MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

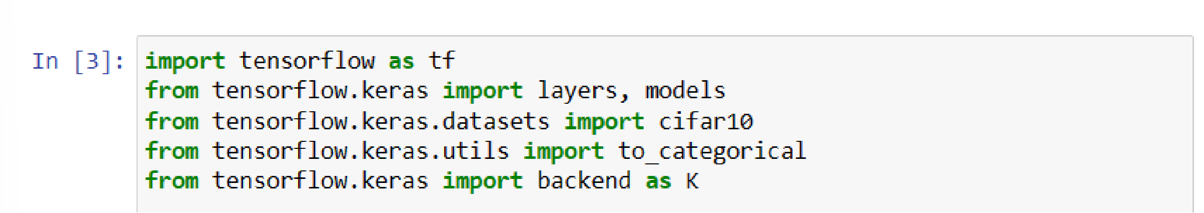
Volodymyr Shymkovych

Design and implementation of software systems with neural networks

LABORATORY WORK #4

kulubecioglu mehmet
IM-14 FIOT

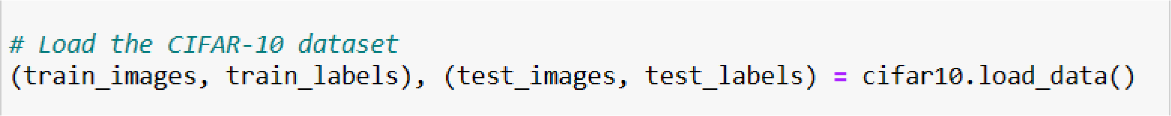
Kyiv
IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE
2024



```
In [3]: import tensorflow as tf
        from tensorflow.keras import layers, models
        from tensorflow.keras.datasets import cifar10
        from tensorflow.keras.utils import to_categorical
        from tensorflow.keras import backend as K
```

Import Libraries:

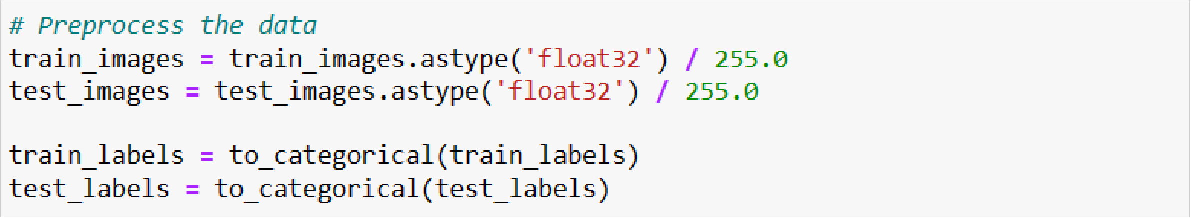
- Import TensorFlow and related modules to create and train neural networks.
- To create neural network layers and models using layers and models from tensorflow.keras.
- cifar10 and to_categorical are used to load the CIFAR-10 dataset and encode the labels one-hot.
- To check image data format using backend from tensorflow.keras.



```
# Load the CIFAR-10 dataset
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
```

Loading the CIFAR-10 Dataset:

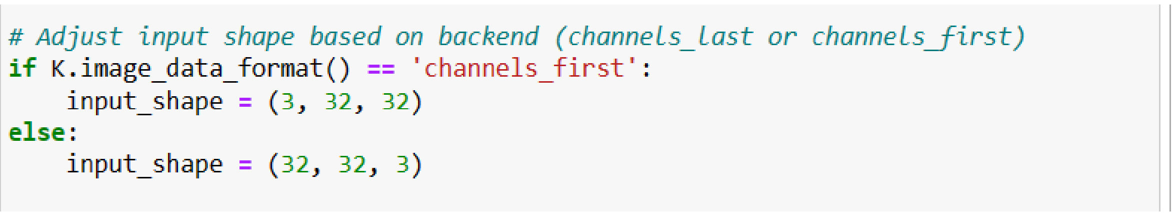
- Loading the CIFAR-10 dataset consisting of 60,000 32x32 color images.



```
# Preprocess the data
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

Data Preprocessing:

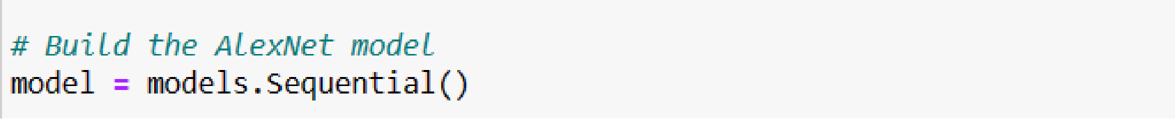
- Dividing the pixel values of images by 255 to normalize them to a range of 0 to 1.
- Encoding tags one-hot using to_categorical.



```
# Adjust input shape based on backend (channels_last or channels_first)
if K.image_data_format() == 'channels_first':
    input_shape = (3, 32, 32)
else:
    input_shape = (32, 32, 3)
```

Setting the Login Type:

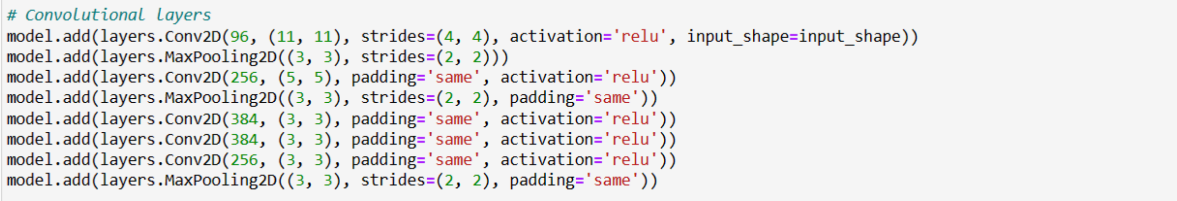
- Determining the input type depending on the image data format used by Keras (channels_first or channels_last).



```
# Build the AlexNet model
model = models.Sequential()
```

Creating a Sequential Model:

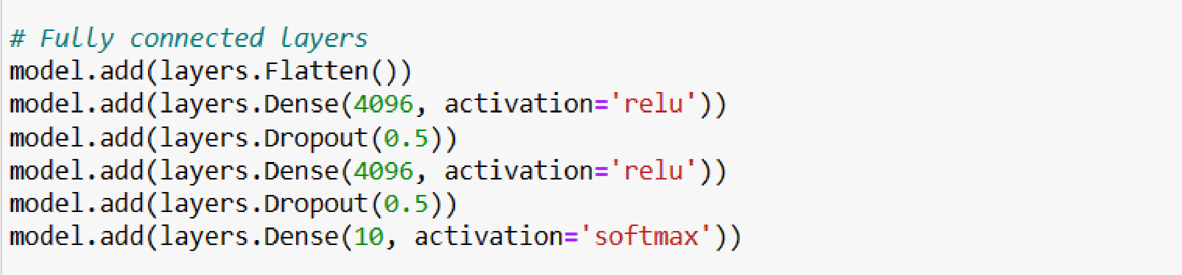
- Creating a sequential model using Keras.



```
# Convolutional layers
model.add(layers.Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=input_shape))
model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))
model.add(layers.Conv2D(256, (5, 5), padding='same', activation='relu'))
model.add(layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
model.add(layers.Conv2D(384, (3, 3), padding='same', activation='relu'))
model.add(layers.Conv2D(384, (3, 3), padding='same', activation='relu'))
model.add(layers.Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(layers.MaxPooling2D((3, 3), strides=(2, 2), padding='same'))
```

Convolution Layers:

- I added convolution layers with certain parameters; kernel size, number of steps, activation function and border.
- I added max-pooling layers to reduce spatial dimensions.

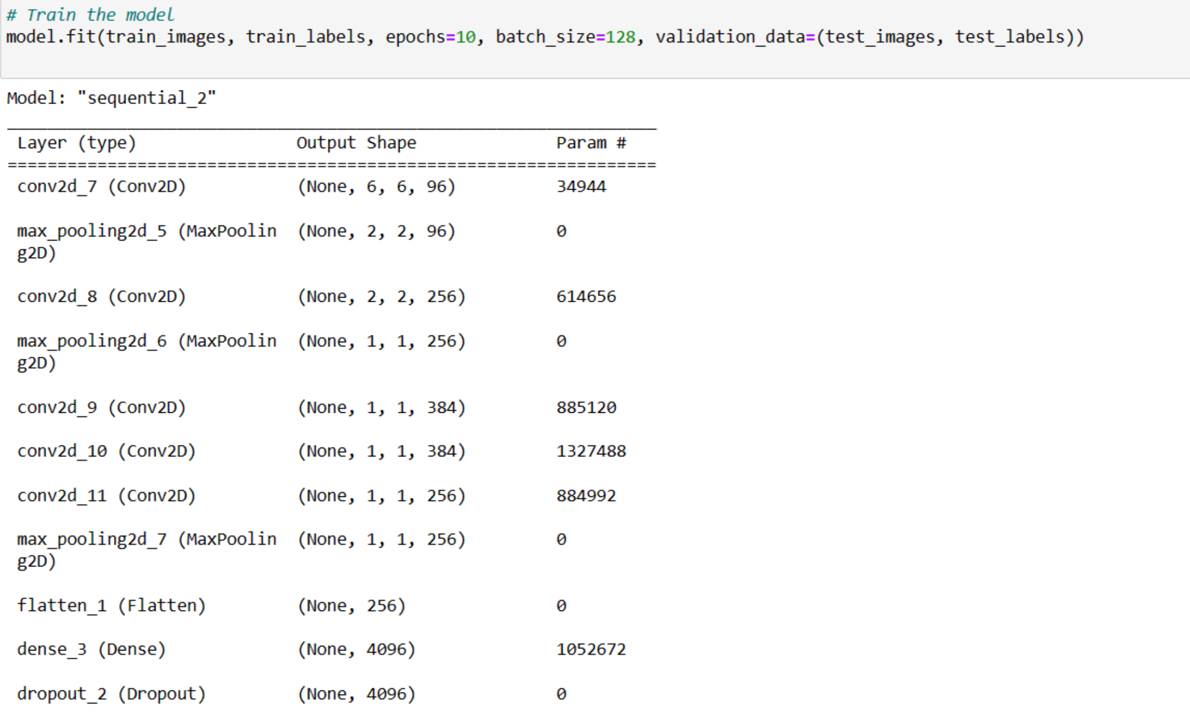


```
# Fully connected layers
model.add(layers.Flatten())
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))
```

Fully Connected Layers:

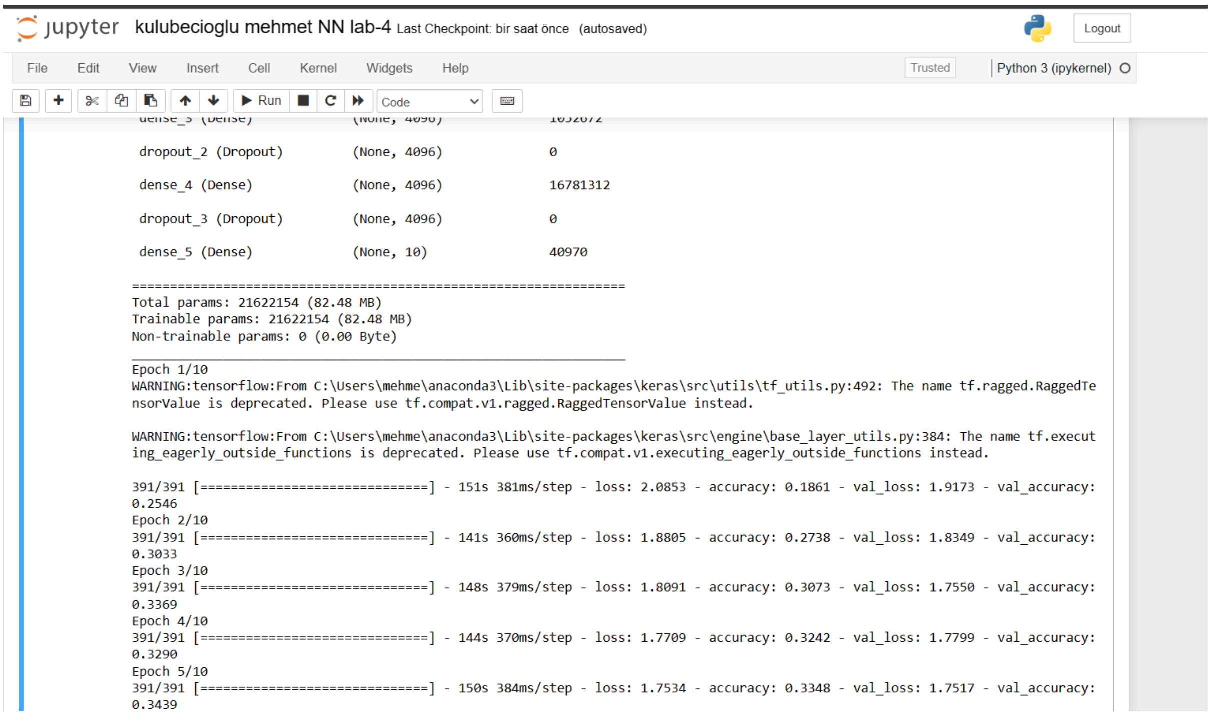
- I added the Flatten layer to flatten the output of the convolution layers.
- I added fully connected layers with ReLU activation function and dropout for orchestration.
- I added the last dense layer for the 10-class classification.

My Output:



```
# Train the model
model.fit(train_images, train_labels, epochs=10, batch_size=128, validation_data=(test_images, test_labels))
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)           (None, 6, 6, 96)          34944

 max_pooling2d_5 (MaxPoolin  (None, 2, 2, 96)          0
 g2D)

 conv2d_8 (Conv2D)           (None, 2, 2, 256)         614656

 max_pooling2d_6 (MaxPoolin  (None, 1, 1, 256)         0
 g2D)

 conv2d_9 (Conv2D)           (None, 1, 1, 384)         885120

 conv2d_10 (Conv2D)          (None, 1, 1, 384)         1327488

 conv2d_11 (Conv2D)          (None, 1, 1, 256)         884992

 max_pooling2d_7 (MaxPoolin  (None, 1, 1, 256)         0
 g2D)

 flatten_1 (Flatten)         (None, 256)               0

 dense_3 (Dense)             (None, 4096)              1052672

 dropout_2 (Dropout)         (None, 4096)              0
```



```
 dense_3 (Dense)             (None, 4096)              1052672

 dropout_2 (Dropout)         (None, 4096)              0

 dense_4 (Dense)             (None, 4096)              16781312

 dropout_3 (Dropout)         (None, 4096)              0

 dense_5 (Dense)             (None, 10)                40970

=================================================================
Total params: 21622154 (82.48 MB)
Trainable params: 21622154 (82.48 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
WARNING:tensorflow:From C:\Users\mehme\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTe
nsorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\mehme\anaconda3\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.execut
ing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

391/391 [==============================] - 151s 381ms/step - loss: 2.0853 - accuracy: 0.1861 - val_loss: 1.9173 - val_accuracy:
0.2546
Epoch 2/10
391/391 [==============================] - 141s 360ms/step - loss: 1.8805 - accuracy: 0.2738 - val_loss: 1.8349 - val_accuracy:
0.3033
Epoch 3/10
391/391 [==============================] - 148s 379ms/step - loss: 1.8091 - accuracy: 0.3073 - val_loss: 1.7550 - val_accuracy:
0.3369
Epoch 4/10
391/391 [==============================] - 144s 370ms/step - loss: 1.7709 - accuracy: 0.3242 - val_loss: 1.7799 - val_accuracy:
0.3290
Epoch 5/10
391/391 [==============================] - 150s 384ms/step - loss: 1.7534 - accuracy: 0.3348 - val_loss: 1.7517 - val_accuracy:
0.3439
```

```
0.3439
Epoch 6/10
391/391 [==============================] - 141s 360ms/step - loss: 1.7316 - accuracy: 0.3513 - val_loss: 1.6978 - val_accuracy:
0.3668
Epoch 7/10
391/391 [==============================] - 142s 363ms/step - loss: 1.6988 - accuracy: 0.3681 - val_loss: 1.7340 - val_accuracy:
0.3566
Epoch 8/10
391/391 [==============================] - 147s 375ms/step - loss: 1.6751 - accuracy: 0.3799 - val_loss: 1.6654 - val_accuracy:
0.3865
Epoch 9/10
391/391 [==============================] - 142s 364ms/step - loss: 1.6456 - accuracy: 0.3926 - val_loss: 1.6478 - val_accuracy:
0.3956
Epoch 10/10
391/391 [==============================] - 147s 376ms/step - loss: 1.6282 - accuracy: 0.4014 - val_loss: 1.6549 - val_accuracy:
0.3912
```

```
Out[3]: <keras.src.callbacks.History at 0x2479f6b0ad0>
```

```
In [ ]:
```