CATCH THEM UNATTENTIVE: AN ORIENTATION AWARE FACE RECOGNITION MODEL

Supplementary Material

February 6, 2024

A. DATASET PREPROCESSING

As outlined in the main paper, we utilized a subset of the Casia WebFace dataset to train the model. The faces within the Casia WebFace dataset have already undergone alignment and resizing to dimensions of 112×112. To ensure the presence of an adequate range of pose variations in the dataset, we applied filtering to certain individual classes. These classes were selected based on containing less than 60 percent of faces with a pose angle of $30°$. Essentially, this indicates that the majority of faces in those classes exhibit a frontal orientation.

A.1. Estimation of Roll and Face Alignment

We determined the roll, yaw, and pitch angles based on the five facial landmarks identified through MTCNN (Multi-Task Cascaded Convolutional Neural Networks). These facial landmarks consist of the left eye (x_{le}, y_{le}), right eye (x_{re}, y_{re}), left tip of the mouth (x_{lm}, y_{lm}), right tip of the mouth (x_{rm}, y_{rm}), and nose (x_n, y_n).

$$dP_x = max\left((x_{re} - x_{le}), 1\right) \tag{1}$$

$$dP_y = y_{re} - y_{le} \tag{2}$$

$$roll, \delta = \tan^{-1}\left(\frac{dPy}{dPx}\right) \tag{3}$$

Here, dP_x and dP_y represent shifts in the eye positions along the x and y directions, respectively. By employing the roll angle δ, we can align the orientation of the faces so that both eyes are parallel to the horizontal axis. Following this alignment process, all landmark positions need to be transformed to new coordinates in a subsequent manner.

$$\alpha = \cos\delta \tag{4}$$

$$\beta = \sin\delta \tag{5}$$

$$xr = \alpha \cdot x + \beta \cdot y + (1 - \alpha) \cdot (x_n/2) - \beta \cdot (y_n/2) \tag{6}$$

$$yr = -\beta \cdot x + \alpha \cdot y + \beta \cdot (x_n/2) + (1 - \alpha) \cdot (y_n/2) \tag{7}$$

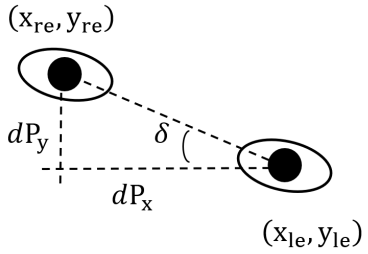

Fig. 1: Facial alignment involves utilizing facial landmarks like the left and right eyes for reference. By applying a rotational adjustment of δ degrees to the face, it is aligned in a manner that aligns both eyes horizontally on the same level.

Here, (x, y) represents the coordinate before rotation, while (xr, yr) indicates the coordinate after rotation. The original location of the nose is denoted by (x_n, y_n).

A.2. Estimation of Pitch and Yaw

To estimate the pitch and yaw, we need to determine two factors: the average distance between the eyes and mouth, and the average distance between the eyes and nose.

				
Roll	0.10	-0.52	0.08	-1.66
Yaw	4.93	-19.54	-55.88	23.07
Pitch	7.10	5.23	5.57	6.22

Fig. 2: The pose angles were calculated using the described algorithm. All faces were aligned using the roll angle, making the roll angles irrelevant in these cases.

$$
\begin{aligned}
dX_{mtoe} &= (xr_{re} - xr_{le} + xr_{lm} - xr_{rm})/2 \\
dY_{mtoe} &= (yr_{rm} - yr_{le} + yr_{lm} - yr_{re})/2 \\
dX_{ntoe} &= (xr_{re} - xr_{n} + xr_{lm} - xr_{n})/2 \\
dY_{ntoe} &= (yr_{rm} - yr_{le} + yr_{lm} - yr_{n})/2
\end{aligned} \tag{8}
$$

Here, dX_{mtoe} and dY_{mtoe} represent the average distance between the eyes and the mouth in the x and y directions, respectively. Similarly, dX_{ntoe} and dY_{ntoe} indicate the average distance between the eyes and the nose in the x and y directions, respectively. The coordinates (xr_{re}, yr_{re}), (xr_{le}, yr_{le}), (xr_{rm}, yr_{rm}), (xr_{lm}, yr_{lm}), and (xr_{n}, yr_{n}) correspond to the rotated positions of the right eye, left eye, right tip of the mouth, left tip of the mouth, and nose, respectively.
The pose angles of roll, yaw, and pitch will span the range from $-90°$ to $+90°$. Occasionally, the angles computed using the previously mentioned method produce values outside this range. When this occurs, we have truncated the values to stay within the $-90°$ to $+90°$ interval. A collection of pose angle examples determined through this methodology is presented in Figure 2.

A.3. Low Resolution Augmentation

To enhance face recognition performance when dealing with images of varying scales, we integrated a low-resolution augmentation approach. This technique involves resizing images to a scale chosen randomly from 20 percent to 100 percent of the original size. Subsequently, these downscaled images are brought back to their original size through upscaling. During both downscaling and upscaling, we select an interpolation method such as nearest neighbor, bilinear, bicubic, area, or lanczos at random for each operation.

Fig. 3: An illustration of low-resolution augmentation is provided in this example. The initial image is presented as the original, while the subsequent image demonstrates the effect after applying the low-resolution augmentation technique.

B. ANALYSIS OF ORIENTATION TOLERANT MARGIN AND GRADIENT

The orientation tolerant loss function is given as follows,

$$
\mathcal{L}_{CE}(x_i) = -\log \frac{\exp(f(\theta_{y_i}))}{\exp(f(\theta_{y_i})) + \sum_{j \neq y_i}^{C} \exp(f(\theta_j))}, \tag{9}
$$

where affinity function $f(\cdot)$ is defined as follows:

$$
f(\theta_j) = \begin{cases} s \cdot \big(cos(\theta_j - \alpha_i) + \beta_i\big) & j = y_i \\ s \cdot cos(\theta_j) & j \neq y_i \end{cases} \tag{10}
$$

The derivative of $\mathcal{L}_{CE}$ w.r.t j^{th} class weight vector W_j is given as follows,

$$
\frac{\partial \mathcal{L}_{CE}(x_i)}{\partial W_j} = \Big(P_j^{(i)} - \mathbb{1}(y_i = j)\Big) \frac{\partial f(\cos\theta_j)}{\partial \cos\theta_j} \frac{\partial \cos\theta_j}{\partial W_j} \tag{11}
$$

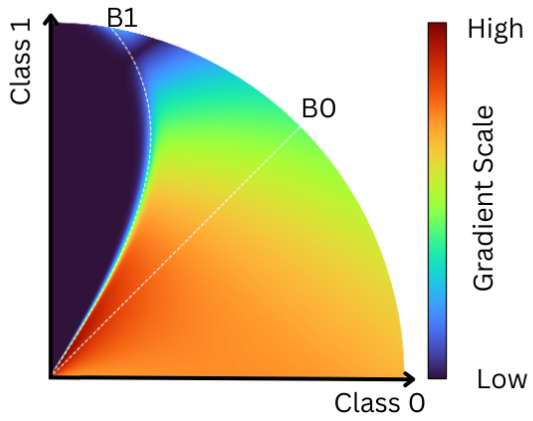

Fig. 4: Gradient scale g by varying the difficulty indicator ξ and angular distance θ from the true class (class 1) center.

The expression $(P_j^{(i)} - \mathbb{1}(y_i = j))\frac{\partial f(\cos\theta_j)}{\partial \cos\theta_j}$ here is a scalar quantity that changes in relation to the image difficulty indicator ξ. Consequently, this scalar term can be designated as the gradient scaling factor, denoted by g. On the other hand, the component $\frac{\partial \cos\theta_j}{\partial W_j}$ represents the gradient direction aspect and remains independent of ξ. In the context of the proposed loss function, the gradient scaling factor g is formulated as follows:

$$
g = \Big(P_j^{(i)} - \mathbb{1}(y_i = j)\Big)\, s \left(\cos\alpha_i + \frac{\cos\theta_i \sin\alpha_i}{\sqrt{1 - \cos^2\theta_i}}\right) \tag{12}
$$

Figure 4 illustrates how the gradient changes with respect to variations in the image difficulty indicator ξ. The observation

reveals that an image with a higher difficulty (ξ closer to -1) generates a more pronounced gradient, especially when it's situated near the boundary with a small θ value. Conversely, an image with lower difficulty (ξ closer to $+1$) yields a greater gradient when the sample is positioned far from the boundary, corresponding to a larger θ value.