

## 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	<b>Total params: 22,109,990</b> <b>Trainable params: 307,206</b> <b>Non-trainable params: 21,802,784</b>	Screenshot 1
2	Accuracy	<b>Training Accuracy - 92.8%</b> <b>Validation Accuracy - 85.6%</b>	Screenshot 2

Screenshots - Please refer to the next page:

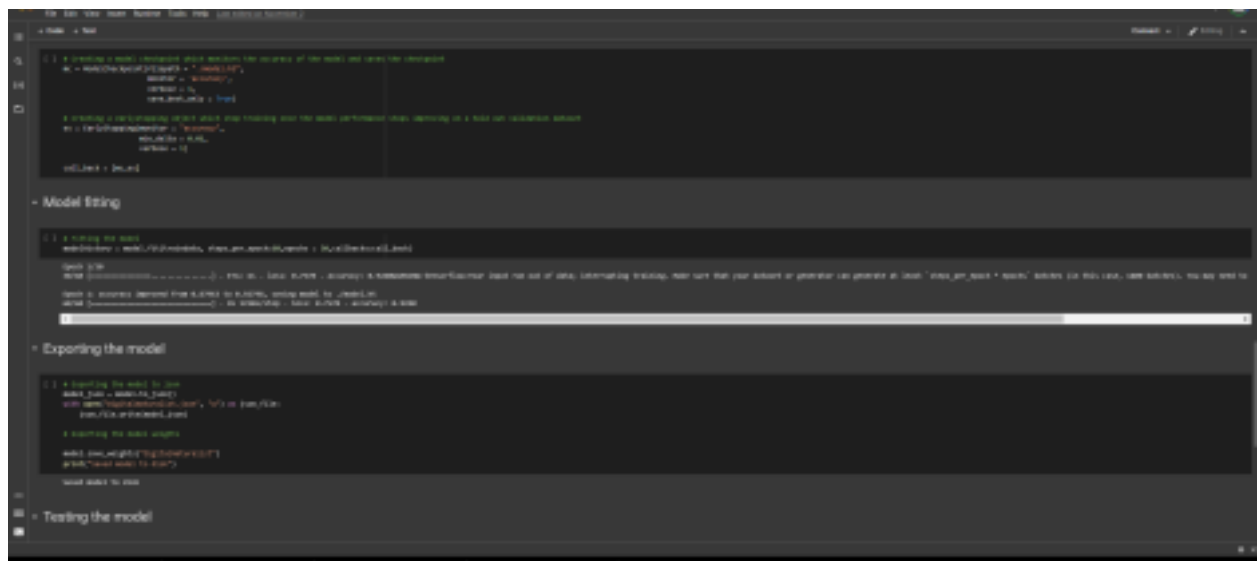
Screenshot 1 :



The screenshot shows a Jupyter Notebook interface with a table of model parameters. The table has columns for parameter names, their values, and the function used to generate them. The parameters include various weights, biases, and learning rates, all initialized with random values or specific functions like 'randn' or 'rand'. The table is titled 'Model Parameters' and has a total of 10 parameters listed.

Parameter	Value	Function
W1	0.0001	randn(1,1)
b1	0.0001	randn(1,1)
W2	0.0001	randn(1,1)
b2	0.0001	randn(1,1)
W3	0.0001	randn(1,1)
b3	0.0001	randn(1,1)
W4	0.0001	randn(1,1)
b4	0.0001	randn(1,1)
W5	0.0001	randn(1,1)
b5	0.0001	randn(1,1)
W6	0.0001	randn(1,1)
b6	0.0001	randn(1,1)
W7	0.0001	randn(1,1)
b7	0.0001	randn(1,1)
W8	0.0001	randn(1,1)
b8	0.0001	randn(1,1)
W9	0.0001	randn(1,1)
b9	0.0001	randn(1,1)
W10	0.0001	randn(1,1)
b10	0.0001	randn(1,1)
W11	0.0001	randn(1,1)
b11	0.0001	randn(1,1)
W12	0.0001	randn(1,1)
b12	0.0001	randn(1,1)
W13	0.0001	randn(1,1)
b13	0.0001	randn(1,1)
W14	0.0001	randn(1,1)
b14	0.0001	randn(1,1)
W15	0.0001	randn(1,1)
b15	0.0001	randn(1,1)
W16	0.0001	randn(1,1)
b16	0.0001	randn(1,1)
W17	0.0001	randn(1,1)
b17	0.0001	randn(1,1)
W18	0.0001	randn(1,1)
b18	0.0001	randn(1,1)
W19	0.0001	randn(1,1)
b19	0.0001	randn(1,1)
W20	0.0001	randn(1,1)
b20	0.0001	randn(1,1)
W21	0.0001	randn(1,1)
b21	0.0001	randn(1,1)
W22	0.0001	randn(1,1)
b22	0.0001	randn(1,1)
W23	0.0001	randn(1,1)
b23	0.0001	randn(1,1)
W24	0.0001	randn(1,1)
b24	0.0001	randn(1,1)
W25	0.0001	randn(1,1)
b25	0.0001	randn(1,1)
W26	0.0001	randn(1,1)
b26	0.0001	randn(1,1)
W27	0.0001	randn(1,1)
b27	0.0001	randn(1,1)
W28	0.0001	randn(1,1)
b28	0.0001	randn(1,1)
W29	0.0001	randn(1,1)
b29	0.0001	randn(1,1)
W30	0.0001	randn(1,1)
b30	0.0001	randn(1,1)
W31	0.0001	randn(1,1)
b31	0.0001	randn(1,1)
W32	0.0001	randn(1,1)
b32	0.0001	randn(1,1)
W33	0.0001	randn(1,1)
b33	0.0001	randn(1,1)
W34	0.0001	randn(1,1)
b34	0.0001	randn(1,1)
W35	0.0001	randn(1,1)
b35	0.0001	randn(1,1)
W36	0.0001	randn(1,1)
b36	0.0001	randn(1,1)
W37	0.0001	randn(1,1)
b37	0.0001	randn(1,1)
W38	0.0001	randn(1,1)
b38	0.0001	randn(1,1)
W39	0.0001	randn(1,1)
b39	0.0001	randn(1,1)
W40	0.0001	randn(1,1)
b40	0.0001	randn(1,1)
W41	0.0001	randn(1,1)
b41	0.0001	randn(1,1)
W42	0.0001	randn(1,1)
b42	0.0001	randn(1,1)
W43	0.0001	randn(1,1)
b43	0.0001	randn(1,1)
W44	0.0001	randn(1,1)
b44	0.0001	randn(1,1)
W45	0.0001	randn(1,1)
b45	0.0001	randn(1,1)
W46	0.0001	randn(1,1)
b46	0.0001	randn(1,1)
W47	0.0001	randn(1,1)
b47	0.0001	randn(1,1)
W48	0.0001	randn(1,1)
b48	0.0001	randn(1,1)
W49	0.0001	randn(1,1)
b49	0.0001	randn(1,1)
W50	0.0001	randn(1,1)
b50	0.0001	randn(1,1)
W51	0.0001	randn(1,1)
b51	0.0001	randn(1,1)
W52	0.0001	randn(1,1)
b52	0.0001	randn(1,1)
W53	0.0001	randn(1,1)
b53	0.0001	randn(1,1)
W54	0.0001	randn(1,1)
b54	0.0001	randn(1,1)
W55	0.0001	randn(1,1)
b55	0.0001	randn(1,1)
W56	0.0001	randn(1,1)
b56	0.0001	randn(1,1)
W57	0.0001	randn(1,1)
b57	0.0001	randn(1,1)
W58	0.0001	randn(1,1)
b58	0.0001	randn(1,1)
W59	0.0001	randn(1,1)
b59	0.0001	randn(1,1)
W60	0.0001	randn(1,1)
b60	0.0001	randn(1,1)
W61	0.0001	randn(1,1)
b61	0.0001	randn(1,1)
W62	0.0001	randn(1,1)
b62	0.0001	randn(1,1)
W63	0.0001	randn(1,1)
b63	0.0001	randn(1,1)
W64	0.0001	randn(1,1)
b64	0.0001	randn(1,1)
W65	0.0001	randn(1,1)
b65	0.0001	randn(1,1)
W66	0.0001	randn(1,1)
b66	0.0001	randn(1,1)
W67	0.0001	randn(1,1)
b67	0.0001	randn(1,1)
W68	0.0001	randn(1,1)
b68	0.0001	randn(1,1)
W69	0.0001	randn(1,1)
b69	0.0001	randn(1,1)
W70	0.0001	randn(1,1)
b70	0.0001	randn(1,1)
W71	0.0001	randn(1,1)
b71	0.0001	randn(1,1)
W72	0.0001	randn(1,1)
b72	0.0001	randn(1,1)
W73	0.0001	randn(1,1)
b73	0.0001	randn(1,1)
W74	0.0001	randn(1,1)
b74	0.0001	randn(1,1)
W75	0.0001	randn(1,1)
b75	0.0001	randn(1,1)
W76	0.0001	randn(1,1)
b76	0.0001	randn(1,1)
W77	0.0001	randn(1,1)
b77	0.0001	randn(1,1)
W78	0.0001	randn(1,1)
b78	0.0001	randn(1,1)
W79	0.0001	randn(1,1)
b79	0.0001	randn(1,1)
W80	0.0001	randn(1,1)
b80	0.0001	randn(1,1)
W81	0.0001	randn(1,1)
b81	0.0001	randn(1,1)
W82	0.0001	randn(1,1)
b82	0.0001	randn(1,1)
W83	0.0001	randn(1,1)
b83	0.0001	randn(1,1)
W84	0.0001	randn(1,1)
b84	0.0001	randn(1,1)
W85	0.0001	randn(1,1)
b85	0.0001	randn(1,1)
W86	0.0001	randn(1,1)
b86	0.0001	randn(1,1)
W87	0.0001	randn(1,1)
b87	0.0001	randn(1,1)
W88	0.0001	randn(1,1)
b88	0.0001	randn(1,1)
W89	0.0001	randn(1,1)
b89	0.0001	randn(1,1)
W90	0.0001	randn(1,1)
b90	0.0001	randn(1,1)
W91	0.0001	randn(1,1)
b91	0.0001	randn(1,1)
W92	0.0001	randn(1,1)
b92	0.0001	randn(1,1)
W93	0.0001	randn(1,1)
b93	0.0001	randn(1,1)
W94	0.0001	randn(1,1)
b94	0.0001	randn(1,1)
W95	0.0001	randn(1,1)
b95	0.0001	randn(1,1)
W96	0.0001	randn(1,1)
b96	0.0001	randn(1,1)
W97	0.0001	randn(1,1)
b97	0.0001	randn(1,1)
W98	0.0001	randn(1,1)
b98	0.0001	randn(1,1)
W99	0.0001	randn(1,1)
b99	0.0001	randn(1,1)
W100	0.0001	randn(1,1)
b100	0.0001	randn(1,1)

Screenshot 2:



The screenshot shows a Jupyter Notebook interface with code for model training and evaluation. The code is organized into sections: 'Model Training', 'Exporting the model', and 'Testing the model'. The 'Model Training' section includes code for defining the model architecture, training the model, and evaluating its performance. The 'Exporting the model' section includes code for saving the model weights and biases. The 'Testing the model' section includes code for loading the model and testing it on new data. The code is written in Python and uses the PyTorch library for deep learning.

```
# Model Training
def train_model(model, data_loader, optimizer, device):
    # Training loop
    for epoch in range(10):
        # Training phase
        model.train()
        for batch_idx, (data, target) in enumerate(data_loader):
            data, target = data.to(device), target.to(device)
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, target)
            loss.backward()
            optimizer.step()
        # Validation phase
        model.eval()
        val_loss = 0
        for batch_idx, (data, target) in enumerate(val_loader):
            data, target = data.to(device), target.to(device)
            output = model(data)
            loss = criterion(output, target)
            val_loss += loss.item()
        val_loss /= len(val_loader)
        print('Epoch: {} \t Training Loss: {} \t Validation Loss: {}'.format(epoch, loss, val_loss))
    return model

# Exporting the model
def export_model(model, path):
    # Save model weights and biases
    torch.save(model.state_dict(), path)

# Testing the model
def test_model(model, data_loader, device):
    # Test loop
    model.eval()
    for batch_idx, (data, target) in enumerate(data_loader):
        data, target = data.to(device), target.to(device)
        output = model(data)
        loss = criterion(output, target)
        print('Test Loss: {}'.format(loss.item()))
```