



МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«МИРЭА – Российский технологический университет»  
**РТУ МИРЭА**

---

Институт кибербезопасности и цифровых технологий  
Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

---

## **Отчет по практике**

По дисциплине

«Анализ защищенности систем искусственного интеллекта»

Тема: «Практика 7: Создание и использование генеративных  
противоречивых примеров (GAN- based Adversarial Examples)

»

Студент Макаров Павел Андреевич  
Группа БМО-02-23

Работу проверил  
Спирин А.А.

Москва, 2024


```

import tensorflow as tf
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt
# Загрузка данных MNIST
(train_images, _), (_, _) = tf.keras.datasets.mnist.load_data()
train_images = train_images / 255.0
# Добавление одного измерения (для работы CNN)
train_images = np.expand_dims(train_images, axis=-1)
# Генератор
def build_generator():
    model = tf.keras.Sequential()
    model.add(layers.Dense(128, activation='relu', input_dim=100))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())
    model.add(layers.Dense(784, activation='tanh'))
    model.add(layers.Reshape((28, 28, 1)))
    return model
# Дискриминатор
def build_discriminator():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (3, 3), padding='same', input_shape=(28, 28, 1)))
    model.add(layers.LeakyReLU())
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(1, activation='sigmoid'))
    return model
# Создание моделей
generator = build_generator()
discriminator = build_discriminator()
# Компиляция дискриминатора
discriminator.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# Создание GAN
gan_input = layers.Input(shape=(100,))
generated_image = generator(gan_input)
discriminator.trainable = False
validity = discriminator(generated_image)
gan = tf.keras.Model(gan_input, validity)
gan.compile(optimizer='adam', loss='binary_crossentropy')

```

```
# Функция обучения GAN
def train_gan(generator, discriminator, gan, epochs=100, batch_size=64):
    half_batch = batch_size // 2
    for epoch in range(epochs):
        # Обучение дискриминатора
        idx = np.random.randint(0, train_images.shape[0], half_batch)
        real_images = train_images[idx]
        noise = np.random.normal(0, 1, (half_batch, 100))
        fake_images = generator.predict(noise)
        real_labels = np.ones((half_batch, 1))
        fake_labels = np.zeros((half_batch, 1))
        d_loss_real = discriminator.train_on_batch(real_images, real_labels)
        d_loss_fake = discriminator.train_on_batch(fake_images, fake_labels)
        # Обучение генератора через дискриминатор
        noise = np.random.normal(0, 1, (batch_size, 100))
        valid_labels = np.ones((batch_size, 1))
        g_loss = gan.train_on_batch(noise, valid_labels)
        if epoch % 1000 == 0:
            print(f'{epoch} [D loss: {0.5 * np.add(d_loss_real, d_loss_fake)}] [G loss: {g_loss}]')
# Обучение GAN
train_gan(generator, discriminator, gan)

1/1 ----- 0s 67ms/step
0 [D loss: [0.7249831 0.2109375]] [G loss: [array(0.74450785, dtype=float32), array(0.74450785, dtype=float32), array(0.140625, dtype=float32)]]
1/1 ----- 0s 20ms/step
1/1 ----- 0s 22ms/step
1/1 ----- 0s 19ms/step
1/1 ----- 0s 21ms/step
1/1 ----- 0s 19ms/step
1/1 ----- 0s 22ms/step
1/1 ----- 0s 29ms/step
1/1 ----- 0s 31ms/step
# Генерация противоречивых примеров
def generate_adversarial_examples(generator, n_samples):
    noise = np.random.normal(0, 1, (n_samples, 100))
    generated_images = generator.predict(noise)
    return generated_images
# Генерация 100 примеров
adversarial_images = generate_adversarial_examples(generator, 100)
# Визуализация противоречивых примеров
plt.figure(figsize=(10, 10))
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(adversarial_images[i].reshape(28, 28), cmap='gray')
    plt.axis('off')
plt.show()

4/4 ----- 0s 25ms/step


# Загрузка обученных моделей
model1 = tf.keras.models.load_model('mnist_model1.h5')
model2 = tf.keras.models.load_model('mnist_model2.h5')
# Оценка первой модели на противоречивых примерах
adv_images_resized = adversarial_images.reshape(-1, 28, 28, 1)
loss1, acc1 = model1.evaluate(adv_images_resized, np.ones((100, 10))) #Примерные метки
print(f'Accuracy of model1 on adversarial GAN examples: {acc1}')
# Оценка второй модели на противоречивых примерах
loss2, acc2 = model2.evaluate(adv_images_resized, np.ones((100, 10))) #Примерные метки
print(f'Accuracy of model2 on adversarial GAN examples: {acc2}')

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. 'model.compile_metrics' will be empty until you train or evaluate the model.
4/4 ----- 0s 7ms/step - accuracy: 0.0000e+00 - loss: 193.5952
Accuracy of model1 on adversarial GAN examples: 0.0
4/4 ----- 0s 6ms/step - accuracy: 0.0000e+00 - loss: 391.9700
Accuracy of model2 on adversarial GAN examples: 0.0
```