Report
Pattern Recognition

Shekhar "Shakes" Chandra

Version 1.3 Beta

Creating and maintaining open source algorithms either for pattern recognition and analysis or in general in the current landscape of big data and artificial intelligent, is important for anyone working with data, whether it be an open source developer, a researcher, an engineer or a data/computer scientist.

In this report, you will choose a recognition problem from a list provided and solve it with the concepts and materials taught in this course. The problems (and associated data) provided are very close to real research problems that are still being solved! It is hoped that you get an opportunity to have a go at real world data crunching and pattern analysis by creating your own solutions to these problems. This will give you an opportunity to practice all the necessary skills required to develop and maintain your own open source project(s) and writing a mini report in the form of a read me file and a GitHub pull-request. These are skills that are highly desirable in many employment opportunities involving machine learning, big data and artificial intelligence.

You will also have an opportunity to learn how to collaborate on an open source project by forking a pre-exiting repository and adding your project to it via a pull-request. You will then be required to write the documentation and instructions for that algorithm. You will be required to complete the necessary software engineering tasks to complete the report with tasks such as design, commenting, pull requests etc. Your algorithm will directly contribute to the open source PatternFlow library on GitHub. Upon submission, the teaching team will provide a professional code review of your submission, so that you get an idea of what to expect when doing such a submission at a work place or for an open source project.

1 The Report

Your objective is to solve a recognition problem for your choice (see below), document your solution and code, then submit it for code review and acceptance into the course repository. The problems are divided into three difficulty levels: Easy, Normal and Hard. Additional marks are awarded for completing the problems with Normal and Hard difficulties. The following problems (and associated requirements) are available with the difficulties indicated:

ATTENTION: The tasks are currently being finalised. Please check Blackboard for latest version of this file.

1. Segment the ISIC data set with the Improved UNet [1] with all labels having a minimum Dice similarity coefficient of 0.8 on the test set. [Easy Difficulty]

2. Create a suitable multi-layer GCN model to carry out a semi supervised multi-class node classification using Facebook Large Page-Page Network dataset [2] with reasonable accuracy. (You may use this partially processed dataset where the features are in the form of 128 dim vectors.) You should also include a TSNE or UMAP embeddings plot with ground truth in colors. [Normal Difficulty]

3. Detect lesions within the ISIC data set with the Mask RCNN [3] (pre-trained Mask RCNN model allowed) or a YOLO network such as the original YOLO [4] with all detections having a minimum Intersection Over Union of 0.8 on the test set and a suitable accuracy for classification. [Normal Difficulty]

4. Segment the (downsampled) Prostate 3D data set with the UNet3D [5] with all labels having a minimum Dice similarity coefficient of 0.7 on the test set. See also CAN3D [6] for more details and use the data augmentation library here. [Normal Difficulty]

5. Classify Alzheimer's disease (normal and AD) of the ADNI brain data (see Appendix for link) using the Perceiver transformer [7] set having a minimum accuracy of 0.8 on the test set. [Hard Difficulty]

6. Create a classifier based on Siamese network [8] to classify Alzheimer's disease (normal and AD) of the ADNI brain data set (see Appendix for link) having a minimum accuracy of 0.8 on the test set. [Medium/Hard Difficulty]

7. Create a generative model of one of the OASIS brain, ADNI brain or the OAI AKOA knee data set (see Appendix for links) using a VQVAE [9] or VQVAE2 [10] that has a "reasonably clear image" and a Structured Similarity (SSIM) of over 0.6. [Hard Difficulty]

8. Create a generative model of one of the OASIS brain, ADNI brain or the OAI AKOA knee data set (see Appendix for links) using StyleGAN [11] or StyleGAN2 [12] that has a "reasonably clear image". [Hard Difficulty]

A listing and links to the various data sets is provided in Appendix A. Additional recognition problems may become available and will be added to this list. You are also able to propose your own network for the above problems or an entirely different problem to the course coordinator for approval. The teaching staff will decide what constitutes "reasonably clear image" and the SSIM is not strict, but both will be negotiable with the staff depending the difficulty of the problem.

2 Submission

The submission of the final report will be done via a GitHub pull-request and a Turn-it-in submission. The pull request should contain

1. the code of your algorithm and test script(s) written to run the algorithm (20 Marks),

2. commit log to provide evidence of originality (5 Marks, Pass Hurdle)

3. documentation via a README.md file (10 Marks)

4. pull-request, with clear description, completed properly (5 Marks, Pass Hurdle)

The Turn-it-in submission should be a PDF version of the README.md file. **You are required to make the pull request to receive any marks for the assessment. You must complete a pull-request (with a working version of the algorithm committed) at least once before the assessment deadline.** The following sections describe the individual tasks to be completed in detail.

3 Checklist

The following steps should be followed to complete this assessment task:

1. fork the PatternFlow library on GitHub
2. checkout the topic-recognition branch within your fork
3. choose a problem via section 1 and create a folder for your problem in the 'recognition' folder of the PatternFlow trunk with a meaningful name
4. complete the solution to the recognition problem by building your model, making sure to commit any progress to the topic-recognition branch of your fork.
5. complete the test driver script that calls and runs your algorithm
6. complete the documentation of your algorithm
7. lodge a pull request
8. complete a Turn-it-in submission of the PDF of the README.md file.

4 Recognition Problem (20 Marks)

Your implementation must include the following files

1. **"modules.py"** containing the source code of the components of your model. Each component must be implementated as a class or a function
2. **"dataset.py"** containing the data loader for loading and preprocessing your data
3. **"train.py"** containing the source code for **training**, **validating**, **testing** and **saving** your model. The model should be imported from "modules.py" and the data loader should be imported from "dataset.py". Make sure to plot the losses and metrics during training
4. **"predict.py"** showing example usage of your **trained** model. Print out any results and / or provide visualisations where applicable
5. **"README.MD"** to sufficiently document your project (see Section 6).

Note: You may create other helper files such as "utils.py" to better organise your project. You may reuse some of your own elements from a previous demo, but the final model must be substantially different from your previous work.

Marking Criteria

1. Algorithm solves the respective pattern recognition problem appropriately (5 Marks)
2. Algorithm implemented in TF/PyTorch demonstrated and functions as intended to solve the respective problem (3 Marks)
3. Good design implemented in TF/PyTorch that solves the problem (1 Marks)
4. Commenting (1 Mark)

5. Algorithm is normal or above difficulty while implemented in TF/PyTorch while solving the respective problem (5 Marks)
6. Algorithm is hard difficulty while implemented in TF/PyTorch while solving the respective problem (5 Marks)

5 Commit Log (5 Marks, Pass Hurdle)

Requirements

1. **Ensure that when you are implementing your algorithm to solve the recognition problem, you are committing regularly and keeping a detailed commit log.** You should for example commit whenever you reach any small milestone, such as creating a module or a working test script, completed the algorithm and completed the documentation etc.
2. **You must have a reasonable commit log showing progress and development of your algorithm. You should not commit your entire submission as one commit.** You will not receive any marks if your algorithm only contains a single commit. You may not pass the assessment if your commit log does not provide sufficient evidence of individual work and originality.

Marking Criteria

1. Meaningful commit messages with evidence of individual work and originality (2 Marks)
2. Progressive commits and logical log structure providing evidence of individual work and originality (3 Marks)

6 Documentation (10 Marks)

Once your algorithm has been implemented, you will need to provide sufficient comments in your code and documentation of your algorithm in a README.md file.

Requirements

1. **The readme file should contain a title, a description of the algorithm and the problem that it solves (approximately a paragraph), how it works in a paragraph and a figure/visualisation.**
2. It should also list any dependencies required, including versions and address reproduciblility of results, if applicable.
3. provide example inputs, outputs and plots of your algorithm
4. The read me file should be properly formatted using GitHub markdown
5. Describe any specific pre-processing you have used with references if any. Justify your training, validation and testing splits of the data.

Marking Criteria

1. description and explanation of the working principles of the algorithm implemented and the problem it solves (5 Marks)
2. description of usage and comments throughout scripts (3 Marks)
3. proper formatting using GitHub markdown (2 Mark)

7 Pull Request (5 Marks)

Once your algorithm module, test script and read me file are completed, lodge a pull-request to the Pattern-Flow repository via the topic-recognition branch. **A pull-request must be made to receive any marks for this assessment. This pull request must have a working version of the algorithm committed.**

You can see a guide on creating pull requests via GitHub here and a guide about pull-requests via GitHub here. See also this tutorial by Atlassian that explains the concept quite well.

Marking Criteria

1. Creating a pull-request into correct branch with a working and demonstrable version of the algorithm (2 Marks)
2. Incorporating feedback into the pull request (2 Marks)
3. Description of and comments within the pull request (1 Mark)

8 Tips and Advice

It is best advised to try an Easy problem first especially if you are not so confident with TF/PyTorch and problems provided. Once working, expand up to the Normal problem. For generative problems, try to get your code working on a simpler data set such as the MNIST first and then attempt it with the data for your problem. You should see the previous years pull requests and the type of expected content. Hard problems should only be attempted if you are extremely confident with your deep learning!

9 Academic Integrity and Misconduct

Academic Integrity is a core value of the UQ community and as such high academic integrity expectations apply to assessments. Using online resources for self-learning is fine, however **simply re-using entire blocks of code for them is not acceptable and may result in a violation of the Student Charter as a UQ student and an investigation into Academic Misconduct**. You must follow the guidelines for Academic Integrity found here, which also applies to code. Where possible, you should write your own code and if certain elements are required from online resources should be referenced in your pull-request or code. For more information see the UQ policy here and a tutorial on understanding this issue further go to the official UQ tutorial.

10 Conclusion

By completing this report, you will obtain experience in developing and maintaining your own open source project, how to contribute to other open source projects and good practices in algorithmic design and implementation. All the while, you will become even more knowledgeable in Tensorflow and pattern recognition!

Appendix

A Datasets

1. OASIS Brain data set (9K images) - This is part of the OASIS brain study with segmentation labels. The preprocessed version of this data set can be found on the course Blackboard site (under Course Help/Resources).
2. OAI Acelerated Osteoarthritis knee data set (18K images) - This is part of the Osteoarthritis Initiative and comes with only labelled laterality (left/right knee labelling) in the filename. The preprocessed version of this data set can be found on the course Blackboard site (under Course Help/Resources).
3. ISIC 2017 challenge data for skin cancer - This is part of the ISIC 2017 challenge and comes with segmentation labels and lesion cass labels.
4. The ADNI dataset for Alzheimer's disease. Look at the DOWNLOAD > Image Collections > Advanced Search area to download the data. The preprocessed version of this data set can be found on the course Blackboard site (under Course Help/Resources) and has two classes: Alzheimer's disease (AD) and Cognitive Normal (CN).

References

[1] F. Isensee, P. Kickingereder, W. Wick, M. Bendszus, and K. H. Maier-Hein, "Brain Tumor Segmentation and Radiomics Survival Prediction: Contribution to the BRATS 2017 Challenge," Feb. 2018. [Online]. Available: https://arxiv.org/abs/1802.10508v1

[2] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale Attributed Node Embedding," *arXiv:1909.13021 [cs, stat]*, Mar. 2021, arXiv: 1909.13021. [Online]. Available: http://arxiv.org/abs/1909.13021

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2980–2988.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv:1506.02640 [cs]*, May 2016, arXiv: 1506.02640. [Online]. Available: http://arxiv.org/abs/1506.02640

[5] O. Cicek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, ser. Lecture Notes in Computer Science, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, Eds. Cham: Springer International Publishing, 2016, pp. 424–432.

[6] W. Dai, B. Woo, S. Liu, M. Marques, C. B. Engstrom, P. B. Greer, S. Crozier, J. A. Dowling, and S. S. Chandras, "CAN3D: Fast 3D Medical Image Segmentation via Compact Context Aggregation," *arXiv:2109.05443 [cs, eess]*, Sep. 2021, arXiv: 2109.05443. [Online]. Available: http://arxiv.org/abs/2109.05443

[7] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General Perception with Iterative Attention," *arXiv:2103.03206 [cs, eess]*, Jun. 2021, arXiv: 2103.03206. [Online]. Available: http://arxiv.org/abs/2103.03206

[8] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015, p. 0.

[9] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural Discrete Representation Learning," *arXiv:1711.00937 [cs]*, May 2018, arXiv: 1711.00937. [Online]. Available: http://arxiv.org/abs/1711.00937

[10] A. Razavi, A. v. d. Oord, and O. Vinyals, "Generating Diverse High-Fidelity Images with VQ-VAE-2," *arXiv:1906.00446 [cs, stat]*, Jun. 2019, arXiv: 1906.00446. [Online]. Available: http://arxiv.org/abs/1906.00446

[11] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *arXiv:1812.04948 [cs, stat]*, Mar. 2019, arXiv: 1812.04948. [Online]. Available: http://arxiv.org/abs/1812.04948

[12] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and Improving the Image Quality of StyleGAN," *arXiv:1912.04958 [cs, eess, stat]*, Mar. 2020, arXiv: 1912.04958. [Online]. Available: http://arxiv.org/abs/1912.04958