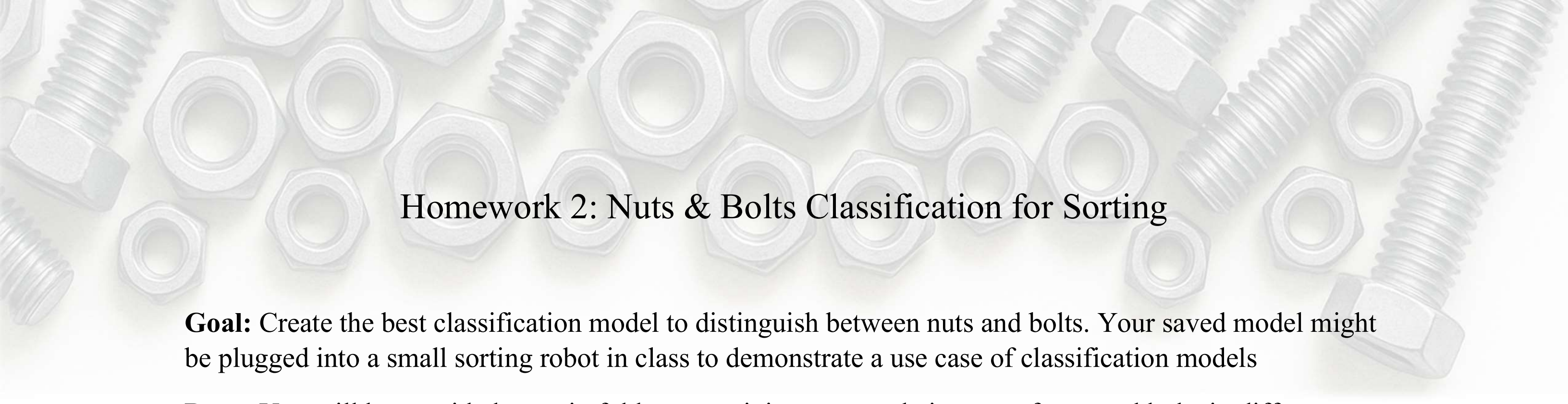

Homework 2: Nuts & Bolts Classification for Sorting

Goal: Create the best classification model to distinguish between nuts and bolts. Your saved model might be plugged into a small sorting robot in class to demonstrate a use case of classification models

Data: You will be provided two zip folders containing grey scale images of nuts and bolts in different orientations. You are welcome and encouraged to add data augmentation for training data to improve the generalizability of your model.

Model Inputs: The model inputs will be the images of either a nut or bolt. You must make you input images of size (300,300,1) so that it will be compatible with the sorting robot.

Model Outputs: Your model should output one number between 0 and 1. Any numbers less than 0.5 will be labeled as a bolt while numbers greater than 0.5 will be labeled as a nut. The closer a number is to 0.5 the less confident your model is in its prediction.

Assignment:

1. Split data into training, testing, and validation. One way of doing this is through tensorflow's ImageDataGenerator or Split Folders. You can also add data augmentation with the ImageDataGenerator.
2. Using google collab (or TensorFlow version 2.18): construct and train a CNN model for binary classification of the nuts and bolts. Your model should take in images of size 300 by 300 by 1. The output layer should consist of 1 neuron with a sigmoid activation function.
3. Performance Metrics
    a. Report training curves (training/validation loss vs. epoch and training/validation accuracy vs epoch)
    b. Training time for the model (time/epoch, time til best model)
    c. Test model with reserved test data and report confusion matrix, accuracy, precision, recall, F1 score, the receiver operating characteristic (ROC), and area under curve (AUC)
4. Implementation
    a. Save your model using the command (replace last name with your last name):

```
model.save('./nuts_and_bolts_model_lastname.keras')
```

    b. A small sorting robot has been constructed that is designed to use these models. It consists of an Arduino, two servo motors, and a camera. It works by uploading a classification model. Then a nut or bolt will be placed on it. The camera will snap a picture, use the classification model to determine what it is and drop it in the corresponding bucket. That's why the TensorFlow version, input shape, and output shape are so important-to ensure compatibility.