**CSE47201 Computer Vision Programming Assignment 2 [100 points]**

**Deadline : until ~~Nov. 22, 23:59~~ Nov. 26 23:59**

* Download the training/testing hand pose estimation database using the link below, respectively:

[Link1]<https://unistackr0-my.sharepoint.com/:f:/g/personal/muhammad_unist_ac_kr/Emn0_ClYyK5MomtME0U6nSYBGbH_kWNouoBBx1tuMVadGQ?e=biVcE5>

[Link2]<https://drive.google.com/file/d/1QhoWMa1rRWem0ceTOeOnSkD_K5l5BL9F/view?usp=sharing>

* There are 2342 training images (\*.jpg) and their annotations (\*.pkl) in the train folder; while there are 500 testing images (\*.jpg) and their annotations (\*.pkl) in the test folder.
* Use the training database when fine-tuning your model (in Problem 2) while using the testing database when evaluating your model (in Problem 1 and 3).
* Download the network architecture (model.py), main code (main.py) and weight file(\*.pth) contained in the link below:

<https://drive.google.com/drive/folders/1xhXO4uIWZ2yN3JWfqr132dTbc9xkdJqj?usp=sharing>

* If you have any questions, please post it on the blackboard/discussion/‘assignment 2’ board.
* You can write reports either in Korean or English.
* Fonts in pink are recently added to prevent confusions in problems (Last update: 2022.11.02. 14:15).

1. [30 points] We provided the `Tester’ class that initializes the hand pose estimation network with the weights `pretrained\_weight.pth’ provided.

1) [5 points] Run the main.py using the colab code below and visualize outputted 21 32x32 heatmap images in your report for the `106.jpg’ in the test folder. Each 32x32 heatmap is encoding spatial locations of your hand joints in 256x256 images (It is 8 times reducing the resolution). Please note that the index of the 21 heatmaps are as in Figure 1(right).

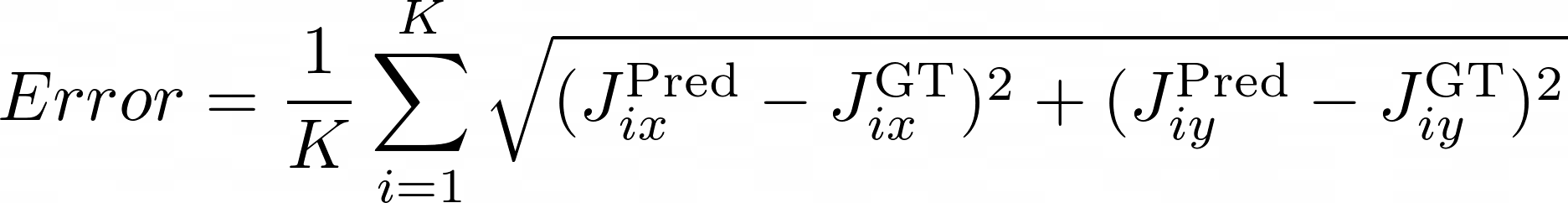
| from google.colab import drive  drive.mount('/content/drive')  !python main.py |
| --- |

2) [5 points] Run the main.py again to visualize skeleton estimation results overlaid with RGB images as in Figure 1(Left) for `106.jpg’. Please confirm that the **`heatmap2skeleton()’** method defined in the `Tester’ class is used to obtain the (batch\_size)x21x2 skeletons from the estimated heatmaps. Please also confirm that `plot\_hand()’ method defined in the `Tester’ class is essential for the visualization (Please confirm that the color of each finger is: Thumb (blue), Index finger (pink), Middle finger (red), Ring finger (green), Pinky (Orange)).

<RGB> Blue: (0,0,255), Pink: (255,0,255), Red: (255,0,0), Green: (0,255,0), Orange: (255,122,0)

|  |
| --- |
| Figure 1. (Left) Example result for hand pose estimation, (Right) Skeleton order. |

3) [15 points] Testing error for each image is measured as follows:



where K denotes the number of joints (=21). and denote the i-th joint’s xy-coordinate values for the prediction and ground-truth skeletons, respectively.

Please implement the ‘calc\_error()’ method that calculates the error for 1 image in

the `Tester’ class. It measures the error in the pixel unit. (You can freely modify(add or

remove) input arguments for the calc\_error() function.)

def calc\_error(self):

err = 0

...

return err

4) [5 points] Please implement the code for calculating the average error for the

whole 500 testing images provided and printing it in the `test()’ method of `Tester’

class. (Apply calc\_error() for each image and print the averaged error for 500 images.)

1. [40 points] Implement the `train’ method of the `Trainer’ class for fine-tuning your hand pose estimation network on the training data provided, when called from the main() function as follows:

trainer = Trainer(epochs, batchSize, learningRate)

trainer.train()

* 1. Perform the fine-tuning after initializing your network with a weight provided. (Ground-truths for xy coordinate values of 21 skeleton joints are provided in the .pkl file with a variable name ‘coord2d’. `coord2d’ variable is an array whose size is 21x2. Please refer to the order of 21 joint indexes in Figure 1(Right).)
  2. Properly tune your learning rate, batchSize and # of epochs etc. to properly train your network. Your network should perform better on the testing images than the initial model to get the full score for this problem.(There would be the score deduction if the result become strange. You need to properly train the network.)
  3. You may need to use skeleton2heatmap() function to output 21 32x32 heatmaps from the 21 skeleton joints when writing your training code.
  4. You may need to use the MSE loss (ie. nn.MSELoss()) between predicted results and ground-truth heatmaps.
  5. Please do not change the structure of the hand pose estimation network (ie. model.py).

1. [20 points] Do the following with your fine-tuned network obtained from Problem 2:

1) [5 points] Run your fine-tuned network obtained from Problem 2 on the 500 testing images again and visualize the obtained skeletons with an input RGB image for the `106.jpg’ as in Figure 1(Left).

2) [5 points] Also measure and report the error on 500 testing images again with your fine-tuned network.

3) [10 points] After that, discuss how your results have been changed before and after the fine-tuning.

1. [10 points] Note and apply the below details.
   1. Send all files(.zip) via **Blackboard**.
   2. In **2022xxxx\_seungryulbaek\_cv\_ass2.zip** file, you need to submit each problem’s .ipynb(you should show your result). If you want to submit only \*.py, please accompany your result in a report (Unless there will be a penalty). Independently, you need to submit a separate ‘report.pdf’ or ‘report.docx’ file as your report.
   3. Also, you need to attach your fine-tuned weights obtained from Problem 2 in the name with `finetunedweight.pth’, to get the full score for it.  
       <Example>

2022xxxx\_seungryulbaek\_cv\_ass2.zip

|

--- finetunedweight.pth

|

--- main.py and 1.ipynb, 1.pdf, 2.ipynb, 2.pdf, 3.ipynb, 3.pdf (This is for

colab users, for non-colab users who can make .ipynb and .pdf)

|

--- main.py and executed screen captured. (This is for non-colab

users)

|

--- report.pdf or .docx (Not mandatory)

|

|--- etc.

\* IPYNB->PDF Tool: <https://htmtopdf.herokuapp.com/ipynbviewer/>

* 1. Code should have some **comment** that increase readability(it is also grading points)

| **<Tip>**  Maximum keeping time for one session in the CoLab is 12 hours when you use GPU mode, but if you remain in your CoLab session without typing anything over 90 mins, the session would be automatically shut down. So, please refer below links to prevent it.  <https://naenjun.tistory.com/18>  <https://stackoverflow.com/questions/57113226/how-to-prevent-google-colab-from-disconnecting> |
| --- |