3. My proposed improvements:

- Use pip-compile or similar to handle the dependencies in order to be sure about what packages are you using and being able to replicate it in another environment. -> **IMPLEMENTED**
- Check the format and the validity of the APIs input, in order to avoid the user using incompatible values and returning an appropriate error in order to make the user aware of the problem. -> **IMPLEMENTED**
- Writing more logs for debugging and using a logger instead of the prints, this enables writing logs using different levels to separate the ones important in the production environment from the ones used for debugging.
- Handling possible errors in the execution and returning specific error codes or messages to let the user know what is happening.
- Add unit tests for the adapters class functions.
- Automate the deployment and testing using a Makefile for example.

5. You can serve any tensorflow model using the tensorflow serving, you just need to:

- Having the models files saved in an accessible folder with a numbered subfolder to handle the different versions.
- Adding the models in *object-counter/tmp/model/model_config.config* file by specifying the **name**, **base_path** and **model_platform** for each model.
- Call the desired model and version by changing the url http://localhost:8501/v1/models/{**model_name**}/versions/{**version_number**}. This part can be changed in **TFSObjectDetector** initialisation.

The model to use can be passed for example as an environment variable if we want it to depend on the environment the app is running on, or we can let the user choose the model by adding it as an input parameter for the API request.