

Project Development Phase

Model Performance Test

Team Members	POOVENTHAN D PRANESH S SABAREESHWARAN S SANDEEP P K
TEAM ID	PNT2022TMID04681
Project Name	Digital Naturalist - AI Enabled tool for Biodiversity Researchers
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in the model performance testing template.

S. N o.	Parameter	Values	Screenshot
1	Model Summary	Total params: 22,109,990 Trainable params: 307,206 Non-trainable params: 21,802,784	Screenshot 1
2	Accuracy	Training Accuracy - 92.8% Validation Accuracy - 85.6%	Screenshot 2

Screenshots - Please refer to the next page:

Screenshot 1 :

The screenshot shows a Jupyter Notebook interface. The top part of the notebook contains a table with 10 rows of data. The columns are: a numerical value, a categorical variable, a numerical value, a categorical variable, a numerical value, a categorical variable, a numerical value, a categorical variable, a numerical value, and a categorical variable. The data is as follows:

Value	Category	Value	Category	Value	Category	Value	Category	Value	Category
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100

Below the table, there is a loading message: "loading the dataset".

Screenshot 2:

```
# Model Training

# 1. Creating a model checkpoint about accuracy of the model and saving the checkpoint
def create_checkpoint(model, accuracy):
    # Save the model
    torch.save(model.state_dict(), 'model.pth')
    # Save the accuracy
    torch.save(accuracy, 'accuracy.pth')

# 2. Training a performance object about how training over the model performance about depending on a loss on validation about
def train_model(model, accuracy):
    # Save the model
    torch.save(model.state_dict(), 'model.pth')
    # Save the accuracy
    torch.save(accuracy, 'accuracy.pth')

# 3. Evaluating the model
def evaluate_model(model, accuracy):
    # Save the model
    torch.save(model.state_dict(), 'model.pth')
    # Save the accuracy
    torch.save(accuracy, 'accuracy.pth')

# 4. Exporting the model to save
def export_model(model, accuracy):
    # Save the model
    torch.save(model.state_dict(), 'model.pth')
    # Save the accuracy
    torch.save(accuracy, 'accuracy.pth')

# 5. Exporting the model weights
def export_weights(model, accuracy):
    # Save the model
    torch.save(model.state_dict(), 'model.pth')
    # Save the accuracy
    torch.save(accuracy, 'accuracy.pth')

# 6. Testing the model
```