# Skill5

## 2100030910

## Sec-23

## Main.py

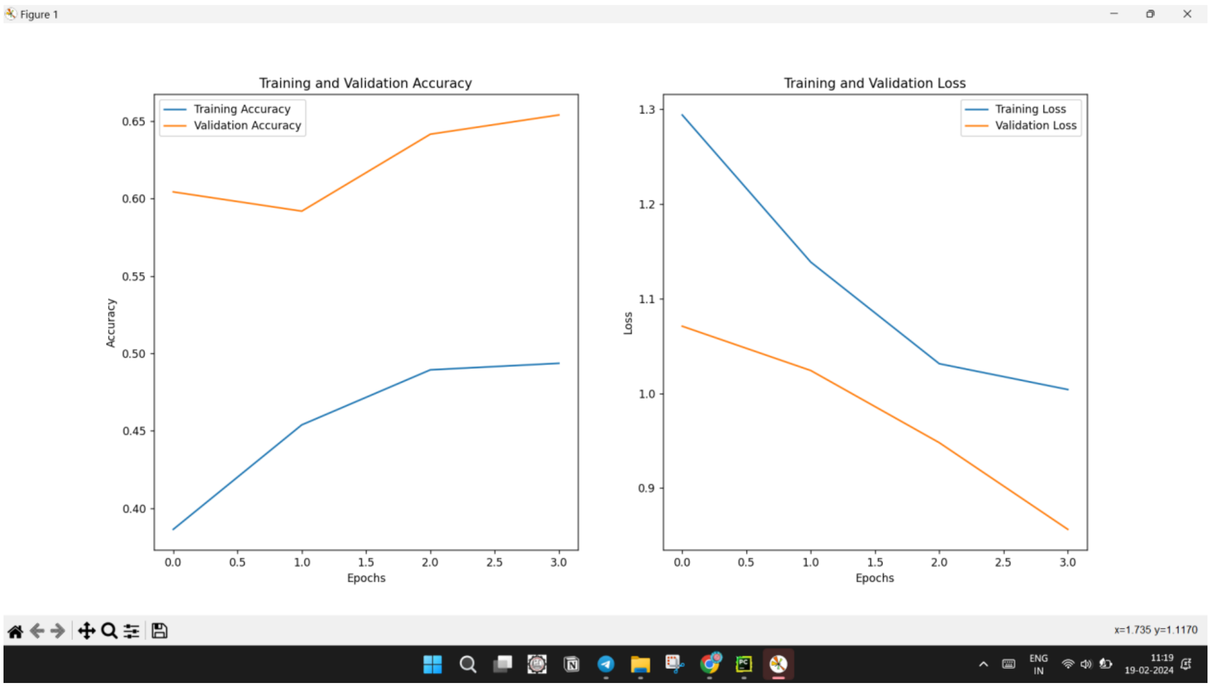
```
import numpy as np
import pandas as pd
import os
import model2 as mc

train_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\train'
valid_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\valid'

cloud_directory = r'C:\Users\dell\PycharmProjex\dlSkill\Skill\data\cloudy'
cloud_train_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\train\cloudy'
cloud_valid_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\valid\cloudy'

water_directory = r'C:\Users\dell\PycharmProjex\dlSkill\Skill\data\water'
water_train_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\train\water'
water_valid_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\valid\water'

green_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\data\green_area'
green_train_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\train\green'
green_valid_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\valid\green'

desert_directory = r'C:\Users\dell\PycharmProjex\dlSkill\Skill\data\desert'
desert_train_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\train\desert'
desert_valid_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\valid\desert'

cloud_image_files = [f for f in os.listdir(cloud_directory) if
                     f.lower().endswith(('.jpg', '.jpeg'))]


water_image_files = [f for f in os.listdir(water_directory) if
                     f.lower().endswith(('.jpg', '.jpeg'))]


desert_image_files = [f for f in os.listdir(desert_directory) if
```

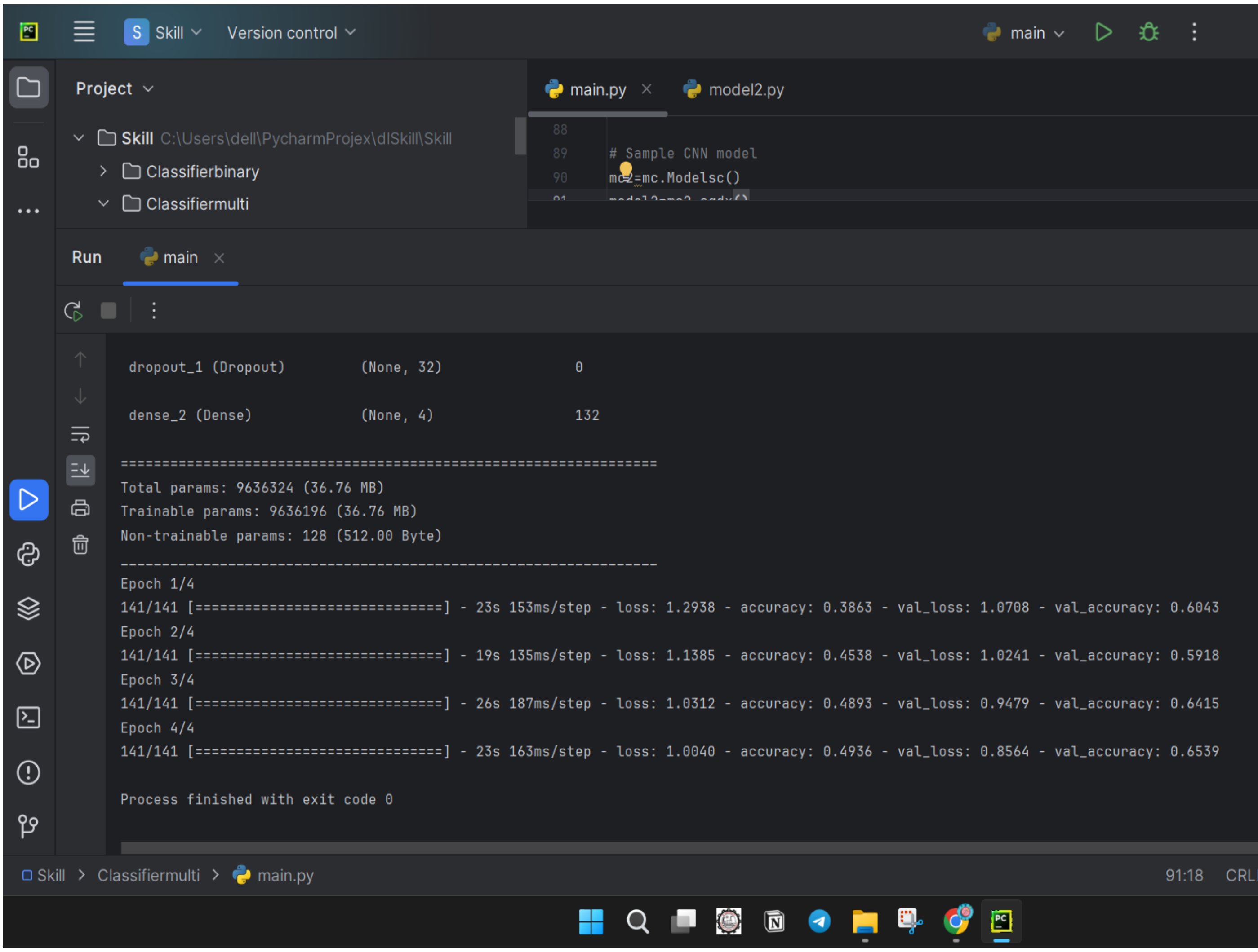
```
                      f.lower().endswith(('.jpg', '.jpeg'))]


green_image_files = [f for f in os.listdir(green_directory) if
                     f.lower().endswith(('.jpg', '.jpeg'))]



os.makedirs(train_directory, exist_ok=True)
os.makedirs(valid_directory, exist_ok=True)


import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
BatchNormalization, Dropout

# Directories
train_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\train'
valid_directory =
r'C:\Users\dell\PycharmProjex\dlSkill\Skill\genData\valid'

# Image data preprocessing
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    # rotation_range=40,
    # width_shift_range=0.2,
    # height_shift_range=0.2,
    # shear_range=0.2,
    # zoom_range=0.2,
    # horizontal_flip=True,
    # fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1.0 / 255)

batch_size = 32

train_generator = train_datagen.flow_from_directory(
    train_directory,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)

validation_generator = validation_datagen.flow_from_directory(
    valid_directory,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

# Sample CNN model
mc2=mc.Modelsc()
model2=mc2.sgdx()
```

```
# Display model summary
model2.summary()

# Training the model
epochs = 4

history = model2.fit(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator
)

# Plotting accuracy and loss curves
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

## Model.py

```
import tensorflow as tf
from tensorflow.keras import layers


class Modelsc:
    def sgdx(self):
        model = tf.keras.Sequential()
        model.add(layers.Flatten(input_shape=(224, 224, 3)))  # Flatten
layer to convert input to 1D

        model.add(layers.Dense(64, activation='relu'))
        model.add(layers.Dropout(0.5))
        model.add(layers.BatchNormalization())

        model.add(layers.Dense(32, activation='relu'))  # Additional dense
layer for complexity
        model.add(layers.Dropout(0.5))

        model.add(
            layers.Dense(4, activation='softmax'))  # Output layer with
softmax activation for multiclass classification

        model.compile(optimizer='sgd', loss='categorical_crossentropy',
metrics=['accuracy'])

        return model
```





```
# Sample CNN model
model=mc.Modelsc()
```

```
 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 4)                 132

=================================================================
Total params: 9636324 (36.76 MB)
Trainable params: 9636196 (36.76 MB)
Non-trainable params: 128 (512.00 Byte)
_________________________________________________________________
Epoch 1/4
141/141 [==============================] - 23s 153ms/step - loss: 1.2938 - accuracy: 0.3863 - val_loss: 1.0708 - val_accuracy: 0.6043
Epoch 2/4
141/141 [==============================] - 19s 135ms/step - loss: 1.1385 - accuracy: 0.4538 - val_loss: 1.0241 - val_accuracy: 0.5918
Epoch 3/4
141/141 [==============================] - 26s 187ms/step - loss: 1.0312 - accuracy: 0.4893 - val_loss: 0.9479 - val_accuracy: 0.6415
Epoch 4/4
141/141 [==============================] - 23s 163ms/step - loss: 1.0040 - accuracy: 0.4936 - val_loss: 0.8564 - val_accuracy: 0.6539

Process finished with exit code 0
```