EyePAD++: A Distillation-based approach for joint Eye Authentication and Presentation Attack Detection using Periocular Images

Prithviraj Dhar\textsuperscript{1}, Amit Kumar\textsuperscript{2}, Kirsten Kaplan \textsuperscript{2}, Khushi Gupta\textsuperscript{2}, Rakesh Ranjan\textsuperscript{2}, Rama Chellappa\textsuperscript{1}

\textsuperscript{1}Johns Hopkins University, \textsuperscript{2}Reality Labs, Meta

\{pdhar1,rchella4\}@jhu.edu, \{akumar14,kkaplan,khushigupa,rakeshr\}@fb.com

Abstract

A practical eye authentication (EA) system targeted for edge devices needs to perform authentication and be robust to presentation attacks, all while remaining compute and latency efficient. However, existing eye-based frameworks a) perform authentication and Presentation Attack Detection (PAD) independently and b) involve significant pre-processing steps to extract the iris region. Here, we introduce a joint framework for EA and PAD using periocular images. While a deep Multitask Learning (MTL) network can perform both the tasks, MTL suffers from the forgetting effect since the training datasets for EA and PAD are disjoint. To overcome this, we propose Eye Authentication with PAD (EyePAD), a distillation-based method that trains a single network for EA and PAD while reducing the effect of forgetting. To further improve the EA performance, we introduce a novel approach called EyePAD++ that includes training an MTL network on both EA and PAD data, while distilling the ‘versatility’ of the EyePAD network through an additional distillation step. Our proposed methods outperform the SOTA in PAD and obtain near-SOTA performance in eye-to-eye verification, without any pre-processing. We also demonstrate the efficacy of EyePAD and EyePAD++ in user-to-user verification with PAD across network backbones and image quality.

1. Introduction

Eye Authentication (EA) using irises has been widely used for biometric authentication. With the current advancements in head-mounted technology, eye-based authentication is likely to become an essential part of authenticating users against their wearable devices. While highly accurate, EA systems are also vulnerable to ‘Presentation Attacks’ (PA) \[14, 47\]. These attacks seek to fool the authentication system by presenting artificial eye images, such as printed iris images of an individual \[5, 16\], or cosmetic contacts \[21\]. While researchers have proposed methods to train networks that achieve SOTA performance in either

![Figure 1. EA pipelines segment out the iris region from the periocular image and normalize the iris image before feeding it to the network. Segmentation/detection is also used in most PAD pipelines. We train a single network for both EA and PAD without any pre-processing steps, using the entire periocular image.](image-url)
We propose a distillation-based method called EyePAD++. To incrementally learn EA and PAD tasks, while minimizing the forgetting effect. In summary, we make the following contributions in this work:

1. We propose a user-to-user verification protocol that can perform user-to-user verification, i.e. consider both left and right eyes of a given test subject (query user) and verify it against one or more pairs of left-right irises of same or different user (i.e. gallery user). To this end, we propose a new evaluation protocol to match the left-right pair of a query user with that of a gallery user.

We consider the problem of EA and PAD as a disjoint multitask learning problem because the authentication task assumes real images, which is why the current datasets for EA do not include PAD labels. A possible single-network solution is to train a deep multitask network for both tasks, alternately training the EA and PAD branches with their respective dataset each iteration (as done in [37]). However, several works [18, 22, 27] have shown that Multitask Learning (MTL) frameworks for disjoint tasks demonstrate forgetting effect (see Sec. 3). Hence, we propose two novel knowledge distillation-based techniques called EyePAD and EyePAD++ to incrementally learn EA and PAD tasks, while minimizing the forgetting effect. In summary, we make the following contributions in this work:

1. We propose a user-to-user verification protocol that can be used to authenticate one query user against one or many samples of a gallery user. This is more practical than the existing protocol for eye-to-eye verification.

2. To the best of our knowledge, we are the first to explore the problem of EA and PAD using a single network. We introduce a new metric called Overall False Rejection Rate (OFRR) to evaluate the performance of the entire system (EA and PAD), using only authentication data.

3. We propose a distillation-based method called Eye Authentication with Presentation Attack Detection (EyePAD) for jointly performing EA and PAD. To further improve the verification performance, we propose EyePAD++. EyePAD++ inherits the versatility of the EyePAD network through distillation and combines it with the specificity of multitask learning. EyePAD++ consistently outperforms the existing baselines for MTL, in terms of OFRR. We show the efficacy of EyePAD and EyePAD++ across different network backbones (Densenet121 [20], MobilenetV3 [19] and HRnet64 [44]), and image quality degradation (blur and noise). Additionally, we apply our methods to jointly perform eye-to-eye verification and PAD, following the commonly used train-test protocols. Although the current SOTA approaches use pre-processing, our proposed methods outperform the existing SOTA in PAD task, and obtain comparable user-to-user verification performance without any pre-processing.

### 2. Related work

**Eye authentication using irises:** Daugman [6, 7] introduced the first automated system for EA by applying Gabor Filters to the normalized image for generating spatial barcode-like features (IrisCode). More recently, several works have proposed using deep features for EA. [12] proposed DeepIrisNet, the first deep learning-based framework for generalized EA, followed by [13, 32, 43]. [53] presents UniNet, that consists of two components: one for generating discriminative features (FeatNet) and the other for segmenting the iris and non-iris region (MaskNet). Both of these components accept the normalized iris images that also require segmentation. [45] uses dilated convolution kernels for training CNNs for EA. [51] presents an encoder-decoder pipeline to extract multi-level iris features and use an attention module to combine the multi-level features.

**Eye-based Presentation Attack Detection:** PAD in periocular images has received significant attention from the deep learning community in the past few years [3, 11, 31, 33, 41, 47]. [25, 46] propose fusing handcrafted and CNN features to detect PA. [10] fuses the features from different layers in a deep network extracted for normalized iris images for PAD. [41, 47] show that DenseNet architecture helps to achieve high PAD accuracy. [17] proposes dividing the iris region into overlapping patches and training CNNs using these patches. [3] introduces an attention guided mechanism to improve PAD accuracy. [11] introduces a binary pixel-wise supervision with self attention to help the network to find patch-based cues and achieve high performance in PAD.

All of the EA algorithms use the normalization process proposed in [7] that requires iris segmentation. Similarly, most PAD algorithms also use auxiliary pre-processing steps such as iris detection/segmentation. A brief summary of

| Method               | EA | PAD | Pre-processing                  |
|----------------------|----|-----|----------------------------------|
| IrisCode [29]        | ✓  | x   | Segmentation, geometric normalization |
| Ordinal [42]         | ✓  | x   | Segmentation, geometric normalization |
| UniNet [53]          | ✓  | ✓   | Segmentation, geometric normalization |
| DFRNet [45]          | ✓  | x   | Segmentation, geometric normalization |
| [35]                | ✓  | ✓   | Segmentation, geometric normalization |
| [14]                | ✓  | ✓   | Cropping                         |
| [33]                | ✓  | ✓   |                                 |
| DensePAD [47]        | x  | ✓   | Segmentation, geometric normalization |
| [17]                | ✓  | ✓   | Segmentation with UIST [38]      |
| D-net-PAD [41]       | ✓  | ✓   | Detection with VeriEye           |
| [3]                 | ✓  | ✓   | Detection with [2]               |
| PBS, A-PBS [11]      | ✓  | ✓   | None                             |
| EyePAD (ours)        | ✓  | ✓   | None                             |
| EyePAD++ (ours)      | ✓  | ✓   | None                             |

Table 1. Pre-processing steps in recent EA/PAD frameworks
the preprocessing steps in EA and PAD is given in Table 1.

### Disjoint Multitask Learning and Knowledge Distillation

Disjoint multitask learning (MTL) is the process of training a network to perform multiple tasks using data samples that have labels for either of the tasks, but not for all the tasks. Training a single network for EA and PAD is a disjoint MTL task because EA datasets do not include PAD labels. One solution is to follow the existing disjoint multitask learning strategies [24, 28, 37] and update each branch of the network alternately. However, it is well known [22, 27] that alternating training suffers from the forgetting effect [27] and degrades performance in multitask learning. Knowledge Distillation (KD) [15] has been commonly used to reduce forgetting in continual learning [9, 27, 39, 40, 52]. Inspired by this, [22, 26] employ feature-level KD for multitasking. In [22], KD is used to distill the information from the network from a previous iteration \( i - 1 \) that was updated for task A (teacher), while training it to perform task B in the current iteration \( i \) (student). However, in this scenario, the teacher network is not fully trained in the initial few iterations and thus the distillation step may not help preserve task A information. Similar to [22], we propose strategies employing feature-level KD for disjoint multitasking (EA and PAD). But, unlike [22], we ensure that the teacher network in our proposed methods is fully trained in one or more tasks.

### 3. Proposed Approach

Our objective is to build a single network that is proficient in performing two disjoint tasks: EA and PAD. We intend to build this framework for edge devices with limited on-device compute. Thus, we exclude any pre-processing step for detecting or segmenting the iris region and use the entire periorcular image as input. Mutiltask Learning (MTL) is a possible approach in this scenario. Most of the MTL methods for disjoint tasks [37] alternately feed the data from different tasks. However, as shown in [22], MTL demonstrates the forgetting effect. Consider an MTL network with shared backbone and different heads designed to perform two tasks A and B. Suppose that the training batches for task A and B are fed to this MTL network alternately. Here, the weights of the shared backbone modified by the gradients corresponding to the loss for task A in iteration \( i \) may be rewritten in the next iteration \( (i + 1) \) by the gradients corresponding to the loss for task B. This may lead to forgetting of task A. Therefore, instead of MTL, we propose to use knowledge distillation to learn both tasks through a single network. Here, we intend to first train a teacher network \( M_t \) for EA, following which we train a student network \( M_s \) for PAD, while distilling the authentication information from \( M_t \) to \( M_s \) to minimize the forgetting effect.

#### 3.1. Eye Authentication with Presentation Attack Detection (EyePAD) and EyePAD++

We now explain the steps in our proposed methods: EyePAD and EyePAD++ (Fig. 2b):

**Step 1**: We train the teacher network \( M_t \) using periorcular images from the EA dataset to perform EA. Similar to [45], we use triplet loss to train \( M_t \). We first extract features \( f_i \) for all the images using the penultimate layer of \( M_t \). To select the \( n^{th} \) triplet in a given batch, we randomly select an anchor feature \( f_{a}^{(n)} \) belonging to category \( C \). After that, we select the hardest positive feature \( f_{pos}^{(n)} \) and the hardest negative feature \( f_{neg}^{(n)} \) as follows:

\[
f_{pos}^{(n)} = \argmax_{i \in C, i \neq a} (\| f_i^{(n)} - f_a^{(n)} \|^2), f_{neg}^{(n)} = \argmin_{i \neq C} (\| f_i^{(n)} - f_a^{(n)} \|^2)
\]

Then we compute the triplet loss \( L_{id} \) for the entire batch (of size \( N \)) as:

\[
L_{id} = \frac{1}{N} \sum_{i=1}^{N} \max(\| f_{pos}^{(n)} - f_a^{(n)} \|^2, \| f_{neg}^{(n)} - f_a^{(n)} \|^2 + \alpha, 0)
\]

where \( \alpha \) denotes the distance margin.

**Step 2** (Feature-level knowledge distillation - EyePAD):

We initialize a student network \( M_s \) using \( M_t \), and train it for PAD. Let \( I \) be an image from the PAD dataset. \( I \) is fed to...
both $M_t$ and $M_s$, to obtain features $f_t$ and $f_s$, extracted using the penultimate layer of the corresponding networks. To constrain $M_s$ to process an eye image like $M_t$, we employ feature-level KD and minimize the cosine distance between $f_t$ and $f_s$ using the proposed distillation loss $L_{dis}$.

$$L_{dis}(f_s, f_t) = 1 - \frac{f_s \cdot f_t}{\|f_s\| \|f_t\|}.$$  \hspace{1cm} (2)

Our application of feature-level KD is inspired by [40]. Note that we do not apply KD on the output scores as done in [27], since eye-based matching protocols like [45] use the features from the penultimate layer (and not the output score vector). Additionally, we would like $M_s$ to classify a given image as live (also referred to as ‘real’ or ‘bona-fide’) or spoof using $L_{pad}$, which is a standard cross-entropy classification loss. Combining these constraints, we train $M_s$ using the multitask classification loss $L_{multi}$ as

$$L_{multi} = L_{pad} + \lambda_1 L_{dis},$$  \hspace{1cm} (3)

where $\lambda_1$ is used to weight $L_{dis}$. In this step, the teacher $M_t$ remains frozen. To evaluate the verification performance, we use features extracted from the penultimate layer of the trained $M_s$ for the test EA data and perform user-to-user verification. We feed the test PAD data to $M_s$ and evaluate its performance in live/spoof classification. We find that the student network $M_s$ obtained from EyePAD is a versatile network that is effective for both EA and PAD. However, compared to $M_t$ (that was only trained for EA), $M_s$ obtains slightly lower verification performance. We hypothesize that $M_s$ demonstrates this drop in performance because it was never trained for EA. Therefore, we introduce an additional step to train an MTL network (initialized with $M_s$) while distilling the versatility of the EyePAD student to this network (Fig. 2b).

**Step 3 (EyePAD++):** We initialize a new student network $M_s$ using $M_s$. $M_s$ is trained for both EA and PAD in an MTL fashion. Following the commonly used strategies for disjoint multitasking [37], the batches from EA and PAD data are alternated after every iteration. To reduce forgetting, we additionally constrain $M_s^*$ to mimic $M_s$, which acts as its teacher, using the same knowledge distillation used in step 2. We feed the training image to both $M_s$ and $M_s^*$ and obtain features $f_s$ and $f_s^*$ respectively. $M_s$ remains frozen in this step. We use them to compute $L_{dis}$ as:

$$L_{dis}(f_s, f_s^*) = 1 - \frac{f_s \cdot f_s^*}{\|f_s\| \|f_s^*\|}.$$  \hspace{1cm} (4)

When authentication data is fed to $M_s^*$ (say, during iteration $i$), we compute $L_{multi}$ as follows:

$$L_{multi}^{id} = L_{id} + \lambda_2 L_{dis}$$  \hspace{1cm} (5)

Here, $L_{id}$ is the triplet loss from Eq. 1. When PAD data is fed to $M_s^*$ (during iteration $i + 1$), we compute $L_{multi}^{pad}$ as:

$$L_{multi}^{pad} = L_{pad} + \lambda_2 L_{dis}$$  \hspace{1cm} (6)

where $L_{pad}$ is a standard classification loss used in step 2 of EyePAD. Thus, $L_{multi}^{id}$ and $L_{multi}^{pad}$, are alternately used to optimize $M_s^*$. For inference, we use the trained $M_s^*$ for user-to-user verification and PAD. Hyperparameter details for EyePAD and EyePAD++ are provided in the supplementary material.

4. Experiments

4.1. Baseline methods

**Single task networks:** To estimate the standard user-to-user verification (EA) and PAD performance, we test the ‘EA only’ and ‘PAD only’ networks. The ‘EA only’ network is the teacher network $M_t$ used in Step 1 of EyePAD.

**Multitask Learning (MTL):** We train a multitask network for EA and PAD by alternately feeding EA and PAD batches (see step 3 of EyePAD++) and alternately optimizing using $L_{pad}$ (Sec. 3.1) and $L_{id}$ (Eq. 1).

**Multi-teacher Multitasking (MTMT)** [26]: MTMT [26] is a recently proposed multitask framework that combines MTL with multi-teacher knowledge distillation. MTMT has been shown to outperform MTL and other SOTA multitask methods such as GradNorm [4]. Here, single task networks are first trained in specific tasks. An MTL network $M$ is then trained for multiple tasks, while information from the single task networks is distilled into $M$. A key difference between MTMT and EyePAD++ is that MTMT enforces distillation from multiple task-specific teachers whereas EyePAD++ includes distillation from a single teacher that is proficient in performing multiple tasks. We implement MTMT as one of our baselines for joint EA and PAD (Fig. 2a).

Firstly, we train two single task models: $M_{auth}$ for EA and $M_{pad}$ for PAD, and then distill information from them while training a student MTL network $M$. We use the same feature-level distillation used in EyePAD (Step 2) and EyePAD++. A given image is fed to $M_{auth}$, $M$ and $M_{pad}$, generating features from the penultimate layers $f_{auth}$, $f_M$ and $f_{pad}$, respectively. $L_{dis}$ then constrains $f_M$ to be closer to $f_{auth}$ and $f_{pad}$. $M_{auth}$ and $M_{pad}$ remain frozen in this step. We alternately feed the training batches for EA and PAD. So, when EA data is forwarded to $M_{auth}$, $M_{pad}$, $M$, we optimize $M$ using $L_{mtmt}$:

$$L_{mtmt}^{id} = L_{id} + \lambda_{auth} L_{dis}(f_{auth}, f_M) + \lambda_{pad} L_{dis}(f_{pad}, f_M)$$  \hspace{1cm} (7)

Here $L_{id}$ is the triplet loss defined in Eq. 1. $\lambda_{auth}$, $\lambda_{pad}$ denote the distillation weights from teacher $M_{auth}$ and $M_{pad}$.
The datasets we use in this work are academic datasets [1, 48–50] with high quality images (Fig. 3a). However, real-world authentication on edge devices rely on small sensors which capture low-resolution images. Also, environmental conditions like lighting may further degrade the image quality. Therefore, in addition to using the original datasets, we also perform experiments by degrading the datasets (separately): (i) Blur: We add Gaussian blur with a random kernel size between 1 and 5 to the training images, and add blur with kernel size of 5 to the test images (Fig. 3b). (ii) Noise: We add Additive White Gaussian Noise with a standard deviation \( \sigma = 3.0 \) (Fig. 3c).

### 4.2. Datasets and network architectures used

We summarize the datasets used in our work in Table 2.

| User-to-user verification (EA) | PAD |
|-------------------------------|-----|
| Data | # images | Data | # images |
| Train | 206 users from ND-Iris-0405 | 7949 | Train split of CU-LivDet (2013,2015,2017) | 14600 |
| Test | 150 users from ND-Iris-0405 | 4231 | Test split of CU-LivDet (2013,2015,2017) | 7532 |

Table 2. Statistics for datasets used for EA with PAD

respectively. Similarly, when PAD data is forwarded, we optimize \( M \) using \( L_{\text{mtmt}} \).

\[
L_{\text{mtmt}} = L_{\text{pad}} + \lambda_{\text{auth}} L_{\text{dis}}(f_{\text{auth}}, f_{M}) + \lambda_{\text{pad}} L_{\text{dis}}(f_{\text{pad}}, f_{M})
\]

(8)

Here \( L_{\text{pad}} \) is standard classification loss for live/spoof classification. We provide the hyperparameter information for MTMT [26] in the supplementary material.

### 4.3. User to user verification protocol

Most experiments in EA [45, 53] train the model on the left irises of all the users and evaluate them in terms of the eye-to-eye verification accuracy for the right irises. However, in a real-world authentication system, the gallery will most likely have both left and right eye images (instead of only right eye images) for an authorized user, and thus both left and right query images can be used for verification. Moreover, it is more practical to authenticate a user using both eyes, as opposed to only the right eye. Hence, we propose matching one pair of left-right eyes (query) to \( K \) pairs of left-right eyes (gallery). We provide the detailed user-to-user verification protocol in Protocol 1. To match query user \( q_A \) and gallery user \( g_B \), we first randomly select one left eye and one right eye image from \( q_A \). Then, we select \( K \) left eye and \( K \) right eye images from \( g_B \). After that, we feed the left and right query images to a model \( M \) and compute their respective features \( f_{q_A}^{(L)}, f_{q_A}^{(R)} \).

For gallery user \( B \), we compute the features for the \( K \) left eye images using \( M \) and average them to compute a single feature \( f_{g_B}^{(L)} \) (Line 12 of Protocol 1). In the same way, we compute the average feature \( f_{g_B}^{(R)} \) by for the right eye gallery

**Networks used:** We implement our proposed methods and baselines using the DenseNet121 backbone [20]. This is motivated by this architecture repeatedly demonstrating high PAD performance [11, 41, 47]. To demonstrate the generalizability of EyePAD and EyePAD++, we repeat our experiments using the HRnet64 [44] and MobilenetV3 [19].
Figure 4. User-to-user verification: Verifying query user A (q_A) against gallery user B (g_B) with K pairs for gallery user B.

**Protocol 2 Computing OFRR**

1: Required: Model M. Query dataset Q, Gallery dataset G
2: Required: Similarity dictionary S from Protocol 1
3: Required: Query dictionaries Q_L, Q_R from Protocol 1
4: Required: Similarity threshold t_auth from Protocol 1
5: Required: PAD threshold t_pad for SAR=5%
6: Initialize: Spoof rejects X_{spoof} = 0, EA rejects X_{auth} = 0
7: for Query user q_A, gallery user g_A in Q, G do
8: q_A^{(L)} ← Q_L[q_A], q_A^{(R)} ← Q_R[q_A]
9: Left query PAD logit o_A^{(L)} = M(q_A^{(L)})
10: Right query PAD logit o_A^{(R)} = M(q_A^{(R)})
11: if o_A^{(L)} > t_pad or o_A^{(R)} > t_pad then
12: X_{spoof} = X_{spoof} + 1 (falsely rejected as spoof)
13: else if S[q_A, g_A] < t_auth then
14: X_{auth} = X_{auth} + 1 (falsely rejected as non-match)
15: end if
16: end if
17: end if
18: end for
19: Overall false rejection rate OFRR = (X_{spoof} + X_{auth})/|Q|

We then compute the similarity between query user A and gallery user B as:

\[ s(q_A, g_B) = \frac{1}{2} \left( \frac{f_A^{(L)} \cdot f_B^{(L)}}{\| f_A^{(L)} \| \| f_B^{(L)} \|} + \frac{f_A^{(R)} \cdot f_B^{(R)}}{\| f_A^{(R)} \| \| f_B^{(R)} \|} \right) \]  

Based on the similarity threshold, a match/non-match is predicted (Fig. 4).

**4.4. Metrics for EA and PAD**

Performing the similarity computation (Eq. 9) for every possible pairs from (Q, G) and varying the similarity threshold for deciding match/non-match, we compute the ROC curve and report the True Acceptance Rates (TARs) at FAR=10^{-4}, 10^{-3}, 10^{-2}. In the biometrics literature [8, 30, 36], it is common to use several gallery samples in authentication. But, for authentication on edge devices, the number of gallery samples that can be used depends on the storage capacity of the edge device. Therefore, for evaluating the EA performance of a given model, we use one query left-right pair and \( K = 1/2/5 \) gallery left-right pair(s) for verification.

PAD performance is evaluated with four commonly used metrics: (i) True Detection Rate (TDR) at a False Detection Rate of 0.002, (ii) Attack Presentation Classifier Error Rate (APCER), that is the fraction of spoof samples misclassified as Live, (iii) Bonafide Presentation Classifier Error Rate (BPCER), that is the fraction of live samples misclassified as spoof, (iv) Half Total Error Rate (HTER), the average of APCER and BPCER. Following the protocol in [48], we use a threshold of 0.5 for computing APCER and BPCER.

While these metrics gauge either the EA or PAD performance, they cannot jointly evaluate PAD and EA. Hence, we define a new metric in the next subsection.

**4.4.1 Overall False Reject Rate (OFRR)**

We evaluate EA performance on the test EA data (ND-Iris-0405) and the PAD performance on the test subset of CU-LivDet datasets from the 2013, 2015 and 2017 challenges. An ideal metric must measure PAD and EA performance simultaneously on a single dataset. Such a metric must measure: How often does the model reject true users from accessing the system? A true user in the EA dataset can be falsely rejected as: (1) ‘Spoof’ by the PAD pipeline, or (2) ‘Non-match’ by the user-to-user verification pipeline. In this regard, we introduce a new metric called Overall False Rejection Rate (OFRR) for true query users in the EA test subset. The steps for computing the OFRR of true users are summarized in Protocol 2. To determine OFRR, we must first set thresholds for the rates at which PAD misclassifies spoof as live (i.e. Spoof Acceptance Rate or SAR) and EA falsely accepts non-match pairs (FAR). The PAD threshold \( t_{pad} \) is computed as the point where SAR=5\% when PAD test data is fed to the model. Similarly, when EA test data is fed to the model, the similarity threshold \( t_{auth} \) is computed as the point where the user-to-user verification results in FAR=10^{-3}. After computing \( t_{pad} \) and \( t_{auth} \), we feed the EA test data to the model again. We then compute the number of query users falsely rejected as spoof (\( X_{spoof} \)) using \( t_{pad} \). Here, we reject a query user if at least one of the associated eye images is classified as spoof (Line 12 in Protocol 2). For those query users classified as live, we verify them for EA against matching gallery users, using our user-to-user verification protocol (Fig 4). We compute the number of query users that are falsely rejected as non-match \( X_{auth} \) using \( t_{auth} \). Finally, we compute the Overall False Reject Rate (OFRR) as

\[ \text{OFRR} = \frac{X_{spoof} + X_{auth}}{|Q|} \]
### 4.5. Results

**EA and PAD with DenseNet backbone:** We perform the EA and PAD experiments using the DenseNet121 backbone for the original datasets and degraded datasets (Table 3). *EyePAD++ obtains the lowest OFRR in most cases.*

### Table 3. PAD results on CU-LivDet

| Method       | OFRR (↑) | TDR (↑) | APCER (↑) | BPCER (↑) | HTER (↑) |
|--------------|----------|---------|-----------|-----------|----------|
| EA only      | 0.100    | 0.034   | 0.762     | 0.332     | 0.483    |
| PAD only     | 0.072    | 0.084   | 0.887     | 0.000     | 0.079    |
| MTL          | 0.093    | 0.092   | 0.906     | 0.065     | 0.033    |
| MTMT [26]    | 0.034    | 0.033   | 0.938     | 0.126     | 0.027    |

### Table 4. User-to-user verification results on ND-Iris-0405 (EA)

| Method       | OFRR (↑) | TDR (↑) | APCER (↑) | BPCER (↑) | HTER (↑) |
|--------------|----------|---------|-----------|-----------|----------|
| EA only      | 0.110    | 0.011   | 0.877     | 0.098     | 0.095    |
| PAD only     | 0.085    | 0.088   | 0.890     | 0.096     | 0.095    |
| MTL          | 0.100    | 0.034   | 0.762     | 0.332     | 0.483    |
| MTMT [26]    | 0.034    | 0.033   | 0.938     | 0.126     | 0.027    |
| EyePAD       | 0.093    | 0.092   | 0.906     | 0.065     | 0.033    |
| EyePAD++     | 0.072    | 0.084   | 0.887     | 0.000     | 0.079    |

### Table 5. EA and PAD with MobilenetV3, trained and evaluated on the (top) original, (middle) blurred, (bottom) noisy data: For user-to-user verification, we report TAR@FAR=10^−4, 10^−3, 10^−2. For PAD, we report TDR@FDR=0.002 and APCER, BPCER, HTER. OFRR jointly measures EA and PAD performance on ND-Iris-0405. *EyePAD++ obtains the lowest OFRR. Bold: Best, Underlined: Second best.*

| Method       | OFRR (↑) | TDR (↑) | APCER (↑) | BPCER (↑) | HTER (↑) |
|--------------|----------|---------|-----------|-----------|----------|
| EA only      | 0.100    | 0.034   | 0.762     | 0.332     | 0.483    |
| PAD only     | 0.072    | 0.084   | 0.887     | 0.000     | 0.079    |
| MTL          | 0.093    | 0.092   | 0.906     | 0.065     | 0.033    |
| MTMT [26]    | 0.034    | 0.033   | 0.938     | 0.126     | 0.027    |
| EyePAD       | 0.093    | 0.092   | 0.906     | 0.065     | 0.033    |
| EyePAD++     | 0.072    | 0.084   | 0.887     | 0.000     | 0.079    |

The user-to-user verification and OFRR protocols are run consecutively, and depend on random samples of left and right images of query and gallery users as shown in Protocols 1 (Lines 5,6,7,8) and 2 (Lines 2, 3). Hence, we compute these metrics ten times and report the average.
of the the problem settings. Moreover, EyePAD++ obtains higher user-to-user verification performance than existing multitasking baselines at most FARs. This demonstrates the advantage of the additional distillation step combined with MTL. The PAD performance demonstrated by EyePAD++ is also comparable to that of other multitasking baselines.

**EA and PAD with HRnet64 and MobilenetV3 backbone:** To demonstrate the generalizability of our proposed methods, we repeat the same experiments with the HRnet64 [44] backbone (Table 4). However, training a Densenet or HRNet64 model is computationally expensive. So, we also perform the same experiment with the MobilenetV3 [19] backbone, that is much more computationally efficient than Densenet (Table 5). More detailed results with 1 or 2 gallery pairs are provided in the supplementary material. Once again we find that EyePAD++ obtains lower OFRR than MTL and MTMT [26]. The superiority of EyePAD++ with MobilenetV3 indicates that EyePAD++ can be used for performing EA and PAD on compute engines with low capacity that are available on edge devices.

**Eye-to-eye verification with PAD:** To compare our proposed methods with current SOTA in PAD and EA, we perform eye-to-eye verification with PAD. Here, for EA, we follow [45, 53] and use the first 25 left eye images of every user in the ND-Iris-0405 dataset [1] for training. We use the first 10 right eye images of the users for testing. For evaluating EA, we use the same eye-to-eye verification in [45, 53]. For PAD, we follow [11, 41] and only use the official train and test split of the CU-LivDet-2017 dataset. We perform this experiment with DenseNet121. While training and testing our methods and baselines (Sec. 4.1), we exclude pre-processing. Fig. 5 shows the ROC curves for EA and PAD obtained by all the methods. From Table 6, we infer that EyePAD and EyePAD++ achieve better PAD performance (i.e. TDR @ FDR=0.002) than the current SOTA PAD algorithms, without any pre-processing. Moreover, EyePAD++ achieves higher EA performance (TAR at FAR=10^{-3}) than the comparable baselines (i.e. EA only network, MTL and MTMT). The EA performance for EyePAD and EyePAD++ is comparable to but slightly lower than that of the SOTA [45, 53], with a difference of less than 4%. We believe that this difference is due to excluding pre-processing steps for limiting computational cost.

**EyePAD++ v/s MTMT** [26]: Both EyePAD++ and MTMT combine MTL with feature-level KD. However, the student MTL network in MTMT does not inherit the ‘versatility’ through distillation since its teachers are single-task models that are not versatile. On the other hand, EyePAD++ uses distillation from a single versatile teacher ($M_s$), that is proficient in both the tasks. As a result, the student network $M_a$ in EyePAD++ inherits the versatility of its teacher network $M_s$ through distillation. This enables EyePAD++ to outperform [26] in almost all of problem settings (Tables 3, 4, 5, 6). Thus, for training an MTL network with distillation, we show that using a single teacher proficient in both the tasks is better than using two teachers proficient in single tasks in our disjoint multitasking problem.

### 5. Conclusion

In this work, we propose two knowledge distillation-based frameworks: EyePAD and EyePAD++ for joint EA and PAD tasks. For evaluating EA, we present a new user-to-user verification protocol and introduce a new metric to jointly measure user-to-user verification and PAD. Our proposed methods outperform the existing baselines (MTL and MTMT) in most of the problem settings. We evaluate our methods using different network backbones and multiple image quality degradation. Additionally, we evaluate our methods to perform eye-to-eye verification with PAD (following previous work). Although we do not use any pre-processing, EyePAD and EyePAD++ outperform the SOTA in PAD and obtain eye-to-eye verification performance that is comparable to SOTA EA algorithms.
Acknowledgement

This work was done when the first author was an intern at Meta. This research is partially supported by a MURI from the Army Research Office under the Grant No. W911NF-17-1-0304. This is part of the collaboration between US DOD, UK MOD and UK Engineering and Physical Research Council (EPSRC) under the Multidisciplinary University Research Initiative.

References

[1] Kevin W Bowyer and Patrick J Flynn. The ND-IRIS-0405 iris image dataset. arXiv preprint arXiv:1606.04853, 2016.
[2] Cunjian Chen and Arun Ross. A multi-task convolutional neural network for joint iris detection and presentation attack detection. In 2018 IEEE Winter Applications of Computer Vision Workshops (WACVW), pages 44–51. IEEE, 2018.
[3] Cunjian Chen and Arun Ross. An explainable attention-guided iris presentation attack detector. In WACV (Workshops), pages 97–106, 2021.
[4] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In International Conference on Machine Learning, pages 794–803. PMLR, 2018.
[5] Adam Czajka. Database of iris printouts and its application: Development of liveness detection method for iris recognition. In 2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR), pages 28–33. IEEE, 2013.
[6] John Daugman. How iris recognition works. In The essential guide to image processing, pages 715–739. Elsevier, 2009.
[7] John G Daugman. High confidence visual recognition of persons by a test of statistical independence. IEEE transactions on pattern analysis and machine intelligence, 15(11):1148–1161, 1993.
[8] P Dhar, C Castillo, and R Chellappa. On measuring the iconicity of a face. In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 2137–2145. IEEE, 2019.
[9] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyun Wu, and Rama Chellappa. Learning without memorizing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5138–5146, 2019.
[10] Meiling Fang, Naser Damer, Fadi Boutros, Florian Kirchbuchner, and Arjan Kuijper. Deep learning multi-layer fusion for an accurate iris presentation attack detection. In 2020 IEEE 23rd International Conference on Information Fusion (FUSION), pages 1–8. IEEE, 2020.
[11] Meiling Fang, Naser Damer, Fadi Boutros, Florian Kirchbuchner, and Arjan Kuijper. Iris presentation attack detection by attention-based and deep pixel-wise binary supervision network. In 2021 IEEE International Joint Conference on Biometrics (IJCB), pages 1–8. IEEE, 2021.
[12] Abhishek Gangwar and Akanksha Joshi. Deepirisnet: Deep iris representation with applications in iris recognition and cross-sensor iris recognition. In 2016 IEEE international conference on image processing (ICIP), pages 2301–2305. IEEE, 2016.
[13] Fei He, Ye Han, Han Wang, Jinchao Ji, Yuanning Liu, and Zhiqiang Ma. Deep learning architecture for iris recognition based on optimal gabor filters and deep belief network. Journal of Electronic Imaging, 26(2):023005, 2017.
[14] Lingxia He, Haiqing Li, Fei Liu, Nianfeng Liu, Zhenan Sun, and Zhaofeng He. Multi-patch convolutional neural network for iris liveness detection. In 2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS), pages 1–7. IEEE, 2016.
[15] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In NIPS Deep Learning and Representation Learning Workshop, 2015.
[16] Steven Hoffman, Renu Sharma, and Arun Ross. Convolutional neural networks for iris presentation attack detection: Toward cross-dataset and cross-sensor generalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1620–1628, 2018.
[17] Steven Hoffman, Renu Sharma, and Arun Ross. Iris+ ocular: Generalized iris presentation attack detection using multiple convolutional neural networks. In 2019 International Conference on Biometrics (ICB), pages 1–8. IEEE, 2019.
[18] Yan Hong, Li Niu, Jianfu Zhang, and Liqing Zhang. Beyond without forgetting: Multi-task learning for classification with disjoint datasets. In 2020 IEEE International Conference on Multimedia and Expo (ICME), pages 1–6. IEEE, 2020.
[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1314–1324, 2019.
[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4707–4717, 2017.
[21] Ken Hughes and Kevin W Bowyer. Detection of contact-lens-based iris biometric spoofs using stereo imaging. In 2013 46th Hawaii International Conference on System Sciences, pages 1763–1772. IEEE, 2013.
[22] Dong-Jin Kim, Jinsoo Choi, Tae-Hyun Oh, Youngjin Yoon, and In So Kweon. Disjoint multi-task learning between heterogeneous human-centric tasks. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1699–1708. IEEE, 2018.
[23] Gabriela Y Kimura, Diego R Lucio, Alceu S Britto Jr, and David Menotti. Cnn hyperparameter tuning applied to iris liveness detection. arXiv preprint arXiv:2003.00833, 2020.
[24] Iasonas Kokkinos. Ubersnet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6129–6138, 2017.
[25] Andrey Kuehlkamp, Allan Pinto, Anderson Rocha, Kevin W Bowyer, and Adam Czajka. Ensemble of multi-view learning systems (BTAS), pages 1–8. IEEE, 2016.
[50x131][38] Christian Rathgeb, Andreas Uhl, Peter Wild, and Heinz Hof-

[50x88][39] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg

[50x185][27] Zhizhong Li and Derek Hoiem. Learning without forgetting. [50x614][28] An-An Liu, Yu-Ting Su, Wei-Zhi Nie, and Mohan Kankan-

[50x239][36] R Ranjan, A Bansal, J Zheng, H Xu, J Gleason, B Lu, A Nan-

[50x293][35] Ramachandra Raghavendra, Kiran B Raja, and Christoph

[50x561][29] Libor Masek et al. Recognition of human iris patterns for

[50x561]IEEE transactions on pattern analysis and machine intelli-

[50x656], pages 163–176. Springer, 2020. 3, 4, 5, 7, 8, 12, 13

[50x624], 39(1):102–114, 2016. 3

[50x646], 40(12):2935–2947, 2017. 2, 3, 4

[50x614], 39(1):102–114, 2016. 3

[50x635], pages 163–176. Springer, 2020. 3, 4, 5, 7, 8, 12, 13

[50x656], 40(12):2935–2947, 2017. 2, 3, 4

[50x689], 10(4):864–879, 2015. 2

[50x710], 2017. 3

[50x720], 2017. 3

[50x88][52] Sergey Zagoruyko and Nikos Komodakis. Paying more at-

[50x141]IEEE Transactions on Information Forensics and Security

[50x699], 14(12):3233–

3245, 2019. 1, 2, 3, 4, 5, 8

[50x410], 6:18848–18855,

2017. 2

[50x356], pages 162–169, 2017. 2

[50x345], pages 0–0, 2019. 1, 2, 5

[50x378], 2017. 2

[50x399], pages 572–579, 2018. 2

[50x410], 2017 Conference on Computer Vision and Pattern Recognition Workshops, pages 162–169, 2017. 2

[50x418] and Afzel Noore. Detecting textured contact lens in un-

controlled environment using densepad. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 0–0, 2019. 1, 2, 5

[50x422] and Afzel Noore. Fusion of handcrafted and deep learning features for large-scale multiple iris pres-

sentation attack detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 572–579, 2018. 2

[50x454], Naman Kohli, Akshay Agarwal, Mayank

Vatsa, Richa Singh, and Afzel Noore. Fusion of handcrafted and deep learning features for large-scale multiple iris pres-

entation attack detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 572–579, 2018. 2

[50x475], 14(12):3233–

3245, 2019. 1, 2, 3, 4, 5, 8

[50x486] and Afzel Noore. Detecting textured contact lens in un-

controlled environment using densepad. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 0–0, 2019. 1, 2, 5

[50x496], Naman Kohli, Akshay Agarwal, Mayank

Vatsa, Richa Singh, and Afzel Noore. Fusion of handcrafted and deep learning features for large-scale multiple iris pres-

entation attack detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 572–579, 2018. 2

[50x507], pages 158–165. 2

[50x518], 31(12):2211–2226, 2008. 2, 8

[50x529] and Renu Sharma and Arun Ross. D-netpad: An explainable and

interpretable iris presentation attack detector. In 2016 International Conference on Biometrics (ICB), pages 1–10. IEEE, 2020. 1, 2, 5, 8

[50x540], Xinggang Wang, et al. Deep high-resolution represen-

tation learning for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 2020. 2, 5, 8, 11

[50x550]. PhD thesis, Citeseer, 2003. 2, 8

[50x551], 32(5):831–846, 2009. 5

[50x551], 32(5):831–846, 2009. 5

[50x571]. PhD thesis, Citeseer, 2003. 2, 8

[50x592], 40(12):2935–2947, 2017. 2, 3, 4

[50x603], 10(4):864–879, 2015. 2

[50x614], pages 163–176. Springer, 2020. 3, 4, 5, 7, 8, 12, 13

[50x624], 39(1):102–114, 2016. 3

[50x635], pages 163–176. Springer, 2020. 3, 4, 5, 7, 8, 12, 13

[50x646], 40(12):2935–2947, 2017. 2, 3, 4

[50x656], 40(12):2935–2947, 2017. 2, 3, 4

[50x667], 2015. 3, 4

[50x678], 2017 12th IEEE International Conference on

Automatic Face & Gesture Recognition (FG 2017), pages 17–24. IEEE, 2017. 2, 3, 4

[50x689], 10(4):864–879, 2015. 2

[50x699], 14(12):3233–

3245, 2019. 1, 2, 3, 4

[50x710], 2017. 3

[50x720], 2017. 3

[50x730], 2017. 3

[50x740], 2017. 3

[50x750], 2017. 3

[50x760], 2017. 3

[50x770], 2017. 3

[50x780], 2017. 3

[50x790], 2017. 3

[50x800], 2017. 3

[50x810], 2017. 3

[50x820], 2017. 3

[50x830], 2017. 3

[50x840], 2017. 3

[50x850], 2017. 3

[50x860], 2017. 3

[50x870], 2017. 3

[50x880], 2017. 3

[50x890], 2017. 3

[50x900], 2017. 3

[50x910], 2017. 3

[50x920], 2017. 3

[50x930], 2017. 3

[50x940], 2017. 3

[50x950], 2017. 3

[50x960], 2017. 3

[50x970], 2017. 3

[50x980], 2017. 3

[50x990], 2017. 3

[50x1000], 2017. 3
Supplementary material

In this supplementary material, we provide the following information:

Section A1: Train and test split for user-to-user verification.
Section A2: Hyperparameters for EyePAD and EyePAD++.
Section A3: Detailed results with HRnet and MobilenetV3 backbones.
Section A4: Ablation experiments for $\lambda_1$ (EyePAD).
Section A5: Hyperparameters for baseline methods.

A1. Train and test splits for ND-Iris-0405 dataset [1]

In section 4.2 of the main paper, we mention that we randomly split the users into two subsets: $U_{\text{train}}$ (for training) and $U_{\text{test}}$ (for evaluation). All the images for users in the training split are used for training. Here, we provide the train and test split to enable researchers reproduce our protocol.

Train user IDs: '04200' '04203' '04214' '04233' '04239' '04261' '04265' '04267' '04284' '04286' '04288' '04302' '04309' '04313' '04320' '04327' '04336' '04339' '04349' '04351' '04361' '04370' '04378' '04379' '04382' '04387' '04394' '04395' '04397' '04400' '04407' '04408' '04409' '04418' '04419' '04429' '04430' '04434' '04435' '04436' '04440' '04446' '04447' '04453' '04460' '04471' '04472' '04475' '04476' '04477' '04479' '04481' '04482' '04485' '04495' '04496' '04502' '04504' '04505' '04506' '04511' '04512' '04514' '04530' '04533' '04542' '04553' '04560' '04575' '04577' '04578' '04581' '04587' '04588' '04589' '04593' '04596' '04597' '04598' '04603' '04605' '04609' '04613' '04615' '04622' '04626' '04628' '04629' '04632' '04633' '04634' '04644' '04647' '04653' '04670' '04684' '04687' '04691' '04692' '04695' '04699' '04701' '04702' '04703' '04712' '04715' '04716' '04720' '04721' '04725' '04729' '04734' '04736' '04737' '04738' '04742' '04744' '04745' '04747' '04748' '04751' '04756' '04757' '04758' '04763' '04765' '04768' '04772' '04773' '04774' '04776' '04777' '04778' '04782' '04783' '04785' '04787' '04790' '04792' '04794' '04801' '04802' '04803' '04813' '04815' '04816' '04818' '04831' '04832' '04839' '04840' '04841' '04843' '04846' '04847' '04850' '04854' '04857' '04858' '04859' '04868' '04869' '04870' '04871' '04872' '04873' '04876' '04877' '04887' '04889' '04890' '04891' '04892' '04894' '04897' '04898' '04899' '04901' '04905' '04908' '04909' '04910' '04911' '04912' '04914' '04915' '04919' '04920' '04922' '04923' '04928' '04930' '04931' '04932' '04934'

Test user IDs: '02463' '04201' '04202' '04213' '04217' '04221' '04225' '04273' '04285' '04297' '04300' '04301' '04311' '04312' '04314' '04319' '04322' '04324' '04334' '04338' '04341' '04343' '04344' '04347' '04350' '04372' '04385' '04388' '04404' '04423' '04427' '04444' '04449' '04451' '04456' '04459' '04461' '04463' '04470' '04473' '04488' '04493' '04507' '04509' '04513' '04519' '04531' '04537' '04556' '04557' '04559' '04580' '04585' '04595' '04600' '04612' '04621' '04631' '04641' '04662' '04664' '04667' '04673' '04675' '04681' '04682' '04683' '04689' '04693' '04697' '04705' '04708' '04709' '04711' '04714' '04719' '04722' '04724' '04726' '04727' '04728' '04730' '04731' '04732' '04733' '04743' '04746' '04749' '04754' '04760' '04762' '04767' '04770' '04775' '04784' '04786' '04796' '04798' '04806' '04810' '04811' '04812' '04821' '04822' '04823' '04827' '04829' '04830' '04833' '04838' '04842' '04848' '04849' '04851' '04853' '04855' '04856' '04860' '04862' '04865' '04868' '04874' '04875' '04881' '04885' '04887' '04889' '04893' '04895' '04896' '04900' '04902' '04903' '04904' '04906' '04907' '04913' '04916' '04917' '04918' '04921' '04924' '04925' '04926' '04927' '04929' '04933' '04935' '04936'

It is also mentioned in the main paper that the images for the test users are then randomly divided into query and gallery sets. We provide the images in the query and gallery sets in query_set.txt and gallery_set.txt, respectively. These files are provided here. We also provide a readme file (README.txt) for the readers’ convenience, wherein we provide information about the left/right labels.

A2. Training details for EyePAD, EyePAD++

We train all the models in our work with a batch size of 64 in our experiments, for 100 epochs. We use data augmentation such as random horizontal flip, random rotation (30 degrees) and random jitter. The detailed hyperparameter information for training models for user-to-user verification with PAD is provided in Table A3.

While training Densenet121 network for eye-to-eye verification with PAD, we use $\lambda_1 = 2.0$ (EyePAD) and $\lambda_2 = 2.0$ (EyePAD++). All the other parameters used in this experiment are same as those mentioned in the first row of Table A3.

A3. Detailed results

A3.1. EA and PAD with HRnet64

In Table A1, we provide the full version of Table 4 from the main paper, where we present results with HRnet64 [44]. Here, we report the user-to-user verification performance
with $K = 1, 2, 5$ gallery pairs. In most of the scenarios, **EyePAD++ outperforms the existing baselines in terms of the OFRR score.**

**A3.2. EA and PAD with MobilenetV3**

In Table A2, we provide the full version of Table 5 from the main paper, where we present results with MobilenetV3 [19]. Here, we report the user-to-user verification performance with $K = 1, 2, 5$ gallery pairs. **EyePAD++ outperforms the existing baselines in terms of the OFRR score,** in all the problem settings.

**A4. Ablation study: Effect of $\lambda_1$ in EyePAD**

The hyperparameter $\lambda_1$ is used to weight the feature-level distillation loss $L_{dis}$ (Eq 2 of the main paper). $L_{dis}$ is used to preserve the EA information, while student $M_s$ (initialized with $M_t$) is trained for PAD. Using Densenet121 and the original (clean) EA and PAD datasets, we analyze the effect of $\lambda_1$ on user-to-user verification perfor-
Figure A1. PAD performance (TDR@FDR=0.002) vs User-to-user verification performance (TAR@FAR=10\(^{-3}\)) obtained by the EyePAD student network, for different values of \(\lambda_1\).

A5. Training details for baselines

For training the MTL baseline for user-to-user or eye-to-eye verification with PAD, we use the same parameter values mentioned in Table A3, except for \(\lambda_1, \lambda_2\). The hyperparameters used for training MTMT [26] are provided in Table A4. While training MTMT [26] for eye-to-eye verification with PAD (using Densenet121 and the original dataset), we use \(\lambda_{auth} = 1.0\) and \(\lambda_{pad} = 1.0\). The rest of the hyperparameters are same as those mentioned in the first row of Table A4.