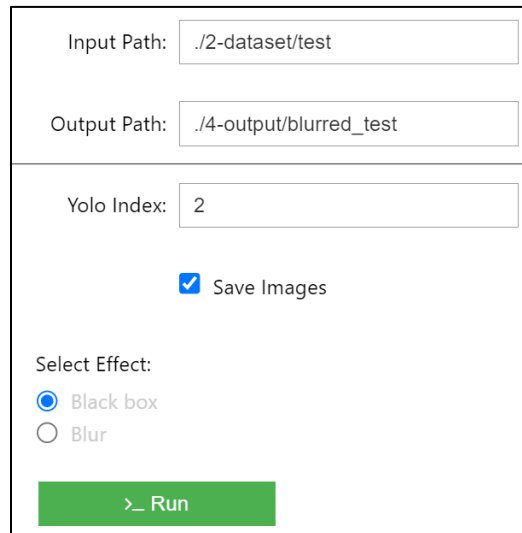**Inviol Technical Test Project**

Producing Head Labels in YOLO.txt Format to Reduce Manual Work by Labelers

This is how the final product looks like:

1- Clone the repo https://github.com/BunNybuger/Inviol_TechTest
2- Open the head_detector.ipynb file
3- Create a new work environment (optional)
4- Install the requirements.txt by running first cell (This will take a while)
5- Run the second cell and input the variables as desired
6- Click RUN! (The first run might take longer due to downloading dependencies and you may need to restart kernel after installing dependencies.)

Since Yolo .txt format was needed, the code generates .txt files with the same name as each image in the following format:

<center><class_index> <x_center> <y_center> <width> <height></center>

All coordinates are normalized by image width and height, ranging from 0 to 1.

Since question mentioned "people" and "helmets" are tagged already, the third class, "head," is assumed to be the class with yolo_class_index = 2 (users can change this if needed).

Additionally, apart from saving the images, the output in Jupyter Notebook is previewed as the model works, allowing users to check the work quality while waiting for the model to process.
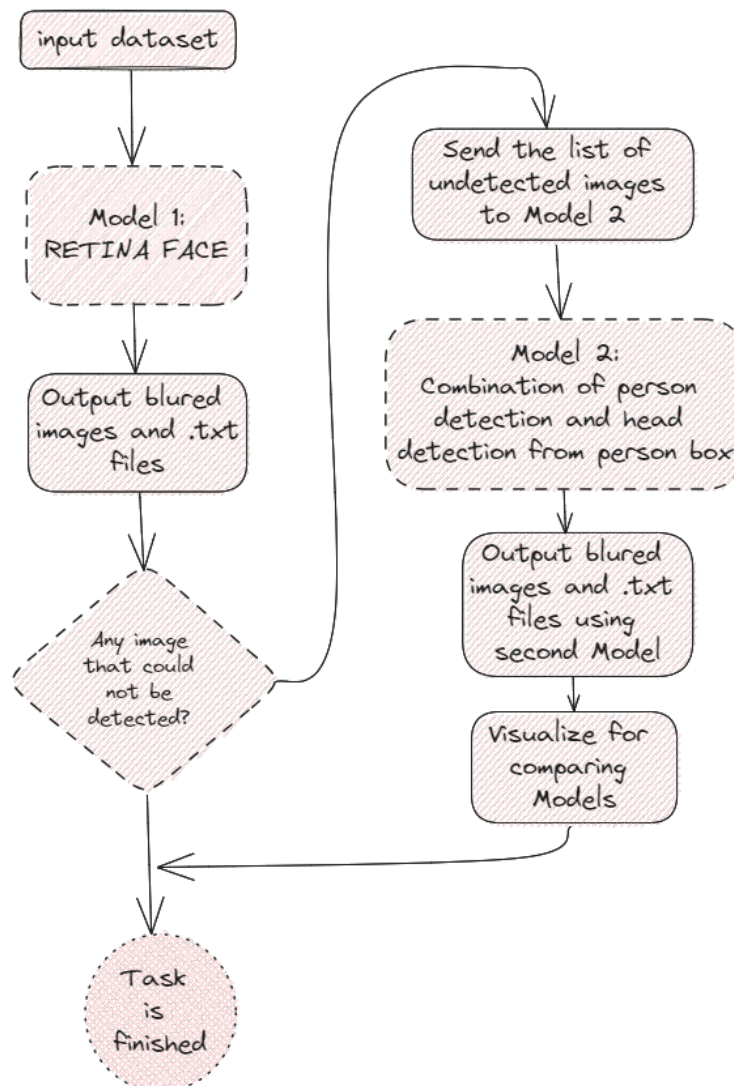
Original Image   Processed Image

To increase the performance and detect as many faces/heads as possible, the pipeline consists of two separate models. Here's how the pipeline works:
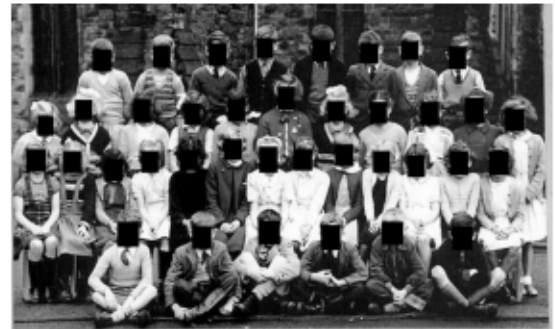
My path to develop the final product:

My strategy was to first attempt to detect as many faces as possible, I searched online and found some good result from the RETINA face model. I used this source: https://github.com/serengil/retinaface to implement it. Here are the results after implementation:

- Speed: approx. 2.45 seconds per pic

1 Some Good results

For more accurate head detection and to address the mentioned problems with Retina, a second method was used:

- – The image is first fed to a pretrained YOLO model to detect a person.
- – Then, the detected person's bounding box is fed to another pretrained YOLO model to detect the head.
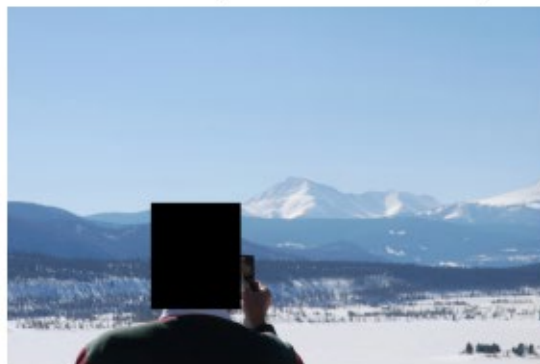
Here are the results:

- Speed: approx. 2.25 seconds per picture
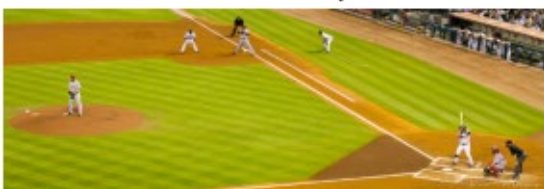- Good for detecting heads from angles that Retina is not trained on.
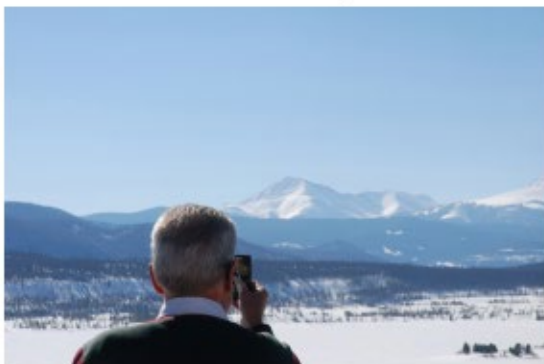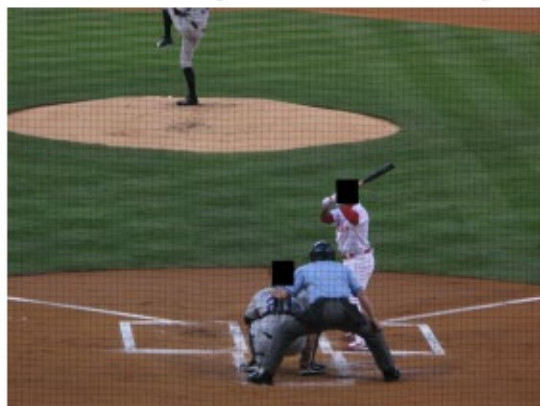
Retina output   Head from person box output

2 Failed cases by retina, detected by second model

The main file to run the project is Final.ipynb, but you can also run it from Mixed_pipline.py and Mixed_pipline.ipynb (or play with the codes from there! 😊).

More Advanced Usage:

If you want to use this script directly on your dataset with preexisting YOLO-style labels (i.e., you want to add head labels to your existing labels), follow the steps below:

1- Backup your dataset!
2- In the script, set the output path to be the same as the input path.
3- Set Visualization to False.

4- Find this line in the code: with open(txt_output_path, 'w') as f: and change 'w' to 'a' (append mode instead of write mode). It should be like: with open(txt_output_path, 'a') as f:

Note: This will work flawlessly if your preexisting txt files have \n at the end.

My understanding from the task explanation meeting was that you need some model immediately, just to get the job done. With that in mind, my strategy was to produce a model that could do the job to a good extent, as fast as possible. So, I first solved the problem quickly using the Retinal model, which provided a model that did the job. This was done in approximately 5 hours. After that, I improved the results by testing other models and creating a complete pipeline with two different models.

Here's how much time I spent on each task:

Row	Task description	Time spent
1	Searching for different models, choosing Retina, testing it, and preparing a dirty initial implementation that does the job! (i.e., produce the YOLO .txt files)	5 hrs
2	Evaluating Retina problems and trying to solve them	1 hr
3	Preparing the second model and making it work	2 hrs
4	Preparing a pipeline and mixing two models	3 hrs
5	Preparing the nice Widget to make the job easy for users, cleaning the code, making it more maintainable	3 hrs
6	Preparing the report, requirements.txt, and finalizing	2 hrs
7	Final Testing	1 hr

Suggestions for Further Improvements in this Project:

To increase speed:

Currently, the model is processing one image at a time. It is possible to change the method the dataset is loaded to use batch processing, which can have a significant impact on the running time.

To increase accuracy:

Using a better model to detect heads from a person's box and also detect the person from the image. The current implementation used a pretrained model that was readily available from a previous project. However, there are better person detectors out there, and substituting another model in place of the model available in .\1-models\Person_Detector.pt could yield better results, as seen in the last image.

Recommend specifying whether you are looking for face detection or head detection. If head detection is needed, the detected boxes using Retina should be multiplied by a factor of approximately 1.4 to return a bigger area covering the entire head.

To Ensure that code runs everywhere:

To make the code run regardless of the underlying environment even more easily, it is recommended to publish a package and or using Docker.

Here are some pictures where the second model performed well:

And these are some strange and funny mistakes made by the second model. This is why the pipeline is using Retina first and then the second model. (P.S. These problems are solvable, easily! Just replace "*.\1-models\Person_Detector.pt*" with a better person detector model) I did not do that because I assume you data set is more of people and do not have animals.

Mostafa Papen

23/07/2023