МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание объектов на фотографиях

Студент гр. 8383 _____ Шишкин 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).

Задачи.

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)

Выполнение работы.

Был загружен датасет из cifar10. Мы не рассматриваем каждый пиксель как независимое входное значение, и поэтому мы не переносим изображение в одномерное пространство. Задаются следующие гиперпараметры:

- batch_size — количество обучающих образцов, обрабатываемых одновременно за одну итерацию алгоритма градиентного спуска;
- num_epochs — количество итераций обучающего алгоритма по всему обучающему множеству;
- kernel_size — размер ядра в сверточных слоях;
- pool_size — размер подвыборки в слоях подвыборки;
- conv_depth — количество ядер в сверточных слоях;
- drop_prob (dropout probability) — мы будем применять dropout после каждого слоя подвыборки, а также после полносвязного слоя;
- hidden_size — количество нейронов в полносвязном слое MLP.

  Для модели №1 задаются параметры, представленные в листинге 1. Так как у меня нет графического процессора, то количество эпох сокращено до 10.

Листинг 1 – Гиперпараметры модели №1

```
batch_size = 32   # in each iteration, we consider 32 training
examples at once
num_epochs = 10   # we iterate 10 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
```

Сеть состоит из четырех слоев Convolution_2D и слоев MaxPooling2D после второй и четвертой сверток. После этого выходное изображение слоя подвыборки трансформируется в одномерный вектор (слоем Flatten) и проходит два полносвязных слоя (Dense). На всех слоях, кроме выходного полносвязного слоя, используется функция активации ReLU, последний же слой использует softmax. Для регуляризации нашей модели после каждого слоя подвыборки и первого полносвязного слоя применяется слой Dropout.

Результат выполнения программы для модели №1 (RESULT – точность и потери на тестовых данных):

45000/45000 [==============================] - 49s 1ms/step - loss: 0.6018 - acc: 0.7874 - val_loss: 0.7212 - val_acc: 0.7636

RESULT:

[141.1863461730957, 0.5963]

Точность у модели №1 на тренировочных данных составляет примерно 78%, на проверочных – 77%, а на тестовых – 60%.

В модели №2 были удалены слои Dropout. Результат выполнения программы для модели №2:

Epoch 10/10 - 45s - loss: 0.0840 - acc: 0.9716 - val_loss: 1.4874 - val_acc: 0.7458

RESULT:

[546.693963671875, 0.5577]

Точность у модели №2 на тренировочных данных составляет примерно 97%, на проверочных – 75%, а на тестовых – 56%. Так как Dropout помогает исключить ситуации с переобучением, то после его удаления, можно сделать вывод, что наша сеть начала переобучаться.

Для модели №3 вернем слои Dropout и уменьшим kernel_size до 2.

Epoch 10/10 - 38s - loss: 0.6615 - acc: 0.7668 - val_loss: 0.7157 - val_acc: 0.7520

RESULT:

[264.81757685546876, 0.3942]

Скорость обучения заметно увеличилась (с 49с за эпоху до 38с), при этом точность ухудшилась, по сравнению с 1-й моделью (77% на тренировочных, 72% на проверочных и 40% на тестовых.

Для модели №4 увеличим kernel_size до 4.

Epoch 10/10 - 50s - loss: 0.6876 - acc: 0.7583 - val_loss: 0.6918 - val_acc: 0.7636

RESULT:

[162.09647088623046, 0.5223]

Эта модель показывает себя лучше предыдущей: точность на тренировочных данных осталась почти такой же (упала с 77% до 76), на валидационных поднялась с 75 до 76%, а на тестовых возросла с 40% до 52. Но это все равно хуже, чем когда размер ядра свертки был 3x3.

Для модели №5 увеличим kernel_size до 5.

Epoch 10/10 - 58s - loss: 0.7228 - acc: 0.7476 - val_loss: 0.7755 - val_acc: 0.7364

RESULT:

[131.94034506835936, 0.553]

Можно заметить, что с увеличением размера ядра свертки увеличивается и время на обучение. При этом точность у этой модели еще хуже, чем у предыдущей: на тренировочных упала с 76 до 75%, на проверочных с 76 до 74, но на тестовых увеличилась с 52 до 56%.

Для модели №6 увеличим kernel_size до 6.

Epoch 10/10 - 73s - loss: 0.7844 - acc: 0.7241 - val_loss: 0.8134 - val_acc: 0.7248

RESULT:

[298.12688134765625, 0.4028]

Точность стала хуже. На тренировочных данных вместо 75% стала 72, на проверочных упала с 74 до 72%, на тестовых – с 56 до 40%.

Выводы.

Была построена и обучена сверточная нейронная сеть. Исследована работа без слоя Dropout. Исследована работа сети при разных размерах ядра свертки. Лучше всего себя показала модель №1 со слоями Dropout и с размером ядра свертки, равном 3x3.