2023 年臺灣國際科學展覽會
優勝作品專輯

作品編號	190030
參展科別	電腦科學與資訊工程
作品名稱	Face Pose Estimation using ResNet50 in the Metaverse
得獎獎項	
國　　家	Macau
就讀學校	Macau Pui Ching Middle School
指導教師	Thomas Lao
作者姓名	Bryan Lao

關鍵詞

作者照片

Abstract

Face pose estimation has many possible applications, ranging from driver attention measurement systems to applications in the metaverse, which this project will be focused on. Rather than using a more traditional landmark-to-pose method where the head pose is estimated via keypoints, our method trains a simple convolutional neural network, using the dataset 300W_LP, where the images are simply inputted into the network. The model is fitted with three fully connected layers that are linked to the each of the three Euler angles (yaw, pitch, and roll), alongside multiple loss functions, which improve the robustness of the network.

Introduction

Face pose estimation has been used since for several years now, with Murphy-Chutorian and Trivedi making a first assessment of the first techniques. More recently, different approaches have been proposed together with some annotated datasets for both training and testing those systems. The interest of the research community is mainly due to many applications that require or are improved by a reliable head pose estimation: face recognition with aliveness detection, human-computer interaction, people behavior understanding are some examples.

In previous work, head pose estimation was done via landmark-to-pose, where the head pose is computed with the use of landmarks on the target face and by solving the 2D to 3D correspondence problem with a mean human head model [3]. Some other methods may include other methods, such as using gaze estimation to, in turn, estimate head pose [9]. Gaze estimation methods are primarily categorized into model-based and appearance-based. Model-based approaches use a geometric model of the eye, usually required either high resolution images or a person specific calibration stage to estimate personal eye parameters. In contrast, appearance-based methods learn a direct mapping from intensity images or extracted eye features to gaze directions, thus being potentially applicable to relatively low-resolution images and mid-distance scenarios.[2]

Three main methods exist for automatically estimating head pose. Dynamic approaches, also called differential or motion-based approaches, track the position and orientation of the head through video sequences using pair-wise registration (for example, between two frames). Their strength is user-independence and higher precision for relative pose in short time scales. But they are typically susceptible to long time scale accuracy drift due to accumulated uncertainty over time. They also usually require the initial position and pose of the had to be set either manually or using a supplemental automatic pose detector. Static user-independent approaches detect head pose from single images without temporal information and without any previous knowledge of the user appearance. These approaches can be applied automatically without initialization, but they tend to return coarser estimates of the head pose. Static user-dependent approaches, also called keyframe-based or template-based approaches, use information previously acquired about the user (automatically or manually) to estimate the head position and orientation. These approaches are more accurate and suffer only bounded drift over time, but they lack the relative precision of dynamic approaches. They also require a procedure for learning the appearance of individual users.[1]

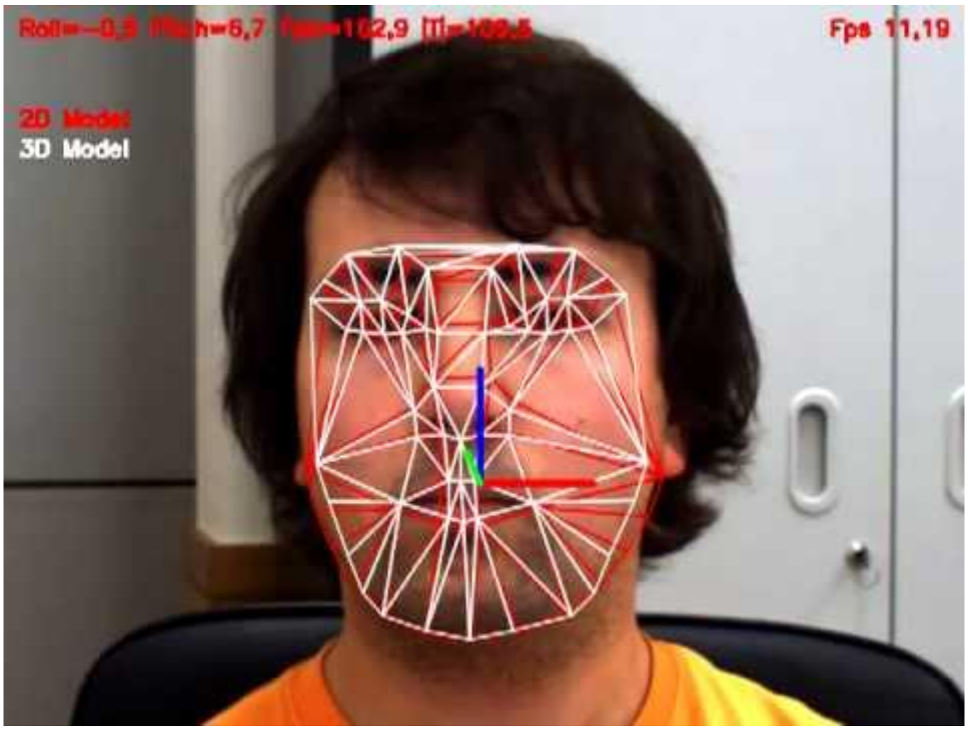

Figure 1. landmark to pose detection method

In the meantime, a large effort has been devoted to applications in the automotive field, such as monitoring drivers and passengers. This is because in the U.S alone, around 3,100 people were killed and about 424,000 were injured in car crashes involving a distracted driver in 2019 [4]. This accounts for approximately 8.1% of all car crash fatalities in that year. A study from the Virginia Tech Transportation Institute and the National Highway Traffic Safety Administration in 2006 found that, almost 80% of crashes and 65% of all near crashes involved some form of driver inattention within three seconds of the crash [5]. The same study also stated that reaching for a moving object increased the risk of a car crash nine times. This has been noticed by car manufacturers such as Tesla, Ford, General Motors, et cetera to use face pose estimation to monitor the attentiveness of the driver. However, these systems have faults, for example, Tesla's driver attention monitoring camera was found out to still allow the driver to continue using autopilot despite the camera being obscured [6]. In addition, current systems of face pose detection (landmark-to-pose methods) are quite unreliable under low-resolution imaging.

A new application for head pose estimation can be found in VR headsets, as they are quite heavy, which may dissuade some to use VR products for continuous use. As such, head pose estimation algorithms can replace the current systems in place used to predict avatars in virtual games and environments.

This paper revolves around using a convolutional network, rather than landmark, or keypoint-based methods, due to problems that may occur during the two-step process. Firstly, insufficient keypoints will lead to the failure of pose recovery. Second, the quality of the 3D model has an impact on the accuracy of the pose estimation. It can be very computationally expensive to adapt a generic head model (which may introduce errors) onto a participant and requires significant amounts of data.

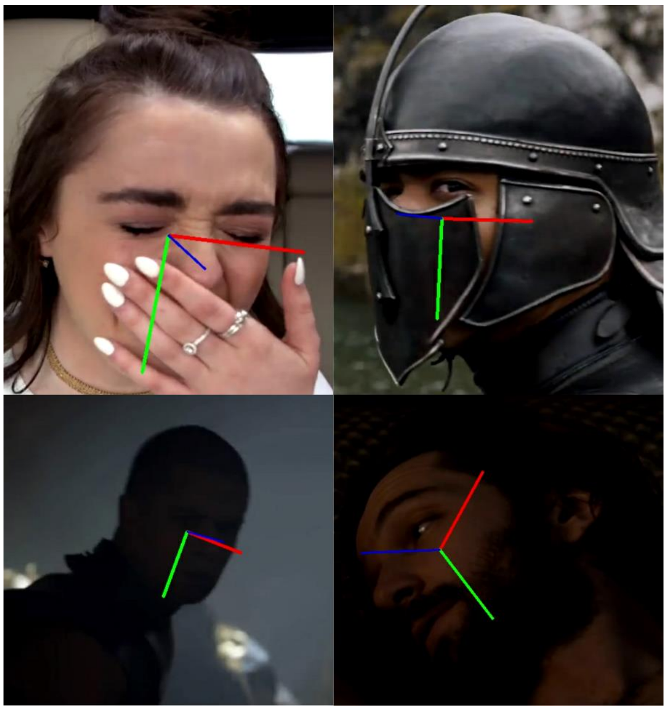

Figure 2. Pose detection using method.

Together with the estimation of the upper-body and shoulder pose, head monitoring is one of the key technologies required to set up autonomous driving cars, human-car interaction for entertainment, and driver's attention measurement.

The purpose of this paper is to introduce a new way of working on head pose estimation, by introducing a multiple loss function via three fully connected layers, that are in turn connected to a residual neural network. This new network is proposed to be faster and more accurate than other models.

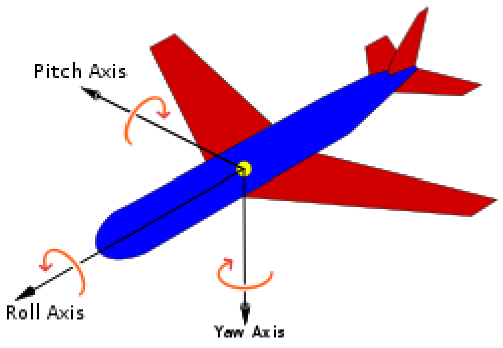

Figure 3. euler angles.

Methodology

This chapter mainly focuses on the methods on which this project was done, mainly through ResNet, a convolutional neural network with additional items.

2.1 ResNet

ResNet (Residual Neural Network), a type of network that mainly focuses on eliminating the issue of a degradation problem in deep neural networks. Degradation problem occurs when a model gets deeper, the gradient of the variables from shallow layers is unable to be propagated to the deeper layers. ResNet removes this issue by implementing a "identity shortcut connection", where it skips one or more layers, thereby also removing the possibility that exploding or vanishing gradient will happen. It works by setting $H(x)=F(x) + x$ as its desired underlying mapping. The stacked non-linear layers are then fitted onto $F(x)$. The original mapping can now be recast as $F(x) + x$. As shown in Figure 4, the shortcut connection, x, is added to $F(x)$ to obtain the desired mapped of $H(x)$. In essence, the shortcut connection preserves the original input. There are two distinct

ResNet designs: a basic block, and a bottleneck design. A basic block is just a regular design, but with the "identity shortcut connection". Bottleneck design, however, is designed to be used in very deep networks due to computational considerations. As Figure 5 shows, it changes the 256 channels into 64, thus reducing the computational power needed to train deeper networks.

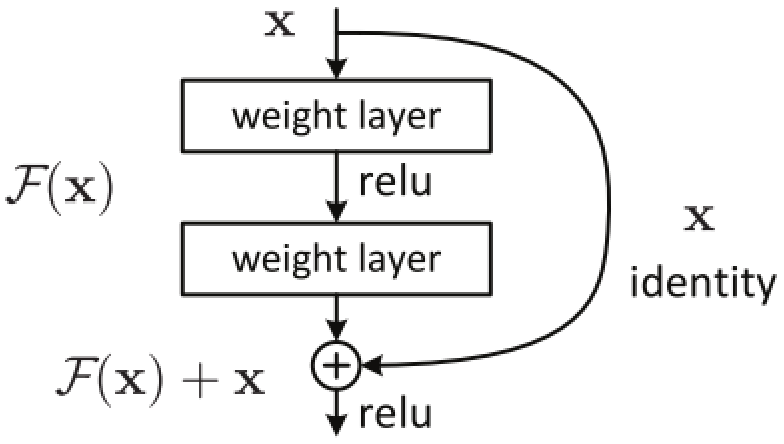
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 4. Identity shortcut connection

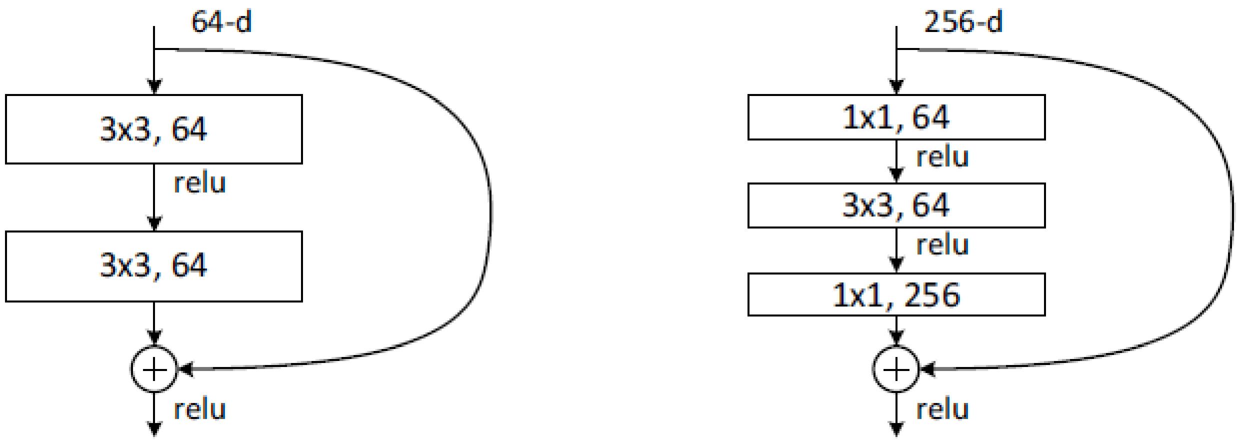

Figure 5. Basic block(left) and bottleneck block(right)

2.2 Convolution Neural Network

CNNs, or Convolutional Neural Networks, are made up of three layers; convolutional, pooling, and fully connected. The convolutional layer is the core building block of the CNN, and the majority of computation occurs here. It requires a few components, the input data, the filter, and the feature map. The filter is a two dimensional array of weights and is generally a 3x3 (or odd numbered) matrix. The way it works is by applying it on an area of the image, and the dot product is calculated between the input pixels and the filter. The dot product is then fed into the output array, and the entire process is repeated until the kernel has swept across the entire image. The final output from the dot products of the input is known as a feature map.

The pooling layer is identical to the convolutional layer, in that the pooling operation sweeps a filter across the entire input, but this filter does not have any weights. The kernel applies an aggregation function the values within the field, populating the output array. There are two types of pooling: max pooling, and average pooling. Max pooling selects the pixel with the highest value to send to the output array. This approach is used more frequently than average pooling. Average pooling, as the name implies, calculates the average value within the field to send to the output array. A lot of information is lost in the pooling layer, but it also comes with some benefits, such as reduced complexity, improved efficiency, and limits the risk of overfitting.

The fully connected layer, as the name indicates, is fully connected. Each node in the output layer connects directly to a node in the previous layer, whilst the values on the input image are not directly connect to the output layer. This layer performs classification based on thcxz e features extracted through the previous layers and their different filters. The previous two layers used rectified linear functions (ReLU), while fully connected layers usually use Softmax functions to classify inputs. ReLU is a non-linear function that will output the input directly if it is positive; any other case will be treated as a zero.

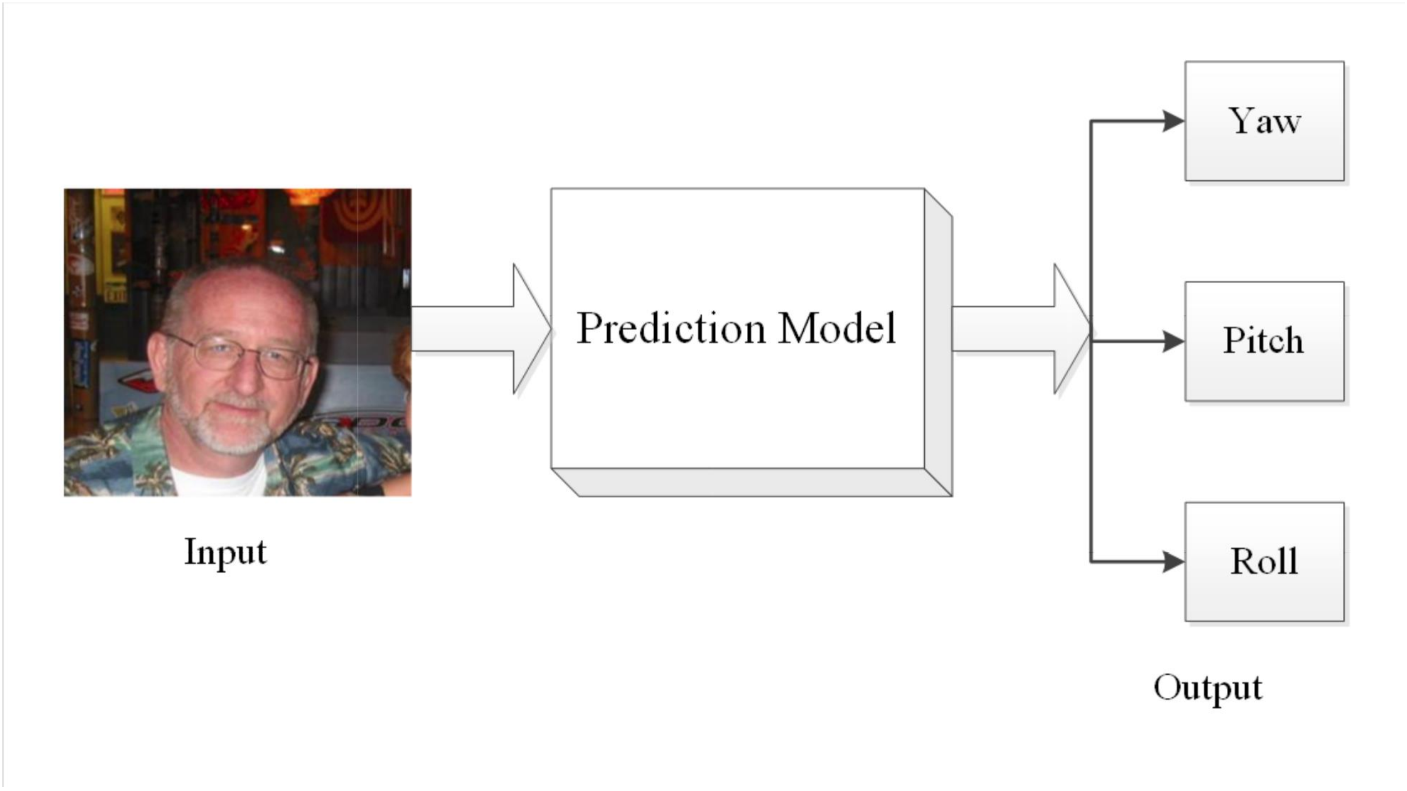

Figure 6. simple framework of prediction model

2.3 Multiple loss functions

Normally, in other head pose algorithms, usually only one type of loss function is used, either classification or regression. In contrast, three separate loss functions have been used, each for every Euler angle. The losses are a combination of binned pose classification and regression. By using a Softmax layer, cross-entropy, and MSE loss, the network learns to predict the pose in a very reliable manner. Each Euler angle has a cross-entropy loss, in order to back propogate signals which improves learning. This is shown in Figure 3.

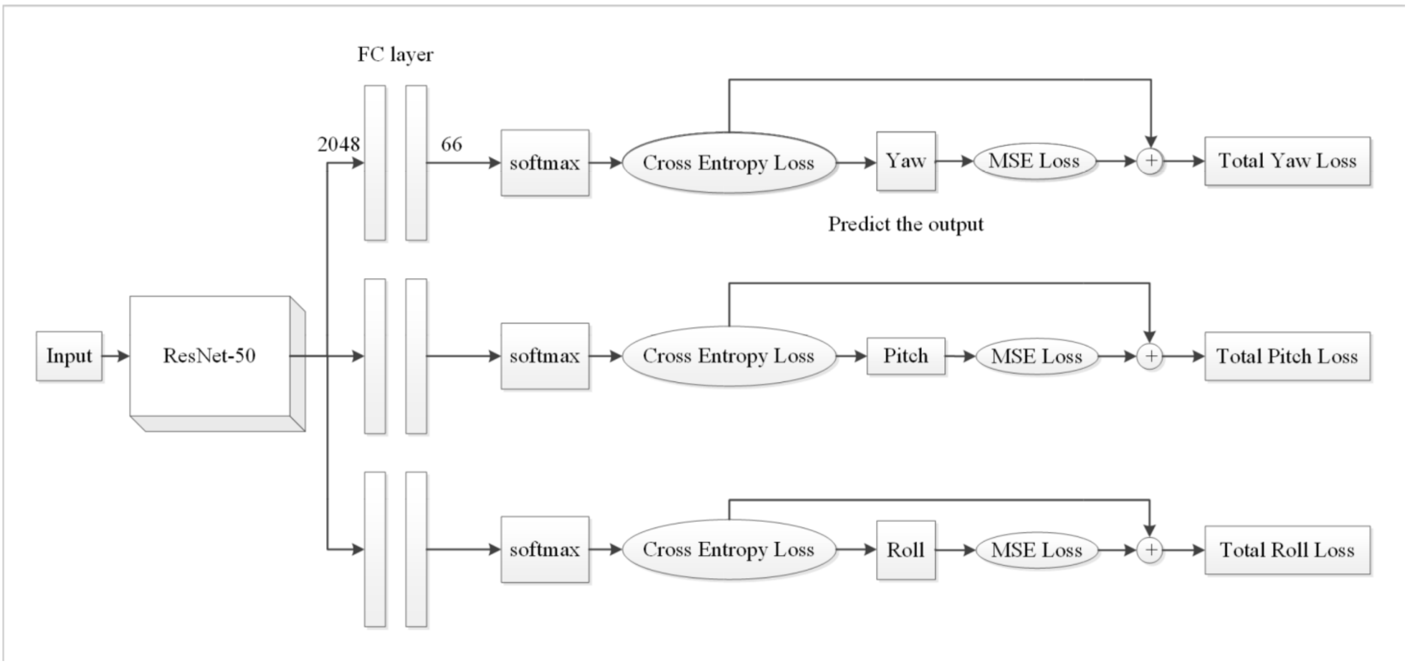

Figure 7. multiple loss functions

Section Three

3.1 Experimental design and dataset

3.1.1 Experimental design

This chapter will focus on the dataset and model structure used for the experiments as well as the training process. The model uses 300W_LP for both training and testing purposes; for each image, the three Euler Angles are used as classification labels and the specific values are used as regression labels, and the model outputs the three Euler angles of the image. Testing will be done via using a different part of the same dataset, by inputting the images onto the model, and by checking the difference between the correct Euler angle value, and the predicted amount by the model.

We use some classical networks (such as AlexNet and LeNet) to replace the core network, ResNet, to select the optimal model by comparing the performance of the models. This will be done in two parts, first by getting the training loss rate of the three Euler angles, and then by testing the model on the 300W_LP dataset, where the test error is then calculated. We use the same network structure, the same training method, and only change the loss function one using a single loss loss function and one using a multiple loss function to verify the effectiveness of our innovation points.

3.1.2 Dataset

300W_LP is based on another dataset, the 300W dataset. It only contains 600 outdoor and indoor images with human faces. As opposed to other datasets, 300W has more unusual expressions, such as “scream” and “surprise”, rather than more orthodox “neutral” or “happy” expressions. 300W_LP (Large Pose) is based on the original 300W images but increases the size to 61,225 samples by creating variants of the images with different poses. it also includes several other databases, such as AFW, LFPW, HELEN, IBUG, and XM2VTS.

3.2 Experiment procedure

3.2.1 Dataset preprocessing

The image is resized into 240 pixels, and then the image will be randomly cropped at a random location, with a size of 224. Normalization will then be done on the image, using means of 0.485, 0.456, 0.406, and a sequence of standard deviations of 0.229, 0.224, 0.225.

3.2.2 Experiment platform

The training was ran on a Intel Core i7-10750H, and on a Nvidia Geforce RTX 2070 Super. The python version used is 3.8, the pytorch version used is 1.12.1(GPU enabled). The training was done primarily with a GPU.

3.2.3 Model

Table I: Model layer layout

Layer	Output shape	param	
Conv2d-1	[-1, 64, 120, 120]	9,408	
BatchNorm2d-2	[-1, 64, 120, 120]	128	
ReLU-3	[-1, 64, 120, 120]	0	
MaxPool2d-4	[-1, 64, 60, 60]	0	
Conv2d-5	[-1, 64, 60, 60]	4,096	×2
BatchNorm2d-6	[-1, 64, 60, 60]	128	
ReLU-7	[-1, 64, 60, 60]	0	
Conv2d-11	[-1, 256, 60, 60]	16,384	
BatchNorm2d-12	[-1, 256, 60, 60]	512	
Conv2d-13	[-1, 256, 60, 60]	16,384	
BatchNorm2d-14	[-1, 256, 60, 60]	512	
ReLU-15	[-1, 256, 60, 60]	0	
Bottleneck-16	[-1, 256, 60, 60]	0	
Conv2d-17	[-1, 64, 60, 60]	16,384	
BatchNorm2d-18	[-1, 64, 60, 60]	128	
ReLU-19	[-1, 64, 60, 60]	0	
Conv2d-20	[-1, 64, 60, 60]	36,864	×2
BatchNorm2d-21	[-1, 64, 60, 60]	128	
ReLU-22	[-1, 64, 60, 60]	0	
Conv2d-23	[-1, 256, 60, 60]	16,384	
BatchNorm2d-24	[-1, 256, 60, 60]	512	
ReLU-25	[-1, 256, 60, 60]	0	
Bottleneck-26	[-1, 256, 60, 60]	0	×2
Conv2d-37	[-1, 128, 60, 60]	32,768	
BatchNorm2d-38	[-1, 128, 60, 60]	256	
ReLU-39	[-1, 128, 60, 60]	0	
Conv2d-43	[-1, 512, 30, 30]	65,536	
BatchNorm2d-44	[-1, 512, 30, 30]	1,024	
Conv2d-45	[-1, 512, 30, 30]	131,072	
BatchNorm2d-46	[-1, 512, 30, 30]	1,024	
ReLU-47	[-1, 512, 30, 30]	0	
Bottleneck-48	[-1, 512, 30, 30]	0	
Conv2d-49	[-1, 128, 30, 30]	65,536	
BatchNorm2d-50	[-1, 128, 30, 30]	256	
ReLU-51	[-1, 128, 30, 30]	0	
Conv2d-52	[-1, 128, 30, 30]	147,456	×3
BatchNorm2d-53	[-1, 128, 30, 30]	256	
ReLU-54	[-1, 128, 30, 30]	0	
Conv2d-55	[-1, 512, 30, 30]	65,536	
BatchNorm2d-56	[-1, 512, 30, 30]	1,024	
ReLU-57	[-1, 512, 30, 30]	0	
Bottleneck-58	[-1, 512, 30, 30]	0	
Conv2d-79	[-1, 256, 30, 30]	131,072	×2

BatchNorm2d-80	[-1, 256, 30, 30]	512	
ReLU-81	[-1, 256, 30, 30]	0	×2
Conv2d-85	[-1, 1024, 15, 15]	262,144	
BatchNorm2d-86	[-1, 1024, 15, 15]	2,048	
Conv2d-87	[-1, 1024, 15, 15]	524,288	
BatchNorm2d-88	[-1, 1024, 15, 15]	2,048	
ReLU-89	[-1, 1024, 15, 15]	0	
Bottleneck-90	[-1, 1024, 15, 15]	0	
Conv2d-91	[-1, 256, 15, 15]	262,144	
BatchNorm2d-92	[-1, 256, 15, 15]	512	
ReLU-93	[-1, 256, 15, 15]	0	
Conv2d-94	[-1, 256, 15, 15]	589,824	
BatchNorm2d-95	[-1, 256, 15, 15]	512	
ReLU-96	[-1, 256, 15, 15]	0	×5
Conv2d-97	[-1, 1024, 15, 15]	262,144	
BatchNorm2d-98	[-1, 1024, 15, 15]	2,048	
ReLU-99	[-1, 1024, 15, 15]	0	
Bottleneck-100	[-1, 1024, 15, 15]	0	
Conv2d-141	[-1, 512, 15, 15]	524,288	
BatchNorm2d-142	[-1, 512, 15, 15]	1,024	×2
ReLU-143	[-1, 512, 15, 15]	0	
Conv2d-147	[-1, 2048, 8, 8]	1,048,576	
BatchNorm2d-148	[-1, 2048, 8, 8]	4,096	
Conv2d-149	[-1, 2048, 8, 8]	2,097,152	
BatchNorm2d-150	[-1, 2048, 8, 8]	4,096	
ReLU-151	[-1, 2048, 8, 8]	0	
Bottleneck-152	[-1, 2048, 8, 8]	0	
Conv2d-153	[-1, 512, 8, 8]	1,048,576	
BatchNorm2d-154	[-1, 512, 8, 8]	1,024	
ReLU-155	[-1, 512, 8, 8]	0	
Conv2d-156	[-1, 512, 8, 8]	2,359,296	
BatchNorm2d-157	[-1, 512, 8, 8]	1,024	×2
ReLU-158	[-1, 512, 8, 8]	0	
Conv2d-159	[-1, 2048, 8, 8]	1,048,576	
BatchNorm2d-160	[-1, 2048, 8, 8]	4,096	
ReLU-161	[-1, 2048, 8, 8]	0	
Bottleneck-162	[-1, 2048, 8, 8]	0	
AvgPool2d-173	[-1, 2048, 1, 1]	0	
Linear-174	[-1, 66]	135,234	
Linear-175	[-1, 66]	135,234	
Linear-176	[-1, 66]	135,234	

ResNet50 is the core network of this model. The overall layout of the model is shown in Table I. x[number] means the layers in that bracket are multiplied by that number. The first convolutional layer has a kernel size of 7, stride of 2, and a padding of 3. The max pooling layer has a kernel size of 3, stride of 2, and padding of 1. These values represent the first sublayer of each respective layer. The middle convolutional layer has a kernel size and stride of 1. The average pooling has a size of 7. The input of the fully connected layer is 2048 dimensions, and the output is 66.

3.2.4 Training process

The first step in the procedure is to download the dataset online. The second step is to do pre-processing on the dataset, such as making a list of all the required training images from the 300W_LP dataset. Images from the dataset have been altered from the original 300W dataset, where the faces are distorted in order to achieve a different pose from the original. Pre-processing also includes the content presented in 3.2.1. The processed dataset is set as a batchsize of 16, and then put into the model for training. The final step is use Xavier initialization and Adaptive Moment Estimation (Adam), a gradient descent optimization algorithm. The learning rate is set to 0.0001, the epoch is set as 150, and multiple loss functions stated in 2.3, after which training on the model can start.

Section Four

This section mainly focuses on the comparison with other models, such as AlexNet, LeNet, and Model1, a copy of AlexNet without the ReLU function. In addition, ResNet-50 with singular loss functions is included in the training and testing. Resnet-50 with multiple loss functions is shown to perform better than all other networks in both experiments.

4.1 Experiment results and analysis

ResNet is replaced with AlexNet, LeNet, and Model1 as the core network, which are compared with each other in order to select the most optimal model. To ensure fairness, all variables are kept the same through all models during training. The training results are shown below. AlexNet is a CNN structure that contains eight layers; the first five are convolutional layers, some with max-pooling layers. The last three layers are fully connected layers, and it uses ReLU as an activation function [7]. LeNet is one of the oldest CNNs, first made in 1989, by Yann LeCun et al. in Bell Labs, where the first practical applications were applied using the backpropagation algorithm. LeNet consists of seven layers, which is composed of convolutional and pooling layers sandwiched together [8]. Model1 is essentially AlexNet, but without the ReLU activation function. It is made specifically for the purpose of comparison with the other models in this training experiment.

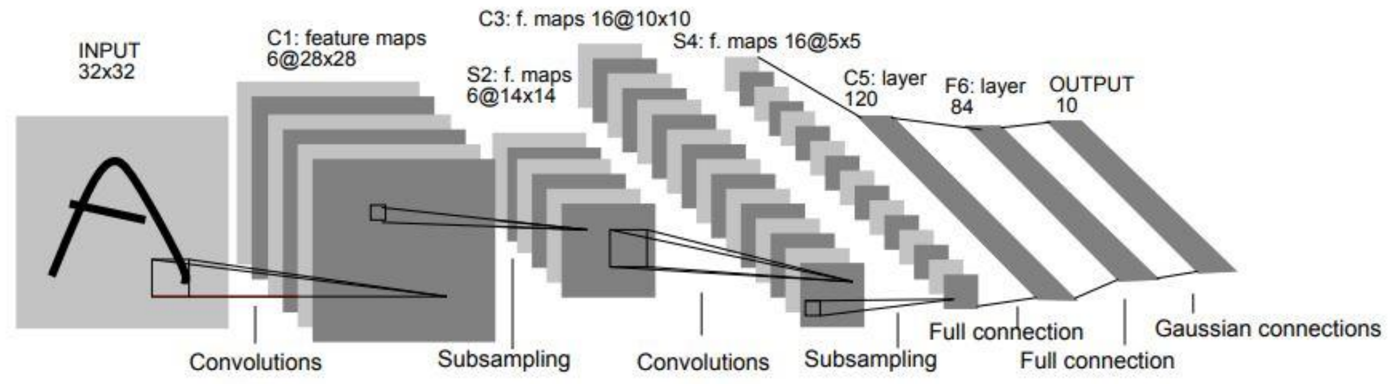

Figure 8. LeNet

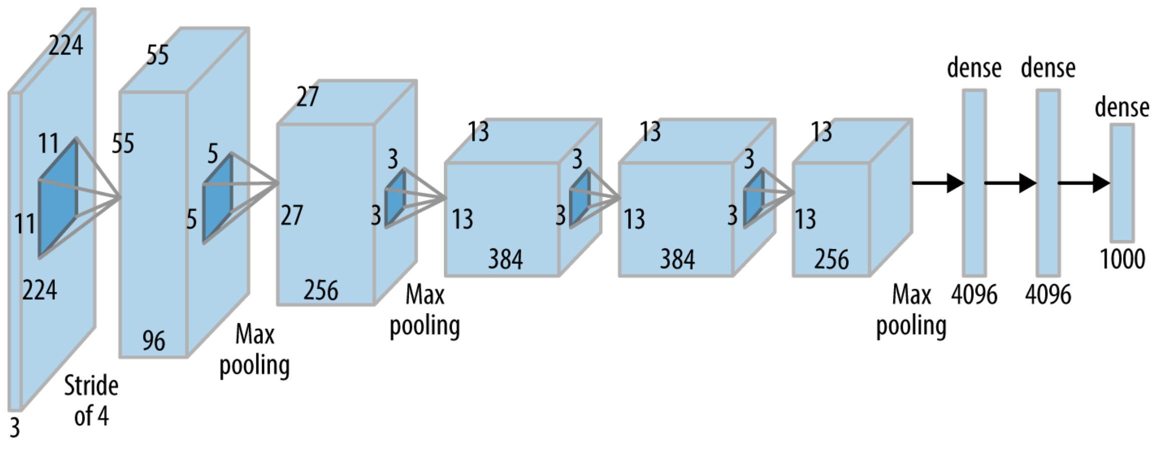

Figure 9. AlexNet

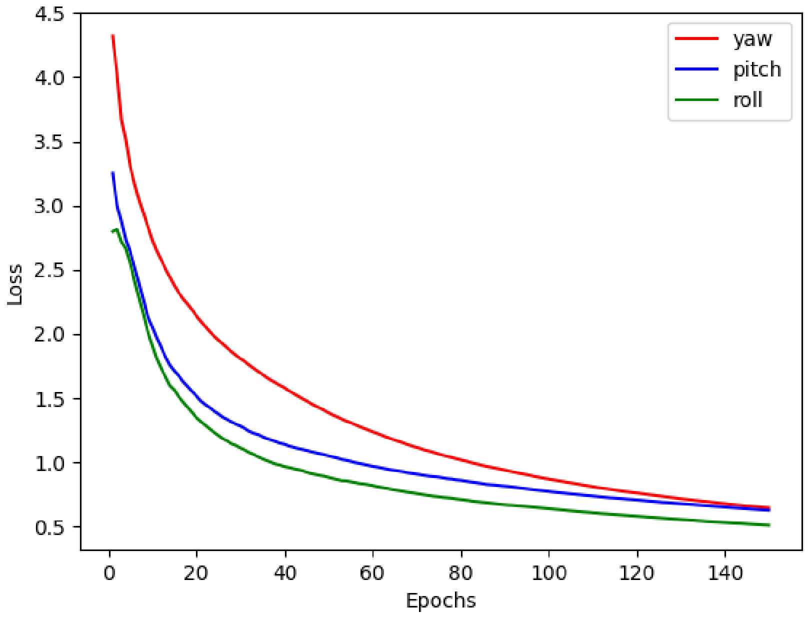

Figure 10. ResNet-50 loss rate during training

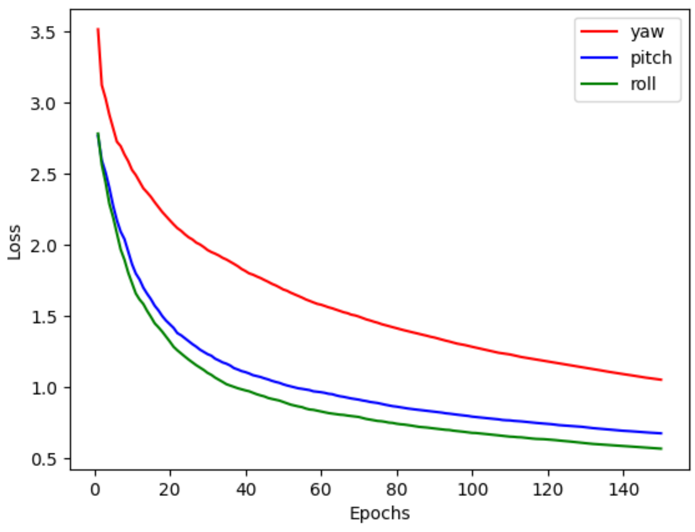

Figure 11. AlexNet loss rate during training

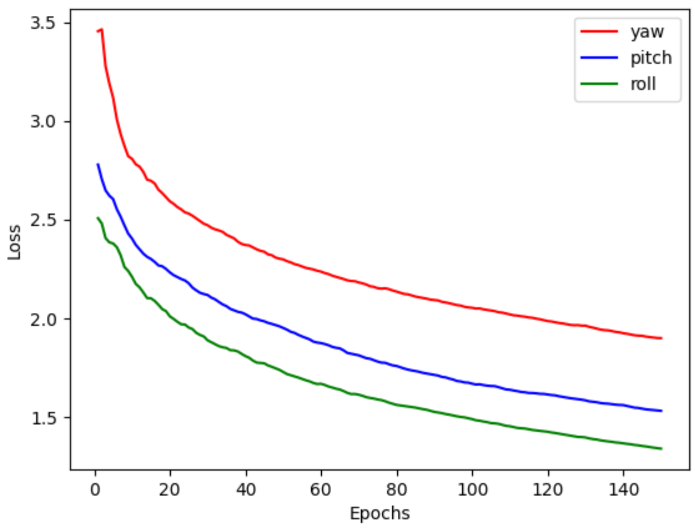

Figure 12. LeNet loss rate during training

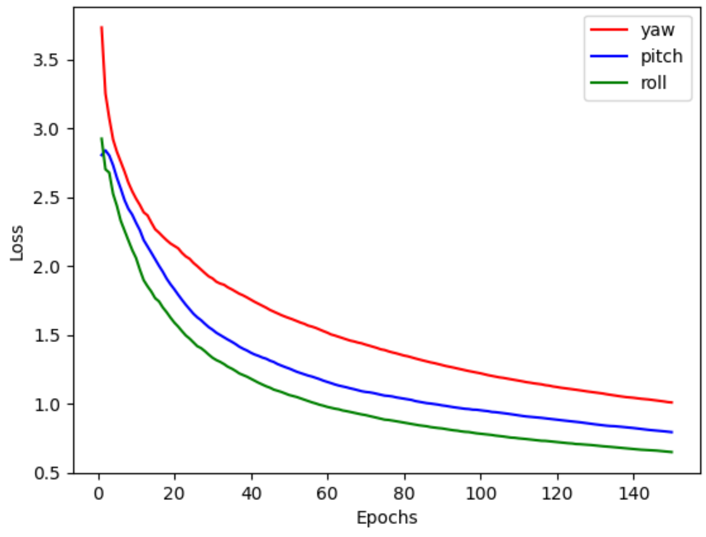

Figure 13. Model1 loss rate during training

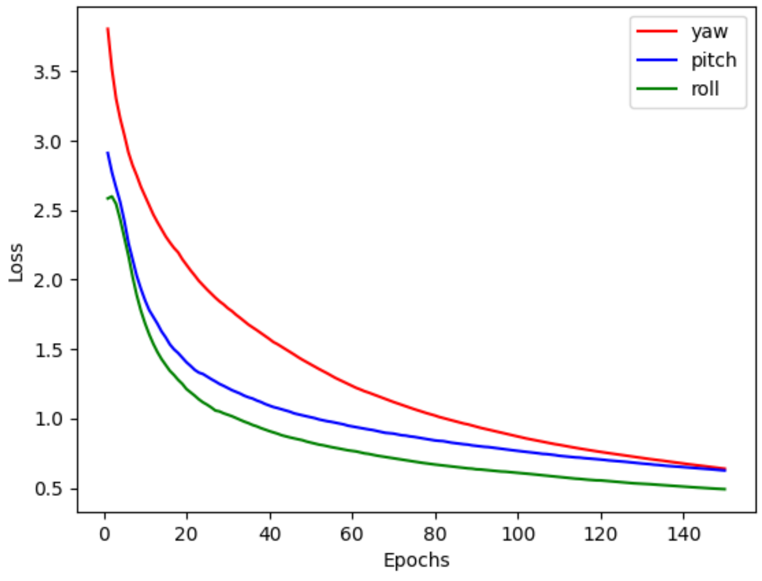

Figure 14. ResNet50 with singular loss function

The models are trained with an epoch of 150, with each line corresponding to its own Euler angle. ResNet-50 shows the lowest loss rate out of all the models, with LeNet showing the highest. AlexNet slightly beats out Model1 in loss rate, due to the included ReLU activation function. The loss rate of ResNet-50 is initially high compared to the other models but drops off significantly after the 20th epoch. The other models share this characteristic but is not as significant as the change from ResNet-50. Decrease of loss rate is also noticeably smoother in ResNet-50 with multiple loss functions, a feature exclusive only to it. ResNet-50 with singular loss function has the same loss rate at the end, but has a more rough cure to curve to it. Yaw has the highest loss rate, with pitch in-between and roll being the easiest to predict.

In short, this experiment shows that the method used by this project, ResNet-50, has the lowest loss rate out of the four selected training models, making it the most reliable in predicting Euler angles.

4.2 Testing process and analysis

The testing of the model is done via inputting data from the dataset 300W_LP. It first starts by loading the snapshot of the model, after which the dataset is processed via batch normalization and resizing the data. Images from the dataset are then added onto a variable which counts the number of images tested. It then saves a mean absolute error of the difference between the predicted angles and the actual angles, which is added onto a total error count. Testing is concluded when the dataset is finished, and the program outputs the test error of the model in the three Euler angles.

Table 2. Test error in degrees of the model on the 10,414 test images

Model	Yaw	Pitch	Roll
ResNet-50	1.5670	1.5465	1.5427
AlexNet	1.6808	1.5961	1.5577
LeNet	2.4538	2.6342	2.4024
Model1	1.7721	1.6376	1.6221
ResNet-50 w/ multiply loss only	3.6381	6.8251	4.3999
ResNet-50 w/ classification loss only	4.0731	7.3569	4.9830

ResNet-50 is shown to be noticeably better in performance compared to the others, most notably in the yaw angle. The names of the model are shown on the left, and the three Euler angles are noted on the right three columns. LeNet is shown to be the worst out of the four, due to its old age. The absence of a ReLU function can be detrimental, as Model1 is worse than AlexNet in all Euler angles. ResNet-50 with only a singular loss function is firmly inferior to multiple loss functions, with the difference being around 2 degrees with ResNet-50 with multiple loss functions.

Conclusion

Head pose estimation is an important tool in the modern world, as a large effort has been made to implement this in the automotive industry, due to the alarming number of fatalities and injuries in crashes and incidents regarding distracted driving. Some effort has been put into this, but most have not yet been available for mass use, as they have been plagued with issues, such as allowing autopilot to continue to be used despite the camera being obscured from vision input.

This project focuses on the improvement of face pose estimation, using a estimation method which does not include landmark estimation, contrary to the normal landmark-to-pose estimation methods done by previous works. The network chosen for this project is ResNet-50, a network which mainly focuses on the elimination of the degradation problem in deeper neural networks. It is shown to be noticeably better against AlexNet, LeNet, and Model1, a copy of AlexNet without the ReLU function. By adding a multiple loss function, this network can reliably predict head pose. Low resolution images can be more easily predicted, compared with the earlier method of landmark-to-pose estimation.

References

[1] Palmero, C. , et al. "Recurrent CNN for 3D Gaze Estimation using Appearance and Shape Cues." (2018).

[2] Mansourya, M. , et al. "3D gaze estimation from 2D pupil positions on monocular head-mounted eye trackers." ACM (2016).

[3] N. Ruiz, E. Chong and J. M. Rehg, "Fine-Grained Head Pose Estimation Without Keypoints," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018, pp. 2155-215509, doi: 10.1109/CVPRW.2018.00281.

[4] Distracted driving. *Centers for Disease Control and Prevention*. Available at: https://www.cdc.gov/transportationsafety/Distracted_Driving/index.html [Accessed September 29, 2022].

[5] Klauer, C. et al., 2015. The impact of driver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data. *VTechWorks Home*. Available at: https://vtechworks.lib.vt.edu/handle/10919/55090 [Accessed September 29, 2022].

[6] Barry, K. & Bartlett, with J.S., Tesla driver monitoring fails to keep driver attention on road. *Consumer Reports*. Available at: https://www.consumerreports.org/car-safety/tesla-driver-monitoring-fails-to-keep-driver-focus-on-road-a3964813328/ [Accessed September 29, 2022].

[7] Verma, A., 2020. Alexnet: An Explanation of Paper with Code. *Medium*. Available at: https://towardsdatascience.com/alexnet-8b05c5eb88d4 [Accessed September 30, 2022].

[8] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[9] Zhang, X. *et al.* (2017) “It’s written all over your face: Full-face appearance-based gaze estimation,” *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* [Preprint]. Available at: https://doi.org/10.1109/cvprw.2017.284.

【評語】190030

This project focuses on the face pose estimation in images. The project applies the CNN technology. The report has good quality. The idea is described clearly. Some comments are given below.

The topic is not a new one. More literature survey is suggested to compare the proposed algorithm and the one in the previous works.

More experiments are suggested for the performance evaluation as well as the corresponding discussion on the performance evaluation.