ECE 157B/272B Homework 2: Defect Classification [1]

Due Date: Sunday, February 9th, 2020, 11:59PM

Introduction

In homework 1, you were able to achieve over 80% accuracy based on cross-validation on the training set, simply by calling the built-in modules from the Python scikit-learn library [1]. In fact, feature values in the pokemon data set follow several correlated Gaussian distributions. However, it is more difficult for the machine learning algorithms to capture the underlying data distribution in more complicated applications. In this assignment, you are given a larger data set from such an application, and your job, similar to homework 1, is to wisely choose or design a machine learning model to get the best classification accuracy.

The story behind this homework is as follows:
You are the machine learning expert in your company's team. You came to work one day, and your boss told you that the company is verifying a new product line. But, the verification process is very slow due to the use of visual inspection. The verification process requires engineers to go through thousands of wafer images and identify ones that fall into a certain defect class. He showed you a sample of each defect class below. The circular object is a wafer, the green pixels represent passing parts and the yellow pixels represent failing parts.

Knowing that you were able to achieve high accuracy in your last project, your boss wants you to use machine learning to speed up the verification process. Your job is to build a neural network model that can correctly classify the wafers into different classes.

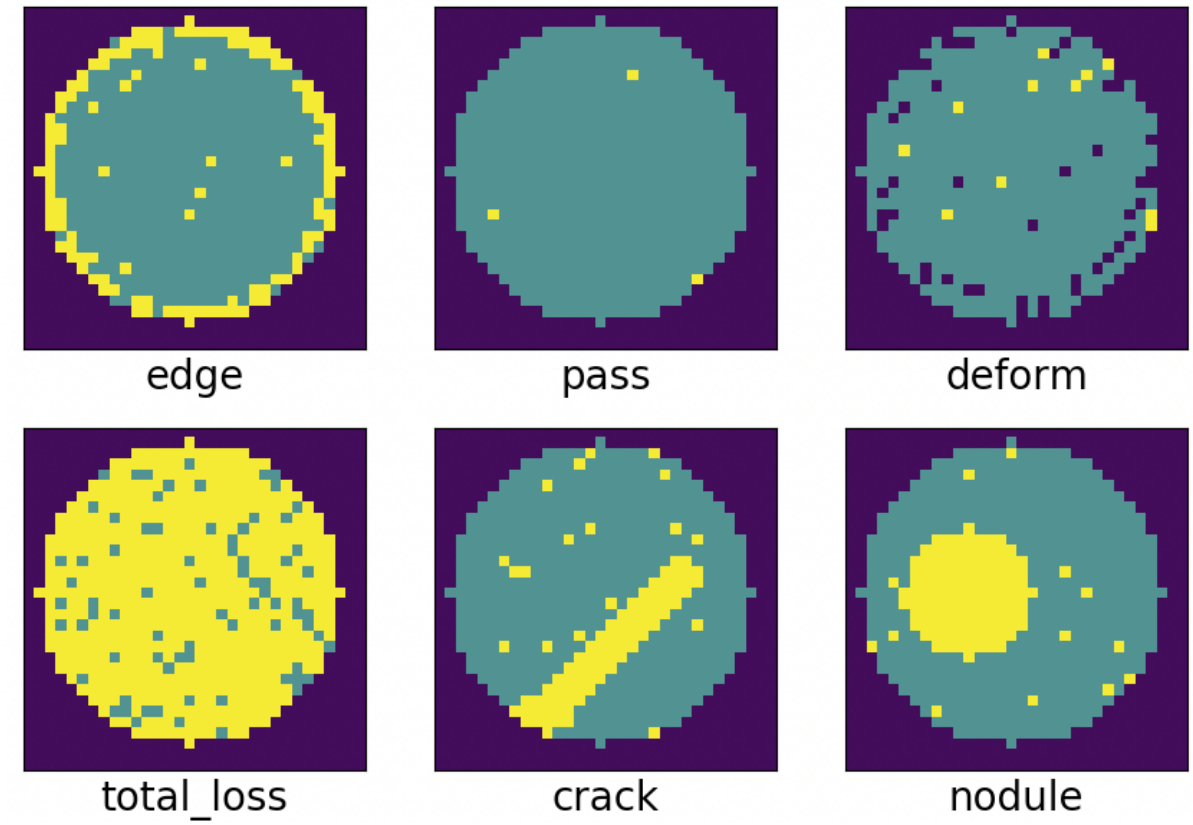

Figure 1: Figure 1: Wafer Class Samples

[1] Designed by Jenny Zeng. Modified by Min Jian Yang

Dataset

*** Make sure to download the latest version of the data sets on Piazza**

Download the training data set: train_data.npy. The training data is stored in an 400-by-64-by-64 matrix as numpy array. Labels (train_label.npy) for the training data are stored in another numpy array with 400 string values. We provide you sufficient samples, but you can decide the exact amount of training data to use. There are 6 types of labels: pass, total_loss, edge, nodule, crack and deform. Download the unlabeled test set: test_data.npy. It contains 400 test samples with dimensions 64-by-64.

Workflow

The workflow is the same as the previous homework, we suggest that you

1. Load data
2. Preprocess data so they can be fed into the Machine Learning Algorithm
3. Split data into training data and validation data
4. Build and train a neural network
5. Obtain performance measurement (repeat 4-5 until satisfied)
6. Predict test data's label using the trained model

Grading Criteria

For this assignment, we publish a CodaLab competition (here), where you can submit your code and get evaluation scores that will be listed on a leaderboard. The score is determined by how many correct labels you get on the test set. If you are not familiar with CodaLab Competitions, check out here: participating in a competition. Register a CodaLab Competitions account and log into CodaLab Competitions. After your submission finishes running, please choose to submit it to the leader-board. The top five students who achieve the highest score on the test set will get bonus points.

For this homework, you are tasked to build a neural network model for the image classification task using Keras [2], and it must be able to achieve at least an 90% accuracy on the test dataset. The test dataset's distribution is very similar to the training data. Therefore, if you split the training data and validation data wisely and obtain an 90% accuracy on your validation dataset, you should be able to obtain an 90% accuracy on the test dataset. You need to run the provided test set (test_data.npy) through your neural network, and save your prediction results with numpy.save() with the filename 'prediction.npy'.

You should explain the following in the report:

- Overview of the project objective
- Explain the workflow in detail
- A diagram of your neural network. It should show the number of layers in the neural network, the layer type (e.g. Fully-Connected, Convolution, Pooling...), and the parameters in each layer
- Explain why you choose this neural network architecture? For example, are there certain properties that allow it to achieve high accuracy? Are there certain properties that make it suitable for image classification tasks?
- Did you use any training tricks (e.g. normalization, dropout, regularization, fine-tuning...) to enhance the performance of your neural network? If so, give a brief explanation of the trick and how much did it increase your model's accuracy.

- How did you split the data set into training and validation set? Why you made that choice?
- Show your experiment results using a confusion matrix and a graph of accuracy vs. epoch. What do the confusion matrix and the graph tell you in term of the model's performance? Overfitting, underfitting, accuracy, precision, recall?

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

What to Turn In

Please include the following items in a **single zip** file.

- All Python scripts. Please comment in the script.
- Classification results (as npy file) for the test set with the best model.
- A detailed report.
- In a paragraph, share your thoughts and leave questions if any.

Name your folder as **ECE157B_HW2_LastName_FirstName** or **ECE272B_HW2_LastName_FirstName**. Then submit a single attachment to: minjianyang@ucsb.edu

References

[1] Supervised learning classifiers. https://scikit-learn.org/stable/supervised$_{l}earning.htmlsupervised-learning$

[2] Keras Image Classification Tutorial. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

[3] Keras MNIST CNN Tutorial. https://keras.io/examples/mnist_cnn/

[4] Python numpy package. http://www.numpy.org/

Good Luck!
Min Jian Yang