Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



**«Разработка нейросетевых систем»**

**Домашнее задание №1**

**ИСПОЛНИТЕЛЬ:**

Демирев Н.К.

Группа ИУ5-21М

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2023 г.

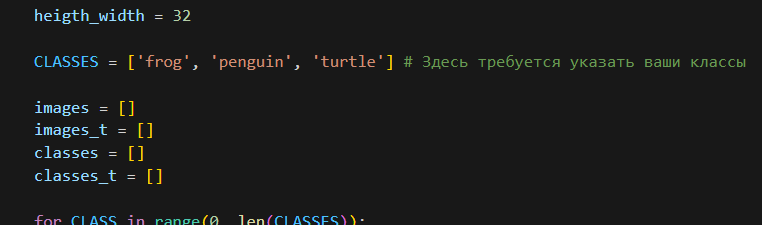
Москва 2023

# Задание

Необходимо создать и разметить собственный набор данных, состоящий из изображений. Набор содержит не менее 3 классов и не менее 100 экземпляров каждый. Изображения можно скачать из интернета или объединить несколько существующих датасетов. Создать web-приложение для классификации изображений полученного набора данных. Использовать аугментацию данных, регуляризацию, перенос обучения.

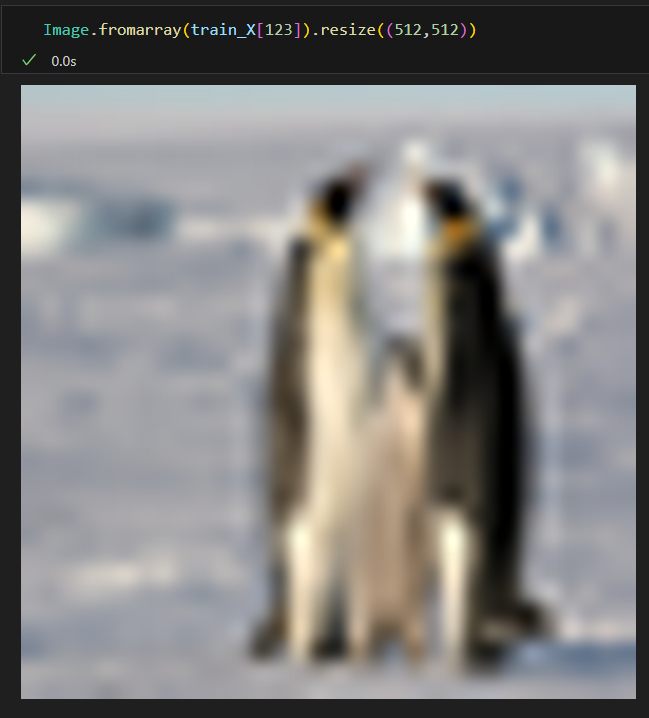
# Выполнение

Имеем 3 класса: лягушка, пингвин, черепаха (океаническая)



Ресайз изображений:

Изначально изображения были в разрешении 5419x3613 px



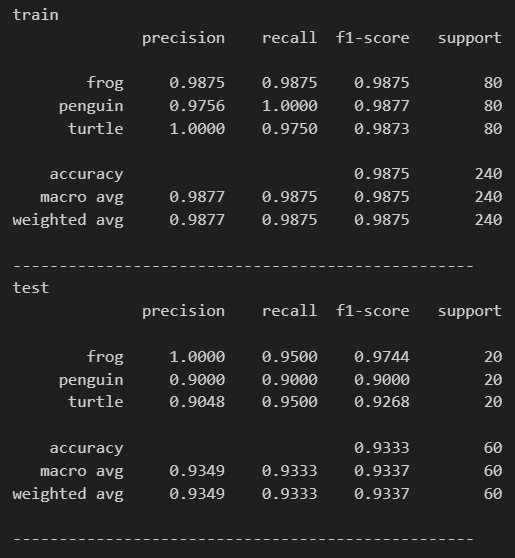
Аугментация изображений:



Этап обучения:

Эпох = 250

Скорость обучения = 0.001



Необходимые настройки сервера:

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [

            BASE\_DIR / "templates"

        ],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

imageClassList = {'0': 'frog', '1': 'penguin', '2': 'turtle'}

def scoreImagePage(request):

    return render(request, 'scorepage.html')

def predictImage(request):

    fileObj = request.FILES['filePath']

    fs = FileSystemStorage()

    filePathName = fs.save('images/'+fileObj.name,fileObj)

    filePathName = fs.url(filePathName)

    modelName = request.POST.get('modelName')

    scorePrediction = predictImageData(modelName, '.'+filePathName)

    context = {'scorePrediction': scorePrediction}

    return render(request, 'scorepage.html', context)

def predictImageData(modelName, filePath):

    img = Image.open(filePath).convert("RGB")

    img = np.asarray(img.resize((32, 32), Image.ANTIALIAS))

    sess = onnxruntime.InferenceSession(r'G:\repos\Deep-learning\dz1\myModel.onnx')

    outputOFModel = np.argmax(sess.run(None, {'input': np.asarray([img]).astype(np.float32)}))

    score = imageClassList[str(outputOFModel)]

    return score

Результат:

