Model Development Phase Template

Date	28 November 2024
Team ID	739996
Project Title	Deep Fruit Veg: Automated Fruit And Veg Identification
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

Paste the screenshot of the model training code

```
class LR_ASK(keras.callbacks.Callback):
  def __init__(self, model, epochs, ask_epoch):
    super(LR_ASK, self).__init__()
    self.set_model(model)
    self.ask = ask_epoch
    self.epochs = epochs
    self.ask = True
    self.lowest_vloss = np.inf
    self.best_weights = self.model.get_weights()
    self.best_epoch = 1
    self.plist = []

    def get_list(self):
      return(self.plist)

    def on_train_begin(self,logs=None):
      if self.ask_epoch==0:
        print('you set ask_epoch=0, ask epoch will be set to 1',flush=True)
        self.ask_epoch=1
      if self.ask_epoch >=self.epochs:
        print('ask_epoch>=epochs, will train for',epochs,'epochs',flush=True)
      if self.epochs==1:
        self.ask=False
      else:
        print('Training will proceed until epoch',ask_epoch,'then you will be asked to')
        print('enter H to halt training or enter an integer for how many more epochs to run then be asked again')
        self.start_time.time()
```

```
def on_train_end(self,logs=None):
  print('loading model with weights from epoch',self.best_epoch)
  self.model.set_weights(self.best_weights)
  tr_duration=time.time()-self.start_time
  hours=tr_duration//3600
  minutes=(tr_duration-(hours*3600))//60
  seconds=tr_duration-((hours*3600)+(minutes*60))
  msg=f'training elapsed time was{str(hours)} hours,{minutes:4.1f} minutes,{seconds:4.2f} seconds'
  print(msg,flush=True)
  def on_epoch_end(self,epoch,logs=None):
    v_loss=logs.get('val_loss')
  if epoch>0:
    deltav=self.lowest_vloss-v_loss
    pimprov=(deltav/self.lowest_vloss)*100
    self.plist.append(pimprov)
  else:
    pimprov=0.0
  if v_loss<self_lowest_vloss:
    self.lowest_vloss=v_loss
    self.best_weights=self.model.get_weights()
    self.best_epoch=epoch+1
    print(f'\n validation loss of {v_loss:7.4f} is {pimprov:7.4f} % below lowest loss, saving weights from epoch {str(epoch+1):3s} as best weights')
  else:
    pimprov=abs(pimprov)
    print(f'\n validation loss of {v_loss:7.4f} is {pimprov:7.4f} % above lowest loss of {self.lowest_vloss:7.4f} keeping weights from epoch{str(sel
    if self.ask:
      if epoch+1==self.ask_epoch:
```

```
    print(f'\n validation loss of {v_loss:7.4f} is {pimprov:7.4f} % below lowest loss, saving weights from epoch {str(epoch+1):3s} as best weights')
  else:
    pimprov=abs(pimprov)
    print(f'\n validation loss of {v_loss:7.4f} is {pimprov:7.4f} % above lowest loss of {self.lowest_vloss:7.4f} keeping weights from epoch{str(sel
    if self.ask:
      if epoch+1==self.ask_epoch:
        print('\n enter H to end training or an integer for the number of additional epochs to run then ask again')
        ans=input()
    if ans=='H' or ans=='h' or ans=='0':
      print('you entered',ans,'training halted on epoch',epoch+1,'due to user input\n',flush=True)
      self.model.stop_training=True
    else:
      self.ask_epoch+=int(ans)
      if self.ask_epoch>self.epochs:
        print('\n you enterd maximum number of epochs as',self.epochs,'cannot train for',self.ask_epoch,flush=True)
      else:
        print('You enterd ',ans,'Training will continue to epoch',self.ask_epoch,flush=True)
        lr=float(tf.keras.backend.get_value(self.model.optimizer.lr))
        print(f'current LR  is {lr:7.5f}')
        ans=input('')
        if ans=='':
          print(f'keeping current LR of {lr:7.5f}')
        else:
          new_lr=float(ans)
          tf.keras.backend.set_value(self.model.optimizer.lr,new_lr)
          print('changing LR to',ans)
```

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
Model 1