Model Development Phase Template

Date	25 October 2024
Team ID	739949
Project Title	Bird Species Classification
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training for your bird species classification project involves preparing the dataset using Image Data Generator for data augmentation and preprocessing. The CNN model architecture consists of multiple convolutional layers to extract features from images, followed by max-pooling layers to reduce the spatial dimensions Plots of training and validation accuracy, as well as training and validation loss, are created to visualize how well the model learns over time. The model is then saved for future use. You can customize various aspects like the number of epochs, batch size, and model architecture to better suit your dataset and task requirements.

Initial Model Training Code (5 marks):

```
#model torch.hub.load('facebookresearch/deit:main', 'resmlp 12, pretrained-True)
torch.backends.cudnn.benchmark = True
model= torchvision.models.efficientnet_b0(pretrained=True)
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments oth
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/efficientnet_b0_rwightman-7f5810bc.pth" to /root/.
100%|██████████| 20.5M/20.5M [00:00<00:00, 66.9MB/s]
```

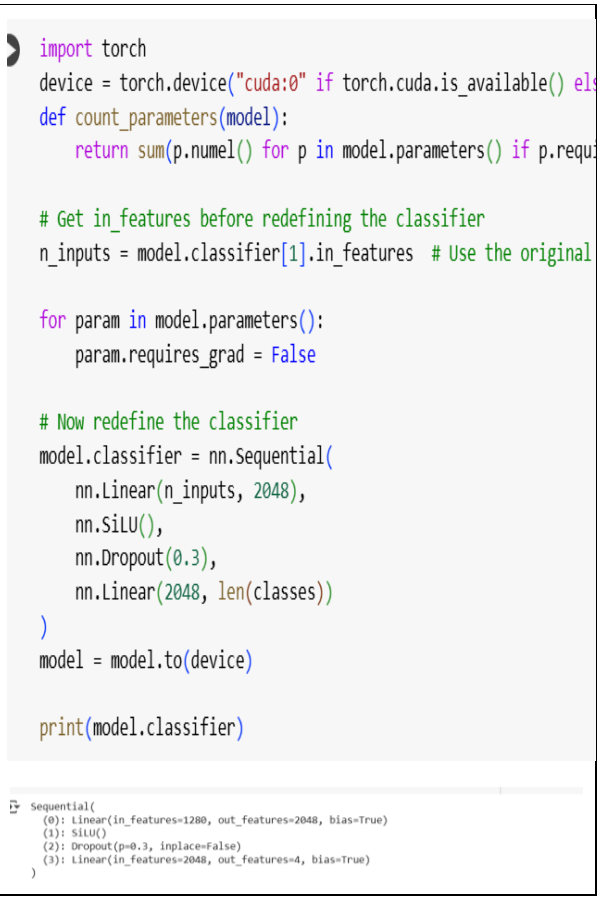
```
import os

# Path to the train and validation directories
train_dir = '/content/drive/MyDrive/Bird Species Classification/manasa/train'
val_dir = '/content/drive/MyDrive/Bird Species Classification/manasa/val'

# Check the structure
print("Training Classes:", os.listdir(train_dir))
print("Validation Classes:", os.listdir(val_dir))
```

```
Training Classes: ['002.Laysan_Albatross', '001.Black_footed_Albatross', '003.Sooty_Albatross', '004.Gr
Validation Classes: ['002.Laysan_Albatross', '001.Black_footed_Albatross', '003.Sooty_Albatross', '004.
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics

CNN	A Convolutional Neural Network (CNN) is a deep learning model designed for analyzing visual data, particularly images. It consists of several layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply filters (kernels) to input images to extract important features, such as edges and textures. Pooling layers reduce the spatial dimensions of the feature maps, helping to reduce computational load and prevent overfitting. After convolution and pooling, the data is flattened into a 1D vector, which is then passed through fully connected layers to make predictions or classifications.	`history = model.fit(`
		`  train_generator,`
		`  validation_data=val_generator,`
		`  epochs=10  # Adjust the number of epochs as needed`
		`)`
		
		Epoch 1/10
		/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/
		self._warn_if_super_not_called()
		112/112 ━━━━━━━━━━━━━━━━━━━━ **83s** 632ms/step - accuracy: 0.0166 - loss: 5.
		Epoch 2/10
		112/112 ━━━━━━━━━━━━━━━━━━━━ **69s** 585ms/step - accuracy: 0.0130 - loss: 4.
		Epoch 3/10
		112/112 ━━━━━━━━━━━━━━━━━━━━ **73s** 619ms/step - accuracy: 0.0160 - loss: 4.
		Epoch 4/10
		112/112 ━━━━━━━━━━━━━━━━━━━━ **68s** 579ms/step - accuracy: 0.0275 - loss: 4.
		Epoch 5/10
		112/112 ━━━━━━━━━━━━━━━━━━━━ **81s** 570ms/step - accuracy: 0.0286 - loss: 4.
		Epoch 6/10
		112/112 ━━━━━━━━━━━━━━━━━━━━ **69s** 574ms/step - accuracy: 0.0332 - loss: 4.
		Epoch 7/10
		112/112 ━━━━━━━━━━━━━━━━━━━━ **68s** 573ms/step - accuracy: 0.0379 - loss: 4.
		Epoch 8/10
		
		`fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (19,7))`
		`sns.set_style("darkgrid")`
		
		`ax[0].plot(acc, '*-',label = 'Training accuracy')`
		`ax[0].plot(val_acc, '*-',label = 'Validation accuracy')`
		`ax[0].set_xlabel('Epochs')`
		`ax[0].set_ylabel('Accuracy')`
		`ax[0].set_title('Epochs & Training Accuracy', fontsize = 17)`
		`ax[0].legend(loc='best')`
		
		`ax[1].plot(loss, '*-',label = 'Training loss')`
		`ax[1].plot(val_loss, '*-',label = 'Validation loss')`
		`ax[1].set_xlabel('Epochs')`
		`ax[1].set_ylabel('loss')`
		`ax[1].set_title('Epochs & loss', fontsize = 17)`
		`ax[1].legend(loc='best')`
		`sns.set_style("darkgrid")`

	CNNs are highly effective for image recognition tasks because they automatically learn spatial hierarchies of features, making them robust in handling complex image data. The architecture can be customized based on the complexity of the problem, and CNNs are often trained using large datasets to achieve high accuracy.	

```
import torch
device = torch.device("cuda:0" if torch.cuda.is_available() els
def count_parameters(model):
    return sum(p.numel() for p in model.parameters() if p.requi

# Get in_features before redefining the classifier
n_inputs = model.classifier[1].in_features  # Use the original

for param in model.parameters():
    param.requires_grad = False

# Now redefine the classifier
model.classifier = nn.Sequential(
    nn.Linear(n_inputs, 2048),
    nn.SiLU(),
    nn.Dropout(0.3),
    nn.Linear(2048, len(classes))
)
model = model.to(device)

print(model.classifier)
```

```
Sequential(
  (0): Linear(in_features=1280, out_features=2048, bias=True)
  (1): SiLU()
  (2): Dropout(p=0.3, inplace=False)
  (3): Linear(in_features=2048, out_features=4, bias=True)
)
```