (trained on full images or more comprehensive data) and the student model's outputs (trained on patches). PLD typically involves directly applying knowledge distillation techniques at the level of individual patches. It often uses loss functions such as Kullback-Leibler divergence or mean squared error between the teacher model's outputs (trained on full images or more comprehensive data) and the student model's outputs (trained on patches).

If the patch does not contain informative features generalizable across various samples (e.g., patches of background or non-distinctive facial regions), the student model may still try to mimic the teacher's output for that patch, potentially learning to replicate non-generalizable, specific features. This will cause a gradual error build-up.

4.4 Angular Penalty Loss

Angular penalty loss provides more compact intra-class clustering and wider inter-class separation than cross-entropy loss. Compact clustering leaves more room on the latent feature space to accommodate the new classes. We want to obtain a feature extractor which can rapidly adapt to continually coming new tasks, as well as be stable to overcome catastrophic forgetting for the previously learned tasks. Thus, we want to use a loss function that: 1) minimizes the distance between intra-class feature vectors, and 2) maximizes the distance between inter-class feature vectors. The compact intraclass clustering and wide inter-class separation will leave more room in the latent feature space for the incrementally arriving new classes and hence lead to better classification.

First, we use cosine similarity as the distance metric to measure data similarity and compute scores. It has two effects: 1) it makes training focus on the angles between normalized features instead of absolute distance in the latent feature space, and 2) the normalized weight parameters of the fully connected layer can be regarded as the center of each category



$$L_{AP} = -\frac{1}{N}\sum_{j=1}^{N}\log(\frac{e^{s(\cos(\theta_j)-m)}}{e^{s(\cos(\theta_j)-m)}+\sum_{i\neq j}e^{s\cos(\theta_i)}}) \tag{4}$$

The scale factor s is set to 30 and the cosine margin m is set to 0.4 for all experiments.

Chapter 5: Dataset, Training and metric details

5.1 Datasets and Evaluation metrics

Datasets

The UMDFace dataset, comprising 367,888 face annotations across 8,277 subjects, has been maintained by the University of Maryland since its release in 2016. It presents challenges such as pose and expression variation evident in each image column. ArcFace (MS1Mv2), with 5.8 million images representing 85,000 identities, stands out for its refined quality achieved through semi-automatic processes and ethnicity-specific annotators. Conversely, VGGFace2, managed by the Visual Geometry Group at Oxford, offers 3.31 million images divided into 9,131 classes, presenting notable challenges due to significant variations within classes, particularly in terms of age and pose. RetinaFace, a semi-automatic refinement of MS-Celeb-1M, provides 5.1 million images across 93,000 identities, offering further refinement through ethnicity-specific annotation and facial landmark detection. Finally, CasiaFace, featuring 494,414 face images of 10,575 identities, stands out for its unprocessed nature, derived from web-scraping and thus containing challenging elements like low resolution and occlusion.

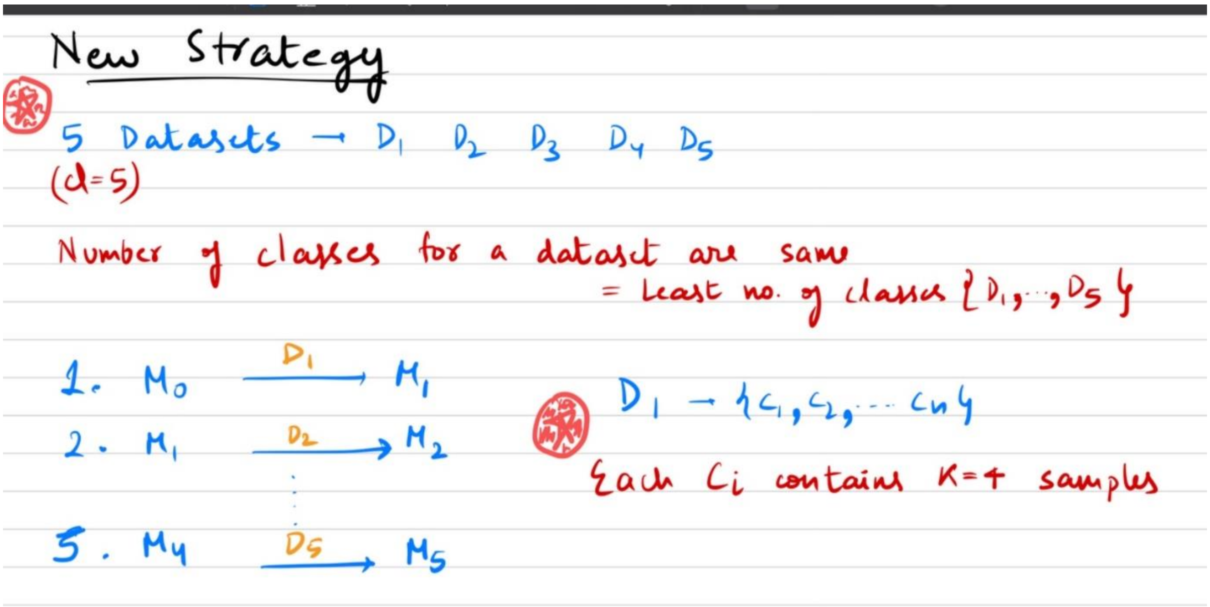
Evaluation metric

For a class, we take K=4 samples for training, one sample as query and rest of the samples in the gallery data. We then repeat this process for all the classes in the dataset. This gives a big gallery dataset to look for the image while testing, we then use top-1 accuracy to detect our test image in from the gallery dataset. We calculate the distance of a query image with all of the images in the gallery and report the top detection as 0 if not of same class and 1 if of same class.

5.2 Implementation Details

We utilize a ResNet-50 pretrained on ImageNet as the feature extractor. Note that the last stride is set to 1. The model is trained for 50 epochs with 150 iterations per epoch using an Adam optimizer. The learning rate is set to 3.5 x 10^{-4} initially and decays by x0.1 at 25th and 35th epochs. The batch size is set to B = 128. In specific, each batch is composed of 32 identities and 4 images per identity. The input images are resized to 256 x 128 with data augmentations including random cropping, horizontal flipping and erasing. For the patch sampling step, patch features are sampled from the last feature map of size 16 x 8. The scorer for predicting sampling probabilities is a two-layer perceptron with 4096 hidden units. We Hinton and set the temperature $\tau = 2$. The whole architecture is implemented with PyTorch. During evaluation, the retrieval result is computed based on the Euclidean distance of image-level features.

5.3 Train-strategy



New Strategy

5 Datasets → D_1 D_2 D_3 D_4 D_5
$(d=5)$

Number of classes for a dataset are same
= Least no. of classes $\{D_1, \cdots, D_5\}$

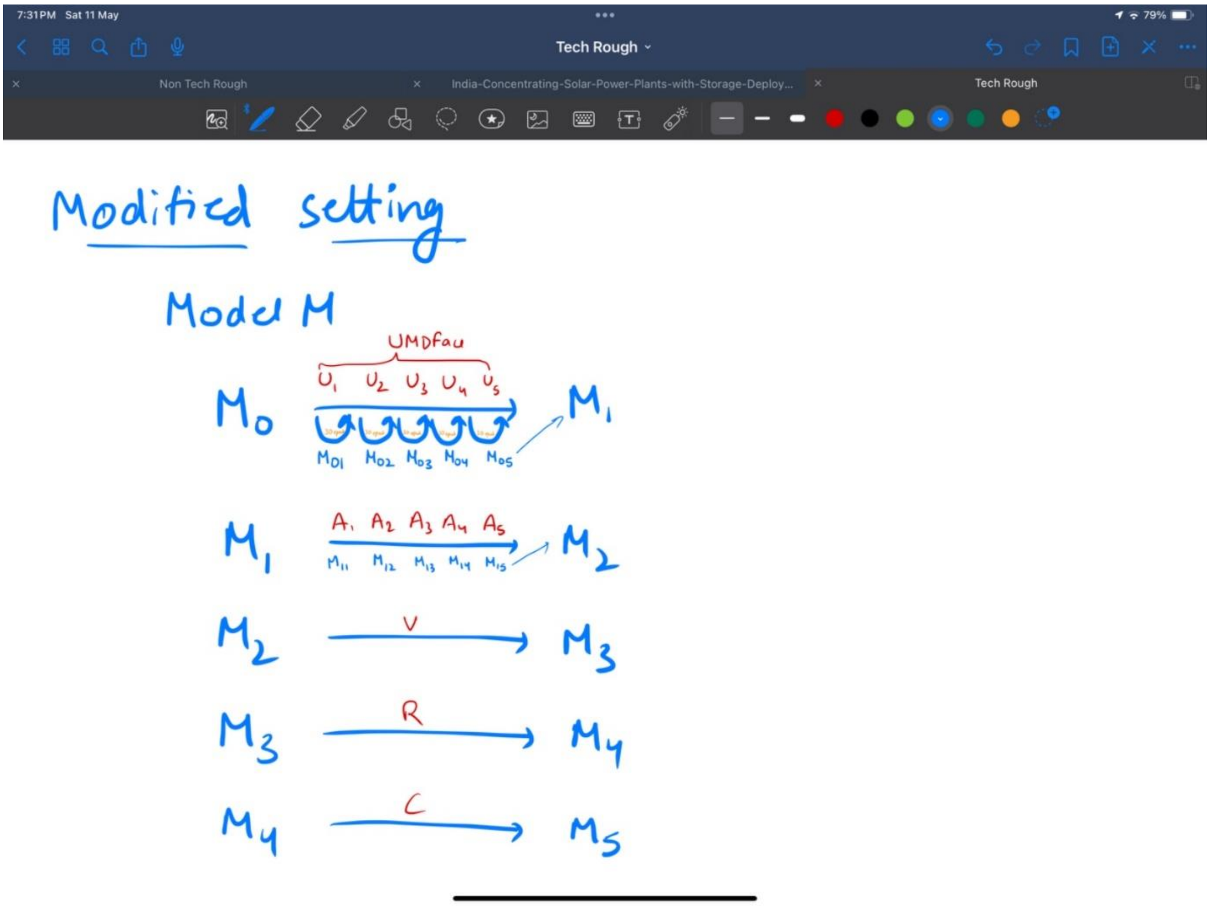
1. $M_0 \xrightarrow{D_1} M_1$
2. $M_1 \xrightarrow{D_2} M_2$

⋮

5. $M_4 \xrightarrow{D_5} M_5$

$D_1 \rightarrow \{c_1, c_2, \cdots c_n\}$

Each C_i contains $K=4$ samples


7:31PM Sat 11 May
79%
Tech Rough
Non Tech Rough
India-Concentrating-Solar-Power-Plants-with-Storage-Deploy...
Tech Rough

Modified setting

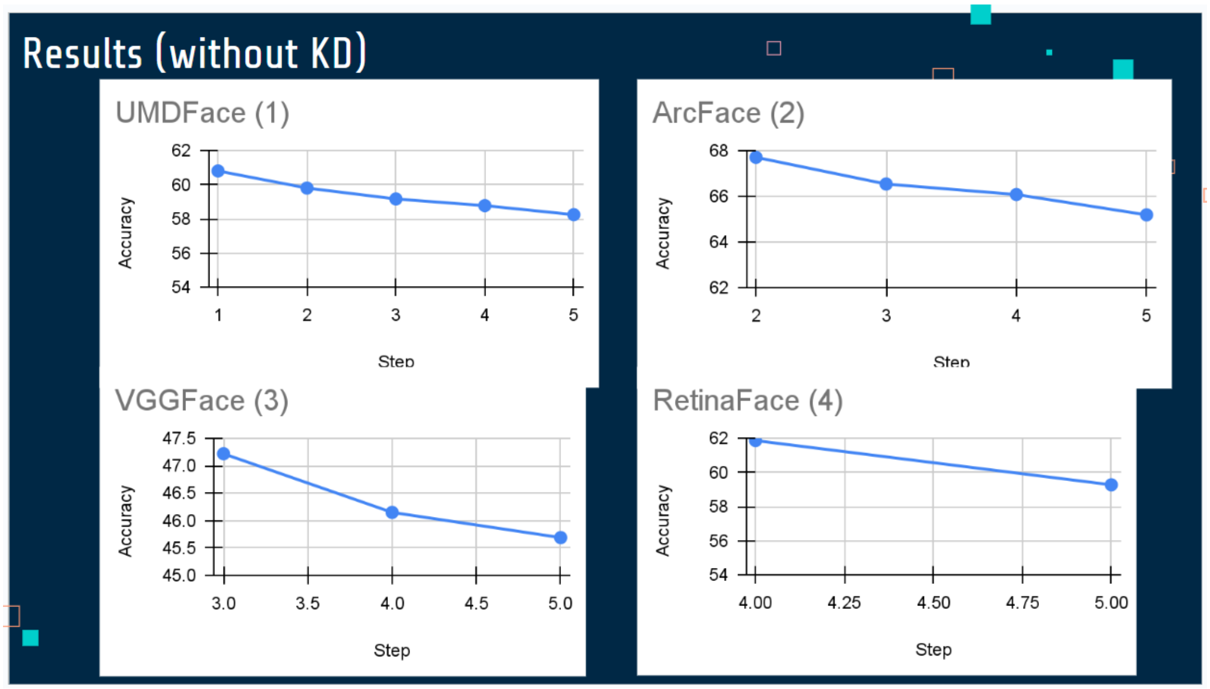
Model M

UMDfau
U_1 U_2 U_3 U_4 U_5
M_0
M_1
M_{01} M_{02} M_{03} M_{04} M_{05}
M_1
A_1 A_2 A_3 A_4 A_5
M_2
M_{11} M_{12} M_{13} M_{14} M_{15}
M_2
V
M_3
M_3
R
M_4
M_4
C
M_5

5.4 Results

Without KD, train-strategy 1

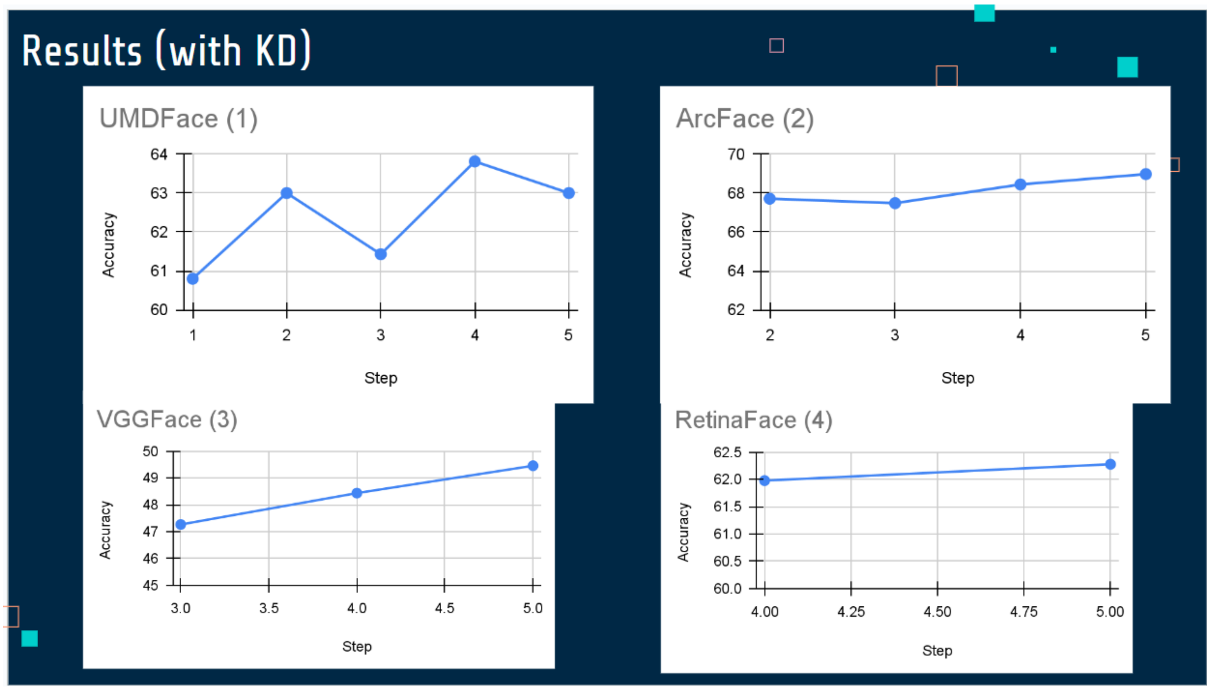
	UMDFace	ArcFace	VGGFace	RetinaFace	CasiaFace
UMDFace	**60.8**				
ArcFace	**59.8**	**67.71**			
VGGFace	**59.16**	**66.54**	**47.22**		
RetinaFace	**58.77**	**66.08**	**46.15**	**61.86**	
CasiaFace	**58.24**	**65.19**	**45.69**	**59.27**	**42.26**



Results (without KD)

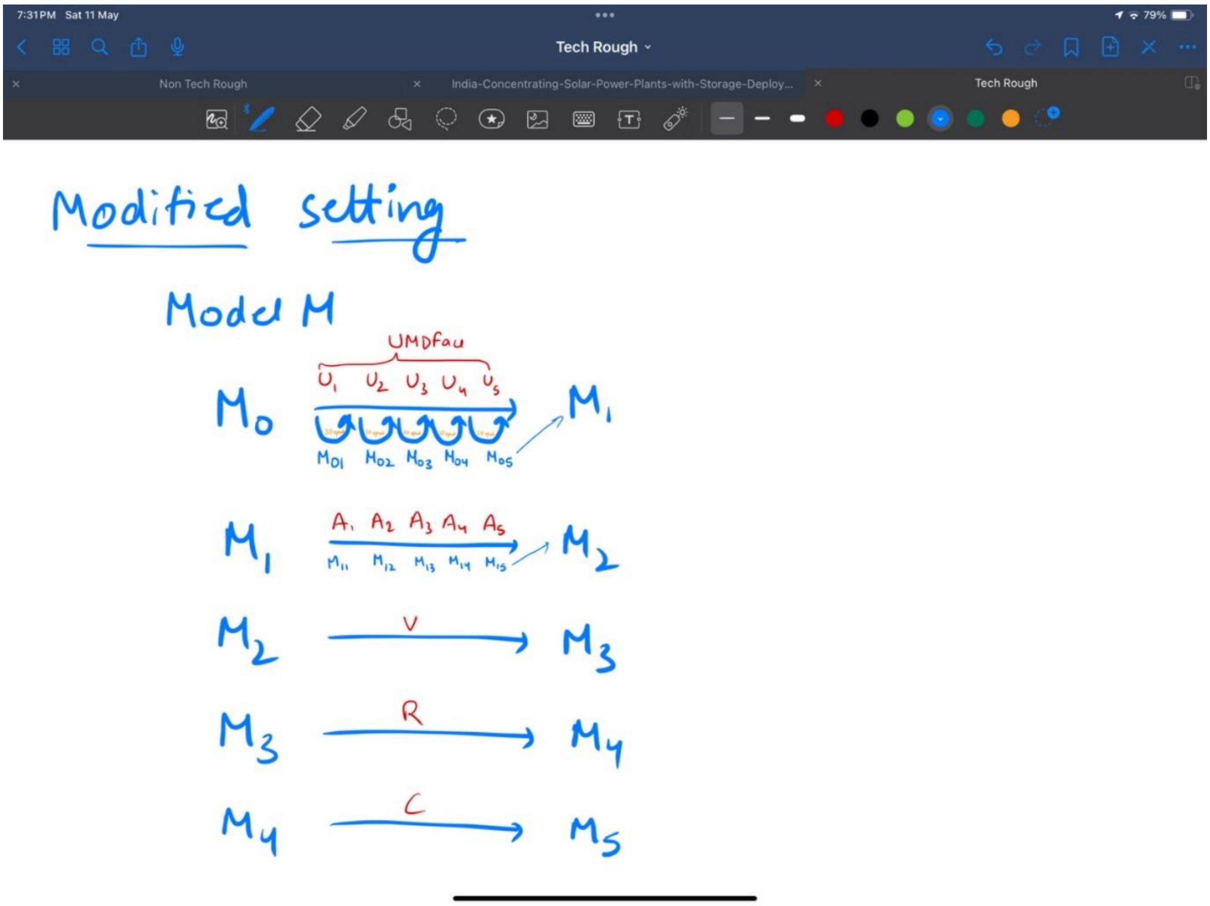
With KD, train-strategy 1

	UMDFace	ArcFace	VGGFace	RetinaFace	CasiaFace
UMDFace	**60.8**				
ArcFace	**63**	**67.71**			
VGGFace	**61.43**	**67.48**	**47.27**		
RetinaFace	**63.81**	**68.44**	**48.44**	**61.98**	
CasiaFace	**63**	**68.97**	**49.46**	**62.28**	**42.2**



Results (with KD)

Training Strategy Modification (Novelty)



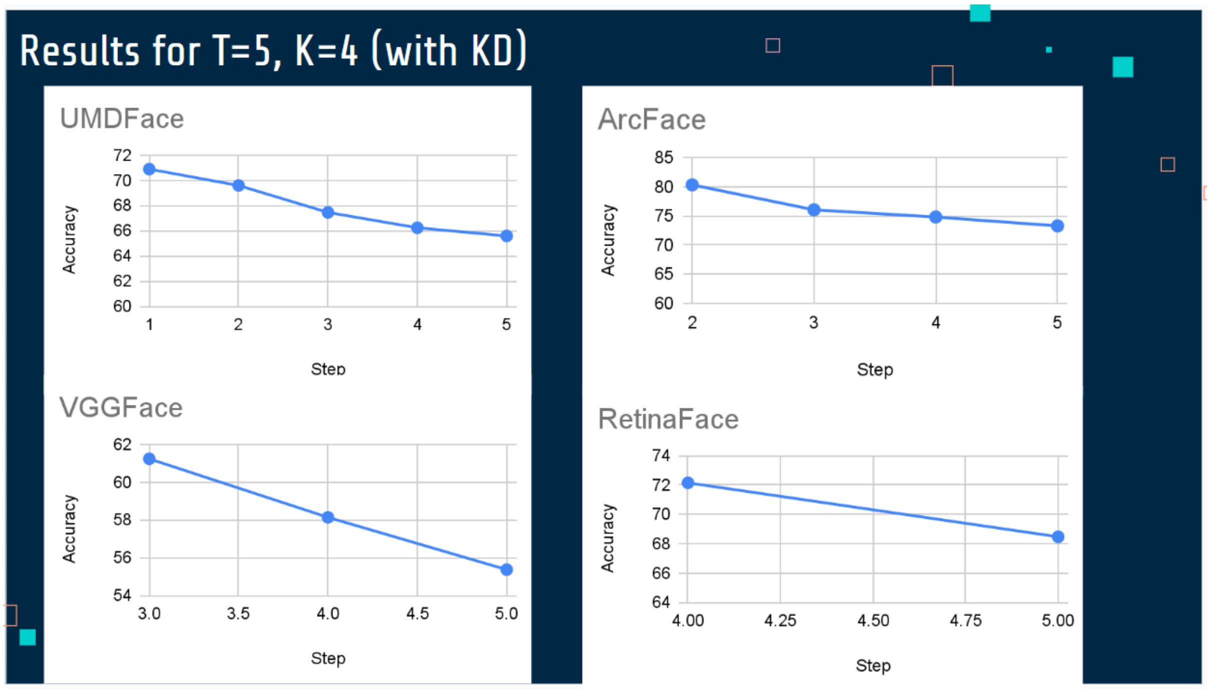
7:31 PM Sat 11 May
79%
Tech Rough
Non Tech Rough
India-Concentrating-Solar-Power-Plants-with-Storage-Deploy...
Tech Rough

Modified setting

Model M

	UMDFace	ArcFace	VGGFace	RetinaFace	CasiaFace

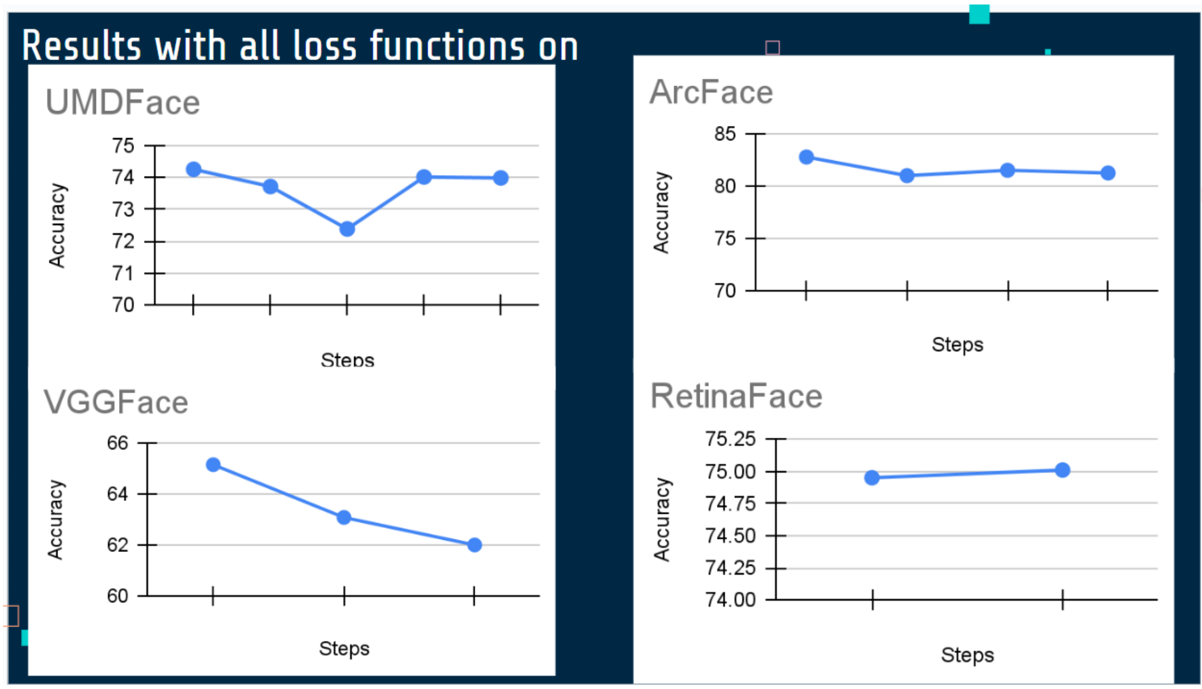
UMDFace	70.89				
ArcFace	69.59	80.3			
VGGFace	67.45	76.03	61.23		
RetinaFace	66.23	74.79	58.13	72.14	
CasiaFace	65.58	73.27	55.36	68.47	58.72



Results for T=5, K=4 (with KD)

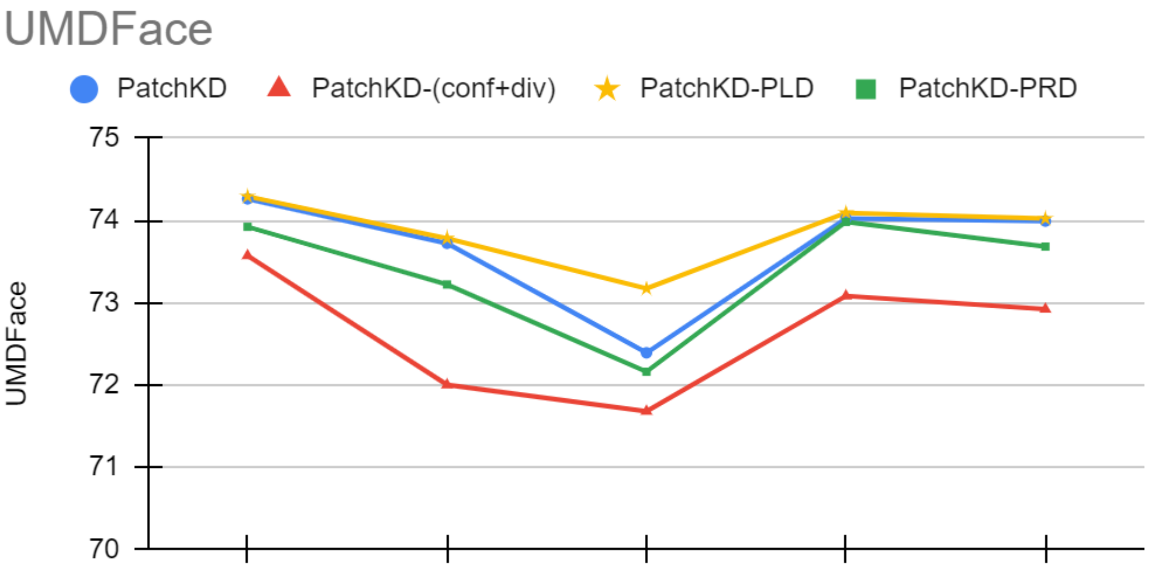
	UMDFace	ArcFace	VGGFace	RetinaFace	CasiaFace
UMDFace	74.26				
ArcFace	73.72	82.81			

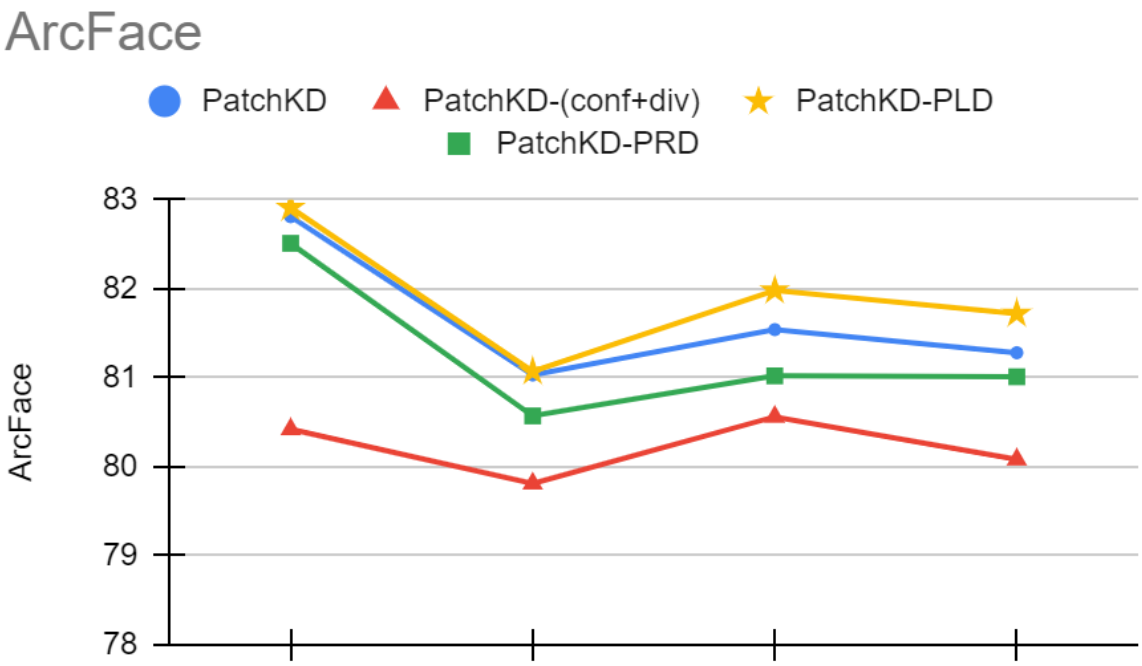
VGGFace	72.39	81.03	65.16		
RetinaFace	74.02	81.54	63.09	74.95	
CasiaFace	73.99	81.28	62.01	75.01	55.8

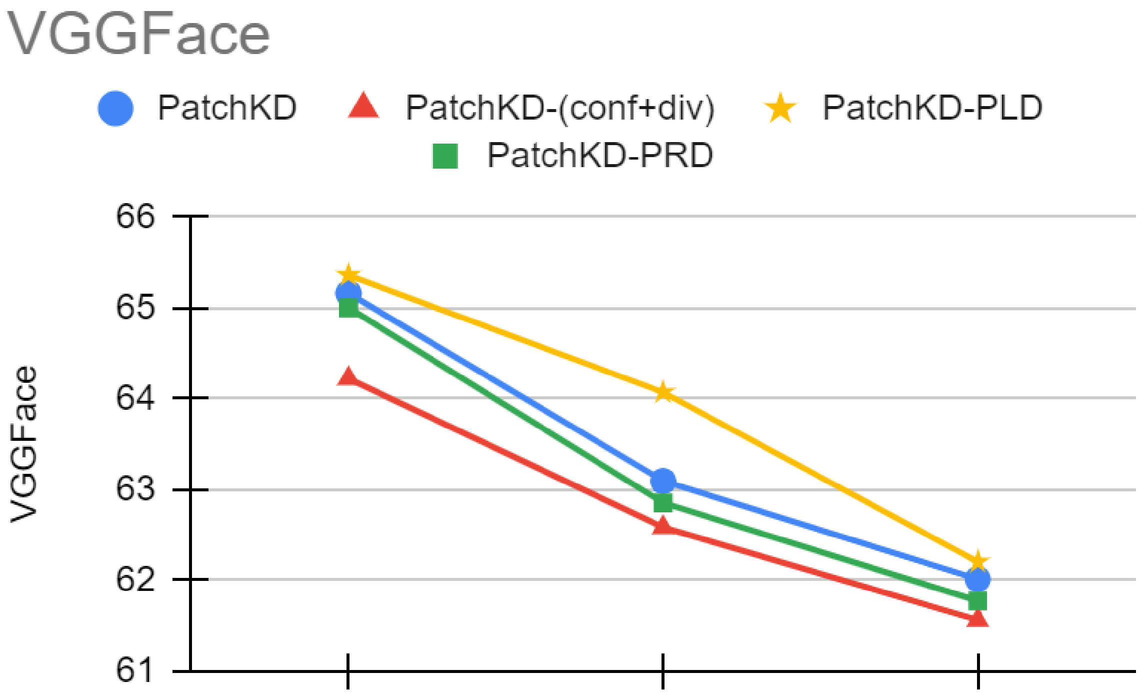


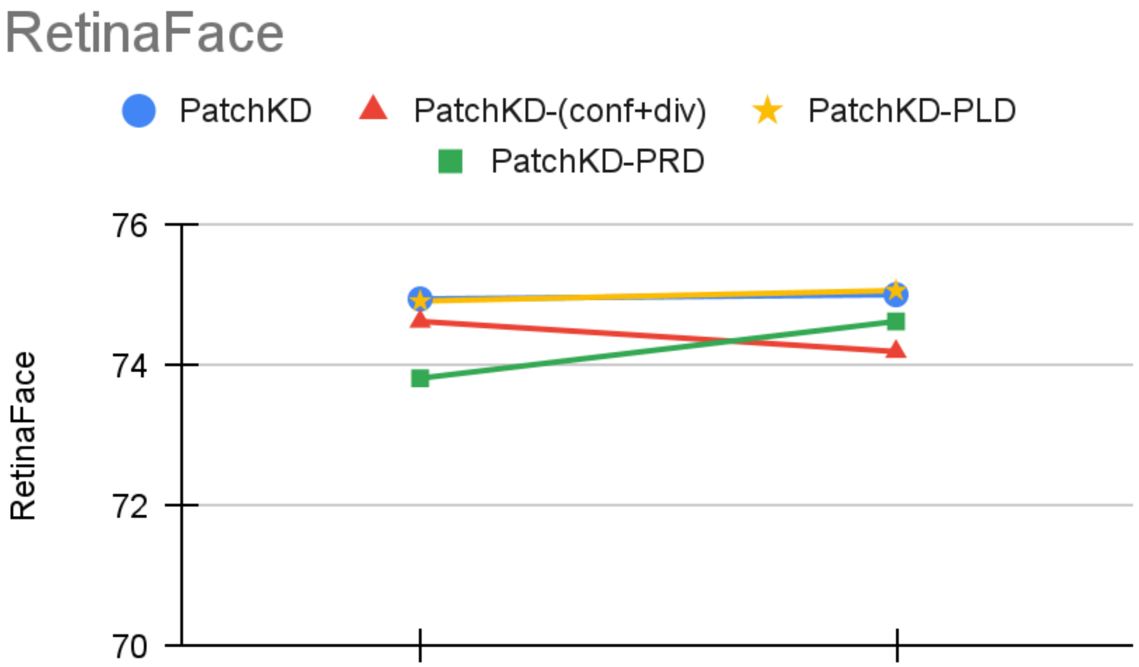
Results with all loss functions on

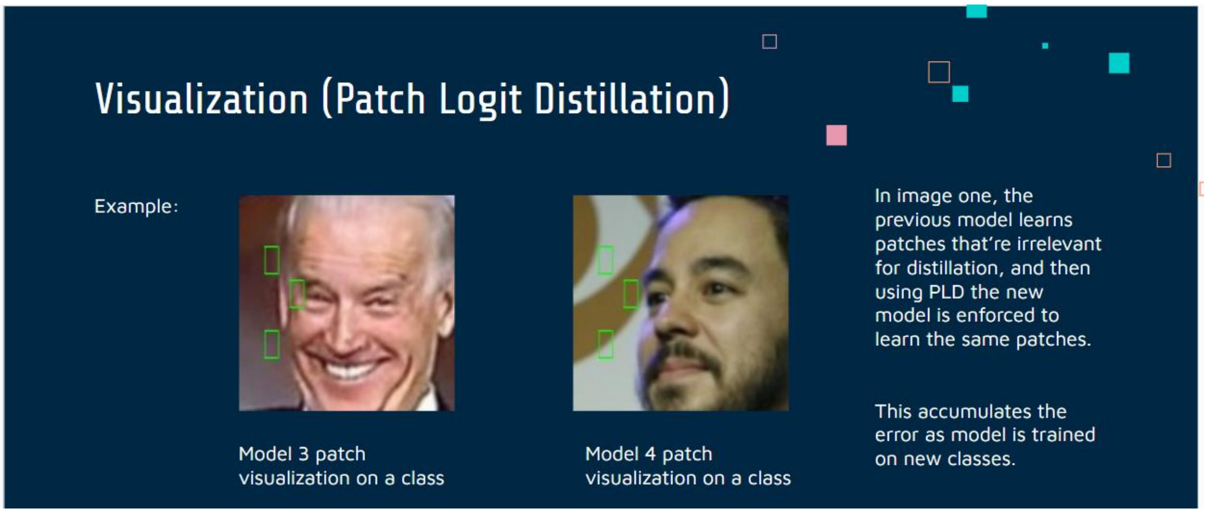
ABLATION STUDIES




ArcFace
● PatchKD
▲ PatchKD-(conf+div)
★ PatchKD-PLD
■ PatchKD-PRD
ArcFace
83
82
81
80
79
78


VGGFace
● PatchKD
▲ PatchKD-(conf+div)
★ PatchKD-PLD
■ PatchKD-PRD
VGGFace
66
65
64
63
62
61





Visualization (Patch Logit Distillation)

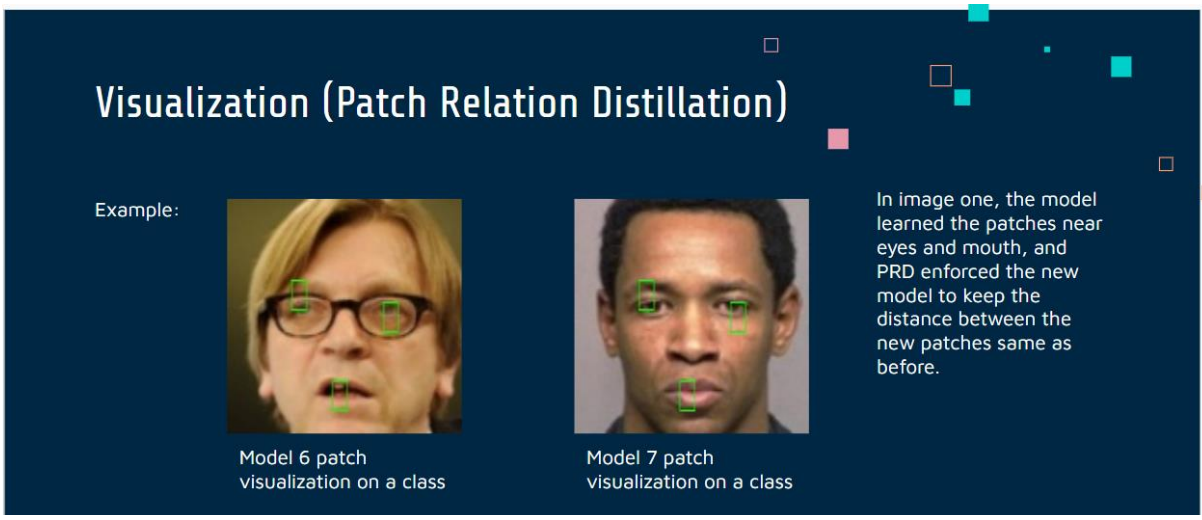
Example:

Model 3 patch visualization on a class

Model 4 patch visualization on a class

In image one, the previous model learns patches that're irrelevant for distillation, and then using PLD the new model is enforced to learn the same patches.

This accumulates the error as model is trained on new classes.



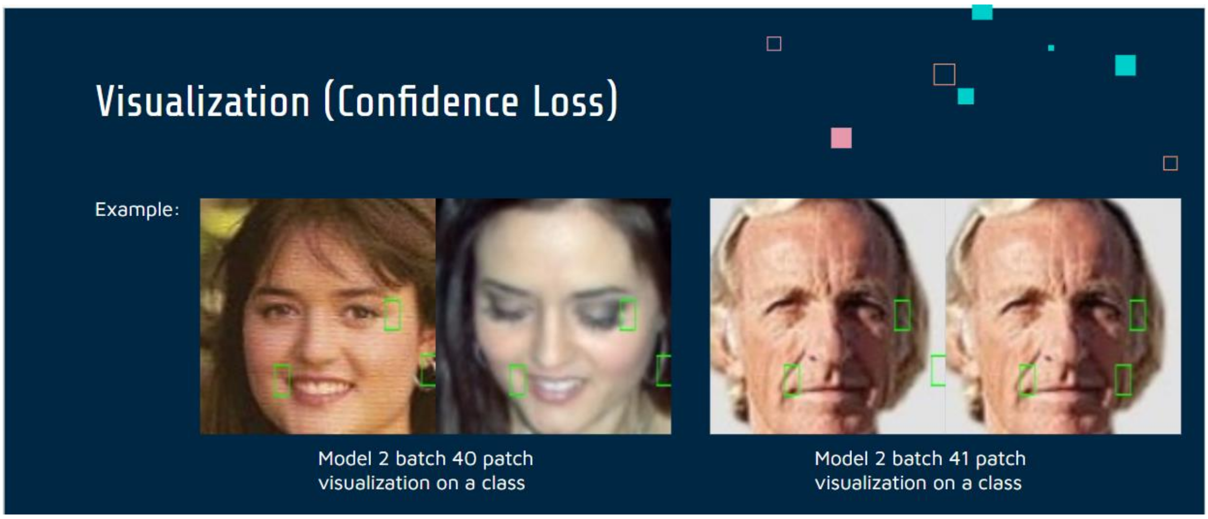
Visualization (Patch Relation Distillation)

Example:

Model 6 patch visualization on a class

Model 7 patch visualization on a class

In image one, the model learned the patches near eyes and mouth, and PRD enforced the new model to keep the distance between the new patches same as before.

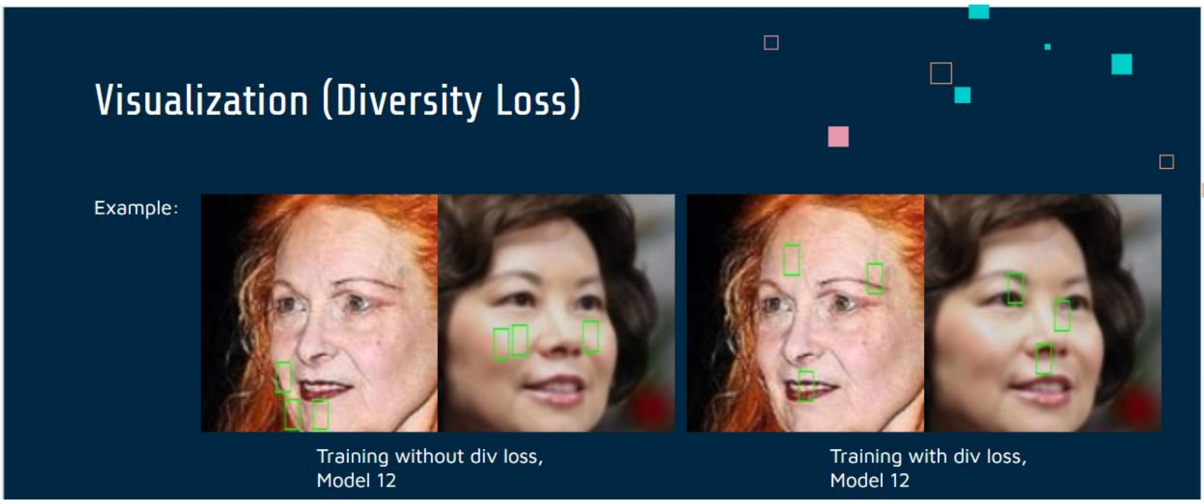


Visualization (Confidence Loss)

Example:

Model 2 batch 40 patch
visualization on a class

Model 2 batch 41 patch
visualization on a class



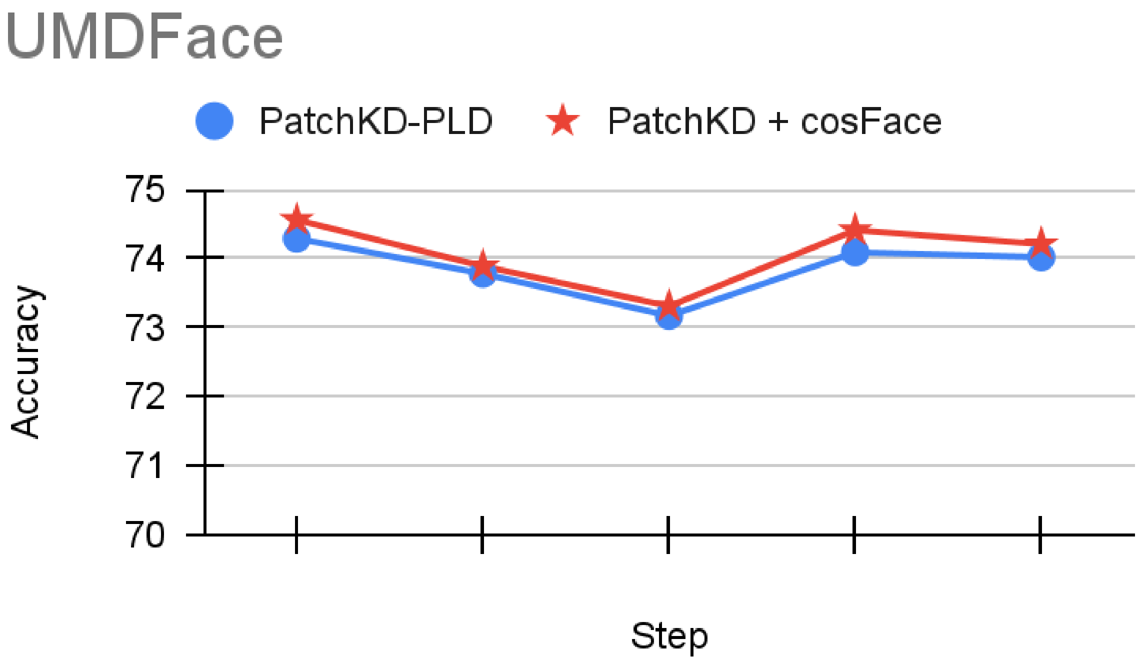
Visualization (Diversity Loss)

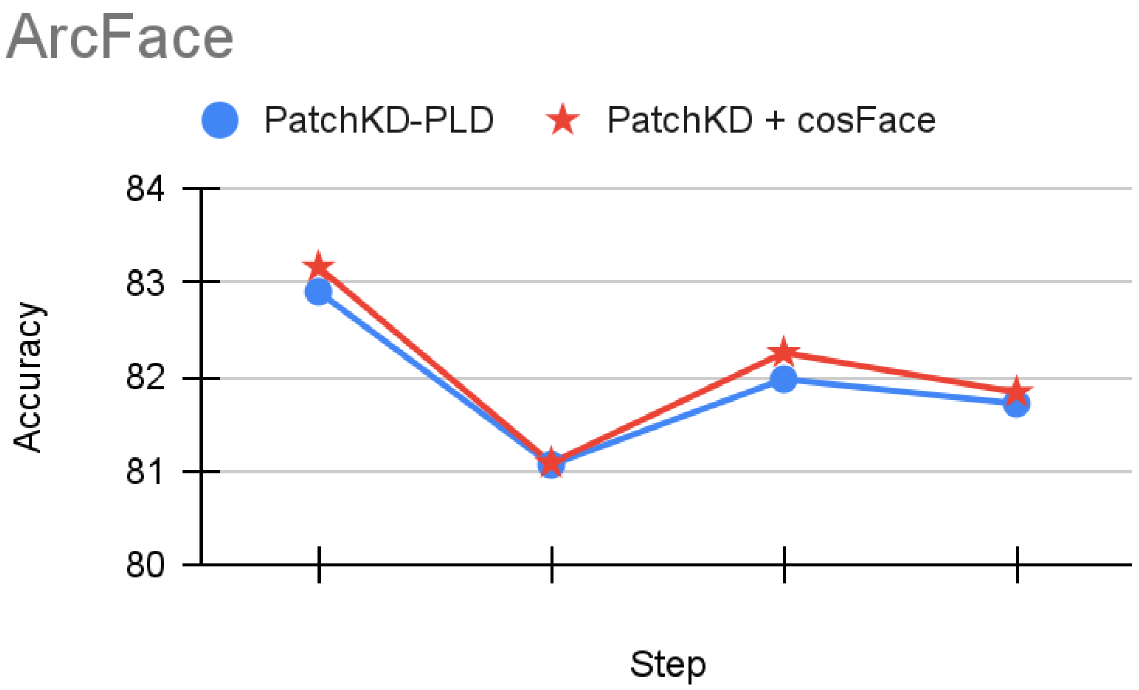
Example:

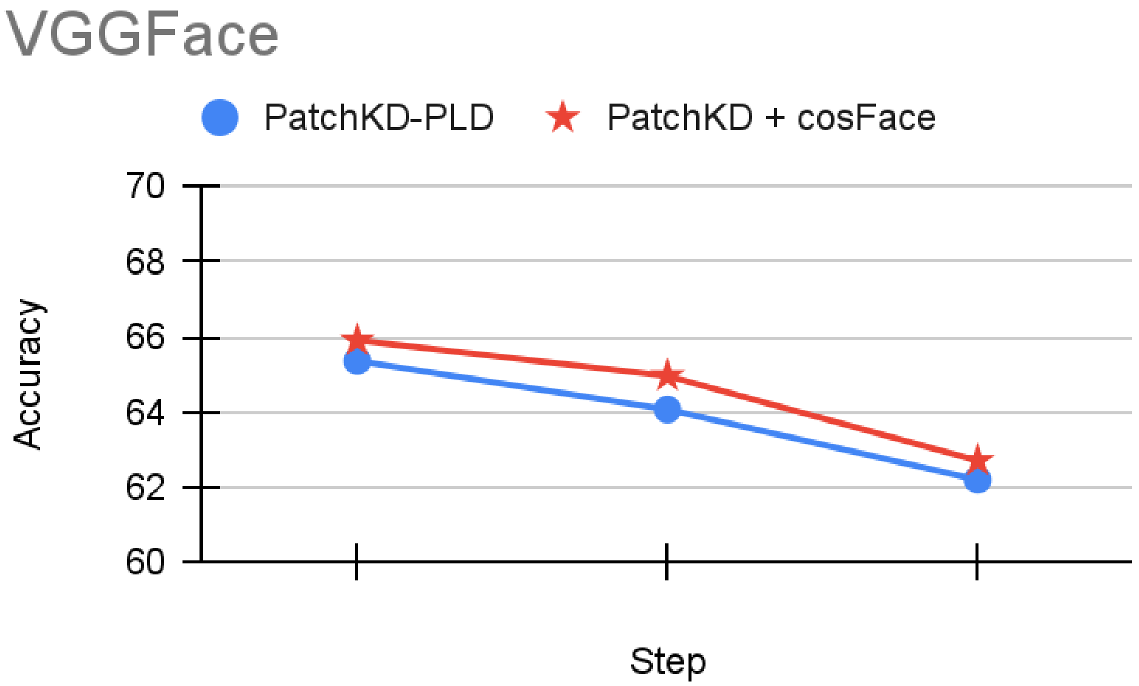
Training without div loss,
Model 12

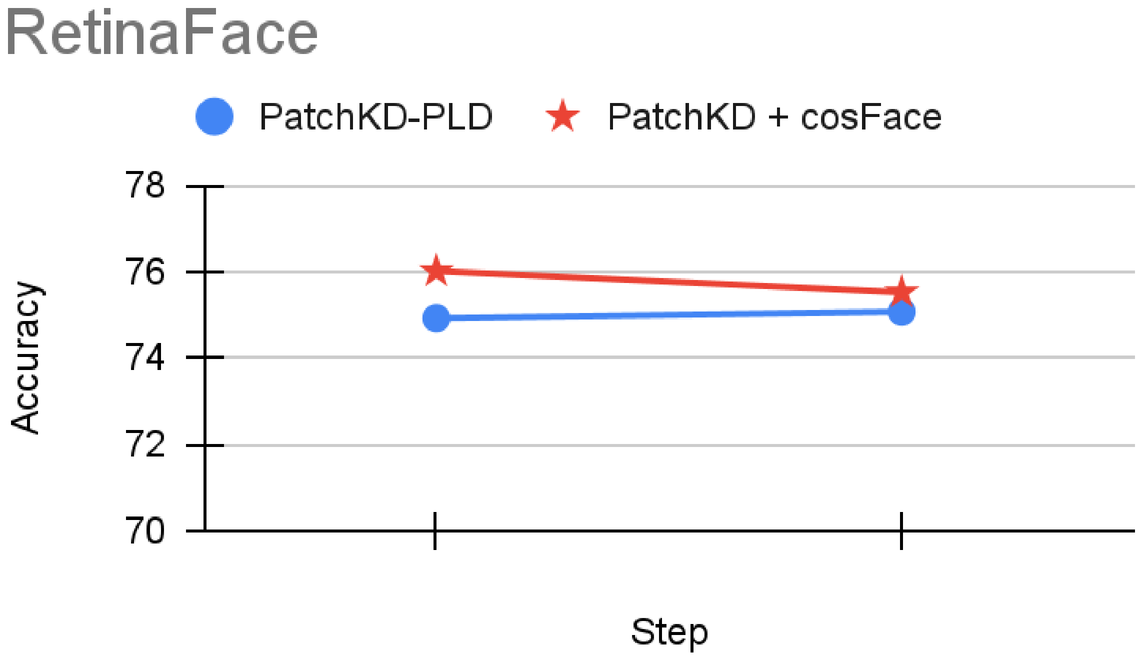
Training with div loss,
Model 12

cosface	UMDFace	ArcFace	VGGFace	RetinaFace	CasiaFace
UMDFace	**74.56**				
ArcFace	**73.89**	**83.17**			
VGGFace	**73.31**	**81.09**	**65.91**		
RetinaFace	**74.41**	**82.26**	**64.96**	**76.01**	
CasiaFace	**74.21**	**81.84**	**62.71**	**75.52**	**57.02**


UMDFace
● PatchKD-PLD
★ PatchKD + cosFace
Accuracy
75
74
73
72
71
70
Step


ArcFace
● PatchKD-PLD
★ PatchKD + cosFace
Accuracy
84
83
82
81
80
Step





Chapter 6: Future Work

In our pursuit of enhancing knowledge distillation robustness, we've embarked on modifying the Patch-Level Distillation (PLD) loss function to achieve greater resilience. Transitioning from logit-based distillation to a feature-based approach, we aim to cultivate more robust representations while ensuring scalability. Our strategy entails testing at T=10 and T=20 steps, as well as K=2 and K=1 shots, to comprehensively evaluate the implications of the cosFace loss function. Through these rigorous evaluations, we seek deeper theoretical insights into the efficacy and applicability of patch-based loss functions, further advancing our understanding and refining our methodologies for knowledge distillation.

Chapter 7: References

1. https://arxiv.org/pdf/1612.00796
2. https://doi.org/10.1007/s10462-019-09734-3
3. https://openaccess.thecvf.com/content_CVPRW_2020/papers/w15/Hayes_Lifelong_Machine_Learning_With_Deep_Streaming_Linear_Discriminant_Analysis_CVPRW_2020_paper.pdf
4. https://openaccess.thecvf.com/content_CVPRW_2020/papers/w15/Hayes_Lifelong_Machine_Learning_With_Deep_Streaming_Linear_Discriminant_Analysis_CVPRW_2020_paper.pdf
5. https://proceedings.neurips.cc/paper/2000/file/155fa09596c7e18e50b58eb7e0c6ccb4-Paper.pdf
6. https://www.researchgate.net/publication/261048597_Incremental_learning_algorithm_for_face_recognition_using_DCT
7. https://openaccess.thecvf.com/content_ECCVW_2018/papers/11130/Belouadah_DeeSIL_Deep-Shallow_Incremental_Learning._ECCVW_2018_paper.pdf
8. http://vision.stanford.edu/documents/Fei-FeiFergusPerona2006.pdf
9. https://proceedings.neurips.cc/paper_files/paper/2004/file/ef1e491a766ce3127556063d49bc2f98-Paper.pdf
10. https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
11. https://proceedings.neurips.cc/paper_files/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf
12. https://ieeexplore.ieee.org/document/8373804
13. https://openaccess.thecvf.com/content/CVPR2021/papers/Bang_Rainbow_Memory_Continual_Learning_With_a_Memory_of_Diverse_Samples_CVPR_2021_paper.pdf
14. https://proceedings.neurips.cc/paper_files/paper/2017/file/0efbe98067c6c73dba1250d2beaa81f9-Paper.pdf
15. https://openaccess.thecvf.com/content_ECCV_2018/papers/Arslan_Chaudhry__Riemannian_Walk_ECCV_2018_paper.pdf
16. https://openaccess.thecvf.com/content_cvpr_2017/papers/Rebuffi_iCaRL_Incremental_Classifier_CVPR_2017_paper.pdf