ADVANCE Labs - Harassment in Images Detection Lab

The development of this document is partially funded by the National Science Foundation's Security and Trustworthy Cyberspace Education, (SaTC-EDU) program under Award No. 2114920.

1 Lab Overview

In this lab, you will keep learning about how AI/ML can be used to detect societal issues such as cyberbullying. Cyberbullying is bullying performed via electronic means such as mobile/cell phones or the Internet. The objective of this lab is for students to gain practical insights into online harassment such as cyberbullying via images, and to learn how to find cyberbullying from images using pre-trained AI/ML solutions to defend against this problem.

In this lab, students will be given a starter-code to find a cyber bullying and non-cyberbullying images from publicly available dataset. Their task is to follow the instructions provided in the Jupyter notebook, to use a pre-trained AI/ML model on the given dataset. In additon students will learn the Approach towards analysing the cyberbullying in images using a dataset, there are four steps:
(1) Understand and identify the factors related to cyberbullying in images.
(2) Load the pre-trained model.
(3) Fine-tune the model with a small dataset.
(4) Evaluate your fine-tuned model with the test dataset

More over this lab covers the following topics:

- Detection of a cyberbullying in images
- AI-based classifier models to predict cyberbullying vs. non-cyberbullying in images

2 Lab Environment

This ADVANCE lab has been designed as a Jupyter notebook. ADVANCE labs have been tested on the Google Colab platform. We suggest you to use Google Colab, since it has nearly all software packages preinstalled, is free to use and provides free GPUs. You can also download the Jupyter notebook from the lab website, and run it on your own machine, in which case you will need to install the software packages yourself (you can find the list of packages on the ADVANCE website). However, most of the ADVANCE labs can be conducted on the cloud, and you can follow our instructions to create the lab environment on the cloud.

3 Lab Tasks

3.1 Getting Familiar with Jupyter Notebook

The main objective of this lab is to learn how pre-trained AI/ML model can be used to detect online harassment through images, such as cyberbullying. Before proceeding to that, let us get familiar with the Jupyter notebook environment.

Jupyter notebooks have a Text area and a Code area. The Text area is where you'll find instructions and notes about the lab tasks. The Code area is where you'll write and run code. Packages are installed using `pip`, and need to be preceded with a `!` symbol. Try accessing the lab environment for this task here.

The lab has three areas: one text area and two code areas. Follow the instructions for the three areas, fill the three areas with the instructed content and add a screenshot to your report.

3.2 Cyberbullying Detection in images

In this lab, you will use pre-trained AI/ML model to detect cyberbullying in images. You will use a dataset of real world images to train your AI model, evaluate the performance of a pre trained AI classifier model and check the result with one random instance from the dataset. You can access the lab by clicking here.

3.2.1 Download the pre-trained model, test dataset and the dependencies

In this lab, we provide three publicly available dataset: auxes dataset, poses dataset, and images.You can download the model and dataset as per the lab instruction.

In this lab, you will be using the auxes, poses, and image data set, which consist of real-world cyberbullying images. Run all the code of the lab. Report your results according to each designed task and discuss your idea at the end.

3.2.2 How to identify cyberbullying in images

Follow the instructions in the text areas and run the subsequent codes to load example image from a code snippet, as follows. Here is a sample from the lab:

```
testImage = img.imread('/content/cyberbullying_data/cyberbullying_data_splits_clean/
    test/cyberbullying/7.s-s-unshaven-sad-ashamed-man-doing-loser-sign-hand-fingers-his
    -front-funny-depressed-face-expression-s-139158713.jpg')
```

There are 5 factors to measure cyberbullying in images

1. Body-pose
2. Facial Emotion
3. Object
4. Gesture
5. Social Factors

Refer to the table from lab environment which shows analysis of cyberbullying factors in images.

3.2.3 Load Dataset

Follow the instructions in the text areas and run the subsequent codes to load your data from a predefined class, as follows. Here is a sample from the lab:

```
class PosesDataset(Dataset)
```

This predefined class checks cyberbullying and non-cyberbullying images from the dataset, and include the generated tokens in your report. You can add a code block to run your code.

Generate a test set from the poses and auxes dataset by following the lab instruction. Here is a sample from the lab:

```
test_set = PosesDataset('cyberbullying_data/cyberbullying_data_splits_clean/test/', '
    cyberbullying_data/cyberbullying_poses/test/', 'cyberbullying_data/
    cyberbullying_data_auxes/test/')
```

3.2.4 Load pre-trained AI classifier model

After you have loaded the dataset, the next task is to load a pre-trained AI classifier model. Follow the lab instruction to load the AI model. Do you know any other classifier model which can be used here?

```
# load vgg16 pre-trained model
orig = models.vgg16(pretrained = True)
```

3.2.5 Generate the detection results from the test dataset

Now it is time to run your fine-tuned model on a test dataset. Run your model on the test data and report your results here. Use lab instruction:

```
with torch.no_grad():
    ...
print('Test loss is: {:.3f}'.format((sum(running_loss) / len(running_loss)).item()))
print('The accuracy for test dataset is: {}%'.format((correct / total) * 100))
```

3.2.6 Task1: Write code to generate result report containing: Accuracy, Precision, Recall, and F1-Score

After Generating the detection results from the test dataset. Now, let's learn some different evaluation metrics. You are supposed to read the reference first and complete the code to generate a result report containing Accuracy, Precision, Recall, and F1-Score. Please refer to the given description.

```
# TODO: Complete the following code to calculate the accuracy, precision, recall and
    F1 score.
acc =
precision =
recall =
f1 =

print('The accuracy for test dataset is: {}%'.format(acc * 100))
print('The precision for test dataset is: {}%'.format(precision * 100))
print('The recall for test dataset is: {}%'.format(recall * 100))
print('The f1 score for test dataset is: {}%'.format(f1 * 100))
```

3.2.7 Task2: Write code to plot the confusion matrix

Please look into the given link to understand what is the confusion matrix. https://en.wikipedia.org/wiki/Confusion_matrix Write code to plot the confusion matrix (you are allowed to borrow any python tools, such as scikit-learn (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html))

3.3 Check with one random instance from the test dataset

After generating results from the test dataset, you now need to randomly select an instance in the test dataset using,

```
picture_index  = "11" #@param [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
index = int(picture_index)
instance = test_set[index]
```

Plot the image for results as per the lab instructions.

3.3.1 Task3: Complete the given code to fine-tune the model with the training dataset

After passing the pre-trained checkpoints to the VGG model, to make sure the model can have some knowledge of our specific dataset, we normally fine-tune the model.

Follow the instruction to prepare the training data, and the optimizer. Then, you need to complete the following code cell to fine-tune the model with the training dataset.

```
# TODO: complete the following code by replace "___", to mimic fine-tune (further
    training) the model
ft_model.train()

epochs = ___
for epoch in range(epochs):
    for i, data in enumerate(train_loader):
        inputs = data['image'].to(device)
        aux = data['aux'].to(device)
        labels = data['label'].to(device)
        optimizer.zero_grad() # zero the parameter gradients
        outputs = ft_model(inputs, aux) # forward pass
        loss = criterion(___, ___) # compute loss via comparing model's outputs and
    our predefined labels
        loss.backward() # backward pass
        optimizer.step() # update weights
        running_loss.append(loss.item()) # save loss
        _, predicted = torch.max(outputs.data, 1) # get predictions
        total += labels.size(0) # update total
        correct += (___ == ___).sum().item() # update correct predictions
        if i % 50 == 0: # print every 50 mini-batches
            print('Epoch: %d, Iteration: %d, Loss: %.4f, Accuracy: %.4f' % (epoch, i,
    loss.item(), correct / total))
            correct, incorrect, total = ___, ___, ___ # reset correct, incorrect, and
    total. hint: float is better than int
```

Note: This code is a very basic version that helps us keep training the model with the training set, Recall from the last lecture, we can have a validation set to help us decide when to stop training the model. If you are interested, you can try to split the training set into a training set and a validation set and use the validation set to help you decide when to stop training the model.

3.3.2 Task4: Write code to evaluate the fine-tuned model

Now, with your fine-tuned model, you can further test it with the same test dataset:

```
ft_model.eval()
# TODO: write code to evaluate the model on the test set
```

Then, let us compare the the two results from the different models you have. Is the fine-tuned model's prediction accuracy better than the previous model? If not, think about the possible reasons for it.

3.3.3 Task5: Write code to visualize the image and predict

After Generating the detection results, you now need to write code to visualize a specific image; refer to the given path.

Then find the `picture_index` of the chosen image by comparing it with the previous visualization cell. Get its prediction with your fine-tuned model and check if it is correct.

4 Submission instruction

You need to submit a detailed lab report, with screenshots, to describe what you have done and what you have observed. You also need to provide an explanation for the observations that are interesting or surprising. Please also list important code snippets followed by an explanation. Simply attaching code without any explanation will not receive credits.