ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий

Высшая школа программной инженерии

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Проектирование интеллектуальных систем управления»

Студент А. М. Потапова
гр. 3530202/90202

Руководитель Ю. Н. Кожубаев

Санкт-Петербург
2022 г

Ход работы

В качестве примера мною было выбрано создание простой сети глубокого обучения для классификации.

В этом примере показано, как создать и обучить простую свртоточную нейронную сеть для классификации глубокого обучения. Сверточные нейронные сети являются важными инструментами для глубокого обучения и особенно подходят для распознавания изображений.

- **Загрузка и изучение данных изображения.**

Шаг 1

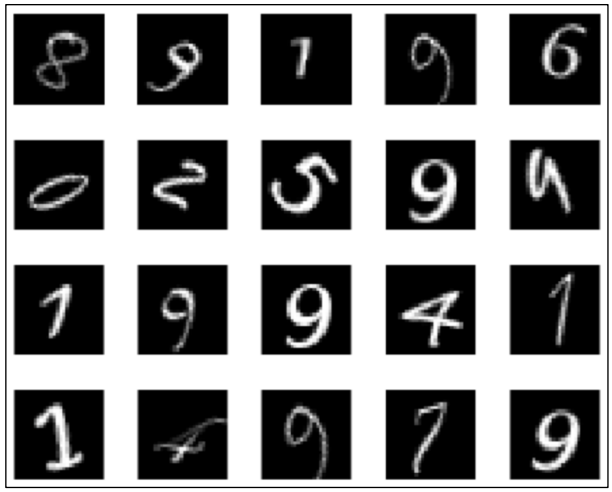
Загрузим данные образца цифры как хранилище данных изображения. ImageDatastore автоматически помечает изображения на основе имен папок и сохраняет данные как объект ImageDatastore. Хранилище данных изображений позволяет хранить большие данные изображений, в том числе данные, которые не помещаются в памяти, и эффективно считывать пакеты изображений во время обучения сверточной нейронной сети.

```
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet','nndemos', ...
    'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
    'IncludeSubfolders',true,'LabelSource','foldernames');
```

Отобразим некоторые изображения в хранилище данных.

```
figure;
perm = randperm(10000,20);
for i = 1:20
    subplot(4,5,i);
    imshow(imds.Files{perm(i)});
end
```

Результат



Шаг 2

Подсчитаем количество изображений в каждой категории. labelCount — это таблица, содержащая метки и количество изображений с каждой меткой. Хранилище данных содержит 1000 изображений для каждой из цифр 0–9, всего 10000 изображений. Мы можем указать количество классов в последнем полносвязном слое нашей сети в качестве аргумента OutputSize.

```
labelCount = countEachLabel(imds)
```

Результат

labelCount = 10×2 table

	Label	Count
1	0	1000
2	1	1000
3	2	1000
4	3	1000
5	4	1000
6	5	1000
7	6	1000
8	7	1000
9	8	1000

Шаг 3

Укажем размер изображений во входном слое сети и проверим размер первого изображения в digitData.

```
img = readimage(imds,1);
size(img)
```

Получили, что каждое изображение имеет размер 28 на 28 на 1 пиксель.

- **Определение обучающих и проверочных наборов**

Разделим данные на обучающие и проверочные наборы данных, чтобы каждая категория в обучающем наборе содержала 750 изображений, а проверочный набор содержал оставшиеся изображения с каждой метки. splitEachLabel разделяет хранилище данных digitData на два новых хранилища данных, trainDigitData и valDigitData.

```
numTrainFiles = 750;
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
```

- **Определение сетевой архитектуры**

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

Определение параметров обучения

После определения структуры сети укажем параметры обучения. Обучим сеть, используя стохастический градиентный спуск с импульсом (SGDM) с начальной скоростью обучения 0,01. Установим максимальное количество эпох равным 4. Эпоха — это полный цикл обучения на всем наборе обучающих данных. Проконтролируем точность сети во время обучения, указав данные проверки и частоту проверки. Перетасуем данные каждую эпоху. Включим график прогресса обучения и отключим вывод командного окна.

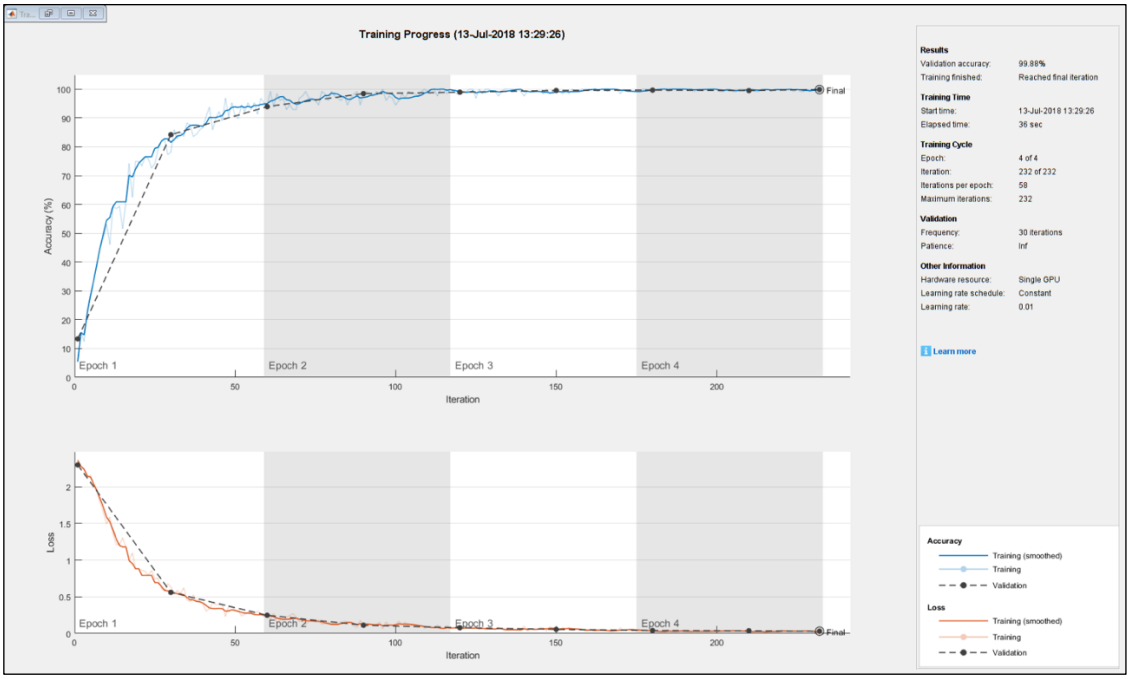
```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

Обучение сети используя обучающие данные

Обучим сеть, используя архитектуру, определенную слоями, обучающими данными и вариантами обучения. По умолчанию trainNetwork использует графический процессор, если он доступен, в противном случае он использует ЦП. Мы также можем указать среду выполнения, используя аргумент пары "имя-значение" ExecutionEnvironment для trainingOptions. На графике хода обучения показаны потери и точность мини-пакетов, а также потери и точность проверки. Потеря — это кросс-энтропийная потеря. Точность — это процент изображений, которые сеть правильно классифицирует.

```
net = trainNetwork(imdsTrain,layers,options);
```

Результат



Классифицирование проверочных изображений и вычисление точности

Предскажем метки данных проверки, используя обученную сеть, и рассчитаем окончательную точность проверки. Точность — это доля меток, которые сеть предсказывает правильно.

```
YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;

accuracy = sum(YPred == YValidation)/numel(YValidation)
```

Результат

```
accuracy = 0.9988
```

Вывод

В ходе данной работы мне удалось создать простую сети глубокого обучения для классификации. В моем случае более 99% предсказанных меток совпадают с истинными метками проверочного набора.