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 №2:**

**«Разработка полностью связанной нейронной сети с помощью библиотеки MXNet»**

Выполнил:

Чугунов Е. А.

Преподаватель:

Кустикова В. Д.

Нижний Новгород

2017

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 полносвязную 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.

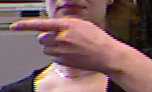
Итого, получается набор из 43839 изображений, которые распределены по 20 классам.

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 полностью связанных нейронных сетей

Было разработано 3 различных конфигурации нейронной сети:

1. Один скрытый слой, функция активации – Sigmoid:
2. Один скрытый слой, функция активации – Tanh:
3. Два скрытых слоя, функции активации – Tanh.

Во всех случаях, функция активации на выходном слое – Softmax: .

Увеличение числа слоев больше двух дает заметное ухудшение точности распознавания и добавляет необходимость корректировать большое количество переменных, чтобы добиться приемлемого результата.

На Рис. 1 показана архитектура нейронной сети с двумя скрытыми слоями, функция активации которых – Tanh.

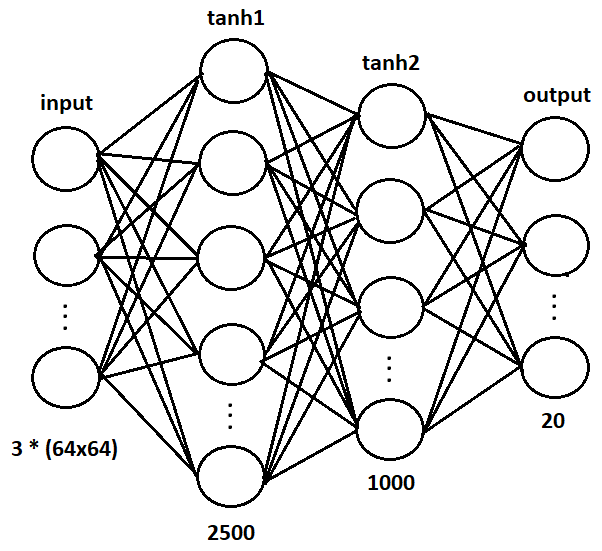


Рис. 1. Архитектура сети с двумя скрытыми слоями

# Результаты работы

## Расчет ошибки

Формула для расчета точности классификатора:

## Результаты тестирования

### Архитектура #1

* Batch size = 256
* Число скрытых слоев = 1
* Функция активации на скрытом слое – Sigmoid.
* Число нейронов скрытого слоя = 4000.
* Скорость обучения = 0.002.
* Число эпох – 30.

Точность на обучающей выборке = 0.912, на тестовой выборке = 0.546.

### Архитектура #2

* Batch size = 256
* Число скрытых слоев = 1
* Функция активации на скрытом слое – Tanh.
* Число нейронов скрытого слоя = 4000.
* Скорость обучения = 0.0005.
* Число эпох – 30.

Точность на обучающей выборке = 0.891, на тестовой выборке = 0.593.

### Архитектура #3

* Batch size = 256
* Число скрытых слоев = 2
* Функция активации на первом скрытом слое – Tanh.
* Функция активации на втором скрытом слое – Tanh.
* Число нейронов первого скрытого слоя = 2500.
* Число нейронов второго скрытого слоя = 1000.
* Скорость обучения = 0.0005.
* Число эпох – 30.

Точность на обучающей выборке = 0.751, на тестовой выборке = 0.476.

# Заключение

Результаты тестирования показывают, что однослойная полностью связанная нейронная сеть справляется с поставленной задачей лучше всего. В данном случае функция активации tanh оказалась гораздо эффективнее сигмоиды. Тестирование сети с двумя скрытыми слоями, где функция активации на обоих – sigmoid не дает вероятности распознавания выше 30%.

Однослойная сеть оказалась по многим параметрам (точность, время обучения) эффективнее двухслойной. Сеть с тремя слоями не показывает точность распознавания выше 20%.

Полностью связанная сеть не является особенно эффективной – максимальная точность распознавания составляет около 60%. 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=256,

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=256,

shuffle=True)

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

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

#two layers: learning\_rate = 0.001

fc1 = mx.sym.FullyConnected(data = flat, num\_hidden = 2500)

act1 = mx.sym.Activation(data=fc1, act\_type='tanh')

fclast = mx.sym.FullyConnected(data = act1, num\_hidden = 1000)

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

#one layer: learning\_rate = 0.002

#fclast = mx.sym.FullyConnected(data = flat, num\_hidden = 4000)

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

#one layer: learning\_rate = 0.0005

#fclast = mx.sym.FullyConnected(data = flat, num\_hidden = 4000)

#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':0.001},

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)