|  |  |
| --- | --- |
|  | **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)]

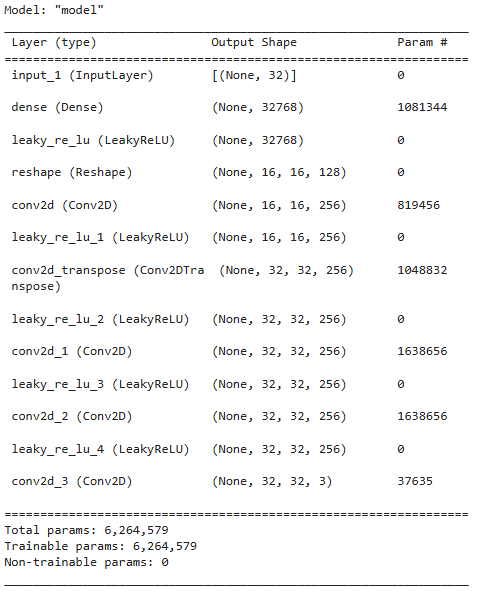
Tensorflow 2.7.0

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

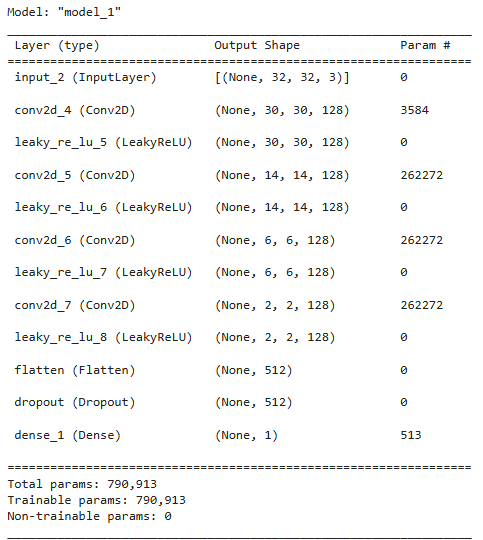
|  |
| --- |
| **import** numpy **as** np  latent\_dim **=** 32  height **=** 32  width **=** 32  channels **=** 3  generator\_input **=** keras**.**Input(shape**=**(latent\_dim,))  x **=** keras**.**layers**.**Dense(128 **\*** 16 **\*** 16)(generator\_input)  x **=** keras**.**layers**.**LeakyReLU()(x)  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() |



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') |



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.