VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE

MACHINE LEARNING
BAD POSTURE DETECTOR

Instructor

Ph.D. An Truong Pham Nguyen

Students

Thuan The Cao - 20520793
Thai Quoc Nguyen - 20520304
Nhan Trung Nguyen - 20520670

Class

CS117.M21.KHCL

Ho Chi Minh City - 2022

Contents

1 Introduction

1.1 Problem context

One day when I was talking to my friends, I noticed that he often bit his nail, a bad habit that greatly affect hygiene, when there are something bothers him. I think to myself about how I could help my dear friend curl his misbehavior. Suddenly, an idea about an wonderful AI model that can detect such kind of behavior and notify its user popped up in my head. Luckily, There is also an machine learning project are coming up this semester. To make this model even more awesome, My team expand this detector to solve other kind of bad behaviors such as hunched back, face scratching, acne popping... I think this project would be perfect for improving our grades and my friend hygiene.

1.2 Inputs and Outputs

Input:

- A image of a person taken with a laptop webcam
- Minimum image resolution at 480p
- The light enough to clearly see the person in the image (ISO camera $>$ 100)
- The distance from the location of the person to the web cam is 45-55cm and can clearly see at least from chest to face

Output: An input image with the following information: the label of this image belongs to 1 of 12 classes and bounding box with confidence threshold with model Yolov4 or Yolov5 and without with model VGG16.

1.3 Labeling constraints

To ensure that every members of our team agree on how the labeling work is done and our dataset's ground truths are of high quality, we have create labeling constraints. Those constraints are very straight forward that it can be express in one sentence: Each picture will have only one bounder box, only big enough to contain the whole body of the laptop user.

Figure 1: Correctly labeled



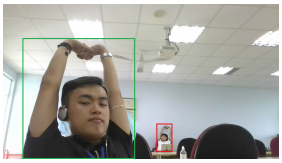


Figure 2: Incorrectly labeled

1.4 Dataset analysis

Our dataset includes 12 classes, there are: sitting straight, sitting with hunched back, sleeping with face on the table, holding a phone, sleeping and leaning back against a chair, nail biting, head scratching, face scratching, chin propping with one hand, chin propping with two hands, stretching and siting with one leg on the chair.

- Total images: 9077 for 12 classes, each class contains around 750 images, classes almost have the same number of images.
- We use 80% for training section and 20% for validation section. Train and Val data were automatically random shuffed and divided by Roboflow.
- Test set that the model has never been exposed to: 1222 for VGG16 and 1185 images for Yolov4 and Yolov5 (because when we used Roboflow to label, somehow some of our image were rejected) and around 93-103 images for each class.
- All data used for the subject is collected by ourselves by taking pictures with computer webcams in many places.

Comment: The data is completely created by the group for individual purposes, so it ensures consistency, uniformity and no data pre-processing step.

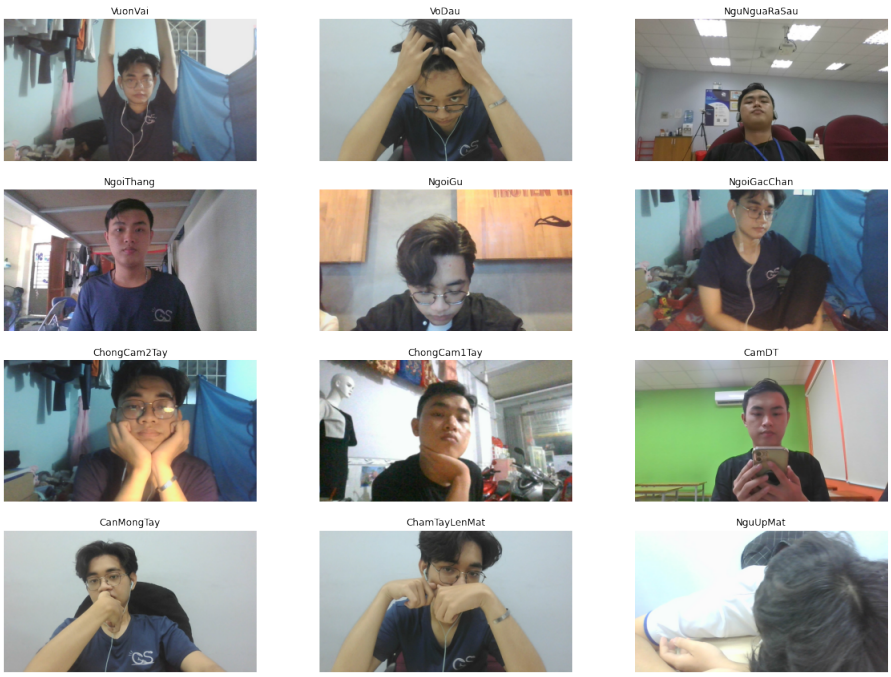


Figure 3: First image in the dataset of each class

1.5 Detailed description of the dataset

Review for dataset:

- After detailed analysis of the dataset, we found that although we had a relatively uniform amount of images for each class, but in each class, there contains small group of data (which can be considered as sub-classes) are not consistent and balanced $->$ unbalanced data.
- Because these selected models work by extracting the image feature of each class, here the image of each class is influenced by specific colors: clothes color, background color,...

  Conclusion: these reasons can cause overfitting for our model.

Dataset for each model:

Model	Yolov4	Yolov5	VGG16
Data	corresponding to an image file, there will be an equivalent .txt label file with the same name	corresponding to an image file, there will be an equivalent .txt label file with the same name	images are separated into each folder belongs to it's class

Figure 4: Data for each class

Some difficult cases to handle:

- Between classes such as: sitting straight and sitting with hunched back; chin propping with one hand, chin propping with two hands, nail biting and stretching doesn't make too much of a difference.

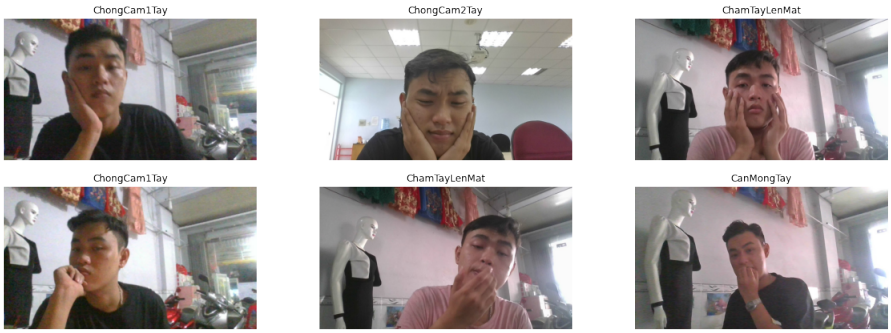
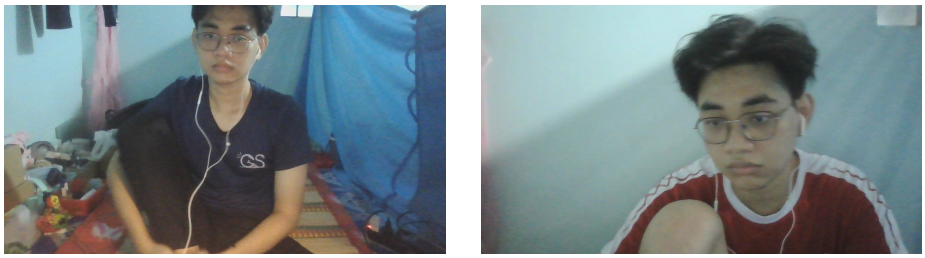
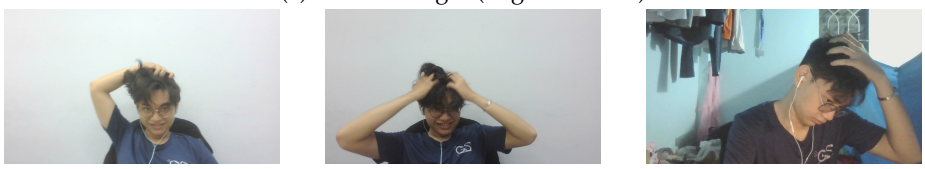


Figure 5: Similarities between classes

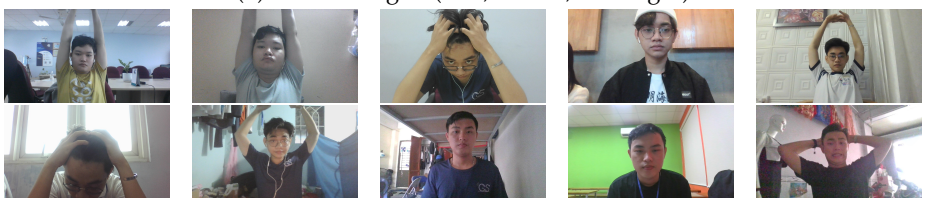
- Because our postures are extremely diverse and even though we use the same name, between different moments are not exactly the same. This causes great difficulty in identification and classification. $->$ Confusion.
- Within our classes, there in lies some sub-classes that have cause our model tons of trouble, bias towards toward some classes in our predefined data and even bias within the sub-classes of one class. Those sub-classes include:
  - Camera angle: Left, Centre, Right, ... For instance, if the posture involves the use of hands like "Nail biting", "Face scratching", or "Chin propping" then the image will be classified based on the side of the hands (e.g: Left for left hand, right for right hand, center for both hands). Otherwise, we will classify the picture base on the direction that the body is facing.
  - Camera height: High and Low. The picture camera angle is high when we could see below the chest even if it is obstructed by objects.
  - Location: AI Club, Thuan's home, Nhan's home, Thai's home, cafe,...



(a) Camera height (High and Low)



(b) Camera angles (Left, Centre, and Right)



(c) Location

Figure 6: Sub-classes

- To make the situation even worse, the amount of samples in those three sub-classes are not balance, so there will be bias in our model toward a specific side of our body if we don't balance out those sub-classes.

Class	Member	Total	Position		
			Left/Left hand	Centre/2 hands	Right/Right hand
VoDau	Thuận	257	50(19.5%)	151(58.8%)	56(21.8%)
	Nhân	252	97(38.5%)	60(23.8%)	95(37.7%)
	Thái	258	66(25.6%)	146(56.6%)	45(17.4%)
ChamTayLenMat	Thuận	262	117(44.66%)	61(23.28%)	84(32.06%)
	Nhân	257	136(55.06%)	9(3.64%)	102(41.30%)
	Thái	260	102(39.23%)	71(27.31%)	87(33.46%)

Figure 7: Sub-classes' statistics

- Here, we can see that in "Head scratching" (Vo Dau), the distributions of each side of the body are not balanced. "Centre/2 hands" is more common, contributing about 46% of the "Head scratching"'s data, compare to "left/left hand" (28%) and "right/right hand" (26%). And in "Face scratching" (ChamTayLenMat), "Left/Left hand" is the most prominent sub-class compare to "Right/Right hand" and "Centre/2 hands".

2 Training - Evaluation

2.1 Explanation of model selection

After learning about the dataset we use, we decided to choose 3 models: Yolov4, Yolov5 and VGG16. Because these are CNN (convolutional neutral network) models, it is possible to extract features through layers. Reasons for our choice:

1. These model are well-known and used by a lot of people for the task detection and classification.
2. Because of stratified architecture, learning features from different levels, the level of detail of the features of the object to be detected in the image is also increased through Feature Maps.
3. These are pretrained models, so it has learned to self-adjust the features it needs to extract to match the corresponding tasks.

2.2 VGG16

Through a competition on Kaggle, we accidentally got acquainted with the use of model VGG16 in the problem of distinguishing cats and dogs. Realizing that our problems also have similar characteristics - they are all classification problems, we tried to apply the model VGG16 to this problem.

2.2.1 Training

First of all we upload all our dataset contain 12 folders - 12 classes into Google Drive.

Train model with colab:

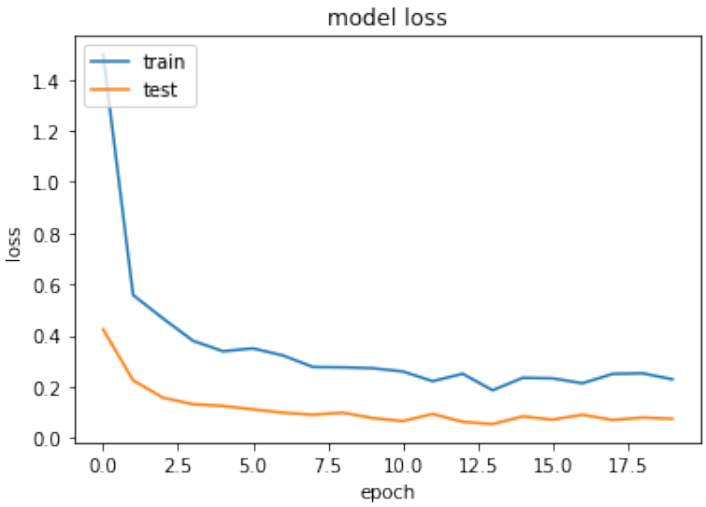
1. We use utils.image_dataset_from_directory from tensorflow.keras to automatically random divide data with ratio 80, 20 for train and val.
2. Load model VGG16 from tensorflow.keras.applications without the last 3 layers of full connected. Then add input_layer, rescaling layer, model VGG16 and 3 full_connected layers to complete the model.
3. Compile the model with optimizer Adam (Adam optimizer is an algorithm that combines the techniques of RMS prop and momentum), loss = SparseCategoricalCrossentropy and metrics is accuracy.
4. Then we also define model_earlyStopping (in 6 iters, if val_loss has no difference more than 1e-4 then stop training) and model_checkpoint (to save model for continue training or use to predict in future)

```python
train_ds = utils.image_dataset_from_directory(
    Train_dir,
    validation_split=0.2,
    subset="training",
    seed=47,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=batch_size
)

val_ds = utils.image_dataset_from_directory(
    Train_dir,
    validation_split=0.2,
    subset="validation",
    seed=47,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=batch_size
)
```

Figure 8: Data preparation

We defined 50 epochs for our model but it automatically stopped in epoch 20 due to model_earlyStopping. Then we plot val_loss of train and val of our model




```
Epoch 1/50
230/230 [==============================] - ETA: 0s - loss: 1.4981 - accuracy: 0.5830
Epoch 1: saving model to /content/drive/MyDrive/Máy_học_CS114/[Final]/[Model_checkpoint_VGG16]/cp.ckpt
230/230 [==============================] - 2818s 12s/step - loss: 1.4981 - accuracy: 0.5830 - val_loss: 0.4238 - val_accuracy: 0.8671
Epoch 2/50
230/230 [==============================] - ETA: 0s - loss: 0.5580 - accuracy: 0.8116
Epoch 2: saving model to /content/drive/MyDrive/Máy_học_CS114/[Final]/[Model_checkpoint_VGG16]/cp.ckpt
230/230 [==============================] - 51s 218ms/step - loss: 0.5580 - accuracy: 0.8116 - val_loss: 0.2249 - val_accuracy: 0.9363
Epoch 3/50
230/230 [==============================] - ETA: 0s - loss: 0.4665 - accuracy: 0.8369
Epoch 3: saving model to /content/drive/MyDrive/Máy_học_CS114/[Final]/[Model_checkpoint_VGG16]/cp.ckpt
230/230 [==============================] - 52s 220ms/step - loss: 0.4665 - accuracy: 0.8369 - val_loss: 0.1569 - val_accuracy: 0.9455
```

Figure 9: Loss and Val_loss change through 20 epochs. Loss reduced in 2 first epochs.

We can see loss reduced very fast from epoch 1 to epoch 2 ($1.4981 - > 0.5580) - >$ this model can be overfitting. Despite the large number of our photos, they are spread over 12 classes. In addition, there is a lack of diversity and uniformity in each class.

Difficulty: Although we used model_checkpoint, also tried to save model and save weights, but somehow the above methods cannot reload the model when we need to use it again, so every time we need to test, we have to train from the beginning then waste too much time.

2.2.2 Evaluation

With this model, the metric we use to evaluate is Accuracy. This is the way to calculates how often predictions equal labels. It is quite different from the next two model in following sections, because Yolov4 and Yolov5 are detection and classification model but VGG16 only the classification model.

This metric creates two local variables, total and count that are used to compute the frequency with which y_pred matches y_true. This frequency is ultimately returned as binary accuracy: an idempotent operation that simply divides total by count.

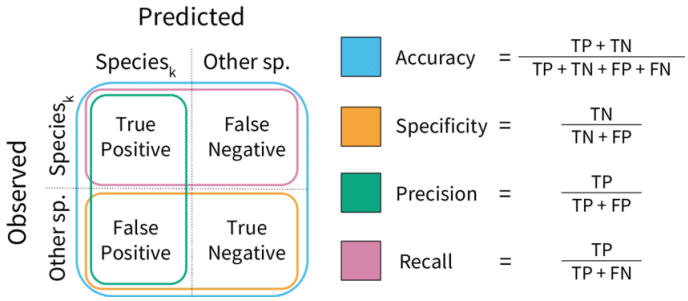


Figure 10: Illustration for metric "Accuracy"

Source:`https://www.researchgate.net/figure/Model-performance-metrics-Visual-representation-of-the-classification-model-metrics_fig1_328148379`

2.2.3 Evaluation results based on test data

Testing process:

Although model VGG16 uses the evaluation metric Accuracy, this metric is used to evaluate the val set during training but could not be used to test datatest $- >$ we conducted manual testing on each image and use keras.preprocessing, pandas, csv, glob2, numpy, os libraries to support testing and save results.

At first, we save the path of each image and it's label into a csv file names

testLabel.csv then we wrote down the predictions (also contains the path of each image and it's predicted label) into predictLabel.csv and then compared them together.

Testing result:

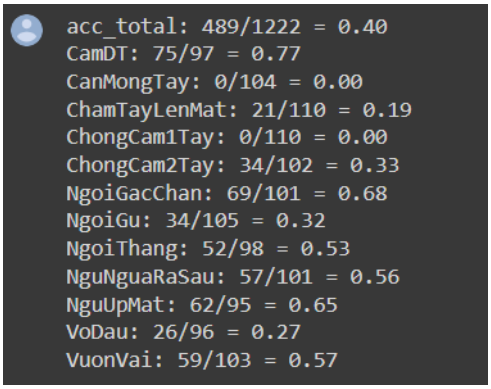


Figure 11: The result of test dataset when use only VGG16

Analyze the results:

The model has extremely bad prediction results with the datatest set, although the loss and accuracy on the train and val sets (train and val were split from the same dataset with ratio 80:20) during training is very good. $->$ Model is extremely overfitting

The most true predicted classes: holding a phone, stretching and sitting with one leg on the chair, sleeping with face on the table while the rest of the classes performed very badly and were barely usable.

2.2.4 Improvement:

After the Q&A and the teacher's advice, we adjusted the **datatrain** for VGG16, we cropped 9075 images with the size of the bounding box in the datatrain for yolov4 and yolov5 $->$ new train dataset for VGG16. Then when we perform the test, we use the model yolov5 that has been trained for the detection part. We cropped the detection part and used the model VGG16 for classification.

Train with the new dataset and re-test:

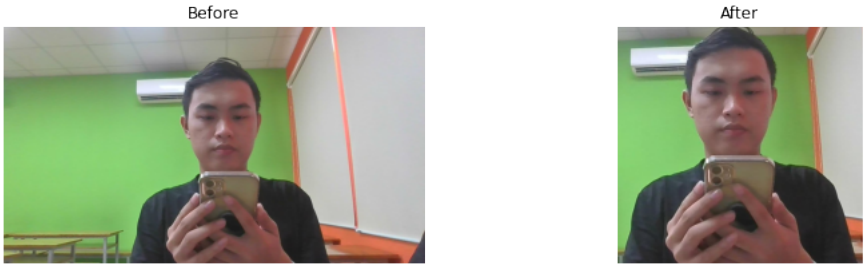


Figure 12: Datatrain after cropped

PREDICT

ACTUAL

Confusion_matrix using Logistic Regression is :

```
[[51  8  2  9  5  1  1  0  3  2  1  9]
 [ 9  8  8 23  7 10  0  1 15  1  7 14]
 [ 3  4 24 15 10  7  0  0 23  5  7  6]
 [ 9  6 19 32 11  2  3  1  0  7  1 13]
 [17  7 12 10 29  3  9  1  0  3  1 10]
 [ 3  1  0  0  0 88  0  0  9  0  0  0]
 [15  0  6  7  2  0  7  1  1 26  0 30]
 [ 2 16  1 17  0 10  0 41 11  0  0  0]
 [ 0 15  7  0  0  2  0  2 57  0  2  4]
 [ 1  0  0  0  0  0  0  0  0 76  1  0]
 [ 6  1  3  0  0  0  0  0  0 22  2 40 21]
 [ 3  0  1  0  0  0  0  0 22  2  1 74]]
```

Classification_report using Logistic Regression is :

	precision	recall	f1-score	support
0	0.43	0.55	0.48	92
1	0.12	0.08	0.09	103
2	0.29	0.23	0.26	104
3	0.28	0.31	0.29	104
4	0.45	0.28	0.35	102
5	0.72	0.87	0.79	101
6	0.35	0.07	0.12	95
7	0.87	0.42	0.57	98
8	0.35	0.64	0.45	89
9	0.61	0.97	0.75	78
10	0.66	0.42	0.51	95
11	0.41	0.72	0.52	103
accuracy			0.45	1164
macro avg	0.46	0.46	0.43	1164
weighted avg	0.46	0.45	0.43	1164

Figure 13: The result of test dataset when use yolov5 for detection to crop the image then use VGG16 for classification

		PRED											
		CamDT	CanMongTay	ChamTayLenMat	ChongCam1Tay	ChongCam2Tay	NgoiGacChan	NgoiGu	NgoiThang	NguNguaRaSau	NguUpMat	VoDau	VuonVai
TRUE	CamDT	55%	9%	2%	10%	5%	1%	1%	0%	3%	2%	1%	10%
	CanMongTay	9%	8%	8%	22%	7%	10%	0%	1%	15%	1%	7%	14%
	ChamTayLenMat	3%	4%	23%	14%	10%	7%	0%	0%	22%	5%	7%	6%
	ChongCam1Tay	9%	6%	18%	31%	11%	2%	3%	1%	0%	7%	1%	13%
	ChongCam2Tay	17%	7%	12%	10%	28%	3%	9%	1%	0%	3%	1%	10%
	NgoiGacChan	3%	1%	0%	0%	0%	87%	0%	0%	9%	0%	0%	0%
	NgoiGu	16%	0%	6%	7%	2%	0%	7%	1%	1%	27%	0%	32%
	NgoiThang	2%	16%	1%	17%	0%	10%	0%	42%	11%	0%	0%	0%
	NguNguaRaSau	0%	17%	8%	0%	0%	2%	0%	2%	64%	0%	2%	4%
	NguUpMat	1%	0%	0%	0%	0%	0%	0%	0%	0%	97%	1%	0%
	VoDau	6%	1%	3%	0%	0%	0%	0%	0%	23%	2%	42%	22%
	VuonVai	3%	0%	1%	0%	0%	0%	0%	0%	21%	2%	1%	72%

Figure 14: VGG16's confusion matrix

Note: 1: holding a phone; 2: nail biting; 3: face scratching; 4: chin probbing with one hand; 5: chin probbing with two hands; 6: siting with one leg on the chair; 7: sitting with hunched back; 8: sitting straight; 9: sleeping and leaning back against a chair; 10: sleeping with face on the table; 11: head scratching; 12: stretching; .

Evaluation: With cropping the entire image of the train dataset and using an additional model dectection to support the classification, the accuracy only increases by 5%.

Analyze the false prediction pictures: Because this is the classification model then we input an image, it will give us back a result that this image belongs to 1 of 12 classes $-> $ Only have false prediction

- The confusion between classes has the same posture or the same background because the model has no detection part.

  EX: With **nail biting** class up to 23 photos were mistaken for chin probbing with one hand class. And with the **face scratching** class, there are also 23 images mistakenly transferred to the stretching class because in the training set of the stretching class, there are images that do not raise the hand completely, but only bring it across the face, and also mistakenly switch to the chin probbing with one hand class (15) and nine probbing with two hands (10), the same goes for classes **chin probbing with one hand and chin probbing with two hands**, the model has difficulty classifying these 3 classes (there is confusion between these 3 classes). Class **stretching and head scratching** both have 22 images that are mistaken for sleeping and leaning back against a chair because they both lean back.

- Almost images we use for test have the background, the color like: clothes,... different from which we used to train.

  EX: In the **biting nail** class, although the AP ratio is high, the amount of TP is only about half. The remaining photos cannot be labeled, in which 48 photos of Nhan were taken at a cafe and 5 photos of Thuan were taken at the AI club room, these are the shooting locations that are not included in the training set of the nail biting class. In the **face scratching** class, there are 26 photos that can't be classified because they were taken at different locations in the training session with Nhan in the cafe (15) and Thuan in the AI club room (11). In class **chin probbing** with one hand,

there are 26 photos that can't be classified due to different locations in the training set with Nhan at the cafe (25) and Thuan at the cafe (3), and 30 pictures in **sitting with hunched back** class were taken at the cafe by Thuan $->$ has confusion to most classes.

- The distance from the person to the webcam is closer than the binding regulations.

  EX: In class **chin probbing with one hand** there are 25 photos, **nail biting** has 35 photos which are photos taken relatively close to the camera than conventional.

- Due to user perform many postures at the same time.

  EX: As in class sitting straight due to some pictures besides bad posture there is also sitting straight back $->$ there is wide confusion on the other classes, the same with sitting with hunched back.

Analyze the true prediction pictures:

- Compared with other classes, siting with one leg on the chair had very different characteristics.
- Many similar classes like stretching classes, sleeping with face on the table most of the images are predicted into these classes so with correct labels for high TP.
- Since this is only a classification model, if the test images are taken at the same location and have the same details as the train image, the results will be better.

2.3 Yolov4

Currently, yolov4 is still rated as one of the best state-of-the-art object detector models. This model also uses many datasets to train before, such as MS COCO and ImageNet.

2.3.1 Training

With the topic we have chosen, in the model training step we will use the Pre-trained Weights file yolov4.conv.137 because this is a Pretrained Weights file that saves time retraining the entire model from scratch.

About data for training: Yolov4 requires train and val data will include the image and attached .txt file, which contains information about the label, coordinates (x,y) of the bounding-box's center point, height and the width of the bounding box. As mentioned above, we use Roboflow for labeling and creating .txt files. Basically the data consists of 9077 images and 9077 .txt files, then Roboflow will divide the dataset by the ratio of 80% train and 20% val.

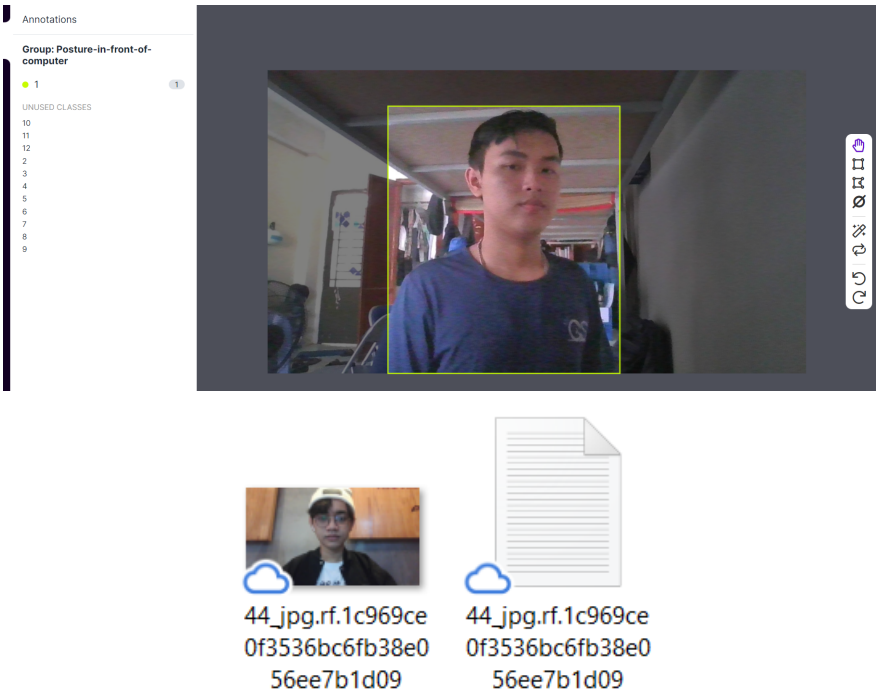


Figure 15: Labeling process using Roboflow and Image with it's .txt file

Train model with colab:

1. Clone the source code needed to train the model from `https://github.com/AlexeyAB/darknet`
2. Clone file yolov4.conv.137
3. Upload dataset folder into darknet folder on Google Drive
4. Create train and val .txt files contain path to each image in train and val folder
5. Write down 12 classes into obj.names and create file obj.data contain number of class, path to train.txt, val.txt and obj.data file, path to backup folder
6. Edit the file yolov4-custom.cfg, edited parameters to fit the dataset:
   - width = 416, height = 416
   - classes = 12
   - filters = 51
   - max_batches = 24000
   - steps=19200,21600
7. Training first time and continue training with yolov4-custom_last.weights in backup folder

Theoretically the model would have to do 24000 iterations, however it reach around 5878 iters we observed that the mAP got relatively high (99.73%) and the avg loss almost didn't decrease anymore so we stopped working. train the model again.

$-$ > Explain: our dataset contains 12 classes and the number of image used for training in each class around 600s, so with the powerful model, it was soonly reach the high mAP.

2.3.2 Evaluation

Model Yolov4 will be evaluated based on two parameters: avg_loss and mAP, but here it is, we want to compare with the Yolov5 model in the following section, the team will only focus on the mAP index as well as AP50, AP75.

To use mAP we need to pay attention to the following things:

IoU: the degree of overlap between bboxes, specifically between the grounth truth bounding box - which exactly encloses the object's bbox - is the bbox that we have label with the bounding box that the model predicts. IoU thresh is an important indicator, assuming IoU thresh = 0.5, after calculating bbox, if bbox is above 0.5, it will be counted as true otherwise false.

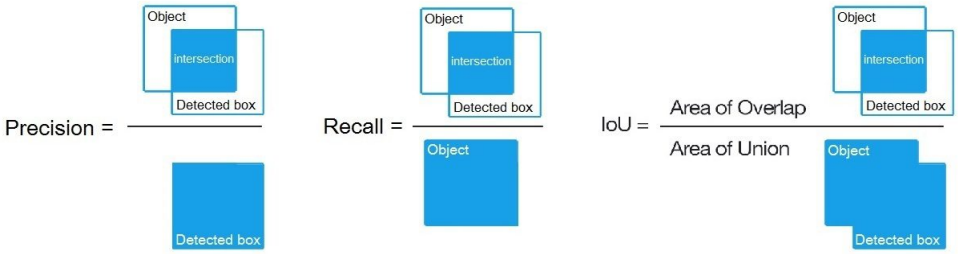


Figure 16: Relationship between Precision, Recall and IoU.
Source:`https://www.kdnuggets.com/2020/08/metrics-evaluate-deep-learning-object-detectors.html`

- IoU (Intersection over Union): Area of intersection of bounding boxes divided by Area of union of the predicted bounding box
- Precision: The number of True Positive(TP) divided by the sum of True Positive(TP) False Positive(FP)
- Recall: The number of True Positive divided by the sum of True Positive(TP) False Negative (FP)

AP: is an index showing the relationship between Precision and Recall (Average Precision)

AP50: is the accuracy with IoU = 0.5 for a class

AP75: is the accuracy with IoU = 0.75 for a class

− >mAP (means Average Precision) for the whole model is calculated by combining AP of each class then devide by 12

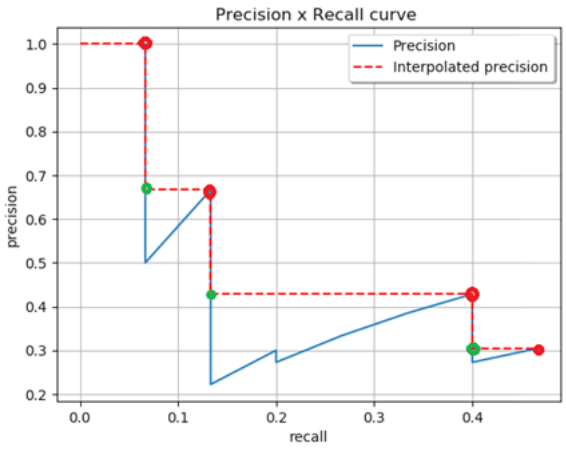


Figure 17: Illustrating the calculation of AP according to the AUC method Source:`https://blogcuabuicaodoanh.wordpress.com/2020/02/22/mean-average-precision-map-trong-bai-toan-object-detection/`

Formula: $AP = \sum_1^n (R_n - R_{n-1}) P_n$. After we get all AP for each class, we can calculate mAP for the whole model.

2.3.3 Evaluation results based on test data

Testing process:

Output request: The detected image must have the following information: predicted class name, surrounding bounding box, confidence threshold.

Progress: write the entire path to the test image file in the test.txt file. Then change the path to the val directory in the obj.data file become test.txt. Use the command: **"!./darknet detector map data/obj.data cfg/yolov4-custom.cfg backup/yolov4-custom_last.weights"** to check the model's mAP with IoU = 50.

Testing result:

```
1188
 detections_count = 4262, unique_truth_count = 1185
class_id = 0, name = NgoiThang, ap = 85.63%       (TP = 73, FP = 20)
class_id = 1, name = NgoiGu, ap = 65.87%          (TP = 41, FP = 26)
class_id = 2, name = NguUpMat, ap = 82.55%        (TP = 54, FP = 0)
class_id = 3, name = CamDT, ap = 96.34%           (TP = 73, FP = 2)
class_id = 4, name = NguNguaRaSau, ap = 86.41%            (TP = 69, FP = 23)
class_id = 5, name = CanMongTay, ap = 91.61%      (TP = 48, FP = 0)
class_id = 6, name = VoDau, ap = 83.44%           (TP = 45, FP = 0)
class_id = 7, name = ChamTayLenMat, ap = 81.97%           (TP = 42, FP = 3)
class_id = 8, name = ChongCam1Tay, ap = 74.46%            (TP = 56, FP = 23)
class_id = 9, name = ChongCam2Tay, ap = 94.39%            (TP = 95, FP = 20)
class_id = 10, name = VuonVai, ap = 87.45%        (TP = 91, FP = 29)
class_id = 11, name = NgoiGacChan, ap = 92.79%            (TP = 81, FP = 8)

 for conf_thresh = 0.50, precision = 0.83, recall = 0.65, F1-score = 0.73
 for conf_thresh = 0.50, TP = 768, FP = 154, FN = 417, average IoU = 68.91 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.852427, or 85.24 %
Total Detection Time: 774 Seconds

Set -points flag:
 `-points 101` for MS COCO
 `-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
 `-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset
```

Figure 18: Result of model Yolov4 with IoU = 50

```
 detections_count = 4262, unique_truth_count = 1185
class_id = 0, name = NgoiThang, ap = 0.00%        (TP = 0, FP = 141)
class_id = 1, name = NgoiGu, ap = 0.07%           (TP = 2, FP = 94)
class_id = 2, name = NguUpMat, ap = 0.18%         (TP = 1, FP = 64)
class_id = 3, name = CamDT, ap = 0.02%            (TP = 1, FP = 84)
class_id = 4, name = NguNguaRaSau, ap = 0.18%             (TP = 3, FP = 108)
class_id = 5, name = CanMongTay, ap = 0.00%       (TP = 0, FP = 63)
class_id = 6, name = VoDau, ap = 0.12%            (TP = 1, FP = 64)
class_id = 7, name = ChamTayLenMat, ap = 0.07%            (TP = 2, FP = 66)
class_id = 8, name = ChongCam1Tay, ap = 0.00%             (TP = 0, FP = 116)
class_id = 9, name = ChongCam2Tay, ap = 0.00%             (TP = 0, FP = 145)
class_id = 10, name = VuonVai, ap = 0.00%         (TP = 0, FP = 149)
class_id = 11, name = NgoiGacChan, ap = 0.03%             (TP = 1, FP = 105)

 for conf_thresh = 0.25, precision = 0.01, recall = 0.01, F1-score = 0.01
 for conf_thresh = 0.25, TP = 11, FP = 1199, FN = 1174, average IoU = 0.87 %

 IoU threshold = 95 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.95) = 0.000553, or 0.06 %
Total Detection Time: 27 Seconds
```

Figure 19: Result of model Yolov4 with IoU = 95

		PRED												
		CamDT	CanMongTay	ChamTayLenMat	ChongCam1Tay	ChongCam2Tay	NgoiGacChan	NgoiGu	NgoiThang	NguNguaRaSau	NguUpMat	VoDau	VuonVai	FN
TRUE	CamDT	80%	0%	0%	1%	0%	2%	0%	0%	0%	0%	0%	0%	16%
	CanMongTay	0%	42%	0%	9%	3%	0%	0%	0%	0%	0%	0%	0%	46%
	ChamTayLenMat	0%	0%	36%	19%	10%	0%	0%	0%	0%	0%	0%	5%	30%
	ChongCam1Tay	0%	0%	0%	58%	11%	0%	0%	1%	0%	0%	0%	0%	30%
	ChongCam2Tay	1%	0%	0%	1%	90%	0%	4%	0%	0%	0%	0%	0%	4%
	NgoiGacChan	2%	0%	0%	0%	0%	79%	1%	0%	0%	0%	0%	0%	18%
	NgoiGu	0%	0%	0%	12%	0%	3%	39%	0%	0%	0%	0%	0%	46%
	NgoiThang	0%	0%	0%	3%	0%	2%	0%	73%	0%	0%	0%	3%	18%
	NguNguaRaSau	0%	0%	0%	0%	0%	0%	0%	7%	67%	0%	1%	0%	25%
	NguUpMat	0%	0%	0%	0%	0%	0%	21%	0%	0%	61%	1%	0%	17%
	VoDau	0%	0%	2%	0%	0%	0%	0%	0%	10%	0%	58%	19%	10%
	VuonVai	0%	0%	0%	0%	0%	0%	0%	0%	10%	0%	0%	89%	1%

Figure 20: Yolov4's confusion matrix

Analyze the results:

The detected image file has a bounding box, we will attach it on github

Based on the collected results, when we set the threshold IoU = 50, the models had relatively satisfactory results. Most classes have relatively high AP ($>$ 80%) except: sitting with hunched back and chin probbing with one hand.

False positives are relatively high because classes have poor prediction results, leading to a relatively high but not high level of recall and precision. High number of False Nagative images (293) these are images that the model cannot detect. However, the mAP is quite good.

Analyze the false prediction and undetected pictures:

- False prediction come from the misunderstanding like we mention before (some classes nearly the same).

  EX: There are 26 photos misclassified in **sleeping with face on the table** including 20 photos from sitting with hunched back class. In the **head scratching** class, there are 96 images, but only 45 images can be classified. The main reason is due to confusion over classes similar to stretching (18) and sleeping and leaning back against a chair (10). In class **face scratching** there are 21 pictures mistakenly transferred to class chin probbing with one hand

- Almost images we use for test have the background, the color like: clothes,... different from which we used to train.

  EX: In the **biting nail** class, although the AP ratio is high, the amount of TP is only about half. The remaining photos cannot be labeled, in which 48 photos of Nhan were taken at a cafe and 5 photos of Thuan were taken at the AI club room, these are the shooting locations that are not included in the training set of the nail biting class. In the **face scratching** class, there are 26 photos that can't be classified because they were taken at different locations in the training session with Nhan in the cafe (15) and Thuan in the AI club room (11). In class **chin propping** with one hand, there are 26 photos that can't be classified due to different locations in the training set with Nhan at the cafe (25) and Thuan at the cafe (3), and 30 pictures in **sitting with hunched back** class were taken at the cafe by Thuan.

- The distance from the person to the webcam is closer than the predefined constraints.

  EX: In class **chin propping with one hand** there are 25 photos, **nail biting** has 35 photos which are photos taken relatively close to the camera than conventional.

- Due to user perform many kinds of postures at the same time.

  EX: 5 images in **chin propping with two hands** had the overlapping bounding box.

Analyze the true prediction pictures:

- These class with AP high have a marked difference from the others, so it is easy for the model to identify. For example, class holding a phone has 73 TP but only 2 FP
- The model will work well with the image that have the same texture like train data such as: background, clothes and posture.

With the IoU = 95, the model hardly works. With the IoU = 50 we see that the average IoU for each image prediction is 60.80%, so when we require the higher IoU this make model can not active $->$ Need to be improved.

Folder contains 1185 predicted images with IoU = 50 we will attach on github.

2.4 Yolov5

After trying out Yolov4 and VGG16 and seeing their incredible performance, Yolov5 will be chosen as our third and final model for solving this problem. Being a part of the Yolo family's models, Yolov5 possess the finest objects detection and recognition ability in today's standard.

2.4.1 Training

Yolov5 have 5 different kinds of model ranging from the smallest but least accurate model, yolov5s, to the largest and most reliable model, yolov5x. Because our application will be apply on laptop, our model must be light and quick in order to cope with the lack of computational power. Yolov5s will be our chosen model for that reason.

The data format of Yolov5 consist of a folder containing only images that we compute on, a folder containing the ground truth .txt files of those images, and finally a .yaml file containing the file path to the image folder, the number of classes that we wish to predict.

The ground truth files consist of the label of their corresponding images, the coordinate of the centre point of their bounding box, and their heights and widths. The order of it is as follows: $<$ *labelname* $><$ *xcenter* $><$ *ycenter* $><$ *width* $><$ *height* $>$.For example: 11 0.5060390625 0.5 0.8193046875000001 1.

In order to make the long classes names shorter, making it easier for us to see the labels of our bounder boxes, we had to made acronyms for all of our classes in Yolov5. All of it are as follows:

1. SS for "Sitting straight"
2. HB for "Hunched back"
3. N for "Napping"

4. HP for ”Holding phone”
5. RC for ”Resting on chair”
6. NB for ”Nail biting”
7. HS for ”Head scratching”
8. FS for ”Face scratching”
9. CP1H for ”Chin propping with 1 hand”
10. CP2H for ”Chin propping with 2 hands”
11. S for ”Stretching”
12. LC for ”Legs on chair”

Hyper-parameters:

- Pixels: This values will determine the size of your input picture. Yolov5 default value is 640.
- Batches: This will determine how many pictures your model will learn in each train iteration. The larger the better. However, your computer may fail to run if the batch size is too high. Yolov5 default value is 16.
- Epochs: This is how many time model will train. It will depend on the size of your dataset. More epochs for big dataset, less epochs for small dataset! Follow this tip to prevent overfitting. Yolov5 default value is 3.

The training process:

1. Connect to yolov5 folder via Google drive
2. Installing yolov5 dependencies, importing libraries and defining helper functions
3. Connect to Weight and Bias account for ease of training process's visualization (optional)
4. Run the train.py file, appending the file path to the data.yaml, the chosen .pt weight file and defining hyper-parameters like the picture resolution, the number of batches, epochs,...

2.4.2 Evaluation results

The Yolov5 model evaluation system use a myriad of metrics for detail and trust worthy results. It includes: IoU, recall, precision, mAP_0.5, mAP_0.5:0.95, cls_loss, box_loss, obj_loss. A detailed look at these metrics:

- box_loss: bounding box regression loss (Mean Squared Error).

- obj_loss (Object loss): the confidence of object presence is the objectness loss (Binary Cross Entropy).
- cls_loss (Classification loss): the classification loss (Cross Entropy).
- mAP@0.5 (Mean Average Precision at 0.5): the averaged AP of all object classes with IoU threshold set to 0.5
- mAP@0.5:0.95 (Mean Average Precision between 0.5 to 0.95): the averaged AP of all object classes and at 10 different IoU thresholds, starting at 0.5 to 0.95 with 0.05 steps.

When we run validation on Yolov5, the result format will be difference than that of Yolov4 and VGG16. Every time we run, the results will be save in a new folder. And in those folder, there will be a myriad of graphs, ground truths and predictions mosaics, and a confusion matrix. Optionally you can also add .txt files containing the validation results of each picture you tested on. However this step is not necessary in this project.

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 38/38 [00:29<00:00, 1.30it/s]
all	1185	939	0.696	0.671	0.698	0.509
SS	1185	87	0.906	0.552	0.739	0.585
HB	1185	61	0.902	0.607	0.784	0.678
N	1185	62	1	0.968	0.984	0.768
HP	1185	71	0.753	0.901	0.878	0.636
RC	1185	83	0.864	0.916	0.843	0.645
NB	1185	62	0	0	0	0
HS	1185	79	0.574	0.342	0.483	0.446
FS	1185	71	1	0.0986	0.549	0.405
CP1H	1185	75	0.584	0.88	0.686	0.474
CP2H	1185	92	0.556	0.924	0.775	0.522
S	1185	98	0.613	0.888	0.767	0.39
LC	1185	98	0.6	0.98	0.887	0.561

Figure 21: Validation results

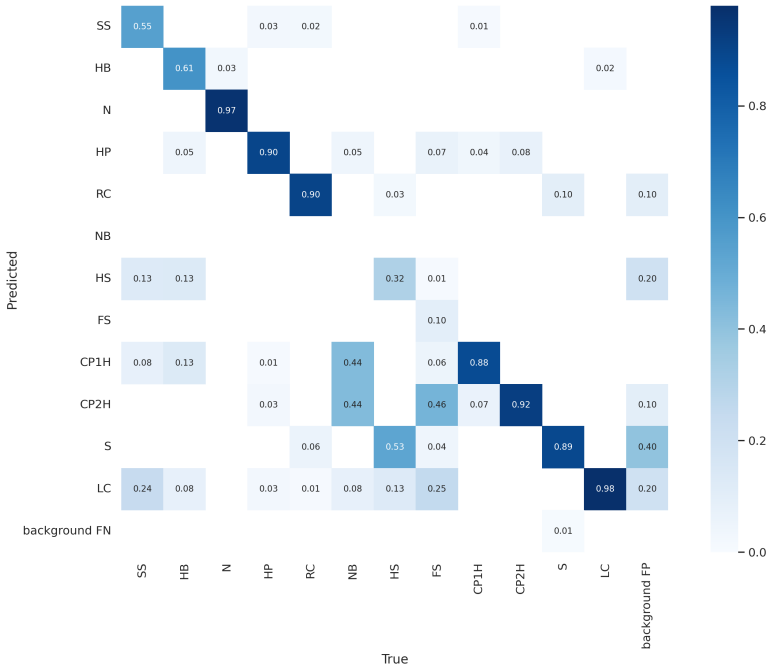
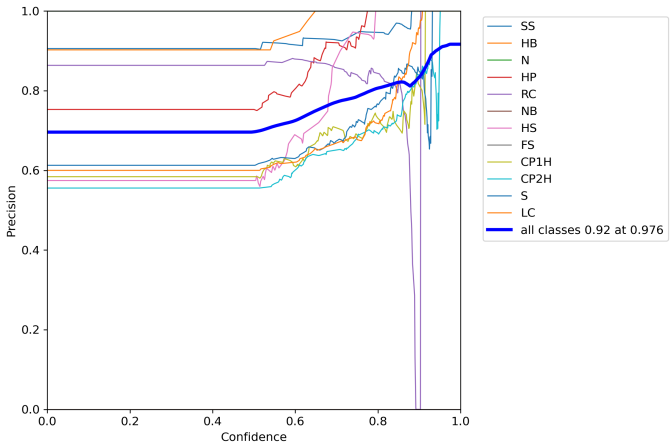
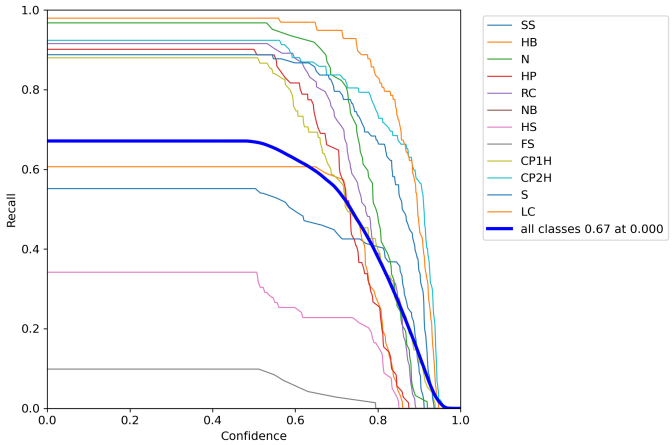


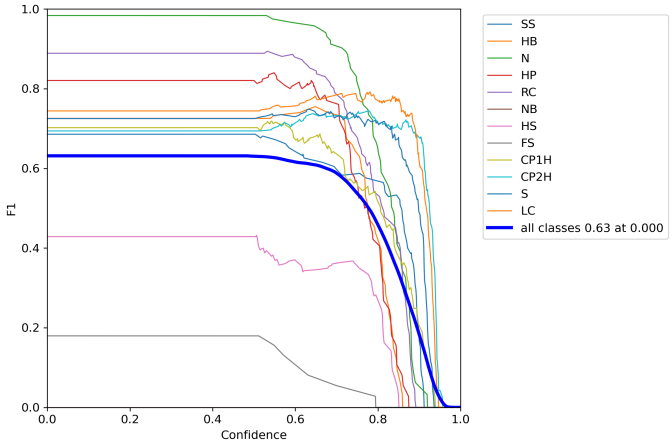
Figure 22: Confusion matrix



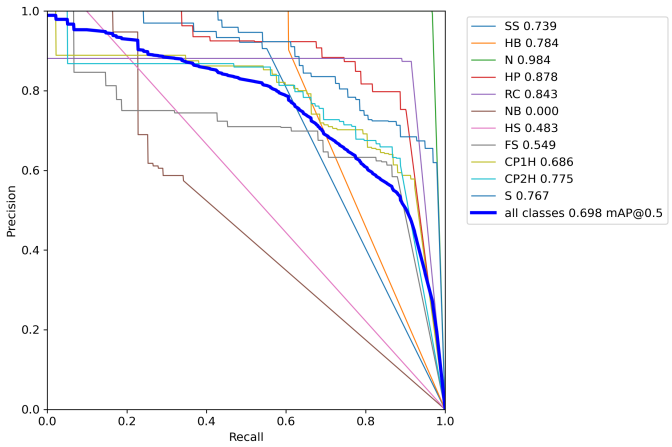
(a) Precision curve



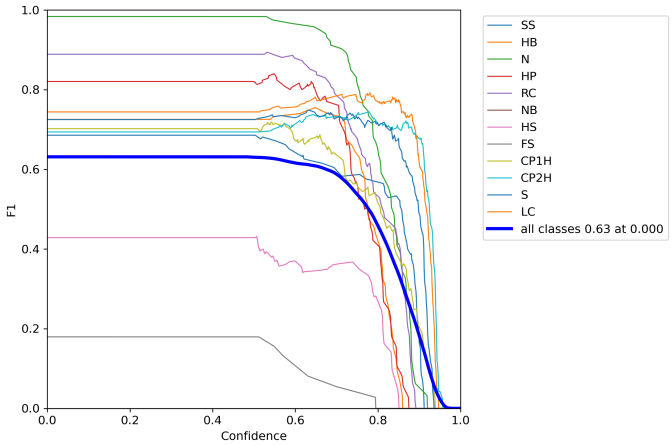
(b) Recall curve



(c) Recall curve



(d) Precision/Recall curve



(e) F1 curve

Figure 23: Yolov5's validation graphs

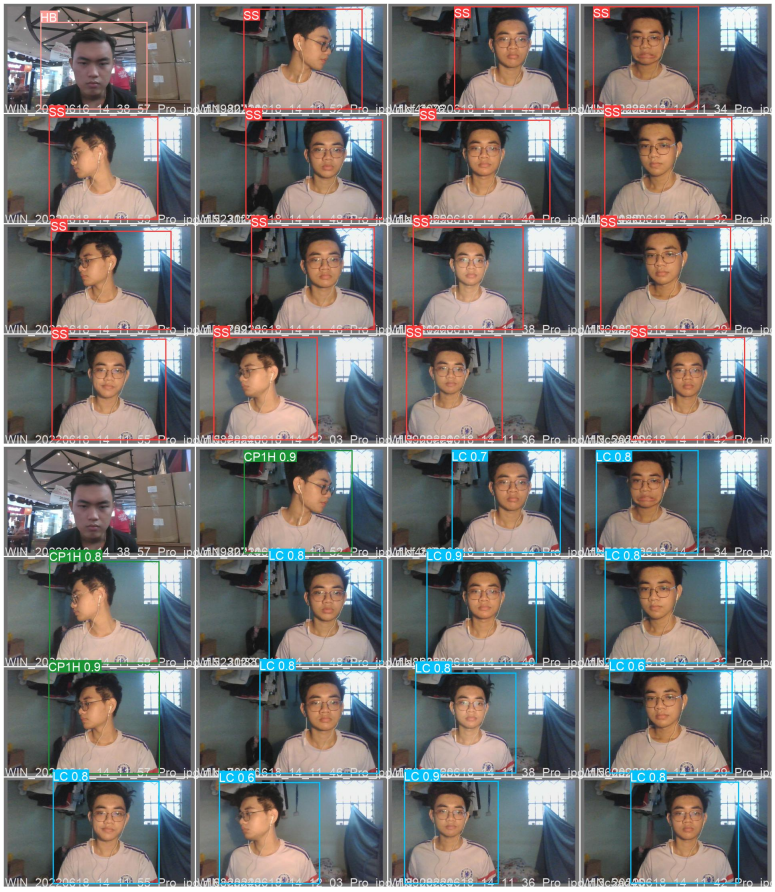


Figure 24: Validation batch 0 ground truths and predictions

Figure 25: Validation batch 1 ground truths and predictions

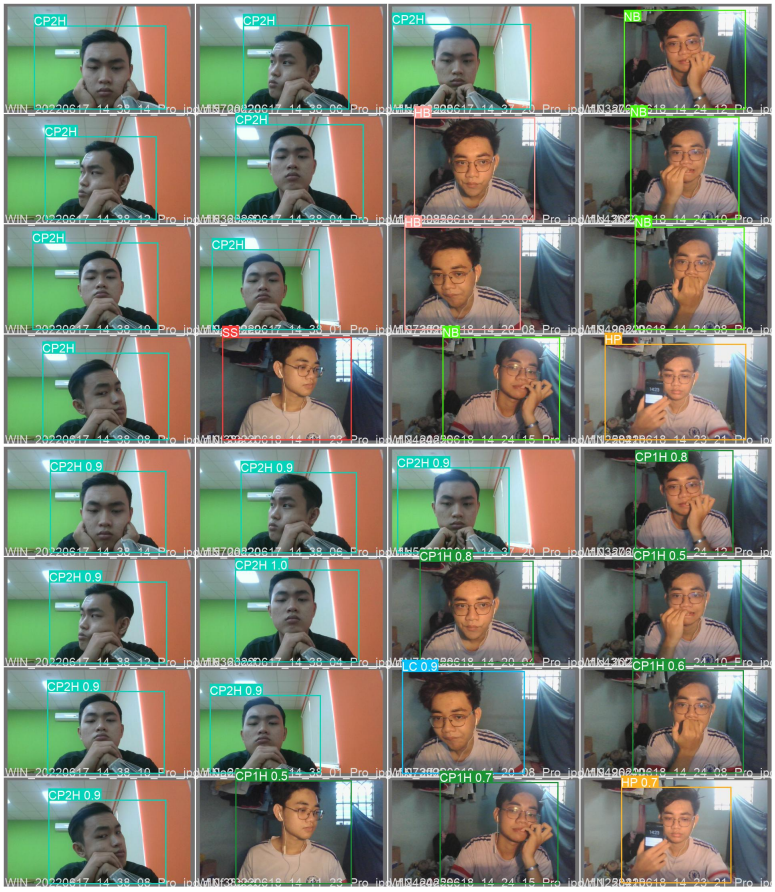


Figure 26: Validation batch 2 ground truths and predictions

2.4.3 Evaluation results based on test data

Our model's objective is being able to detect not only precisely but also not neglecting any classes, meaning every classes must have both high precision and high recall, so the metric that we will focus on is mAP@0.5 since mAP@0.5 are computed with both Precision and Recall at confident threshold 0.5.

Among all of the classes that we have tested, Napping, Legs on Chair, Resting on chair fared the best compare to the other classes. With mAP@0.5 equal to 0.954, 0.876, 0.833 respectively while the average value is only 0.723 (Base on observation on the Precision and Recall Graph). However, There are some classes that perform terribly, below the average value of mAP@0.5, namely Hunched Back, Head Scratching, and Face Scratching.

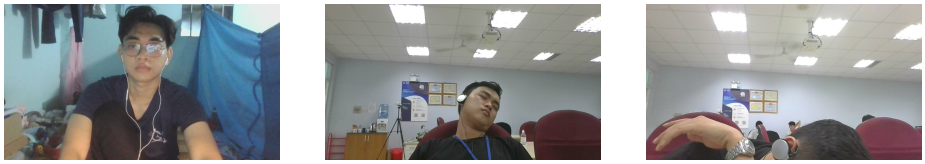


Figure 27: Best performing classes

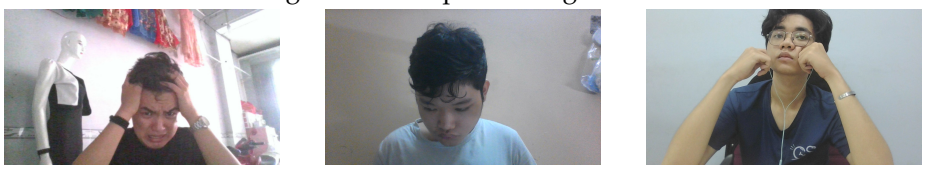
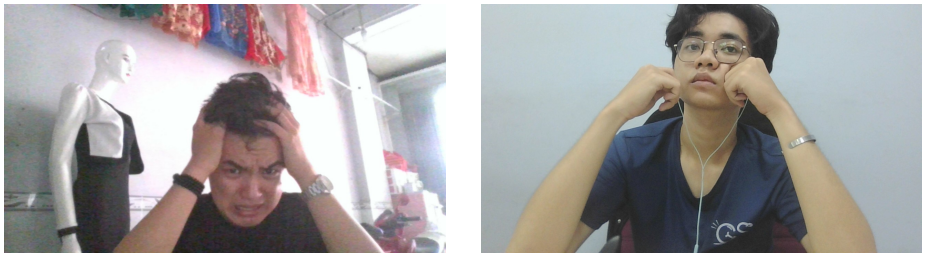


Figure 28: Worst performing classes

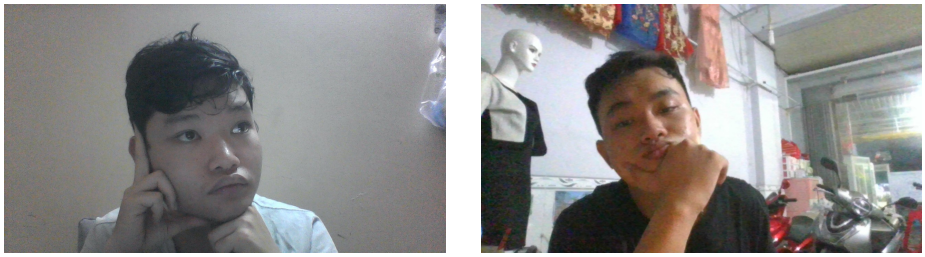
The reason for such abnormally big values is because the uniqueness of them. For instance, only the "Napping" images shown the top of our heads in the picture, only the "Leg on Chair" classes showed our knees, even our ankles occasionally, and only "Resting on Chair" images show our necks the most compare to other class.

On the other hand, some of the other classes are not so unique as the aforementioned classes, therefore there performance suffered greatly. To illustrate my point, The "Hunched back" images are quite similar to other classes like "Sitting straight", "Resting on chair", ..., The "Head scratching" and "Face scratching" images are not only alike to each other but also to other classes such as "Chin propping with 1 hand", "Chin propping with 2 hands", "Nail biting".

Figure 29: Hunched back in comparison



(a) Head and Face scratching



(b) 1 hand and 2 hands chin probing

Figure 30: Face and Head scratching in comparison

Another issue that we need to address is most of the worst performing classes have unbalanced sub-classes. For example, ”Head scratching” has three totally distinct sub-classes, namely ”Head scratching with left hand”, ”Head scratching with right hand”, and ”Head scratching with both hands”.

Figure 31: Head scratching sub-classes

3 Comparison

After we have done predictions with VGG16, Yolov4, Yolov5 on our test data, we found out that there are some profound similarities and stark differences among our models. Now, we will address and analyze them in depth so that we can decide which of our models perform the best in this project.

Confusion matrix of the models illustrate to us that:

Similarities:

- Napping (NguUpMat) and Stretching (VuonVai) both are the best classes across all of our models. Stretch's recall is acceptable in VGG16 (72%), but it is exceptionally better in Yolov4 (89%) and Yolov5 (89%). While Stretching performance is astonishing in Yolov5 with almost perfect statistics (98%), but not so impressive in Yolov4 (87%) and VGG16 (79%)
- Nail Biting (CanMongTay), Face Scratching (ChamTayLenMat) perform pretty badly across all three models. Regarding Nail Biting, it fared worst in Yolov5 with 0% recall, slightly better in VGG16 with 8% recall, and best in Yolov4 with 42% recall. Face scratching performs worst in Yolov5 with 10% recall, better in VGG16 with 23%, and best in Yolov4 with 36%.

Differences:

- The best performing classes of the models differ from each other. For instance, VGG16 has Napping with (97% Recall), Yolov4 has Chin Propping with 2 hands (90% Recall), and Yolov5 with Leg on Chair (98% Recall)
- The classes with the worst performance were Nail biting in Yolov5 (0% Recall), Face scratching in Yolov4 (36% Recall), and Hunched Back (7% Recall)

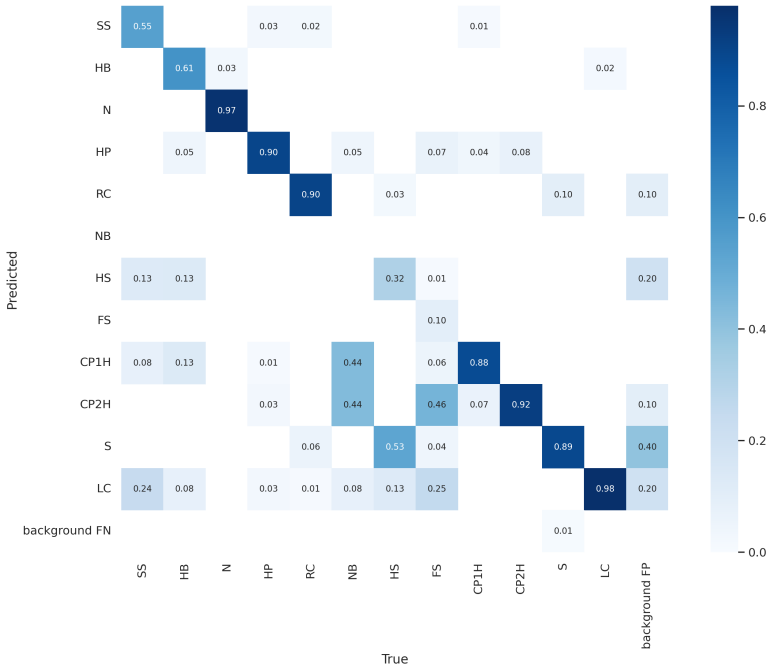
After we have analyzed the similarities and the differences of our three models, we have confirmed that the problems lie mostly in our data set and not because of the variety of our models.

		PRED											
		CamDT	CanMongTay	ChamTayLenMat	ChongCam1Tay	ChongCam2Tay	NgoiGacChan	NgoiGu	NgoiThang	NguNguaRaSau	NguUpMat	VoDau	VuonVai
TRUE	CamDT	55%	9%	2%	10%	5%	1%	1%	0%	3%	2%	1%	10%
	CanMongTay	9%	8%	8%	22%	7%	10%	0%	1%	15%	1%	7%	14%
	ChamTayLenMat	3%	4%	23%	14%	10%	7%	0%	0%	22%	5%	7%	6%
	ChongCam1Tay	9%	6%	18%	31%	11%	2%	3%	1%	0%	7%	1%	13%
	ChongCam2Tay	17%	7%	12%	10%	28%	3%	9%	1%	0%	3%	1%	10%
	NgoiGacChan	3%	1%	0%	0%	0%	87%	0%	0%	9%	0%	0%	0%
	NgoiGu	16%	0%	6%	7%	2%	0%	7%	1%	1%	27%	0%	32%
	NgoiThang	2%	16%	1%	17%	0%	10%	0%	42%	11%	0%	0%	0%
	NguNguaRaSau	0%	17%	8%	0%	0%	2%	0%	2%	64%	0%	2%	4%
	NguUpMat	1%	0%	0%	0%	0%	0%	0%	0%	0%	97%	1%	0%
	VoDau	6%	1%	3%	0%	0%	0%	0%	0%	23%	2%	42%	22%
	VuonVai	3%	0%	1%	0%	0%	0%	0%	0%	21%	2%	1%	72%

(a) VGG16

		PRED												
		CamDT	CanMongTay	ChamTayLenMat	ChongCam1Tay	ChongCam2Tay	NgoiGacChan	NgoiGu	NgoiThang	NguNguaRaSau	NguUpMat	VoDau	VuonVai	FN
TRUE	CamDT	80%	0%	0%	1%	0%	2%	0%	0%	0%	0%	0%	0%	16%
	CanMongTay	0%	42%	0%	9%	3%	0%	0%	0%	0%	0%	0%	0%	46%
	ChamTayLenMat	0%	0%	36%	19%	10%	0%	0%	0%	0%	0%	0%	5%	30%
	ChongCam1Tay	0%	0%	0%	58%	11%	0%	0%	1%	0%	0%	0%	0%	30%
	ChongCam2Tay	1%	0%	0%	1%	90%	0%	4%	0%	0%	0%	0%	0%	4%
	NgoiGacChan	2%	0%	0%	0%	0%	79%	1%	0%	0%	0%	0%	0%	18%
	NgoiGu	0%	0%	0%	12%	0%	3%	39%	0%	0%	0%	0%	0%	46%
	NgoiThang	0%	0%	0%	3%	0%	2%	0%	73%	0%	0%	0%	3%	18%
	NguNguaRaSau	0%	0%	0%	0%	0%	0%	0%	7%	67%	0%	1%	0%	25%
	NguUpMat	0%	0%	0%	0%	0%	0%	21%	0%	0%	61%	1%	0%	17%
	VoDau	0%	0%	2%	0%	0%	0%	0%	0%	10%	0%	58%	19%	10%
	VuonVai	0%	0%	0%	0%	0%	0%	0%	0%	10%	0%	0%	89%	1%

(b) Yolov4



(c) Yolov5

Figure 32: Confusion matrices in comparison

According to the statistics we collected from the models, we found that:

- When it comes to Precision, Yolov4 show us the best results with 74% precision, slightly better than Yolov5 with 71.8%, and completely overshadowing the small result of VGG16 with only 56%.
- In terms of Recall, Yolov4 still take the lead with 75% compare to only 60% of Yolov5, and 46% in VGG16.

With that being said, We could totally agree that Yolov4 is the most promising model in this project.

4 Applications and Development Direction

4.1 Applications

The problem can be extended to more users, in the 4.0 technology era when laptops and computers appear everywhere and the frequency of use increases, the adjustment of sitting posture or correction of bad habits is extremely important. Can be applied for parents to adjust for children or remind ourselves when unconsciously performing bad habits.

4.2 Development Direction

Data

- Because this is a problem for the subject project, the dataset we prepare is encapsulated in 3 people and the shooting locations are not diverse, so the model can be easily overfitting, so it is necessary to increase the intensity. diversity for data to extend users.
- Pay attention to clean Data, although the data needs to be diverse, it must have quality and ensure the set constraints.
- Need to ensure a balance between classes when increasing data.

Model

- Improving data will help the model avoid overfitting better.
- Consider using a pretrained model or maybe train the entire model from scratch for more optimal results.
- Adjust the complexity of networks to see if the results are better, learn to add early_stopping to models Yolov4 and Yolov5.

Extension We think it is possible to expand the problem and develop a similar problem to identify and inform students's personal working behaviors during class time through an external camera linked to the teacher's computer who teach in that class.

5 Conclusion

6 References

VGG16:

[1] https://medium.com/m/global-identity?redirectUrl=https%3A%2F%2Ftowardsdatascience.com%2Fstep-by-step-vgg16-implementation-in-keras-for-beginners-a833c

[2] https://keras.io/api/applications/vgg/

Yolov4:

[1] https://github.com/AlexeyAB/darknet

[2] https://www.youtube.com/watch?v=N-GS8cmDPog&t=1056s

[3] https://www.youtube.com/watch?v=ebAykr9YZ30&t=308s

Yolov5:

[1] https://github.com/ultralytics/yolov5

[2] https://docs.ultralytics.com/