1. **Применение методов искусственного интеллекта для ЦОС.**

***Работа 10. Выбор лучших параметров сверточной нейронной сети***

При решении с помощью методов цифровой обработки задач классификации объектов (изображений, звуковых сигналов) в дополнение к методам ЦОС можно использовать методы искусственного интеллекта, в частности, сверточные нейронные сети. Типичная структура сверточной нейронной сети имеет вид рис. 1

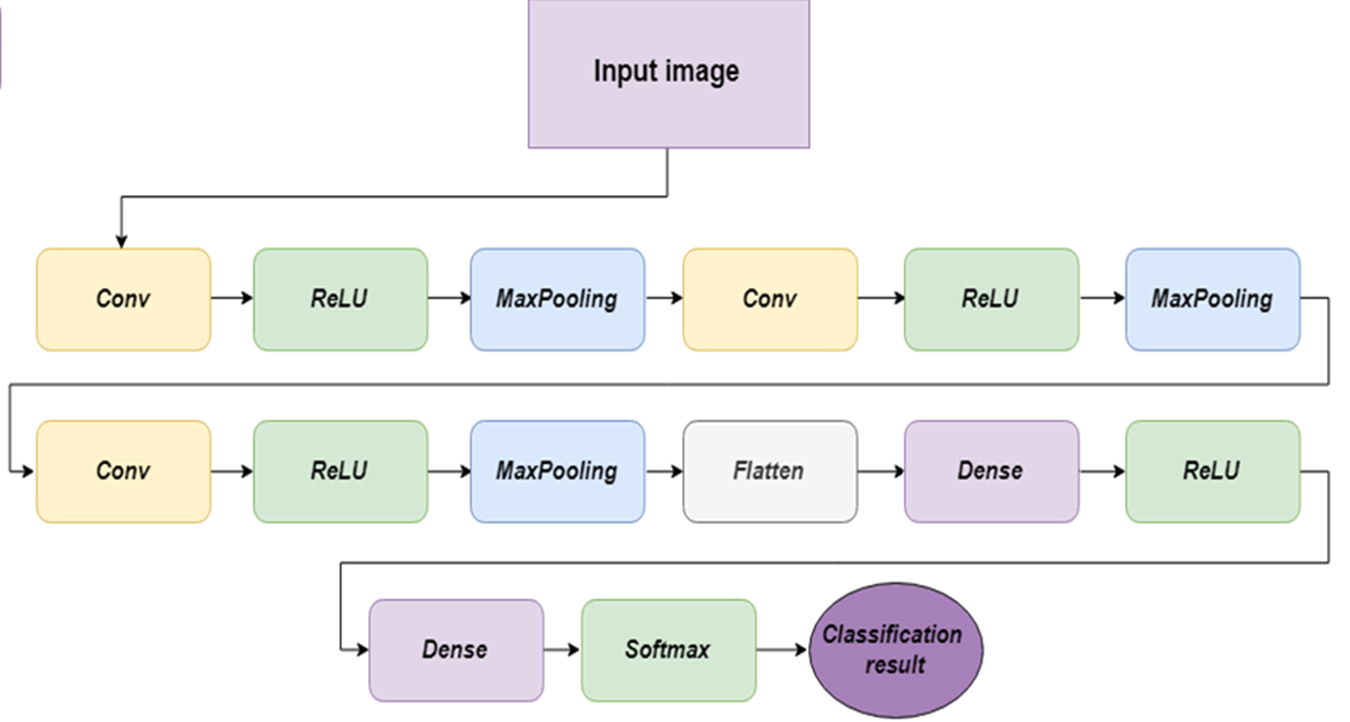


Рис. 1

Такая сверточная нейронная сеть, спроектированная для решения задачи распознавания болезней сельскохозяйственных растений по изображениям листьев, может быть успешно использована и для других целей, например, для распознавания рукописных цифр.

Качество работы нейронной сети определяется временем обучения и достигаемым процентом правильного распознавания. Качество зависит от выбора гиперпараметров нейронной сети: количества слоев, количества нейронов в слоях, количества эпох при обучении, размеров масок в слоях Conv и MaxPoolling, размера мини-батчей. Теоретически обоснованных правил выбора гиперпараметров не существует, выбор производится эмпирически по результатам (времени обучения и процентах правильного распознавания) при многократных запусках процесса обучения нейронной сети. Самые общие принципы выбора гиперпараметров приведены ниже.

Минимальное количество слоев -3: свёрточный слой, слой подвыборки и полносвязный слой.

Количество скрытых нейронов должно быть между размером входного слоя и размером выходного слоя. Количество скрытых нейронов должно составлять 2/3 размера входного слоя плюс размер выходного слоя. Количество скрытых нейронов должно плавно убывать о первого слоя до последнего.

При небольших размерах мини-батчей удвоение размеров батча позволяет обучаться в два раза быстрее без дополнительных вычислений. При слишком больших размерах обучающего пакета параллельные вычисления не приводят к ускорению обучения.

Оптимальное количество эпох — это область между состояниями недообучения и переобучения модели. Определяется это количество эмпирически в контексте конкретной задачи. А определить эту область можно с помощью усредненного графика минимизации функции потерь по результатам нескольких итераций обучения.

**Цель работы**: научиться оценивать влияние гиперпараметров обучения (количество эпох обучения, размер мини-выборки, количество нейронов во входном слое, количество скрытых слоев) на качество обучения нейронной сети.

**Базовые программные средства**

Используем программу сверточной нейронной сети *classificatorR1.py* из папки *Python\_150\_digit.py*. Для обучения используется 150 рукописных изображений цифр от 0 до 9, представленных в формате *.PNG*. Файлы изображений распределены в 3 папки: *train\_dir* – 100 файлов для обучения нейронной сети, val\_dir – 30 файлов для валидации и test\_dir – 20 файлов для тестирования. Размер изображений цифр – 28х28 пикселов.

Текст программы *Python\_150\_digit.py* приведен ниже.

from tensorflow.python.keras.preprocessing.image import ImageDataGenerator

from tensorflow.python.keras.models import Sequential

from tensorflow.python.keras.layers import Conv2D, MaxPooling2D

from tensorflow.python.keras.layers import Activation, Dropout, Flatten, Dense

# Каталог с данными для обучения

train\_dir = 'train\_dir'

# Каталог с данными для тестирования

test\_dir = 'test\_dir'

# Каталог с данными для валидации

val\_dir = 'val\_dir'

# Размеры изображения

img\_width, img\_height = 28, 28

# Размерность тензора на основе изображения для входных данных в нейронную сеть

input\_shape = (img\_width, img\_height, 3)

# Количество эпох

epochs = 200

# Размер мини-выборки

batch\_size = 20

# Количество изображений для обучения

nb\_train\_samples = 100

# Количество изображений для проверки

nb\_validation\_samples = 30

# Количество изображений для тестирования

nb\_test\_samples = 20

model = Sequential()

model.add(Conv2D(16, (3, 3), input\_shape=input\_shape))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Flatten())

model.add(Dense(20))

model.add(Activation('relu'))

model.add(Dropout(0.25))

model.add(Dense(9))

model.add(Activation('softmax'))

model.compile(loss='sparse\_categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

#model.summary()

datagen = ImageDataGenerator(rescale=1. / 255)

train\_generator = datagen.flow\_from\_directory(

train\_dir,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode='sparse')

val\_generator = datagen.flow\_from\_directory(

val\_dir,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode='sparse')

test\_generator = datagen.flow\_from\_directory(

test\_dir,

target\_size=(img\_width, img\_height),

batch\_size=batch\_size,

class\_mode='sparse')

model.fit\_generator(

train\_generator,

steps\_per\_epoch=nb\_train\_samples // batch\_size,

validation\_data=val\_generator,

validation\_steps= nb\_validation\_samples // batch\_size,

epochs=epochs,

shuffle = True)

model.save('digit\_diagnosis.h5')

print("Score: ", scores)

Программа обучения нейронной сети *classificatorR1.py* выводит полученные в результате обучения значения accuracy (доля правильного распознавания) и loss (потери) Одной из наиболее распространенных функций потерь при обучении нейронных сетей является среднеквадратичная ошибка (Mean Squared Error, MSE).

Эта метрика используется для оценки разницы между предсказанными и фактическими значениями и может быть применена к различным задачам, таким как регрессия и классификация. MSE является квадратом отклонения между предсказанными и фактическими значениями и позволяет учесть все ошибки, независимо от их знака.

MSE = (1/n) \* Σ(y\_true — y\_pred)^2

где:

MSE – значение среднеквадратичной ошибки;

n – количество наблюдений;

y\_true – истинное значение;

y\_pred – предсказанное значение.

Принцип работы среднеквадратичной ошибки заключается в следующем: для каждого примера из обучающего набора данных рассчитывается разница между предсказанным и фактическим значением, эта разница возведена в квадрат и затем усредняется для всех примеров. Таким образом, мы получаем среднюю величину отклонения и можем использовать ее для оценки качества работы нейронной сети.

Источник: https://uchet-jkh.ru/i/srednekvadraticnaya-osibka-neironnoi-seti-osnovy-i-primenenie

Одно из основных преимуществ среднеквадратичной ошибки заключается в том, что она дифференцируема, что позволяет использовать ее при оптимизации с помощью градиентного спуска. Это позволяет нейронной сети корректировать веса своих нейронов и находить оптимальные значения для достижения наилучшего результата.

Источник: https://uchet-jkh.ru/i/srednekvadraticnaya-osibka-neironnoi-seti-osnovy-i-primenenie

Программа обучения нейронной сети также создает файл обученной нейронной сети *digit\_diagnosis.h5*

В программе тестирования test\_1\_TNN.py, использующей программу обученной нейронной сети*, digit\_diagnosis.h5* предусмотрено представление результатов классификации в форме списка вероятностей отнесения результатов классификации ко всем 10 возможным классам.

Предусмотрена возможность также задания в программе, обученной нейронной сети порога распознавания level. Если все полученные вероятности меньше level, выводится сообщение «ошибка распознавания»

Пример: файлы для тестирования: digit6102 – рукописное искаженное незначительно изображение числа 6 -правильное распознавание digit 6045 – значительно искаженное изображение числа 6 -ошибка распознавания

Ниже приведен текст программы тестирования test\_1\_TNN.py,

import os

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '3'

import numpy as np

import tensorflow as tf

from tensorflow.python.keras import utils

#Подготовка правильных ответов

model = tf.keras.models.load\_model('digit\_diagnosis.h5')

path = 'image\_for\_test/digit6102.png'

image = tf.keras.preprocessing.image.load\_img(path, target\_size=(28, 28))

x = tf.keras.preprocessing.image.img\_to\_array(image)

classes = model.predict(np.array([x]))

print(classes) #Выведет набор вероятностей принадлежности тестового изображения к каждому классу

print('номер класса, предсказанного нейросетью')

print(np.argmax(classes)+1)

#print('вероятность принадлежности тестового изображения к одному из классов')

classes = model.predict(np.array([x]))

#вывод элемента 3 тензора classes нулевого ранга (т.е. вектора)

#поскольку элементы в тензоре нумеруются, начиная с нуля, фактически элемент будет 4

print('максимальная вероятность принадлежности')

print(np.max(classes))

level=0.85 #задание порога распознавания

print('порог распознавания=',level)

if np.max(classes)<level:

print('ошибка распознавания!')

В протоколе обучающей программы приводятся показатели accuracy и loss,

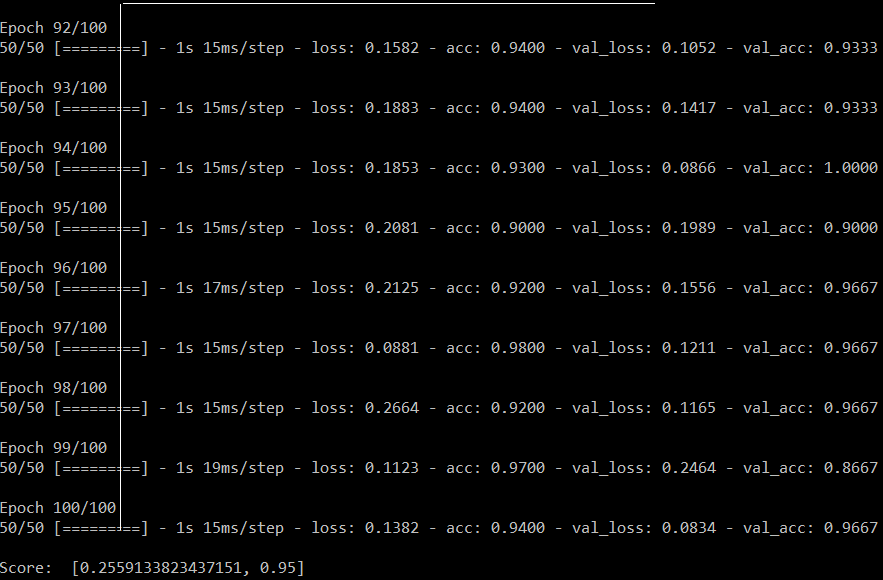
полученные для обучающего и валидационного множества исходных данных на каждой эпохе, а также итоговые значения показателей accuracy и loss, полученные для тестирующего множества исходных данных

Пример протокола выполнения обучающей программы

а)начальный участок:



б)заключительный участок:



**Программа работы**

1. Запишите значения параметров, заданных в базовой программе: количество скрытых слоев, количество нейронов в слоях, количество эпох обучения, размер мини-выборки, и результаты работы обучающей программы (значения acc и loss) для массивов данных train, val и test.
2. Последовательно изменяйте значения гиперпараметров и запускайте обучающую программу добиваясь улучшения качества по итоговому показателю score (окончательный итог).

**Указания по выполнению работы**

1. При подборе количества эпох обучения попробуйте обучать сеть в течение 50, 75, 100 и 125 эпох. Выберите количество эпох, при котором самая высокая доля верных ответов нейросети на тестовых данных или, что то же, по итоговому показателю *score*.
2. При подборе размера мини-выборки используйте размер мини-выборки 20, 50, 100. Выберите значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.
3. При подборе количества нейронов во входном слое Используйте значения 500, 700, 900, 1200. Выберите значение, при котором самая высокая доля верных ответов нейросети на тестовых данных.
4. При исследовании влияния количества слоев в нейросети Попробуйте добавить скрытый слой с разным количеством нейронов: 500, 700, 900, 1200 и выберите наиболее подходящее количество нейронов скрытого слоя. Оцените, как изменяется время обучения при добавлении скрытого слоя с разным количеством нейронов и оцените – увеличивается ли при этом доля верных ответов.

**Содержание отчета**

1. Цель работы.
2. Программа работы.
3. Таблицы с результатами измерений по пунктам программы.
4. Фрагменты протоколов обучения.
5. Графики функций точности и потерь (acc и loss)
6. Промежуточные результаты работы по пунктам программы.
7. Выводы.