

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

ОТЧЕТ

на тему

**Знакомство с программными средствами
машинного обучения нейронных сетей**

Магистрант:

А.С. Долматович

МИНСК 2019

Ход работы

Для выполнения лабораторной работы использовалась среда Anaconda Navigator.

Порядок выполнения:

- 1) Создано новое окружение (tensorflow-env)
- 2) Установлен пакет tensorflow
- 3) Установлен пакет keras

The screenshot shows the Anaconda Navigator interface. On the left sidebar, the 'Environments' tab is active, and 'tensorflow-env' is highlighted with a red box and a red '1'. The main panel shows the 'Installed' tab for the 'tensorflow' channel. A table lists installed packages, with 'tensorflow' highlighted by a red box and a red '2'.

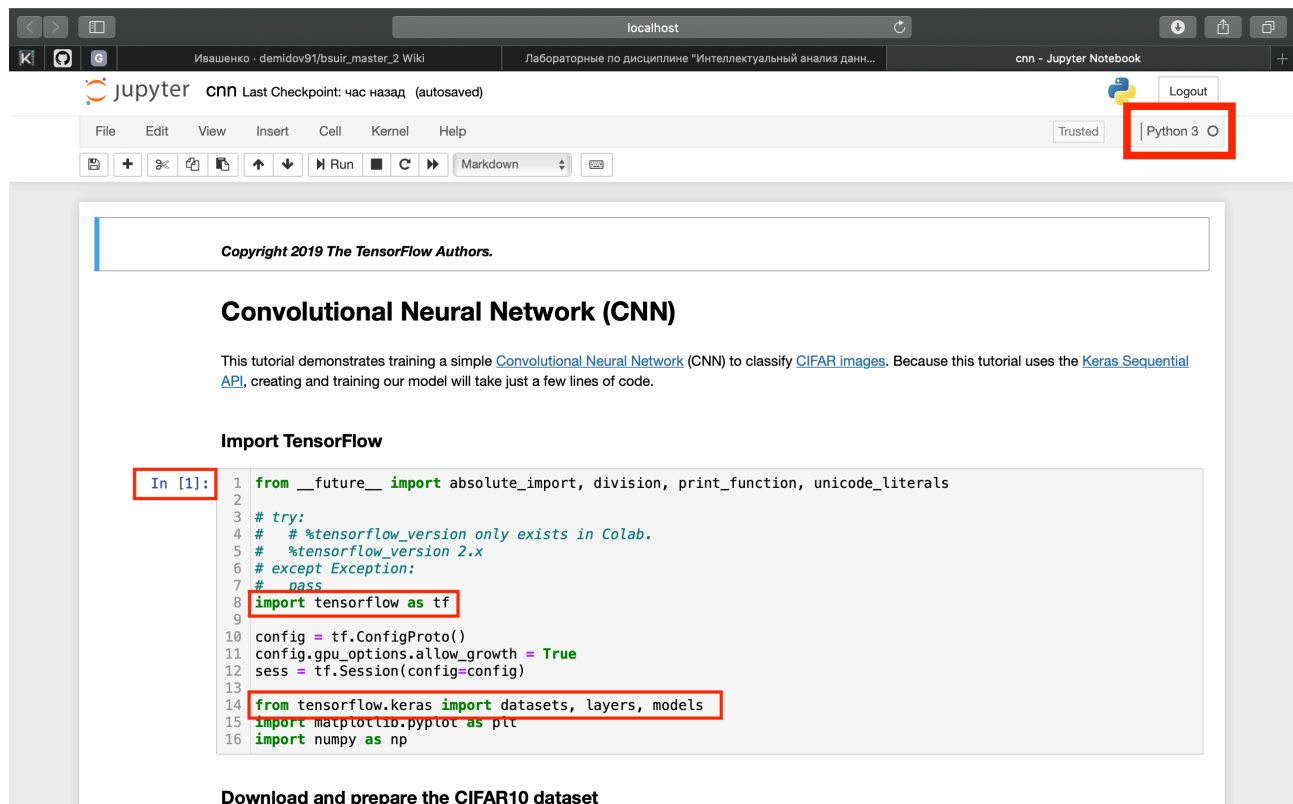
Name	Description	Version
✓ keras	Deep learning library for theano and tensorflow	2.2.4
✓ python-dateutil	Extensions to the standard python datetime module.	2.8.1
✓ tensorboard	Tensorboard lets you watch tensors flow	1.15.0
✓ tensorflow	Tensorflow is a machine learning library.	1.15.0
✓ tensorflow-base	Tensorflow is a machine learning library, base package contains only tensorflow.	1.15.0
✓ tensorflow-estimator		1.15.1

The screenshot shows the Anaconda Navigator interface with the 'tensorflow-env' environment selected. The main panel shows the 'Installed' tab for the 'ker' channel. A table lists installed packages, with 'keras' highlighted by a red box and a red '3'.

Name	Description	Version
✓ ipykernel	Ipython kernel for jupyter	5.1.3
✓ keras	Deep learning library for theano and tensorflow	2.2.4
✓ keras-applications	Applications module of the keras deep learning library.	1.0.8
✓ keras-base		2.2.4
✓ keras-preprocessing	Data preprocessing and data augmentation module of the keras deep learning library	1.1.0
✓ mkl_fft	Numpy-based implementation of fast fourier transform using intel (r) math kernel library.	1.0.15

6 packages available matching "ker"

4) Запущен демонстрационный пример (cifar10 CNN)



Copyright 2019 The TensorFlow Authors.

Convolutional Neural Network (CNN)

This tutorial demonstrates training a simple [Convolutional Neural Network](#) (CNN) to classify [CIFAR images](#). Because this tutorial uses the [Keras Sequential API](#), creating and training our model will take just a few lines of code.

Import TensorFlow

```
In [1]: 1 from __future__ import absolute_import, division, print_function, unicode_literals
2
3 # try:
4 #     %tensorflow_version only exists in Colab.
5 #     %tensorflow_version 2.x
6 # except Exception:
7 #     pass
8 import tensorflow as tf
9
10 config = tf.ConfigProto()
11 config.gpu_options.allow_growth = True
12 sess = tf.Session(config=config)
13
14 from tensorflow.keras import datasets, layers, models
15 import matplotlib.pyplot as plt
16 import numpy as np
```

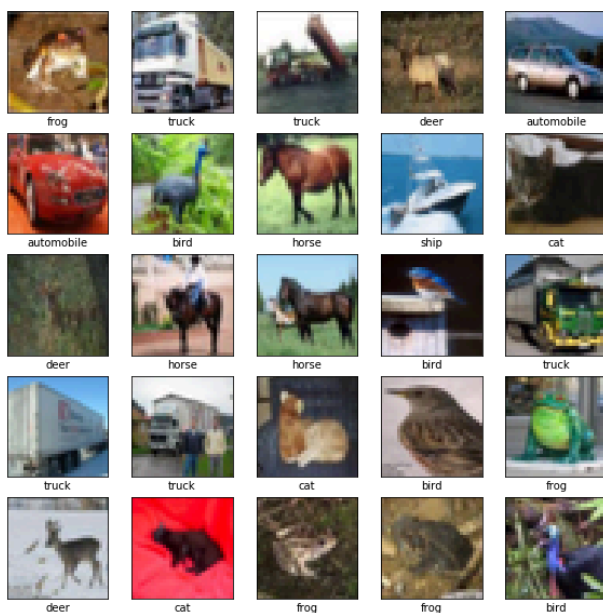
Download and prepare the CIFAR10 dataset

5) Демонстрация работы

Download and prepare the CIFAR10 dataset

Downloading data from <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz>
170500096/170498071 [=====] - 6s 0us/step

Verify the data



Create the convolutional base

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36928
Total params: 56,320		
Trainable params: 56,320		
Non-trainable params: 0		

Add Dense layers on top

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65600
dense_1 (Dense)	(None, 10)	650
Total params: 122,570		
Trainable params: 122,570		
Non-trainable params: 0		

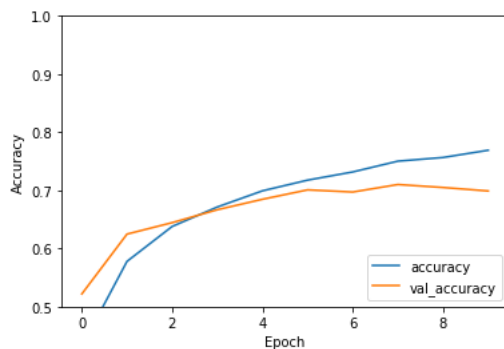
Compile and train the model

Train on 50000 samples, validate on 10000 samples

```
Epoch 1/10
50000/50000 [=====] - 17s 350us/sample - loss: 1.5525 - accuracy: 0.4318 - val_loss: 1.3106 - val_accuracy: 0.5218
Epoch 2/10
50000/50000 [=====] - 13s 251us/sample - loss: 1.1804 - accuracy: 0.5778 - val_loss: 1.0679 - val_accuracy: 0.6247
Epoch 3/10
50000/50000 [=====] - 13s 251us/sample - loss: 1.0293 - accuracy: 0.6378 - val_loss: 1.0181 - val_accuracy: 0.6444
Epoch 4/10
50000/50000 [=====] - 13s 251us/sample - loss: 0.9350 - accuracy: 0.6713 - val_loss: 0.9505 - val_accuracy: 0.6666
Epoch 5/10
50000/50000 [=====] - 13s 251us/sample - loss: 0.8626 - accuracy: 0.6991 - val_loss: 0.9081 - val_accuracy: 0.6846
Epoch 6/10
50000/50000 [=====] - 12s 249us/sample - loss: 0.8080 - accuracy: 0.7175 - val_loss: 0.8715 - val_accuracy: 0.7008
Epoch 7/10
50000/50000 [=====] - 12s 249us/sample - loss: 0.7608 - accuracy: 0.7316 - val_loss: 0.8829 - val_accuracy: 0.6971
Epoch 8/10
50000/50000 [=====] - 12s 250us/sample - loss: 0.7163 - accuracy: 0.7502 - val_loss: 0.8533 - val_accuracy: 0.7100
Epoch 9/10
50000/50000 [=====] - 13s 250us/sample - loss: 0.6909 - accuracy: 0.7565 - val_loss: 0.8538 - val_accuracy: 0.7049
Epoch 10/10
50000/50000 [=====] - 12s 249us/sample - loss: 0.6567 - accuracy: 0.7690 - val_loss: 0.8818 - val_accuracy: 0.6989
```

Evaluate the model

10000/1 - ls - loss: 1.1445 - accuracy: 0.6989



Ответы на вопросы

- Как задать модель нейронной сети. Какие есть интерфейсы и их параметры?
 - The Sequential model API

```
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

- Model (functional API)

```
from keras.models import Model
from keras.layers import Input, Dense

a = Input(shape=(32,))
b = Dense(32)(a)
model = Model(inputs=a, outputs=b)
```

- Как задать весовые коэффициенты нейронной сети?
 - sample_weight
 - class_weight
- Как задать полносвязный слой нейронной сети?
 - model.add(Dense(32, input_shape=(16,)))
- Как задать свёрточный слой нейронной сети?
 - layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3))
- Какие есть средства для работы с рекуррентными нейросетями?
 - class RNN
 - class SimpleRNN
 - GRU
 - LSTM
 - ConvLSTM2D
 - CuDNNGRU
 - CuDNNLSTM
- Как задать функцию активации нейронной сети и какие поддерживаются в keras?

- class Activation или argument activation
 - elu
 - softmax
 - selu
 - softplus
 - softsign
 - relu
 - tanh
 - sigmoid
 - hard_sigmoid
 - exponential
 - linear
 - LeakyReLU
 - PReLU
 - ELU
 - ThresholdedReLU
 - Softmax
 - ReLU
- Чем отличается linear от ReLU, softplus?
 - softplus - $\log(\exp(x) + 1)$.
 - linear - линейная функция активации
 - ReLU - $\max(x, 0)$ для значений по умолчанию,
 - $f(x) = \max_value$ для $x \geq \max_value$
 - $f(x) = x$ для $\text{threshold} \leq x < \max_value$
 - $f(x) = \text{negative_slope} * (x - \text{threshold})$ во всех остальных случаях
- Как задать функцию ошибки\ потерь нейронной сети?
 - `model.compile(loss=loss.mean_squared_error, optimizer='sgd')`
 Можно либо передать имя существующей функции потерь, либо передать символическую функцию TensorFlow, которая возвращает скаляр для каждой точки данных и принимает следующие два аргумента:
 - `y_true`: настоящие ярлыки
 - `y_pred`: предсказания.
- Чем отличается `mean_squared_error` от `cosinus_proximity`, по каким формулам они вычисляются?
 - `cosinus_proximity`

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

- `mean_squared_error`

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

- Как задать метод обучения нейронной сети?

```
from keras import optimizers

model = Sequential()
model.add(Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(Activation('softmax'))

sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

- Чем отличается SGD от rprop, Adadelata, Adam; nesterov от momentum?
 - SGD - оптимизатор стохастического градиентного спуска.
 - Adadelata - это более надежное расширение Adagrad, которое адаптирует скорости обучения на основе движущегося окна обновлений градиентов, а не накапливает все прошлые градиенты. Таким образом, Adadelata продолжает учиться, даже если было сделано много обновлений.
 - Adam - алгоритм для градиентной оптимизации стохастических целевых функций первого порядка, основанный на адаптивных оценках моментов более низкого порядка.
 - momentum: float >= 0. Параметр, который ускоряет SGD в соответствующем направлении
 - nesterov: bool. Нужно ли применять Нестеров импульс.
- Как указать обучающую выборку?
 - model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_data=(x_test, y_test), shuffle=True)

СПИСОК ИСТОЧНИКОВ

[1] Оф. сайт Chocolatey. — Электронный ресурс. — Режим доступа: <https://chocolatey.org/> — Дата доступа: 22.11.2019.

[2] Оф. сайт Python. — Электронный ресурс. — Режим доступа: <https://www.python.org/> — Дата доступа: 22.11.2019.

[3] Оф. сайт PIP. — Электронный ресурс. — Режим доступа: <https://pypi.org/project/pip/> — Дата доступа: 22.11.2019.

[4] Документация виртуальной среды Python. — Электронный ресурс. — Режим доступа:

<https://docs.python.org/3/library/venv.html> — Дата доступа: 22.11.2019.

[5] Пример глубокого обучения с помощью библиотеки Keras. — Электронный ресурс. — Режим

доступа: <https://keras.io> — Дата доступа: 22.11.2019.