Skill3

2100030910

Sec-23

Main.py

```
import numpy as np
import pandas as pd
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, img_to_array
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense,
GlobalAveragePooling2D, Dropout, Flatten, BatchNormalization
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow import keras
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.efficientnet import preprocess_input
from PIL import Image
import shutil
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping
import model as mC

train_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\genData2\train'
valid_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\genData2\valid'

cloud_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\data\cloudy'

cloud_train_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\genData2\train\cloudy'
cloud_valid_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\genData2\valid\cloudy'

water_directory = r'C:\Users\dell\PycharmProjects\dlSkill\Skill\data\water'
green_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\data\green_area'
desert_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\data\desert'

non_cloud_train_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\genData2\train\noncloudy'
non_cloud_valid_directory =
r'C:\Users\dell\PycharmProjects\dlSkill\Skill\genData2\valid\noncloudy'


cloud_image_files = [f for f in os.listdir(cloud_directory) if
                     f.lower().endswith(('.jpg', '.jpeg'))]
# Specify a new name format
```

```
# You can customize the format according to your needs


water_image_files = [f for f in os.listdir(water_directory) if
                     f.lower().endswith(('.jpg', '.jpeg'))]
# Specify a new name format
# You can customize the format according to your needs


desert_image_files = [f for f in os.listdir(desert_directory) if
                      f.lower().endswith(('.jpg', '.jpeg'))]
# Specify a new name format
# You can customize the format according to your needs
# In this example, we're using a base name and appending an incremental
number


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


# Loop through the image files and rename them







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



cloud_train_files, cloud_valid_files = train_test_split(cloud_image_files,
test_size=0.2, random_state=42)
for file in cloud_train_files:
    source_file_path = os.path.join(cloud_directory, file)
    destination_file_path = os.path.join(cloud_train_directory, file)
    shutil.copy(source_file_path, destination_file_path)

for file in cloud_valid_files:
    source_file_path = os.path.join(cloud_directory, file)
    destination_file_path = os.path.join(cloud_valid_directory, file)
    shutil.copy(source_file_path, destination_file_path)

water_train_files, water_valid_files = train_test_split(water_image_files,
test_size=0.2, random_state=42)
for file in water_train_files:
    source_file_path = os.path.join(water_directory, file)
    destination_file_path = os.path.join(non_cloud_train_directory, file)
    shutil.copy(source_file_path, destination_file_path)

for file in water_valid_files:
    source_file_path = os.path.join(water_directory, file)
    destination_file_path = os.path.join(non_cloud_valid_directory, file)
    shutil.copy(source_file_path, destination_file_path)

desert_train_files, desert_valid_files =
train_test_split(desert_image_files, test_size=0.2, random_state=42)
for file in desert_train_files:
    source_file_path = os.path.join(desert_directory, file)
```

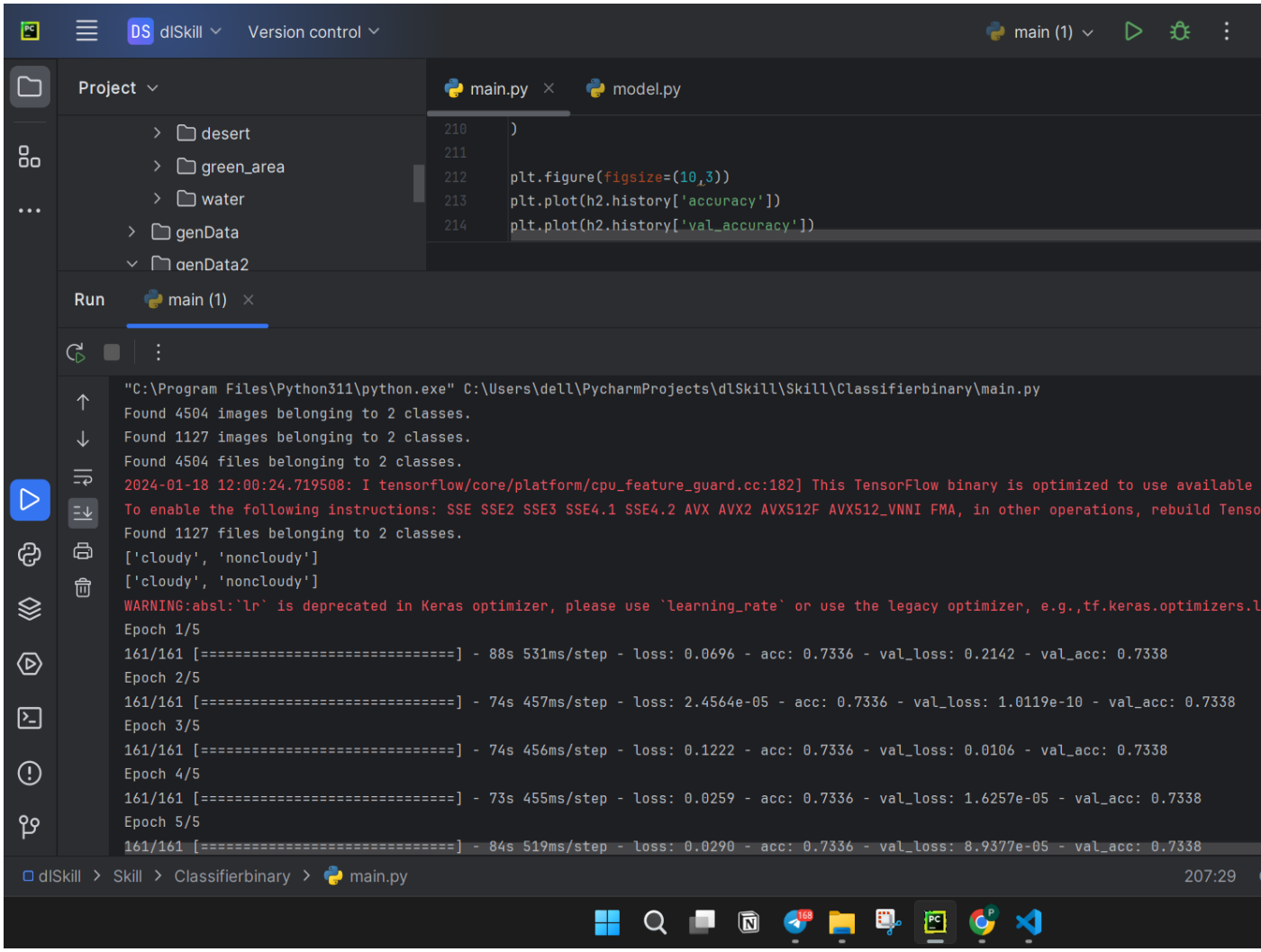
```
    destination_file_path = os.path.join(non_cloud_train_directory, file)
    shutil.copy(source_file_path, destination_file_path)

for file in desert_valid_files:
    source_file_path = os.path.join(desert_directory, file)
    destination_file_path = os.path.join(non_cloud_valid_directory, file)
    shutil.copy(source_file_path, destination_file_path)

green_train_files, green_valid_files = train_test_split(green_image_files,
test_size=0.2, random_state=42)
for file in green_train_files:
    source_file_path = os.path.join(green_directory, file)
    destination_file_path = os.path.join(non_cloud_train_directory, file)
    shutil.copy(source_file_path, destination_file_path)

for file in green_valid_files:
    source_file_path = os.path.join(green_directory, file)
    destination_file_path = os.path.join(non_cloud_valid_directory, file)
    shutil.copy(source_file_path, destination_file_path)

# Creating training image data generator
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_imagenerator = 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'

)

train_generator = train_imagenerator.flow_from_directory(
    train_directory,
    target_size=(224,224),
    batch_size=20,
    class_mode='categorical',
#     classes=class_labels,
    shuffle=True
)

val_imagenerator = ImageDataGenerator(rescale=1.0/255)

validation_generator = val_imagenerator.flow_from_directory(
    valid_directory,
    target_size=(224,224),
    batch_size=20,
    class_mode='binary',
    shuffle=True
)

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
batch_size = 28
image_size = (224, 224)

# Create ImageDataGenerator
```

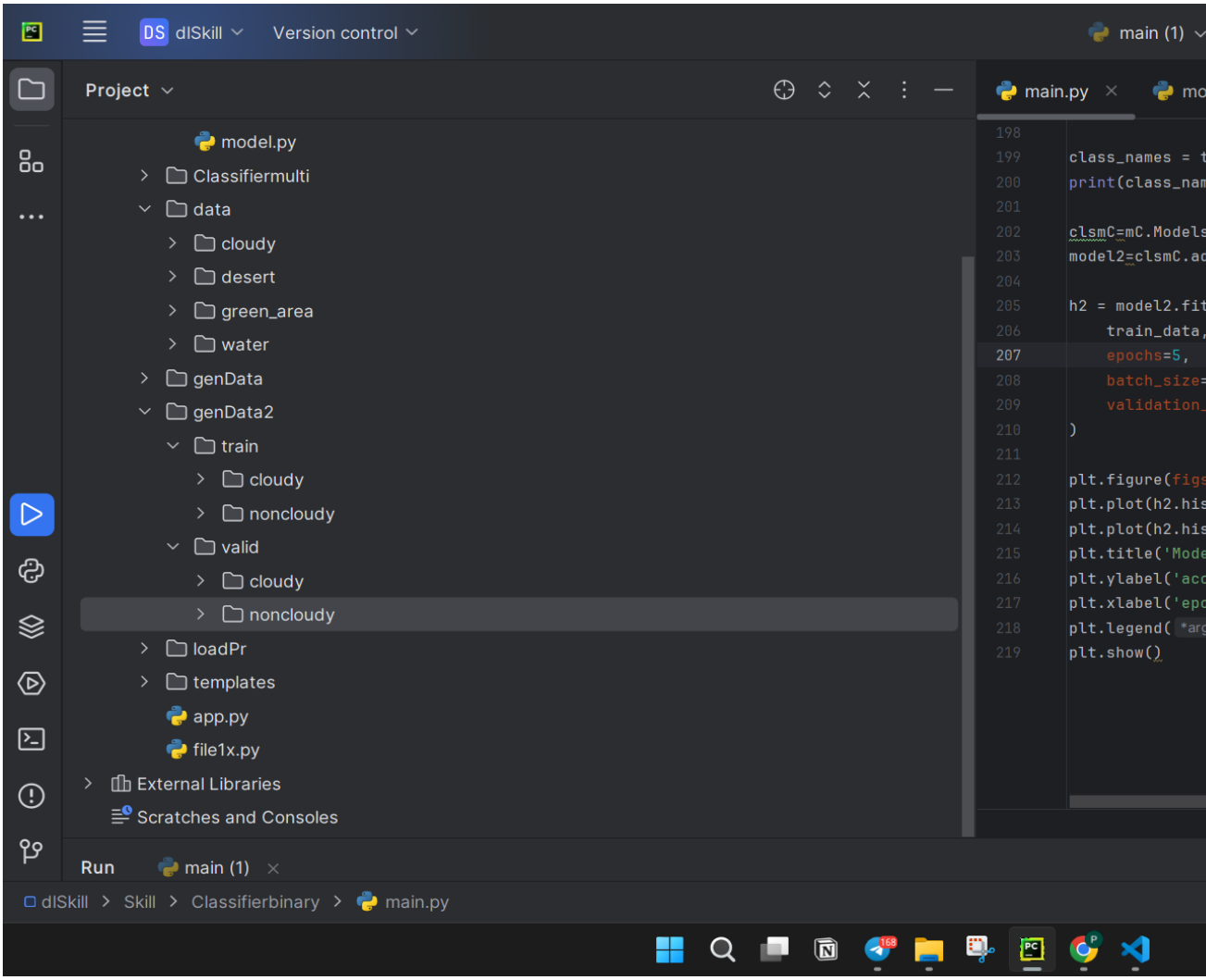
```
train_imagenerator = ImageDataGenerator(
    rescale=1.0/255,   # Normalize pixel values to [0, 1]
    # You can add other data augmentation options here
)

# Create a tf.data.Dataset using image_dataset_from_directory
train_data = tf.keras.preprocessing.image_dataset_from_directory(
    train_directory,
    seed=45,
    shuffle=True,
    image_size=image_size,
    batch_size=batch_size
)

val_data = tf.keras.preprocessing.image_dataset_from_directory(
    valid_directory,
    seed=45,
    shuffle=False,
    image_size=image_size,
    batch_size=batch_size
)

class_names = train_data.class_names
print(class_names)

class_names = train_data.class_names

plt.figure(figsize=(12, 8))
for images, labels in train_data.take(1):
    for i in range(4):
        ax = plt.subplot(2, 2, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")

augmented_images, labels = train_generator.next()

plt.figure(figsize=(12, 8))
for i in range(min(6, augmented_images.shape[0])):
    ax = plt.subplot(2, 3, i + 1)
    plt.imshow(augmented_images[i])
    plt.title(int(np.argmax(labels[i])))  # Convert one-hot encoded label
to integer category
    plt.axis("off")

class_names = train_data.class_names
print(class_names)

clsmC=mC.Modelsc()
model2=clsmC.adam()

h2 = model2.fit(
    train_data,                  # Training data
    epochs=5,                 # Number of training epochs
    batch_size=batch_size,         # Batch size
    validation_data=val_data,    # Early stopping callback
)

plt.figure(figsize=(10,3))
plt.plot(h2.history['acc'])
plt.title('Model Accuracy')
```

```
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Val'], loc='upper left')
plt.show()
```

model.py

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
class Modelsc:
    def adam(self):
        model = tf.keras.Sequential([
            layers.Conv2D(8, (3, 3), padding="valid", input_shape=(224,
224, 3), activation='relu'),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.BatchNormalization(),

            layers.Conv2D(16, (3, 3), padding="valid", activation='relu'),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.BatchNormalization(),

            layers.Conv2D(32, (4, 4), padding="valid", activation='relu'),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.BatchNormalization(),

            layers.Conv2D(64, (4, 4), padding="valid", activation='relu'),
            layers.MaxPooling2D(pool_size=(2, 2)),
            layers.BatchNormalization(),

            layers.Flatten(),
            layers.Dense(64, activation='relu'),
            layers.Dropout(0.15),
            layers.Dense(1, activation='softmax')
        ])

        model.compile(optimizer=tf.keras.optimizers.Adam(lr=1e-5),
                      loss=tf.keras.losses.BinaryCrossentropy(),
                      metrics=['acc'])
        return model
```


dlSkill
Version control
main (1)
Project
desert
green_area
water
genData
genData2
main.py
model.py
)
plt.figure(figsize=(10,3))
plt.plot(h2.history['accuracy'])
plt.plot(h2.history['val_accuracy'])
Run
main (1)
"C:\Program Files\Python311\python.exe" C:\Users\dell\PycharmProjects\dlSkill\Skill\Classifierbinary\main.py
Found 4504 images belonging to 2 classes.
Found 1127 images belonging to 2 classes.
Found 4504 files belonging to 2 classes.
2024-01-18 12:00:24.719508: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild Tenso
Found 1127 files belonging to 2 classes.
['cloudy', 'noncloudy']
['cloudy', 'noncloudy']
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.l
Epoch 1/5
161/161 [==============================] - 88s 531ms/step - loss: 0.0696 - acc: 0.7336 - val_loss: 0.2142 - val_acc: 0.7338
Epoch 2/5
161/161 [==============================] - 74s 457ms/step - loss: 2.4564e-05 - acc: 0.7336 - val_loss: 1.0119e-10 - val_acc: 0.7338
Epoch 3/5
161/161 [==============================] - 74s 456ms/step - loss: 0.1222 - acc: 0.7336 - val_loss: 0.0106 - val_acc: 0.7338
Epoch 4/5
161/161 [==============================] - 73s 455ms/step - loss: 0.0259 - acc: 0.7336 - val_loss: 1.6257e-05 - val_acc: 0.7338
Epoch 5/5
161/161 [==============================] - 84s 519ms/step - loss: 0.0290 - acc: 0.7336 - val_loss: 8.9377e-05 - val_acc: 0.7338
dlSkill > Skill > Classifierbinary > main.py
207:29


DS dlSkill
Version control
main (1)
Project
model.py
Classifiermulti
data
cloudy
desert
green_area
water
genData
genData2
train
cloudy
noncloudy
valid
cloudy
noncloudy
loadPr
templates
app.py
file1x.py
External Libraries
Scratches and Consoles
main.py
class_names = t
print(class_nam
clsmC=mC.Models
model2=clsmC.ac
h2 = model2.fit
train_data,
epochs=5,
batch_size=
validation_
)
plt.figure(figs
plt.plot(h2.his
plt.plot(h2.his
plt.title('Mode
plt.ylabel('acc
plt.xlabel('epo
plt.legend(
plt.show()
Run
main (1)
dlSkill
Skill
Classifierbinary
main.py