МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ **Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

Направления подготовки: «Прикладная математика и информатика», «Фундаментальная информатика и информационные технологии»

Магистерские программы: «Системное программирование», «Компьютерная графика и моделирование живых и технических систем»

**Образовательный курс «Методы глубокого обучения для решения задач компьютерного зрения»**

**Лабораторная работa №3:**

**«Разработка сверточной нейронной сети с помощью библиотеки MXNet»**

Выполнил:

Чугунов Е. А.

Преподаватель:

Кустикова В. Д.

Нижний Новгород

2018

Оглавление

[Цель работы 3](#_Toc506129771)

[Задачи 3](#_Toc506129772)

[Решаемая задача 4](#_Toc506129773)

[Подготовка тренировочных данных 5](#_Toc506129774)

[Архитектуры полностью связанных нейронных сетей 6](#_Toc506129775)

[Результаты работы 7](#_Toc506129776)

[Расчет ошибки 7](#_Toc506129777)

[Результаты тестирования 7](#_Toc506129778)

[Архитектура #1 7](#_Toc506129779)

[Архитектура #2 7](#_Toc506129780)

[Архитектура #3 7](#_Toc506129781)

[Заключение 8](#_Toc506129782)

[Приложение 9](#_Toc506129783)

[Код программы 9](#_Toc506129784)

# Цель работы

Получить базовые навыки работы с библиотекой MXNet и разработать сверточную нейронную сеть.

## Задачи

* Установить библиотеку MXNet и проверить корректность установки, используя её для решения задачи классификации рукописных цифр набора данных MNIST.
* Выбрать практическую задачу распознавания образов для выполнения практических работ
* Разработать программы/скрипты для подготовки тренировочных и тестовых данных в формате, который обрабатывается библиотекой.
* Разработать несколько архитектур полностью связанных нейронных сетей в формате, который принимается библиотекой.
* Обучение разработанных глубоких моделей.
* Тестирование обученных глубоких моделей.

# Решаемая задача

В качестве данных для обучения и тестирования сети используется набор изображений ASL (american sign language) Finger Spelling Dataset. Набор содержит более 60000 цветных изображений различного размера в формате .png. На изображениях представлены жесты из американского языка жестов. Набор содержит изображения 24 различных жестов. Отсутствуют жесты для букв j и z, так как они обозначаются активными жестами и распознать их можно только при наличии серии кадров. Набор изображений поделен на 5 категорий A, B, C, D, E – они отличатся фоном и человеком, делающим жесты. Кроме для каждого изображения из набора данных имеется карта глубин, необходимая для исключения заднего плана и увеличения точности распознавания. При решении данной задачи карта глубин использоваться не будет – это сильно повлияет на вероятность распознавания.

Для упрощения задачи была исключена категория D, включение которой сильно влияет на вероятность распознавания. Также были исключены буквы m, n, s, t, так как они очень похожи внешне и серьезно повлияют на точность. К тому же уменьшение числа изображений сильно увеличит скорость обучения сети.

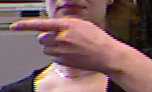
Итого, получается набор из 43839 изображений, которые распределены по 20 классам.

Примеры изображений:

Буква “A”:



Буква “G”:



Буква “O”:



# Подготовка тренировочных данных

Для обучения с использованием библиотеки MXNet изображения должны храниться в формате .rec. Для его генерации нужно создать файл листа .lst следующего формата:

<номер изображения> <класс изображения> <путь к изображению>

Была написана программа на языке C++, создающая два таких файла traininglist.lst и testinglist.lst.

В каждой папке из данного набора данных содержится примерно 500-700 изображений. Для обучения используются первые 400 изображений из каждой папки, все остальные изображения используются для тестирования. Таким образом, число обучающих изображений – 32000 (при этом для каждого класса имеется равное число изображений), число изображений для тестирования – 11839.

Создание файлов в формате .rec происходит с помощью скрипта im2rec.py с использованием следующих команд для командной строки:

python im2rec.py traininglist.lst F:\Download\dataset --center-crop True --resize 64 --exts .png --shuffle True

python im2rec.py testinglist.lst F:\Download\dataset --center-crop True --resize 64 --exts .png --shuffle True

Такие команды производят следующие манипуляции с изображениями:

* Обрезают изображения, чтобы стороны были равны.
* Скалируют изображения так, чтобы меньшая сторона была равна 64.
* Меняют порядок изображений на случайный.

Таким образом, все изображения становятся квадратными, с размерами 64х64.

Для обучения и тестирования используются получившиеся файлы traininglist.rec и testinglist.rec.

# Архитектуры нейронных сетей

Было разработано 2 различных конфигурации нейронной сети:

1. Один сверточный слой, один полностью связанный слой.
2. Два сверточных слоя, один полностью связанный слой.

Во всех случаях, функции активации на каждом слое – tanh: . Такой выбор функции активации объясняется высокими показателями точности при её использовании

Функция активации на выходном слое – Softmax: .

Увеличение числа полностью связанных слоев снижает точность распознавания очень сильно. Также, два сверточных слоя не дают желаемого результата – точность распознавания не поднимается выше 25%. Поэтому окончательное тестирование проводилось на первой архитектуре.

На Рис. 1 показана тестируемая архитектура сверточной нейронной сети.

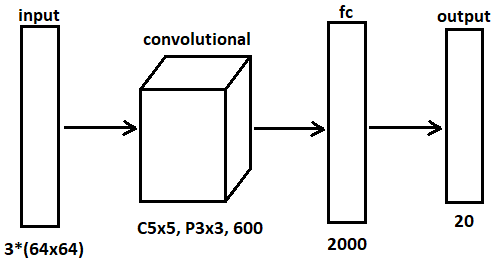


Рис. 1. Тестируемая архитектура нейронной сети

# Результаты работы

## Расчет ошибки

Формула для расчета точности классификатора:

## Результаты тестирования

### Архитектура сети

* Batch size = 128
* Число сверточных слоев = 1
* Ядро сверточного слоя: 5x5, stride: (4x4), число фильтров: 600.
* Ядро pooling: 3x3, stride: (2x2).
* Функция активации на сверточном слое – tanh.
* Число полностью связанных слоев = 1
* Функция активации на полностью связанном слое – tanh.
* Число нейронов полностью связанного слоя = 2000.
* Скорость обучения = 0.008.
* Число эпох – 30.

Точность на обучающей выборке = 0.9915, на тестовой выборке = 0.7028.

# Заключение

Результаты тестирования показывают, что сеть с одним сверточным и одним полностью связанным слоем на данной выборке показывает себя лучше всего. При таком количестве слоев, увеличение числа эпох увеличивает точность классификации в большинстве случаев. Но при добавлении второго сверточного или полностью связанного слоя точность классификации начинает падать от эпохи к эпохе. Скорее всего это вызвано переобучением сети.

Сверточнаая нейронная сеть оказалась лучше обычной полностью связанной сети, но её результат еще далек от идеала: шанс ошибиться при распознавании – 30%. С другой стороны, распознавание жестов – непростая задача, так как многие жесты визуально очень похожи. При этом, одни и те же жесты могут сильно отличаться друг от друга в зависимости от поворота руки.

# Приложение

## Код программы

import mxnet as mx

import logging

import OpenSSL.SSL

logging.getLogger().setLevel(logging.DEBUG)

train\_iter = mx.io.ImageRecordIter(

path\_imglist="F:/Download/dataset/traininglist.lst",

path\_imgrec="F:/Download/dataset/traininglist.rec",

data\_shape=(3, 64, 64),

batch\_size=128,

shuffle=True)

test\_iter = mx.io.ImageRecordIter(

path\_imglist="F:/Download/dataset/testinglist.lst",

path\_imgrec="F:/Download/dataset/testinglist.rec",

data\_shape=(3, 64, 64),

batch\_size=128,

shuffle=True)

input = mx.sym.var('data')

# first conv layer

conv1 = mx.sym.Convolution(data=input, kernel=(5,5), num\_filter=600, stride=(4,4))

tanh1 = mx.sym.Activation(data=conv1, act\_type="tanh")

pool1 = mx.sym.Pooling(data=tanh1, pool\_type="max", kernel=(3,3), stride=(2,2))

#conv2 = mx.sym.Convolution(data=pool1, kernel=(3,3), num\_filter=400, stride=(1,1))

#tanh2 = mx.sym.Activation(data=conv2, act\_type="tanh")

#pool2 = mx.sym.Pooling(data=tanh2, pool\_type="max", kernel=(3,3), stride=(2,2))

flat = mx.sym.flatten(data = pool1)

#one layer: learning\_rate = 0.008

fclast = mx.sym.FullyConnected(data = flat, num\_hidden = 2000)

actlast = mx.sym.Activation(data=fclast, act\_type='tanh')

last = mx.sym.FullyConnected(data=actlast, num\_hidden=20)

output = mx.sym.SoftmaxOutput(data = last, name = 'softmax')

model = mx.mod.Module(symbol = output, context = mx.gpu())

model.fit(train\_iter,

eval\_data = test\_iter,

optimizer = 'sgd',

optimizer\_params = {'learning\_rate':0.008},

eval\_metric = 'acc',

batch\_end\_callback = mx.callback.Speedometer(200, 200),

num\_epoch = 30)

acc = mx.metric.Accuracy()

model.score(test\_iter, acc)

print(acc)