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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искусственные нейронные сети»

Тема: Распознавание объектов на фотографиях

Студентка гр. 8383 ____ Аверина О.С.

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)

Требования

1. Построить и обучить сверточную нейронную сеть
2. Исследовать работу сети без слоя Dropout
3. Исследовать работу сети при разных размерах ядра свертки

Выполнение работы

В работе будет использоваться модель, состоящая из четырех слоев свертки, двух слоев субдискретизации и двух полносвязных слоев. Сверточные слои выделяют наиболее полезные признаки в изображении и для каждого такогопризнака формируют карту признаков. Слой субдискретизации (пулинга) уменьшает разрешение изображения, тем самым сокращая количество параметров модели. Полносвязные слои осуществляют непосредственно классификацию.

Зададим начальные параметры модели.

- 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

```
batch_size = 32
num_epochs = 25
kernel_size = 3
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512
```

Загрузим и подготовим данные к обработке.

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
num_train, depth, height, width = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)
Y_train = utils.to_categorical(y_train, num_classes)
Y_test = utils.to_categorical(y_test, num_classes)
```

Модель 1.

Построим модель. В нее входят входной слой, два сверточных слоя, слой субдискретизации, слой Dropout, затем еще раз для сверточных, слой субдискретизации, слой Dropout. Выходной полносвязный слой использует функцию активации softmax для осуществления классификации изображений.

```
inp = Input(shape=(depth, height, width))
conv_1 = Convolution2D(conv_depth_1, kernel_size, padding='same',
activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, kernel_size, padding='same',
activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)

conv_3 = Convolution2D(conv_depth_2, kernel_size, padding='same',
activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, padding='same',
activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)

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)
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
history = model.fit(X_train, Y_train,
          batch_size=batch_size, epochs=num_epochs,
          verbose=1, validation_split=0.1)
model.evaluate(X_test, Y_test, verbose=1)
```

Обучим модель в течение 25 эпох батчами по 32 образца, а затем оценим модель на тестовом множестве:

```
Epoch 15/15
704/704 [==============================] - 12s 17ms/step - loss: 0.4088
- accuracy: 0.8516 - val_loss: 0.6435 - val_accuracy: 0.7936
       313/313 [==============================] - 2s 6ms/step - loss:
```

```
215.1989 - accuracy: 0.5709
```

Можно заметить, что на обучающих данных модель достигает достаточной точности и малых потерь, однако на тестовых данных точность ниже и бОльшие потери.

На рисунках 1 и 2 изображены графики ошибок и точности на тренировочном и тестовом множествах в процессе обучения.

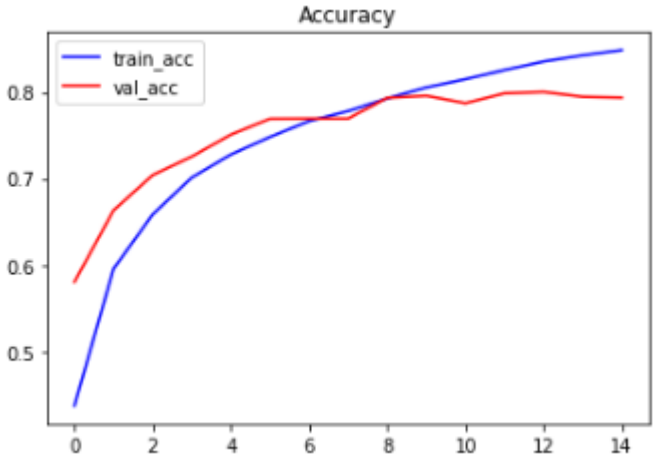

Рисунок 1

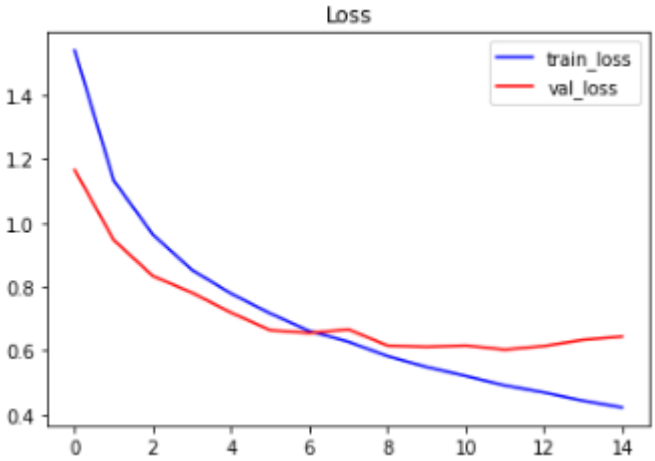

Рисунок 2

По графикам заметно, что произошло переобучение модели после 7 эпохи. Сократим количество эпох до 15.

Модель 2.

Для того, чтобы изучить влияние слоя Dropout на процесс обучения модели, уберем все такие слои из 1 модели.

```
inp = Input(shape=(depth, height, width))

conv_1 = Convolution2D(conv_depth_1,
kernel_size,padding='same',
activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1,
kernel_size,padding='same',
activation='relu')(conv_1) pool_1 =
MaxPooling2D(pool_size=(pool_size,
pool_size))(conv_2)

conv_3 = Convolution2D(conv_depth_2,
kernel_size,padding='same',
activation='relu')(pool_1) conv_4 =
Convolution2D(conv_depth_2, kernel_size,
padding='same',  activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size,
pool_size))(conv_4)

flat = Flatten()(pool_2)
hidden = Dense(hidden_size, activation='relu')(flat)
out = Dense(num_classes,
activation='softmax')(hidden)

model = Model(inputs=inp, outputs=out)
```

Обучим полученную модель и оценим ее на тестовом множестве:

```
Epoch 15/15
1407/1407 [==============================] - 17s 12ms/step - loss:
0.0679 - accuracy: 0.9779 - val_loss: 1.8334 - val_accuracy: 0.7334
       313/313 [==============================] - 2s 5ms/step - loss:
  741.5773 - accuracy: 0.5001
```

Точность модели на валидационном множестве оказалась хуже, чем точность модели со слоями дропаута.

На рисунках 5 и 6 изображены графики ошибок и точности на валидационном множестве моделей со слоями дропаута и без.

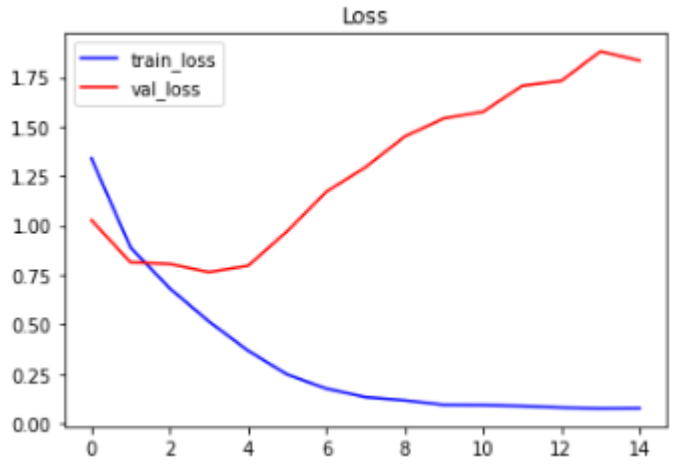

Рисунок 5

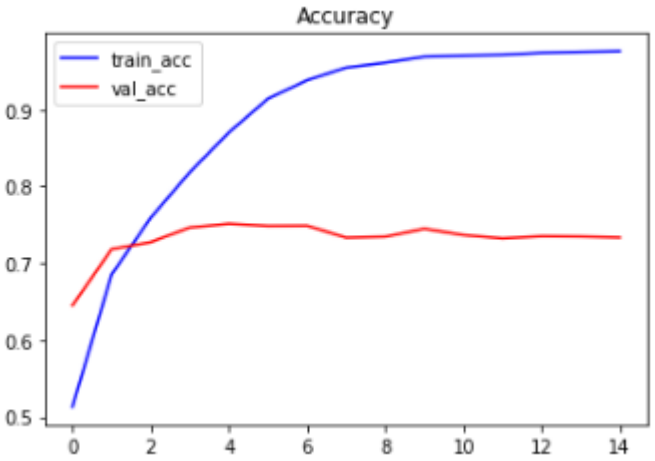

Рисунок 6

Потери модели без дропаута значительно выше, а точность – ниже, что говорито том, что дропаут положительно влияет на обучение модели.

Модель 3.

Исследуем работу сети при разных размерах ядра свертки *n*. Вернем в модель слои Dropout и изменим размер ядра, задаваемый параметром `kernel_size`, на 5 и на 2. Обучим модели с этими параметрами и оценим точность на тестовом множестве:

Размер ядра – 5:

```
Epoch 15/15
704/704 [==============================] - 16s 23ms/step - loss:
0.5240 - accuracy: 0.8155 - val_loss: 0.7162 - val_accuracy: 0.7758
313/313 [==============================] - 2s 7ms/step - loss:
219.7635 - accuracy: 0.5472
```

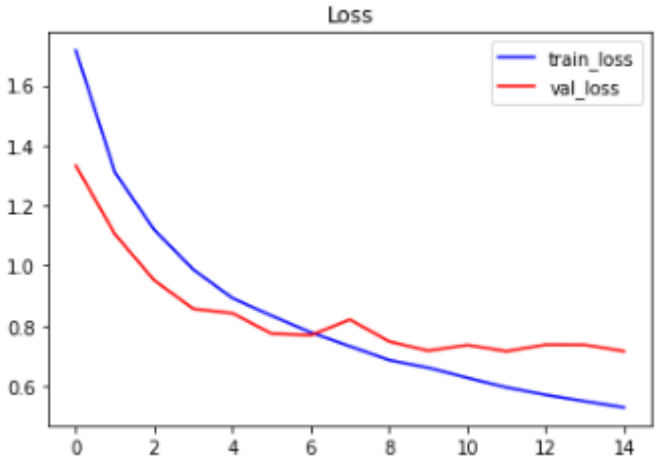

Рисунок 7

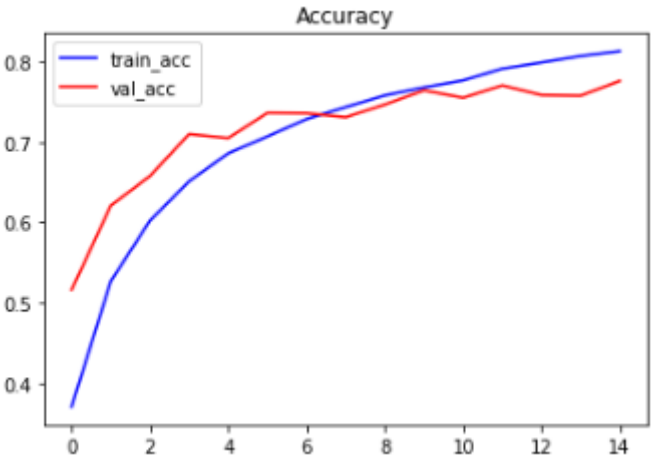

Рисунок 8

Размер ядра 2:

```
Epoch 15/15
704/704 [==============================] - 12s 17ms/step - loss:
0.5101 - accuracy: 0.8177 - val_loss: 0.6574 - val_accuracy: 0.7782
313/313 [==============================] - 2s 5ms/step - loss:
365.6061 - accuracy: 0.4091
```

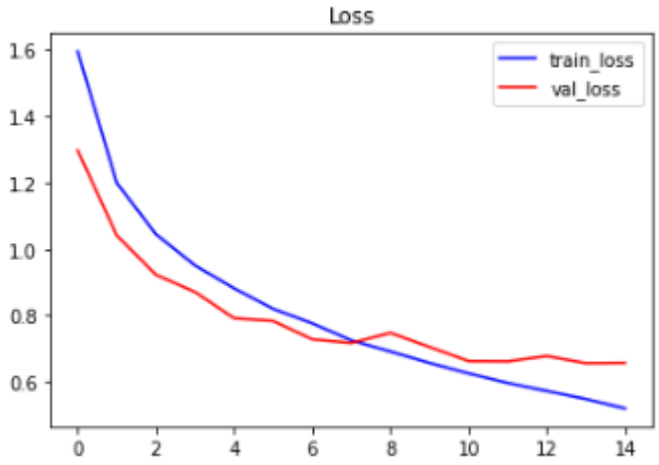

Рисунок 9

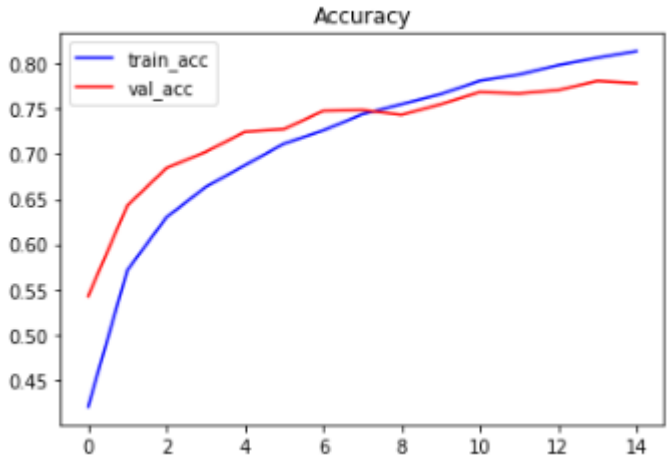

Рисунок 10

На графиках видно, что, чем меньше размер ядра, тем выше точность модели на тестовом множестве и тем меньше потери.

Сравним точности исследованных моделей:

	Исходная	Без Dropout	Размер ядра 5	Размер ядра 2
Точность	0.5709	0.5001	0.5472	0.4091
Потери	215.1989	741.5773	219.7635	365.6061

Из таблицы видно, что в ходе исследования лучшие показатели были у исходной модели.

Выводы

В ходе выполнения лабораторной работы была создана нейронная сеть со сверточной архитектурой, осуществляющая классификацию изображений. Изучен принцип работы сверточных нейронных сетей, исследовано влияние слоя Dropout на процесс обучения, рассмотрена работа нейронной сети при различных размерах ядра свертки.