Assignment 2 - Group: 86

Tutors: (list all your tutors)

Group members: Jiawei Sun (500409987), Jenny Lin (470322974), Jinxuan Yang (SID HERE)

**Introduction** the dataset that you chose, discuss its relevance indiverse applications, and give an overview of the methods you used.

Chess Positions dataset is download from Kaggle website, with 100,000 images generated by custom build tool. The website describes dataset “*a randomly generated chess positions of 5-15 pieces (2 kings and 3-13 pawns/pieces). Images were generated using 28 styles of chess boards and 32 styles of chess pieces totalling 896 board/piece style combinations*” [1]. Labels can be found at file name in Forsyth–Edwards Notation format [2]. In most cases chess positions are legal.

Images are same dimensions of 400 by 400 pixels. No resizing, rotation, or adjustment of regions are required. From observation, 32 styles of chessboard have similar layout and image segmentation is relative straight forward. Each image is partitioning to individual 64 squares location attribute using common chess notation of “*by the letters a to h, from left to right from the white player’s point of view, and the ranks by the numbers 1 to 8, with 1 being closest to the white player”* [3]. Individual squares are either place a piece of chess pieces or only with background. Variants of chess pieces are observed in sizes, colours, and shapes. Classes of piece is defined as: [ ‘q’, ‘k’, ‘b’, ‘n’, ‘r’, ‘p’, ‘Q’,’K’,’B’,’N’,’R’, ‘P’]

In this exercise, we will be using machine learning algorithm to develop classifier to assist identify labels (I.E. FEN format) for any random image data. [JL: we might need second objective (if time allows. As our current goal duplicate with previous 1 &2)

**Previous work** successful techniques utilised on the same or similar datasets and how they are different to yours.

There are several previous works using same datasets. Chess-Fen-generate [5] and Chess-FEN-Generator-Improved [6] use deep neural networks (python library TensorFlow Keras model) to identify image to FEN labels. Improved-Version-Using- PyTorch [7] uses another natural language processing library (PyTorch)

Pervious work around chess game. Python chess library [8] developed serious of codes about board, pieces, and chess movement. Extension of engine stockfish [9] [10] provide analysing and evaluating a position.

One work use python to develop rules for chess movement [11] and while the other develop ‘a chess engine and play with it’ in GUI [12]. Chess-evaluations-starter [13] provide dataset contain FEN and corresponding evaluations.

|  |
| --- |
| List of pervious work – need to sort and move to reference.   * <https://www.kaggle.com/koryakinp/chess-fen-generator> [5] * [https://www.kaggle.com/meditech101/chess-fen-generator-improved [6](https://www.kaggle.com/meditech101/chess-fen-generator-improved%20%5b6)] * [https://www.kaggle.com/ashwinbhatt/improved-version-using-pytorch [7](https://www.kaggle.com/ashwinbhatt/improved-version-using-pytorch%20%5b7)] * <https://python-chess.readthedocs.io/en/v0.15.0/core.html> [8] * [https://github.com/official-stockfish/Stockfish [9](https://github.com/official-stockfish/Stockfish%20%5b9)] * <https://python-chess.readthedocs.io/en/latest/engine.html#analysing-and-evaluating-a-position> [10] * <https://impythonist.wordpress.com/2017/01/01/modeling-a-chessboard-and-mechanics-of-its-pieces-in-python/> [11] * [https://github.com/fsmosca/Python-Easy-Chess-GUI](https://github.com/fsmosca/Python-Easy-Chess-GUI%20%5b11) [12] * [https://www.kaggle.com/ronakbadhe/chess-evaluations-starter](https://www.kaggle.com/ronakbadhe/chess-evaluations-starter%20%5b8) [13] * <https://en.wikipedia.org/wiki/Stockfish_(chess)> [this is reference] * <https://www.chessprogramming.org/Stockfish> [this is reference] * <https://hxim.github.io/Stockfish-Evaluation-Guide/> [this is reference] * <https://pypi.org/project/stockfish/> [this is reference] * <https://github.com/arnabdotorg/Playing-Card-Recognition> [this is reference]   Codes around computer vision of image recognition in cards |

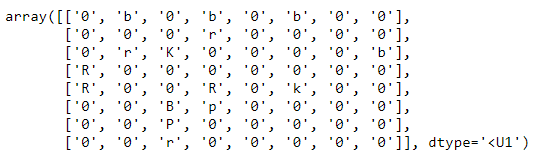
**Methods** Explain the theory behind each of them and discuss your design choices. This part should at least include pre-processing approaches and machine learning techniques used.

**Pre-processing**

Label:

Labels contain 8 smaller sections (separated by a dash ‘-‘). Each section represent piece placed at rank 1 to 8. From Example label “B3K3-3r4-Q1rNN3-4R3-N1P1k1Q1-7P-8-bB6,” first section indicates that at rank 1 (top row of the cheeseboard), there are pieces of white bishop, unoccupied, unoccupied, unoccupied, white knight, unoccupied, unoccupied, and unoccupied). Label is pre-processes into array of [‘B’, ‘0’,’0’,’0’,’K’,’0’,’0’,’0’]. Zero (0) indicate unoccupied squares.

Illustration for image ‘1b1B1b2-2pK2q1-4p1rB-7k-8-8-3B4-3rb3.jpeg’ is converted to length 64 array. The matrix representation is showed as the following:



Note that the array shown above is of shape (8, 8). In our code the input shape is (64,).

There are in total 13 labels in our case (extracted from comments in code):

# pawn = "P", knight = "N", bishop = "B", rook = "R", queen = "Q" and king = "K"

# White pieces are designated using upper-case letters ("PNBRQK") while black pieces use lowercase ("pnbrqk")

# we use 0 to note an empty grid.

Images:

Images firstly are read into python using skimage.io.imread(), then partitioned into 64 squares.

Others pre-processing (Team considers to apply further image derivatives, or PCA, Gaussian blurring)

**Machine learning technique**

Convolutional Neural Network Classifier (CNN)

10-fold Cross Validation:

To do a 10-fold cross validation for our classifiers, we randomized the listing order of the 80k images in the training set, and divided them into 10 separate groups, with each group having 8k images. These groups are later used in our validation stage.

Pre-processing:

After looking into the process of generating the dataset [9], we found that the data is generated by a C# application which uses 28 board images and 32 different themes of chess pieces, meaning that the board images contains identical backgrounds and chess pieces. Any image of the dataset could be split into 64 grids of 50 \* 50 pixels by simple array indexing.

Creating Batches: the easy way, and the hard way.

From the overview of the dataset, we could conclude that, to create batches for our TensorFlow convolutional neural network (hereinafter referred to as CNN), the simplest way would be to split each image into 64 grids, and feed them to the training device (in our case, it is an Nvidia GeForce graphics card). This way, the samples in each individual batch will be guaranteed to own the same theme, thus facilitating our training.

Yet the one-board-after-another way to create batches could slow down the pipeline to a great extent. As we divide the 80k images into 80k batches, given the performance overhead of copying data from CPU to GPU, the training process took about 54 min to finish.

**Experiment** results and comparisons for the implemented algorithms. Include runtime, hardware and software specifications of the computer that you used for performance evaluations. You are then expected to include meaningful comments on the results of your experiments, and reflect on your design choices.

**Conclusion** sum up your results and provide suggestion for meaningful future work.

**Reference**

1. Chess Positions Dataset, Kaggle, assessed 25/10/2020, <https://www.kaggle.com/koryakinp/chess-positions>
2. Forsyth–Edwards Notation, Wikipedia, viewed 25/10/2020, <https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation>
3. Chessboard, Wikipedia, viewed 25/10/2020, <https://en.wikipedia.org/wiki/Chessboard>
4. How To Read "FEN", Chess, Com. Viewed 25/10/2020, <https://www.chess.com/blog/GuessWhoIAm/how-to-read-quotfenquot>
5. Chess FEN Generator, Kaggle, viewed 25/10/2020, <https://www.kaggle.com/koryakinp/chess-fen-generator>
6. Chess FEN Generator Improved, Kaggle, viewed 25/10/2020, <https://www.kaggle.com/meditech101/chess-fen-generator-improved>
7. Solem, Jan Erik. Programming Computer Vision with Python . 1st ed. Sebastopol, CA: O’Reilly, 2012. Print.
8. Howse, Joseph. OpenCV Computer Vision with Python : Learn to Capture Videos, Manipulate Images, and Track Objects with Python Using the OpenCV Library . Birmingham: Packt Publishing, 2013. Print.
9. chess-generator, GitHub project, <https://github.com/koryakinp/chess-generator>
10. TensorFlow (r2.3) API documentation, <https://www.tensorflow.org/api_docs/python/tf>
11. Stockfish, UCI Chess engine, <https://github.com/official-stockfish/Stockfish>
12. CDrill Chess, <https://sites.google.com/view/cdrill/download>
13. Tutorial 5: Cross-Validation on TensorFlow Flowers Dataset, <https://medium.com/fenwicks/tutorial-5-cross-validation-with-tensorflow-flowers-34f7ac36230b>

**Appendix**

**Hyperlinks of the datasets & external open-source libraries**

|  |  |
| --- | --- |
| TensorFlow – machine learning library | https://www.tensorflow.org/install/pip#windows |
| OpenCV – computer vision and machine learning software library | https://opencv.org/ |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Contribution of each group member**

Group members:

Jiawei Sun (Unikey: jsun4242, SID: 5004099987) –33%

Jenny Lin (Unikey: tlin4302, SID: 470322974) – 33%

Jinxuan Yang () – 33%

**Unsorted data for reports**

|  |  |
| --- | --- |
| Some reference | |
| Book: Programming Computer Vision with Python | <https://www.oreilly.com/library/view/programming-computer-vision/9781449341916/>PDF uploaded to google driveA lot of functions for image processingPage 27 – apply PCA of imagesPage 33 - Image derivativesPage 39 -Image de-noisingPage 52 - Scale-Invariant Feature TransformPage 66 -Matching using local descriptorsPage 86 - Registering imagesPage 161- Chapter 6 - clustering method to group images base on similarity or contentPage 215 - Classifying images - hand gesture recognitionPage 226 – SVMPage 228 - Optical Character RecognitionPage 232 - Rectifying images ( this should be useful)Page 237 Image Segmentation + page 251 - normalized cuts algorithm.+ Page 255 Examples image segmentation by minimizing the Chan-Vese model using ROF de-noising. |
| Book: [OpenCV computer vision with Python](https://sydney.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma991020893349705106&context=L&vid=61USYD_INST:sydney&lang=en&search_scope=MyInst_and_CI&adaptor=Local%20Search%20Engine&tab=Everything&query=any,contains,Programming%20Computer%20Vision%20with%20Python&mode=basic) | We can find online version at university library [OpenCV computer vision with Python](https://sydney.primo.exlibrisgroup.com/discovery/fulldisplay?docid=alma991020893349705106&context=L&vid=61USYD_INST:sydney&lang=en&search_scope=MyInst_and_CI&adaptor=Local%20Search%20Engine&tab=Everything&query=any,contains,Programming%20Computer%20Vision%20with%20Python&mode=basic) |
| Deep Reinforcement Learning – example of chessboard | <https://colab.research.google.com/drive/1Xk9MibJ9Fli5tIlDvo88hcZrI76rqZN5>  Board matrix representation |
| Image segmentation | https://au.mathworks.com/discovery/image-segmentation.html |
| Chess strategy | https://en.wikipedia.org/wiki/Chess\_strategy |
| Stockfish (chess) | <https://en.wikipedia.org/wiki/Stockfish_(chess)>  <https://www.chessprogramming.org/Stockfish>  <https://hxim.github.io/Stockfish-Evaluation-Guide/>  <https://pypi.org/project/stockfish/> |
| python-chess: a pure Python chess library | <https://python-chess.readthedocs.io/en/latest/> |
|  |  |