UCCS Watchlist Challenge

Tomáš Repák

27.5.2024

1 Intro

Firstly, I would like to thank the organizers, and especially Mr. Furkan Kasim for the opportunity to partake in the UCCS Watchlist Challenge and for kind and informative responses to my inquiries. It has been a great educational exercise.

It is worth mentioning that I have signed up for the challenge without any prior hands-on experience in face recognition or face identification domains. I chose to sign up for the competition to study the problem, and find out whether my code works reasonably well. My goal was to implement a rather simple and old-school technique and obtain a proof that they work, before perhaps moving on to the more complicated things in the future. There are already theoretical plans in place for potential future application.

As my application to the challenge got approved on 22nd March and I had to code everything from scratch, while also being quite time and resource restricted, I did not expect to even surpass the baseline, as the baseline face identification model seems to be more advanced compared to my approach. My personal goal within this challenge is to ensure that the system can work reasonably well and is not much worse than the baseline or the less-performing solutions. According to the results, my goals were achieved. The results suggest that the submitted model is not a complete RNG-machine, despite performing less than the baseline. Since only four teams participated in the challenge, the results are not extra informative, however I got an idea about where my model stands at the moment, for which I am glad.

2 Approach description

2.1 Environment

The whole project is developed in pure PyTorch. I chose not to opt for PyTorch Lightning as it would cost undesired time overhead for me to code it properly and I do not think it was necessary in the current state of the project. Hydra config environment was used to configure individual experiments and their settings.

Hardware-wise, I used solely my personal desktop PC, equipped with a single NVIDIA RTX 2080 GPU (8GB VRAM) and a Ryzen 9 5900X CPU. During the evaluation stage of the challenge, It was **very** convenient to be in possession of 64GB RAM capabilities.

2.2 Datasets

Datasets used are the following. The same settings were used for all submissions and individual models. Sampling was done using a random seed, and the experiments will therefore not be reproducible.

- **UCCS Watchlist dataset** - UnConstrained College Students dataset. See
  `https://vast.uccs.edu/Opensetface/`
  `https://exposing.ai/uccs/`
- **LFW dataset** - Labeled Faces in the Wild. See `https://vis-www.cs.umass.edu/lfw/`
- **CelebA dataset** - See `https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`

LFW and CelebA dataset images were aligned and cropped to match the nature of the UCCS images. See Figures 1 and 2 for reference. The detections were done using MTCNN [fac] detector. Initially, I was planning on using some surroundings around the face, but this proved to confuse the model.

All datasets were split into train (80%) and validation (20%) sets. No rigorous evaluation was conducted prior to the final submission.

For the sake of simplicity, all models were trained using the same image input shape ($224 \times 224 \times 3$), with the same transformations and augmentations. Augmentations consist of *GaussianBlur* to fight blurriness in test images, *RandomRotation* to make the model more robust and *ColorJitter* with brightness set to 0.5 and contrast set to 0.3. Other parameters of the *ColorJitter* are set to null.

Notably, I do not use $HorizontalFlip$, as this contradicts my idea of identifying people. If a person has a unique recognizable element (such as a birthmark,

Figure 1: Example of preprocessing of the CelebA dataset

Figure 2: Example of preprocessing of the LFW dataset

scar, ...) on his/her left cheek, a scenario where this birthmark appears on the other cheek never happens. Using $HorizontalFlip$ could break this assumption and could potentially lead to decreased performance of the model.

2.3 Gallery

The gallery of identities for all models has been constructed in the same way. For each subject directory, all its images were loaded and again an MTCNN face detector with default parameters was run on them to detect faces. For each detected face (There could be no detections or more detections. Such images were completely discarded from computing the gallery), I have computed the face embedding and then for each subject, the embeddings were averaged to produce a final embedding, which is matched against the probes.

3 Submissions

Up to this point, everything was identical for each and every model included in all submissions. Individual submissions and their nuances are now explained in further detail.

As it is common in online competitions, I have opted for an ensemble model. My ensemble consists of three individual models. Most of my models are **eva02_tiny** Vision Transformers from HuggingFace [eva], with just a single exception being a MobileNetV2 CNN. I was nudged to choose smaller and tinier versions of the models in order to run the trainings to their meaningful end in an acceptable amount of time. For example, I could run the training with **eva02_small**, but the time per epoch increased significantly, up to the point where I chose not to pursue experiments based on such model.

Each of the heads required around 30 hours to train. However, some heads are recycled throughout different submissions. Altogether there are 5 fully trained unique models.

3.1 Submission 1

- **Head 1** - **EvaTiny** trained from scratch, using Triplet Loss and Euclidean distance. Produced embeddings are not normalized to $[0, 1]$. Sometimes I encountered values as high as 2.5 or.
- **Head 2** - **EvaTiny** trained from scratch, using Triplet Loss with Cosine distance. Produced embeddings are normalized to $[0, 1]$.
- **Head 3** - **pretrained EvaTiny**, finetuned using Triplet loss with Cosine distance. Produced embeddings are normalized to $[0, 1]$.

3.2 Submission 2

Identical to Submission 1. The only difference is that the final scores are squared. Small confidences are shrunk, big confidences remain strong.

3.3 Submission 3

- **Head 1 - pretrained MobileNetV2**, finetuned using Triplet Loss with Euclidean distance. Produced embeddings are normalized to $[0, 1]$.
- **Head 2 - EvaTiny** trained from scratch, using Triplet Loss with Cosine distance. Produced embeddings are normalized to $[0, 1]$.
- **Head 3 - pretrained EvaTiny**, finetuned using Triplet loss with Cosine distance. Produced embeddings are normalized to $[0, 1]$.

3.4 Submission 4

Identical to Submission 3. The only difference is that the final scores are squared. Small confidences are shrunk, big confidences remain strong.

3.5 Submission 5

Submission 5 substantially differs from the previous one. I wanted to assess the capabilities of just a single pretrained EvaTiny transformer and using that deduce the influence of the other models in previous ensembles. As such, Submission 5 is not an ensemble.

- **Head 1 - pretrained EvaTiny**, finetuned using Triplet loss with Cosine distance. Produced embeddings are normalized to $[0, 1]$.

3.6 Submission 6

Identical to Submission 5. The only difference is that the final scores are squared. Small confidences are shrunk, big confidences remain strong.

3.7 Post-evaluation note

I was informed by the organizers that squaring the values does not affect the final evaluation of the challenge. As such, only Submissions 1, 3 and 5 were considered during the official evaluation.

3.8 Euclidean distance to Cosine similarity computation

Since some of the models in the ensemble use Euclidean distance instead of Cosine similarity, its values are not bound to the $[0, 1]$ range. As such, it is necessary to convert these distances to something that resembles Cosine distance, so that the evaluation can be carried out smoothly.

In my approach, I have empirically selected a threshold $t = 1.75$ for the EvaTiny model and $t = 8.5$ for the MobileNetV2 model via manual trial-and-error experiments. These experiments consisted of manually running inference on various sample videos and images containing people's faces, using various values of this threshold. The purpose of the threshold is to determine when a face embedding is considered matching or not. Probes with distance lower than this threshold were identified to be the same as the gallery identity. I have adjusted and selected the thresholds in quite a restrictive way so that if a given embedding has the distance lower than the threshold, it is strongly expected that it is the matching identity. Distances above the threshold should be considered non-matching, but I did not want to strictly clip the values close to the boundary to zero.

During inference, distances to all subjects are computed. As a following step, both the scores and the threshold are divided by the maximum of the distances to normalize everything to the $[0, 1]$ range. The final score s_M for a model M, probe image x and a gallery identity i is then computed as follows:

$$s_M(x, i) = \begin{cases} max(1/(1 + dist(M(x), i)), 0.75), & \text{if } dist(M(x), i) \leq t_1 \\ max(w - dist(M(x), i), 0), & \text{otherwise} \end{cases}$$

where $w = 0.4$ in the case of EvaTiny and $w = 0.6$ in the case of MobileNetV2. It is a little complicated formula, but the idea behind it is that I want to have scores in range $[0.75, 1]$ for the distances that satisfy the *lower than threshold* condition, and $[0, w]$ values for the distances that do not satisfy the condition. See Figures 3 and 4 for graphs depicting example scenarios.

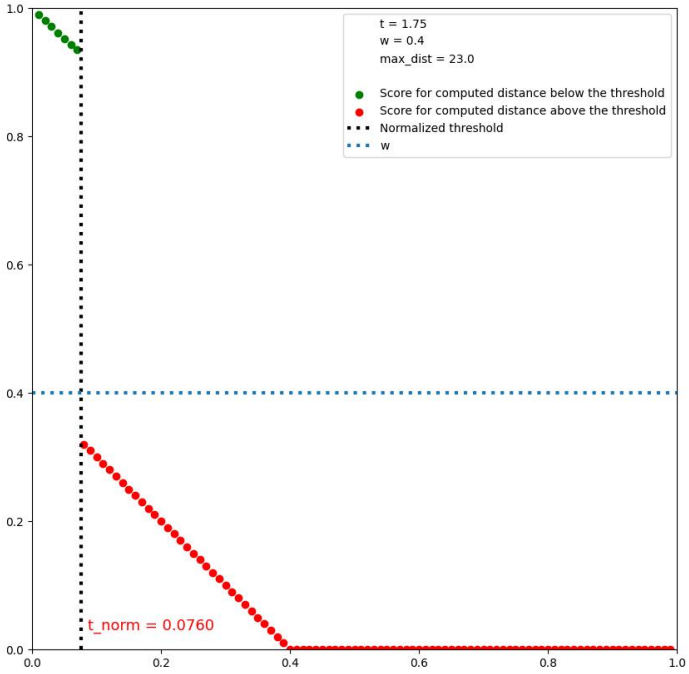

Figure 3: Graph of the score mapping function for the **EvaTiny** model. Here, $t = 1.75$, $w = 0.4$ and random $max\_dist = 23.0$

4 Known issues & Room for improvement

- **Design** - My submitted ensemble is not exactly the best - models are very similar and based mostly on the same architecture. The current design aims to be as seamlessly runnable as possible, due to time and resource constraints. I have tried training an additional EfficientNet-B0 model, however the training collapsed and the embeddings were nonsensical. As such, this submission consists of only models that worked after a brief manual inspection.

- **Reproducibility** - In the current state, no reproducibility is ensured. Data entries are sampled at random, initial weights for non-pretrained models are also initialized at random. Incorporating PyTorch Lightning framework into the system for further experiments could be a reasonable idea, as the framework can ensure reproducibility.

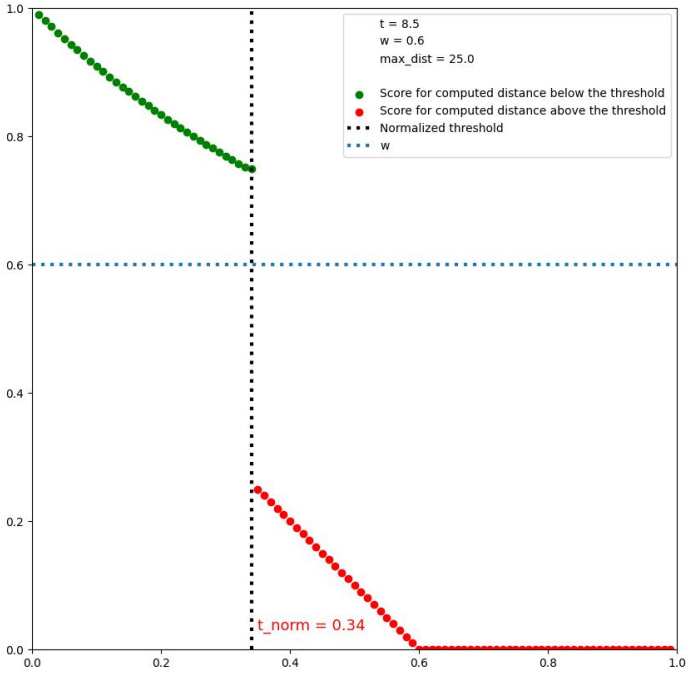

Figure 4: Graph of the score mapping function for the **MobileNetV2** model. Here, $t = 8.5$, $w = 0.6$ and random $max\_dist = 25.0$

- **Augmentations & image size** - There is only a small amount of augmentations used. In the initial phase, I aimed for rather simple but effective augmentations to make sure the system can work at least somehow. More aggressive augmentations could diminish the benefit of using various augmentations. Similarily to image size, a compromise has been done to ensure as fast training of new models as possible. Currently the environment is not fully prepared for models accepting different image sizes.
- **Evaluation** - There was no rigorous evaluation conducted during the development of the system prior to the submission. Everything relies on a small amount of manually carried out experiments.
- **Scalability** - Since the challenge is now concluded, any time constraints are lifted. Therefore, experiments with larger models are now possible.

5 Code

Code is available at `https://github.com/Fapannen/CyberuSentry/tree/main`. Models will not be released due to licensing limitations until the UCCS Dataset becomes publicly available.

References

[eva] https://huggingface.co/timm/eva02_small_patch14_224.mim_in22k.

[fac] https://github.com/timesler/facenet-pytorch.