ДОДАТОК А

Вихідний код для навчання мереж

```
import os
import shutil
import random
import glob
import matplotlib.pyplot as plt
# Organize data into train, test directories
main_path = "C:/Users/elfognom/PycharmProjects/CCNs"
with_mask_path =
"C:/Users/elfognom/PycharmProjects/CCNs/dataset/with_mask"
incorrect_mask_path =
"C:/Users/elfognom/PycharmProjects/CCNs/dataset/incorrect_mask"
without_mask_path =
"C:/Users/elfognom/PycharmProjects/CCNs/dataset/without_mask"
print(len(os.listdir(with_mask_path)))
print(len(os.listdir(incorrect_mask_path)))
print(len(os.listdir(without_mask_path)))
train_size = 540
test_size = 60
os.chdir(main_path)
# train:
os.makedirs('C:/Users/elfognom/PycharmProjects/CCNs/data/train
/with_mask')
os.makedirs('C:/Users/elfognom/PycharmProjects/CCNs/data/train
/incorrect_mask')
os.makedirs('C:/Users/elfognom/PycharmProjects/CCNs/data/train
/without_mask')
# test:
os.makedirs('C:/Users/elfognom/PycharmProjects/CCNs/data/test/
with_mask')
os.makedirs('C:/Users/elfognom/PycharmProjects/CCNs/data/test/
incorrect_mask')
```

```
      os.makedirs('C:/Users/elfognom/PycharmProjects/CCNs/data/test/
without_mask')
      os.chdir(with_mask_path)
      for i in random.sample(glob.glob('*'), train_size):
          shutil.copy(i,
'C:/Users/elfognom/PycharmProjects/CCNs/data/train/with_mask')
      os.chdir(incorrect_mask_path)
      for i in random.sample(glob.glob('*'), train_size):
          shutil.copy(i,
'C:/Users/elfognom/PycharmProjects/CCNs/data/train/incorrect_mask')
      os.chdir(without_mask_path)
      for i in random.sample(glob.glob('*'), train_size):
          shutil.copy(i,
'C:/Users/elfognom/PycharmProjects/CCNs/data/train/without_mask')
      os.chdir(with_mask_path)
      for i in random.sample(glob.glob('*'), test_size):
          shutil.copy(i,
'C:/Users/elfognom/PycharmProjects/CCNs/data/test/with_mask')
      os.chdir(incorrect_mask_path)
      for i in random.sample(glob.glob('*'), test_size):
          shutil.copy(i,
'C:/Users/elfognom/PycharmProjects/CCNs/data/test/incorrect_mask')
      os.chdir(without_mask_path)
      for i in random.sample(glob.glob('*'), test_size):
          shutil.copy(i,
'C:/Users/elfognom/PycharmProjects/CCNs/data/test/without_mask')
      os.chdir('../../')
      print(len(os.listdir('C:/Users/elfognom/PycharmProjects/CCNs/d
ata/train/with_mask')))
      print(len(os.listdir('C:/Users/elfognom/PycharmProjects/CCNs/d
ata/train/incorrect_mask')))
      print(len(os.listdir('C:/Users/elfognom/PycharmProjects/CCNs/d
ata/train/without_mask')))
```

```
      print(len(os.listdir('C:/Users/elfognom/PycharmProjects/CCNs/d
ata/test/with_mask')))
      print(len(os.listdir('C:/Users/elfognom/PycharmProjects/CCNs/d
ata/test/incorrect_mask')))
      print(len(os.listdir('C:/Users/elfognom/PycharmProjects/CCNs/d
ata/test/without_mask')))
      from tensorflow.keras.models import Sequential
      from tensorflow.keras.layers import Dense, Flatten
      from tensorflow.keras.optimizers import SGD
      from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping
      from tensorflow.keras.applications.vgg16 import VGG16
      from tensorflow.keras.applications.mobilenet_v2 import
MobileNetV2
      from tensorflow.keras.applications.xception import Xception
      from keras import metrics
      from keras.preprocessing.image import ImageDataGenerator
      import pandas as pd

      def creating_model(pretrained, train_set, test_set, name):
          # Model Creation:
          print("Working with " + name)
          for layer in pretrained.layers:
              layer.trainable = False
          print("Freezed layers")
          model = Sequential()
          model.add(pretrained)
          model.add(Flatten())
          model.add(Dense(3, activation="softmax"))
          print("Created model")
          model.summary()

          # The Training Itself:
          val_steps = test_set.n // test_set.batch_size
```

```
            steps_per_epoch = train_set.n // train_set.batch_size
            print("Training params:\nval_steps: " + str(val_steps) +
"\nsteps_per_epoch: " + str(steps_per_epoch))
            checkpoint = ModelCheckpoint(name + ".h5",
monitor="val_accuracy", save_best_only=True, verbose=1)
            earlystop = EarlyStopping(monitor="val_accuracy",
patience=12, verbose=1)
            sgd = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
            model.compile(optimizer=sgd,
loss="categorical_crossentropy",
                          metrics=[metrics.Accuracy(),
                                   metrics.AUC(),
                                   metrics.Recall(),
                                   metrics.Precision()])
            history = model.fit(x=training_set,
                                validation_data=test_set,
                                epochs=200,  # 25
                                callbacks=[checkpoint, earlystop],
                                steps_per_epoch=steps_per_epoch,
                                validation_steps=val_steps)
            # saving model
            model.save("saves/model_" + name)
            print("Saved "+name+" model to disk")
            # save history to json:
            hist_df = pd.DataFrame(history.history)
            hist_json_file = name + 'history.json'
            with open(hist_json_file, mode='w') as f:
                hist_df.to_json(f)
            print("Saved "+name+" history to disk")
        train_dir =
'C:/Users/elfognom/PycharmProjects/CCNs/data/train/'
        test_dir = 'C:/Users/elfognom/PycharmProjects/CCNs/data/test/'
        train_datagen = ImageDataGenerator(rescale=1. / 255,
                                           zoom_range=0.3,
```

```
                                        rotation_range=30,
                                        width_shift_range=0.2,
                                        height_shift_range=0.2,
                                        brightness_range=[0.4,
1.5],
                                        horizontal_flip=True)
      training_set = train_datagen.flow_from_directory(train_dir,
                                                       batch_size=32,
                                                target_size=(224, 224),
                                                   shuffle=True,
                                                   color_mode='rgb',
                                                   class_mode='categori
                                                   cal')
      test_datagen = ImageDataGenerator(rescale=1. / 255)
      testing_set = test_datagen.flow_from_directory(test_dir,
                                    batch_size=32,
target_size=(224, 224),
                                   shuffle=True,
color_mode='rgb',
                                  class_mode='categorical')
      print("Created data sets")

      vgg16 = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
      mobilenet = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
      xcept = Xception(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
      creating_model(mobilenet, training_set, testing_set,
"MobileNetV2")
      creating_model(xcept, training_set, testing_set, "Xception")
      creating_model(vgg16, training_set, testing_set, "VGG16")
      import matplotlib.pyplot as plt
      import pandas as pd
```

```
from keras.utils.vis_utils import plot_model
from tensorflow.keras.models import load_model
def plot_res(history, dops):  #
    plt.figure(figsize=(14, 25))
    plt.subplot(3, 2, 1)
    plt.plot(history['loss'])
    plt.plot(history['val_loss'])
    plt.title('Model Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(['train', 'test'], loc='upper left')
    plt.subplot(3, 2, 2)
    plt.plot(history['categorical_accuracy'])
    plt.plot(history['val_categorical_accuracy'])
    plt.title('Model Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend(['train', 'test'], loc='upper left')
    plt.subplot(3, 2, 3)
    plt.plot(history['precision'+dops])
    plt.plot(history['val_precision'+dops])
    plt.title('Model Precision')
    plt.xlabel('Epochs')
    plt.ylabel('Precision')
    plt.legend(['train', 'test'], loc='upper left')
    plt.subplot(3, 2, 4)
    plt.plot(history['recall'+dops])
    plt.plot(history['val_recall'+dops])
    plt.title('Model Recall')
    plt.xlabel('Epochs')
    plt.ylabel('Recall')
    plt.legend(['train', 'test'], loc='upper left')
    plt.subplot(3, 2, 5)
    plt.plot(history['auc'+dops])
```

```
            plt.plot(history['val_auc'+dops])
            plt.title('Model AUC')
            plt.xlabel('Epochs')
            plt.ylabel('AUC')
            plt.legend(['train', 'test'], loc='upper left')
            plt.show()

        def load_and_evauate(name):
            from keras.preprocessing.image import ImageDataGenerator
            test_datagen = ImageDataGenerator(rescale=1. / 255)
            test_dir =
'C:/Users/elfognom/PycharmProjects/CCNs/dataset'
            test_set = test_datagen.flow_from_directory(test_dir,
                                                        batch_size=32,
target_size=(224, 224),
                                                               shuffle=True,
color_mode='rgb',
class_mode='categorical')
            loaded_model = load_model("saves/model_" + name)
            loaded_model.summary()
            print("Loaded model from disk")
            score = loaded_model.evaluate(x=test_set, verbose=1)
            print("Metrics after evaluating")
            i = 0
            for m in loaded_model.metrics_names:
                print(m + ": " + str(score[i]))
                i += 1
            training_history = pd.read_json(name + 'history.json')
            if name == "VGG16":
                plot_res(training_history, '')
            elif name == "Xception":
                plot_res(training_history, '_1')
            elif name == "MobileNetV2":
                plot_res(training_history, '_2')
```

```
load_and_evauate("MobileNetV2")
load_and_evauate("Xception")
load_and_evauate("VGG16")
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	70 с.
	Інші документи	
2.	Презентаційні матеріали	15 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата	
Розробив		Московка А. Д.			Дослідження та розробка систем комп'ютерного зору на основі глибинних нейронних мереж
Перевірив		Дейнеко А. Д			
Н.контр.		Малєєва І.А.			
Затв.		Філатов В.О.			

Шифр групи	Код напр./спец.
СШІзм-20-2	122
ХНУРЕ кафедра ШІ	