Московский Государственный Технический Университет имени Н. Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчёт по лабораторной работе № 4 по дисциплине «Проектирование интеллектуальных систем»

Исполнитель:
Саврасов П.А.
Группа ИУ5-24М
«__» _____ 2021 г.

Преподаватель:
Терехов В.И.
_____
«__» _____ 2021 г.

Москва, 2021 г.

1. Цель работы

Освоить методы сохранения обученной модели. Применить визуализацию процесса обучения нейронной сети через Tensorboard.

2. Подготовка модели

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.python.keras.datasets import cifar10
```

```
(xTrain, yTrain), (xTest, yTest) = cifar10.load_data()
print(xTrain.shape, yTrain.shape)
print(xTest.shape, yTest.shape)
```

```
(50000, 32, 32, 3) (50000, 1)
(10000, 32, 32, 3) (10000, 1)
```

```
model = keras.Sequential(
    [
        layers.InputLayer(input_shape=(32, 32, 3)),
        layers.Conv2D(32, 3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, 3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(128, 3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(10)
    ]
)
model.compile(
    loss = keras.losses.SparseCategoricalCrossentropy(from_logits = True),
    optimizer = keras.optimizers.Adam(learning_rate = 0.001),
    metrics = ['accuracy']
)
```

```
tensorboardCallback = keras.callbacks.TensorBoard(
    log_dir='tbCallbaclk',
    histogram_freq=1
)
```

```
model.fit(
    xTrain,
    yTrain,
    batch_size=128,
    epochs=10,
    verbose=1,
    callbacks=[tensorboardCallback]
)
model.evaluate(xTest, yTest, batch_size=128, verbose=2)
```

```
Epoch 1/10
391/391 [==============================] - 14s 34ms/step - loss: 2.0077 - accuracy: 0.3553
Epoch 2/10
391/391 [==============================] - 13s 33ms/step - loss: 1.4027 - accuracy: 0.4954
Epoch 3/10
391/391 [==============================] - 13s 34ms/step - loss: 1.2560 - accuracy: 0.5550
Epoch 4/10
391/391 [==============================] - 13s 34ms/step - loss: 1.1535 - accuracy: 0.5925
Epoch 5/10
391/391 [==============================] - 13s 33ms/step - loss: 1.0646 - accuracy: 0.6255
Epoch 6/10
391/391 [==============================] - 13s 33ms/step - loss: 0.9929 - accuracy: 0.6496
Epoch 7/10
391/391 [==============================] - 13s 33ms/step - loss: 0.9437 - accuracy: 0.6684
Epoch 8/10
391/391 [==============================] - 13s 33ms/step - loss: 0.8798 - accuracy: 0.6925
Epoch 9/10
391/391 [==============================] - 13s 34ms/step - loss: 0.8410 - accuracy: 0.7053
Epoch 10/10
391/391 [==============================] - 13s 33ms/step - loss: 0.8007 - accuracy: 0.7188
79/79 - 1s - loss: 0.9404 - accuracy: 0.6812

[0.9404182434082031, 0.6812000274658203]
```

3. Сохранение модели

Сохранение и загрузка весов

```
model.save_weights('modelFolder/')
```

```
model.load_weights('modelFolder/')
model.evaluate(xTest, yTest, batch_size=128, verbose=2)
```

```
79/79 - 1s - loss: 1.1108 - accuracy: 0.6670

[1.110833764076233, 0.6669999957084656]
```

Сохранение и загрузка модели целиком

```
model.save('fullModelFolder/')
```

```
INFO:tensorflow:Assets written to: fullModelFolder/assets
```

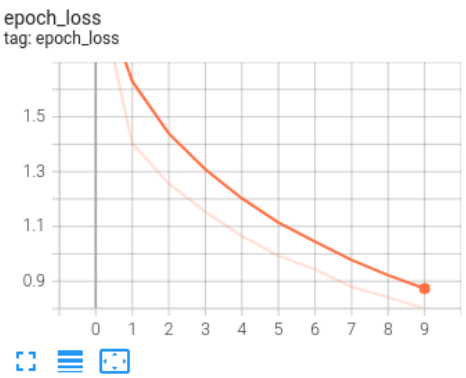
```
model2 = keras.models.load_model('fullModelFolder/')
model2.evaluate(xTest, yTest, batch_size=128, verbose=2)
```

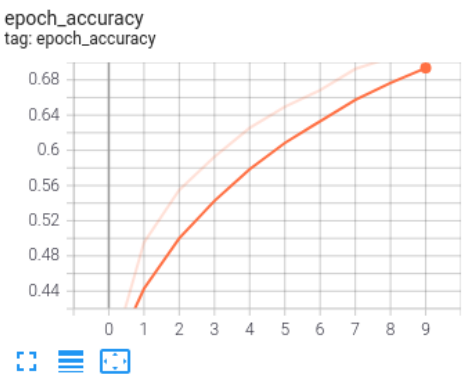
```
79/79 - 1s - loss: 1.1108 - accuracy: 0.6670

[1.110833764076233, 0.6669999957084656]
```

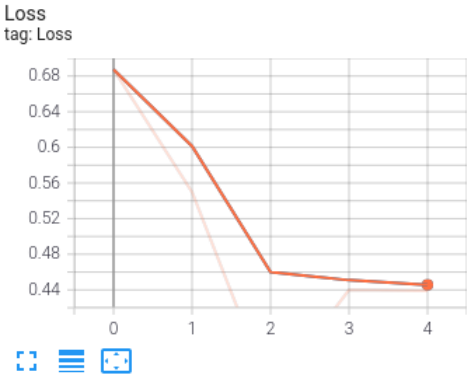
4. Визуализация процесса обучения через колбеки

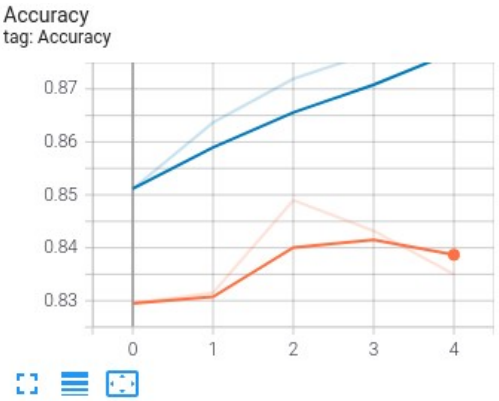
epoch_loss | epoch_accuracy





5. Визуализация процесса обучения через самописный обучающий цикл





```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.python.keras.datasets import cifar10
```

```
(xTrain, yTrain), (xTest, yTest) = cifar10.load_data()
print(xTrain.shape, yTrain.shape)
print(xTest.shape, yTest.shape)
```

```
(50000, 32, 32, 3) (50000, 1)
(10000, 32, 32, 3) (10000, 1)
```

```
model = keras.Sequential(
    [
        layers.InputLayer(input_shape=(32, 32, 3)),
        layers.Conv2D(32, 3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, 3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(128, 3, activation='relu'),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dense(10)
    ]
)

epochsNum = 5
optimizer = keras.optimizers.Adam()
lossFn = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metricAcc = keras.metrics.SparseCategoricalAccuracy()

trainWriter = tf.summary.create_file_writer('log/train/')
testWriter = tf.summary.create_file_writer('log/test/')
trainStep = testStep = 0
batchSize = 100
```

```
for epoch in range(epochsNum):
    print(f'Epoch № {epoch} -----------------------------------')

    # Training loop
    for i in range(len(xTrain) // batchSize):
        xBatch = xTrain[i * batchSize : (i + 1) * batchSize]
        yBatch = yTrain[i * batchSize : (i + 1) * batchSize]
        with tf.GradientTape() as tape:
            yPred = model(xBatch, training=True)
            loss = lossFn(yBatch, yPred)

        gradients = tape.gradient(loss, model.trainable_weights)
        optimizer.apply_gradients(zip(gradients, model.trainable_weights))
        metricAcc.update_state(yBatch, yPred)

    with trainWriter.as_default():
        tf.summary.scalar("Loss", loss, step=epoch)
        tf.summary.scalar('Accuracy', metricAcc.result(), step=epoch)

    print('Train Accuracy: ', metricAcc.result())
    metricAcc.reset_states()

    # Test loop
    for i in range(len(xTest) // batchSize):
        xBatch = xTrain[i * batchSize : (i + 1) * batchSize]
        yBatch = yTrain[i * batchSize : (i + 1) * batchSize]
        yPred = model(xBatch, training=False)
        metricAcc.update_state(yBatch, yPred)

    with testWriter.as_default():
        tf.summary.scalar("Loss", loss, step=epoch)
        tf.summary.scalar('Accuracy', metricAcc.result(), step=epoch)


    print('Test Accuracy: ', metricAcc.result(), '\n')
    metricAcc.reset_states()
```

6. Выводы по работе:

Освоил методы сохранения обученной модели. Применил визуализацию процесса обучения нейронной сети через Tensorboard.

7. Контрольные вопросы:

7.1. Как включить TensorBoard?

При помощи TensorBoard просматривать результаты можно и во время обучения. Для этого в терминале необходимо запустить команду: tensorboard −−logdir=path/to/log−directory

7.2. Как сбросить граф?

С помощью команды tf.reset_default_graph().

7.3. Зачем нужны коллекции?

Коллекция - это объект, в котором мы храним элементы узлов графа.

7.4. Перечислите команды для добавления переменных в сводную статистику.

tf.summary.scalars()
tf.summary.histogram()
tf.summary.merge_all()