Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем
Кафедра Информатики

ЛАБОРАТОРНАЯ РАБОТА №5
«Применение сверточных нейронных сетей (бинарная классификация)

Магистрант:
гр. 956241
Шуба И.А.

Проверил:
Заливако С. С.

Минск, 2020

ХОД РАБОТЫ

Задание.

Данные: Набор данных DogsVsCats, который состоит из изображений различной размерности, содержащих фотографии собак и кошек. Обучающая выборка включает в себя 25 тыс. изображений (12,5 тыс. кошек: cat.0.jpg, …, cat.12499.jpg и 12,5 тыс. собак: dog.0.jpg, …, dog.12499.jpg), а контрольная выборка содержит 12,5 тыс. неразмеченных изображений. Скачать данные, а также проверить качество классификатора на тестовой выборке можно на сайте Kaggle -> https://www.kaggle.com/c/dogs-vs-cats/data

Задание 1.

Загрузите данные. Разделите исходный набор данных на обучающую, валидационную и контрольную выборки.

Задание 2.

Реализуйте глубокую нейронную сеть с как минимум тремя сверточными слоями. Какое качество классификации получено?

Задание 3.

Примените дополнение данных (data augmentation). Как это повлияло на качество классификатора?

Задание 4.

Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, Inception и т.п.), применив передаточное обучение. Как это повлияло на качество классификатора?

Какой максимальный результат удалось получить на сайте Kaggle? Почему?

Результат выполнения:

Задание 1.Загрузите данные. Разделите исходный набор данных на обучающую, валидационную и контрольную выборки.

Объявим несколько констант и загрузим наборы данных.

```
TRAIN_DIR = '../dogs-vs-cats/train/'
TEST_DIR = '../dogs-vs-cats/test1/'

ROWS = 150
COLS = 150
CHANNELS = 3

BATCH_SIZE=128
```

```
original_train_images = [TRAIN_DIR+i for i in os.listdir(TRAIN_DIR)]
random.shuffle(original_train_images)
train_images = original_train_images[:17500]
validation_images = original_train_images[17500:22000]
test_images = original_train_images[22000:]
```

Рисунок 1 – Загрузка и разделение данных

Напишем функцию, обрабатывающую датасет.

```
def prep_data(images):
    count = len(images)
    X = np.ndarray((count, ROWS, COLS, CHANNELS), dtype=np.float32)
    y = np.zeros((count,), dtype=np.float32)

    for i, image_file in enumerate(images):
        img = image.load_img(image_file, target_size=(ROWS, COLS))
        X[i] = image.img_to_array(img)/255
        if 'dog' in image_file.split('/dogs-vs-cats/')[1]:
            y[i] = 1.
        if i%1000 == 0: print('Processed {} of {}'.format(i, count))

    return X, y
```

Рисунок 2 – Функция для обработки датасета

```
X_train, y_train = prep_data(train_images)
```

```
X_validation, y_validation = prep_data(validation_images)
```

```
X_test, y_test = prep_data(test_images)
```

Рисунок 3 – Обработка разделенных данных

```
print('Train: ',X_train.shape, y_train.shape)
print('Validation: ', X_validation.shape, y_validation.shape)
print('Test: ', X_test.shape, y_test.shape)
```

```
Train:  (17500, 150, 150, 3) (17500,)
Validation:  (4500, 150, 150, 3) (4500,)
Test:  (3000, 150, 150, 3) (3000,)
```

Рисунок 4 – Размеры разделенных данных

Задание 2.Реализуйте глубокую нейронную сеть с как минимум тремя сверточными слоями. Какое качество классификации получено?

Реализуем глубокую сверточную сеть.

```
model = Sequential()
model.add(Conv2D(64, kernel_size=(4,4), activation = 'relu', input_shape=(ROWS, COLS ,C
HANNELS), padding='same'))
model.add(Dropout(0.4))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(64, kernel_size = (4, 4), activation = 'relu', padding='same' ))
model.add(Dropout(0.4))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Conv2D(64, kernel_size = (3, 3), activation = 'relu'))
model.add(Dropout(0.4))
model.add(MaxPooling2D(pool_size = (2, 2)))

model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(NUM_CLASSES, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy',
              optimizer='nadam',
              metrics=['accuracy'])
keras2ascii(model)
```

Рисунок 10 – Глубокая сверточная нейронная сеть

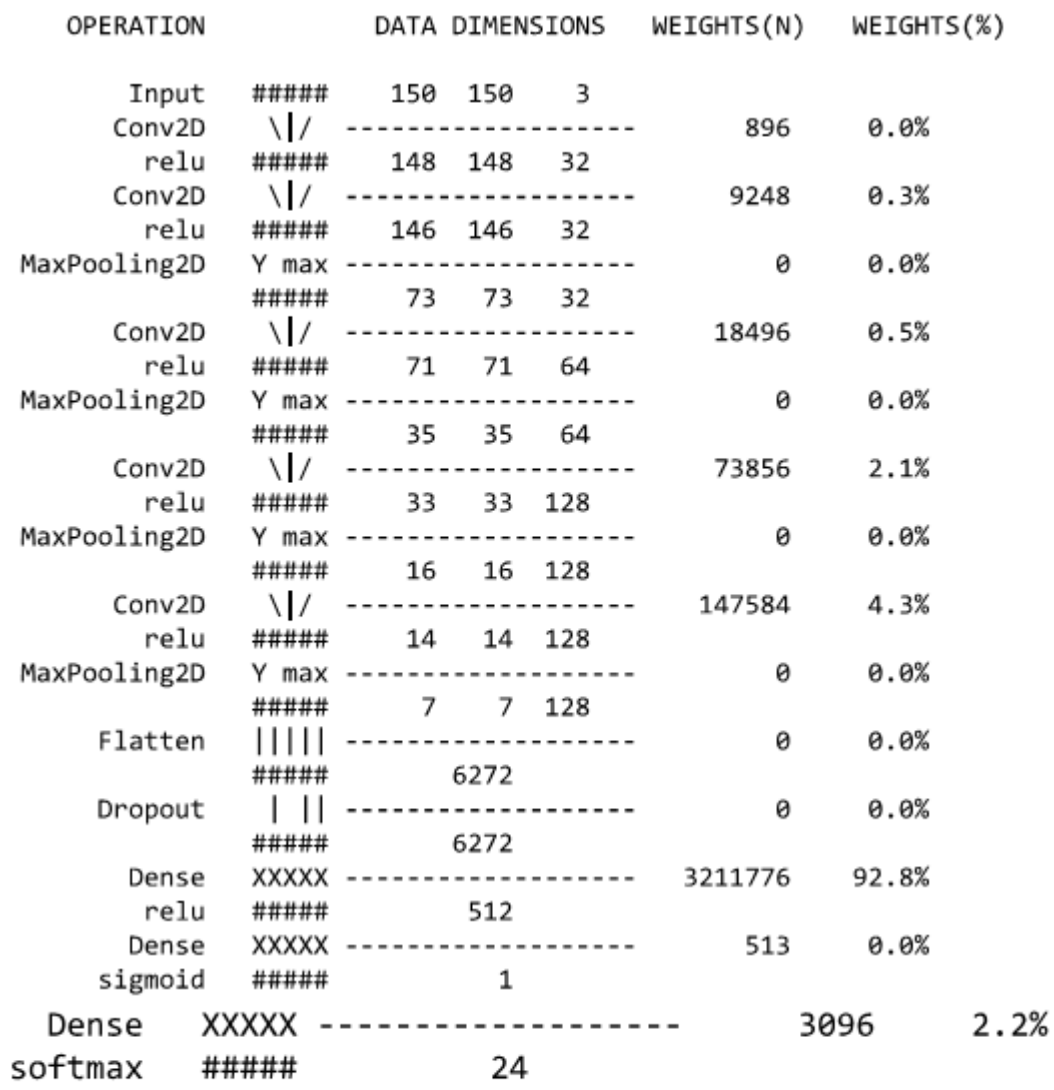
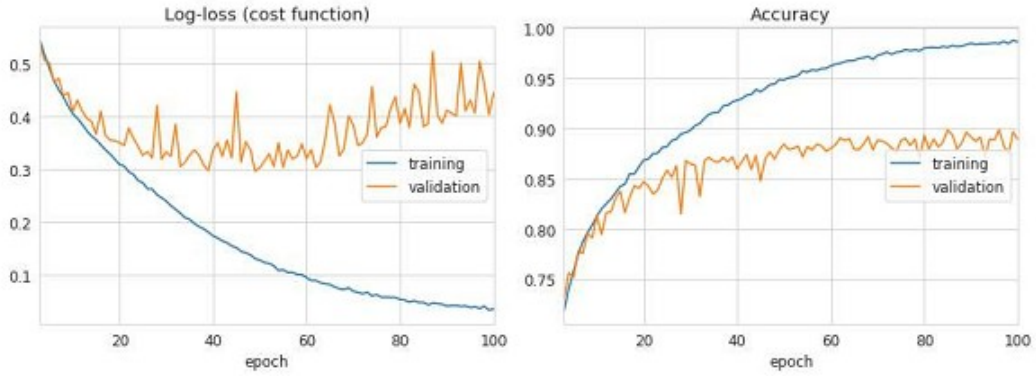


Рисунок 11 – Схема глубокой сверточной нейронной сети

```
train_steps = len(train_images)/BATCH_SIZE
validation_steps = len(validation_images)/BATCH_SIZE
history = model.fit(X_train, y_train,
                    epochs=100,
                    batch_size=BATCH_SIZE,
                    validation_data=(X_validation, y_validation),
                    callbacks=[PlotLossesKeras()],
                    verbose=2)
```




```
Log-loss (cost function):
training   (min:    0.034, max:    0.661, cur:    0.037)
validation (min:    0.297, max:    0.644, cur:    0.445)

Accuracy:
training   (min:    0.595, max:    0.988, cur:    0.986)
validation (min:    0.596, max:    0.899, cur:    0.889)
```

Рисунок 12 – Результаты обучения

Как видно из графиков, точность на валидационных данных на значении ~0,87 остается приблизительно неизменной после 30 эпохи, а точность на тренировочных данных продолжает расти. Скорее всего, модели переобучилась, так как еще значения log-loss также отличаются для обеих выборок. Где-то на 40 эпохе значения на валидационной выборке начитают иметь положительный тренд, чего быть не должно, также заметим, что разброс значений также увеличился, что говорит о нестабильности модели. Максимальная точность на валидационных данных составила 0,899, а минимальный log-loss – 0,297.

```
y_pred = model.predict_classes(X_test)
evaluation1 = model.evaluate(X_test, y_test)
acc1 = accuracy_score(y_test, y_pred)
f1_1 = f1_score(y_test, y_pred, average='weighted')
print(evaluation1)
print('Accuracy: ', acc1)
print('F1 score: ', f1_1)
!telegram-send 'Model 1: Eval: {evaluation1} Acc: {acc1} F1: {f1_1}'
```

```
3000/3000 [==============================] - 13s 4ms/step
[0.42147476291656494, 0.891666666507721]
Accuracy:  0.8916666666666667
F1 score:  0.8916225269318286
```

Рисунок 13 – Результаты модели на контрольной выборке

Таким образом, точность модели на контрольной выборке составила 0,89, а log-loss 0.42.

Задание 3.Примените дополнение данных (data augmentation). Как это повлияло на качество классификатора?

Применим дополнение данных путем генерации изображений с некоторыми преобразованиями.

```
train_datagen = image.ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,)

validation_datagen = image.ImageDataGenerator()
```

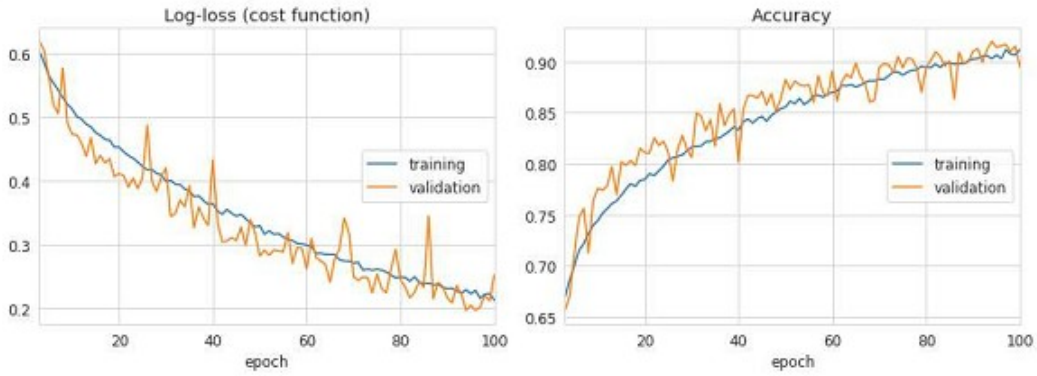
```
train_generator = train_datagen.flow(
    X_train,
    y_train,
    batch_size=BATCH_SIZE)

validation_generator = validation_datagen.flow(
    X_validation,
    y_validation,
    batch_size=BATCH_SIZE)
```

Рисунок 14 – Генерация изображений с некоторыми преобразованиями

```
train_steps = len(train_images)/BATCH_SIZE
validation_steps = len(validation_images)/BATCH_SIZE

history = model.fit_generator(
    train_generator,
    steps_per_epoch=train_steps,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=validation_steps,
    callbacks=[PlotLossesKeras()],
    verbose=2)
```




```
Log-loss (cost function):
training   (min:    0.214, max:    0.680, cur:    0.214)
validation (min:    0.197, max:    0.654, cur:    0.253)

Accuracy:
training   (min:    0.561, max:    0.912, cur:    0.912)
validation (min:    0.598, max:    0.920, cur:    0.895)
```

Рисунок 15 – Результаты обучения

Как видно из графиков, точность и log-loss на валидационных данных приблизительно такая же, как и на тренировочных. Максимальная точность на валидационных данных составила 0,92, а минимальный log-loss – 0,197. Также стоит отметить, что путем аугументации данных, была решена проблема с тем, что было достаточно сильное расхождение графиков для валидационной и контрольной выборок.

```
y_pred = model.predict_classes(X_test)
evaluation2 = model.evaluate(X_test, y_test)
acc2 = accuracy_score(y_test, y_pred)
f1_2 = f1_score(y_test, y_pred, average='weighted')
print(evaluation2)
print('Accuracy: ', acc2)
print('F1 score: ', f1_2)
!telegram-send 'Model 2: Eval: {evaluation2} Acc: {acc2} F1: {f1_2}'
```

```
3000/3000 [==============================] - 11s 4ms/step
[0.245685743590196, 0.905]
Accuracy:  0.905
F1 score:  0.9047909673179647
```

Рисунок 16 – Результаты модели на контрольной выборке

Таким образом, точность модели на контрольной выборке составила 0,905, а log-loss 0,245.

Задание 4.Поэкспериментируйте с готовыми нейронными сетями (например, AlexNet, VGG16, Inception и т.п.), применив передаточное обучение. Как это повлияло на качество классификатора? Какой максимальный результат удалось получить на сайте Kaggle? Почему?

Поэксперементируем с сетью InceptionV3.

```
base_model = InceptionV3(include_top = False, weights = 'imagenet', input_shape = (ROWS
, COLS, CHANNELS))
```

```
model = Sequential()
model.add(base_model)
model.add(Flatten())

model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['accuracy'])
model.summary()
keras2ascii(model)
```

Рисунок 18 –Нейронная сеть с предобученной сетью InceptionV3

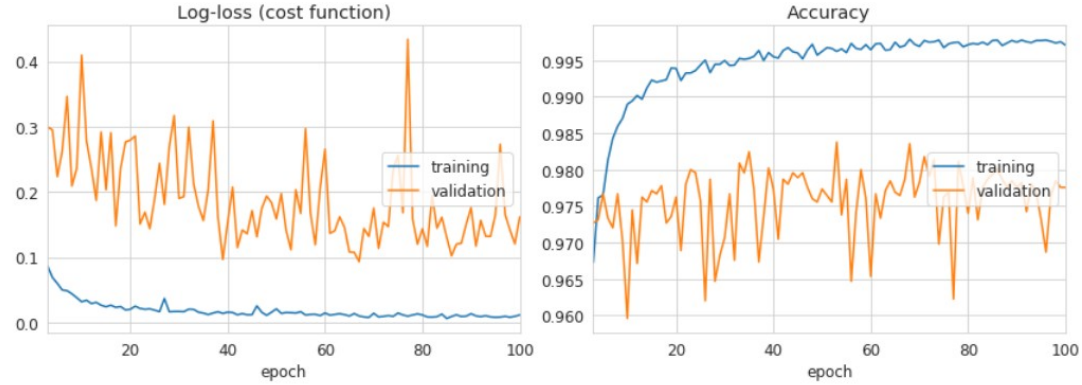
```
OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

    Input   #####    150  150    3
    Model   ????? -------------------  21802784    82.2%
            #####      3    3 2048
  Flatten   ||||| -------------------         0     0.0%
            #####       18432
    Dense   XXXXX -------------------   4718848    17.8%
     relu   #####         256
    Dense   XXXXX -------------------       257     0.0%
  sigmoid   #####           1
```

Рисунок 20 – Схема нейронной сети с предобученной сетью InceptionV3

```
train_steps = len(train_images)/BATCH_SIZE
validation_steps = len(validation_images)/BATCH_SIZE

history = model.fit_generator(
    train_generator,
    steps_per_epoch=train_steps,
    epochs=100,
    validation_data=validation_generator,
    validation_steps=validation_steps,
    callbacks=[PlotLossesKeras()],
    verbose=2)
```




```
Log-loss (cost function):
training   (min:    0.006, max:    0.277, cur:    0.011)
validation (min:    0.093, max:    0.434, cur:    0.162)

Accuracy:
training   (min:    0.892, max:    0.998, cur:    0.997)
validation (min:    0.960, max:    0.984, cur:    0.978)
```

Рисунок 21 – Результаты обучения

Как видно из графиков, точность и log-loss на валидационных данных немного расходятся с тренировочным набором данных. Максимальная точность на валидационных данных составила 0,984, а минимальный log-loss – 0,093.

```
y_pred = model.predict_classes(X_test)
evaluation3 = model.evaluate(X_test, y_test)
acc3 = accuracy_score(y_test, y_pred)
f1_3 = f1_score(y_test, y_pred, average='weighted')
print(evaluation3)
print('Accuracy: ', acc3)
print('F1 score: ', f1_3)
!telegram-send 'Model 3: Eval: {evaluation3} Acc: {acc3} F1: {f1_3}'
```

```
3000/3000 [==============================] - 58s 19ms/step
[0.15614243963162078, 0.9776666666666667]
Accuracy:  0.9776666666666667
F1 score:  0.9776606984139412
```

Рисунок 22 – Результаты модели на контрольной выборке

Таким образом, точность модели на контрольной выборке составила 0,97 , а log-loss 0.15 при обучении на 100 эпохах. Очевидно, что качество классификатора увеличилось по сравнению с предыдущими моделями.

Вывод:

В ходе выполнения лабораторной работы был изучен набор данных «CatsUsDogs», были реализованы сверточные сети с разными архитектурами, максимальный скор, который удалось получить, составил 0,97, а минимальный log-loss – 0,019. Причем в модели не наблюдалось явного переобучения, так как графики точности на тренировочных и валидационных данных практически повторяли друг друга с ростом числа эпох. Было реализовано передаточное обучение. В качестве предобученной модели была использована одна из популярных сетей для работы с изображениями InceptionV3.