Facial Landmarks Detection with Synthetic Dataset

Student: Chaoyi Hu

Student #: 301559775

Date: Dec 2022

Project link: https://github.com/zoehcycy/real-time-facial-landmark

Abstract

Synthetic human facial datasets offer many advantages such as controllable production and elimination of ethical concerns. In this project, we developed a facial landmarks model that trains on synthetic facial data and generalizes to real facial images.

The network architecture we adopted consists of 68 layers featuring a Resnet backbone. For the selection of loss function, we conducted a pair of experiments showing that adopting wing loss reduced the prediction mean error (ME) and improved the prediction success rate (SR) by 7.8% compared to using MSE loss.

We experimented with different portions of synthetic data in the training set. Trained with 100% synthetic data and tested on real images, the model achieved ME $\pm$ Std = 23.69 ± 8.71 and SR = 0.41 (ME threshold = 20.0). When trained with 60% synthetic data and 40% real data, the performance improved to ME $\pm$ Std = 18.64 ± 7.88 and SR = 0.60. However, decreasing the percentage of real data to 10% was shown to be detrimental to prediction accuracy.

To evaluate the potential of the model inferencing on consumer devices in real time, we took special notes to the model size and inference speed. Experimenting with a trimmed network 23.6% smaller in size compared to the full version, we found that it inferences 23.1% faster at the cost of a 29.0% drop in SR, demonstrating a trade-off between model size and performance.

Integrating the landmarks predictor with a face detector, we delivered a desktop demo that runs landmarks inferences in real-time on video inputs at approximately 20 FPS.

1 Introduction

Training with real-world images in facial landmarks detection projects is subject to ethic restraints such as privacy protection, as well as the high cost of manually labelling the facial landmarks. Synthetic human facial images generated using computer graphics offer many advantages, including controllable production, perfect annotations, scalability, and elimination of ethical concerns, thus have the potential to be an ideal substitute. However, subtle differences exist between synthetic and real-world images. For example, real-world images have richer background and texture while synthetic images are generally less photorealistic, synthetic images can generate finer labels at pixel-level, which is not achievable in manually labelled real images, etc. These differences together have produced the domain gap between synthetic and real-world images. Researchers have explored approaches to utilize synthetic data in face-related computer vision tasks. For example, Shrivastava, et al. proposed to refine synthetic images to make them look more real [1]. Ganin, et al. used domain-adversarial training, encouraging the model to ignore differences between different domains [2]. Wood, Et al. established a pipeline to procedurally generate photorealistic synthetic human facial images that can be used for training in face-related computer vision tasks, and presented promising test results matching real-world images [3]. Style-aggregated images produced using GAN has also been used to complement the original image as input to the landmarks model to improve prediction performance [4]. To sum up, the possibility of bridging the domain gap has been demonstrated yet the task remains challenging.

There has been an increasing demand for efficient models that are able to be deployed on consumer devices and run inferences in real time. Large pre-trained networks are impractical for such tasks where computational power is usually limited. For example, VGG-16 has a total of 138 million parameters and occupies over 552 MB of space [5]. Running such a huge network on a typical smartphone with 4 to 8 Gigabytes of RAM will be extremely inefficient. To perform evaluation of deep neural network at a rate of at least 30 times per second on consumer devices, the network needs to be highly efficient and compact. Series of light-weight models have been proposed to suit the increasing demand. SqueezeNet [6] features parameter compression in convolutional neural networks using Fire modules, and has been demonstrated to be able to run on low-power processing platforms such as smartphones and custom processors. Other lightweight networks include SqueezeNext [7], ShuffleNet [8], MobileNet [9], etc. Featuring compactness and high efficiency, these models are designed to be more suitable for real-time uses on consumer devices, aiming at smaller model size that is memory-efficient, and faster inferencing that is computationally efficient.

In this project, we developed a facial landmarks model that trains on synthetic data and generalizes to real facial images. Experiments were conducted to evaluate the prediction performance of the model and its potential in real-time application on consumer devices. By comparing the performances of models trained with different portions of synthetic data, we suggest that a proper mixture of synthetic and real data in the training set may help improve landmarks prediction accuracy and robustness. Evaluations on model sizes and inference speeds demonstrated a trade-off relationship between the size of a model and its performance. In the end, we present a desktop demo for real-time landmarks detection that runs inferences on laptop CPU at around 20 frames per second.

2 Method

Define the Input/Output. Taking 2D images as input, we first use a face detector to crop the facial area out of the entire image. The cropped image is then fed as input to the landmarks model which outputs the 3D landmark coordinates.

2.1 Face Detector

Figure 1 Employing the face detector for image cropping. The cropped facial images are fed as input to the landmarks model.

The face detector is used to crop the facial area(s) from the entire image. Feeding the cropped image as input keeps the landmarks model devoid of distractions from the background. Besides, the face detector allows applications to support landmarks prediction on multiple faces present in the frame.

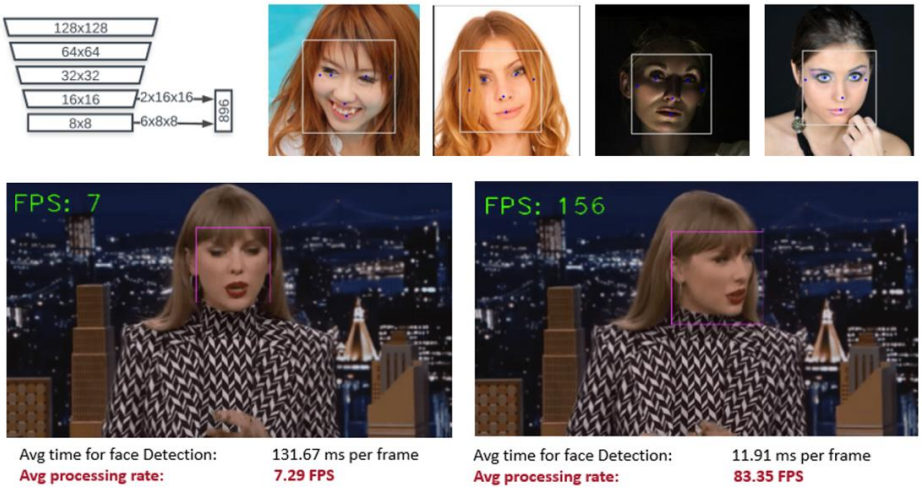

Figure 2 Top-left shows the BlazeFace [10] network architecture. To its right are BlazeFace detection results on images, with the box outlining the facial area and 5 key points annotated automatically. At the bottom, we provided screenshots from the tests of the Dlib classic face detector and the MediaPipe implementation of BlaceFace on video inputs. The inference speed is shown as Frame per Second (FPS) in green to the top-left of the screenshots.

We experimented with two different face detectors to compare their detection speed. The first one is based on classic HOG feature combined with a linear classifier, an image pyramid, and sliding window detection scheme, implemented in Dlib [11]. As is shown in Figure.x, left side of the paired screenshots at the bottom, this model was able to complete the detection in one frame with a single face at an average of 7.29 FPS. This is far from what we expect from an application that runs in real time.

We sought to improve the detection speed. The second model is based on BlazeFace [10], a lightweight face detector proposed by Google in 2019 tailored for mobile GPU inference, capable of running inferences at 200-1000+ FPS on flagship devices. The model architecture is a very compact feature extractor convolutional network related to MobileNet V1/V2 [9], shown on the top-left of Figure 2. We implemented this detector using the pre-trained network provided by MediaPipe. Shown on the top-right of Figure 2 are the test results on facial images. Test screenshots on the same input video is shown on the right side of the paired screenshots at the bottom, BlazeFace runs at a significantly improved speed of 83.35 FPS on average, peaking at over 150 FPS.

Based on the test results described above, in the pipelines below as well as in the demo application, we adopt the second face detector when face cropping from an image is needed. The MediaPipe implementation of

BlazeFace also provides a parameter to switch between different modes depending on the distance of a face to the camera, which can be adjusted in adaptation to specific videos. We choose closeup mode when the face is within 2 meters from the camera and distant mode otherwise.

2.2 Landmarks Model

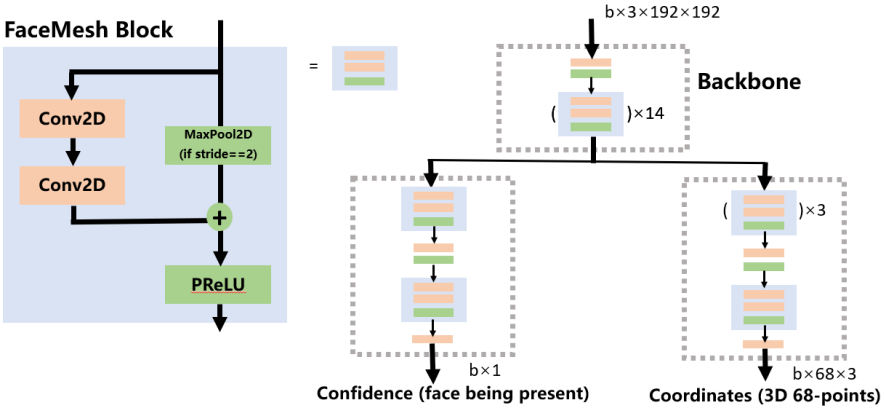

Figure 3 Illustration of the network architecture

Network Architecture. We adopt a convolutional network with a structure similar to the FaceMesh model used in MediaPipe [12]. The network starts with a ResNet backbone and diverges into two branches, one outputting the confidence level of prediction, and the other outputs predicted 3D landmark coordinates. FaceMesh block, consisting of two convolutional layers, adding the maxpool of the block input and activated by PReLU, is the basic component of this network. The backbone includes 14 such blocks, and the coordinates branch has 4, totaling 68 layers. The input image is resized to 192x192 with RGB channels, and the output is 68-point 3D coordinates.

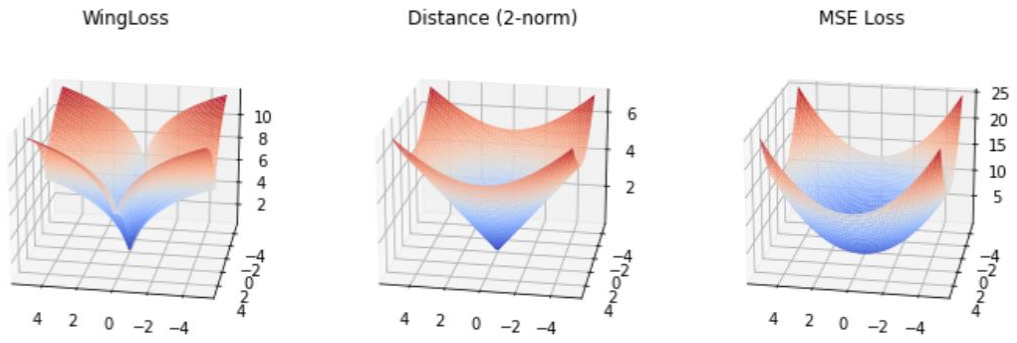

Figure 4 Surface plots of different loss function values with ground-truth at (0, 0). Params for Wing Loss: width=5, curvature=0.5.

Loss Function. In landmarks localization task, the loss function measures how close the predicted landmarks are to the ground-truth labels. A commonly used loss function in this case is the Euclidean distance from the label point. Feng, et al. proposed wing loss for robust landmarks localization in convolutional neural networks[13]. Wing loss introduces two parameters, width and curvature, to customize the amplification on errors near the ground-truth within the range of the width, emphasizing on the contribution of samples with small to medium error to the training process. Figure 4 provides the geometric intuition of wing loss compared to commonly used 2-norm loss and MSE loss. Near the ground-truth location, there is apparent change in the convexity of the loss surface, adding to the penalty within this range and forcing the points to converge to the exact location. In the following section, we present a pair of experiment results training respectively with MSE

loss and Wing loss and demonstrate the improvement of model performance brought by adopting wing loss function.

3 Experiments

3.1 Datasets

Figure 5 Sample images from the Face Synthetics Dataset [3]. 68 label points are shown.

The Face Synthetics Dataset [3]. This dataset consists of 100,000 Synthetic 2D human facial images with 70-point 2D landmarks positions. The images are generated procedurally, from 3d head parametric model, expression, hair, outfit, to texture, light, and environment, producing photorealistic images with rich diversity. In this project, we used 5,000 images randomly selected from the entire dataset.

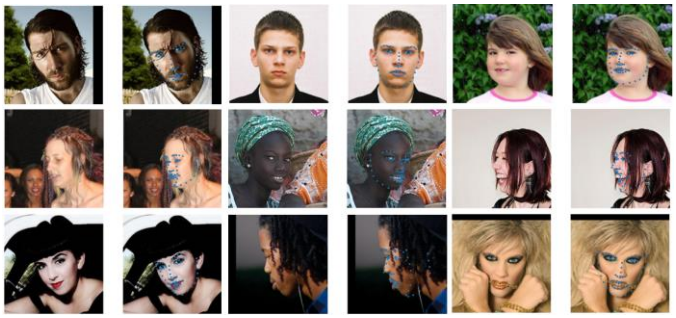

Figure 6 Sample images from AFLW2000-3D [14].

AFLW2000-3D [14]. This dataset consists of 2000 real human facial images with image-level 68-point 3D facial landmarks annotations. The images are in naturally occurring conditions or environments, with extreme poses, expressions and occlusions.

Note that the 70-point annotation system differs from the 68-point system in that the former has two extra points in the center of the left and right eyes.

3.2 Settings

Training configurations. The models were trained on Matpool cloud computing platform. Main hardware configurations include NVIDIA Tesla K80, 3 x Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz, and 8GB of RAM. Empirically, we chose Adam optimizer with a learning rate of 10e-6 and the weight decay was set to 0.0005. Each model was trained until the losses converge, with a batch size of 4. The average training time for each model was around 7 hours.

Training Set. We use 4000 images from the synthetic dataset and 2000 images from the real dataset for training. When training with a certain percentage of synthetic data, the entire training set contains 3000 images in total. For example, a training set with 90% synthetic data consists of 2700 synthetic images plus 300 real images.

Evaluation Metrics. We use mean error (ME) to measure the accuracy, which is defined as the mean Euclidean distance of each predicted landmark to its ground-truth. The results are reported in the form of ME ± Std, with lower Std indicating higher prediction robustness. Success Rate (SR) is the percentage of correct predictions, i.e., the percentage of predictions with ME below a set threshold. Besides the accuracy, the inference speed and model size are also reported as a metric to evaluate the efficiency of a model, indicating its potential for inference on consumer devices in real time.

Evaluation set. For each of the experiments, we evaluate the model on 300 synthetic images, and 300 real images, respectively.

3.3 Results

Training with different loss functions. As was mentioned in 2.2, we trained on the synthetic dataset using MSE loss and wing loss (width=5, curvature=0.5), respectively. Figure 8 shows the loss curves to the number of epochs, and the testing results are listed in Table 1. From this group of experiment results, we can see a 11.1% improvement in SR of the model trained with wing loss compared to the model trained with MSE loss, which suggests wing loss to be a better choice for loss function in this project. All models in the following experiments use wing loss with width=5 and curvature=0.5.

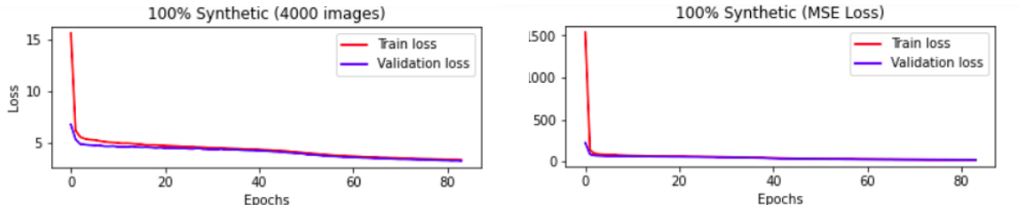

Figure 7 Loss curves. The left one shows the curve of the model trained with wing loss, and the right one is trained with MSE loss.

Table 1 Testing results of models trained with different loss functions

Training set	Loss	Testing set	Mean Error (ME) ± Std	Success Rate (SR)
Synthetic	MSE	Synthetic	20.45 ± 8.43	0.54
		Real	24.27 ± 9.53	0.38
	Wing Loss (w=5, c=0.5)	Synthetic	19.44 ± 9.13	0.60 (↑11.1%)
		Real	23.69 ± 8.71	0.41 (↑7.9%))

Training with different data. When trained purely on the real dataset, the model reaches SR=0.92 when tested on real data, but generalizes poorly to the synthetic dataset. In turn, when trained purely on the synthetic dataset, the model reaches SR=0.60 when tested on the synthetic dataset itself, and SR=0.41 when generalized to the real dataset. The model trained with 60% synthetic data outperforms the one trained purely on synthetic data on both the synthetic test set and the real test set, producing SR=0.66 and 0.60, respectively. However, when we decrease the percentage of real data to 10%, the SR on both test sets dropped significantly.

Figure 9 shows sample landmark predictions of models trained on pure real data on the left column, and pure synthetic data on the right column. Results suggest that the model trained on synthetic data alone generalizes poorly to real images, with incorrect landmark positions and mismatch in facial poses. Figure 10 shows sample predictions of the model trained on 60% synthetic data mixed with 40% real data. The predictions are not of very good quality yet the facial poses are generally correct, demonstrating a improvement in SR compared to the model trained with synthetic data alone.

Table 2 Testing results of models trained with different portions of synthetic data

Training set	Testing set	Mean Error (ME) ± Std	Success Rate (SR)
100% Real	Synthetic	27.04 ± 11.50	0.35
	Real	12.78 ± 5.31	0.92
100% Synthetic	Synthetic	19.44 ± 9.13	0.60
	Real	23.69 ± 8.71	0.41
60% Synthetic, 40% Real	Synthetic	18.18 ± 7.47	0.66
	Real	18.64 ± 7.88	0.60
90% Synthetic, 10% Real	Synthetic	29.20 ± 11.12	0.21
	Real	29.61 ± 11.19	0.19

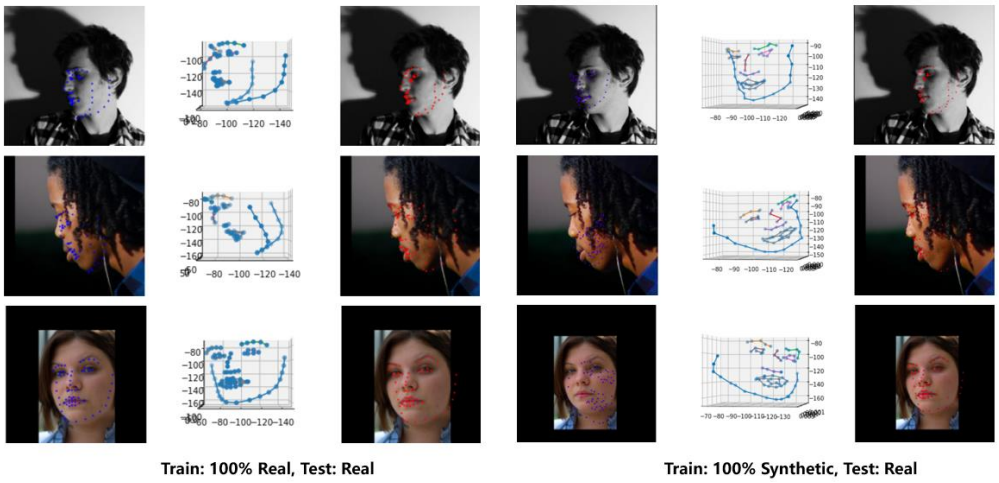

Figure 8 Left: Sample landmark predictions on real images when trained with 100% real data. Right: Sample landmark predictions on real images when trained with 100% synthetic data. Orange points denote ground-truth labels; Blue points are predictions.

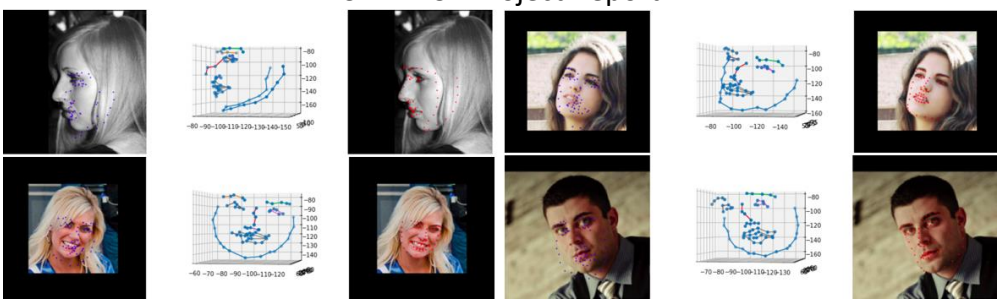

Figure 9 Sample landmark predictions on real images when trained with 60% synthetic data. Left: predictions overlayed on image; Middle: predictions in 3D coordinate system; Right: ground truth overlayed on image.

Training with a shallower network. To explore how the size and depth of a model affect its performance, we experimented with a shallower network with an architecture shown in Figure 11. Compared to the full version of the model, which has 14 FaceMesh blocks in the backbone and 4 in the coordinates prediction branch, the trimmed network has 9 FaceMesh blocks in the backbone and 3 in the coordinates prediction branch. We trained both networks on the synthetic dataset, and reported the results testing on the real test set. After trimming, the model is 23.6% lighter in size and inferences 23.1% faster than its full version. However, this improvement in speed comes with the cost of a drop in prediction accuracy. The SR has decreased by 29%, indicating that there might be a balance between model size and performance.

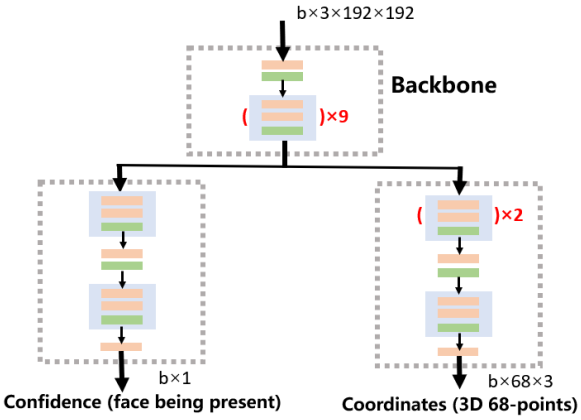

Figure 10 The trimmed network architecture

Table 3 Testing results of models with full and trimmed architectures

Architecture	**FaceMesh**	**FaceMesh (Shallow)**
Num of Layers	68	50
Num of Params	264317	201853 (↓23.6%)
Size	1.2 MB	0.9 MB
Mean Error (ME) ± Std	23.69 ± 8.71	25.43 ± 8.75
Success Rate (SR)	0.41	0.29 (↓29.0%)
Inference time per frame ± Std	7.03 ± 0.59 ms	5.41 ± 0.48 ms (↑23.1%)

Desktop demo. In our project proposal, we promised a desktop demo that runs landmarks predictions in real time. Implemented in Python with necessary libraries including OpenCV, Dlib and MediaPipe, the demo loads video file or webcam capture frames as inputs, runs each from through the face detector to obtain a crop of the facial area, or in some cases, multiple facial crops, which are next fed to the landmarks model and outputs 3D facial landmarks. The demo scripts are attached as project files in the submission package. Below shown are screenshots from the test sessions, with FPS value and landmarks displayed in green to the top-left. The top three screenshots are taken from webcam tests, showing landmark predictions consistent with head poses. On the bottom, two screenshots are taken from video inputs tests showing prediction performances in frames with single and multiple faces present. Inferencing on PC CPU, the model is able to run single-face prediction at an FPS at approximately 20. This value decreases proportionally to the number of faces present in each frame.

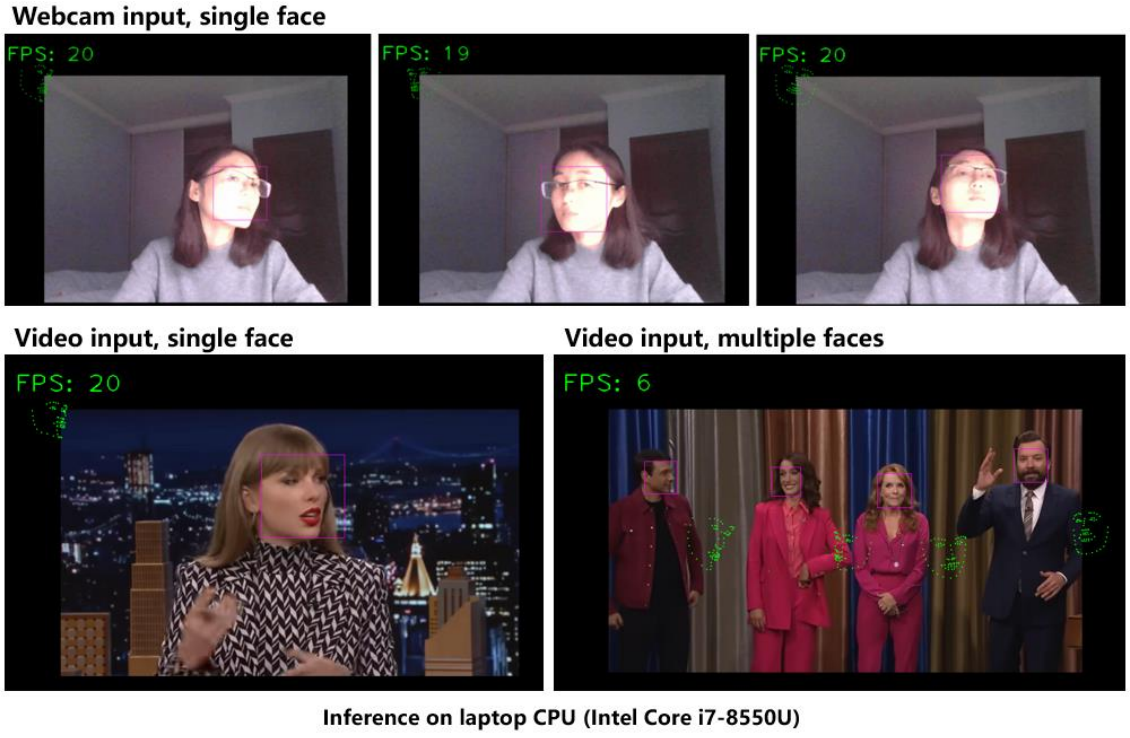

Figure 11 Demo screenshots

4 Conclusion and Prospects

In this project, we conducted experiments to train a facial landmarks model using synthetic facial data and evaluate its ability to generalize to real facial images. We employed a lightweight convolutional network architecture featuring ResNet backbone and wing loss for training. The choice of loss function was made based on experiment results showing that adoption of wing loss improved the prediction SR by 7.8%. Our results suggest that trained with 100% synthetic data, the model achieved ME $\pm$ Std = 23.69 ± 8.71 and SR = 0.41 (ME threshold = 20.0) on real test images. The performance improved to ME $\pm$ Std = 18.64 ± 7.88 and SR = 0.60 when trained with 60% synthetic data and 40% real data, and dropped significantly when the percentage of real data was further decreased to 10%. The results indicates that for the data mixing approach, a proper mixture of synthetic and real data in the training set might help improve the model performance compared to training with purely synthetic data. We further explored the impact of model size on its performance. A trimmed network 23.6% smaller in size compared to the full version inferences 23.1% faster at the cost of a 29.0% drop in SR. Results suggest a trade-off relationship between model inference speed and its prediction accuracy. Employing

the landmarks model trained in this project integrated with a facial detection module, we developed a desktop demo that runs landmarks inferences in real-time on webcam capture or video inputs at approximately 20 FPS.

Overall, we have fulfilled our project design in the project proposal with additional experiments on data mixing in the training set and comparison between MSE and wing loss functions. The prediction accuracy of our models cannot match state-of-the-art landmarks localization models, yet our results did provide some revelations subject to further exploration.

References

[1] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from Simulated and Unsupervised Images through Adversarial Training," Dec. 2016. [Online]. Available: https://arxiv.org/pdf/1612.07828

[2] Y. Ganin *et al.,* "Domain-Adversarial Training of Neural Networks," *Journal of Machine Learning Research 2016*, vol. 17. [Online]. Available: https://arxiv.org/pdf/1505.07818

[3] E. Wood *et al.,* "Fake It Till You Make It: Face analysis in the wild using synthetic data alone," Sep. 2021. [Online]. Available: https://arxiv.org/pdf/2109.15102

[4] X. Dong, Y. Yan, W. Ouyang, and Y. Yang, "Style Aggregated Network for Facial Landmark Detection,"

[5] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," Sep. 2014. [Online]. Available: https://arxiv.org/pdf/1409.1556

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," Feb. 2016. [Online]. Available: https://arxiv.org/pdf/1602.07360

[7] A. Gholami *et al.,* "SqueezeNext: Hardware-Aware Neural Network Design," *Design Automation Conference*, 2018. [Online]. Available: https://arxiv.org/pdf/1803.10615

[8] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," Jul. 2018. [Online]. Available: https://arxiv.org/pdf/1807.11164

[9] A. G. Howard *et al.,* "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017. [Online]. Available: https://arxiv.org/pdf/1704.04861

[10] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs," Jul. 2019. [Online]. Available: https://arxiv.org/pdf/1907.05047

[11] *Dlib Face Detector.* [Online]. Available: http://dlib.net/face_detector.py.html

[12] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs," Jul. 2019. [Online]. Available: https://arxiv.org/pdf/1907.06724

[13] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, and X.-J. Wu, "Wing Loss for Robust Facial Landmark Localisation with Convolutional Neural Networks," Nov. 2017. [Online]. Available: https://arxiv.org/pdf/1711.06753

[14] X. Zhu, X. Liu, Z. Lei, and S. Z. Li, "Face Alignment in Full Pose Range: A 3D Total Solution," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 1, pp. 78–92, 2019, doi: 10.1109/TPAMI.2017.2778152.