PRITHVI AI HACKATHON SUBMISSION

TEAM MEMBER:

1. UMANG SONIKA (B19CSE093)
2. YADAGIRI NAGA SAI VAMSHI VARDHAN (B19EE090)
3. UPADHYAY AKARSH (B19EE085)

PROBLEM STATEMENT:

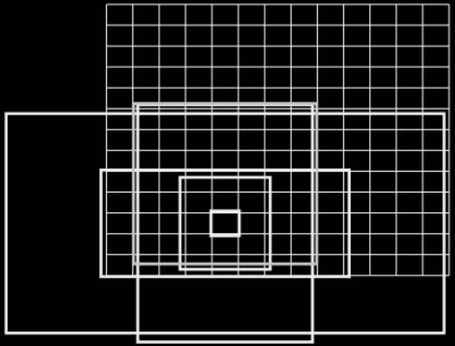
1. Given an image, determine if the image contains defect or not, and if the image contains defect, output the bounding box coordinates of the defect.

2. The image can contain multiple bounding boxes (i.e it can can contain multiple defects)

We had divided the problem into two parts:

a. Classifying the image, if it contains defect or not

b. Predict the bounding box of the images that are predicted to be defective



CLASSIFYING THE IMAGE TO BE DEFECTIVE OR NON DEFECTIVE

a. For the prediction of the image to be defective or non defective, we took the approach of transfer learning methods.

b. We trained the following models on the images and then, we ran the test images through all of these three models, and took the majority out of them

c. The models which we chose to be in transfer learning is:
   i. ResNet50

d. Images Augmentations: We did not do any image augmentation

TRAINING THE MODEL:

While training the model, we took the following hyperparameters:

a. Optimizer: Adam Optimizer
b. Learning Rate: 0.0001
c. Batch Size
d. Loss: Binary Cross-entropy
e. Input Dimension: (200,820)

PREDICT THE BOUNDING BOX OF THE IMAGES THAT ARE PREDICTED TO BE DEFECTIVE

Approach:

a. We thought of using DEtection TRansformer (DETR) for the purpose of predicting the bounding box (since it can predict any number of bounding boxes), however, we had to drop that idea, because of the hardware constraints.

b. So, we tried to use ResNet 50, and that also had the same problem.

c. And as a last try, we used ResNet 18, and low data, and that worked for us.

d. The following were the Hyperparameter:

   i. BATCH_SIZE = 4;
   ii. LEARNING RATE = 5e-5
   iii. Optimizer: Adam
   iv. EPOCHS: 10
   v. LOSS: Mean Squared Error

e. Image Augmentation: We did not use image augmentation, since it would led to out of GPU memory.

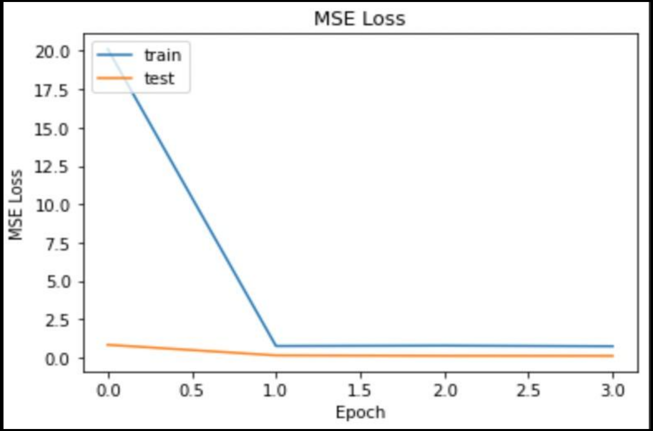
TRAINING:

As mentioned in the previous slide, after having set the hyperparameters, we had used two methods to train the model:

1. Saving the model which had less training loss, after each epoch
2. Saving the model, which had less validation loss, after each epoch

Unfortunately, due to some issues, we were not able to save the results of the first method. However, we were able to save the training progress of the 2nd method. The loss have been described in the below image.

3. Only those losses have been saved which were corresponding to the condition when the model was saved.


MSE Loss
train
test
MSE Loss
Epoch