**Жуковский Павел, 3 курс, 12 группа, кафедра КТС**

**Задание 1.**

Выполните базовую версию программы обучения сверточной нейронной сети распознавания изображений. Попытайтесь улучшить качество обучения сети путем изменения различных гиперпараметров. Сохраните обученую сверточную нейронную сеть распознавания изображений.

**import** **numpy**

**from** **keras.datasets** **import** cifar10

**from** **keras.models** **import** Sequential

**from** **keras.layers** **import** Dense, Flatten, Activation

**from** **keras.layers** **import** Dropout

**from** **keras.layers.convolutional** **import** Conv2D, MaxPooling2D

**from** **keras.utils** **import** np\_utils

**from** **keras.optimizers** **import** SGD

*# Задаем seed для повторяемости результатов*

numpy.random.seed(42)

*# Загружаем данные*

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

*# Размер мини-выборки*

batch\_size = 32

*# Количество классов изображений*

nb\_classes = 10

*# Количество эпох для обучения*

nb\_epoch = 2

*# Размер изображений*

img\_rows, img\_cols = 32, 32

*# Количество каналов в изображении: RGB*

img\_channels = 3

*# Нормализуем данные*

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255

X\_test /= 255

*# Преобразуем метки в категории*

Y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)

Y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)

*# Создаем последовательную модель*

model = Sequential()

*# Первый сверточный слой*

model.add(Conv2D(32, (3, 3), padding='same',

input\_shape=(32, 32, 3), activation='relu'))

*# Второй сверточный слой*

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))

*# Первый слой подвыборки*

model.add(MaxPooling2D(pool\_size=(2, 2)))

*# Слой регуляризации Dropout*

model.add(Dropout(0.25))

*# Третий сверточный слой++++++++++++++++++++++*

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

*# Четвертый сверточный слой*

model.add(Conv2D(64, (3, 3), activation='relu'))

*# Второй слой подвыборки*

model.add(MaxPooling2D(pool\_size=(2, 2)))

*# Слой регуляризации Dropout*

model.add(Dropout(0.25))

*# Слой преобразования данных из 2D представления в плоское*

model.add(Flatten())

*# Полносвязный слой для классификации*

model.add(Dense(512, activation='relu'))

*# Слой регуляризации Dropout*

model.add(Dropout(0.5))

*# Выходной полносвязный слой*

model.add(Dense(nb\_classes, activation='softmax'))

*#*

*# Задаем параметры оптимизации*

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=**True**)

model.compile(loss='categorical\_crossentropy',

optimizer=sgd,

metrics=['accuracy'])

*# Обучаем модель*

model.fit(X\_train, Y\_train,

batch\_size=batch\_size,

epochs=nb\_epoch,

validation\_split=0.1,

shuffle=**True**,

verbose=2)

*# Оцениваем качество обучения модели на тестовых данных*

scores = model.evaluate(X\_test, Y\_test, verbose=0)

print("Точность работы на тестовых данных: **%.2f%%**" % (scores[1]\*100))

*# Генерируем описание модели в формате json*

model\_json = model.to\_json()

*# Записываем модель в файл*

json\_file = open("cifar\_model.json", "w")

json\_file.write(model\_json)

json\_file.close()

model.save\_weights("cifar\_model.h5")

print("Сохранили Model")

**Задание 2.**[**¶**](https://htmtopdf.herokuapp.com/ipynbviewer/temp/53037fa07ee8ff0f5da6d4db8f398a0f/lab7.html?t=1605194572659#%D0%97%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-2.)

Загрузите сохраненную сверточную нейронную сеть распознавания изображений.

**from** **keras.models** **import** model\_from\_json

**import** **numpy** **as** **np**

**from** **PIL** **import** Image

json\_file = open("cifar\_model.json", "r")

loaded\_model\_json = json\_file.read()

json\_file.close()

loaded\_model = model\_from\_json(loaded\_model\_json)

loaded\_model.load\_weights("cifar\_model.h5")

loaded\_model.compile(loss='categorical\_crossentropy', optimizer='sgd', metrics=['accuracy'])

print ("Загрузили Model")

*# \*\*\*\*\* Загружаем изображение в Keras:*

**import** **numpy** **as** **np**

**from** **keras.preprocessing** **import** image

img\_path = 'plane.jpg'

img = image.load\_img(img\_path, target\_size=(32, 32))

*# В отличие от рукописных цифр, в этот раз изображение цветное и его размер 32х32, в соответствии с форматом CIFAR-10. Преобразуем картинку в массив numpy:*

x = image.img\_to\_array(img)

x /= 255

x = np.expand\_dims(x, axis=0)

*# Запускаем распознавание объекта:*

prediction = loaded\_model.predict(x)

*# Для удобства вывода задаем список с названиями классов объектов:*

classes=['самолет', 'автомобиль', 'птица', 'кот', 'олень', 'собака', 'лягушка', 'лошадь', 'корабль', 'грузовик']

*# Печатаем результат распознавания:*

print(classes[np.argmax(prediction)])

**Вывод**

Train on 45000 samples, validate on 5000 samples

Epoch 1/2

- 201s - loss: 1.7533 - accuracy: 0.3549 - val\_loss: 1.4855 - val\_accuracy: 0.4720

Epoch 2/2

- 192s - loss: 1.3239 - accuracy: 0.5245 - val\_loss: 1.0974 - val\_accuracy: 0.6190

Точность работы на тестовых данных: 60.47%

Сохранили Model

Загрузили Model

Самолет

## Задание 3.

Используйте обученные сверточные нейронные сети для распознавания своих объектов и рукописных цифр.

**import** **numpy**

**from** **keras.models** **import** model\_from\_json

**from** **keras.datasets** **import** cifar10

**from** **keras.models** **import** Sequential

**from** **keras.layers** **import** Dense, Flatten, Activation

**from** **keras.layers** **import** Dropout

**from** **keras.layers.convolutional** **import** Conv2D, MaxPooling2D

**from** **keras.utils** **import** np\_utils

**from** **keras.optimizers** **import** SGD

**import** **numpy** **as** **np**

**from** **PIL** **import** Image

json\_file = open("cifar\_model.json", "r")

loaded\_model\_json = json\_file.read()

json\_file.close()

loaded\_model = model\_from\_json(loaded\_model\_json)

loaded\_model.load\_weights("cifar\_model.h5")

loaded\_model.compile(loss='categorical\_crossentropy', optimizer='sgd', metrics=['accuracy'])

print ("Загрузили Model")

*# \*\*\*\*\* Загружаем изображение в Keras:*

**import** **numpy** **as** **np**

**from** **keras.preprocessing** **import** image

img\_path = 'automobile.png'

img = image.load\_img(img\_path, target\_size=(32, 32))

*# В отличие от рукописных цифр, в этот раз изображение цветное и его размер 32х32, в соответствии с форматом CIFAR-10. Преобразуем картинку в массив numpy:*

x = image.img\_to\_array(img)

x /= 255

x = np.expand\_dims(x, axis=0)

*# Запускаем распознавание объекта:*

prediction = loaded\_model.predict(x)

*# Для удобства вывода задаем список с названиями классов объектов:*

classes=['самолет', 'автомобиль', 'птица', 'кот', 'олень', 'собака', 'лягушка', 'лошадь', 'корабль', 'грузовик']

*# Печатаем результат распознавания:*

print(classes[np.argmax(prediction)])

**Вывод**

Загрузили Model

Автомобиль

**import** **numpy**

**from** **keras.models** **import** model\_from\_json

**from** **keras.datasets** **import** cifar10

**from** **keras.models** **import** Sequential

**from** **keras.layers** **import** Dense, Flatten, Activation

**from** **keras.layers** **import** Dropout

**from** **keras.layers.convolutional** **import** Conv2D, MaxPooling2D

**from** **keras.utils** **import** np\_utils

**from** **keras.optimizers** **import** SGD

**import** **numpy** **as** **np**

**from** **PIL** **import** Image

json\_file = open("cifar\_model.json", "r")

loaded\_model\_json = json\_file.read()

json\_file.close()

loaded\_model = model\_from\_json(loaded\_model\_json)

loaded\_model.load\_weights("cifar\_model.h5")

loaded\_model.compile(loss='categorical\_crossentropy', optimizer='sgd', metrics=['accuracy'])

print ("Загрузили Model")

*# \*\*\*\*\* Загружаем изображение в Keras:*

**import** **numpy** **as** **np**

**from** **keras.preprocessing** **import** image

img\_path = 'plane.jpg'

img = image.load\_img(img\_path, target\_size=(32, 32))

*# В отличие от рукописных цифр, в этот раз изображение цветное и его размер 32х32, в соответствии с форматом CIFAR-10. Преобразуем картинку в массив numpy:*

x = image.img\_to\_array(img)

x /= 255

x = np.expand\_dims(x, axis=0)

*# Запускаем распознавание объекта:*

prediction = loaded\_model.predict(x)

*# Для удобства вывода задаем список с названиями классов объектов:*

classes=['самолет', 'автомобиль', 'птица', 'кот', 'олень', 'собака', 'лягушка', 'лошадь', 'корабль', 'грузовик']

*# Печатаем результат распознавания:*

print(classes[np.argmax(prediction)])

**Вывод**

Загрузили Model

Самолет

**import** **numpy**

**from** **keras.models** **import** model\_from\_json

**from** **keras.datasets** **import** cifar10

**from** **keras.models** **import** Sequential

**from** **keras.layers** **import** Dense, Flatten, Activation

**from** **keras.layers** **import** Dropout

**from** **keras.layers.convolutional** **import** Conv2D, MaxPooling2D

**from** **keras.utils** **import** np\_utils

**from** **keras.optimizers** **import** SGD

**import** **numpy** **as** **np**

**from** **PIL** **import** Image

json\_file = open("cifar\_model.json", "r")

loaded\_model\_json = json\_file.read()

json\_file.close()

loaded\_model = model\_from\_json(loaded\_model\_json)

loaded\_model.load\_weights("cifar\_model.h5")

loaded\_model.compile(loss='categorical\_crossentropy', optimizer='sgd', metrics=['accuracy'])

print ("Загрузили Model")

*# \*\*\*\*\* Загружаем изображение в Keras:*

**import** **numpy** **as** **np**

**from** **keras.preprocessing** **import** image

img\_path = 'iQUBU1QLS.jpg'

img = image.load\_img(img\_path, target\_size=(32, 32))

*# В отличие от рукописных цифр, в этот раз изображение цветное и его размер 32х32, в соответствии с форматом CIFAR-10. Преобразуем картинку в массив numpy:*

x = image.img\_to\_array(img)

x /= 255

x = np.expand\_dims(x, axis=0)

*# Запускаем распознавание объекта:*

prediction = loaded\_model.predict(x)

*# Для удобства вывода задаем список с названиями классов объектов:*

classes=['самолет', 'автомобиль', 'птица', 'кот', 'олень', 'собака', 'лягушка', 'лошадь', 'корабль', 'грузовик']

*# Печатаем результат распознавания:*

print(classes[np.argmax(prediction)])

**Вывод**

Загрузили Model

олень