Internship Title: Improvement for gaze estimation with different neural networks

Internship Date: October 1st, 2021 ~ December 31, 2021

Internship Goal: Improve performance of the trained model in accuracy or reduce the running time during validation

Introduction:

One of the main research directions in Human Interaction laboratory, Reazon Holdings is to develop accurate "Eye-tracking" technology that can be used without constraints in real-time on mobile devices with reasonable cameras and processing abilities with minimal or no calibration. The preliminary results before I joined the team were pretty promising. “GazeCapture” dataset, which was collected under the project "Eye Tracking for Everybody" [1], was trained with CNN to detect eye corners and four points on the face that help in cropping (Figure 1), resulting in the four-times reduction in the CNN size and ten times reduction in training time while improving upon the previously published accuracy.

More and more prospective CNN models are being proposed, encouraging our team to investigate the feasibility of deploying them on “GazeCapture” dataset. Therefore, I am working as an intern to demonstrate results based on the original implementation, which is made up of a typical residual block. The internship work can be divided into three modules: 1) abstract original PyTorch implementation via PyTorch Lightning. 2) compare performance in accuracy with different CNN models. 3) compare running time with different CNN models.

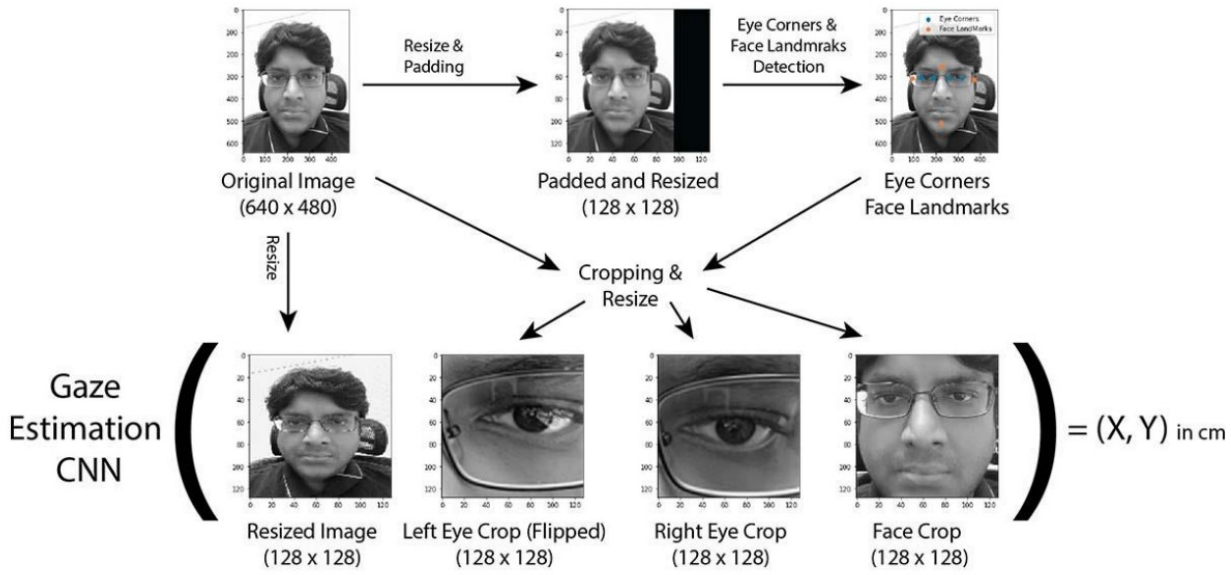

Figure 1: Data pipeline for gaze estimation CNN with typical residual block

Methods:

PyTorch Lightning is an open-source Python library that provides a high-level interface for PyTorch. Abstracting the implementation in PyTorch via PyTorch Lightning can make coding complex networks simple so that the codes can be more flexible and with fewer mistakes. In addition, we do not need to manually distribute CPU and GPU. The details of PyTorch Lightning can be referred via here: Official PyTorch Lightning Therefore, I demonstrate the abstraction here: GazeCapture PyTorch Lightning

We have investigated several neural networks, such as Capsule Network [2], ShuffleNet [3], and GhostNet [4]. Capsule Network was originally proposed for classification tasks [2] while this paper adapted it to gaze estimation problem [5], which is a regression task. The Capsule Network is not like the other two neural networks, which we can directly implement based on a typical residual block. Instead, it is composed of 4 modules for the gaze estimation tasks: convolution layers (can be seen as an encoder), primary capsule, gaze capsule, and decoder [5]. The main advantage of Capsule Network is that it does not use max pooling, which may discard a lot of pixel information. Instead, it utilizes dynamic routing to find the best connections between the output of one capsule and the inputs of the next layer of capsules, and updates coupling coefficients. Please refer to a clear blog for more descriptions: https://cezannec.github.io/Capsule_Networks/ Although we found out that we cannot deploy Capsule Network on "GazeCapture" dataset due to lacking the ground truth of gaze direction for classification in the gaze capsule, borrowing the idea of considering reconstruction loss with the autoencoder from Capsule Network can be one of the potential modification of the original model (GazeCapture PyTorch ReconstLoss).

ShuffleNet is an architecture utilizing two new operations, pointwise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy. From the visualization in Figure 2, we can observe that the modification is also straightforward on a residual block. The implementation can be referred from here: GazeCapture PyTorch Shuffle

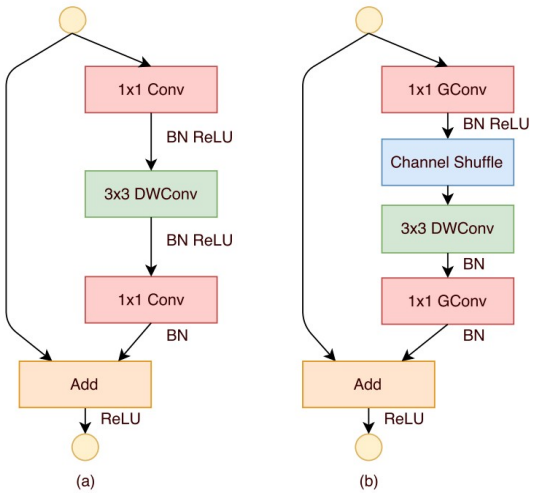

Figure 2: a) bottleneck unit ; b) ShuffleNet unit with channel shuffle

GhostNet is proposed to generate more feature maps from cheap operations so that the lightweight model can be easily established. It is claimed to achieve competitive performance with less computational cost and running time. In our gaze estimation task, the basic model is composed of 4 residual blocks. We can directly replace them with 4 Ghost bottlenecks. The implementation can be referred from here: GazeCapture PyTorch Ghost

Results:

We firstly show the comparison among multiple models with their average error for 30 epochs in Figure 3, indicating that considering reconstruction loss during training makes the performance even worse. In addition, the loss converges slower in ShuffleNet compared with ResBasicBlock (original). However, GhostNet can get the similar trend as ResBasicBlock does, which meets the expectation from GhostNet paper [4].

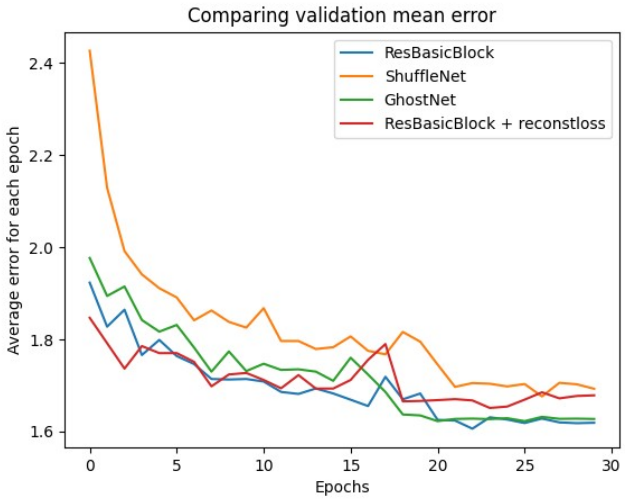

Figure 3: comparison of different models in average error

Since the trained model will be deployed on mobile devices, we are interested in the running time during validation of different trained models. GhostNet is claimed to achieve competitive performance while reducing running time. Therefore, we especially compare the running time in validation between baseline and GhostNet. In order to remedy the influence of bias from CPU usage, we test the validation results of each model 20 times. The result is shown in Figure 4, indicating the distribution of running time for a single image. The average running time for baseline is 2.4033 seconds while GhostNet only requires 2.1527 seconds, which reduces about 10.4 %.

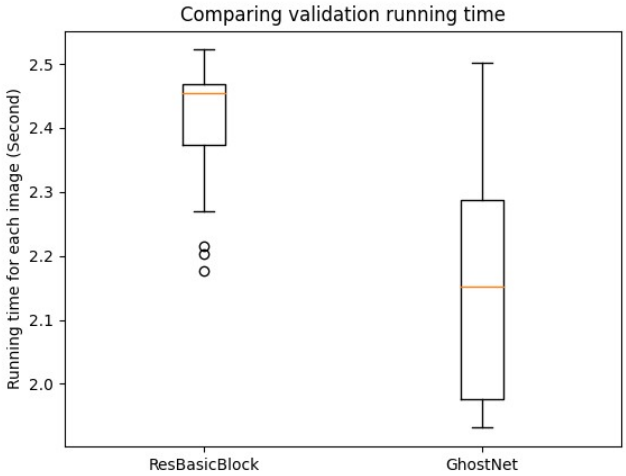

Figure 4: comparison between REsBasicBlock and GhostNet in running time during validation

Eventually, we do further hyperparameter tuning on GhostNet to get a better performance. We consider the basic learning rate: {0.0001, 0.0005, 0.001, 0.002} and a critical hyperparameter in GhostNet: the kernel size d × d of linear operations (i.e. the size of depthwise convolution filters) for calculating ghost feature maps with d in {1, 3, 5, 7}. The results are shown below in Figure 5 and 6.

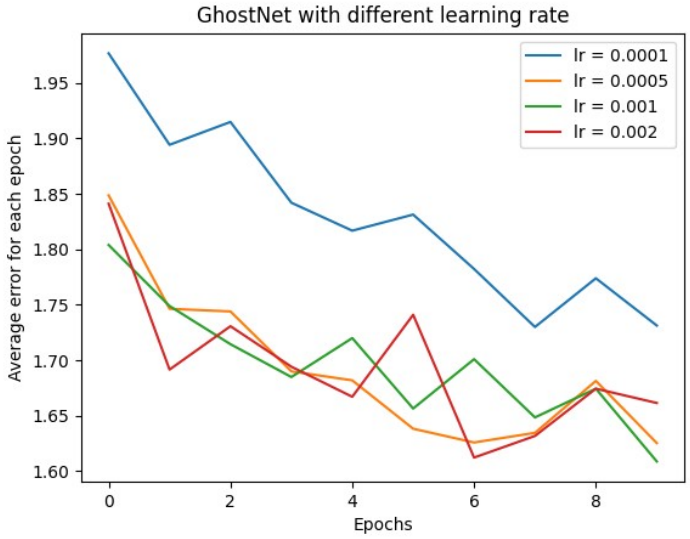

Figure 4: GhostNet with different learning rate

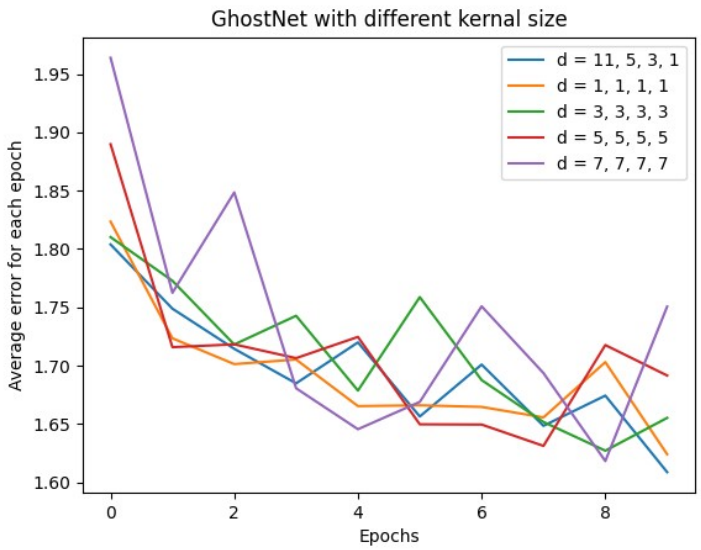

Figure 4: GhostNet with different kernel size

Discussion:

Here we compare multiple models to validate the feasibility of improving the current model. We aim to acquire the trained model with competitive performance while reducing the running time in validation. GhostNet is promising from our results. We do some hyperparameter tuning, but more tuning should be better. To be more specific, the hyperparameters should be critical, where m = n/ s is the number of intrinsic feature maps. In addition, it is interesting to set different kernel sizes for different blocks to see whether the performance can be improved significantly.

References:

[1] K. Krafka et al., "Eye Tracking for Everyone," ArXiv160605814 Cs, Jun. 2016, Accessed: Nov. 09, 2020. [Online]. Available: http://arxiv.org/abs/1606.05814

[2] S. Sabour et al., “Dynamic routing between capsules” In NeurIPS, pages 3859–3869, 2017. Available: https://arxiv.org/abs/1710.09829

[3] X. Zhang et al., “Shufflenet: An extremely efficient convolutional neural network for mobile devices”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. Available: https://arxiv.org/pdf/1707.01083.pdf

[4] K. Han et al., “GhostNet: More features from cheap operations”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. Available: https://arxiv.org/pdf/1911.11907.pdf

[5] B. Mahanama et al., “Gaze-Net: Appearance-based gaze estimation using capsule networks” In Proceedings of the 11th Augmented Human International Conference, Winnipeg, MB, Canada, 27–29 May 2020; pp. 1–4. Available: https://arxiv.org/pdf/2004.07777.pdf