**Sudoku Solver: Winter In Data Science**

**Week 1:**

Intro to Google Colab:

1. [Google Colab Tutorial for Beginners | Get Started with Google Colab](https://www.youtube.com/watch?v=RLYoEyIHL6A&ab_channel=CodewithDogaOzgon)
2. [Google Colab - Connecting to Google Drive!](https://www.youtube.com/watch?v=6UnCrulz-fE&ab_channel=AdrianDolinay)

Python basics:

1. [Python Tutorial - Python Full Course for Beginners](https://www.youtube.com/watch?v=_uQrJ0TkZlc&t=15037s&ab_channel=ProgrammingwithMosh) watch till 3:00:00
2. <https://github.com/Karrthik-Arya/TSS-2021/tree/main/Python%20%26%20its%20Applications/Week-1> Till “Some standard Libraries”
3. <https://github.com/Karrthik-Arya/TSS-2021/tree/main/Python%20%26%20its%20Applications/Week-2> Till “Seaborn”

Note: those who prefer videos can go ahead with the First link + Numpy and Pandas in 3), those who prefer reading can go through 2) and 3)

Basics of OCV:

1. [OpenCV Tutorial: A Guide to Learn OpenCV - PyImageSearch](https://pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/)
2. [Image Filtering Using Convolution in OpenCV | LearnOpenCV #](https://learnopencv.com/image-filtering-using-convolution-in-opencv/)
3. [Image Thresholding in OpenCV](https://learnopencv.com/opencv-threshold-python-cpp/)
4. [Blob Detection Using OpenCV ( Python, C++ ) |](https://learnopencv.com/blob-detection-using-opencv-python-c/)
5. [Edge Detection Using OpenCV | LearnOpenCV #](https://learnopencv.com/edge-detection-using-opencv/)
6. [Contour Detection using OpenCV (Python/C++)](https://learnopencv.com/contour-detection-using-opencv-python-c/)
7. [OpenCV Contour Approximation - PyImageSearch](https://pyimagesearch.com/2021/10/06/opencv-contour-approximation/)

The C/C++ part in the above tutorials is optional as we will be exclusively working with Python during this project.

Hands On task:

The following image contains a bunch of polygons. You have to use techniques learnt above to count the number of polygons with different numbers of sides (basically how many triangles, quadrilaterals etc… are there in the image). You are not supposed to use any DL/ML algorithm here, just simple Open CV and python.

Link to the image: [1670993283171.jpg](https://drive.google.com/file/d/1xxAu8mHql8cw1jlHQ4bm9LjngX7d_KjU/view?usp=sharing)

Hint: You will need to use contour detection and contour approximation learnt in the tutorials.

**Week 2:**

Basics of Neural Networks:

1. [Neural networks - YouTube](https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi)
2. [Neural Networks and Deep Learning (Course 1 of the Deep Learning Specialization) - YouTube](https://www.youtube.com/playlist?list=PLkDaE6sCZn6Ec-XTbcX1uRg2_u4xOEky0)

Basics of CNNs:

1. [Convolutional Neural Networks (Course 4 of the Deep Learning Specialization) - YouTube](https://www.youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF) till video 11 (Week 1). Week 2 is optional you can watch once you finish the compulsory material of this week

Hand-On Tasks:

1. Image classification using Neural networks on MNIST dataset.

Tutorial For implementing Neural Network using Tensorflow: [Build your first Neural Network in TensorFlow 2 | TensorFlow for Hackers (Part I)](https://towardsdatascience.com/building-your-first-neural-network-in-tensorflow-2-tensorflow-for-hackers-part-i-e1e2f1dfe7a0)

1. Image classification using Convolutional Neural networks on MNIST dataset.

Tutorial For implementing CNNs using Tensorflow: [Convolutional Neural Network (CNN) | TensorFlow Core](https://www.tensorflow.org/tutorials/images/cnn)

MNIST Dataset Download and Info Link : [Digit Recognizer | Kaggle](https://www.kaggle.com/competitions/digit-recognizer/data)

Neural networks and CNNs can also be implemented using the PyTorch Library. If you finish with the TensorFlow implementation, you can give this one a shot. Contact us and we will provide you some resources.

**Week 3 & 4:**

1. **Sudoku Puzzle extraction:**

Follow the below-given steps,

1. Find the outermost square of the puzzle. You can use contour detection for this and find the largest square/rectangular contour.
2. Discard the background of the image by using 4 -point transform. This will leave you with the puzzle only.
3. Divide the image into 81 cells; the assumption that all the cells are of equal size can be made.
4. For each of the individual cells, we need to get rid of the grid lines. You can use blob detection or contour detection for this. This process would work since the digit is the largest contour/blob in the cell. There are also inbuilt functions for removing the grid lines. You can also make a fresh copy of the cell by transferring this contour/blob to a blank image. Note that you also need to take note of the cells that do not have any digit in them.

Implementation for the above pipeline: [OpenCV Sudoku Solver and OCR - PyImageSearch](https://pyimagesearch.com/2020/08/10/opencv-sudoku-solver-and-ocr/)

Some other implementations:

i) [Sudoku Solver using Computer Vision and Deep Learning — Part 1 | by Aakash Jhawar](https://aakashjhawar.medium.com/sudoku-solver-using-opencv-and-dl-part-1-490f08701179)

ii) [How to get the cells of a sudoku grid with OpenCV? - Stack Overflow](https://stackoverflow.com/questions/59182827/how-to-get-the-cells-of-a-sudoku-grid-with-opencv)

1. **Digit recognition:**

This part basically needs a CNN. You can use the one which you used for the MNIST task. Now you have to get the predictions of the model on the individual cells we extracted above. You can compare the results and see if they are satisfactory or not. We take the predictions on all the cells containing digits and then create a software representation of the puzzle (A NumPy array would be best).

To solve the puzzle correctly, we need to detect all the digits correctly. The accuracy provided by the model trained on MNIST might not be good enough. You might want to try out deeper CNNs. You can search for some famous CNN architectures online for inspiration. Even with a deeper CNN, it might happen that the model is overfitted on the MNIST data and hence performing poorly on the printed number images we want it to identify. One alternative is to create our own dataset using the Pillows (PIL) library, and train our model on that.

Pillows resources:

i) [Ruining Sudoku — A Data Science project (Part 3: Digits recognition and Sudoku solver) | by Matteo Barbieri](https://towardsdatascience.com/ruining-sudoku-a-data-science-project-part-3-digits-recognition-and-sudoku-solver-5271e6acd81f)

ii) [How to add text on an image using pillow in Python ? - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-add-text-on-an-image-using-pillow-in-python/)

1. **Solving the puzzle:**

This part is simple, and there are multiple algorithms to do this.

Some implementations:

i) [Sudoku | Backtracking-7 - GeeksforGeeks](https://www.geeksforgeeks.org/sudoku-backtracking-7/)

ii) [Coming Back to Old Problems: How I Finally Wrote a Sudoku Solving Algorithm - DEV Community 👩‍💻👨‍💻](https://dev.to/aspittel/how-i-finally-wrote-a-sudoku-solver-177g)