## Работа 11. Выбор параметров нейронной сети прямого распространения для классификации цифровых описаний текстурных изображений.

**Теоретические основы**

При решении задачи классификации изображений с помощью нейронной сети большего эффекта в ряде случаев можно достичь за счет выделения ключевых признаков классифицируемых объектов методами ЦОС до входа в нейронную сеть. В частности, можно преобразовать текстурные изображения в их цифровые описания. Это позволит в десятки раз уменьшить размерность описания изображений и объем исходных данных, необходимых для обучения, использовать нейронную сеть прямого распространения вместо сверточной нейронной сети и за счет этого сократить затраты времени на обучение нейронной сети.

Типичная структура нейронной сети прямого распространения.

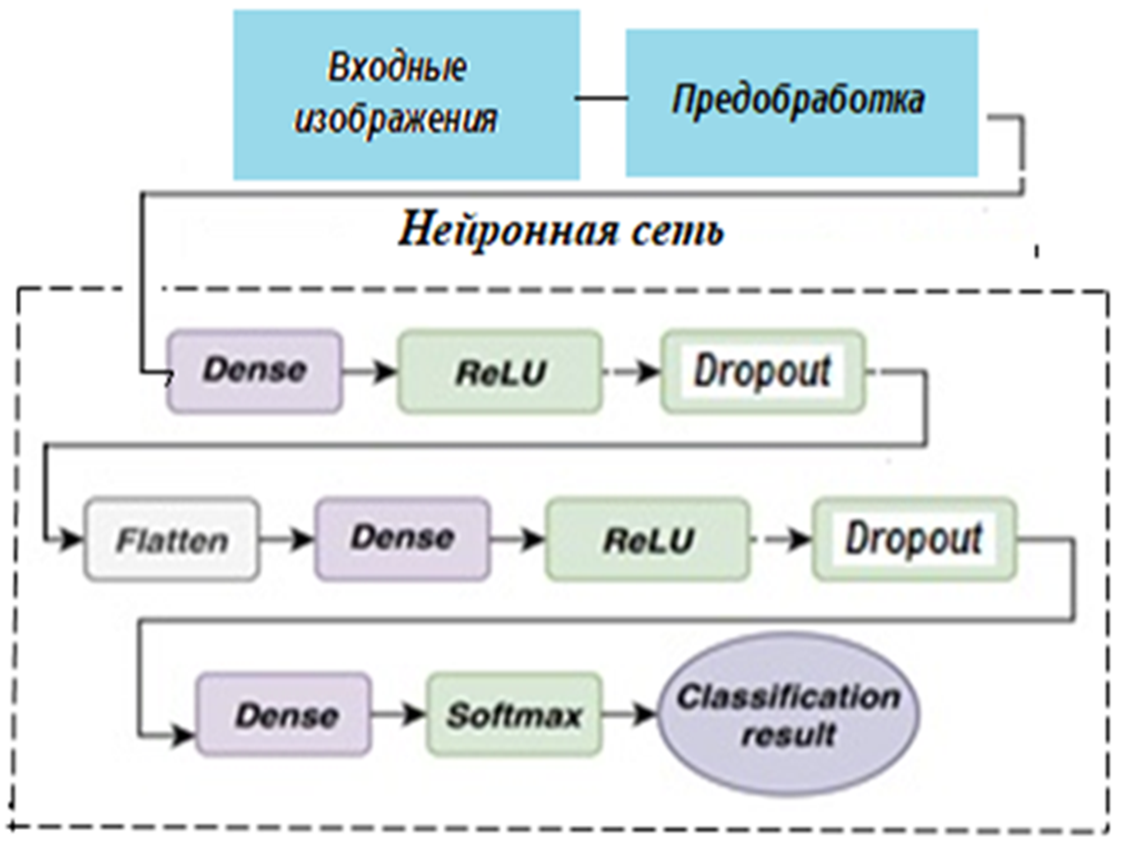


Рис.1

Предобработка выполняет те же функции, что и слои свертки в сверточной нейронной сети – повышение значимости описания изображения при одновременном уменьшении размерности описания изображения. В частности, в блоке предобработки может быть получено цифровое описание изображения.

В качестве цифрового описания изображения листа растения предлагается использовать набор от 4 до 14 параметров Харалика [17, 18] для нескольких, например, 6 или 8 цветовых компонентов изображения, преобразованных в оттенки серого (Grayscale ) возможные значения параметров Харалика в формате вещественных чисел находятся в диапазоне от 0 до 1, в целочисленном формате - в диапазоне от 0 до 255. Таким образом, общее описание одного изображения будет (4 ... 14) x6 = 24 ... 84 байта, что меньше, чем при использовании традиционного подхода, в 1250 ... 357 раз.

Для первоначального обучения нейронной сети можно сформировать набор цифровых данных на основе небольших (около 20) реальных изображений листьев, затем усреднить полученные параметры Харалика и добавить случайные нормально распределенные значения с доверительным интервалом, определенным при моделировании или тестировании. нейронной сети при условии достижения заданного уровня правильного диагноза. Таким образом, можно получить любое количество, например, 1000 различных цифровых описаний для каждой болезни. После этого нейронная сеть будет готова к решению диагностической задачи, и можно будет дополнять датасет и проводить переобучение в процессе работы. Для сокращения времени, затрачиваемого на обучение и диагностику, предлагается предварительно выполнить нормализацию изображения (см. Пример на рис.1).

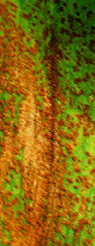
 *A B*



Рис. 1. - Исходное (*A*) и нормализованное (*B*) изображения.

Табл. 1. Усредненные значения параметров Харалика



Для достижения высокого процента правильной диагностики набор усредненных параметров Харалика, рассчитанных по изображениям нескольких листьев, должен быть представлен как цифровое описание целевого изображения. На рис. 2 и 3 показаны примеры нормализованных изображений пшеницы и их цифровых описаний, представленных в виде трехмерных графиков.

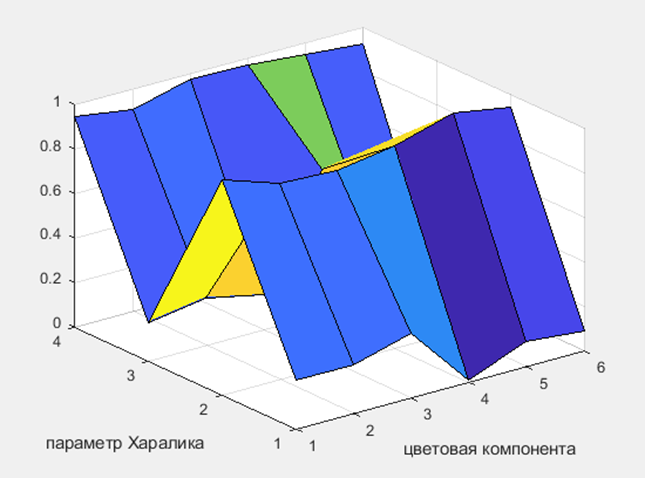




Рис. 2. Нормализованное изображение листа пшеницы, пораженного

болезнью 1, и его цифровое описание в виде трехмерного графика.

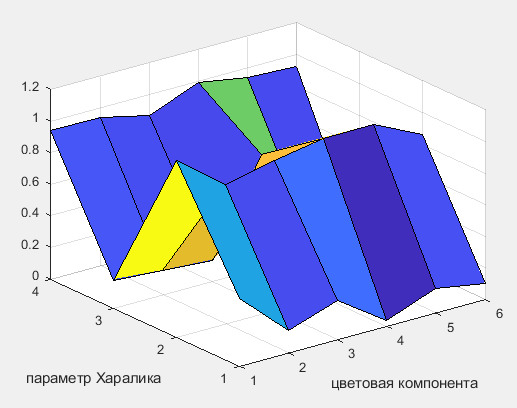
 

Рис.3. Нормализованное изображение листа пшеницы, пораженного болезнью 2, и его цифровое описание в виде трехмерного графика.

Нейронная сеть может использоваться для создания системы распознавания болезней растений на основе трехмерных цифровых описаний.

Для обучения нейронной сети можно создать цифровые описания на основе рандомизации. Для этого используются типовые изображения листьев с признаками различных заболеваний, взятые, например, из энциклопедии или справочника [15, 16], они нормализуются, создается набор данных объемом, например, 1000 цифровых описаний в виде наборов параметров Харалика для каждого заболевания и добавления случайных нормально распределенных значений со значением стандартного отклонения, которое гарантирует правильное распознавание с требуемым уровнем достоверности.

Цифровые описания для обучения, валидации и проверки формируются по такому же принципу.

Матрицы эталонных описаний (наборы параметров Харалика), рассчитанные на основе GLCM - матрицы [17], формируются, в свою очередь, из исходного полутонового изображения.

Первые 4 параметра Харалика [17, 18]:

1. contrast:
2. correlation
3. energy
4. homogeneity

где u, v - координаты матрицы смежности, G - количество уровней серого, μu, μv, σu и σv - средние значения и стандартные отклонения строки u и столбца v матрицы. Максимальное количество параметров Харалика – 14.

**Программа вычисления параметров Харалика**

%Программа вычисления 24-х статистических ключевых признаков

%цветных текстурных изображений ( R, G, B, RG, RB, GB компонентов)

%Вычисление glcm и статистики для цветовых составляющих

%R,G,B (inter\_channel)и для разностных составляющих RG, RB, GB, %(intra\_channel), полученных путем вычитания матриц R, B, G

clc;

clear;

%Обрабатываться будут нормализованные изображения размерностью

%100х300 пикселей

%1-1.bmp-1-20.bmp

he = imread('5-1.png');

%he=imadjust(he,[0.2 0.75],[]);%для выравнивания контрастности нормализованных изображений

%Если выравнивания не делать, то показатели Contrast могут значительно

%отличаться

%выделение r,g,b компонент

r=he(:,:,1);

g=he(:,:,2);

b=he(:,:,3);

%вычисление inter\_channel\_matrix

rgb\_image=im2double(he);%преобразование элементов изображения в формат double

%figure, imshow(he);

fR = rgb\_image (:, :, 1);

fG = rgb\_image (:, :, 2);

fB = rgb\_image (:, :, 3);

%вычисление intra\_channel\_matrix путем вычитания матриц

fRG=fR-fG;

fRB=fR-fB;

fGB=fG-fB;

%Вычисление glcm и статистики для цветовых составляющих

%R,G,B (inter\_channel)

glcm = graycomatrix(fR, 'Offset',[2 0]);%вычисление матрицы glcm

stats\_R = graycoprops(glcm);%вычисление статистических характеристик glcm

glcm = graycomatrix(fG, 'Offset',[2 0]);%вычисление матрицы glcm

stats\_G = graycoprops(glcm);%вычисление статистических характеристик glcm

glcm = graycomatrix(fB, 'Offset',[2 0]);%вычисление матрицы glcm

stats\_B = graycoprops(glcm);%вычисление статистических характеристик glcm

%Вычисление glcm и статистики для разностных составляющих RG, RB, GB, (intra\_channel)

%полученных путем вычитания матриц fR, fB, fG

glcm = graycomatrix(fRG, 'Offset',[2 0]);%вычисление матрицы glcm

stats\_RG = graycoprops(glcm);%вычисление статистических характеристик glcm

glcm = graycomatrix(fRB, 'Offset',[2 0]);%вычисление матрицы glcm

stats\_RB = graycoprops(glcm);%вычисление статистических характеристик glcm

glcm = graycomatrix(fGB, 'Offset',[2 0]);%вычисление матрицы glcm

stats\_GB = graycoprops(glcm);%вычисление статистических характеристик glcm

Rcon=stats\_R.Contrast;

Rcor=stats\_R.Correlation;

Ren=stats\_R.Energy;

Rhom=stats\_R.Homogeneity;

Gcon=stats\_G.Contrast;

Gcor=stats\_G.Correlation;

Gen=stats\_G.Energy;

Ghom=stats\_G.Homogeneity;

Bcon=stats\_B.Contrast;

Bcor=stats\_B.Correlation;

Ben=stats\_B.Energy;

Bhom=stats\_B.Homogeneity;

RGcon=stats\_RG.Contrast;

RGcor=stats\_RG.Correlation;

RGen=stats\_RG.Energy;

RGhom=stats\_RG.Homogeneity;

RBcon=stats\_RB.Contrast;

RBcor=stats\_RB.Correlation;

RBen=stats\_RB.Energy;

RBhom=stats\_RB.Homogeneity;

GBcon=stats\_GB.Contrast;

GBcor=stats\_GB.Correlation;

GBen=stats\_GB.Energy;

GBhom=stats\_GB.Homogeneity;

A=[Rcon Rcor Ren Rhom Gcon Gcor Gen Ghom Bcon Bcor Ben Bhom RGcon RGcor RGen RGhom RBcon RBcor RBen RBhom GBcon GBcor GBen GBhom]

xlswrite('res3D.xlsx',A) %передача вектор-строки параметров Харалика в Excel

%Вектор-строку можно преобразовать в Excel в вектор-столбец с помощью операции транспонирования (ТРАНСП)

%pause;

close all;

clear;

**Цель работы:** научиться оценивать влияние количества цветовых компонентов изображений, параметров Харалика при создании цветовых описаний текстурных изображений листьев и количества цифровых описаний в обучающем наборе на качество обучения нейронной сети.

**Базовые программные средства**

Программа *digital\_description\_generator* создает с использованием рандомизации 150 файлов цифровых описаний в формате .png. для каждой из болезней растений. Количество болезней - 8, количество реализаций для каждой болезни - 150

1. Глобальные папки: *train\_dir, test\_dir, val\_dir*, в каждой глобальной папке - 8 вложенных папок (по числу болезней), в каждой вложенной папке -

* графические файлы: в *train\_dir* - файлы с именами 1-100;
* в *test\_dir* - файлы с именами 101-120;
* в *val\_dir* - файлы с именами 121-150;

До запуска программы должна быть создана папка D:\DDNN с вложенными пустыми папками *train\_dir, test\_dir, val\_dir,* вкаждой из которых должны содержаться пустые папки 1-8 по количеству возможных болезней..

После окончания генерации полученные файлы цифровых описаний вместе с папками нужно перенести в рабочую папку Python\_150\_digit с программой нейронной сети.

*Текст программы*

clc;

clear;

j=1 ; %номер болезни

%m1;%начало блока цифровых описаний.

%m2;%конец блока цифровых описаний.

S=0.032; %СКО случайных отклонений от эталонных значений

% x = 1:8;%номера возможных болезней

%Подлинные результаты измерений параметров Contrast, Correlation,

%Energy,Homogenity для компонентов R,G,B,RG,RB,GB (Считаем их эталонными описаниями для 8 болезней). Вместо этого массива данных нужно

%использовать массив значений, усредненных по обучающей выборке

Rcon = [0.2457 0.124 0.3154 0.2008 0.3102 0.2077 0.2023 0.1599];

Rcor = [0.9497 0.9439 0.8911 0.8028 0.8128 0.9657 0.937 0.9296];

Ren = [0.1195 0.2224 0.1504 0.3312 0.1897 0.1092 0.2122 0.2347 3];

Rhom = [0.9005 0.9471 0.8596 0.925 0.8669 0.9064 0.9062 0.923];

Gcon = [0.2055 0.1204 0.2809 0.2413 0.3021 0.1782 0.2229 0.1494];

Gcor = [0.9126 0.9436 0.8666 0.7935 0.8201 0.9628 0.9378 0.9377];

Gen = [0.1816 0.2485 0.2108 0.2585 0.1797 0.132 0.1415 0.298];

Ghom = [0.9154 0.9477 0.8758 0.9046 0.8637 0.9188 0.8949 0.9281];

Bcon = [0.221 0.1095 0.3115 0.2527 0.313 0.1799 0.1791 0.1634 5];

Bcor = [0.9418 0.9412 0.8015 0.7478 0.8167 0.96 0.8436 0.96];

Ben = [0.143 0.2467 0.2115 0.3014 0.1764 0.1765 0.36 0.3596 ];

Bhom = [0.9115 0.9553 0.8613 0.9102 0.8541 0.917 0.9133 0.9206];

RGcon = [0.0325 0.0272 0.0068 0.0716 0.2173 0.0436 0.0047 0.0353];

RGcor = [0.9443 0.8913 0.7202 0.8351 0.7942 0.8913 0.7728 0.9363];

RGen = [0.7012 0.7232 0.9689 0.5689 0.2338 0.6744 0.9748 0.4623];

RGhom = [0.9838 0.9864 0.9966 0.9642 0.8922 0.9782 0.9977 0.9823];

RBcon = [0.1206 0.0796 0.0778 0.1414 0.4523 0.1227 0.0939 0.1122];

RBcor = [0.9804 0.9222 0.8669 0.9139 0.7994 0.954 0.9383 0.9468];

RBen = [0.2187 0.3535 0.5046 0.2234 0.1276 0.2023 0.3236 0.2454];

RBhom = [0.9401 0.9602 0.9611 0.9305 0.8144 0.9388 0.9532 0.9439];

GBcon = [0.1205 0.0626 0.0764 0.0842 0.3176 0.0818 0.1041 0.1037];

GBcor = [0.964 0.9218 0.8665 0.8384 0.7138 0.9187 0.9267 0.9192];

GBen = [0.1983 0.3974 0.4506 0.4231 0.2515 0.3384 0.2669 0.3425];

GBhom = [0.94 0.9687 0.9618 0.9583 0.8518 0.9591 0.9484 0.9482];

% name='train';

% eval(['!mkdir ' name]);создание папки с именем name

for j=1:8

for mm=1:3

if mm==1

m1=1;m2=100;

end

if mm==2

m1=101;m2=120;

end

if mm==3

m1=121;m2=150;

end

for j1=m1:m2 %цикл генерации цифровых описаний (ЦО).

% к усредненым значениям параметров Харалика добавляется нормально распределенный шум

noise=randn(1,24);

XRcon(j)=Rcon(j)+S\*noise(1);XRcor(j)=Rcor(j)+S\*noise(2);XRen(j)=Ren(j)+S\*noise(3);XRhom(j)=Rhom(j)+S\*noise(4); XGcon(j)=Gcon(j)+S\*noise(5);XGcor(j)=Gcor(j)+S\*noise(6);XGen(j)=Gen(j)+S\*noise(7);XGhom(j)=Ghom(j)+S\*noise(8); XBcon(j)=Bcon(j)+S\*noise(9);XBcor(j)=Bcor(j)+S\*noise(10);XBen(j)=Ben(j)+S\*noise(11);XBhom(j)=Bhom(j)+S\*noise(12); XRGcon(j)=RGcon(j)+S\*noise(13);XRGcor(j)=RGcor(j)+S\*noise(14);XRGen(j)=RGen(j)+S\*noise(15);XRGhom(j)=RGhom(j)+S\*noise(16); XRBcon(j)=RBcon(j)+S\*noise(17);XRBcor(j)=RBcor(j)+S\*noise(18);XRBen(j)=RBen(j)+S\*noise(19);XRBhom(j)=RBhom(j)+S\*noise(20);

XGBcon(j)=GBcon(j)+S\*noise(21);XGBcor(j)=GBcor(j)+S\*noise(22);XGBen(j)=GBen(j)+S\*noise(23);XGBhom(j)=GBhom(j)+S\*noise(24);

%end

DD1(1,1)=XRcon(j);DD1(2,1)=XRcor(j);DD1(3,1)=XRen(j);DD1(4,1)=XRhom(j);

DD1(1,2)=XGcon(j);DD1(2,2)=XGcor(j);DD1(3,2)=XGen(j);DD1(4,2)=XGhom(j);

DD1(1,3)=XBcon(j);DD1(2,3)=XBcor(j);DD1(3,3)=XBen(j);DD1(4,3)=XBhom(j);

DD1(1,4)=XRGcon(j);DD1(2,4)=XRGcor(j);DD1(3,4)=XRGen(j);DD1(4,4)=XRGhom(j);

DD1(1,5)=XRBcon(j);DD1(2,5)=XRBcor(j);DD1(3,5)=XRBen(j);DD1(4,5)=XRBhom(j);

DD1(1,6)=XGBcon(j);DD1(2,6)=XGBcor(j);DD1(3,3)=XGBen(j);DD1(4,6)=XGBhom(j);

A=DD1; %Это-матрица описания цифрового текстурного изображения. Она имеет такой же вид, как и цифровое изображение в формате Grayscale

%Запись матрицы в файл графического формата png

I=mat2gray(A);

k=num2str(j);%перевод номера болезни в текстовый формат

if j==1

if m2==100

path='D:\DDNN\train\_dir\1\';

end

if m2==120

path='D:\DDNN\test\_dir\1\';

end

if m2==150

path='D:\DDNN\val\_dir\1\';

end

end

if j==2

if m2==100

path='D:\DDNN\train\_dir\2\';

end

if m2==120

path='D:\DDNN\test\_dir\2\';

end

if m2==150

path='D:\DDNN\val\_dir\2\';

end

end

if j==3

if m2==100

path='D:\DDNN\train\_dir\3\';

end

if m2==120

path='D:\DDNN\test\_dir\3\';

end

if m2==150

path='D:\DDNN\val\_dir\3\';

end

end

if j==4

if m2==100

path='D:\DDNN\train\_dir\4\';

end

if m2==120

path='D:\DDNN\test\_dir\4\';

end

if m2==150

path='D:\DDNN\val\_dir\4\';

end

end

if j==5

if m2==100

path='D:\DDNN\train\_dir\5\';

end

if m2==120

path='D:\DDNN\test\_dir\5\';

end

if m2==150

path='D:\DDNN\val\_dir\5\';

end

end

if j==6

if m2==100

path='D:\DDNN\train\_dir\6\';

end

if m2==120

path='D:\DDNN\test\_dir\6\';

end

if m2==150

path='D:\DDNN\val\_dir\6\';

end

end

if j==7

if m2==100

path='D:\DDNN\train\_dir\7\';

end

if m2==120

path='D:\DDNN\test\_dir\7\';

end

if m2==150

path='D:\DDNN\val\_dir\7\';

end

end

if j==8

if m2==100

path='D:\DDNN\train\_dir\8\';

end

if m2==120

path='D:\DDNN\test\_dir\8\';

end

if m2==150

path='D:\DDNN\val\_dir\8\';

end

end

n=num2str(j1);%перевод номера ЦО в текстовый формат

fname=strcat(path, n,'.png');

fid = fopen(fname,'w+');

%fprintf(fid,'%0.10f\n',I1);%запись в текстовый файл

imwrite(I,fname);

fclose('all');

% Представление матрицы цифрового описания в виде 3D графического объекта

% figure;

% colormap('default');

% % surf(I)

% meshz(A);

% xlabel('цветовая компонента');

% ylabel('параметр Харалика');

end

end

disp('Готово');

pause;

close all;

clear;

**Текст обучающей программы прямого распространения Python\_150\_NNF**

#Программа нейронной сети прямого распространения для распознавания 6-ти #болезней пшеницы по 3D-цифровым описаниям изображений листьев.

#количество цифровых компонентов - 6 (R,G,B,RG,RB,GB)

#количество параметров Харалика - 4.

#количество цифровых описаний для обучения – 100, для проверки – 30, для тестирования – 20.

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

from tensorflow.python.keras.models import Sequential

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 тестирования

val\_dir = 'val\_dir'

# Размеры изображения

img\_width, img\_height = 4, 6

# 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(Dense(20, input\_shape=input\_shape))

model.add(Activation('relu'))

model.add(Flatten())

model.add(Dense(20))

model.add(Activation('relu'))

model.add(Dropout(0.25))

model.add(Dense(8))

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('plant\_diagnosis\_3D.h5')

scores = model.evaluate\_generator(test\_generator, nb\_test\_samples // batch\_size)

print("Score: ", scores)

**Программа работы**

1. Исследуйте влияние перехода от сверточной нейронной сети к нейронной сети прямого распространения на время обучения.
2. Исследуйте влияние количества цифровых описаний, предназначенных для обучения, на итоговые показатели точность и потерь (значения acc и loss).

**Указания по выполнению работы**

1. Запишите значения параметров, заданных в базовой программе: количество эпох обучения, размер мини-выборки, количество цифровых описаний изображений, время обучения и результаты работы обучающей программы (значения acc и loss) для массива данных *test* при оптимальном количестве эпох. Сравните время обучения с аналогичным показателем в сверточной нейронной сети, полученном при выполнении работы 2-2.
2. Изменяйте количество цифровых описаний изображений из набора: 300, 600, 900, 1500 в папке *train* и регистрируйте время обучения и результаты работы обучающей программы (значения acc и loss) для массива данных *test*.

**Содержание отчета**

1. Цель работы.
2. Программа работы.
3. Таблицы с результатами измерений по пунктам программы.
4. Промежуточные результаты работы по пунктам программы.

Выводы.