HW2 Мельчук А.Б.

Реализовать и обучить *(с нуля)* CHC для задачи классификации изображений на датасете *CIFAR-10*
Библиотеки*: [Python, Tensorflow]*

Переключение версии TensorFlow

In [1]:

```
%tensorflow_version 2.x
```

```
UsageError: Line magic function `%tensorflow_version` not found.
```

In [3]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np

import tensorflow as tf
```

In [4]:

```
tf.__version__
```

Out[4]:

```
'2.1.0'
```

Загрузка и подготовка датасета cifar10

In [5]:

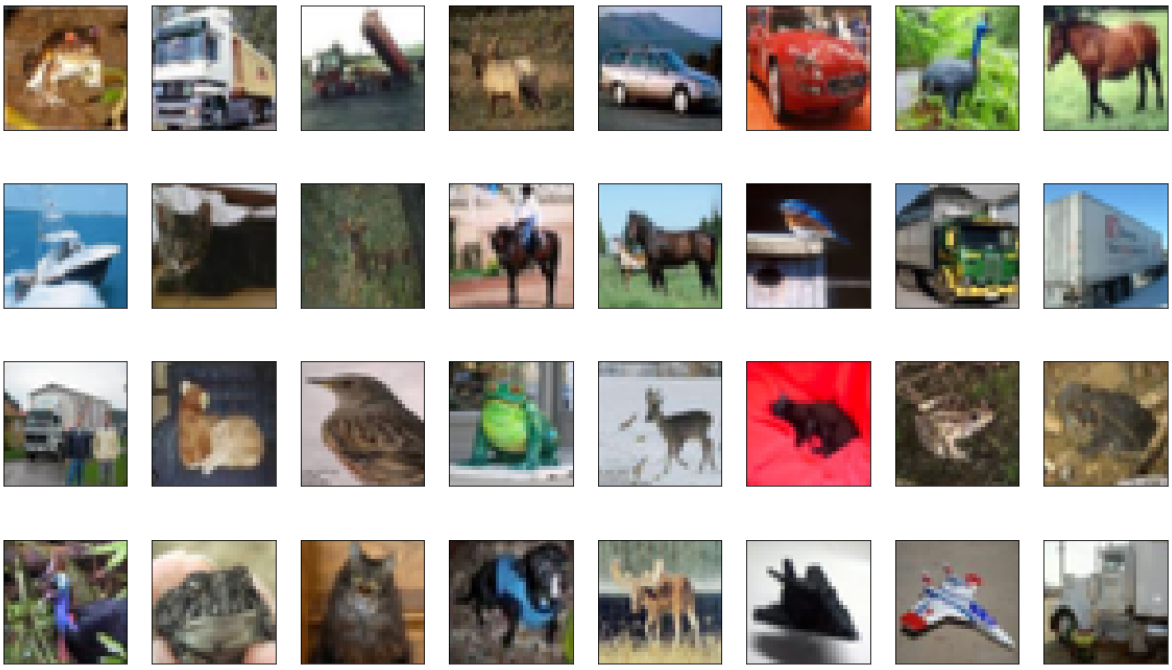
```
(train_x, train_y), (test_x, test_y) = tf.keras.datasets.cifar10.load_data()

train_x = train_x.astype(np.float32) / 255.
test_x = test_x.astype(np.float32) / 255.

train_y = train_y.astype(np.int32).flatten()
test_y = test_y.astype(np.int32).flatten()

print(train_x.shape, train_x.dtype)
print(test_x.shape, test_x.dtype)
print(train_y.shape, train_y.dtype)
print(test_y.shape, test_y.dtype)
```

```
(50000, 32, 32, 3) float32
(10000, 32, 32, 3) float32
(50000,) int32
(10000,) int32
```

Визуализация датасета cifar10

In [6]:

```
some_samples = train_x[:32, ...]

fig = plt.figure(figsize=(20, 12))
for j in range(some_samples.shape[0]):
    ax = fig.add_subplot(4, 8, j+1)
    ax.imshow(some_samples[j])
    plt.xticks([]), plt.yticks([])
plt.show()
```



Создание пайплайна данных

In [7]:

```
NUM_EPOCHS = 4
BATCH_SIZE = 64

train_ds = tf.data.Dataset.from_tensor_slices((train_x, train_y))
train_ds = train_ds.shuffle(buffer_size=train_x.shape[0])
train_ds = train_ds.repeat(NUM_EPOCHS)
train_ds = train_ds.batch(BATCH_SIZE)
```

Создание модели CNN

In [24]:

```
class Model(tf.keras.Model):

    def __init__(self):
        super(Model, self).__init__()

        self.conv32_1 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initia
        self.conv32_2 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initia
        self.conv64_1 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initia
        self.conv64_2 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initia
        self.conv128_1 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_init
        self.conv128_2 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_init
        self.fc1 = tf.keras.layers.Dense(128, activation='relu', kernel_initializer='he_uni
        self.fc2 = tf.keras.layers.Dense(10, activation=None)
        self.max_pool = tf.keras.layers.MaxPooling2D((2, 2), (2, 2))
        self.flatten = tf.keras.layers.Flatten()
        self.dropout1 = tf.keras.layers.Dropout(0.3)

    def call(self, inp):

        out = self.conv32_1(inp)
        out = self.conv32_2(out)
        out = self.max_pool(out)
        out = self.conv64_1(out)
        out = self.conv64_2(out)
        out = self.max_pool(out)
        out = self.conv128_1(out)
        out = self.conv128_2(out)
        out = self.max_pool(out)
        out = self.flatten(out)
        out = self.fc1(out)
        out = self.dropout1(out)
        out = self.fc2(out)

        return out

model = Model()
```

Функция потерь и функция вычисления точности

In [25]:

```
def loss(logits, labels):
    return tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(
        logits=logits, labels=labels))

def accuracy(logits, labels):
    predictions = tf.argmax(logits, axis=1, output_type=tf.int32)
    return tf.reduce_mean(tf.cast(tf.equal(predictions, labels), dtype=tf.float32))
```

Подготовка к обучению

In [26]:

```
LEARNING_RATE = 0.001

# optimizer = tf.keras.optimizers.SGD(LEARNING_RATE)
optimizer = tf.keras.optimizers.Adam(LEARNING_RATE)

# writer = tf.summary.create_file_writer('logs/sgd')
writer = tf.summary.create_file_writer('logs/adam_new_dropout')
```

Цикл обучения модели

In [27]:

```
%%time

for iteration, (images, labels) in enumerate(train_ds):

    # Forward
    with tf.GradientTape() as tape:
        logits = model(images)
        loss_value = loss(logits, labels)

    # Backward
    grads = tape.gradient(loss_value, model.trainable_variables)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))

    # Calc and display loss/accuracy
    if iteration % 200 == 0:
        test_logits = model(test_x[:256, ...])
        accuracy_value = accuracy(test_logits, test_y[:256, ...])

        print("[%4d] Accuracy: %5.2f %%" % (
            iteration, accuracy_value.numpy()*100))

        with writer.as_default():
            tf.summary.scalar('accuracy', accuracy_value, iteration)
            tf.summary.scalar('loss', loss_value, iteration)
```

```
[   0] Accuracy: 15.62 %
[ 200] Accuracy: 42.19 %
[ 400] Accuracy: 53.12 %
[ 600] Accuracy: 60.55 %
[ 800] Accuracy: 65.23 %
[1000] Accuracy: 71.48 %
[1200] Accuracy: 70.70 %
[1400] Accuracy: 69.53 %
[1600] Accuracy: 72.66 %
[1800] Accuracy: 73.83 %
[2000] Accuracy: 72.66 %
[2200] Accuracy: 71.88 %
[2400] Accuracy: 76.95 %
[2600] Accuracy: 75.39 %
[2800] Accuracy: 76.17 %
[3000] Accuracy: 77.73 %
Wall time: 13min 5s
```

Оценка качества модели

In [28]:

```
%%time

test_logits = model(test_x)
accuracy_value = accuracy(test_logits, test_y).numpy()
print("Final Accuracy: %5.2f %%" % (accuracy_value * 100))
```

```
Final Accuracy: 74.36 %
Wall time: 9.03 s
```

TensorBoard

In [32]:

```
%load_ext tensorboard
%tensorboard --logdir logs
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

```
Reusing TensorBoard on port 6006 (pid 24228), started 0:41:37 ago. (Use '!ki
ll 24228' to kill it.)
```

Функция для инференса и отображения результата предсказания

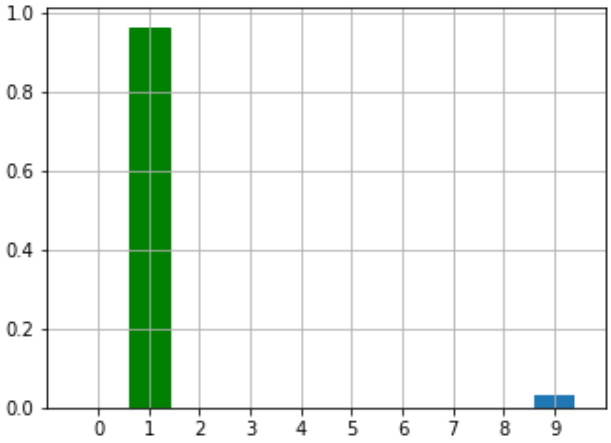
In [33]:

```
def test_item(sample):

    logits = model(sample[None, ...])[0]
    prediction = tf.nn.softmax(logits)
    ans = np.argmax(prediction)

    fig = plt.figure(figsize=(12,4))

    ax = fig.add_subplot(1, 2, 1)
    ax.imshow(sample)
    plt.xticks([]), plt.yticks([])

    ax = fig.add_subplot(1, 2, 2)
    bar_list = ax.bar(np.arange(10), prediction, align='center')
    bar_list[ans].set_color('g')
    ax.set_xticks(np.arange(10))
    ax.set_xlim([-1, 10])
    ax.grid(True)

    plt.show()

    print('Predicted: {}'.format(ans))
```

Запуск предсказания для изображения

In [34]:

```
import random
idx = random.randint(0, test_x.shape[0])
sample = test_x[idx, ...]
test_item(sample)

print('True Answer: {}'.format(test_y[idx]))
```




```
Predicted: 1
True Answer: 1
```