Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК5 «Системы обработки информации»

ДОМАШНЯЯ РАБОТА №1

«ГЛУБОКАЯ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНАЯ СЕТЬ GAN»

по дисциплине: «Методы глубокого обучения»

Выполнил: студент группы ИУК5-21М _____ А. Э. Дармограй
(Подпись) (И.О. Фамилия)

Проверил: _____ Ю. С. Белов
(Подпись) (И.О. Фамилия)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель работы:

Целью выполнения домашней работы является получение практических навыков построения генеративно-состязательной сети.

Задачи:

1. Изучить 2 сети GAN: сеть-генератор и сеть-дискриминатор
2. Разработать модель генеративно-состязательной сети

Задание на домашнюю работу:

На основе разобранного примера реализуйте генеративно-состязательную сеть. В соответствие с выданным вариантом используйте как объект для реализации:

Вариант 2.

2. Изображение животных.

Выполнение работы

Код доступен в репозитории GitHub:

https://github.com/Dariarty/Deep_Learning_Methods

Данную работу выполнял на Python версии 3.9.13 и Tensorflow версии 2.7.0

Код домашней работы №1:

https://github.com/Dariarty/Deep_Learning_Methods/blob/main/src/DR_1/gan.ipynb

В данной работе разарбатываем модель генеративно-состязательной сети (GAN)

В начале я убеждаюсь, что Tensorflow использует GPU. Для этого предварительно установил CUDA и cuDNN, позволяющие использовать Tensorflow на видеокартах Nvidia. Используется видеокарта Nvidia RTX 3060.

```
#В данной работе использую Python 3.9.13 и tensorflow 2.7.0

import sys
import tensorflow as tf
from tensorflow import keras

# Вывод версий Python и Tensorflow
print("Python", sys.version)
print("Tensorflow", tf.__version__)

# Убеждаюсь, что tensorflow использует GPU
available_gpus = tf.config.list_physical_devices('GPU') # Динамическое использование памяти GPU
if available_gpus:
    try:
        for gpu in available_gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        print("Tensorflow uses GPU")
    except RuntimeError as error:
        print("GPU Error:", error)
```

```
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)]
Tensorflow 2.7.0
Tensorflow uses GPU
```

Генератор

Начинаем с модели generator, которая преобразует вектор (из скрытого пространства, полученного во время обучения, который будет выбираться случайно) в изображение-кандидат

```
import numpy as np
latent_dim = 32
height = 32
width = 32
channels = 3
generator_input = keras.Input(shape=(latent_dim,))
x = keras.layers.Dense(128 * 16 * 16)(generator_input)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Reshape((16, 16, 128))(x)
x = keras.layers.Conv2D(256, 5, padding='same')(x)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Conv2DTranspose(256, 4, strides=2, padding='same')(x)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Conv2D(256, 5, padding='same')(x)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Conv2D(256, 5, padding='same')(x)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Conv2D(channels, 7, activation='tanh',
padding='same')(x)
generator = keras.models.Model(generator_input, x)
generator.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 32)]              0

 dense (Dense)               (None, 32768)             1081344

 leaky_re_lu (LeakyReLU)     (None, 32768)             0

 reshape (Reshape)           (None, 16, 16, 128)       0

 conv2d (Conv2D)             (None, 16, 16, 256)       819456

 leaky_re_lu_1 (LeakyReLU)   (None, 16, 16, 256)       0

 conv2d_transpose (Conv2DTra  (None, 32, 32, 256)      1048832
 nspose)

 leaky_re_lu_2 (LeakyReLU)   (None, 32, 32, 256)       0

 conv2d_1 (Conv2D)           (None, 32, 32, 256)       1638656

 leaky_re_lu_3 (LeakyReLU)   (None, 32, 32, 256)       0

 conv2d_2 (Conv2D)           (None, 32, 32, 256)       1638656

 leaky_re_lu_4 (LeakyReLU)   (None, 32, 32, 256)       0

 conv2d_3 (Conv2D)           (None, 32, 32, 3)         37635

=================================================================
Total params: 6,264,579
Trainable params: 6,264,579
Non-trainable params: 0
_________________________________________________________________
```

Дискриминатор

Модель discriminator принимает на входе изображение-кандидат (реальное или искусственное) и относит его к одному из двух классов: «подделка» или «настоящее, имеющееся в обучающем наборе».

```
discriminator_input = keras.layers.Input(shape=(height, width, channels))
x = keras.layers.Conv2D(128, 3)(discriminator_input)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Conv2D(128, 4, strides=2)(x)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Conv2D(128, 4, strides=2)(x)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Conv2D(128, 4, strides=2)(x)
x = keras.layers.LeakyReLU()(x)
x = keras.layers.Flatten()(x)
x = keras.layers.Dropout(0.4)(x)
x = keras.layers.Dense(1, activation='sigmoid')(x)
discriminator = keras.models.Model(discriminator_input, x)
discriminator.summary()
discriminator_optimizer = keras.optimizers.RMSprop(learning_rate=0.0008,
clipvalue=1.0, decay=1e-8)
discriminator.compile(optimizer=discriminator_optimizer,
loss='binary_crossentropy')
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 32, 32, 3)]       0

 conv2d_4 (Conv2D)           (None, 30, 30, 128)       3584

 leaky_re_lu_5 (LeakyReLU)   (None, 30, 30, 128)       0

 conv2d_5 (Conv2D)           (None, 14, 14, 128)       262272

 leaky_re_lu_6 (LeakyReLU)   (None, 14, 14, 128)       0

 conv2d_6 (Conv2D)           (None, 6, 6, 128)         262272

 leaky_re_lu_7 (LeakyReLU)   (None, 6, 6, 128)         0

 conv2d_7 (Conv2D)           (None, 2, 2, 128)         262272

 leaky_re_lu_8 (LeakyReLU)   (None, 2, 2, 128)         0

 flatten (Flatten)           (None, 512)               0

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 790,913
Trainable params: 790,913
Non-trainable params: 0
_________________________________________________________________
```

Состязательная сеть

Состязательная сети объединяет генератор и дискриминатор. Обучение gan будет смещать веса в модели generator так, чтобы увеличить вероятность получить от дискриминатора ответ «настоящее», когда тот будет просматривать поддельное изображение.

```
discriminator.trainable = False
gan_input = keras.Input(shape=(latent_dim,))
gan_output = discriminator(generator(gan_input))
gan = keras.models.Model(gan_input, gan_output)
gan_optimizer = keras.optimizers.RMSprop(learning_rate=0.0004,
clipvalue=1.0, decay=1e-8)
gan.compile(optimizer=gan_optimizer, loss='binary_crossentropy')
```

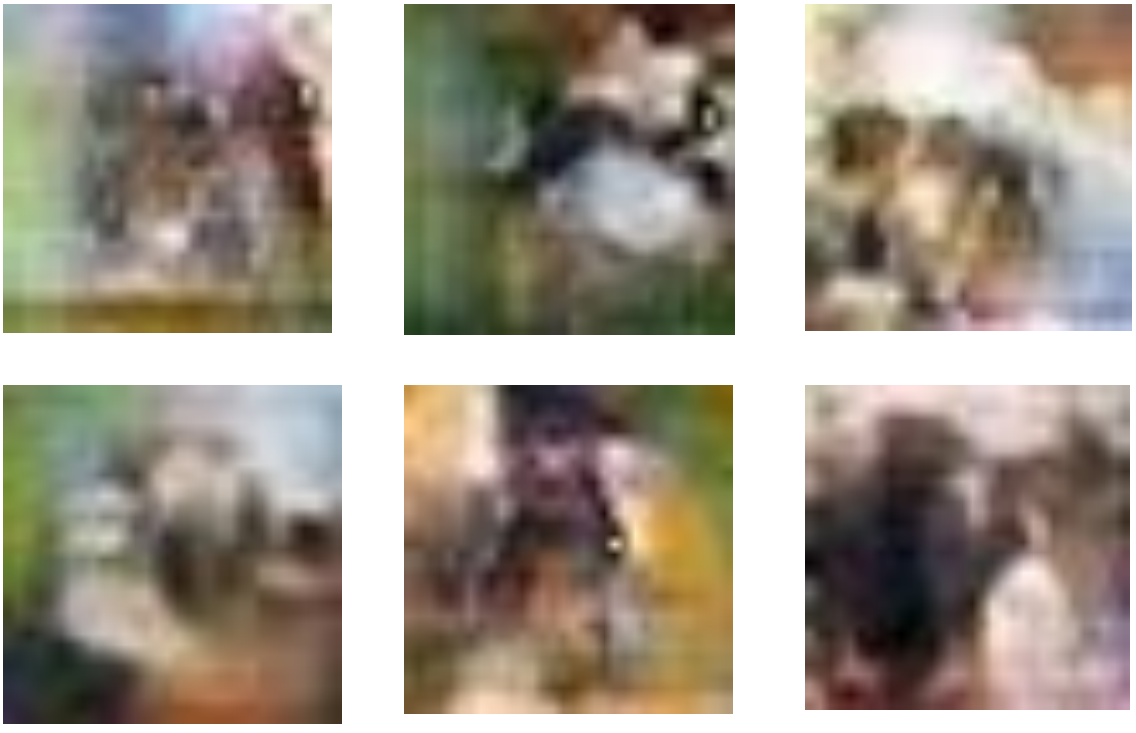
Обучение сети DCGAN

Используются изображения собак. Настоящие и сгенерированные изображения сохраняются через каждые 100 итераций.

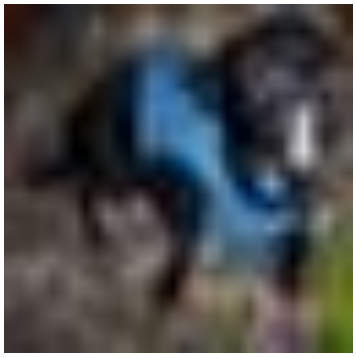
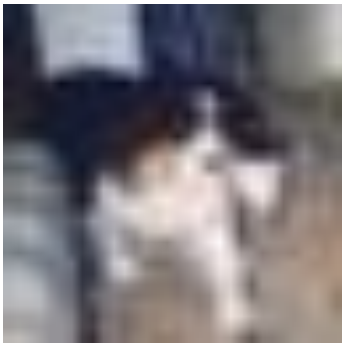
```
import os

(x_train, y_train), (_, _) = keras.datasets.cifar10.load_data()
x_train = x_train[y_train.flatten() == 5] # Изображения собак
x_train = x_train.reshape((x_train.shape[0], height, width,
channels)).astype('float32') / 255.

iterations = 10000
batch_size = 20
save_dir = 'generated_images'
os.makedirs(save_dir, exist_ok=True)

start = 0

for step in range(iterations + 1):
    # Генерация случайного шума
    random_latent_vectors = np.random.normal(size=(batch_size,
latent_dim))

    # Генерация изображений
    generated_images = generator.predict(random_latent_vectors)

    # Подготовка данных: настоящие + сгенерированные изображения
    stop = start + batch_size
    real_images = x_train[start: stop]
    combined_images = np.concatenate([generated_images, real_images])

    # Метки: 1 — подделка, 0 — настоящее
    labels = np.concatenate([np.ones((batch_size, 1)),
np.zeros((batch_size, 1))])
    labels += 0.05 * np.random.random(labels.shape)  # шум на метках для
устойчивости

    # Обучение дискриминатора
    d_loss = discriminator.train_on_batch(combined_images, labels)
```

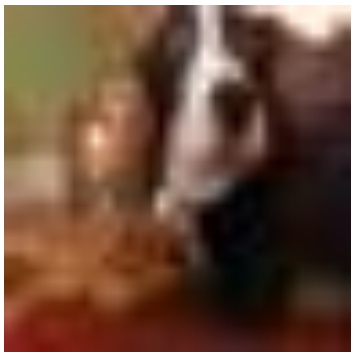
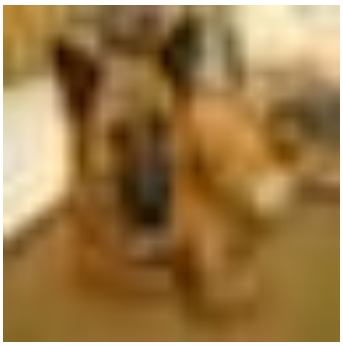
```
    # Новые случайные точки и обучение генератора через GAN
    random_latent_vectors = np.random.normal(size=(batch_size,
latent_dim))
    misleading_targets = np.zeros((batch_size, 1))  # генератор хочет,
чтобы дискриминатор сказал "настоящее"
    a_loss = gan.train_on_batch(random_latent_vectors,
misleading_targets)

    start += batch_size
    if start > len(x_train) - batch_size:
        start = 0

    # Каждые 100 итераций сохранение модели и изображений
    if step % 100 == 0:
        gan.save_weights('gan.h5')
        print('discriminator loss:', d_loss)
        print('adversarial loss:', a_loss)

        # Сохраняем сгенерированное изображение
        img = keras.preprocessing.image.array_to_img(generated_images[0]
* 255., scale=False)
        img.save(os.path.join(save_dir,
f'generated_image_step_{step}.png'))

        # Сохраняем настоящее изображение
        img = keras.preprocessing.image.array_to_img(real_images[0] *
255., scale=False)
        img.save(os.path.join(save_dir, f'real_image_step_{step}.png'))
```

Примеры сгенерированных изображений:



Примеры реальных изображений:

Вывод: в ходе выполнения лабораторной работы были сформированы практические навыки по построению генеративно-состязательной сети. Были изучены сети GAN: сеть-генератор и сеть-дискриминатор, а также разработана модель генеративно-состязательной сети.