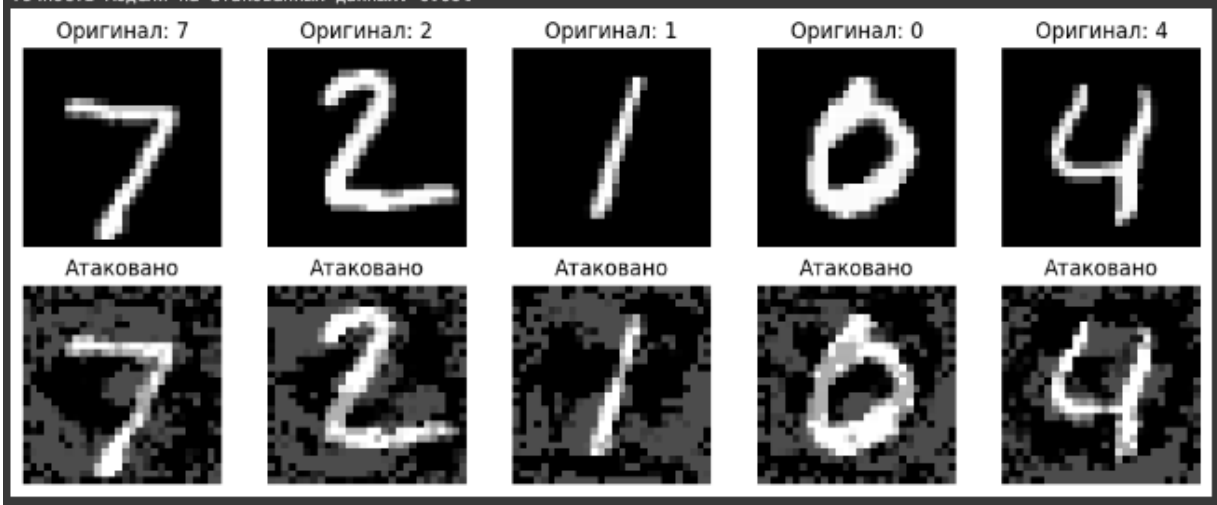
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Загрузка и подготовка данных
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.astype('float32') / 255.0
x_train = x_train.reshape(-1, 28, 28, 1)
y_train = to_categorical(y_train, 10)

# Создание модели
model = Sequential([
    Flatten(input_shape=(28, 28, 1)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Компиляция модели
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Обучение модели
model.fit(x_train, y_train, epochs=5, batch_size=32, validation_split=0.1)

# Сохранение модели
model.save('mnist_model.h5')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the fi
  super().__init__(**kwargs)
Epoch 1/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.8720 - loss: 0.4488 - val_accuracy: 0.9685 - val_loss: 0.1154
Epoch 2/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9625 - loss: 0.1253 - val_accuracy: 0.9745 - val_loss: 0.0924
Epoch 3/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9766 - loss: 0.0799 - val_accuracy: 0.9750 - val_loss: 0.0845
Epoch 4/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9813 - loss: 0.0598 - val_accuracy: 0.9782 - val_loss: 0.0774
Epoch 5/5
1688/1688 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step - accuracy: 0.9878 - loss: 0.0430 - val_accuracy: 0.9780 - val_loss: 0.0716
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras
```

```python
import os
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.datasets import mnist
import numpy as np
import foolbox as fb
import matplotlib.pyplot as plt

# 1. Загрузка обученной модели
model_path = 'mnist_model.h5'

if not os.path.exists(model_path):
    raise FileNotFoundError(f"Файл модели '{model_path}' не найден. Пожалуйста, убедитесь, что путь указан правильно.")

model = load_model(model_path)
print("Модель успешно загружена.")

# 2. Загрузка и предобработка данных
(_, _), (x_test, y_test) = mnist.load_data()
x_test = x_test.astype('float32') / 255.0
x_test = np.expand_dims(x_test, axis=-1)  # Форма: (num_samples, 28, 28, 1)

# Преобразование меток из uint8 в int64
y_test = y_test.astype(np.int64)

# Преобразование в TensorFlow Tensors
x_test_tf = tf.convert_to_tensor(x_test, dtype=tf.float32)
y_test_tf = tf.convert_to_tensor(y_test, dtype=tf.int64)

# 3. Инициализация модели Foolbox
fmodel = fb.TensorFlowModel(model, bounds=(0, 1))
print("Foolbox модель инициализирована.")

# 4. Оценка точности на чистых данных
def evaluate_clean_accuracy(model, x, y):
    predictions = model.predict(x)
    predicted_labels = np.argmax(predictions, axis=1)
    accuracy = np.mean(predicted_labels == y.numpy())
    return accuracy

clean_accuracy = evaluate_clean_accuracy(model, x_test_tf, y_test_tf)
print(f"Точность модели на чистых данных: {clean_accuracy * 100:.2f}%")

# 5. Создание атаки PGD
attack = fb.attacks.LinfPGD()
epsilon = 0.3  # Уровень шума для атаки
```

```python
# 6. Генерация противоречивых примеров
print("Генерация противоречивых примеров с помощью PGD атаки...")
try:
    adversarials, clipped, is_adv = attack(fmodel, x_test_tf, y_test_tf, epsilons=epsilon)
    print("Генерация завершена.")
except tf.errors.InvalidArgumentError as e:
    print("Произошла ошибка при генерации атак:", e)
    print("Убедитесь, что метки имеют тип int32 или int64.")
    raise

# Пример визуализации:
def visualize_adversarials(x, adversarials, y, is_adv, num=5):
    plt.figure(figsize=(10, 4))
    successful_indices = np.where(is_adv.numpy())[0]
    for i in range(min(num, len(successful_indices))):
        idx = successful_indices[i]

        # Оригинальное изображение
        plt.subplot(2, num, i + 1)
        plt.imshow(x[idx].numpy().squeeze(), cmap="gray")
        plt.title(f"Оригинал: {y[idx].numpy()}")
        plt.axis('off')

        # Атакованное изображение
        plt.subplot(2, num, num + i + 1)
        plt.imshow(adversarials[idx].numpy().squeeze(), cmap="gray")
        plt.title("Атаковано")
        plt.axis('off')
    plt.tight_layout()
    plt.show()

def evaluate_adversarial_accuracy(model, adversarials, y, is_adv):
    # Используем только атакованные примеры
    adversarial_examples = adversarials[is_adv]
    adversarial_labels = y[is_adv]
    predictions = model.predict(adversarial_examples)
    predicted_labels = np.argmax(predictions, axis=1)
    accuracy = np.mean(predicted_labels == adversarial_labels)
    return accuracy

adversarial_accuracy = evaluate_adversarial_accuracy(model, adversarials, y_test, is_adv)
print(f"Точность модели на атакованных данных: {adversarial_accuracy * 100:.2f}%")
# Визуализируем первые 5 успешных атак
num_successful = np.sum(is_adv.numpy())
if num_successful >= 5:
    visualize_adversarials(x_test_tf, adversarials, y_test_tf, is_adv, num=5)
else:
    print("Недостаточно успешных атак для визуализации.")
print("\nДополнительная статистика:")
print(f"Общая точность на чистых данных: {clean_accuracy * 100:.2f}%")
print(f"Точность на атакованных данных: {adversarial_accuracy * 100:.2f}%")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be
Модель успешно загружена.
Foolbox модель инициализирована.
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
Точность модели на чистых данных: 97.48%
Генерация противоречивых примеров с помощью PGD атаки...
Генерация завершена.
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Точность модели на атакованных данных: 0.00%
```




```
Дополнительная статистика:
Общая точность на чистых данных: 97.48%
Точность на атакованных данных: 0.00%
```