

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»

Студентка гр. 8383

Сырцова Е.А.

Преподаватель

Жангиров Т.Р.

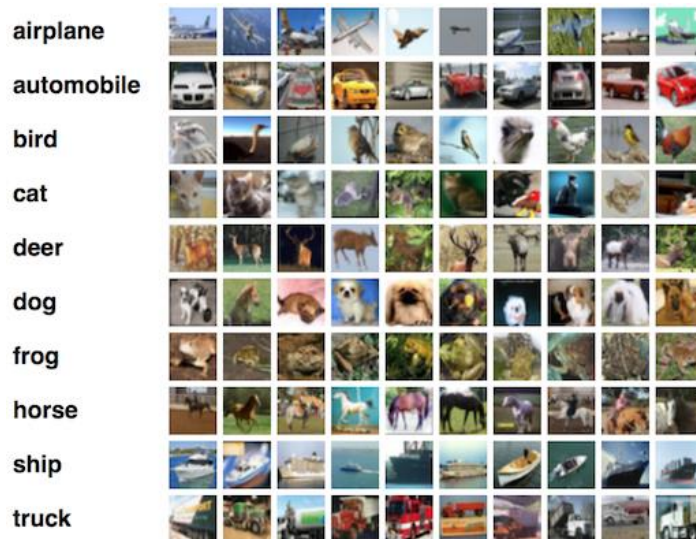
Санкт-Петербург

2021

Цель

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 сеть без слоя Dropout
3. Исследовать работу сети при разных размерах ядра свертки

Ход работы

В качестве практической части построим глубокую сверточную нейронную сеть и применим ее к классификации изображений из набора CIFAR-10.

Зададим 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.

Изначально зададим количество эпох равным 15 и размер ядра 3, слой Dropout включен. Полученный результаты представлены на рис.1-2.

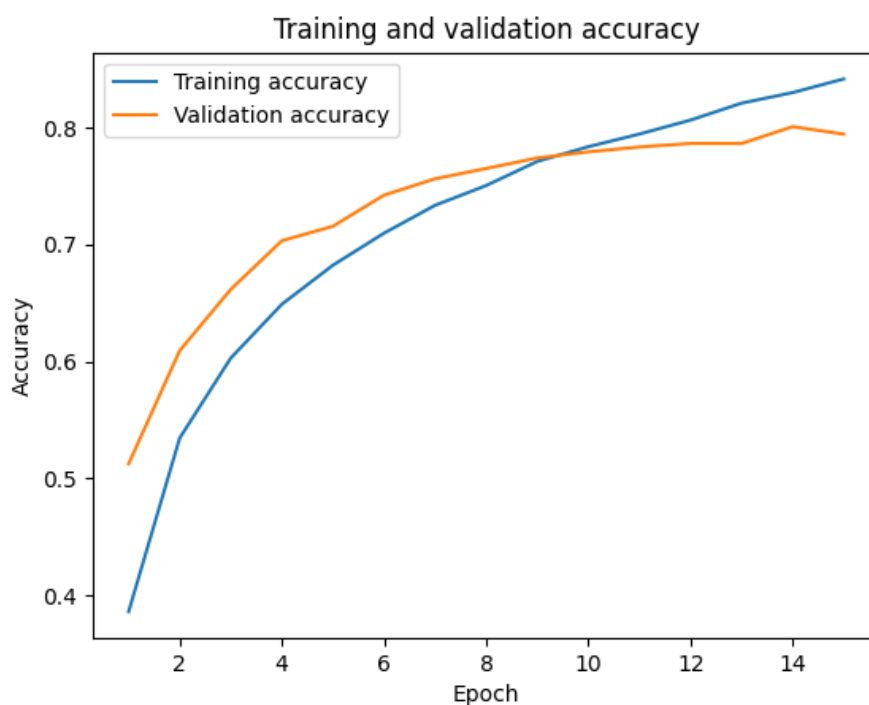


Рисунок 1 – Точность при размере ядра свертки 3×3 и включенном Dropout

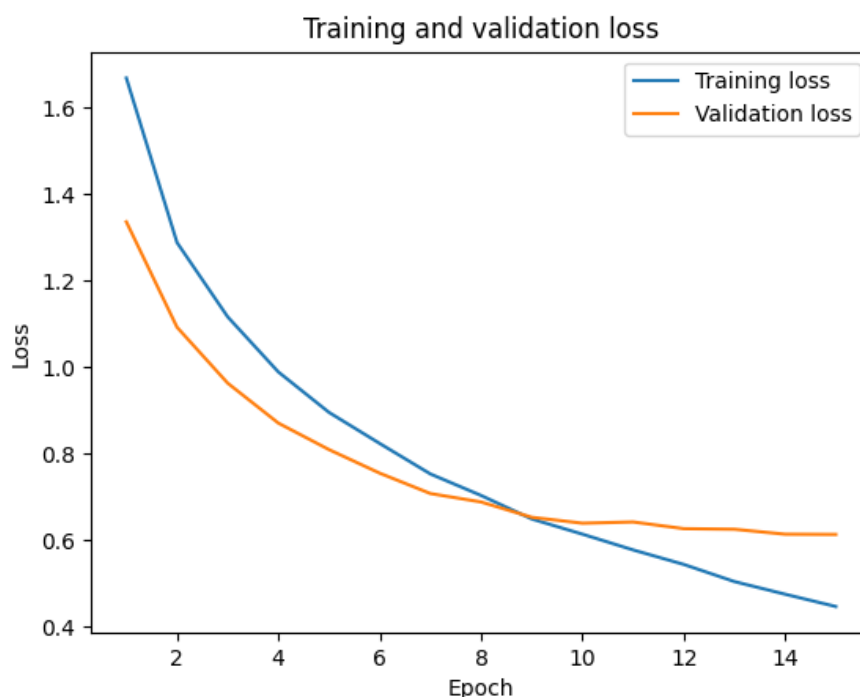


Рисунок 2 – Ошибка при размере ядра свертки 3×3 и включенном Dropout

Точность при данной модели сети составляет примерно 84%.

Далее слой Dropout был отключен, а размер ядра оставим прежним, результаты представлены на рис.3-4.

Точность увеличилась на тренировочный данных, однако на тестовых данных она понижается и становится примерна равной 78%. Как видно из графиков, модель достигает точки переобучения на 10 эпохе, переобучение выражается явно, точность на тестовых данных снижается, а ошибка продолжает расти. Отсутствие слоев Dropout делает модель склонной к быстрому переобучению.

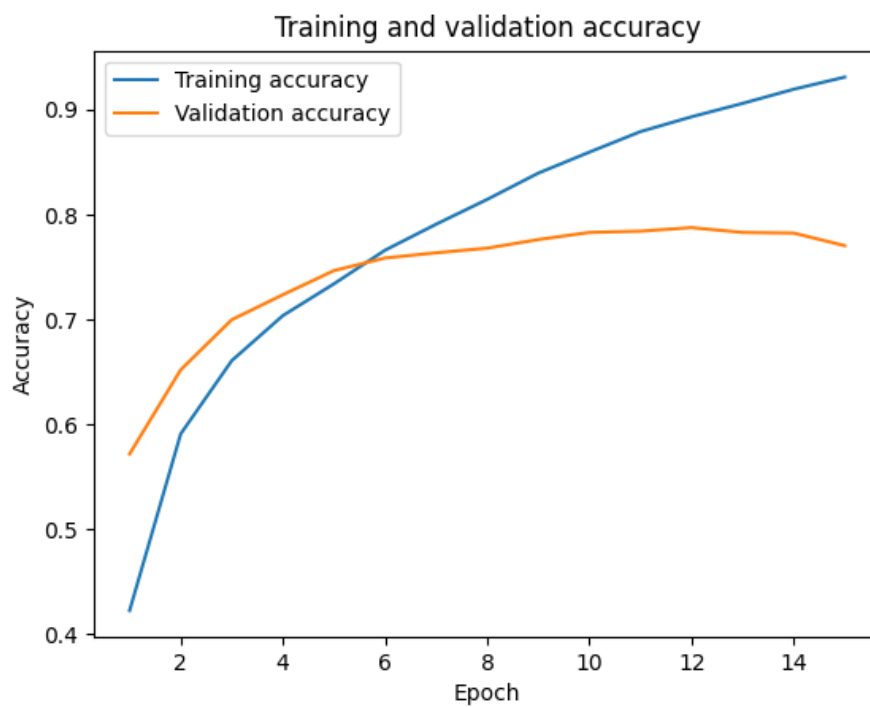


Рисунок 3 – Точность при размере ядра свертки 3×3 и выключенном Dropout

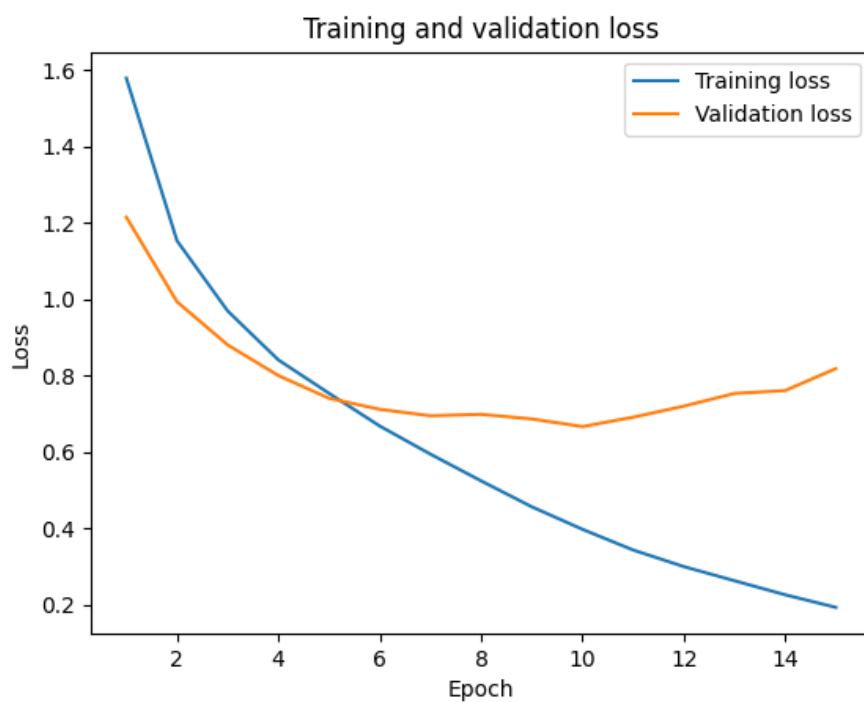


Рисунок 4 – Ошибка при размере ядра свертки 3×3 и выключенном Dropout

Далее исследуем работу сети при разных значениях размера ядра, результаты для размера ядра 5×5 приведены на рис.5-6 и размера ядра 2×2 на рис. 7-8.

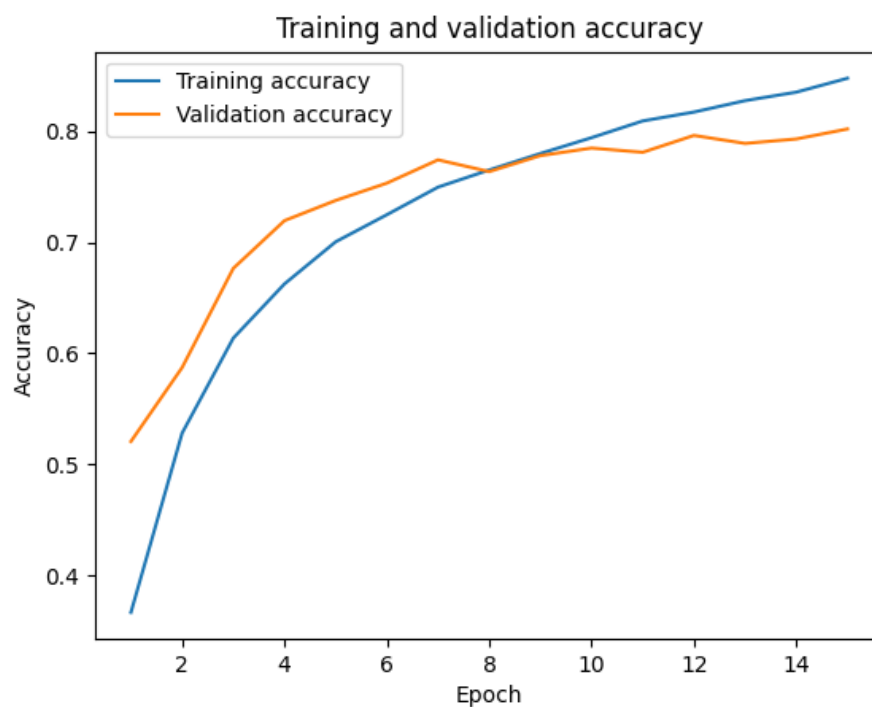


Рисунок 5 – Точность при размере ядра свертки 5×5 и включенном Dropout

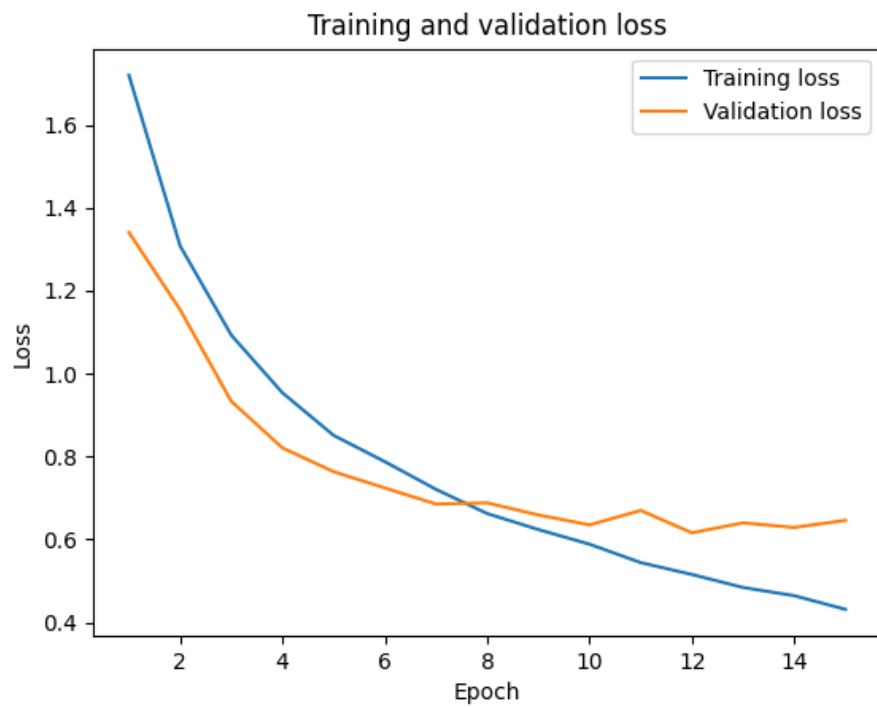


Рисунок 6 – Ошибка при размере ядра свертки 5×5 и включенном Dropout

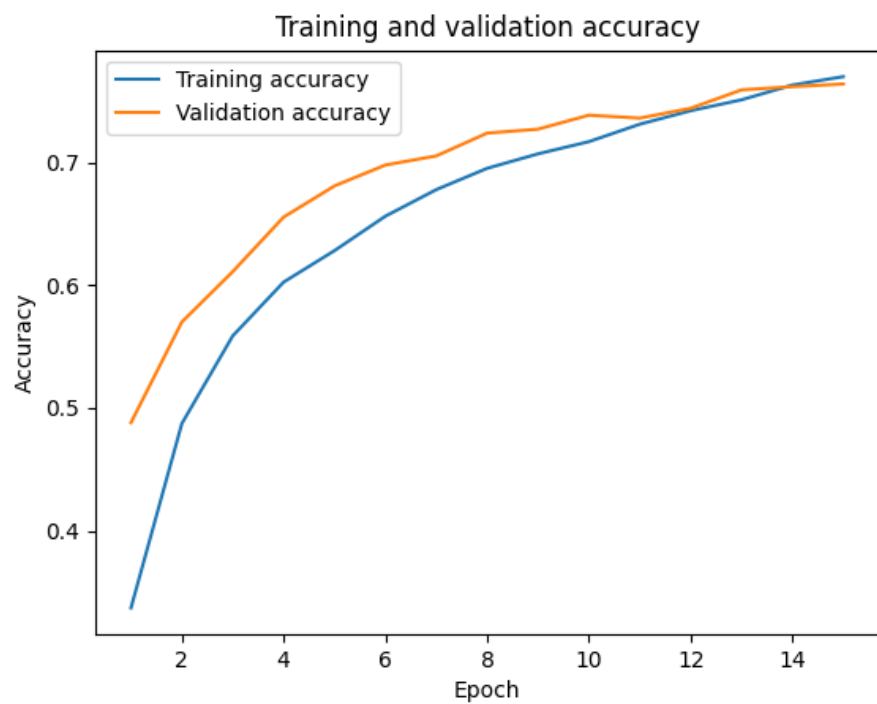


Рисунок 7 – Точность при размере ядра свертки 2×2 и включенном Dropout

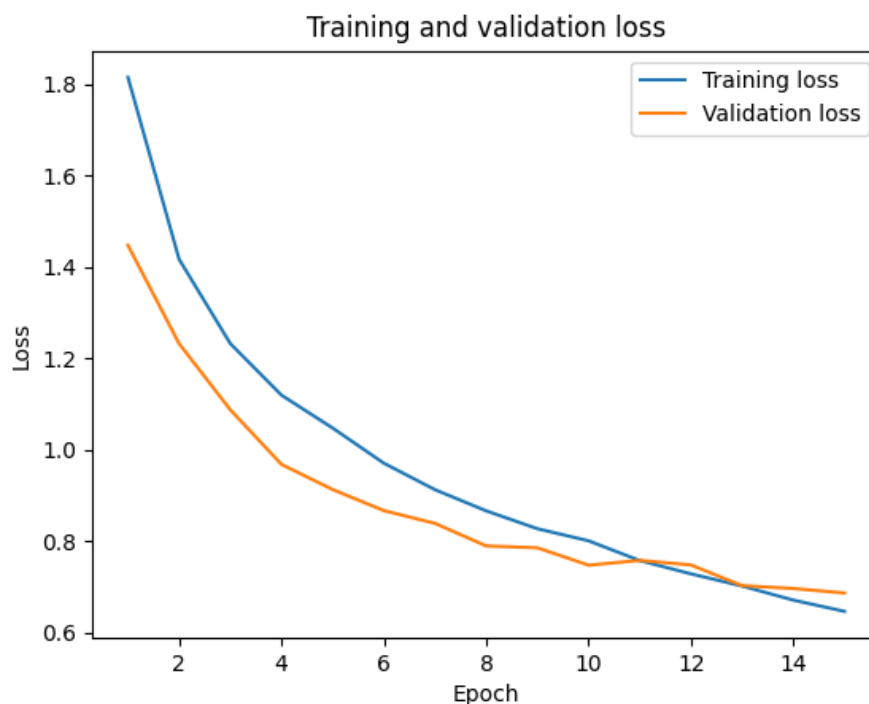


Рисунок 8 – Ошибка при размере ядра свертки 2×2 и включенном Dropout

Из графиков видно, что при меньшем размере ядра и при равенстве других параметров сеть выдает большую точность. Модель с большим размером ядра свертки требует больше вычислительной мощности, но при малом ядре свертки сети сложнее выявить признаки для классификации изображений.

Вывод

В процессе выполнения лабораторной работы было изучено решение задачи классификации цветных изображений с помощью сверточной нейронной сети. Было изучено влияние слоев Dropout на показатели сети во время обучения, а также влияние размера ядра свертки. В результате было выявлено, что применение слоев Dropout позволяет решить проблему переобучения сети, а размер ядра свертки 3×3 оптимальный для решения поставленной задачи.

Код программы представлен в приложении А.

ПРИЛОЖЕНИЕ А

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

batch_size = 256
num_epochs = 15
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

(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 = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)

def build_model(dropout=True):
    inp = Input(shape=(depth, height, width))
    conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
padding='same', activation='relu')(inp)
    conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
padding='same', activation='relu')(conv_1)
    pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
    if dropout:
        drop_1 = Dropout(drop_prob_1)(pool_1)
    else:
        drop_1 = pool_1
    conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size),
padding='same', activation='relu')(drop_1)
    conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size),
padding='same', activation='relu')(conv_3)
    pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
```

```

if dropout:
    drop_2 = Dropout(drop_prob_1)(pool_2)
else:
    drop_2 = 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(inp, 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)
print(model.evaluate(X_test, Y_test, verbose=1))

x = range(1, num_epochs + 1)
plt.plot(x, history.history['loss'], label='Training loss')
plt.plot(x, history.history['val_loss'], label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
plt.clf()

plt.plot(x, history.history['accuracy'], label='Training accuracy')
plt.plot(x, history.history['val_accuracy'], label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()

```

```

build_model()

```