# **9.1. Дообучение нейронной сети**

( <https://www.asozykin.ru/courses/nnpython>

<https://github.com/sozykin/dlpython_course/tree/master/transfer_learning>

<https://gggdomi.github.io/keras-workshop/notebook.html> )

Дообучение нейронной сети для новой

классификации (на 2 класса).

from keras.preprocessing import image

from keras.models import Model

from keras.layers import Dense, GlobalAveragePooling2D

from keras import applications

from keras.preprocessing.image import ImageDataGenerator

from keras import optimizers

from keras.models import Sequential

from keras.layers import Dropout, Flatten, Dense

from keras.models import Model

# Размер изображений

img\_width, img\_height = 150, 150

# Путь к каталогу с изображениями для обучения

train\_data\_dir = 'data/train'

# Путь к каталогу с изображениями для валидации

validation\_data\_dir = 'data/validation'

# Количество изображений для обучения

nb\_train\_samples = 2000

# Количество изображений для валидации

nb\_validation\_samples = 800

# Количество эпох

epochs = 5

# Размер выборки

batch\_size = 16

# Загружаем сеть VGG16 без части, которая отвечает за классификацию

base\_model = applications.VGG16(weights='imagenet', include\_top=False)

# Добавляем слои классификации

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dense(1024, activation='relu')(x)

# Выходной слой с двумя нейронами для классов "кот" и "собака"

predictions = Dense(2, activation='softmax')(x)

# Составляем сеть из двух частей

model = Model(inputs=base\_model.input, outputs=predictions)

# "Замораживаем" сверточные уровни сети VGG16

# Обучаем только вновь добавленные слои

for layer in base\_model.layers:

layer.trainable = False

# Компилируем модель

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['accuracy'])

# Создаем генератор данных для обучения

datagen = ImageDataGenerator(rescale=1. / 255)

train\_generator = datagen.flow\_from\_directory(

train\_data\_dir,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode= 'categorical')

*Found 2000 images belonging to 2 classes.*

# Создаем генератор данных для валидации

validation\_generator = datagen.flow\_from\_directory(

validation\_data\_dir,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode= 'categorical')

*Found 800 images belonging to 2 classes.*

# Обучаем модель с помощью генератора

model.fit\_generator(

train\_generator,

steps\_per\_epoch=nb\_train\_samples,

epochs=epochs,

validation\_data=validation\_generator,

validation\_steps = nb\_validation\_samples)

# Сохраняем нейронную сеть

**Примечание:** Обратите внимание на график зависимости точности на тестовой выборке от количества эпох обучения: <https://gggdomi.github.io/keras-workshop/notebook.htm>

**Структура папки изображений для обучения и тестирования:**

(количество изображений увеличивается за счет случайных преобразований: <https://gggdomi.github.io/keras-workshop/notebook.htm> )

data/

train/

dogs/ ### pictures

dog001.jpg

dog002.jpg

...

cats/ ### pictures

cat001.jpg

cat002.jpg

...

validation/

dogs/ ### pictures

dog001.jpg

dog002.jpg

...

cats/ ### pictures

cat001.jpg

cat002.jpg

...

**Примечание:** Попытайтесь загрузить предварительно подготовленные данные: <https://www.kaggle.com/c/dogs-vs-cats/data>

Использование дообученной сети

для новой классификации.

import numpy as np

from keras.models import model\_from\_json

from keras.preprocessing import image

from keras.applications.vgg16 import preprocess\_input

import matplotlib.pyplot as plt

from scipy.misc import toimage

%matplotlib inline

# Список классов

classes = ['кот', 'собака']

# Загружаем обученную модель

json\_file = open("vgg16\_cat\_dogs.json", "r")

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

json\_file.close()

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

loaded\_model.load\_weights("vgg16\_cat\_dogs.h5")

# Компилируем модель

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

# Загружаем изображение для распознавания

img = image.load\_img('1.jpg', target\_size=(224, 224))

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

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

x = preprocess\_input(x)

# Запускаем распознавание

prediction = loaded\_model.predict(x)

print(prediction)

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