ECE 157B/272B Homework 1:
Wafer Map Failure Pattern Classification

Due date: Sunday, January 16th, 2022, 11:59PM

Introduction

Last quarter, we explored the *traditional machine learning* algorithms, practiced with each type of algorithm by building models for three numerical datasets using Python's Scikit-Learn library, and presented the results using a web-based application.
This quarter, we will continue our machine learning journey and shift our focus to deep learning i.e., deep neural networks and their application. In this first homework, we will get some hands-on experience with the **Convolutional Neural Network (CNN)**, which are commonly used in computer vision tasks, and its implementation using TensorFlow. In this assignment, you will be designing, training and evaluating CNN models for a wafer classification task using free GPU/TPU resources available on Google Cloud.

Google CoLab

Google Colab is a Python development environment that runs in the browser using Google Cloud. It is similar to Jupyter Notebook but with the latest tensorflow package pre-installed. You can learn about CoLab's basic features here. We will be mainly touching three features of Colab in this homework:

- Online python editing environment based on Jupyter.
- Virtual machine with private file system that's associated with your Google account.
- Free GPU and TPU resources.

Wafer Map Failure Pattern Analysis

In semiconductor manufacturing process, batches of functional circuits are fabricated on a slice of silicon material which we call a *wafer*. After going through complicated test stages, the wafer is cut (diced) into pieces, each containing one copy of the circuit. Each of these pieces is called a *die*. In companies like Intel and Qualcomm, one of the important ways to get revenue is to ensure a high *yield* of their manufactured wafers. *Yield* is the fraction of dies on the yielding wafers that are not discarded during the manufacturing process. The *yield per wafer* is quantified as the number of working (good/passing) dies divided by the total number of dies fabricated on that wafer.

Wafer maps provide visual details that locate the bad/failing dies on the wafer. By analyzing the failure pattern on a wafer, experienced engineers can often identify issues in manufacturing and/or test process that caused the failure and take actions accordingly in order to increase quality of production and ultimately to increase revenue.

Problem Formulation

For this homework, we will build and train a convolutional neural network model for classifying the failure types of given wafers using Tensorflow. You can follow the coding template and descriptions in the provided Colab notebook to implement this assignment.

Column	Count	Description
Wafer Map	3300	A list of wafer maps, each is a 64-by-64 Numpy array with data type "float32". Value -1.0 denotes *good/passing*, 1.0 denotes *bad/failing*, 0.0 denotes no die.
Failure Type	3300	A list of labels, each is a string denotes the type of failure pattern from 6 categories: "pass", "edge", "deform", "total_loss", "crack", "nodule"

Table 1: Part One Data Set: Wafers Generated by Min Jian Yang (W21 157B TA)

Column	Count	Description
Lot Name	904	A list of strings, each denoting the manufacturing lot name of a wafer.
Wafer Number	904	A list of integers, each identifying a wafer in its lot.
TSMC Index	904	A list of integers, each indexing a wafer for TSMC internal usage.
Die Count	904	A list of integers, each denoting the number of dies on a wafer.
Wafer Map	904	Same format as in Table.1 but with 2 different shapes: 30-by-34, and 27-by-25
Failure Type	904	Same format as in Table.1 but from 4 categories: "Center", "Edge-Loc", "Loc", "Scratch"

Table 2: Part Two Data Set: Real-world Wafers from TSMC [1]

1 Part one: Classification with a simple data set

We start with a simple data set generated by TAs. The data is stored as a Pandas DataFrame (a table). TAs provided the code for downloading the data set from a shared URLs to your Google Colab file system.

1.1 Data Set

This simple data set contains 3300 wafer maps and their labels, as shown in Table.1. There are 6 types of known labels: *pass*, *total_loss*, *edge*, *nodule*, *crack* and *deform*. The first 300 wafers in the data set are assigned with known labels and we will use these as our training and validation data set. The remaining 3000 wafers are labeled as 'unknown' and we will use our trained CNN model to predict their labels. The accuracy of this prediction is our test accuracy to report on the unknown data. We expect to see a fairly high accuracy (95%) obtained on this first data set. You need to submit your prediction results as a csv file to GradeScope for grading.

1.2 Workflow

The following guiding tasks will help you implement the code and achieve the required accuracy. You are provided with a Colab notebook with incomplete code blocks for you to implement as well as question prompts for you to answer.

- (2 pts) Download the first data set and visualize the failure types by randomly selecting one wafer map from each failure type. Generate a single plot showing one wafer map image for each of the six types.
- (3 pts) Use at least three training and validation split ratio and show the validation prediction accuracy for each split ratio. Explain why the accuracy rises or drops in each configuration. Which ratio did you choose at the end?
- (2 pts) Provide the definitions for the following hyper-parameters.
  1. (1 pt) Number of filters of a CNN layer
  2. (1 pt) Kernel size of a CNN layer

- (2 pts) Why is the provided network using a Softmax activation function in the final dense layer? Can we use a different activation function instead?
- (272B only: 2 pts) We've learned Binary Cross Entropy loss function in class. Why is the provided netowrk using the Categorical Cross Entropy as loss function? How are the two related?
- (8 pts) Make these four modification to the model provided (one at a time) and explain how they affect the accuracy.
    1. (2 pts) Add Dropout layers and explain its effect
    2. (2 pts) Add more CNN layers and explain its effect
    3. (2 pts) Add L2 regularization to CNN layers and explain its effect
    4. (2 pts) Add L2 regularization to Dense layers and explain its effect
- (6 pts) Use image augmentation to generate 100 more images for each failing class (total of five, excluding the 'pass') and retrain your model with the added images. Did the performance of the retrained model improve or decrease? Image augmentation could be rotation, horizontal/vertical flip, etc.
- (2 pts) Show the confusion matrix for the validation predictions (include the axis values).
- (2 pts) Use the model with the best validation accuracy to predict labels for the unknown wafers. Submit your prediction csv file to GradeScope to see your accuracy. Are you able to achieve 95% accuracy on the unknown dataset?

*NOTE: you must build your model layer-by-layer. You are not allowed to call built-in models in Tensorflow (e.g. Xception, VGG16, ResNet...)

2 Part two: Classification with the TSMC data set

At this point, you should already have a good understanding of a CNN model. We will use the architecture you have configured from part one to train and evaluate on a real-world wafer data set from TSMC [1].

2.1 Data Set

The second data set contains 904 labeled wafer maps as shown in Table.2. There are 4 types of labels: *Center*, *Edge*, *Edge-Loc*, and *Scratch*.

2.2 Workflow

- (2 pts) Download the second data set and visualize the class types as you did in part one.
- (8 pts) Prepare data, use the same model architecture as in part one to train and evaluate this TSMC data set, and show the confusion matrix for the validation predictions. Are the accuracy results the same across all the categories?
- (4 pts) Were you able to get a good accuracy ($> 95\%$) on this TSMC data set? If not, explain why the same model performed worse?
- (272B only: 10 pts) Study the paper where the TSMC data set was first proposed [2]. Drawing inspirations from this paper, provide possible solutions to improve the classification accuracy of this data set. (Even better if you come up with your original ideas.)

What to Turn In

Save your Google Colab notebook as ***.ipybn*** file and submit it to Gradescope: https://www.gradescope.com/courses/351582. Course entry code: 5V3YRX

The Colab notebook should contain the following parts for grading:

- Completed and functional code cells. (10 pts as specified in Colab)
- Answers to all the questions and diagrams/plots listed in the **Problem Formulation**. (41 pts for 157B, 53 pts for 272B, as specified above)

Note: Do not close the cell outputs after running the code cells, so that all logging and plots will be saved in the notebook for grading.

References

[1] Ming-Ju Wu, Jyh-Shing R. Jang, and Jui-Long Chen. *MIR-WM811K*. http://mirlab.org/dataSet/public/. 2015.

[2] Ming-Ju Wu, Jyh-Shing R. Jang, and Jui-Long Chen. "Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets". In: *IEEE Transactions on Semiconductor Manufacturing* 28.1 (2015), pp. 1–12. DOI: 10.1109/TSM.2014.2364237.