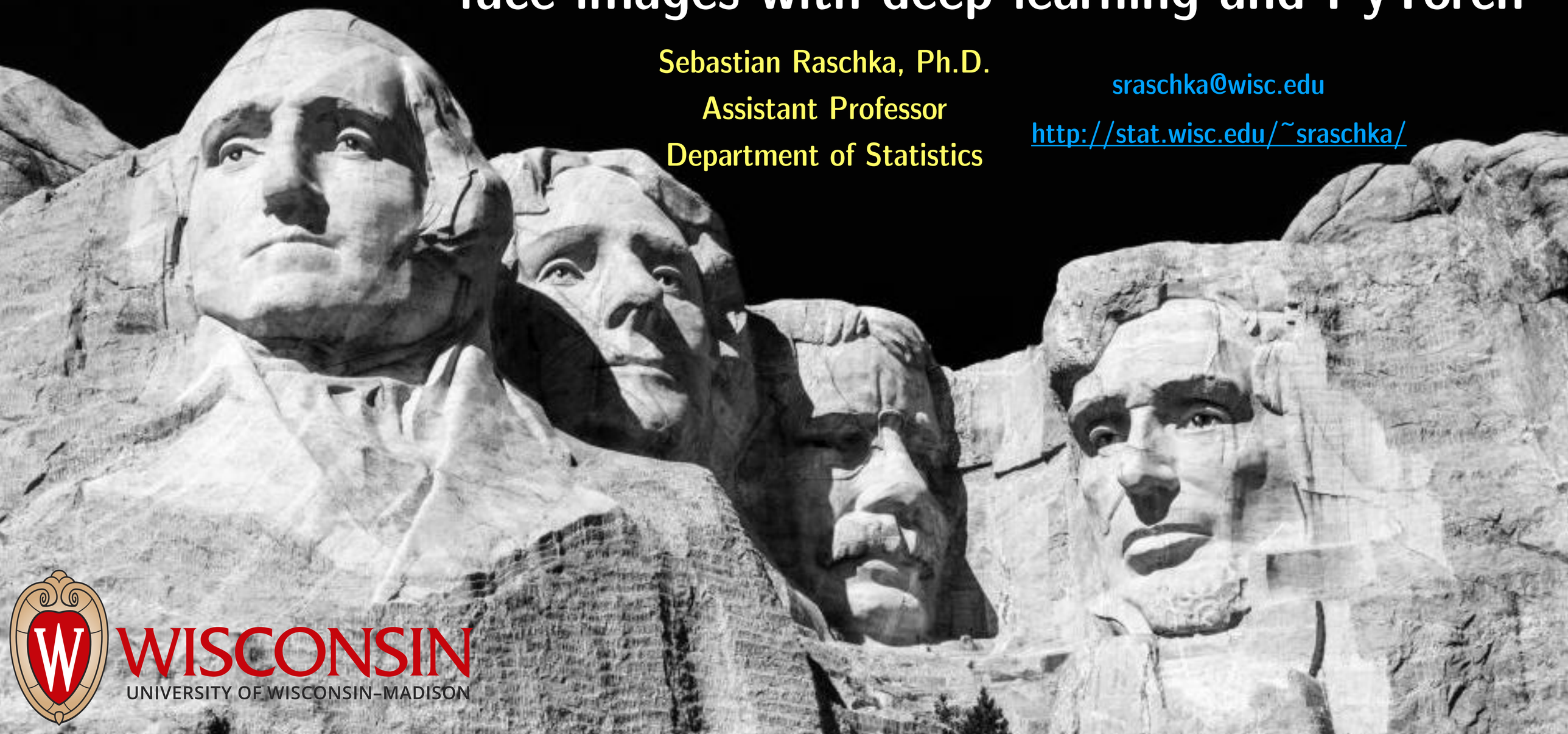
UW-Madison Data Science Club

October 10, 2019

Predicting soft-biometric attributes from face images with deep learning and PyTorch

Sebastian Raschka, Ph.D.
Assistant Professor
Department of Statistics

sraschka@wisc.edu
http://stat.wisc.edu/~sraschka/

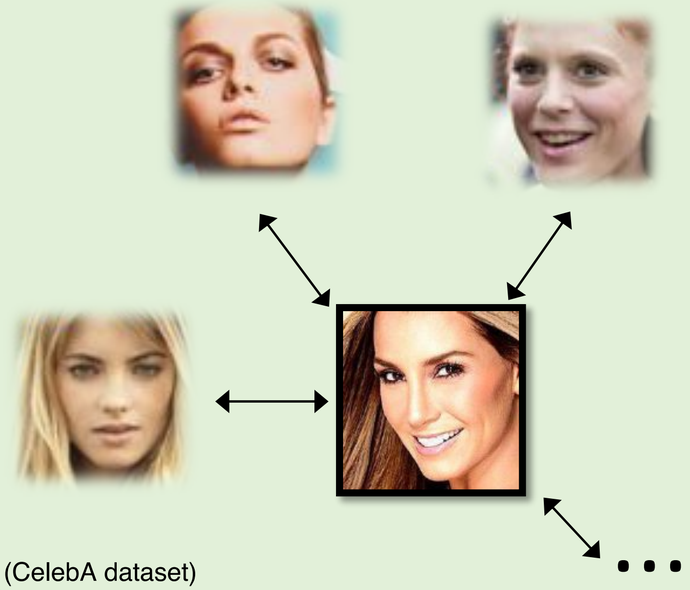


WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Biometric (Face) Recognition

A. Identification

Determine identity of an unknown person
1-to-*n* matching



(CelebA dataset)

B. Verification

Verify claimed identity of a person
1-to-1 matching

(MUCT dataset)

Applications of Biometric (Face) Recognition

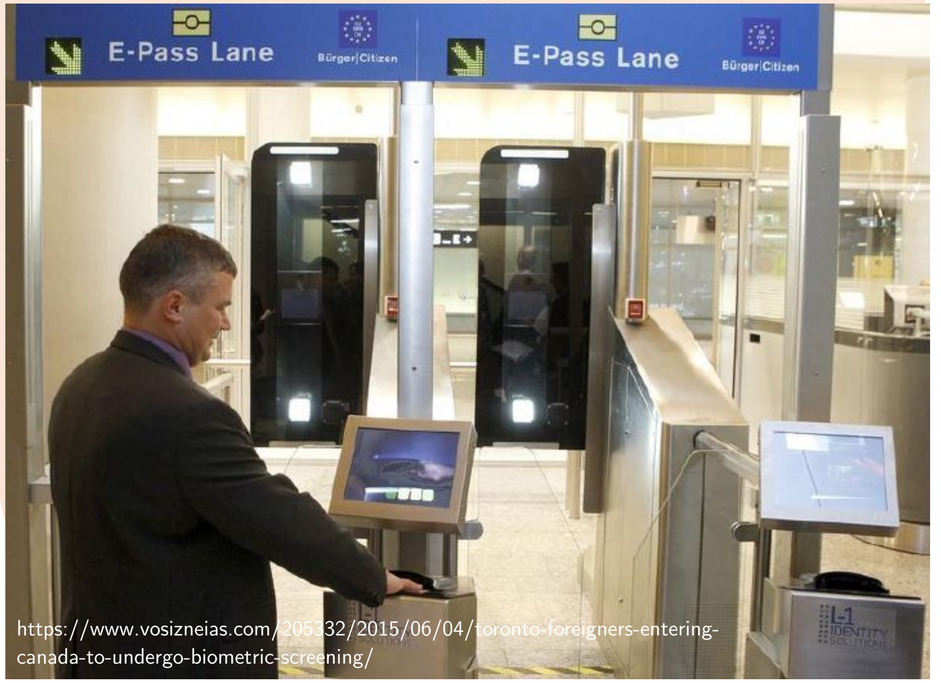
A. Identification

Determine identity of an unknown person
1-to-*n* matching

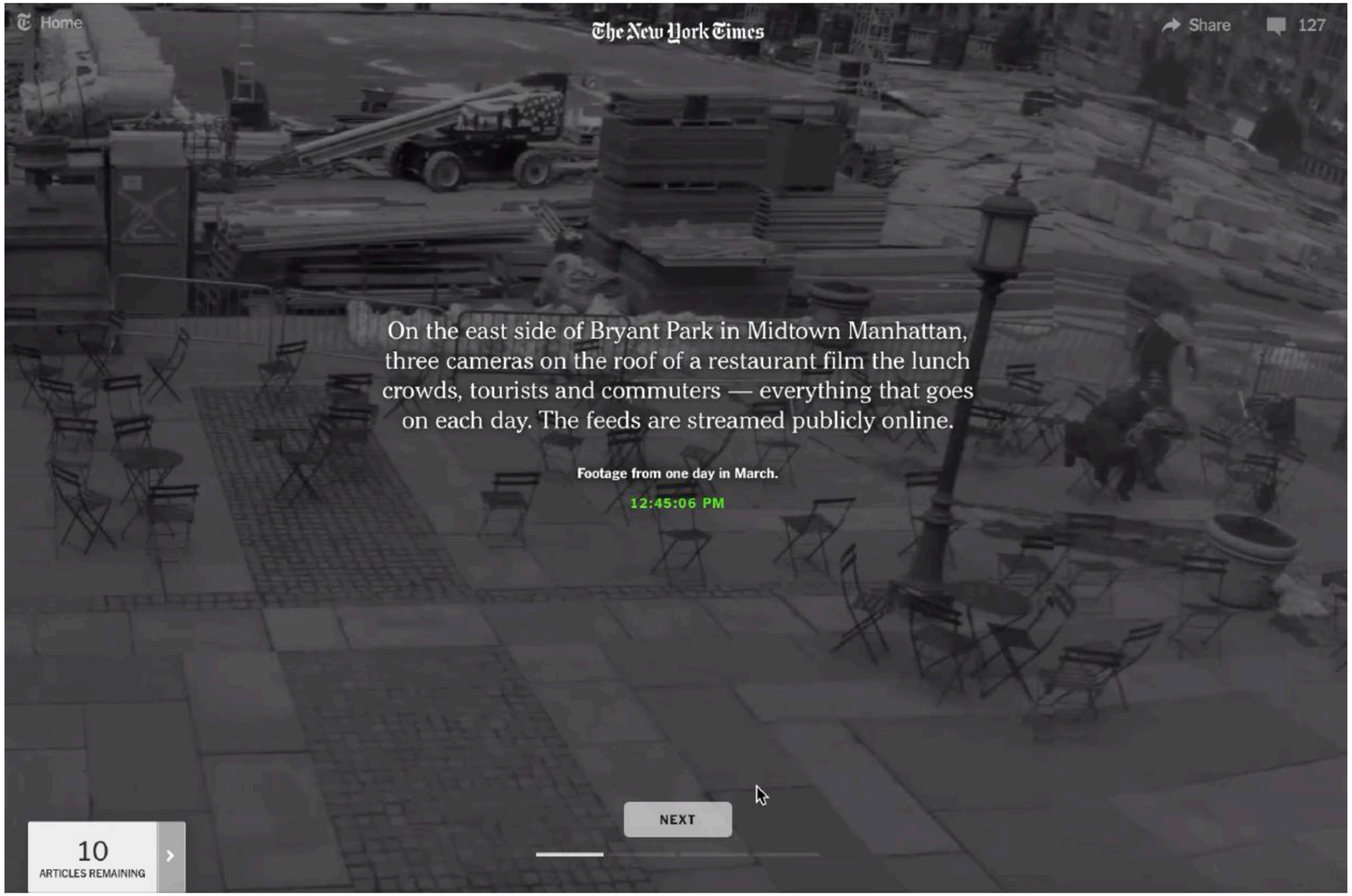
https://www.zdnet.com/article/too-many-false-alarms-for-population-wide-facial-surveillance-nec/

B. Verification

Verify claimed identity of a person
1-to-1 matching

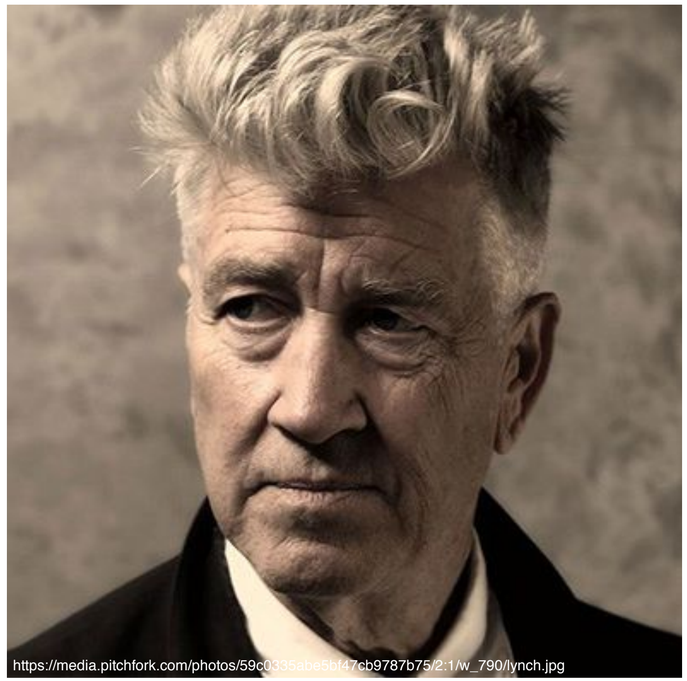


https://www.vosizneias.com/205132/2015/06/04/toronto-foreigners-entering-canada-to-undergo-biometric-screening/



https://www.nytimes.com/interactive/2019/04/16/opinion/facial-recognition-new-york-city.html

Beyond/In Addition to Traits for Biometric Recognition: Soft Biometric Attributes



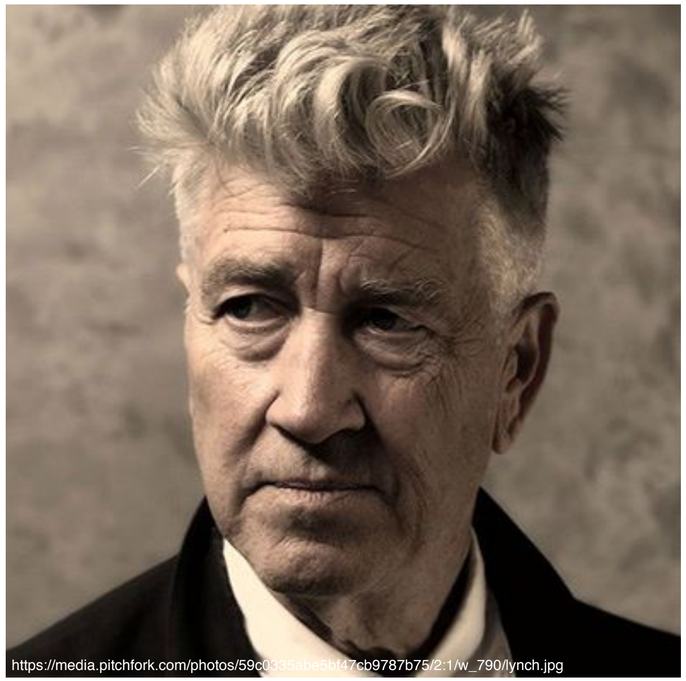
Identity	John Doe

Gender	Male
Age	65
Race	Caucasian
Medical	Healthy

SOFT BIOMETRIC ATTRIBUTES

Part I: Extracting Soft-Biometric Attributes from Face Images

Predicting Gender (... A Simple Task)

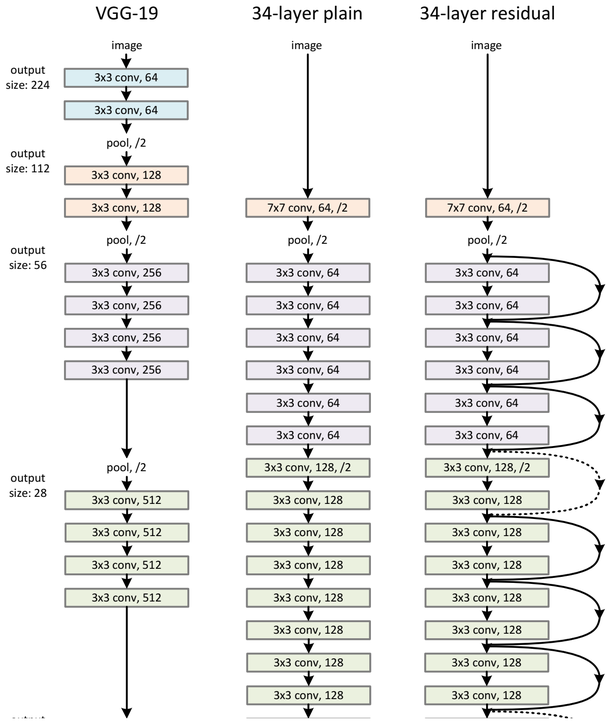


Identity	John Doe

Gender	Male
Age	65
Race	Caucasian
Medical	Healthy

SOFT BIOMETRIC ATTRIBUTES

ResNet-101 Applied to Gender Classification

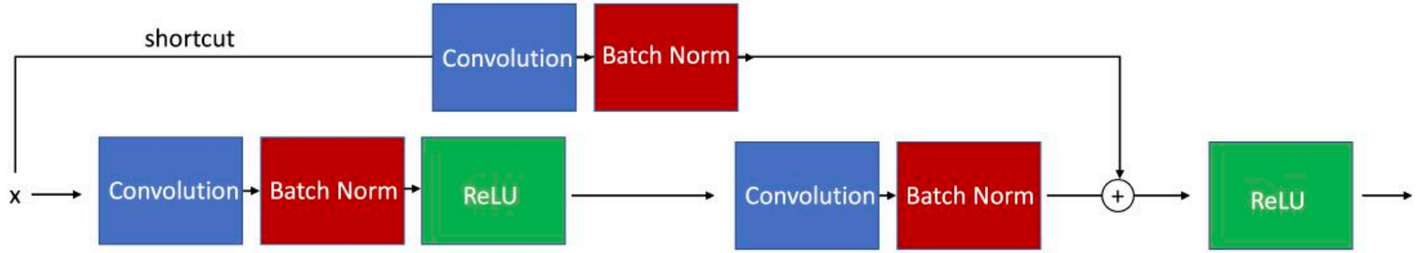


He, Kaiming, et al. "Deep residual learning for im recognition." *Proceedings of the IEEE conference computer vision and pattern recognition*. 2016.

+

Large-scale CelebFaces Attributes (CelebA) Dataset

http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html




```
Epoch: 010/010 | Batch 1250/1272 | Cost: 0.0127
Epoch: 010/010 | Train: 99.183% | Valid: 98.017%
Time elapsed: 121.09 min
Total Training Time: 121.09 min
```

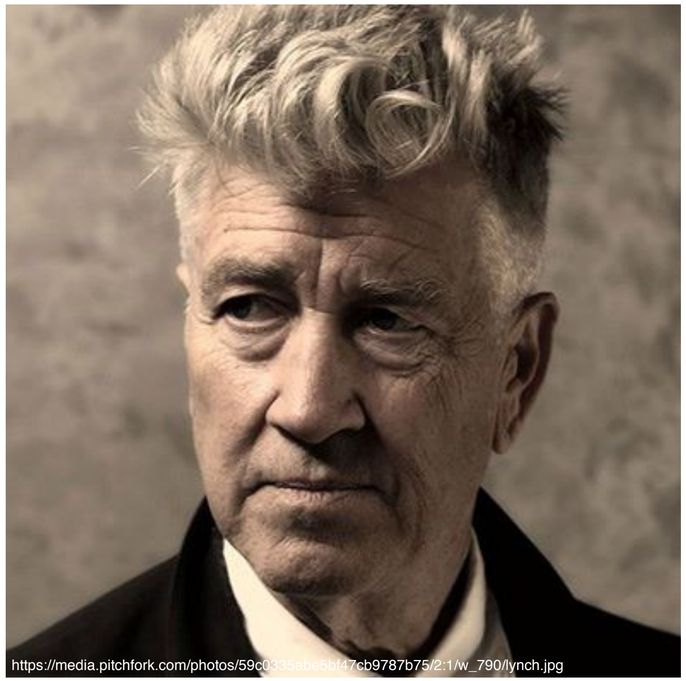
Evaluation

```
In [16]: with torch.set_grad_enabled(False): # save memory during inference
             print('Test accuracy: %.2f%%' % (compute_accuracy(model, test_loader, d
evice=DEVICE)))
```

```
Test accuracy: 97.52%
```

PyTorch code available at:
https://github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/cnn/cnn-resnet101-celeba.ipynb

Estimating the Apparent Age (... A Bit More Difficult)

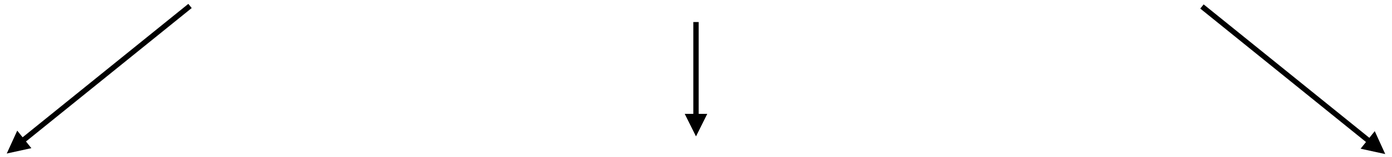


Identity	John Doe
Gender	Male
Age	65
Race	Caucasian
Medical	Healthy

SOFT BIOMETRIC ATTRIBUTES

Types of Labels in Supervised Learning Tasks

Color	Size	Price
green	M	10.1
red	L	13.5
blue	XXL	15.3



Nominal type
Task: classification

Ordinal type
Task: ordinal regression

Continuous
Task: metric regression

Ordinal Regression

Ordinal regression, also called *ordinal classification* or *ranking*
(although ranking is a bit different)

Order dependence like in metric regression,
but no metric distance

discrete values like in classification,
but order dependence/information

$$r_K \succ r_{K-1} \succ \ldots \succ r_1$$

E.g., movie ratings: *great* $\succ$ *good* $\succ$ *okay* $\succ$ *for genre fans* $\succ$ *bad*

★★★★★ ★★★★ ★★★ ★★ ★

Supervised Learning: Ordinal Regression

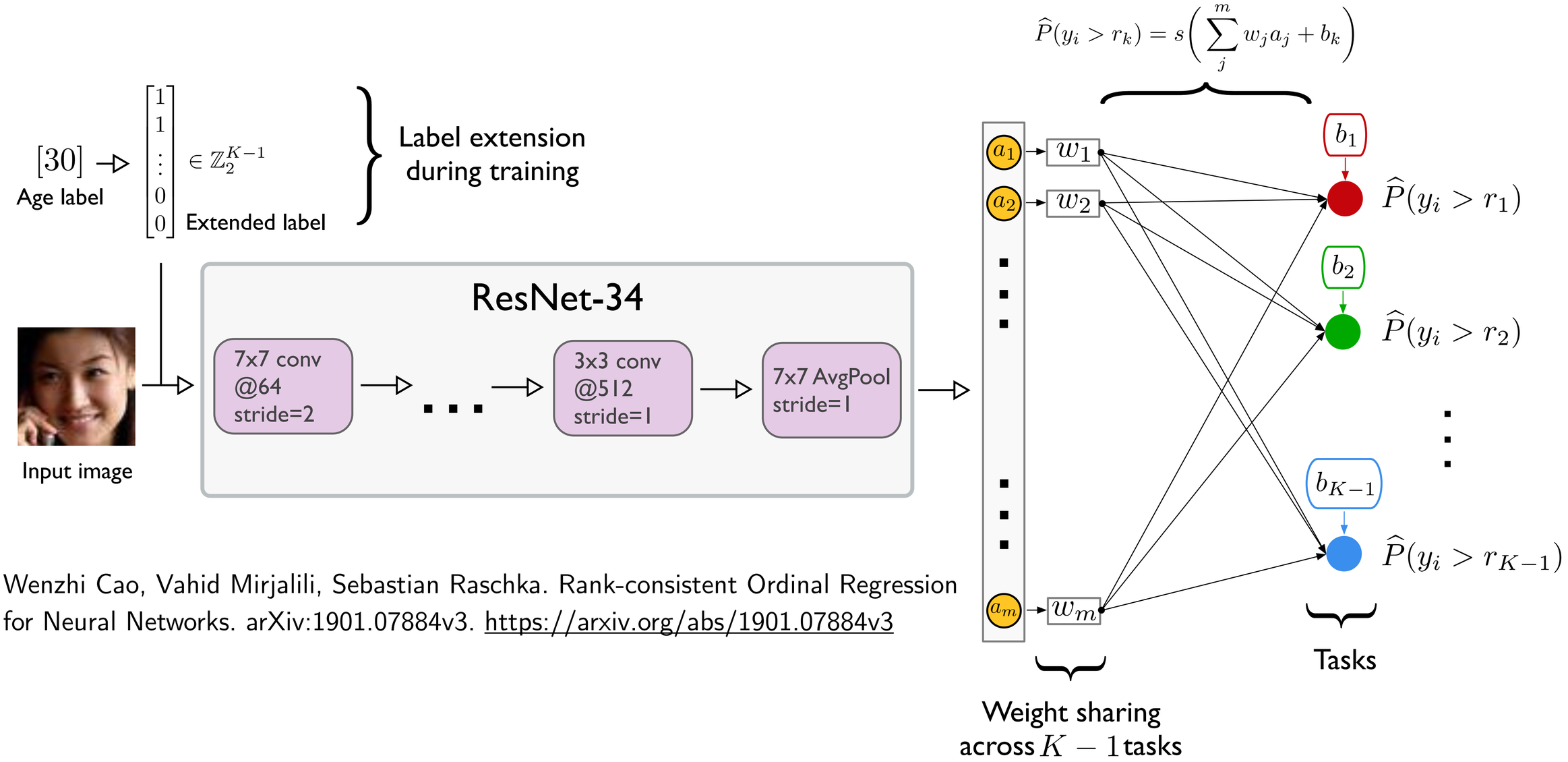
- **Ranking:** Correct order matters
  (0 loss if order is correct, e.g., rank a collection of movies by "goodness")

- **Ordinal Regression:** Correct label matters (as well)
  (E.g., age of a person in years; here, regard aging as a non-stationary process)

Age: 41 ≻ 29 ≻ 18

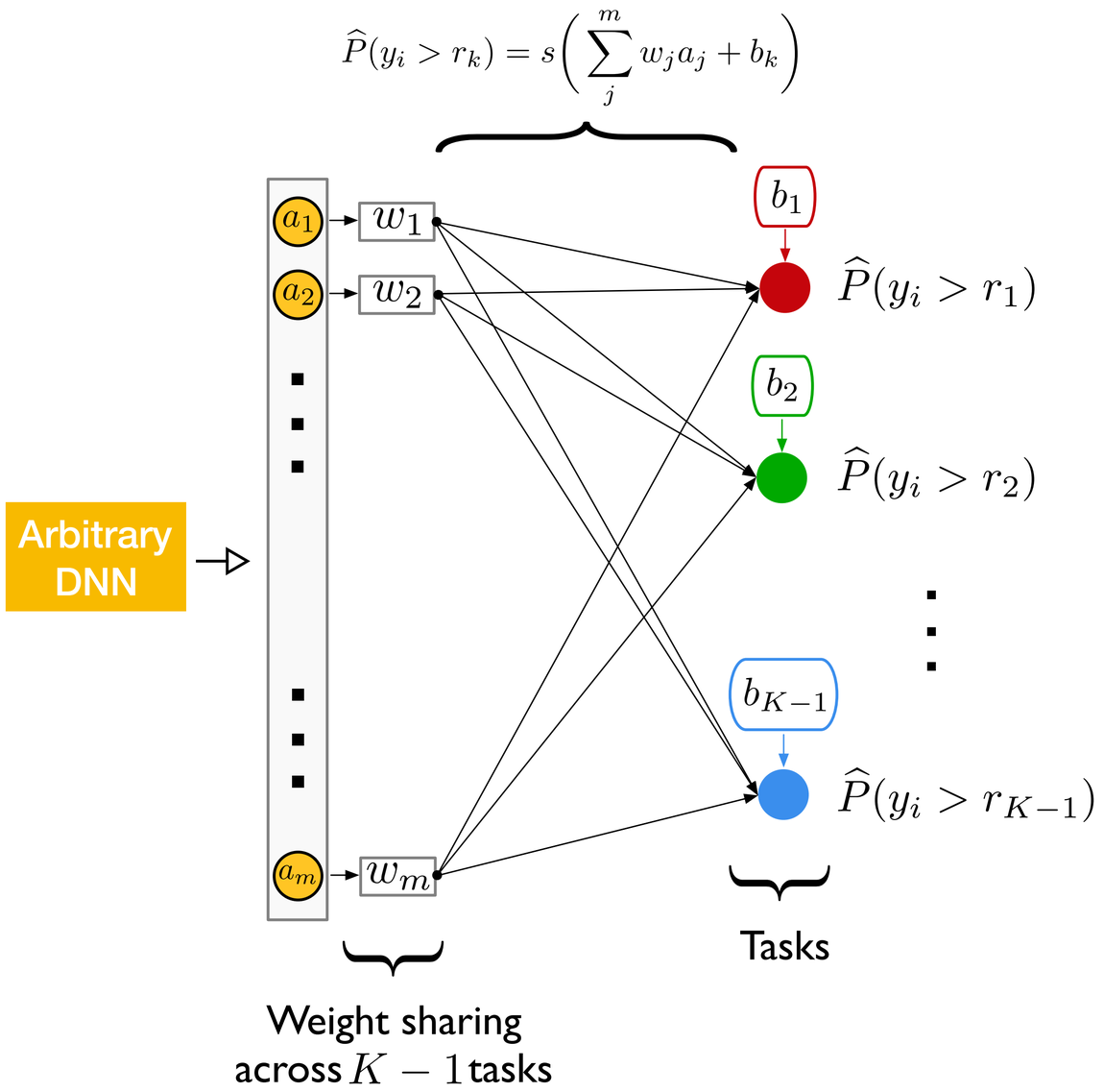
Excerpt from the UTKFace dataset
https://susanqq.github.io/UTKFace/

Consistent Rank Logits (CORAL) for Ordinal Regression with Convolutional Neural Networks Framework



Wenzhi Cao, Vahid Mirjalili, Sebastian Raschka. Rank-consistent Ordinal Regression for Neural Networks. arXiv:1901.07884v3. https://arxiv.org/abs/1901.07884v3

Predicting the Rank/Ordinal Label



Step 1: Converting the estimated probability of each task into a binary label (0/1)

$$f_k\left(\mathbf{x}_i\right) = 1\left\{\widehat{P}\left(y_i^{(k)} = 1\right) > 0.5\right\}$$

Step 2: Summing the K-1 binary labels

$$h\left(\mathbf{x}_i\right) = r_q$$

$$q = 1 + \sum_{k=1}^{K-1} f_k\left(\mathbf{x}_i\right)$$

Hypothesis: Rank Consistency Improves Predictive Performance

Desired property: probability estimates for the *K-1* tasks are decreasing

$$\widehat{P}\left(y_i^{(1)} = 1\right) \geq \widehat{P}\left(y_i^{(2)} = 1\right) \geq \ldots \geq \widehat{P}\left(y_i^{(K-1)} = 1\right) \quad \text{[Rank consistency]}$$

where the predicted empirical probability for task k is defined as

$$\widehat{P}\left(y_i^{(k)} = 1\right) = s\left(g\left(\mathbf{x}_i, \mathbf{W}\right) + b_k\right)$$

Hypothesis: Rank Consistency Improves Predictive Performance

Desired property: probability estimates for the *K-1* tasks are decreasing

$$\widehat{P}\left(y_i^{(1)} = 1\right) \geq \widehat{P}\left(y_i^{(2)} = 1\right) \geq \ldots \geq \widehat{P}\left(y_i^{(K-1)} = 1\right)$$

Rank Consistency can be satisfied by ordered bias units due to the weight constraint

$$b_1 \geq b_2 \geq \cdots \geq b_{K-1}$$

Theorem 1 (ordered biases)**.** *By minimizing loss function defined in Eq.* (4)*, the optimal solution* $(\mathbf{W}^*, \mathbf{b}^*)$ *satisfies* $b_1^* \geq b_2^* \geq \ldots \geq b_{K-1}^*$.

(Detailed proof provided in our paper)

Wenzhi Cao, Vahid Mirjalili, Sebastian Raschka. Rank-consistent Ordinal Regression for Neural Networks. arXiv:1901.07884v3. https://arxiv.org/abs/1901.07884v3

Niu, Z., Zhou, M., Wang, L., Gao, X., & Hua, G. (2016). Ordinal Regression with Multiple Output CNN for Age Estimation. CVPR.

Cao, W., Mirjalili V., Raschka S. (2019). Rank-consistent Ordinal Regression for Neural Networks. arXiv:1901.07884v3. https://arxiv.org/abs/1901.07884v3

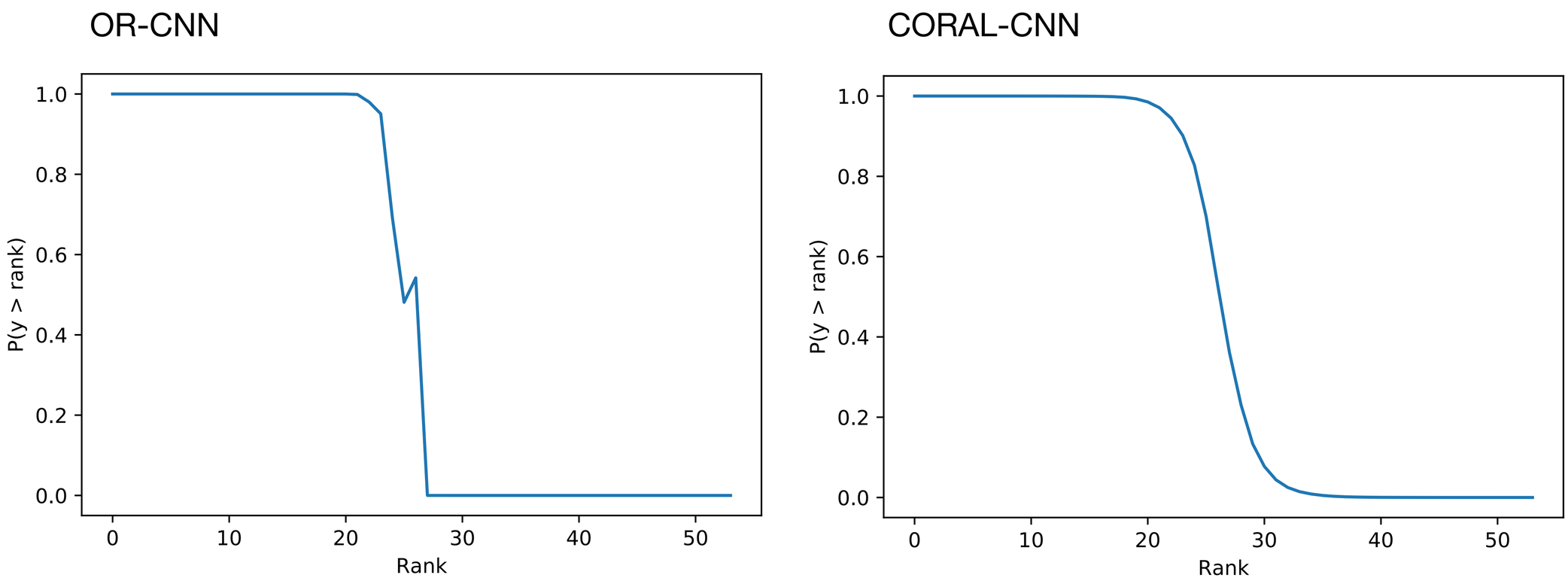


Figure S2: Plots show graphs of the predicted probabilities for each binary classifier task on one test data point in the MORPH dataset by OR-CNN (left subpanel) and CORAL-CNN (right subpanel). In this example, the ordinal regression CNN has an inconsistency at rank 26. The CORAL-CNN does not suffer from inconsistencies such that the rank prediction is a cumulative distribution function.

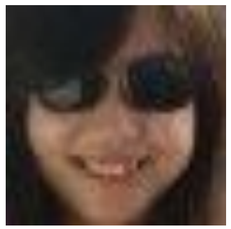
Datasets

MORPH-2

- 55,608 face images
- age range: 16-70 years

AFAD

- 165,501 face images
- age range: 15-40 years



CACD

- 159,449 face images
- age range: 14-62 years

UTKFace

- 16,434 face images
- age range: 21-60 years

Test Results

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - h(\mathbf{x}_i)\right| \qquad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - h(\mathbf{x}_i)\right)^2}$$

Table 1. Age prediction errors on the test sets *without* task importance weighting. All models are based on the ResNet-34 architecture.

Method	Random Seed	MORPH-2		AFAD		UTKFace		CACD	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CE-CNN	0	3.40	4.88	3.98	5.55	6.57	9.16	6.18	8.86
	1	3.39	4.87	4.00	5.57	6.24	8.69	6.10	8.79
	2	3.37	4.87	3.96	5.50	6.29	8.78	6.13	8.87
	AVG ± SD	3.39 ± 0.02	4.89 ± 0.01	3.98 ± 0.02	5.54 ± 0.04	6.37 ± 0.18	8.88 ± 0.25	6.14 ± 0.04	8.84 ± 0.04
OR-CNN (Niu et al., 2016)	0	2.98	4.26	3.66	5.10	5.71	8.11	5.53	7.91
	1	2.98	4.26	3.69	5.13	5.80	8.12	5.53	7.98
	2	2.96	4.20	3.68	5.14	5.71	8.11	5.49	7.89
	AVG ± SD	2.97 ± 0.01	4.24 ± 0.03	3.68 ± 0.02	5.13 ± 0.02	5.74 ± 0.05	8.08 ± 0.06	5.52 ± 0.02	7.93 ± 0.05
CORAL-CNN (ours)	0	2.68	3.75	3.49	4.82	5.46	7.61	5.56	7.80
	1	2.63	3.66	3.46	4.83	5.46	7.63	5.37	7.64
	2	2.61	3.64	3.52	4.91	5.48	7.63	5.25	7.53
	AVG ± SD	**2.64 ± 0.04**	**3.68 ± 0.06**	**3.49 ± 0.03**	**4.85 ± 0.05**	**5.47 ± 0.01**	**7.62 ± 0.01**	**5.39 ± 0.16**	**7.66 ± 0.14**

MORPH-2

55,608 face images

age range: 16-70 years

AFAD

165,501 face images

age range: 15-40 years

UTKFace

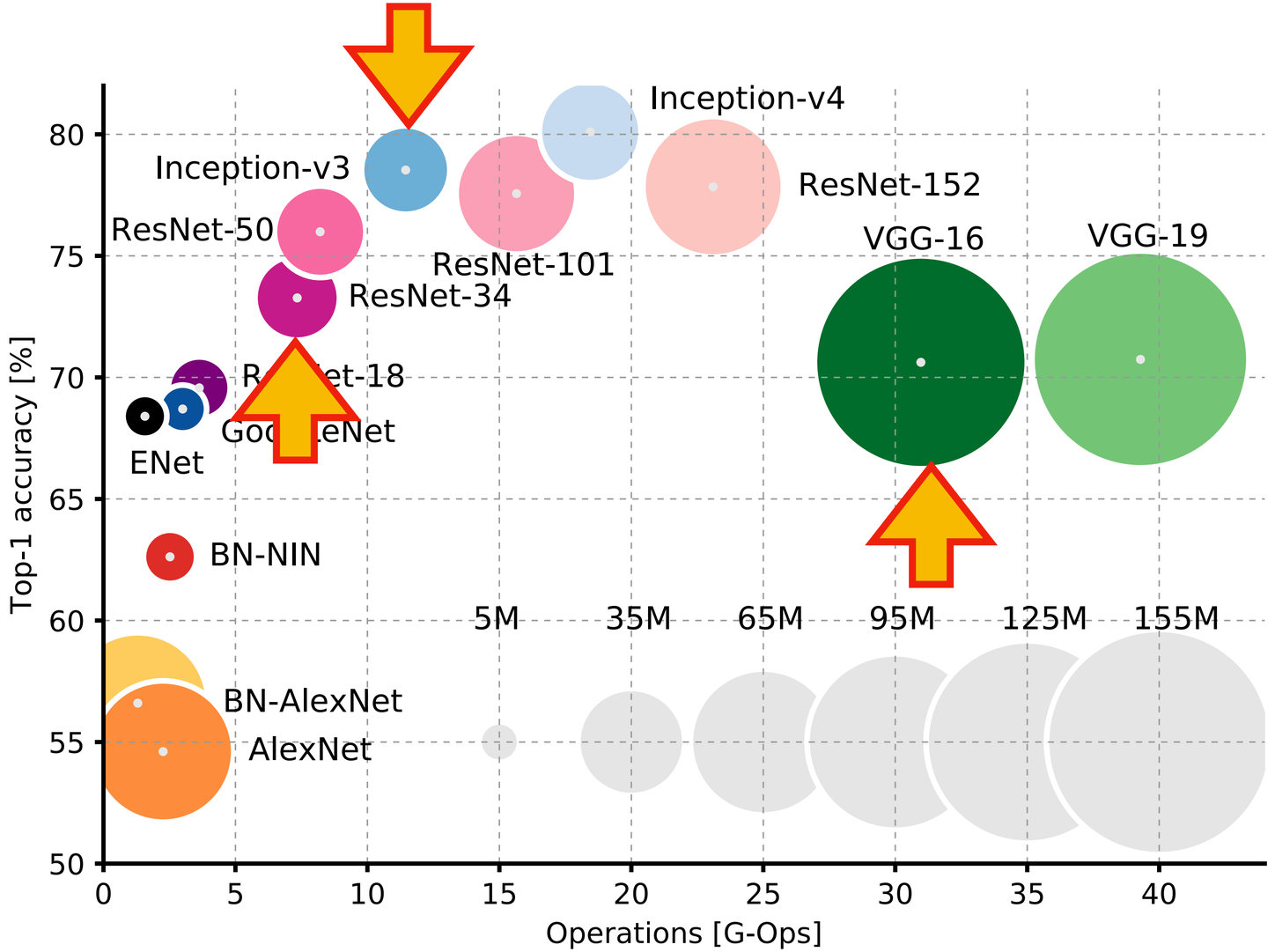
16,434 face images

age range: 21-60 years

CACD

159,449 face images

age range: 14-62 years



Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

Code Demos

Tutorial Overview

Intro: Different approaches for ordinal regression

Part 1: Warm-up
- Implementing a nominal classifier in PyTorch for age prediction

Part 2: Dataset
- Preparing a custom dataset for model training in PyTorch
- MNIST DataLoader example
- UTKFace image dataset for age prediction

Part 3: Nominal Classifier
- Implementing a convolutional neural network classifier for age prediction (VGG-16 with cross entropy loss trained on UTKFace)

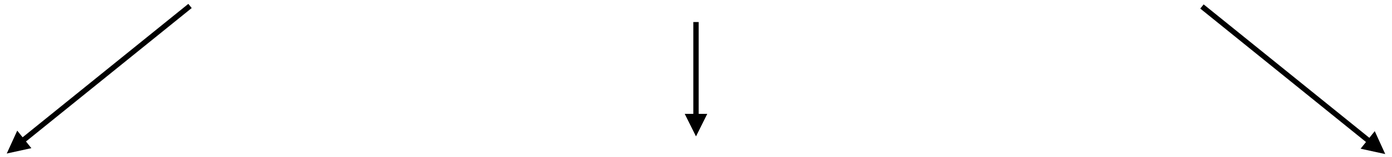
Tutorial Overview

Part 4: Extended Binary Classifier

- Ordinal regression via extended binary classification for age prediction
- Implementing Niu et al.'s ordinal regression approach
- Implementing the CORAL (Consistent Rank Logits) approach

Types of Labels in Supervised Learning Tasks

Color	Size	Price
green	M	10.1
red	L	13.5
blue	XXL	15.3



Nominal type
Task: classification

Ordinal type
Task: ordinal regression

Continuous
Task: metric regression

Supervised Learning: Ordinal Regression

Ordinal regression, also called *ordinal classification* or *ranking*
(although ranking is a bit different)

Order dependence like in metric regression,
but no metric distance

discrete values like in classification,
but order dependence

$$r_K \succ r_{K-1} \succ \ldots \succ r_1$$

E.g., movie ratings: *great* $\succ$ *good* $\succ$ *okay* $\succ$ *for genre fans* $\succ$ *bad*

Supervised Learning: Ordinal Regression

- Ranking: Correct order matters
  (0 loss if order is correct, e.g., rank a collection of movies by "goodness")

- Ordinal Regression: Correct label matters (as well)
  (E.g., age of a person in years; here, regard aging as a non-stationary process)

18 ≻ 29 ≻ 41

Excerpt from the UTKFace dataset
https://susanqq.github.io/UTKFace/

We will work with this dataset
in the hands-on tutorial this afternoon!

Ordinal Regression Categories

Ordinal regression approaches can be grouped into 3 categories:

1. Basic/naive
2. Binary decomposition
3. Threshold approaches

Ordinal Regression Categories

Ordinal regression approaches can be grouped into 3 categories:

1. Basic/naive

- Nominal classification (e.g., cross-entropy loss)
- Metric regression (+ conversion to integer/rounding)

Ordinal Regression Categories

Ordinal regression approaches can be grouped into 3 categories:

2. Binary decomposition

- Convert target labels to binary labels
- Then, use single or multiple models on binary labels

Ordinal Regression Categories

Ordinal regression approaches can be grouped into 3 categories:

3. Threshold approaches

- Learn function mapping labels to Real
- Categories are thresholds/intervals on the Real number line

Part 1: Warm-up

Implementing a nominal classifier in PyTorch for age prediction

Ordinal Regression Categories

Ordinal regression approaches can be grouped into 3 categories:

1. Basic/naive

- Nominal classification (e.g., cross-entropy loss)
- Metric regression (+ conversion to integer/rounding)

Regression loss:

$$\text{MSE} = \frac{1}{n}\sum_i (a^{[i]} - y^{[i]})^2$$

last-layer activation/ predicted label

true label

Multi-class classification loss (cross-entropy):

$$\mathcal{H} = \sum_{i=1}^{n}\sum_{j=1}^{k} -y_j^{[i]} \log\left(a_j^{[i]}\right)$$

Class-Membership Probabilities via Softmax

$$a^{(i)} = P(y = t \mid z_t^{(i)}) = \sigma_{\text{softmax}}(z_t^{(i)}) = \frac{e^{z_t^{(i)}}}{\sum_{j=1}^{k} e^{z_j^{(i)}}}$$

$$t \in \{j \ldots k\}$$

k is the number of class labels

(Basically, softmax is just an exponential function that normalizes the activations so that they sum up to 1)

Loss Functions for Logistic Regression & Multi-layer Neural Network Classifiers

$$\mathcal{L}_{\text{binary}} = -\sum_{i=1}^{n} \Big(y^{[i]} \log(a^{[i]}) + (1 - y^{[i]}) \log(1 - a^{[i]}) \Big)$$

Negative log-likelihood loss / binary Cross Entropy

$$\mathcal{L}_{\text{multiclass}} = \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{[i]} \log \left(a_j^{[i]} \right)$$

(Multi-category) Cross Entropy for k different class labels

This one assumes one-hot encoded labels!

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

(4 training examples, 3 classes)

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

1 training example

$$\mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

(4 training examples, 3 classes)

$$\mathcal{L}_{\text{multiclass}} = \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{[i]} \log\left(a_j^{[i]}\right)$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[2]} &= [(-0) \cdot \log(0.3072)] \\ &\quad + [(-1) \cdot \log(0.4147)] \\ &\quad + [(-0) \cdot \log(0.2780)] \\ &= 0.880200... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[3]} &= [(-0) \cdot \log(0.4263)] \\ &\quad + [(-0) \cdot \log(0.2248)] \\ &\quad + [(-1) \cdot \log(0.3490)] \\ &= 1.05268... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[4]} &= [(-0) \cdot \log(0.2668)] \\ &\quad + [(-0) \cdot \log(0.2978)] \\ &\quad + [(-1) \cdot \log(0.4354)] \\ &= 0.831490... \end{aligned}$$

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &\quad + [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[2]} &= [(-0) \cdot \log(0.3072)] \\ &\quad + [(-1) \cdot \log(0.4147)] \\ &\quad + [(-0) \cdot \log(0.2780)] \\ &= 0.880200... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[3]} &= [(-0) \cdot \log(0.4263)] \\ &\quad + [(-0) \cdot \log(0.2248)] \\ &\quad + [(-1) \cdot \log(0.3490)] \\ &= 1.05268... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[4]} &= [(-0) \cdot \log(0.2668)] \\ &\quad + [(-0) \cdot \log(0.2978)] \\ &\quad + [(-1) \cdot \log(0.4354)] \\ &= 0.831490... \end{aligned}$$

$$\rightarrow \mathcal{L}_{\text{multiclass}} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} -y_j^{[i]} \log\left(a_j^{[i]}\right)$$

$$\approx 0.9335$$

PyTorch already takes care of one-hot encoding

```
##########################
### MODEL
##########################

class LogisticRegression(torch.nn.Module):
    """Multinomial Logistic Regression model"""

    def __init__(self, num_features, num_classes):
        super(LogisticRegression, self).__init__()

        ### Output layer
        self.linear_out = torch.nn.Linear(NUM_FEATURES, NUM_CLASSES)

        # the following two lines are optional
        self.linear_out.weight.detach().normal_(0.0, 0.1)
        self.linear_out.bias.detach().zero_()

    def forward(self, x):
        logits = self.linear_out(x)
        probas = F.softmax(logits, dim=1)
        return logits, probas
```

Instead of doing things "manually" via F.nll_loss, use F.cross_entropy -- it is numerically more stable

```
[24]: print(targets)

tensor([6, 1, 7, 8, 3], device='cuda:0')

[25]: print(logits)

tensor([[-3.5890, -1.9838,  0.4137, -2.3949,  1.5565, -2.9962,  3.4829, -1.6488,
          0.9492,  1.2827],
        [-3.0150,  5.4310,  2.3293,  1.1424, -2.1697, -1.4378, -1.5242, -2.4102,
          2.0650, -2.0856],
        [-3.7686, -0.6936, -2.3394, -0.0395, -0.4688, -0.8471, -4.5556,  8.0915,
          0.5525,  3.4793],
        [-1.6363,  1.2846,  1.4050,  0.5723, -4.4089, -0.1598, -0.3495, -2.3774,
          5.4604, -1.4640],
        [ 0.9780, -2.3383,  0.4330,  6.4788, -4.3987,  3.1476, -0.0177, -4.1507,
          0.6837, -2.8333]], device='cuda:0', grad_fn=<SliceBackward>)

[26]: F.cross_entropy(logits, targets)

[26]: tensor(0.1065, device='cuda:0', grad_fn=<NllLossBackward>)

[27]: F.nll_loss(torch.log_softmax(logits, dim=1), targets)

[27]: tensor(0.1065, device='cuda:0', grad_fn=<NllLossBackward>)
```

PyTorch Loss-Input Cheatsheet

- `torch.nn.functional.binary_cross_entropy` takes logistic sigmoid values as inputs
- `torch.nn.functional.binary_cross_entropy_with_logits` takes logits as inputs
- `torch.nn.functional.cross_entropy` takes logits as inputs (performs `log_softmax` internally)
- `torch.nn.functional.nll_loss` is like `cross_entropy` but takes log-probabilities (log-softmax) values as inputs

Code Examples:

00_check-env.ipynb

01_logistic-regression-mnist-warmup.ipynb

Part 2: Dataset

Preparing a custom dataset for model training in PyTorch

Zhang, Z., Song, Y., & Qi, H. (2017). Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5810-5818).

https://susanqq.github.io/UTKFace/

Custom DataLoader based on MNIST:

02.1_dataloader-mnist.ipynb

Custom DataLoader for the UTKFace dataset (age prediction):

02.2_dataloader-UTKFace.ipynb

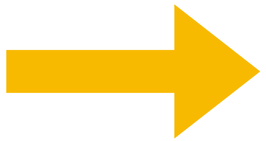
Part 3: Nominal Classifier

Implementing a convolutional neural network classifier for age prediction

Part 3: Nominal Classifier

Implementing a convolutional neural network classifier for age prediction

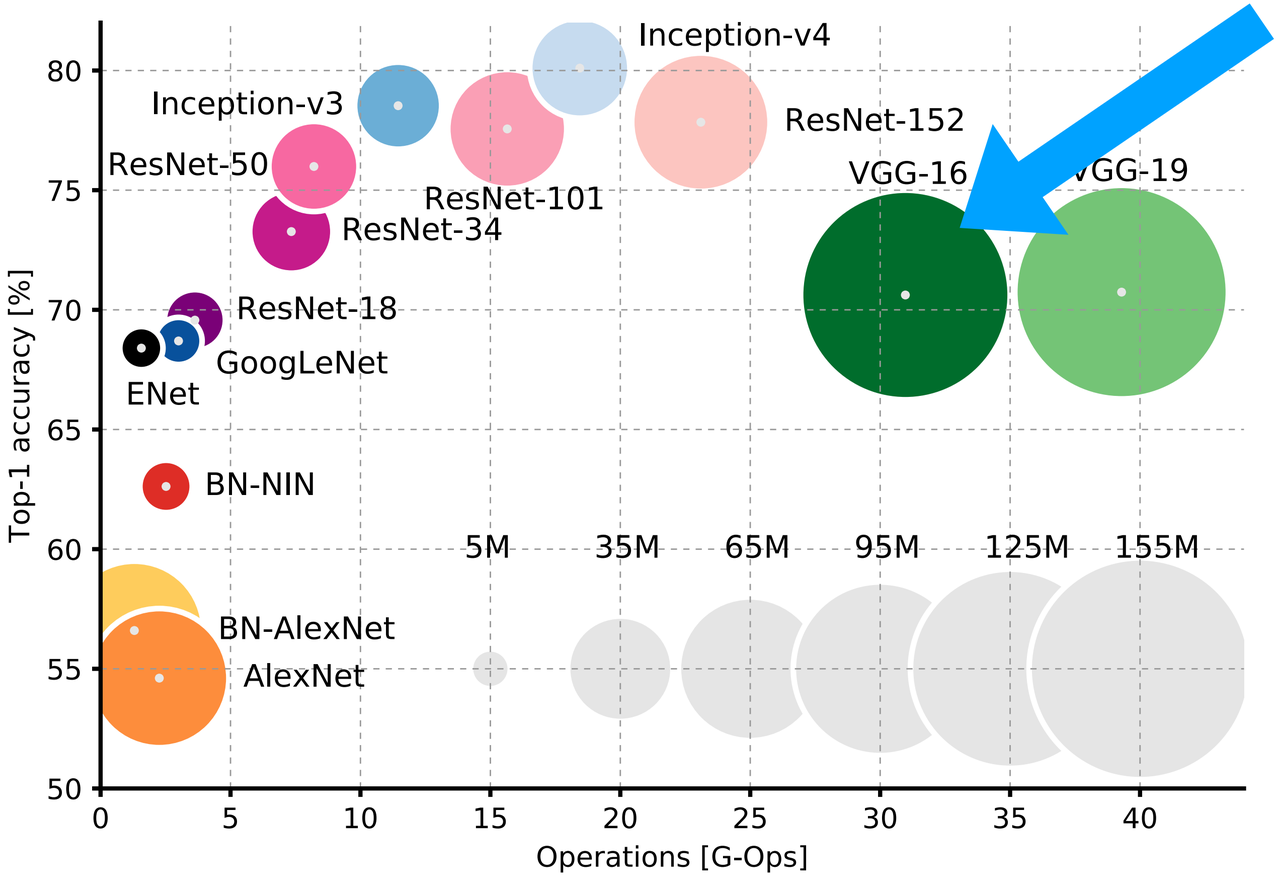
Ordinal regression approaches can be grouped into 3 categories:



1. Basic/naive

2. Binary decomposition

3. Threshold approaches



Model backbone for the rest of this tutorial

Canziani, Alfredo, Adam Paszke, and Eugenio Culurciello. "An analysis of deep neural network models for practical applications." *arXiv preprint arXiv:1605.07678* (2016).

VGG-16

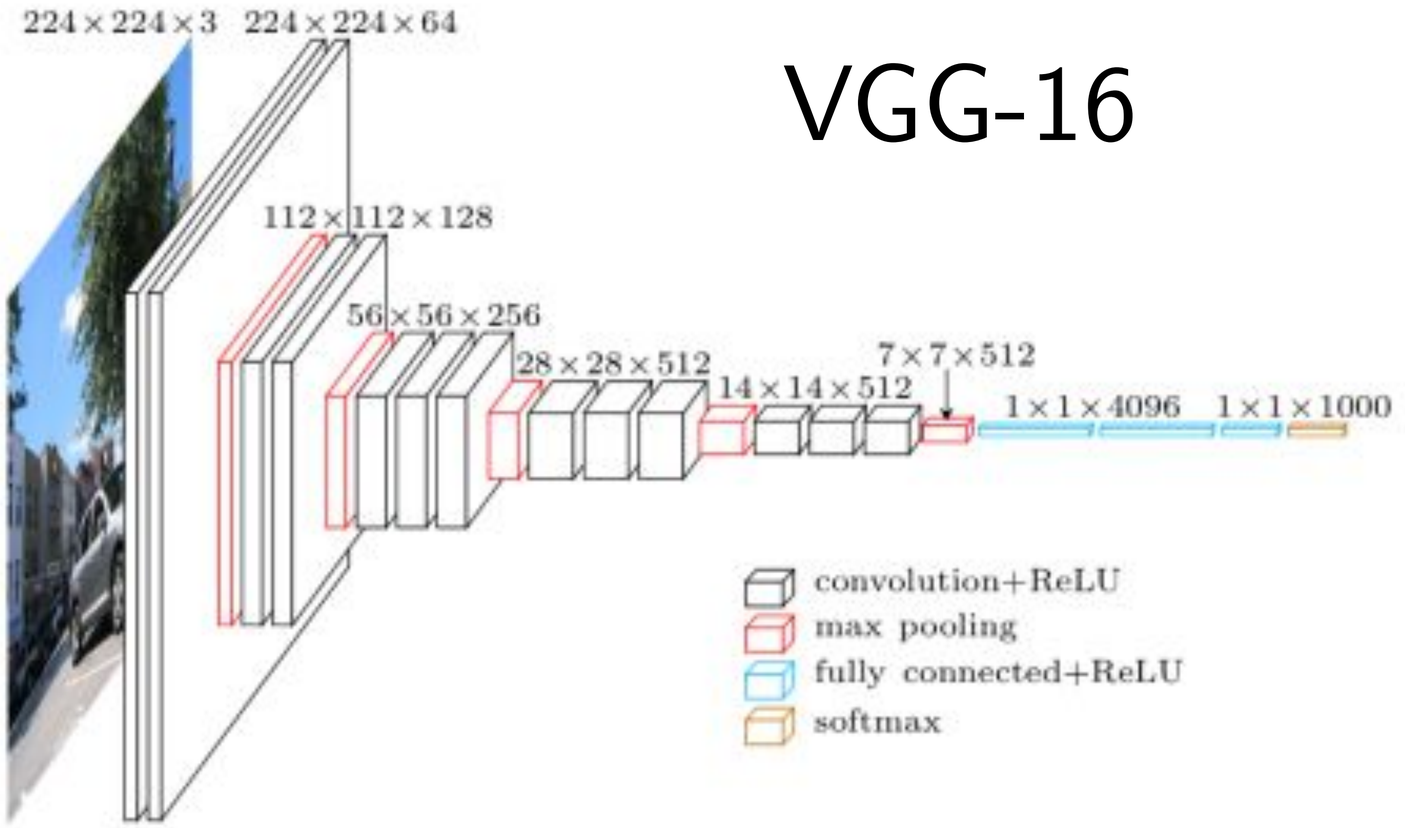


Image Source: http://www.cs.toronto.edu/~frossard/post/vgg16/

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

CNN (VGG-16) on UTKFace

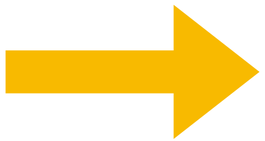
03.1_CrossEntropy.ipynb

Question: Why is Cross Entropy not a good choice for ordinal tasks?

Part 4: Extended Binary Classifier

Ordinal regression via extended binary classification for age prediction

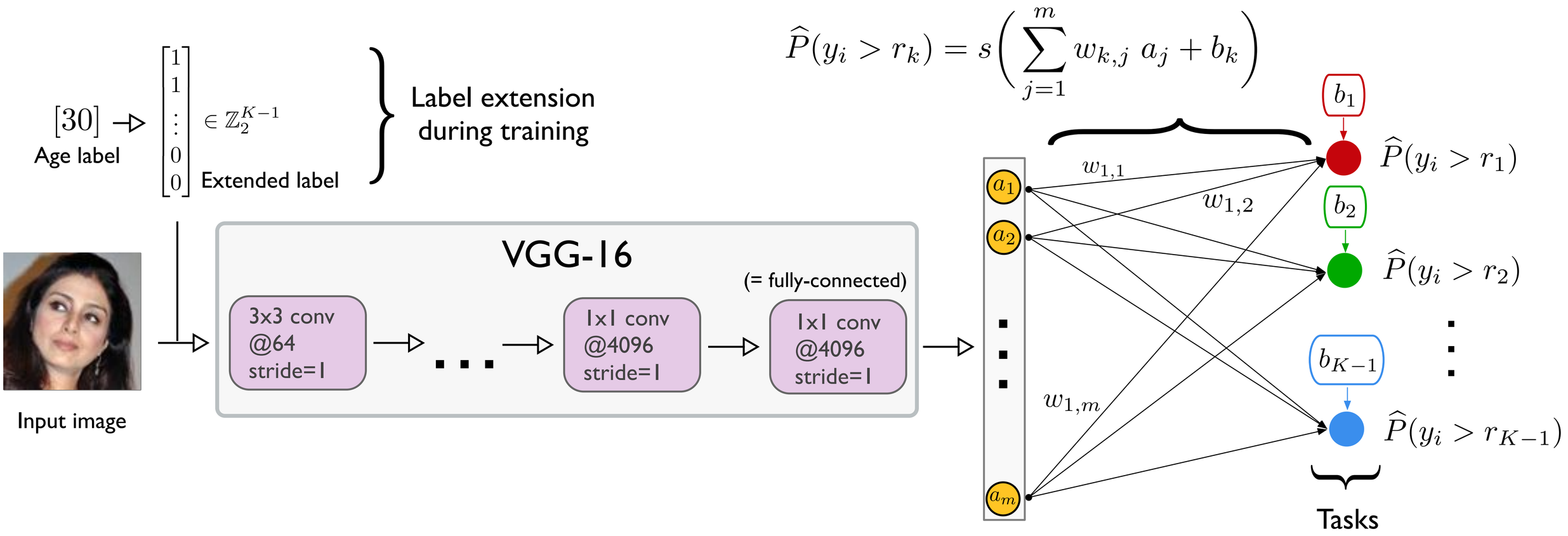
1. Basic/naive



2. Binary decomposition

3. Threshold approaches

Binary Decomposition



Binary Decomposition

the first class is $[0,\ldots,0]^\top$, the second class is $[1,\ldots,0]^\top$, the third class is $[1,1,\ldots,0]^\top$ and so on.

$$\widehat{P}(y_i > r_k) = s\left(\sum_{j=1}^{m} w_{k,j}\, a_j + b_k\right)$$

$[30]$ Age label

$$\begin{bmatrix}1\\1\\\vdots\\0\\0\end{bmatrix} \in \mathbb{Z}_2^{K-1}$$

Extended label

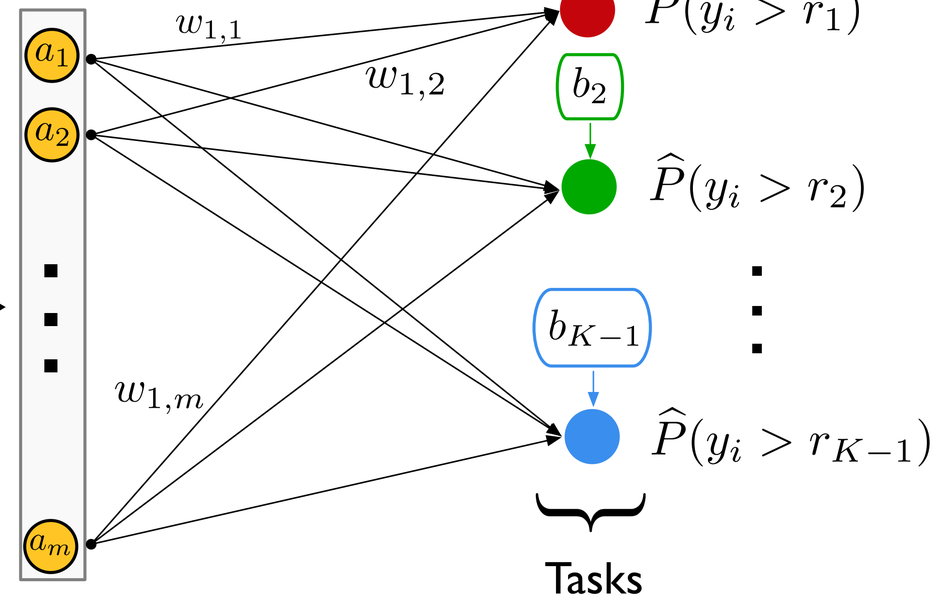
Label extension during training

Input image

VGG-16

(= fully-connected)

3x3 conv @64 stride=1 → . . . → 1x1 conv @4096 stride=1 → 1x1 conv @4096 stride=1



Age label

for each k:

$$1\{\widehat{P}\,(y_i > r_k)\} = f_k\,(\mathbf{x}_i) \in \{0, 1\}$$

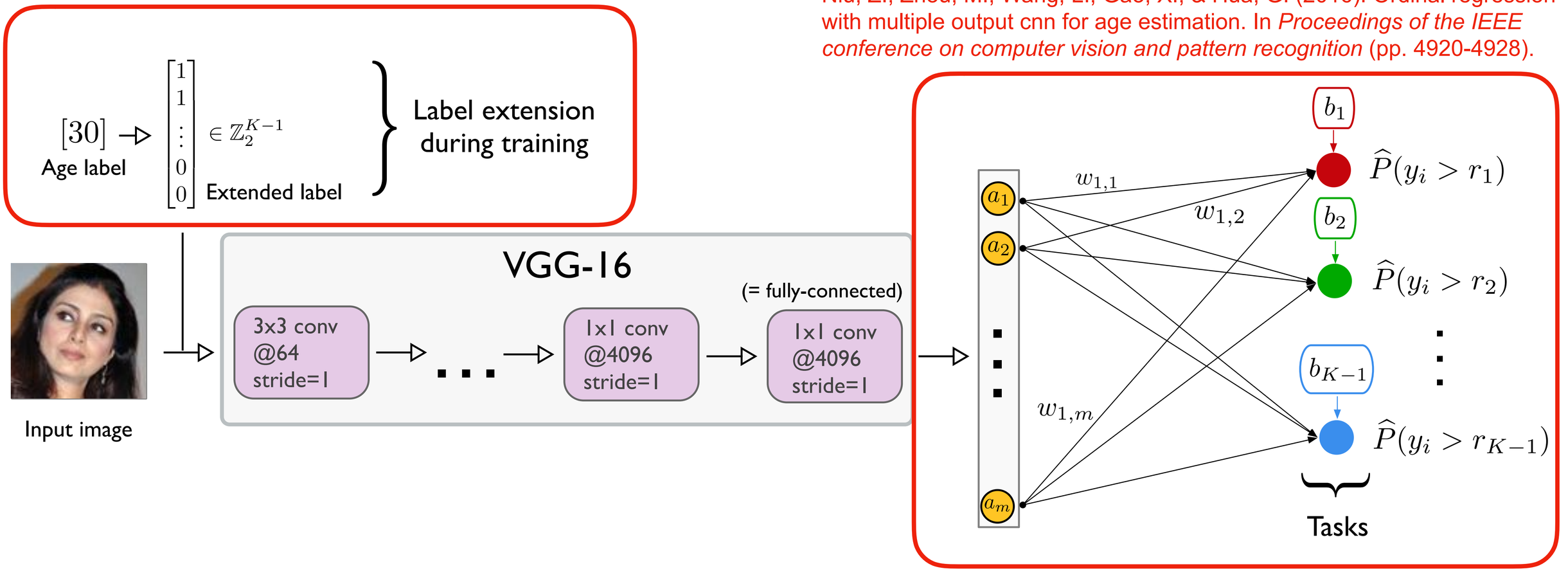
Then, compute rank label:

$$h\,(\mathbf{x}_i) = r_q, \quad q = 1 + \sum_{k=1}^{K-1} f_k\,(\mathbf{x}_i)$$

Binary Decomposition

Li, L., & Lin, H. T. (2007). Ordinal regression by extended binary classification. In *Advances in neural information processing systems* (pp. 865-872).

Niu, Z., Zhou, M., Wang, L., Gao, X., & Hua, G. (2016). Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4920-4928).



Implementing Niu et al.'s approach with a VGG-16 backbone

03.2_DataLoader-for-binary-tasks.ipynb

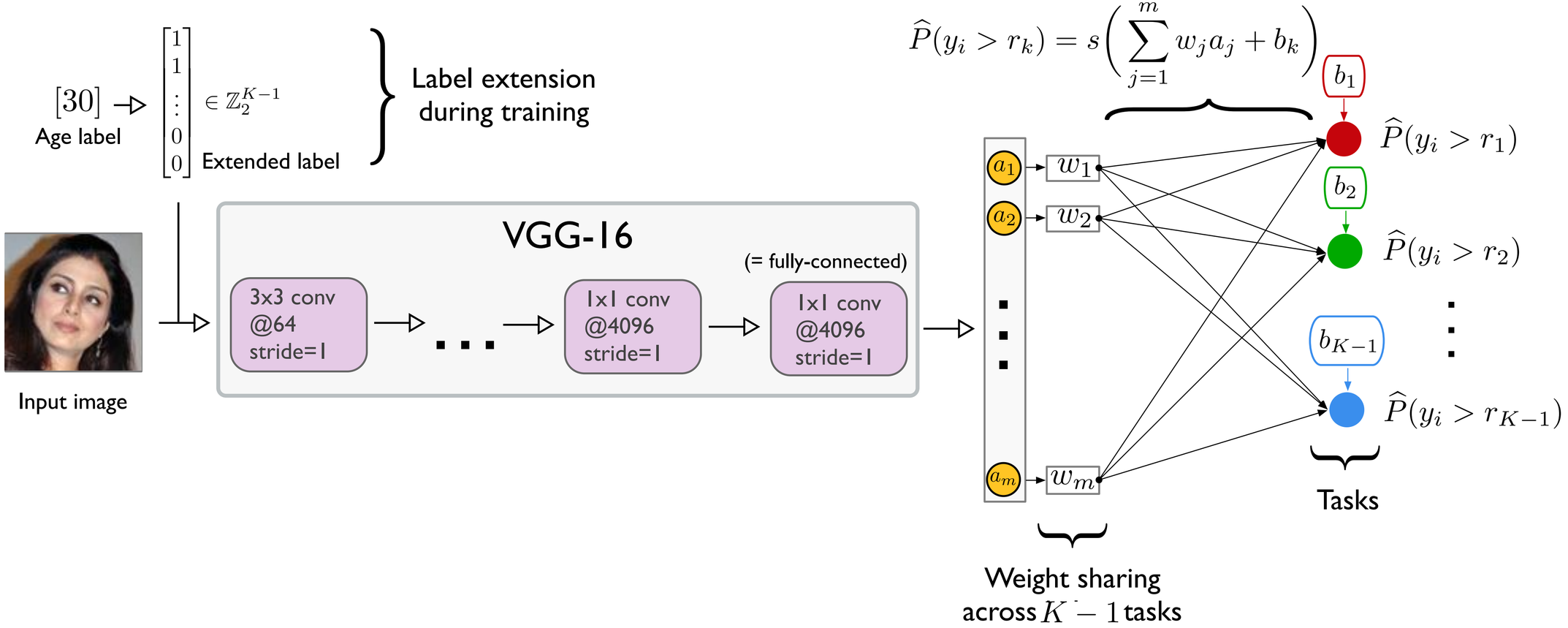
03.3_Niu-et-al.ipynb

Niu et al.'s approach does not guarantee rank consistency

For example, in an age estimation setting, it would be contradictory if the kth binary task predicted that the age of a person was larger than 30, but a previous task predicted it was smaller than 20, which is suboptimal when the $K - 1$ task predictions are combined to obtain the estimated age

CORAL (Consistent Rank Logits)

Wenzhi Cao, Vahid Mirjalili, Sebastian Raschka. Rank-consistent Ordinal Regression for Neural Networks. arXiv:1901.07884v3. https://arxiv.org/abs/1901.07884v3



Theorem 1 (ordered biases). *By minimizing loss function defined in Eq.* (3)*, the optimal solution* $(\mathbf{W}^*, \mathbf{b}^*)$ *satisfies* $b_1^* \geq b_2^* \geq \ldots \geq b_{K-1}^*$.

Proof. Suppose $(\mathbf{W}, b)$ is an optimal solution and $b_k < b_{k+1}$ for some k. Claim: by either replacing b_k with b_{k+1} or replacing b_{k+1} with b_k, we can decrease the objective value L. Let

$$A_1 = \{n : y_n^{(k)} = y_n^{(k+1)} = 1\},\ A_2 = \{n : y_n^{(k)} = y_n^{(k+1)} = 0\},\ A_3 = \{n : y_n^{(k)} = 1,\ y_n^{(k+1)} = 0\}.$$

By the ordering relationship we have $A_1 \cup A_2 \cup A_3 = \{1, 2, \ldots, N\}$. Denote $p_n(b_k) = s(g(\mathbf{x}_n, \mathbf{W}) + b_k)$ and

$$\delta_n = \log(p_n(b_{k+1})) - \log(p_n(b_k)), \quad \delta'_n = \log(1 - p_n(b_k)) - \log(1 - p_n(b_{k+1})).$$

Since $p_n(b_k)$ is increasing in b_k, we have $\delta_n > 0$ and $\delta'_n > 0$.

If we replace b_k with b_{k+1}, the loss terms related to kth task are updated. The change of loss L (Eq. 3) is given as

$$\Delta_1 L = \lambda^{(k)} \Big[- \sum_{n \in A_1} \delta_n + \sum_{n \in A_2} \delta'_n - \sum_{n \in A_3} \delta_n \Big].$$

Accordingly, if we replace b_{k+1} with b_k, the change of L is given as

$$\Delta_2 L = \lambda^{(k+1)} \Big[\sum_{n \in A_1} \delta_n - \sum_{n \in A_2} \delta'_n - \sum_{n \in A_3} \delta'_n \Big].$$

By adding $\frac{1}{\lambda^{(k)}} \Delta_1 L$ and $\frac{1}{\lambda^{(k+1)}} \Delta_2 L$, we have

$$\frac{1}{\lambda^{(k)}} \Delta_1 L + \frac{1}{\lambda^{(k+1)}} \Delta_2 L = - \sum_{n \in A_3} (\delta_n + \delta'_n) < 0,$$

and know that either $\Delta_1 L < 0$ or $\Delta_2 L < 0$. Thus, our claim is justified, and we conclude that any optimal solution $(\mathbf{W}^*, b^*)$ that minimizes L satisfies $b_1^* \geq b_2^* \geq \ldots \geq b_{K-1}^*$. □

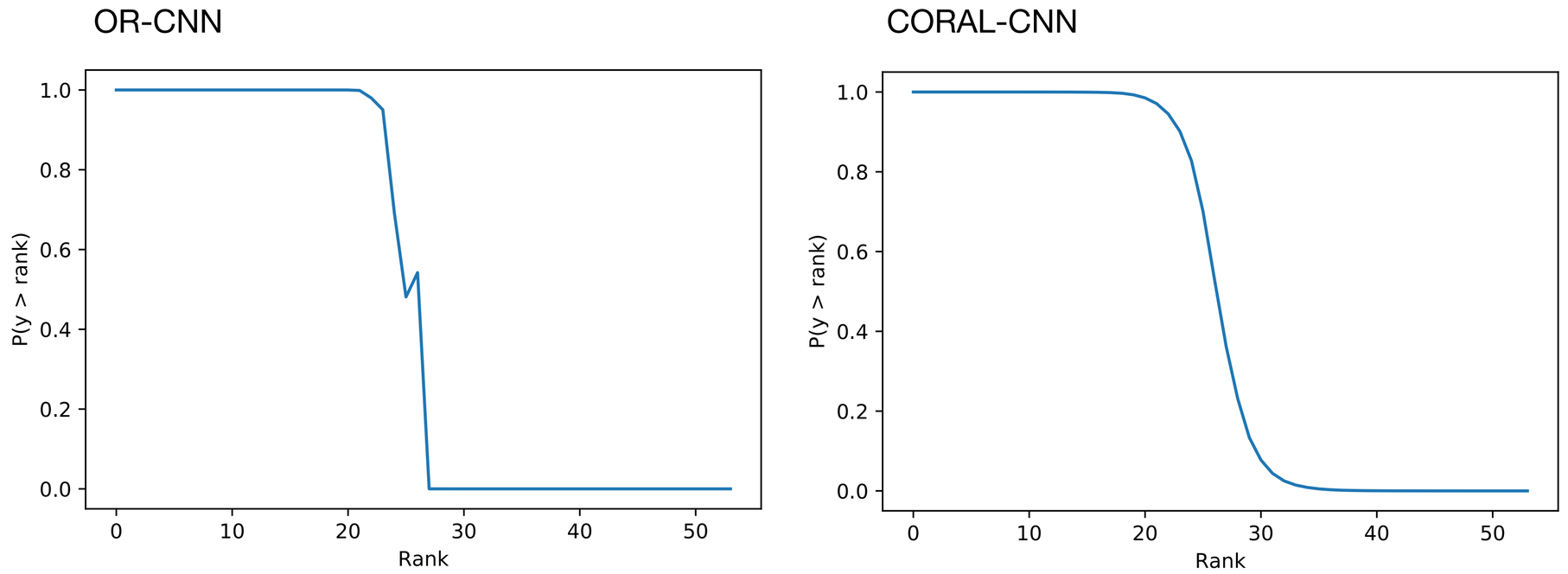


Figure S2: Plots show graphs of the predicted probabilities for each binary classifier task on one test data point in the MORPH dataset by OR-CNN (left subpanel) and CORAL-CNN (right subpanel). In this example, the ordinal regression CNN has an inconsistency at rank 26. The CORAL-CNN does not suffer from inconsistencies such that the rank prediction is a cumulative distribution function.

Wenzhi Cao, Vahid Mirjalili, Sebastian Raschka. Rank-consistent Ordinal Regression for Neural Networks. arXiv:1901.07884v3. https://arxiv.org/abs/1901.07884v3

Implementing CORAL with a VGG-16 backbone

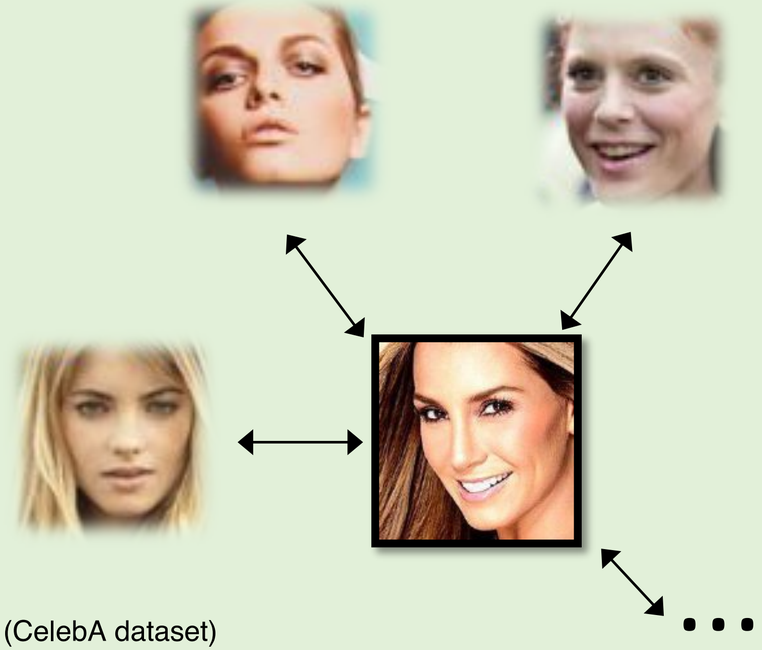
4.1_CORAL.ipynb

Part II: Hiding Soft-Biometric Attributes from Face Images

Biometric (Face) Recognition

A. Identification

Determine identity of an unknown person
1-to-*n* matching

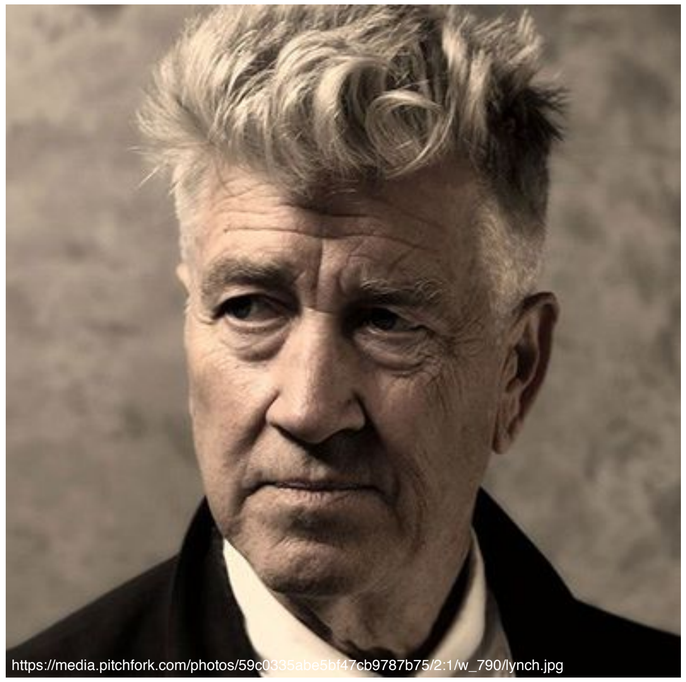


(CelebA dataset)

B. Verification

Verify claimed identity of a person
1-to-1 matching

(MUCT dataset)



Identity	John Doe

Gender	Male
Age	65
Race	Caucasian
Medical	Healthy

SOFT BIOMETRIC ATTRIBUTES

Soft-biometric Attributes: Issues and Concerns

1. **Identity theft:** combining soft biometric info with publicly available data
2. **Profiling:** e.g., gender/race based profiling
3. **Ethics:** extracting data without users' consent

Goal: Selective Privacy

1. Perturb soft-biometric (e.g., gender) information
2. Ensure realistic face images
3. Retain biometric face recognition utility

Face Matcher

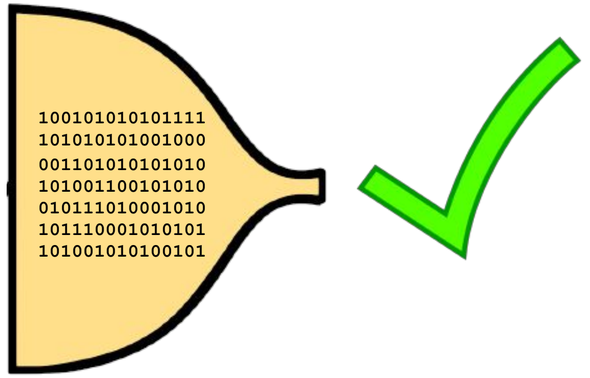
100101010101111
101010101001000
001101010101010
101001100101010
010111010001010
101110001010101
101001010100101

p("same person")

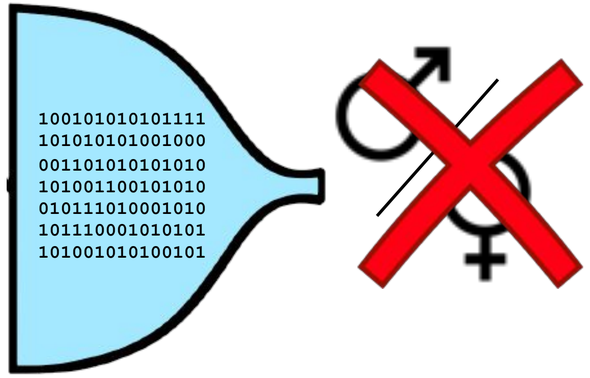
100101010101111
101010101001000
001101010101010
101001100101010
010111010001010
101110001010101
101001010100101

p("male")

Gender Classifier



Face Matcher

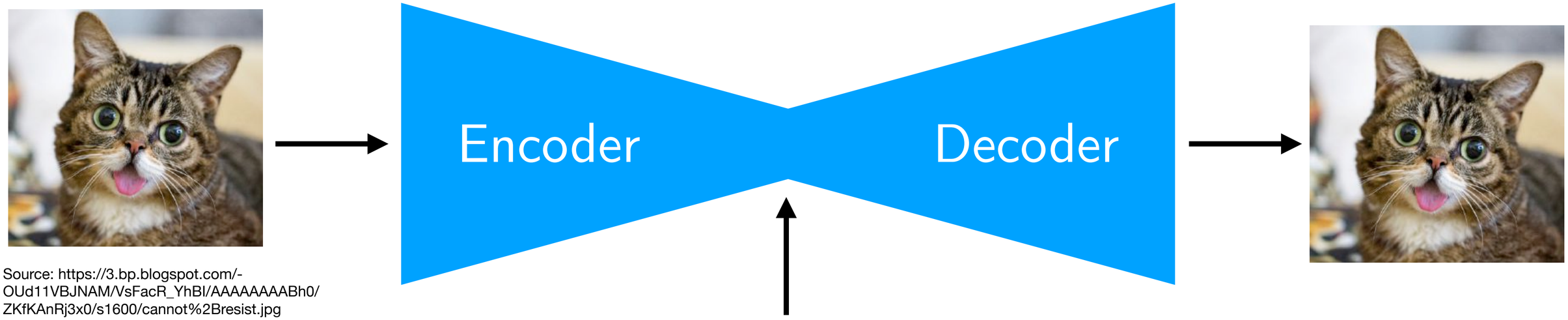


Gender Classifier

Autoencoders

Unsupervised Learning:

Representation Learning/Dimensionality Reduction



Source: https://3.bp.blogspot.com/-OUd11VBJNAM/VsFacR_YhBI/AAAAAAAABh0/ZKfKAnRj3x0/s1600/cannot%2Bresist.jpg

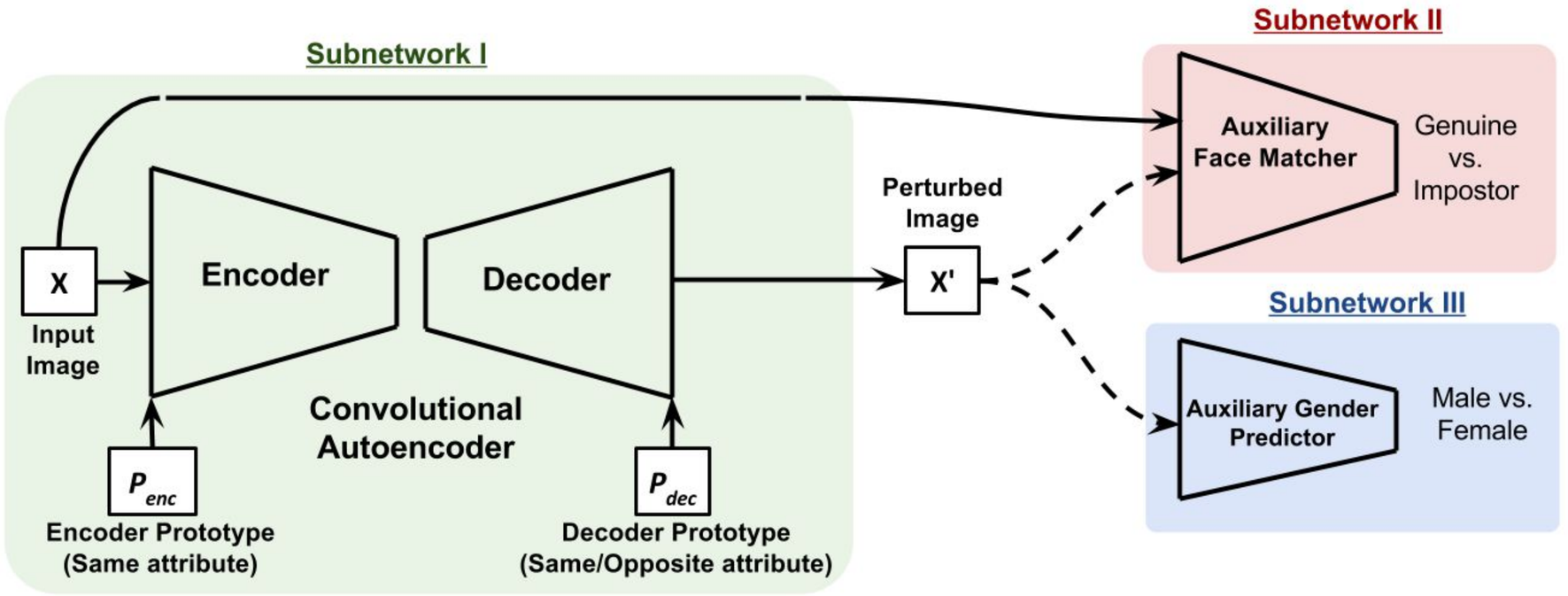
Autoencoder to perturb image

$$\phi(\mathbf{X}) = \mathbf{X}'$$

Face Matcher

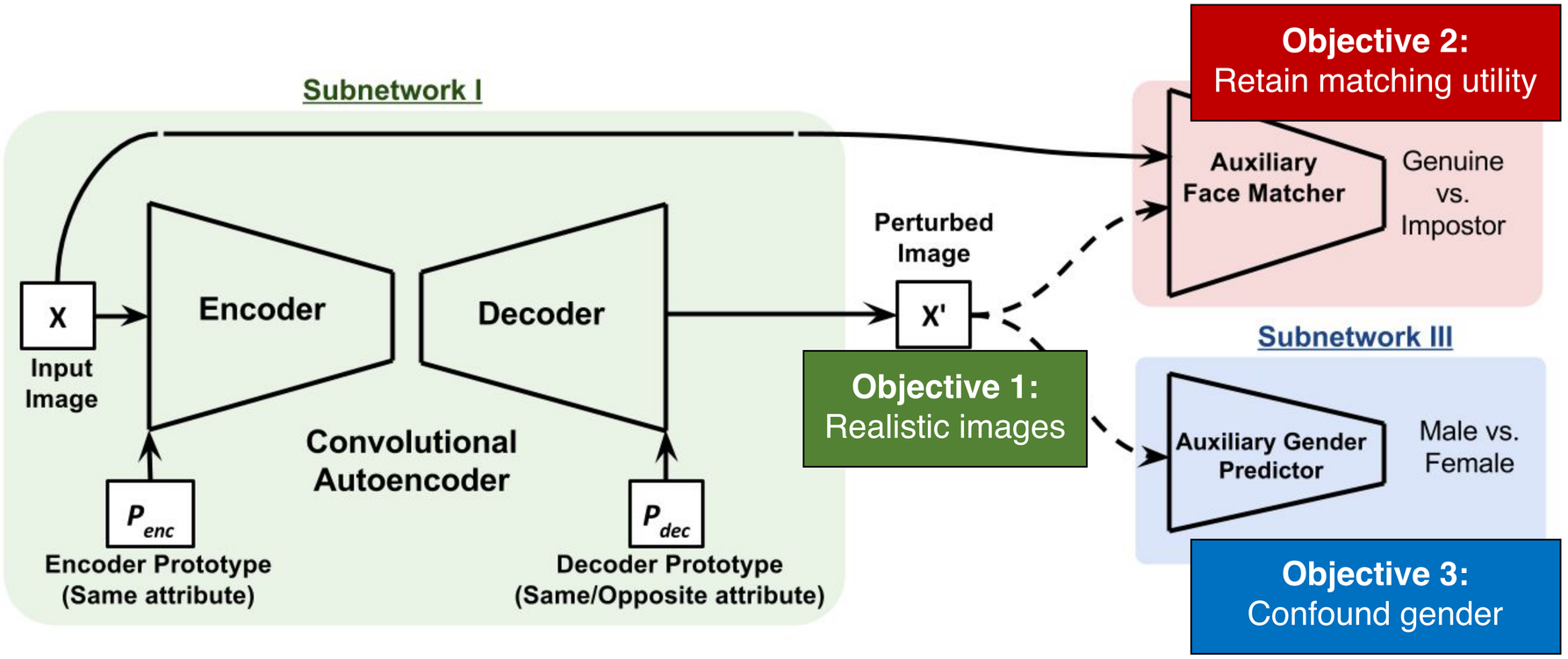
Gender Classifier

General architecture of the semi-adversarial network (SAN)

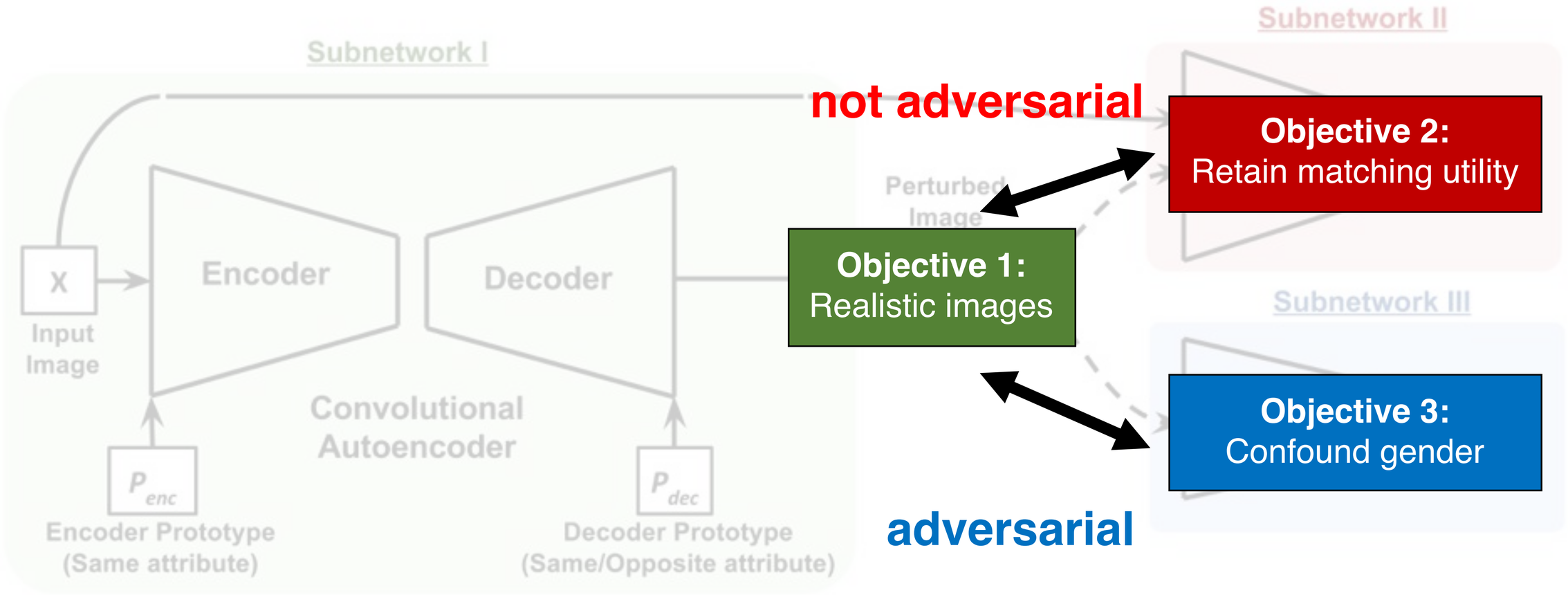


Vahid Mirjalili, Sebastian Raschka, Anoop Namboodiri, and Arun Ross (2018) *Semi-adversarial networks: Convolutional autoencoders for imparting privacy to face images.* Proc. of 11th IAPR International Conference on Biometrics (ICB 2018)

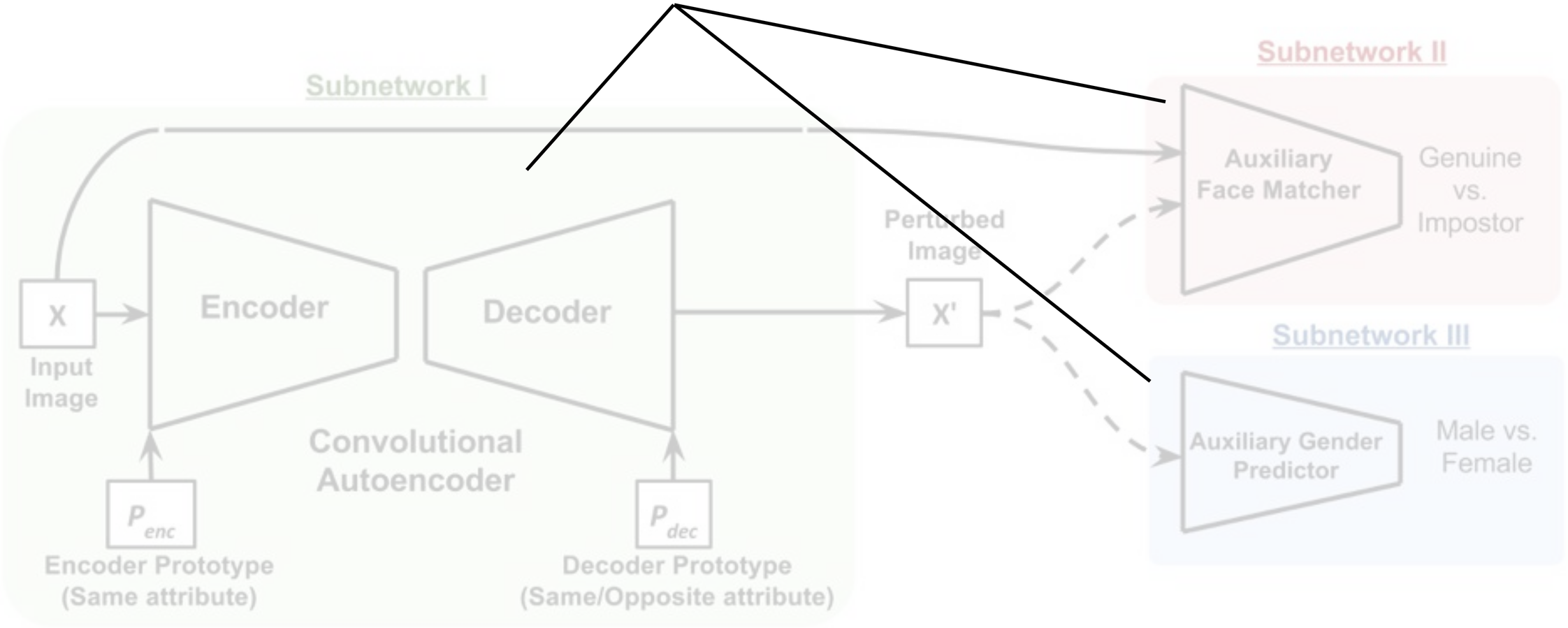
General architecture of the semi-adversarial network (SAN)



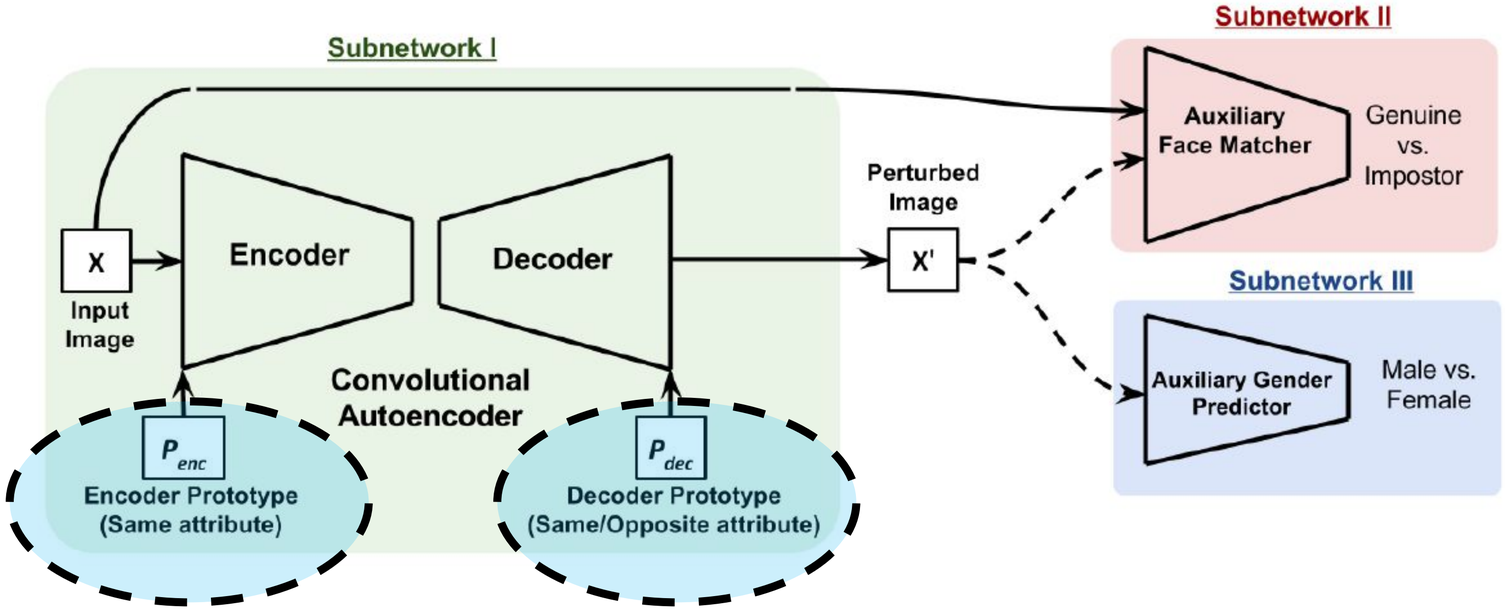
Semi-adversarial network



Convolutional neural networks



Gender prototypes



Class labels

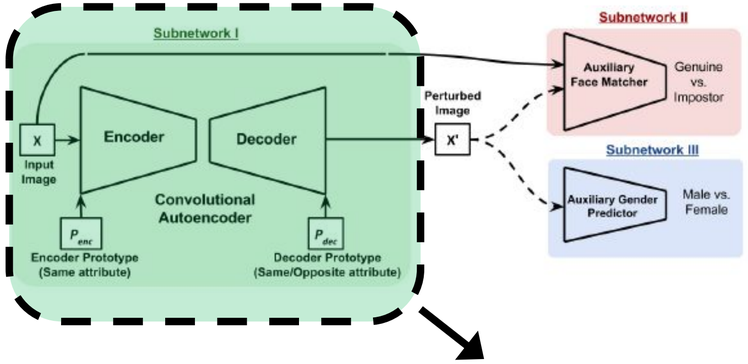
$y \in \{0, 1\}$,

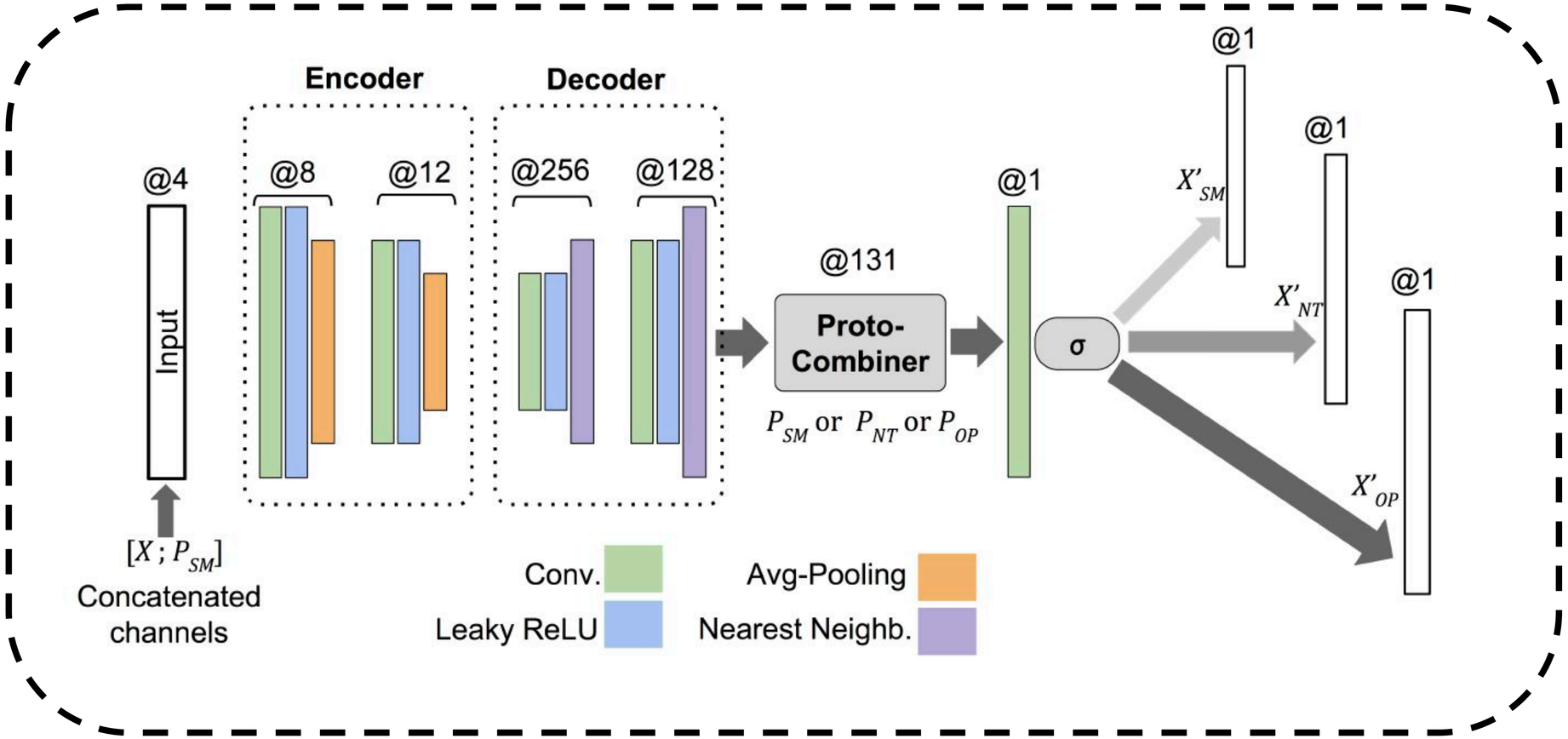
where $0 = \text{female}, 1 = \text{male}$

Same gender prototype: $$P_{\text{SM}}(y) = yP_{\text{Male}} + (1 - y)P_{\text{Female}}$$

Opposite gender prototype: $$P_{\text{OP}}(y) = (1 - y)P_{\text{Male}} + yP_{\text{Female}}$$

Convolutional autoencoder architecture





Cost function for semi-adversarial learning

1. Pixel-wise similarity term
   - Only used during the pre-training of the autoencoder

$$J_D(X, X'_{SM}) = \sum_{i=1}^{224\times224} MSE\left(X^{(i)}, X'^{(i)}_{SM}\right)$$

2. Loss term related gender attribute
   - Correctly predict gender of X'_{SM}
   - Flip the gender prediction of X'_{OP}

$$J_G(X, X'_{SM}, X'_{OP}, y; f_G) = S\left(y, f_G(X'_{SM})\right) + S\left(1-y, f_G(X'_{OP})\right)$$

3. Loss related to matching

$$J_M(X, X'_{SM}; F_M) = \left|\left|F_M(X'_{SM}) - F_M(X)\right|\right|_2^2$$

SAN Examples

Original Inputs

Male: 99%

Female: 98%

Male: 97%

Male: 100%

Outputs

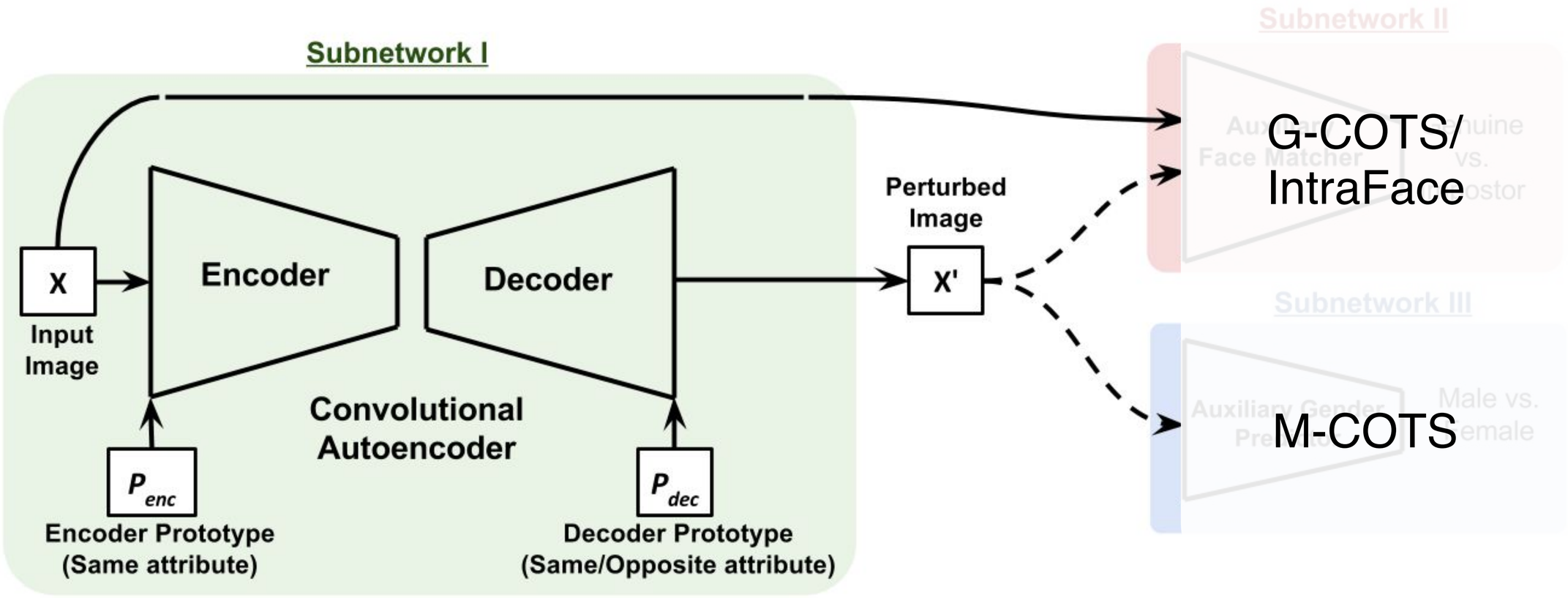
Female: 69%

Male: 99%

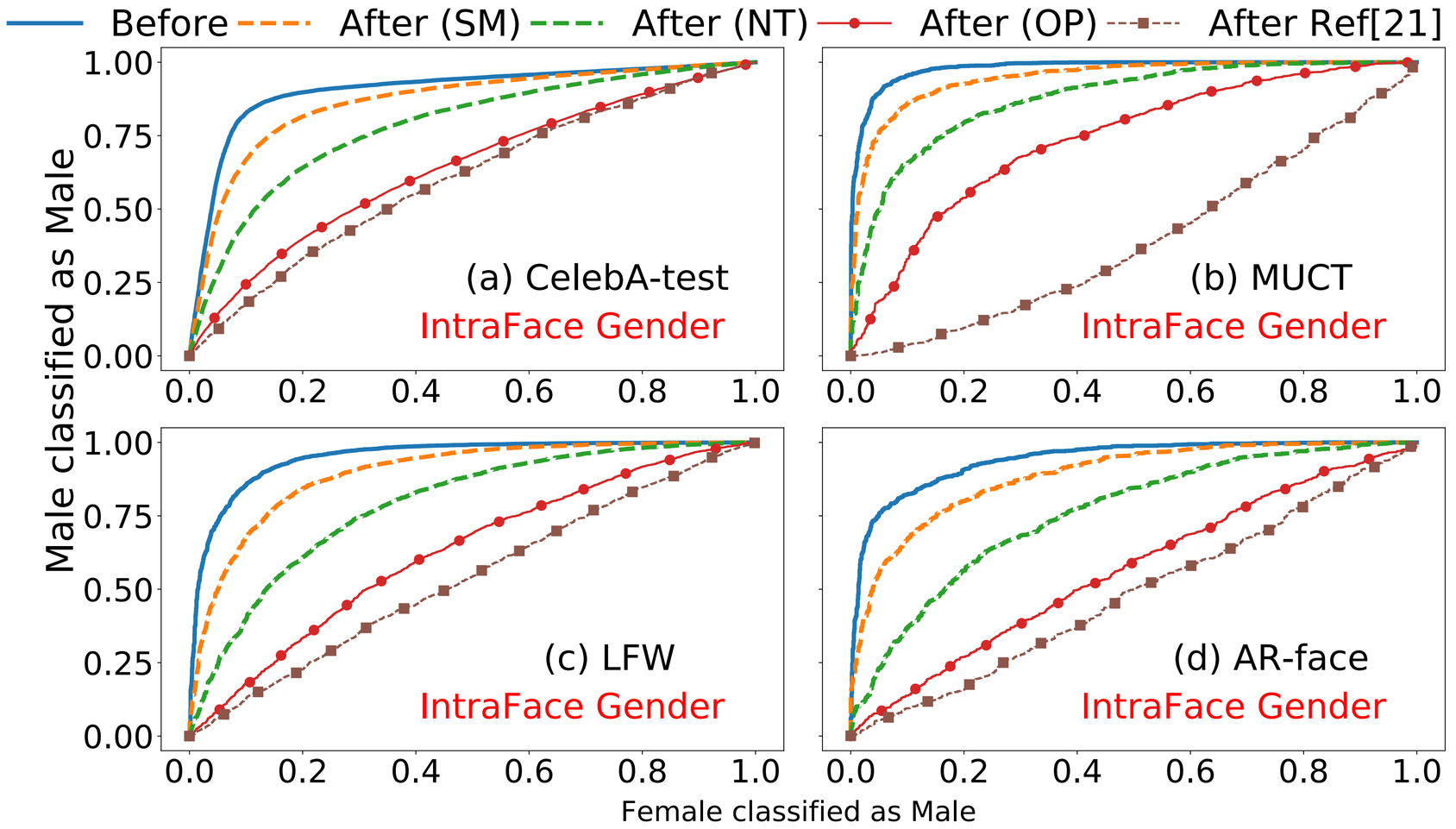
Female: 71%

Female: 58%

Replacing Detachable Parts for Evaluation



IntraFace Gender Classifier Performance

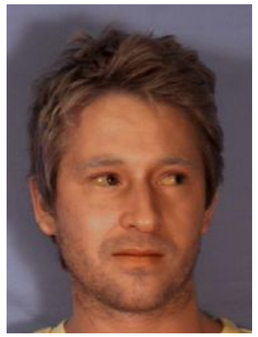


[21] A. Othman and A. Ross. Privacy of facial soft biometrics: Suppressing gender but retaining identity. In *European Conference on Computer Vision Workshop*, pages 682–696. Springer, 2014.

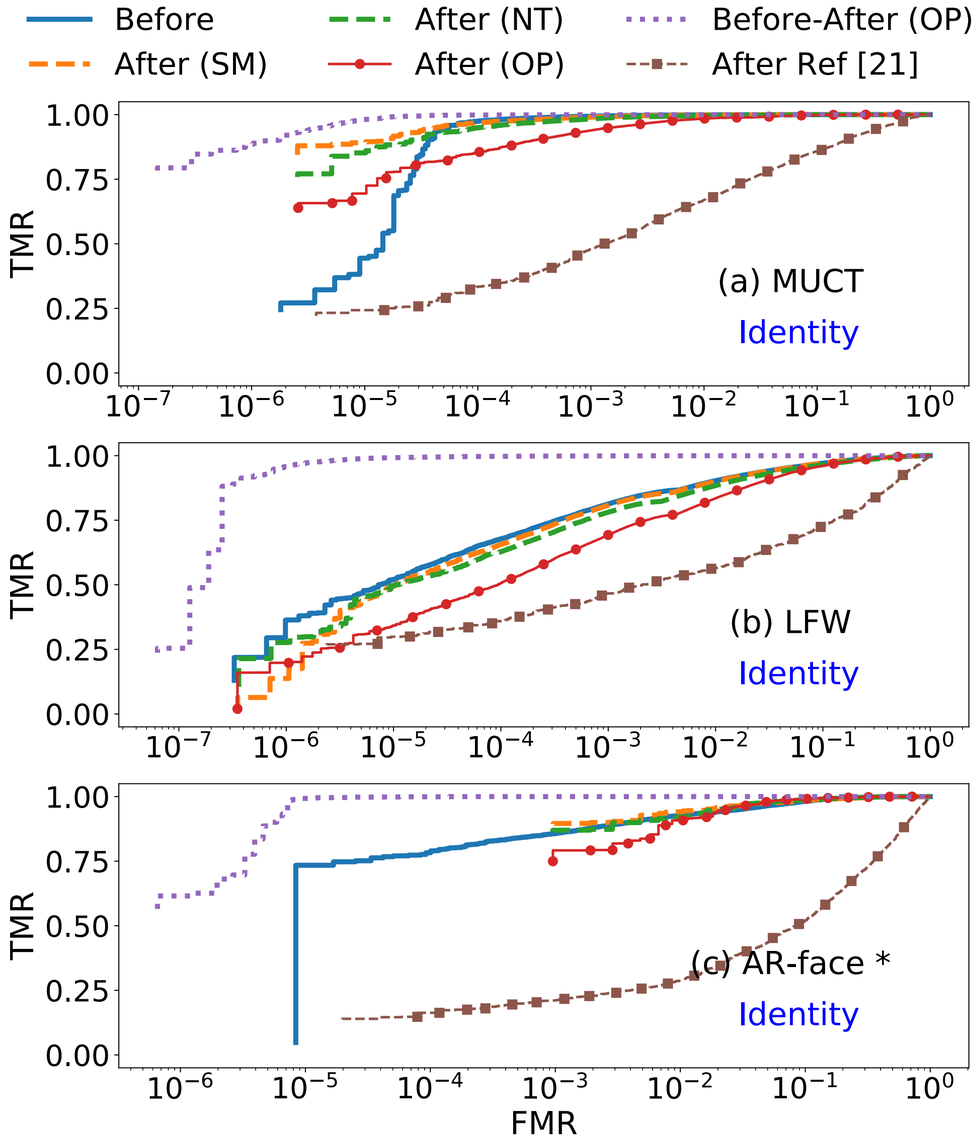
Face matching performance

Before-After (OP)

Multi-subject comparisons



Before
After (SM)
After (NT)
After (OP)
After Ref [1]



[21] A. Othman and A. Ross. Privacy of facial soft biometrics: Suppressing gender but retaining identity. In *European Conference on Computer Vision Workshop*, pages 682–696. Springer, 2014.

Gender Privacy: An Ensemble of Semi Adversarial Networks for Confounding Arbitrary Gender Classifiers

Improvements to construct a more diverse set of SAN models for better generalizability via ensembling

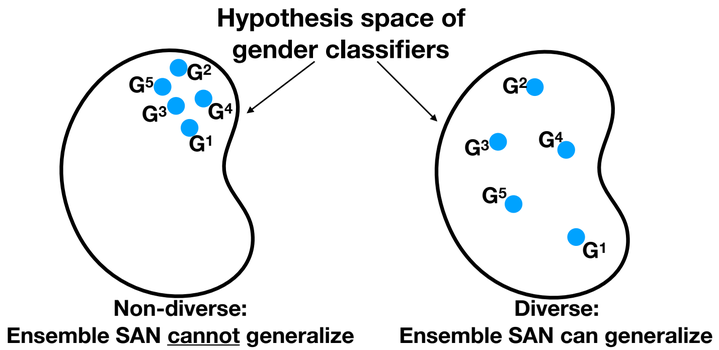


Figure 1: Diversity in an ensemble SAN can be enhanced through its auxiliary gender classifiers (see Figure 2). When the auxiliary gender classifiers lack diversity, ensemble SAN cannot generalize well to arbitrary gender classifiers.

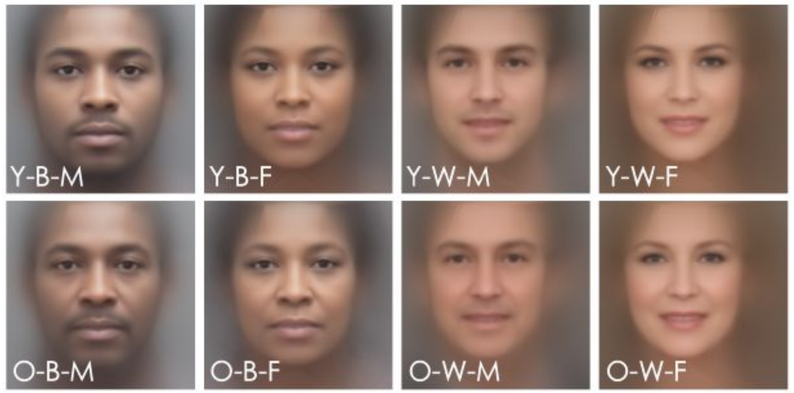
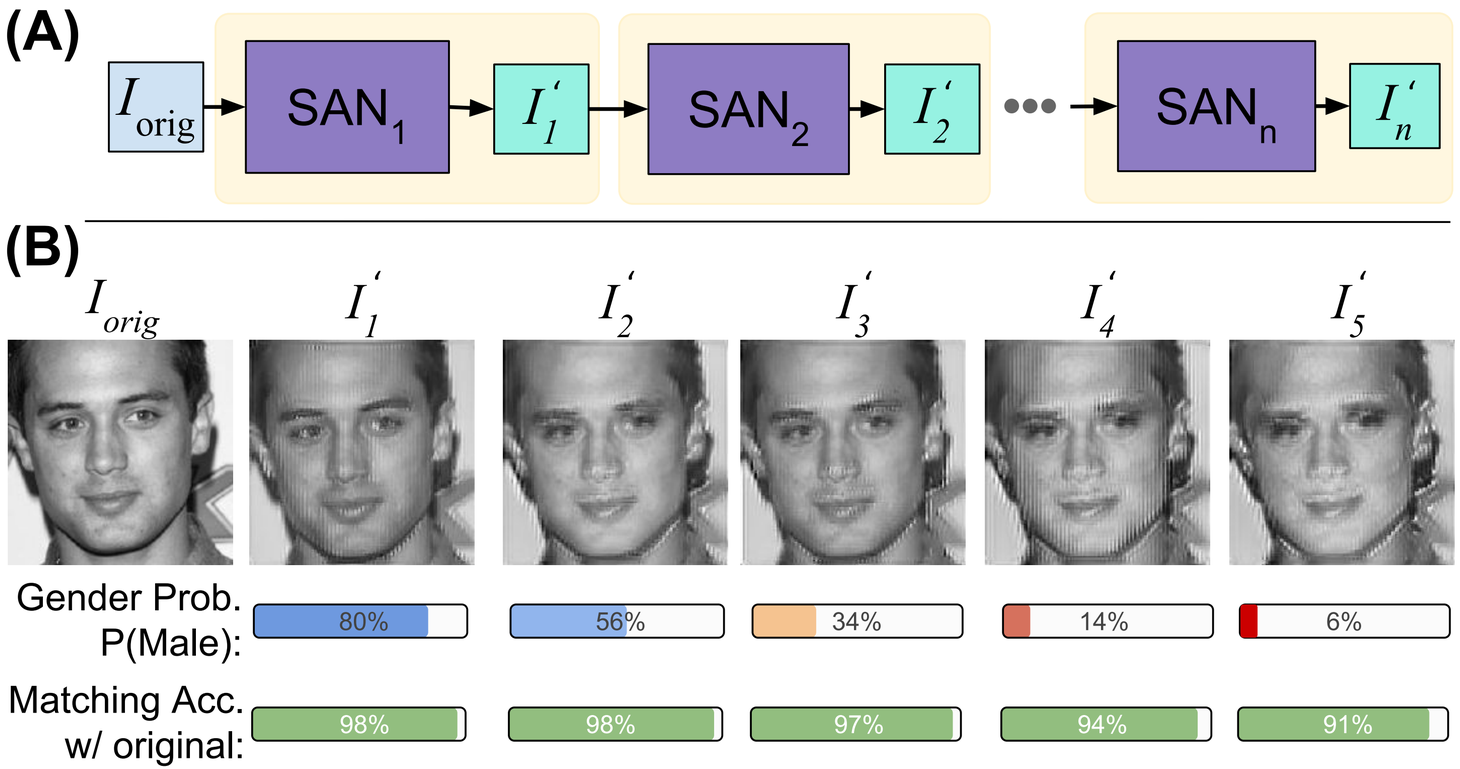


Figure 4: Face prototypes computed for each group of attribute labels. The abbreviations at the bottom of each image refer to the prototype attribute-classes, where Y=young, O=old, M=male, F=female, W=white, B=black.

Vahid Mirjalili, Sebastian Raschka, and Arun Ross (2018) *Gender Privacy: An Ensemble of Semi Adversarial Networks for Confounding Arbitrary Gender Classifiers.* 9th IEEE International Conference on Biometrics: Theory, Applications, and Systems (BTAS 2018)

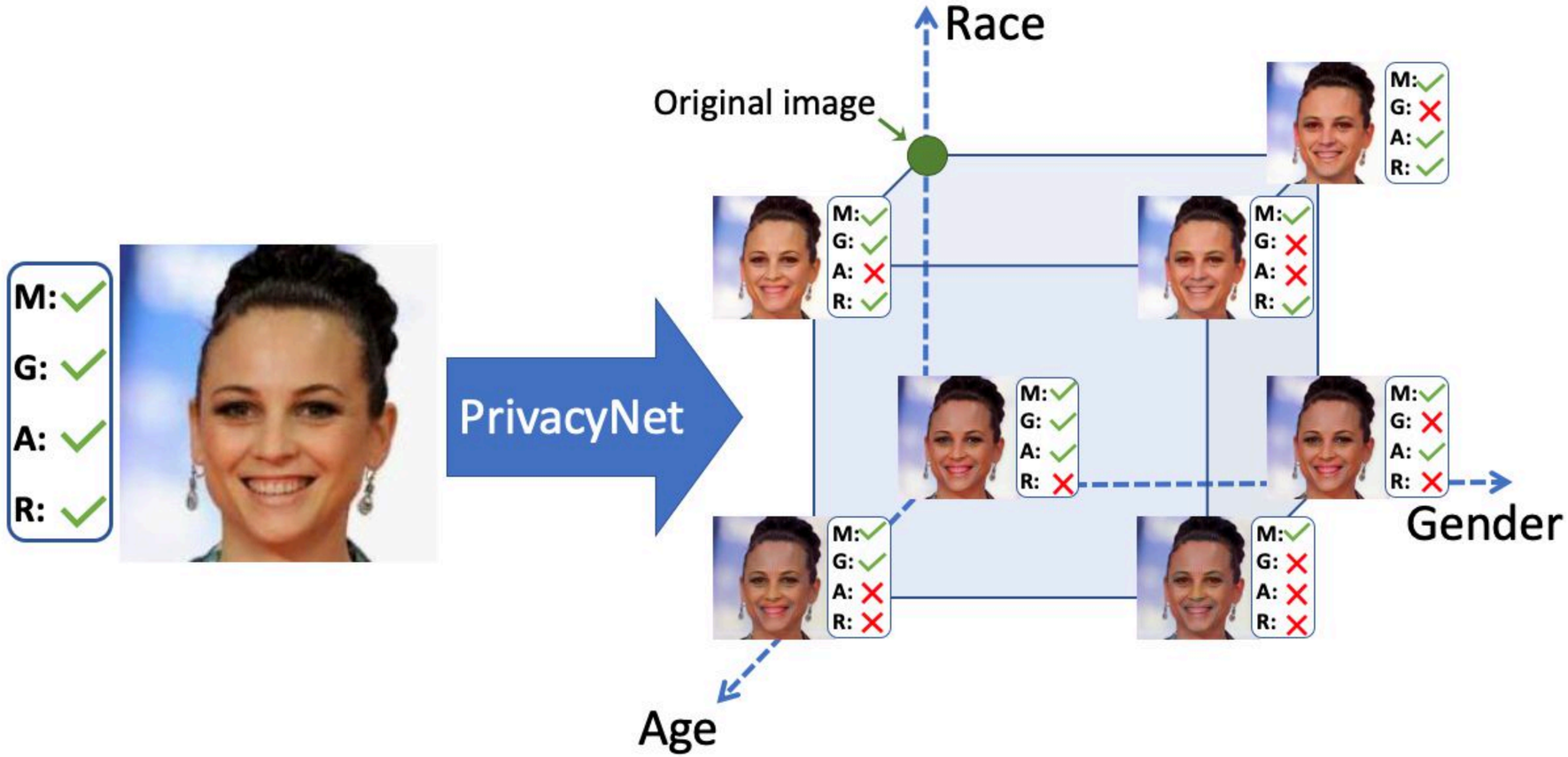
FlowSAN: Privacy-enhancing Semi-Adversarial Networks to Confound Arbitrary Face-based Gender Classifiers



Improvements to better control the perturbations and enhance the removal of soft-biometric information

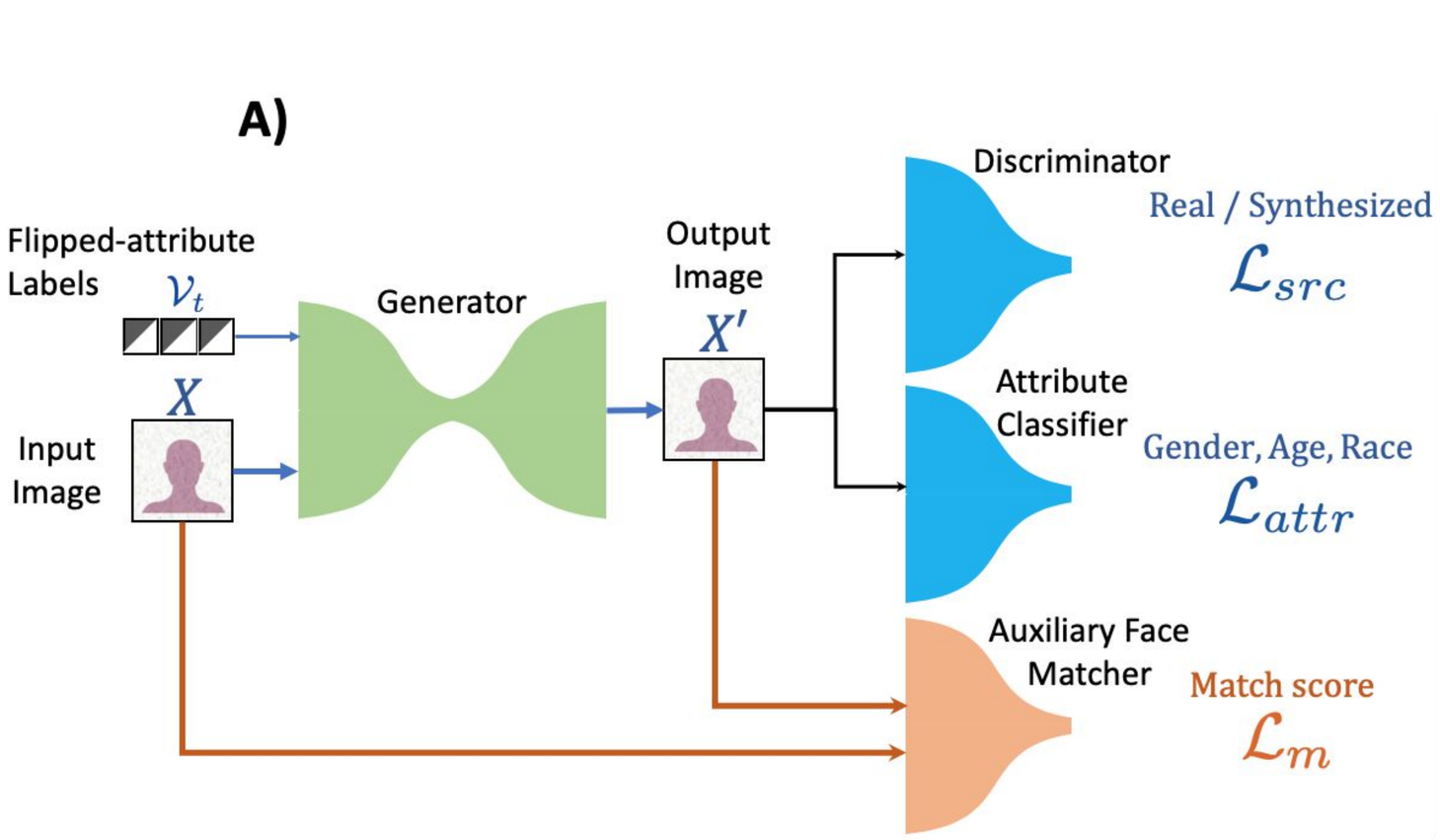
Vahid Mirjalili, Sebastian Raschka, Arun Ross (2019)
FlowSAN: Privacy-enhancing Semi-Adversarial Networks to Confound Arbitrary Face-based Gender Classifiers
IEEE Access 2019, 10.1109/ACCESS.2019.2924619

PrivacyNet: Semi-Adversarial Networks for Multi-attribute Selective Privacy

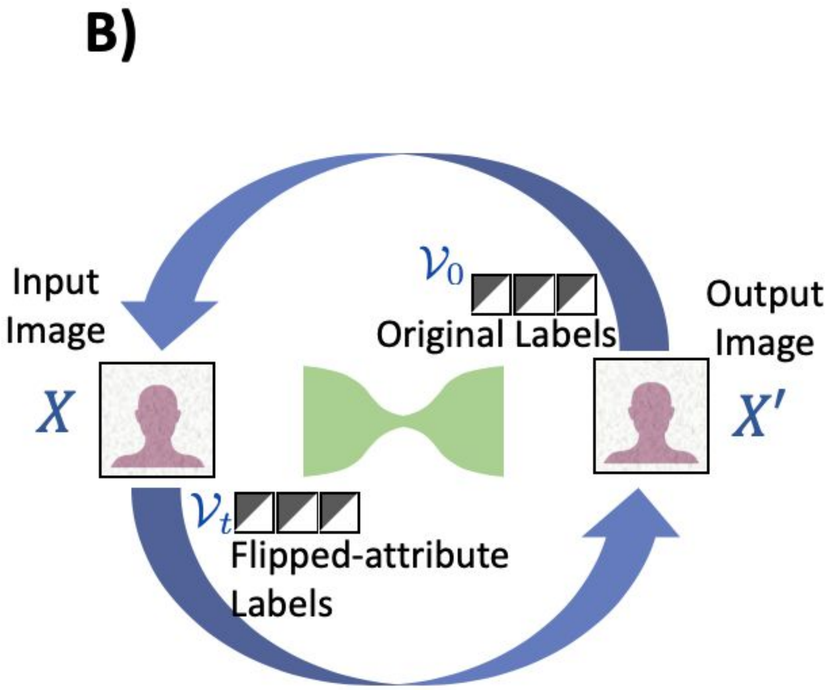


Vahid Mirjalili, Sebastian Raschka, and Arun Ross (2019) *PrivacyNet: Semi-Adversarial Networks for Multi-attribute Differential Privacy* (manuscript in prep.)

PrivacyNet: Semi-Adversarial Networks for Multi-attribute Selective Privacy



Architecture



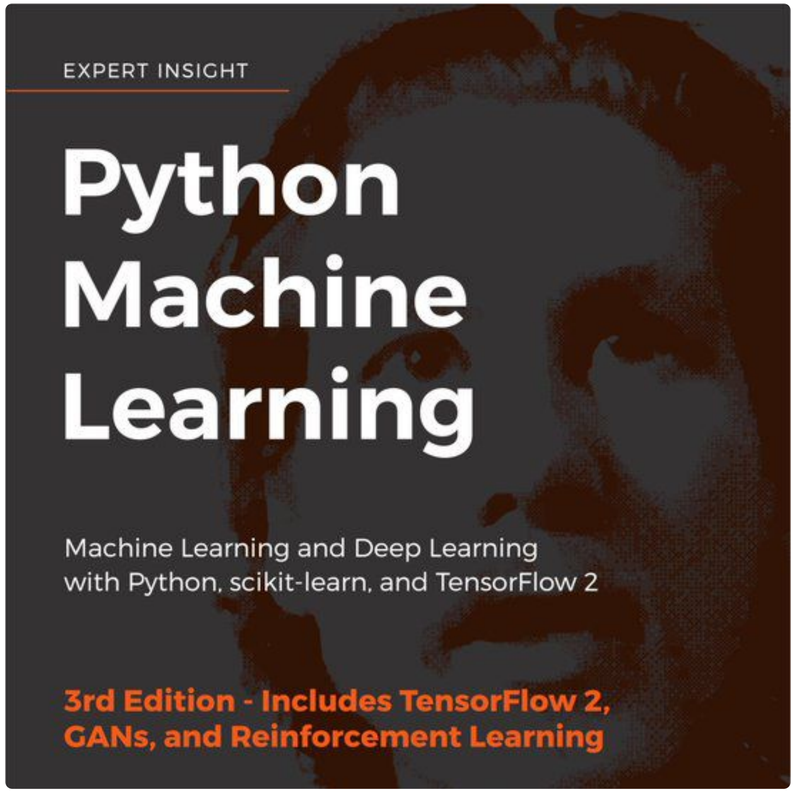
Cycle-consistency constraint

Thank You!

Packt
@PacktPub

'Python Machine Learning - Third Edition' is coming soon.@rasbt & @vmirly are back for the latest edition, revised to cover TensorFlow 2, GANs, and reinforcement learning.

Pre-order your copy on Amazon here: amzn.to/2L0D1Cp



45 9:33 AM - Aug 23, 2019

See Packt's other Tweets