# Реализация нейронной сети FFNN для анализа и классификации геометрических фигур

## Условие

Написать программу моделирования нейронных сетей (НС) заданного типа и показать их работоспособность на практических примерах использования НС для указанной задачи. Параметры НС представлены в таблице 1.

Таблица 1. Параметры модели

| | |
|---|---|
| **Вход** | Изображения на некотором фоне одной из 5 геометрических фигур |
| **Выход** | Какая фигура |
| **Тип НС** | `FFNN` |
| `Ns` | Число элементов в скрытом слое |

## 1 Постановка задачи, связанной с практическим применением НС

Компьютерное зрение на основе методов распознавания геометрической формы получило широкое распространие на производстве в таких областях как промышленный осмотр, идентификация, и автоматический контроль качества продукции. В задаче автоматической сборки, значительный объем информации о детали необходимо распознавать и классифицировать, а ее ориентация должна быть автоматически определена, прежде чем робот (манипулятор) сможет ухватиться за изделие или его часть. Также применяются методы распознавания формы для оптического распознавания символов, рукописного текста, а также медицинских изображений, и.т.д. [4].

*Базовой задачей*, предваряющей перечисленные выше, является задача распознавания и классификации плоских геометрических фигур таких как треугольник, квадрат, пятиугольник, шестиугольник и круг (многоугольник с количеством сторон $\sim 100$).

## 2 Описание теоретической базы рассматриваемой модели НС

Нейронные сети прямого распространения (feed forward neural networks, `FF` или `FFNN`) передают информацию от входа к выходу [1]. Сети `FFNN`, описываются в виде набора слоёв клеток (нейронов). Причём, существуют входные, скрытые и выходные слои (рис. 1) Нейроны одного слоя не связаны между собой, а соседние слои обычно связаны полностью (каждый с каждым). В данной работе будем рассматривать только полносвязные `FFNN`.

Самая простая нейронная сеть имеет два входных нейрона и один выходной, и может использоваться, например, в качестве модели логических вентилей. `FFNN` обычно обучается по методу *обратного распространения ошибки*, в котором сеть получает множества входных и выходных данных (множества пар: вектор входных значений; соответствующий ему вектор выходных значений). Этот процесс называется обучением с учителем, и

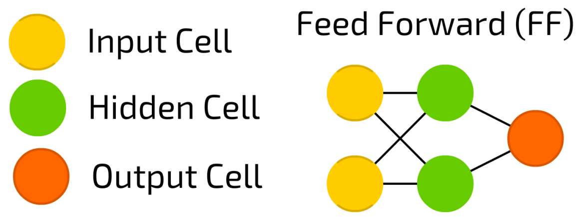


Рис. 1. НС прямого распространения [2]

он отличается от обучения без учителя тем, что во втором случае множество выходных данных сеть составляет самостоятельно.

Упомянутая выше *ошибка* является разницей между вводом и выводом. Если у сети есть достаточное количество скрытых нейронов, она теоретически способна смоделировать взаимодействие между входным и выходными данными.

На практике такие сети используются редко, но их часто комбинируют с другими типами для получения новых.

Рассмотрим вычисления в рамках одного нейрона `FFNN`.

### Активация нейрона и вычисление выхода

Активация нейрона происходит по правилу (1) в векторном виде

$$a(x) = b + \mathbf{w}^T\mathbf{x}. \tag{1}$$

Здесь $\mathbf{x}$ — входной вектор, $\mathbf{w}$ — вектор весовых коэффициентов, $b$ — отклонение (bias).

Активационные функции $g(x)$ бывают разных видов. Наиболее распрострненными являются *сигмоида*, приведена на рис. 2,

$$g(x) = \operatorname{sigm} x = \frac{1}{1 + \exp(-x)}, \tag{2}$$

и *гиперболический тангенс*, на рис. 3

$$g(x) = \tanh x = \frac{\exp(2x) - 1}{\exp(2x) + 1}. \tag{3}$$

В качестве активационной функции также используется функция (4) вида

$$g(x) = \begin{cases} 1, & \mathbf{w}^T\mathbf{x} > \varepsilon, \\ 0, & \mathbf{w}^T\mathbf{x} \geqslant \varepsilon, \end{cases} \tag{4}$$

где $\varepsilon$ — пороговое значение (`treshold`).

Результат (конечный выход нейрона) $h(x)$ вычислятся (5) подстановкой

$$h(x) = g(a(x)) = g(b + \mathbf{w}^T\mathbf{x}). \tag{5}$$

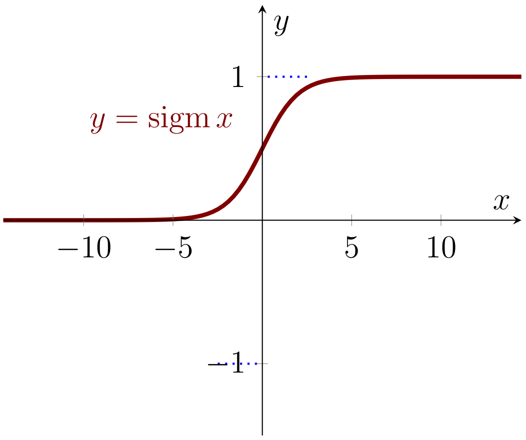


Рис. 2. Сигмоида

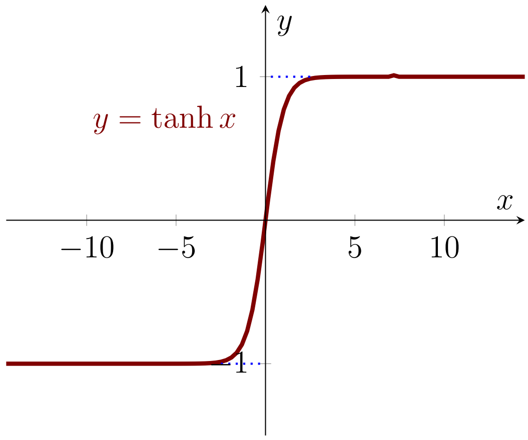


Рис. 3. Гиперболический тангенс

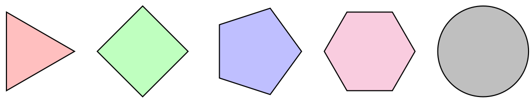

Рис. 4. n-угольники

# 3 Описание разработанных ПМ и решение задачи

Для решения поставленной задачи распознавания геометрических фигур на фоне различных цветов с помощью `FFNN` нейронной сети реализуем обучающую и тестовую выборки изображений. В качестве геометрических фигур рассмотрим правильные $n$–угольники. Для $n = 3$ — треугольник, $n = 4$ — квадрат, $n = 5$ — пятиугольник, $n = 6$ — шестиугольник и $n = 100$ — круг (представлены на рис. 4).

Был реализован алгоритм, позволяющий получить изображения указанных геометрических фигур различных размеров и цветов. Предусмотрена возможность поворота фигуры на некоторый случайный угол.

**Файлы** `Matlab`**:**

`gen_set_of_ngons.m`
`gen_ngon_image.m`
`run_gen_ngon.m`

После того, как изображения были сгенерироны и сохранены на жестком диске, скомпануем по ним сами выборки.

**Выборка № 1 ”Сложная”**

Всего 70 тысяч примеров – изображений размером 28×28 пикселей, по 14 тысяч на каждую фигуру. При этом 60 тысяч отнесем к обучающей выборке, а оставшиеся 10 – к тестовой (валидационной).

Фигуры расположены произвольно относительно центра изображения, произведён поворот на случайный угол. Цвет фигуры и цвет фона также случайные, см. рис. 5.

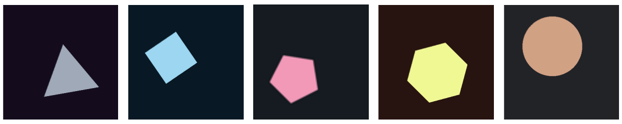

Рис. 5. Примеры сгенерированных фигур

Все цветные изображения переведём в оттенки серого и преобразуем в одномерный массив длиной 784 элемента. Размер выборки такой же, как и в задаче распознавания `FFNN`, – сетью рукописных цифр MNIST. В типовом решении последней задачи используются следующие параметры персептрона:

- число входов нейронной сети: 784 ($= 28 \times 28$);
- число нейронов в скрытом слое: 100;
- число выходных нейронов: 10 (по одному на каждую цифру) с выходным значением от 0 до 1.

Такая сеть тренируется на парах $(I_j, v_j)$, где $I_j$ — $j$-е изображение цифры, $v_j$ — соответствующий ему вектор–метка. Будем использовать те же параметры.
В качестве вектора $v_j$ примем: *треугольник* – [1, 0, 0, 0, 0]; *квадрат* – [0, 1, 0, 0, 0]; *круг* – [0, 0, 1, 0, 0]; *пятиугольник* – [0, 0, 0, 1, 0]; *шестиугольник* – [0, 0, 0, 0, 1].

### Выборка № 2 "Упрощённая"

Содержит 20 тысяч примеров фигур с фиксированными положением, размером и цветом. От изображения к изображению меняется только цвет фона. Данная выборка представляет собой максимально упрощённый вариант выборки № 1. Как и ранее, все изображения переводятся в оттенки серого и преобразуются в одномерный массив длиной 784 элемента.

## Обучение

Для достижения приемлемых результатов распознавания, необходима дополнительная предобработка полученных выборок изображений. До приведения их к одномерному массиву, из общих соображений, желательно минимизировать пространство, занимаемое фоном. Также фигуры желательно центрировать. После преобразования в массив – необходимо привести имеющиеся значения к единому диапазону, например [0, 1] или [-1, 1].
Одним из наиболее простых и быстрых решений для дополнительной предобработки обучающей выборки в данной задаче, по мнению автора, является пробразование Фурье. В `Matlab`:

```
Y = fft2(I); % двумерное преобразование Фурье.\\
% I - матрица/изображение -> Y - спектр.
```

Результатом данной операции является матрица `complex double`. Разместим нулевую компоненту в центре спектра, а затем возьмём модуль комплексного числа для дальнейшей работы с действительными значениями:

```
I = abs(fftshift(Y));
```

Рассмотрим спектры изображений треугольника, квадрата, и круга, соответвтенно (представлены на рис. 6)

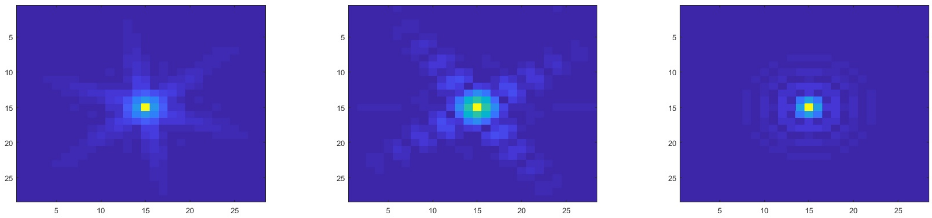

Рис. 6. Спектры изображений некоторых фигур

**Файл** `Matlab`**:**

`load_dataset.m`

В данной работе производились попытки обучить `FFNN`– сеть как на самих изображениях (без использования преобразования Фурье), так и на 2D спектрах этих изображений. В обоих случаях конфигурация нейронной сети совпадала.

Приведем полученную статистику распознавания на множестве примеров, не входивших в число тех, что были использованы непосредственно в процессе обучения. Сгенерируем новые примеры отдельно от существующих обучающих и валидационных выборок.

**Файл** `Matlab`**:**

`load_testdataset.m`

### ”Упрощённый” случай

Конфигурация: 784 входа, 100 нейронов в скрытом слое, 5 выходных нейронов. `TrainFcn`: градиентный спуск. Функция активации нейронов скрытого слоя: сигмоида.

Для успешного обучения по упрощённой выборке (№ 2) достаточно использовать небольшое число обучающих примеров. Здесь не возникает необходимости, например, с целью улучшения качества распознавания добавлять к исходным данным изображений какие-либо дополнительные характеристики и коэффициенты. Предложенный подход с Фурье-преобразованием не даёт лучшие результаты по сравнению с простым подходом, заключающимся в прямой подаче развёрнутого изображения на вход персептрона. Процент распознавания близок к 100%.

### ”Сложный” случай

Конфигурация: 784 входа, 100 нейронов в скрытом слое, 5 выходных нейронов. `TrainFcn`: градиентный спуск. Функция активации нейронов скрытого слоя: сигмоида.

| | |
|---|---|
| Размер обучающей выборки: 5000 ед. | |
| Обучение на изображениях | 297/1000 примеров распознано корректно<br>703 — некорректно |
| Обучение на Фурье-спектрах | 802/1000 примеров распознано корректно<br>198 — некорректно |
| Размер обучающей выборки: 70000 ед. | |
| Обучение на изображениях (рис. 7) | 291/1000 примеров распознано корректно<br>709 — некорректно |
| Обучение на изображениях (рис. 8) | 710/1000 примеров распознано корректно<br>290 — некорректно |

**Файл в Matlab:**

`train_ffnn.m`

Результаты расчетов показывают, что в "сложном" случае подход с Фурье – преобразованием заметно выигрывает.

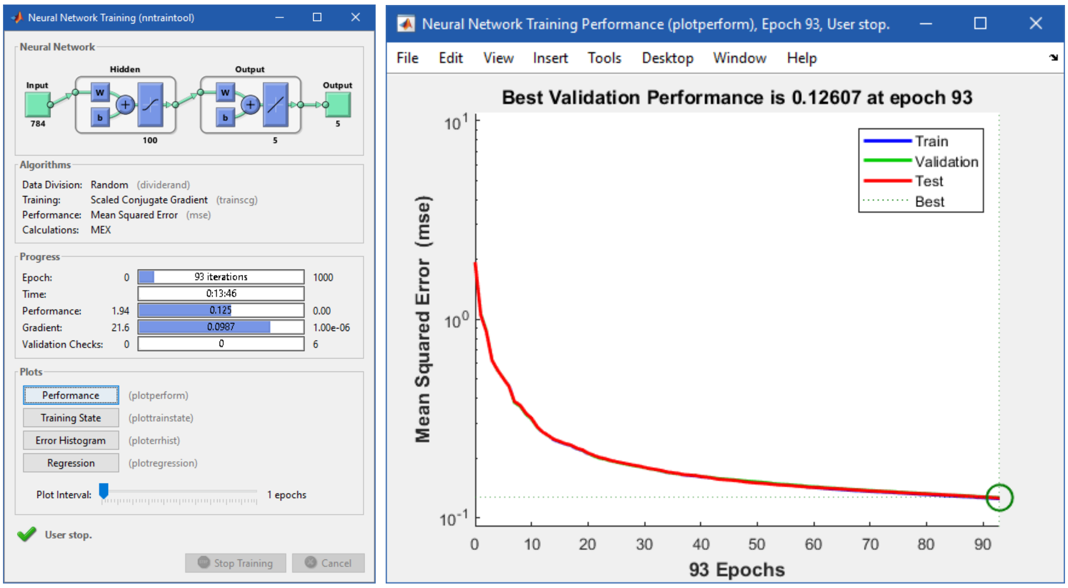


Рис. 7. Инструменты MATLAB. Обучение на изображениях (70000 примеров)

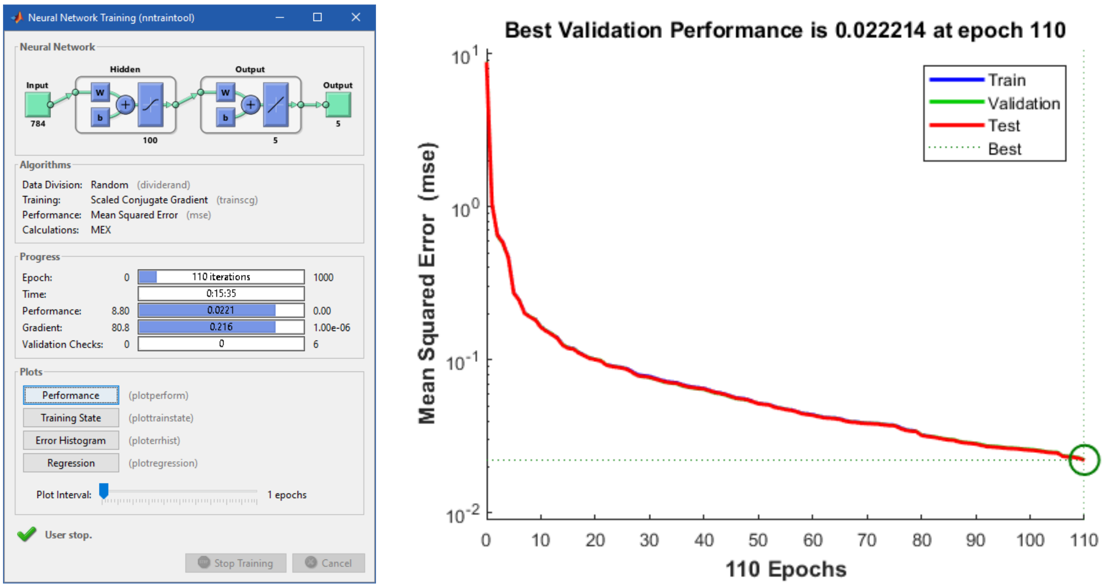


Рис. 8. Инструменты MATLAB. Обучение на Фурье–спектрах (70000 примеров)

## Третий подход

Имея на руках два двумерных массива ($2 \times 28 \times 28$) получим набор из 112–ти коэффициентов по следующему правилу:

| | | |
|---|---|---|
| 1й коэфф. | - | сумма всех элементов 1й строки 1го массива |
| 2й коэфф. | - | сумма всех элементов 2й строки 1го массива |
| 28й коэфф. | - | сумма всех элементов 28й строки 1го массива |
| 29й коэфф. | - | сумма всех элементов 1го столбца 1го массива |
| ... | - | ... |
| 56й коэфф. | - | сумма всех элементов 28го столбца 1го массива |
| 57й коэфф. | - | сумма всех элементов 1й строки 2го массива |
| ... | - | ... |
| 112й коэфф. | - | сумма всех элементов 28го столбца 2го массива |

Для каждого рассматриваемого $n$ – угольника набор полученных таким образом коэффициентов будет отличаться. Например, для круга (рис. 9):

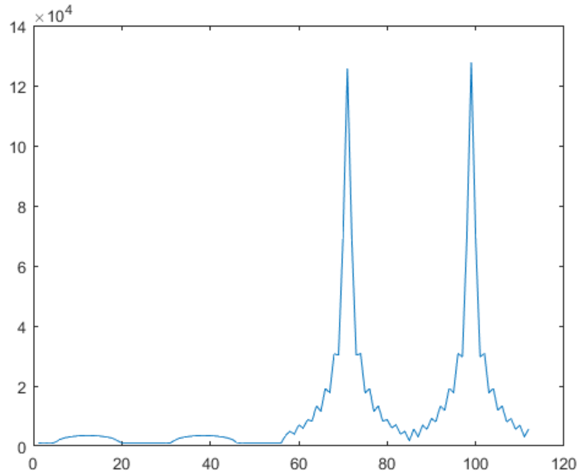


Рис. 9. Введённые коэффициенты для круга

Тогда как для шестиугольника (рис. 10):

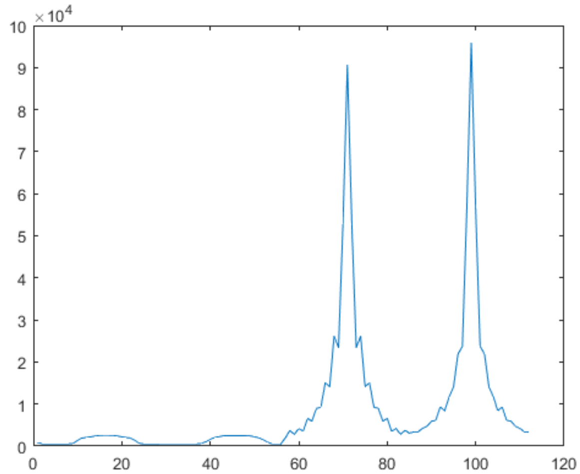


Рис. 10. Введённые коэффициенты для шестиугольника

Обрезав значения по уровню $3 \cdot 10^4$ и отнормировав коэффициенты в диапазоне [0, 1], обучим на полученных данных нейронную сеть следующей конфигурации: 112 входов, 56 нейронов в скрытом слое, 5 выходных нейронов. Метод: градиентный спуск. Функция активации нейронов скрытого слоя: сигмоида.
Результаты распознавания следующие:

| | |
|---|---|
| Размер обучающей выборки: 5000 ед. | |
| Обучение на изображениях | 297/1000 примеров распознано корректно<br>703 — некорректно |
| Третий подход | 738/1000 примеров распознано корректно<br>262 — некорректно |
| Размер обучающей выборки: 70000 ед. | |
| Обучение на изображениях | 297/1000 примеров распознано корректно<br>703 — некорректно |
| Обучение на Фурье-спектрах | 736/1000 примеров распознано корректно<br>264 — некорректно |

## Дополнительно

При решении многих задач классификации имеет смысл разбить исходную задачу на множество подзадач. Сеть, классифицирующая имеющиеся данные по 3–ем классам может быть заменена на 3 сети, решающими задачу классификации по 2–м классах. Для задачи с $k$ классами количество подзадач составит (из комбинаторики):

$$C_n^k = \frac{n!}{k!(n-k)!}.$$

То есть, в нашем случае для 5–ти классов имеем 10 подзадач: На рис. 11 приведены три из них.

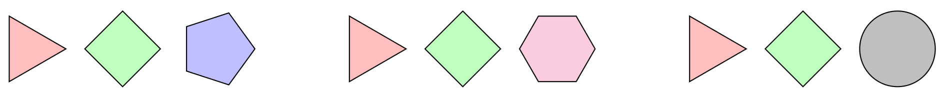

Рис. 11. Первые три из десяти подзадач

# 4 Альтернативные способы решения

Альтернативными являются два подхода классификации фигур, которые во многом противоположны друг другу.
**Первым подходом** является подход на основе спектрального анализа [5]. Представление формы объекта на основе *Фурье дескрипторов* легко организовать в плане вычислений, а результаты будут устойчивы к внешнему шуму. Фурье дескрипторы получаются при помощи преобразования Фурье, применённого к *сигнатуре формы объекта*, границе объекта как одномерной функции.

Существуют и другие сигнатуры формы, например, расстояние от центра объекта (в пикселях) до остальных пикселей, кривизна границы и кумулятивный угол. Геометрические характеристики (инварианты) объекта определяются на этапе определения сигнатуры формы объекта, который может происходить как до Фурье преобразования, так и после. При этом, нижние частоты дескрипторов содержат информацию о форме объекта, а верхние частоты о деталях.
**Вторым подходом** является геометрический подход, который предполагает выделение контура [6]. Он основывается на разработке дескрипторов формы малого разрешения, которые могут быть устойчивы к поворотам, масштабированию и деформации объекта.

В таком подходе граница объекта разбивается на составные части (сегменты), каждая из которых далее описывается при помощи того или иного дескриптора. После чего для

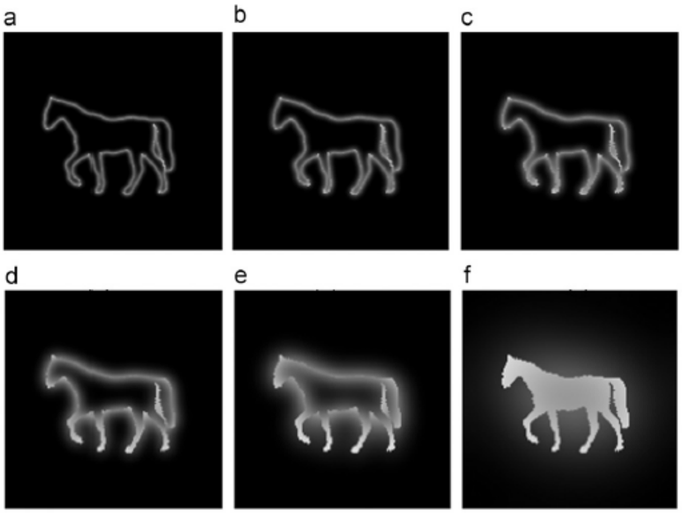


Рис. 12. Пример обработки изображения лошади [5]

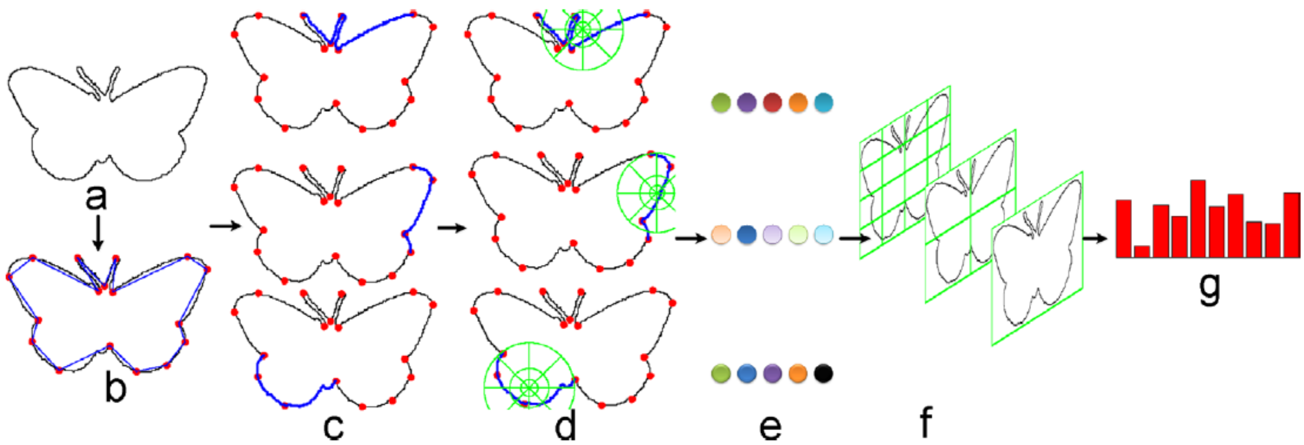


Рис. 13. Пример обработки изображения бабочки [6]

решения задачи классификации применяются методы машинного обучения, например, метод опорных векторов [7].

# 5 Области применения НС заданного типа

НС `FFNN` применяются везде, где встречаются простые задачи кластеризации. Также, `FFNN` применимы в качестве простейших фильтров для обработки как изображений, так и одномерных сигналов.

Для большей производительности и улучшения качества распознавания/кластеризации в `FFNN`сети добавляют множество свёрточных слоёв.

# Список литературы

[1] F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. Psychological Review, 65(6), 1958, pp. 386–408. **DOI: 10.1037/h0042519**

[2] Шпаргалка по разновидностям нейронных сетей. Часть первая. Элементарные конфигурации. **URL:** https://tproger.ru/translations/neural-network-zoo-1/

[3] A simple example of feedforward neural network and image recognition. **URL:** https://dummas.wordpress.com/2012/01/14/a-simple-example-of-feedforward-neural-network-with-image-recognition/

[4] Tung-Hsu Hou and Ming-Der Pern, A Computer Vision-Based Shape-Classification System Using Image Projection and a Neural Network. Int. J. Adv. Manuf. Technol., 15, 1999, pp. 843–850. **DOI: 10.1007/s001700050141**

[5] Cem Direkoğlu, Mark S. Nixon, Shape classification via image-based multiscale description. Pattern Recognition, 44, 2011, pp. 2134–2146. **DOI: 10.1016/j.patcog.2011.02.016**

[6] Xinggang Wang, Bin Feng, et.al., Bag of contour fragments for robust shape classification. Pattern Recognition, 47, 2014, pp. 2116–2125. **DOI: 10.1016/j.patcog.2013.12.008**

[7] К.В. Воронцов. Лекции по методу опорных векторов. **URL:** http://www.ccas.ru/voron/download/SVM.pdf

# Тексты программ

gen_set_of_ngons

```
function gen_set_of_ngons(number_of_images, nS, folder, baseFileName, shift)

% for k = 1:number_of_images
% shift = 20000;
parfor k = 1+shift:number_of_images+shift
img = gen_ngon_image(nS);
fullFileName = [folder  baseFileName '-' int2str(k)  '.png'];
save_image(folder,baseFileName,fullFileName,img.cdata);
disp([int2str(k) '/' int2str(number_of_images+shift)])
end %for k
```

gen_ngon_image.m

```
function out = gen_ngon_image(n)

%% Suppose that domain is [0 100]x[0 100] square
dom = [0 100 0 100];

% Center of n-polygon
xC = randi([35 65],1,1);
yC = randi([35 65],1,1);
center = [xC yC];
nS = n; % number of sides of n-gon
th = linspace(0, 2*pi, nS + 1);
% Rotate the shape by subtracting an offset.
rot = randi([1 20],1,1);
th = th - pi/rot;
R = randi([20 30],1,1);
x = R * cos(th) + center(1);
y = R * sin(th) + center(2);

%% Show image
figure_color         = 0.5 + 0.5 * rand(1,3);
% figure_color = [0 0 1];
background_color     = 0.2*rand(1,3);

h = fill(x, y, figure_color);
set(h,'edgecolor',figure_color);
ax = gca;
set(ax,'xtick',[]); set(ax,'ytick',[]);

ax.XColor = background_color; ax.YColor = background_color ;
axis square; axis(dom);
set(ax,'Color',background_color )

img = getframe(gca);
out = img;
```

run_gen_ngons.m

```
clear all; close all; clc;

% Get path
```

```
CurrentFolder = pwd;


% Generate ngon image
number_of_images = 15000;
shift = 20000;

% Generate set of 3-gons
nS = 4;
folder = [CurrentFolder '\_img_\' int2str(nS) '-gone\'];
mkdir_if_not_exist(folder);
baseFileName = [int2str(nS) '-gone'];
gen_set_of_ngons(number_of_images, nS, folder, baseFileName, shift);
```

`load_dataset.m`

```
triangleFolder = '_img_/3-gone/';       % label: 1
rectangleFolder = '_img_/4-gone/';      % label: 2
circleFolder = '_img_/100-gone/';       % label: 3
pentagonFolder = '_img_/5-gone/';       % label: 4
hexagonFolder = '_img_/6-gone/';        % label: 5

% get list of files
dirData = dir(triangleFolder);
dirIndex = [dirData.isdir];
fileListTriangles = {dirData(¬dirIndex).name}';
...

% number of elements in training dataset
N = 14000;

% initialization of image arrays
trianglesI = zeros(784, N);
rectanglesI = zeros(784, N);
circlesI = zeros(784, N);
...
% initialization of labels
trianglesL = zeros(1, N);
rectanglesL = zeros(1, N);
...

% read data from file
k = 1;
for i = 1:N
fpath = strcat(triangleFolder, fileListTriangles{i});
disp(fpath);
I = imread(fpath);
I = imresize(I, [28 28]); % resize to 28x28 domain
I = fft2(I); % make FFT
% reshape to 1D array
% put 0 component to the center
trianglesI(:, k) = reshape(abs(fftshift(I(:,:,1))), [784,1]);
% put labels
trianglesL(k) = 1;
k = k + 1;

...
end

```

```
N3 = 3 * N;      % 42 000
N5 = 5 * N;      % 70 000
% reinitialization of training dataset no.1
DataSet_Images_Total = zeros(1, 784, N5); % all images
DataSet_Labels_Total = zeros(1, 5, N5); % labels
DataSet_Labels_1_Total = zeros(1, 1, N5);

% reinitialization of training dataset no.2
DataSet_Images = zeros(10, 784, N3); % images (10 gropus/subtasks)
DataSet_Labels = zeros(10, 3, N3);   % labels
DataSet_Labels_1 = zeros(10, 1, N3);

% Fill training dataset no.1 (without separation on subtasks)
k = 1;
for i = 1:N
DataSet_Images_Total(:, k) = trianglesI(:, i);

% Scalar labels (FFNN with 1 output)
% 1 - 3-gone, 2 - 4-gone, 3 - 100-gone,
% 4 - 5-gone, 5 - 6-gone
DataSet_Labels_1_Total(k) = 1;

% Vector labels (FFNN with 5 output)
DataSet_Labels_Total(1,:,k) = [1 0 0 0 0];
k = k + 1;

...
DataSet_Images_Total(:, k) = pentagonesI(:, i);
DataSet_Labels_1_Total(k) = 4;
DataSet_Labels_Total(1,:,k) = [0 0 0 1 0];
k = k + 1;
DataSet_Images_Total(:, k) = hexagonesI(:, i);
DataSet_Labels_1_Total(k) = 5;
DataSet_Labels_Total(1,:,k) = [0 0 0 0 1];
k = k + 1;
end

% Fill training dataset no.2 (without separation on subtasks)
% UNIT 1
k = 1;
for i = 1:N
DataSet_Images(1, :, k) = trianglesI(:, i);
DataSet_Labels(1, :, k) = [1 0 0]; % 3 neurons in the output layer
DataSet_Labels_1(1, :, k) = 1;
k = k + 1;
end
for i = 1:N
DataSet_Images(1, :, k) = rectanglesI(:, i);
DataSet_Labels(1, :, k) = [0 1 0];
DataSet_Labels_1(1, :, k) = 0;
k = k + 1;
end
for i = 1:N
DataSet_Images(1, :, k) = pentagonesI(:, i);
DataSet_Labels(1, :, k) = [0 0 1];
DataSet_Labels_1(1, :, k) = -1;
k = k + 1;
end

```

```
...
```

train_ffnn.m

```
% Train function
trainFcn = 'trainscg';
% trainFcn = 'traingd';
% trainFcn = 'trainb';

% Construct Global NN
net = feedforwardnet(128, trainFcn);
net.layers{1}.transferFcn = 'tansig'; % activation func

% Construct Partial NNs
net1 = feedforwardnet(100, trainFcn);
net2 = feedforwardnet(100, trainFcn);
net3 = feedforwardnet(100, trainFcn);
...

% Go & Train
% Global
images = squeeze(DataSet_Images(1,:,:));
labels = squeeze(DataSet_Labels(1,:,:));
[net, tr] = train(net1, images, labels);

% Partial
% unit 1
images = squeeze(DataSet_Images(1,:,:));
labels = squeeze(DataSet_Labels(1,:,:));
[net1, tr1] = train(net1, images, labels);
% unit 2
images = squeeze(DataSet_Images(2,:,:));
labels = squeeze(DataSet_Labels(2,:,:));
[net2, tr2] = train(net2, images, labels);

...
```

# Содержание