МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание объектов на фотографиях

Студент гр.8382 _____ Фильцин И.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

Распознавание объектов на фотографиях (Object Recognition in Photographs) CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).

Задание

Ознакомиться со сверточными нейронными сетями

Изучить построение модели в Keras в функциональном виде

Изучить работу слоя разреживания (Dropout)

Ход работы

Рассмотрим следующую архитектуру сети. Conv [32] -> Conv [32] -> Pool (Dropout) -> Conv [64] -> Conv [64] -> Pool (Dropout) -> Dense (Dropout) -> Dense. Гиперпараметры следующие:

batch_size = 32 — количество обучающих образцов, обрабатываемых одновременно за одну итерацию алгоритма градиентного спуска;

num_epochs = 200 — количество итераций обучающего алгоритма по всему обучающему множеству;

kernel_size = 3 — размер ядра в сверточных слоях;

pool_size = 2— размер подвыборки в слоях подвыборки;

conv_depth_1 = 32 — количество ядер в сверточных слоях;

conv_depth_2 = 64 — количество ядер в сверточных слоях;

drop_prob_1 = 0.25 — дропаут после пулинга

drop_prob_2 = 0.5 — дропаут в полносвязном слое

hidden_size = 512 — количество нейронов в полносвязном слое MLP.

Обучим данную модель. Результат обучения следующий: точность на тренировочных данных = 0.8888, на валидационных - 0.787, на тестовых -

0.3793. Таким образом точность на тренировочных данных приемлема. При этом точность практически не менялась уже после 90 эпохи.

Изменим значение гиперпараметра num_epochs на 90 и проведем исследование той же самой сети, но без использования dropout.

Точность на тренировочных данных = 0.9895, на валидационных - 0.71, на тестовых - 0.578.

Вернем dropout и изменим размер ядра свертки с 3 на 4.

Точность на тренировочных данных = 0.7898, на валидационных - 0.751, на тестовых - 0.508.

Изменим размер ядра свертки с 4 на 2.

Точность на тренировочных данных = 0.8913, на валидационных - 0.7904, на тестовых - 0.3664.

Таким образом, оптимильный размер ядра свертки равен 3.

Вывод

В ходе лабораторной работы была реализована сеть, осуществляющая распознавание объектов на фотографиях. Было исследовано использование слоя Dropout и влияние размера ядра свертки на результат.

Приложение А.

Исходный код

```
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Convolution2D, MaxPooling2D, Dense, Dropout, Flatten
import tensorflow.keras.utils as utils
import numpy as np

batch_size = 32 # in each iteration, we consider 32 training examples at once
num_epochs = 200 # we iterate 200 times over the entire training set
kernel_size = 3 # we will use 3x3 kernels throughout
pool_size = 2 # we will use 2x2 pooling throughout
conv_depth_1 = 32 # we will initially have 32 kernels per conv. layer...
conv_depth_2 = 64 # ...switching to 64 after the first pooling layer
drop_prob_1 = 0.25 # dropout after pooling with probability 0.25
drop_prob_2 = 0.5 # dropout in the dense layer with probability 0.5
hidden_size = 512 # the dense layer will have 512 neurons

(X_train, y_train), (X_test, y_test) = cifar10.load_data() # fetch CIFAR-10 data
num_train, depth, height, width = X_train.shape # there are 50000 training examples in CIFAR-10

num_test = X_test.shape[0] # there are 10000 test examples in CIFAR-10
num_classes = np.unique(y_train).shape[0] # there are 10 image classes

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train) # Normalise data to [0, 1] range
X_test /= np.max(X_train) # Normalise data to [0, 1] range

Y_train = utils.to_categorical(y_train, num_classes) # One-hot encode the labels
Y_test = utils.to_categorical(y_test, num_classes) # One-hot encode the labels

inp = Input(shape=(depth, height, width)) # N.B. depth goes first in Keras

# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)

# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), activation='relu')(conv_3)
```

```
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)

# Now flatten to 1D, apply Dense —> ReLU (with dropout) —> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)
drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(inputs=inp, outputs=out) # To define a model, just specify its input and output layers

model.compile(loss='categorical_crossentropy', # using the cross—entropy loss function
              optimizer='adam', # using the Adam optimiser
              metrics=['accuracy']) # reporting the accuracy

model.fit(X_train, Y_train, # Train the model using the training set...
          batch_size=batch_size, epochs=num_epochs,
          verbose=1, validation_split=0.1) # ...holding out 10% of the data for validation

model.evaluate(X_test, Y_test, verbose=1) # Evaluate the trained model on the test set!
```