# 《深度学习引论》课程报告

**课题名称： 使用深度学习解决fashion-mnist分类问题**

课题负责人名（学号）： 王君有

同组成员名单（角色）：

指导教师： 章毅

**评阅成绩：**

评阅意见：

提交报告时间：2019年11月 21 日

**使用深度学习解决Fashion-Mnist分类问题**

计算机金融 专业

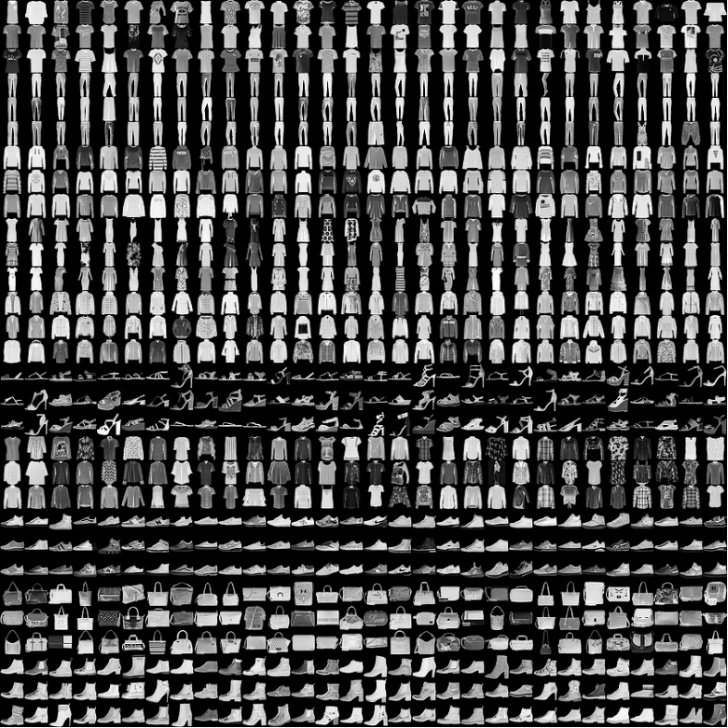
**学生** 王君有 **指导老师** 章毅

**[摘要]** 近年来借着大数据的应用深度学习蓬勃发展，而图像处理是深度学习应用最为广泛的领域之一。在深度学习的图像分类问题中，手写体数字是别的数据集mnist是使用最为广泛的数据集之一，但近些年来已经不足测试神经网络的性能。fashion-mnist数据集是与mnist相同格式的用以替代手写体数字识别的数据集，本实验尝试利用神经网络解决fashion-mnist数据集的图像分类问题。

**关键词**：神经网络 mnist 图像分类

一、问题描述

Fashion-MNIST[1]是一个替代MNIST手写数字集的图像数据集。 它是由Zalando（一家德国的时尚科技公司）旗下的研究部门提供。其涵盖了来自10种类别的共7万个不同商品的正面图片。Fashion-MNIST的大小、格式和训练集/测试集划分与原始的MNIST完全一致。60000/10000的训练测试数据划分，28x28的灰度图片。

图1：fashion-mnist数据集预览大致情况

现使用该数据集，使用matlab编程语言实现基于深度学习的图像分类问题。

二、设计解决方案

本次实验使用MATLAB语言，设计了全连接的用于图像分类的神经网络。并设计五层与七层的神经网络做对比。首先编写程序读入fashion-mnist数据集中的数据，然后设计相应的网络结构和参数，之后将数据输入网络中进行训练然后用测试集进行测试。

三、数据集准备

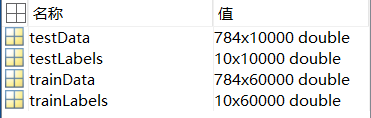
本实验使用的是来自github的fashion-mnist数据集。其中训练集和测试集分别为为60000张28×28的黑白图像，可使用斯坦福大学书写的matlab读取函数读入图像数据，并将标签数据转化为1×10的列向量标签。保存为.mat文件方便读取。

图2：fashion-mnist.mat文件内容

四、网络结构

本次实验的神经网络为全连接神经网络，分别采用五层和七层神经网络进行对比实验，五层神经网络结构为：

表1：实验1网络结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer | Input | Hidden | Hidden | Hidden | Output |
| External input | 784 | 0 | 0 | 0 | 0 |
| Internal input | 56 | 270 | 90 | 30 | 10 |
| Activation |  | Sigm | Sigm | Sigm | Sigm |

而七层神经网络结构为：

表2：实验2网络结构

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Layer | Input | Hidden | Hidden | Hidden | Hidden | Hidden | Output |
| External input | 784 | 0 | 0 | 0 | 0 | 0 | 0 |
| Internal input | 56 | 401 | 191 | 92 | 44 | 21 | 10 |
| Activation |  | Sigm | Sigm | Sigm | Sigm | Sigm | Sigm |

五、训练网络

5.1 实验1（5层全连接神经网络）

设置参数为alpha = 0.05；max\_iter = 600; mini\_batch = 40; 使用MATLAB进行训练。

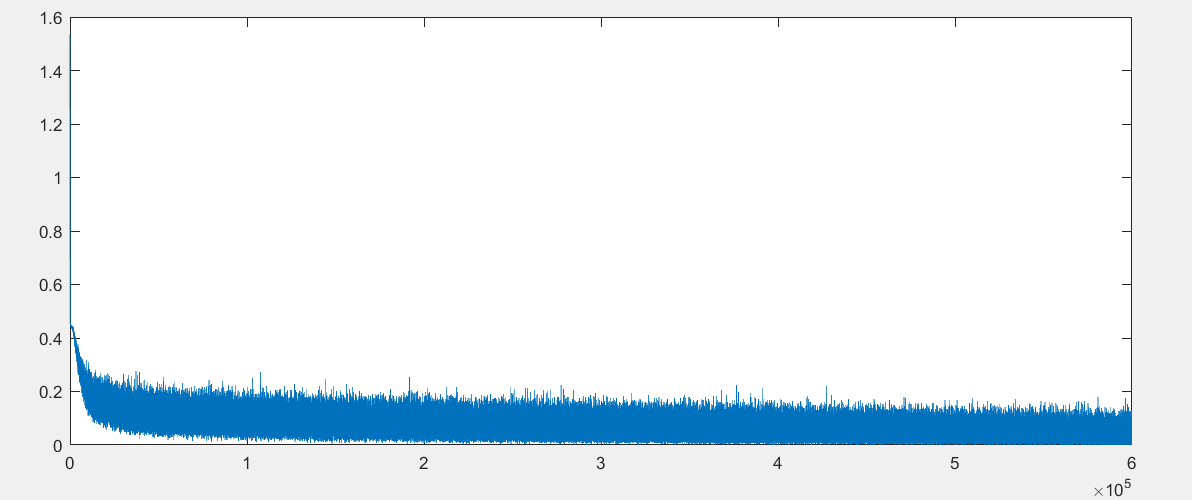


图3：实验1误差下降曲线

5.2 实验2（7层全连接神经网络）

设置参数为alpha = 0.05；max\_iter = 600; mini\_batch = 40; 使用MATLAB进行训练。

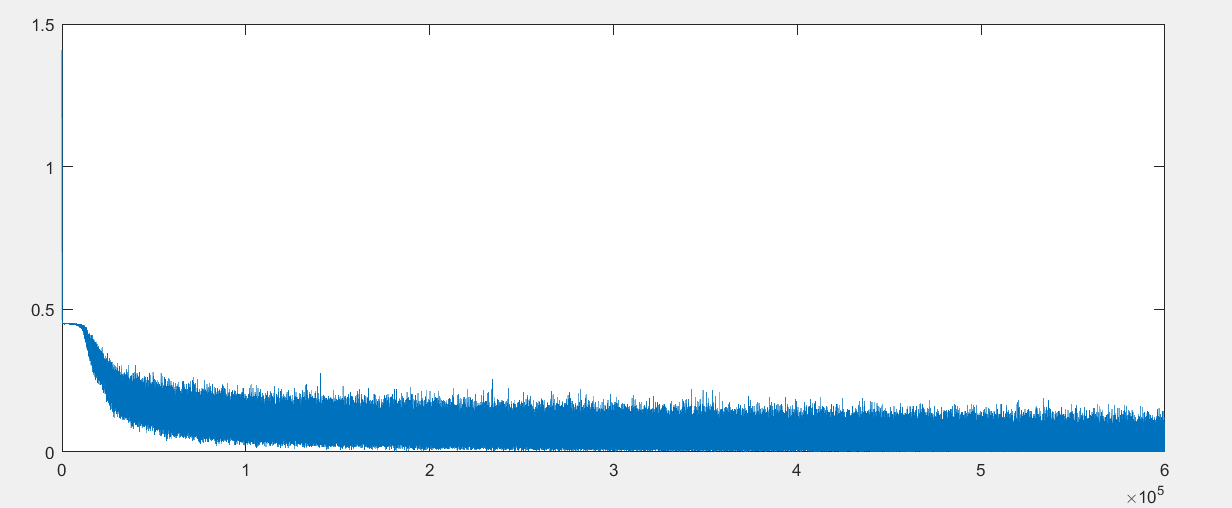


图4：实验2误差下降曲线

六、实验结果

6.1 实验1（5层全连接神经网络）结果为：

Accuracy on training dataset is 96.681667%

Accuracy on testing dataset is 88.750000%

6.2 实验2（7层全连接神经网络）结果为：

Accuracy on training dataset is 97.621667%

Accuracy on testing dataset is 88.460000%

七、结论

BP神经网络是应用最为广泛的网络结构，本次实验采用的全连接神经网络能够较好地识别出fashion-mnist中包含的服饰种类，并在测试集上达到88%左右的正确率，但可以发现全连接神经网络在面对输入数据大小过大和神经网络层数过多的情况时，参数量也会上升导致神经网络训练的速度较慢。

八、实现代码

loadMNISTImages.m

|  |
| --- |
| function images = loadMNISTImages(filename)  %loadMNISTImages returns a 28x28x[number of MNIST images] matrix containing  %the raw MNIST images  fp = fopen(filename, 'rb');  assert(fp ~= -1, ['Could not open ', filename, '']);  magic = fread(fp, 1, 'int32', 0, 'ieee-be');  assert(magic == 2051, ['Bad magic number in ', filename, '']);  numImages = fread(fp, 1, 'int32', 0, 'ieee-be');  numRows = fread(fp, 1, 'int32', 0, 'ieee-be');  numCols = fread(fp, 1, 'int32', 0, 'ieee-be');  images = fread(fp, inf, 'unsigned char');  images = reshape(images, numCols, numRows, numImages);  images = permute(images,[2 1 3]);  fclose(fp);  % Reshape to #pixels x #examples  images = reshape(images, size(images, 1) \* size(images, 2), size(images, 3));  % Convert to double and rescale to [0,1]  images = double(images) / 255;  end |

loadMNISTLabels.m

|  |
| --- |
| function labels = loadMNISTLabels(filename)  %loadMNISTLabels returns a [number of MNIST images]x1 matrix containing  %the labels for the MNIST images  fp = fopen(filename, 'rb');  assert(fp ~= -1, ['Could not open ', filename, '']);  magic = fread(fp, 1, 'int32', 0, 'ieee-be');  assert(magic == 2049, ['Bad magic number in ', filename, '']);  numLabels = fread(fp, 1, 'int32', 0, 'ieee-be');  labels = fread(fp, inf, 'unsigned char');  assert(size(labels,1) == numLabels, 'Mismatch in label count');  fclose(fp);  end |

mnistReader.m(数据集读取脚本)

|  |
| --- |
| % === Introduction ===  % The MNIST dataset is a dataset of handwritten digits, comprising 60 000  % training examples and 10 000 test examples. The dataset can be downloaded  % from http://yann.lecun.com/exdb/mnist/.  % === Usage ===  clear;  close all;  %% read and resize the dataset  % On some platforms, the files might be saved as  % train-images.idx3-ubyte / train-labels.idx1-ubyte  trainData = loadMNISTImages('dataset/train-images-idx3-ubyte');  trainLabels = loadMNISTLabels('dataset/train-labels-idx1-ubyte');  testData = loadMNISTImages('dataset/t10k-images-idx3-ubyte');  testLabels = loadMNISTLabels('dataset/t10k-labels-idx1-ubyte');  train\_len = size(trainLabels, 1);  train\_temp = zeros(10, train\_len);  for i = 1:train\_len  train\_temp(trainLabels(i)+1, i) = 1;  end  trainLabels = train\_temp;  test\_len = size(testLabels, 1);  test\_temp = zeros(10, test\_len);  for i = 1:test\_len  test\_temp(testLabels(i)+1, i) = 1;  end  testLabels = test\_temp;  save fash\_mnist.mat trainData trainLabels testData testLabels  clear train\_temp test\_temp  %% make smaller dataset  ind = randperm(train\_len);  trainData = trainData(:, ind(1:20000));  trainLabels = trainLabels(:, ind(1:20000));  ind = randperm(test\_len);  testData = testData(:, ind(1:5000));  testLabels = testLabels(:, ind(1:5000));  save fash\_mnist\_25k.mat trainData trainLabels testData testLabels |

fc.m

|  |
| --- |
| function [a\_next, z\_next] = fc(w, a, x, f)  z\_next = w \* [x; a];  a\_next = f(z\_next);  end |

bc.m

|  |
| --- |
| function delta = bc(w, z, delta\_next, df)  delta = df(z) .\* (w(:, end-size(z,1)+1:end)' \* delta\_next);  end |

accuracy.m

|  |
| --- |
| function [ acc ] = accuracy( a, y )  mini\_batch = size(a, 2);  [~,idx\_a] = max(a);  [~,idx\_y] = max(y);  acc = sum(idx\_a==idx\_y) / mini\_batch;  end |

lab\_final.m(主程序脚本)

|  |
| --- |
| % clear workspace and close plot windows  clear;  close all;  %% prepare the data set  load fash\_mnist.mat  input\_size=28\*28;  % prepare training data  train\_size=size(trainLabels,2);  X\_train{1}=reshape(trainData,[],train\_size);  X\_train{2}=zeros(0,train\_size);  X\_train{3}=zeros(0,train\_size);  X\_train{4}=zeros(0,train\_size);  X\_train{5}=zeros(0,train\_size);  X\_train{6}=zeros(0,train\_size);  X\_train{7}=zeros(0,train\_size);  % prepare testing data  test\_size = size(testLabels, 2);  X\_test{1} = reshape(testData, [], test\_size);  X\_test{2} = zeros(0, test\_size);  X\_test{3} = zeros(0, test\_size);  X\_test{4} = zeros(0, test\_size);  X\_test{5} = zeros(0, test\_size);  X\_test{6} = zeros(0, test\_size);  X\_test{7} = zeros(0, test\_size);  %% prepare standard speech audio  % Y\_train = trainLabels;  % Y\_test = testLabels;  %% choose parameters  alpha=0.05;  max\_iter=600;  mini\_batch=40;  % layer\_size=[input\_size 56  % 0 270  % 0 90  % 0 30  % 0 10];  layer\_size=[input\_size 56  0 401  0 191  0 92  0 44  0 21  0 10];  L=size(layer\_size,1);    %% define network architecture  sigm=@(s) 1./(1+exp(-s));  dsigm=@(s) sigm(s).\*(1-sigm(s));  relu = @(s) max(0, s);  drelu = @(s) s.\*(s>0);  lin = @(s) s;  dlin=@(s) 1;  fs={[],sigm,sigm,sigm,sigm,sigm,sigm,sigm};  dfs={[],dsigm,dsigm,dsigm,dsigm,dsigm,dsigm};  % fs = {[], relu, relu, relu, relu, relu, relu, relu};  % dfs = {[], drelu, drelu, drelu, drelu, drelu, drelu, drelu};  %% initialize weights  w=cell(L-1,1);  for l=1:L-1  w{l} = (randn(layer\_size(l+1, 2), sum(layer\_size(l, :)))) \* ...  sqrt(6/( layer\_size(l+1, 2) + sum(layer\_size(l,:)) ));  w\_mean{l} = [];  w\_std{l} = [];  end  %% train  J=[];  Acc = []; % array to store accuracy of each mini batch  x=cell(L,1);  a=cell(L,1);  z=cell(L,1);  delta=cell(L,1);  %% loop  for iter=1:max\_iter  ind=randperm(train\_size);  for k=1:ceil(train\_size/mini\_batch)  % preapre the internal input  a{1}=zeros(layer\_size(1,2),mini\_batch);  % prepare external input  for l=1:L  x{l}=X\_train{l}(:,ind((k-1)\*mini\_batch+1:min(k\*mini\_batch,train\_size)));  end  % prepare labels  % prepare targets  y = trainLabels(:,ind((k-1)\*mini\_batch+1:min(k\*mini\_batch,train\_size)));    % batch forward computation  for l=1:L-1  [a{1+l},z{1+l}]=fc(w{l},a{l},x{l},fs{l+1});  end  %cost function and error  J=[J 1/2/mini\_batch\*sum((a{L}(:)-y(:)).^2)];  Acc = [Acc accuracy(a{L}, y)];  mini\_batch = size(a{L}, 2);  delta{L}=(a{L}-y).\*dfs{L}(z{L});  % run bc    for l=L-1:-1:2  delta{l}=bc(w{l},z{l},delta{l+1},dfs{l});  end  % update weight  for l=1:L-1  dw=delta{l+1}\*[x{l};a{l}]' /mini\_batch;  w{l}=w{l}-alpha\*dw;  w\_mean{l} = [w\_mean{l} mean(w{l}(:))];  w\_std{l} = [w\_std{l} std(w{l}(:))];  end  end  % end loop    if mod(iter,1)==0  fprintf('%i/%i epochs: J=%.4f Acc=%.4f\n', iter, max\_iter, J(end), Acc(end));  end  end  %% save model  save model\_7l\_easy.mat w layer\_size J w\_mean w\_std  %% display/listen to some results pairs  % test on training set  a{1} = zeros(layer\_size(1,2), train\_size);  x{1} = X\_train{1};  for l=1:L-1  [a{1+l},z{1+l}]=fc(w{l},a{l},x{l},fs{l+1});  end  train\_acc = accuracy(a{L}, trainLabels);  fprintf('Accuracy on training dataset is %f%%\n', train\_acc\*100);  %%  % test on testing set  a{1}=zeros(layer\_size(1,2),test\_size);  x{1} = X\_test{1};  for l=1:L-1  [a{1+l},z{1+l}]=fc(w{l},a{l},x{l},fs{l+1});  end  test\_acc = accuracy(a{L}, testLabels);  fprintf('Accuracy on testing dataset is %f%%\n', test\_acc\*100); |

# 参考文献

[1] Zalando Research.fashion-mnist.https://github.com/zalandoresearch/fashion-mnist/,2017