Deep vectorization P.R

Egiazarian Vage

This is a short report about the project. You can find more details and code in the repository https://github.com/adasegroup/Deep-vectorization-PR.

1. Problem statement

Writing research code is much different than writing code for the product. In research, one part of code can be written and rewritten multiple times with drastic changes, one more significant difference is that, code often written in a hurry in notebooks(for convenience and fast experiments) and only part was in file format. Because code is rapidly involving and changing it is hard to maintain quality. Not helping the fact that different people writing code in different style and quality. Forcing one style is hard when there is not much time till a deadline. In the end, you end up with a lot of code and branches that should be refactored and rewritten before publishing the code to the public.
The goal of this project is to make an easily reproducible repository (more precisely continue upgrading existing) and good documentation for public use. We have 2 repositories with 15 branches of research code from the ECCV2020 paper Deep vectorization of technical drawings and repository in Github repository, with more than already 22000 lines of code, to combine all the parts. We would like to make the code easy to use and understand and mainly easy to run.

2. Main challenges

Combine code from different branches that are not compatible and make an easily readable and reproducible code. Another challenge is to make documentation for functions in the code.

3.Description of a baseline solution. Some other implementations which will serve as inspiration or baseline for your work
The baseline solution is to release the initial code(code form research) without refactoring(with all branches). Another solution is to live the already refactored repository as it is, without any further improvement and commits.

3.1. Pros and cons of these solutions

1. Releasing research repositories:
   Pros: All raw code would be available to anyone with all experiments and legacy code.
   Cons: Code is a research mess. Almost nobody would try to understand or try to use it.
2. Keeping refactored repository as it is:
   Pros: Time savings.
   Cons: Not all functionality carefully present and well documented(No good documentation, no docker file, and e.t.c. for more detail look at https://github.com/Vahe1994/Deep-Vectorization-of-Technical-Drawings)

3.2. Ideas on how to improve it or how you are going to use it.
Add Jupyter notebooks with an explanation of how to evaluate functions, add a docker file, and a list of requirements. Add trained models and documentations. For more details look at the list below.

4. Roles for the participants
Because this team consists of only one member, all proposed tasks would be done by me. It includes writing project reports, code, and documentation. In the list below you can find a brief description of tasks:

1. Create code for visualization(rewrite tensorboard code to make it work or use wandb).

2. Create a python script for evaluating models on images.
3. Make Jupyter notebooks to show how to use different parts of pipelines with descriptions.
4. Create documentation for the repository.
5. Make description for most used functions
6. Make a docker file or docker image for the repository.
7. Make available models(some of them should be trained again).
8. Make setup.py.
9. Make requirements document.
10. Correct train.py to make it work and where possible refactor code.

6. Project Structure

The project has a module-like structure.
The main modules are cleaning, vectorization, refinement, and merging(each module has an according to folder).
Each folder has Readme with more details. Here is the brief content of each folder.

***** cleaning - model, script to train and run, script to generate synthetic data
***** vectorization - NN models, script to train
***** refinement - refinement module for curves and lines
***** merging - merging module for curves and lines
***** dataset - scripts to download ABC, PFP, cleaning datasets, scripts to modify data into patches, and memory-mapped them.
***** notebooks - a playground to show some function in action.
***** utils - loss functions, rendering, metrics
***** scripts - scripts to run training and evaluation

7. Evaluation results

Look at notebooks pretrain_model_loading_and_evaluation_for_line.ipynb and pretrain_model_loading_and_evaluation_for_curve.ipynb , for an example how to run primitive estimation
and refinement for curve and line.

P.s. For results on bigger datasets please look at the according to paper in the section with evaluation(p 11, table 1.).

5.Link to the GitHub repository
1) Github repository for the course - https://github.com/adasegroup/Deep-vectorization-P.R.
2) Official Github repository for article Deep Vectorization of Technical Drawings - https://github.com/Vahe1994/Deep-Vectorization-of-Technical-Drawings

References:
1) V. Egiazarian, O. Voynov, A. Artemov, D. Volkhonskiy, A. Safin, M. Taktasheva, D. Zorin, and E. Burnaev.
Deep vectorization of technical drawings. arXiv preprint arXiv:2003.05471, 2020
2) Wandb site - https://wandb.ai/site