Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт до лабораторної роботи №5

«Згорткові нейронні мережі типу Inception»

з дисципліни

«Програмні засоби проектування та реалізації нейромережевих систем»

Виконала:
студентка групи ІМ-13
Мартинюк Марія Павлівна
номер у списку: 63

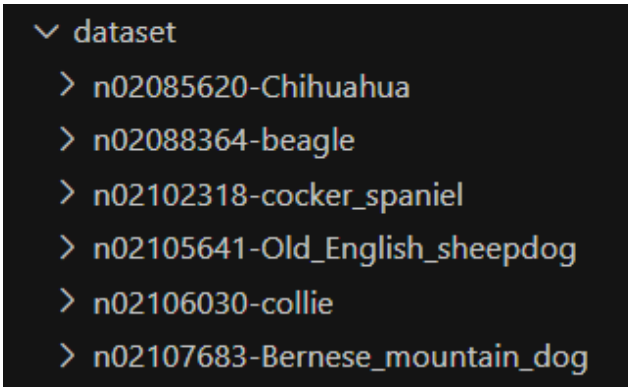
Перевірив:
Шимкович В. М.

Київ 2024

Завдання: Написати програму що реалізує згорткову нейронну мережу Inception V3 для розпізнавання об’єктів на зображеннях. Створити власний дата сет з папки на диску, навчити нейронну мережу на цьому датасеті розпізнавати породу Вашої улюбленої собаки чи кота.

Хід роботи:

За основу мною було обрано датасет Stanford dogs, з якого я обрала набори фото 6 різних порід собак.



Далі перш за все я винесла декілька основних констант: назви обраних порід для зрозумілішого відображення результатів роботи мережі, кількість зображень, на яких пізніше буде протестована модель для майбутнього відображення результату навчання на більш конкретних прикладах та кількість епох.

```
BREED_NAMES = ["Chihuahua", "Beagle", "Cocker Spaniel", "Old English Sheepdog", "Collie", "Bernese Mountain Dog"]
samples_num = 10
epochs = 15
```

Наступним кроком я реалізувала низку додаткових функцій: для відображення графіку залежності точності відносно епох навчання та для відображення фото з передбаченим зображенням та його реальним лейблом.

```
def accuracy_graph_display(history):
    plt.plot(history.history['accuracy'], label='Accuracy', color='green')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy', color='red')
```

```
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0, 1])
    plt.legend(loc='lower right')
    plt.grid(True)
    plt.show()

def image_display(samples_num, prediction, testX, testY):
    fig = plt.figure(figsize=(20, 10))
    for i in range(samples_num):
      fig.add_subplot(samples_num // 2, 2, i + 1)
      image = testX[i][:, :, ::-1]
      plt.imshow(image)
      plt.axis("off")
      plt.title("Predicted: " + BREED_NAMES[np.argmax(prediction[i])] +
" | Actual: " + BREED_NAMES[np.argmax(testY[i])])

    plt.show()
```

Для реалізації заданої за завданням нейронної мережі був створений відповідний клас ImageClassifier, де створювалась модель Inception V3 з попередньо навченими вагами та заморожувались її шари для навчання. Пізніше на основі неї відбувалось створення із додатковими шарами та тренування нової нейронної мережі.

```
class ImageClassifier:
    def __init__(self, trainX, trainY, testX, testY):
        self.trainX = trainX
        self.trainY = trainY
        self.testX = testX
        self.testY = testY

        self.model = self.create_model(input_shape=trainX[0].shape,
num_classes=trainY.shape[1])

    def create_model(self, input_shape, num_classes):
        base_model = InceptionV3(input_shape=input_shape,
include_top=False, weights='imagenet')
        for layer in base_model.layers:
            layer.trainable = False
```

```
        x = layers.Flatten()(base_model.output)
        x = layers.Dense(1024, activation='relu')(x)
        x = layers.Dropout(0.2)(x)
        output = layers.Dense(num_classes, activation='sigmoid')(x)

        model = Model(inputs=base_model.input, outputs=output)
        return model

    def train_model(self, train_datagen, testX, testY, epochs=5):
        self.model.compile(optimizer="adam",
loss="categorical_crossentropy", metrics=["accuracy"])
        history = self.model.fit(train_datagen.flow(self.trainX,
self.trainY), validation_data=(testX, testY), epochs=epochs,
validation_freq=1)
        return history

    def predict(self, data):
        return self.model.predict(data)
```

Після цього в головній функції відбувалась обробка підготовленого завчасно датасету, розподіл даних на тренувальні та тестові, ініціалізація екземплярів класу нейронної мережі ImageClassifier на основі Inception V3. Також відбувалось додаткове тестування моделі на 10 інших фотографіях для наочності результатів навчання. Окрім цього в головній функції відбувався виклик функцій для відображення графіку.

```
def main():
    dataPath = "dataset"
    data = []
    labels = []
    imagePaths = sorted(list(paths.list_images(dataPath)))
    random.seed(42)
    random.shuffle(imagePaths)

    for imagePath in imagePaths:
        image = cv2.imread(imagePath)
        image = cv2.resize(image, (150, 150))
        data.append(image)
        label = imagePath.split(os.path.sep)[-2]
```

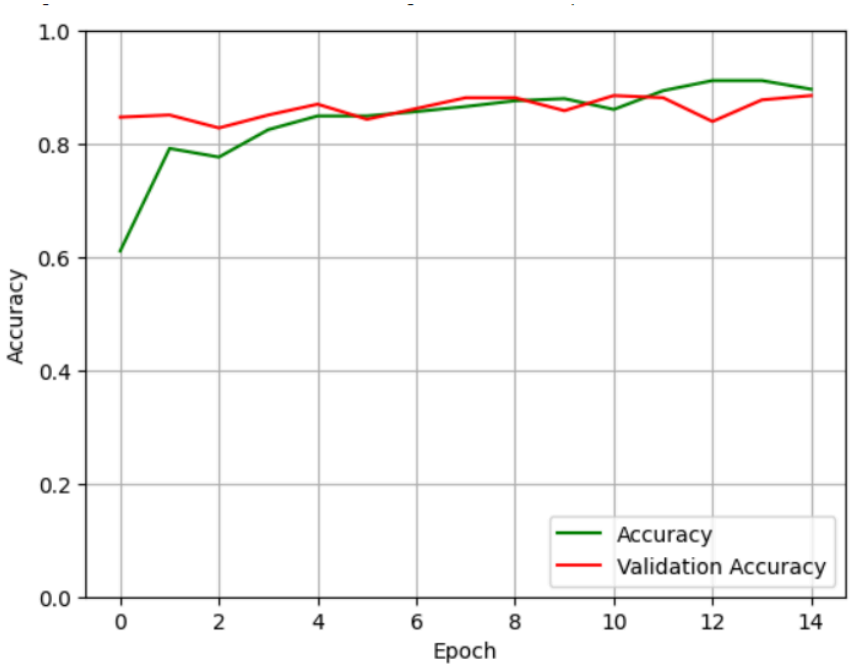
```
        labels.append(label)

    data = np.array(data, dtype="float") / 255.0
    labels = np.array(labels)

    (trainX, testX, trainY, testY) = train_test_split(data, labels,
test_size=0.25, random_state=42)
    enc = OneHotEncoder()
    trainY = trainY.reshape(-1, 1)
    testY = testY.reshape(-1, 1)
    trainY = enc.fit_transform(trainY).toarray()
    testY = enc.transform(testY).toarray()

    train_data = ImageDataGenerator(rotation_range=30,
width_shift_range=0.1, height_shift_range=0.1, shear_range=0.2,
zoom_range=0.2, horizontal_flip=True, fill_mode="nearest")

    classifier = ImageClassifier(trainX, trainY, testX, testY)
    history = classifier.train_model(train_data, testX, testY,
epochs=epochs)
    prediction = classifier.predict(testX)

    accuracy_graph_display(history)
    image_display(samples_num, prediction, testX, testY)

main()
```

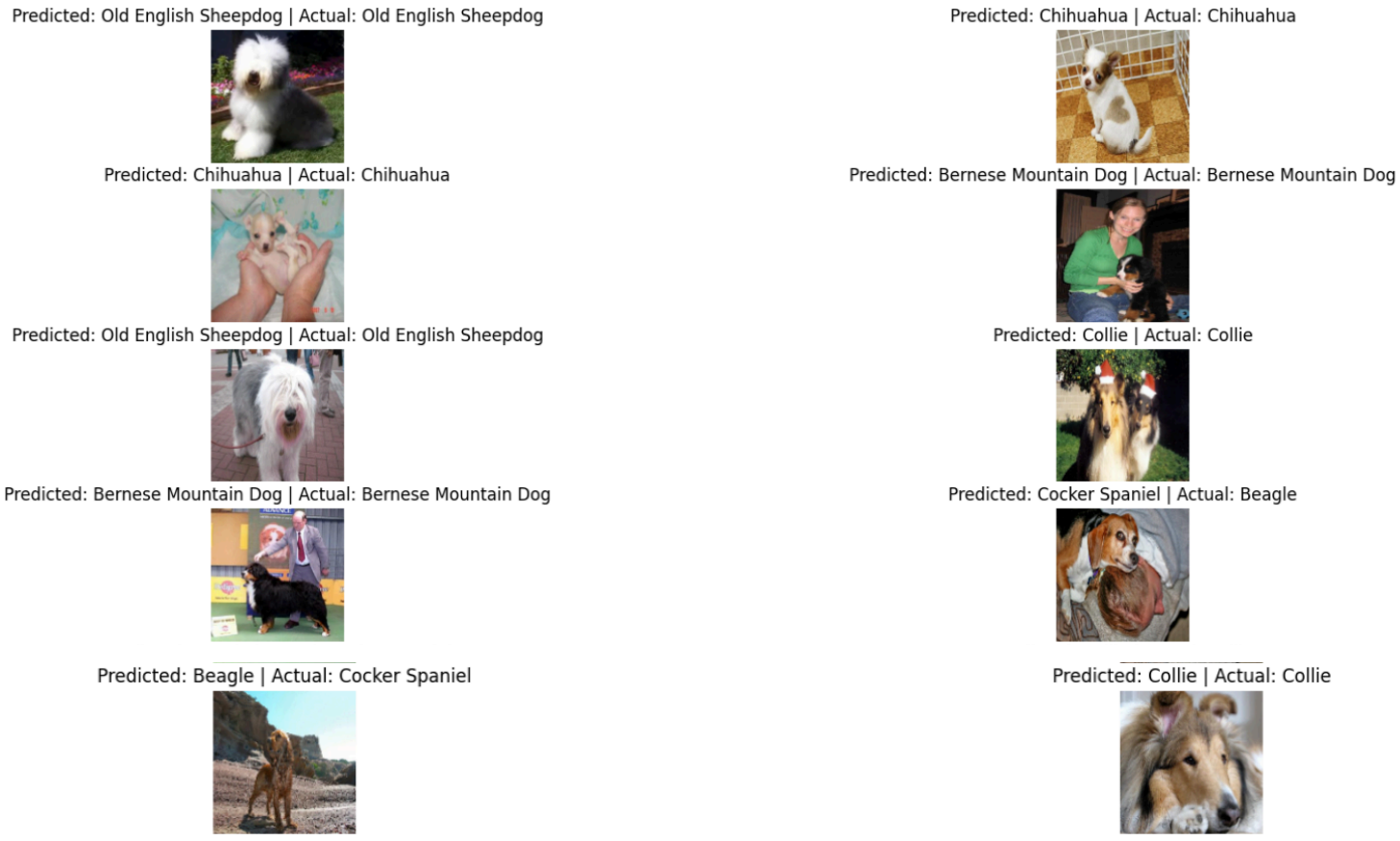
Результати навчання:

По завершенню навчання була досягнута точність 88.55%, що є досить непоганим результатом.

```
Epoch 14/15
25/25 [==============================] - 68s 3s/step - loss: 0.2734 - accuracy: 0.9120 - val_loss: 0.5385 - val_accuracy: 0.8779
Epoch 15/15
25/25 [==============================] - 61s 2s/step - loss: 0.3057 - accuracy: 0.8967 - val_loss: 0.4974 - val_accuracy: 0.8855
```

Графік залежності точності відносно епох навчання:





Висновок:

Як результат виконання лабораторної роботи було створено згорткову нейронну мережу Inception V3 для розпізнавання об’єктів на зображеннях. Окрім цього для наочності було виведено результати навчання у вигляді

графіку з відображення зміни точності відносно епох навчання. Додатково модель було протестовано на іншому наборі даних для перевірки якості роботи моделі в реальних умовах. За отриманими результатами можна зробити висновок, що мережа, пройшовши 15 епох, навчилась класифікувати зображення із точністю 88.55%.