**Государственное общеобразовательное учреждение «Московская областная общеобразовательная школа-интернат естественно-математической направленности» имени П.Л. Капицы**

Исследовательская работа

# Умный замок

**Автор работы:**
Степанов Тимофей,
Кидысюк Александр,
Ковальков Никита,
Владимиров Алексей
9 класс

**Руководитель:**
Мерзляков А. В.

**Московская область, г. Долгопрудный**
**2020**

Область исследования: Нейронные Сети и Искусственный Интеллект в распознавании образов

**Актуальность:** низкая доступность подобных систем безопасности, основанных на распознавании образов, при помощи нейронных сетей. Современность и мало изученность метода.

**Цель:** создать систему защиты, с использованием технологий глубокого обучения и искусственного интеллекта.

**Задачи:**

1. Создать комплекс нейронных сетей для отличия живых лиц от их фотографий и распознавания лиц.
2. Создать графический интерфейс для умного замка
3. Создать dataset из лиц людей
4. Провести тесты системы на компьютере
5. Перенести систему на Raspberry Pi
6. Протестировать систему на Raspberry Pi
7. Составить отчет о работе программы и сделать выводы.

**Методы исследования:**

- Изучение научной литературы и статей в области искусственного интеллекта и глубокого обучения.
- Формирование базы данных, состоящих из фотографий одноклассников, знакомых и родственников в требуемом формате.
- Создание нейронных сетей.
- Тестирование и анализ результатов.

## Ход работы:

### Комплекс нейронных сетей.

Нашу систему можно разделить на 3 основные части: 1 нейронная сеть, определяющая живое лицо или нет. 2 нейронная сеть, отвечающая за распознавание лиц. И 3 часть, это графический интерфейс (далее UI). Нейронные сети будем далее обозначать NN.

### Первая NN.

Является конволюционной нейронной сетью (CNN). Распознает если лицо живое (настоящее, принадлежит человеку стоящему перед камерой) или поддельное (фотография или маска). На вход получает изображения лиц людей, представленные в виде тензоров 3-его ранга, где каждый слой соответствует цвету из палитры RGB, а значение ячеек соответствует яркости пикселя. Имеет VGG подобную архитектуру.

Состоит из данных слоев:

**Conv2D layer** – создает фильтр, которым обрабатываются изображения. Является ключевым элементом любой CNN. Фильтр представляет собой матрицу, значения которой определяются весами, полученными во время тренировки.

**Activation layer** – вызывает указанную в аргументе функцию. Мы используем только функции ReLU и Softmax.

Функция ReLU (rectified linear activation function) возвращает максимум от данного числа и 0.

Функция **Softmax** используется только в перовой нейронной сети. Делает из входного вектора вектор вероятностей отношения к категориям.

**MaxPooling2D layer** – берет максимальные значения из ячеек в квадрате n на n на m-том слое входного тензора и объединяет их в m-тый слой нового тензора

**BatchNormalization layer** – уменьшает значения ячеек входного тензора до значений между 0 и 1. Упрощает вычисления, не умаляя точности.

Таким образом, строение первой нейронной сети имеет вид:

**Conv2D => Activation (ReLU) => Batch Normalization => Conv2D => Activation (ReLU) => Batch Normalization => Conv2D => Activation (ReLU) => Batch Normalization => Conv2D => Activation (ReLU) => Batch Normalization => Activation (Softmax)**

**Вторая NN:**

Основная нейронная сеть – глубокая конволюционная (Inception CNN), построена на архитектуре nn4 small2 с небольшими изменениями. Использует те же слои что и первая NN и еще 2 других:

**ZeroPadding2D layer** – добавляет строки и столбцы наполненные нулями сверху, снизу и по бокам входного тензора

**Lambda layer** – используется для вызова функций. В нашем случаем вызывает Local Response Normalization. Эта функция уменьшает значения ячеек тензора для упрощения вычислений по формуле:

$$b^i_{x,y} = a^i_{x,y} / \left( k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

Где:

**i** указывает на **i**-тый слой входного тензора

**a(x, y)** и **b(x, y)** значение ячеек с координатами **x, y** до использования функции

**N** – кол-во слоев входного тензора

Константы ***(k, α, β, n)*** являются гиперпараметрами.

***k*** используется, чтобы избежать деление на 0.

***α*** - нормализующая константа.

***β*** - контрастирующая константа.

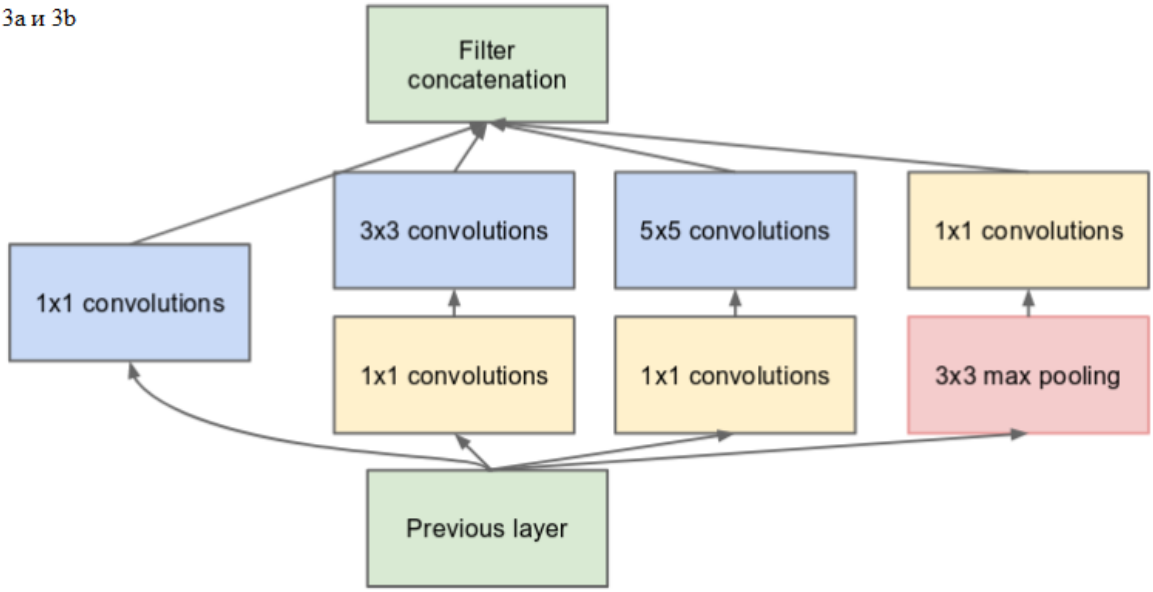
***n*** - определяет смещение по слоям тензора для значения стоящего на **i**-том слое.

Activation layer используется только для вызова ReLU функции.

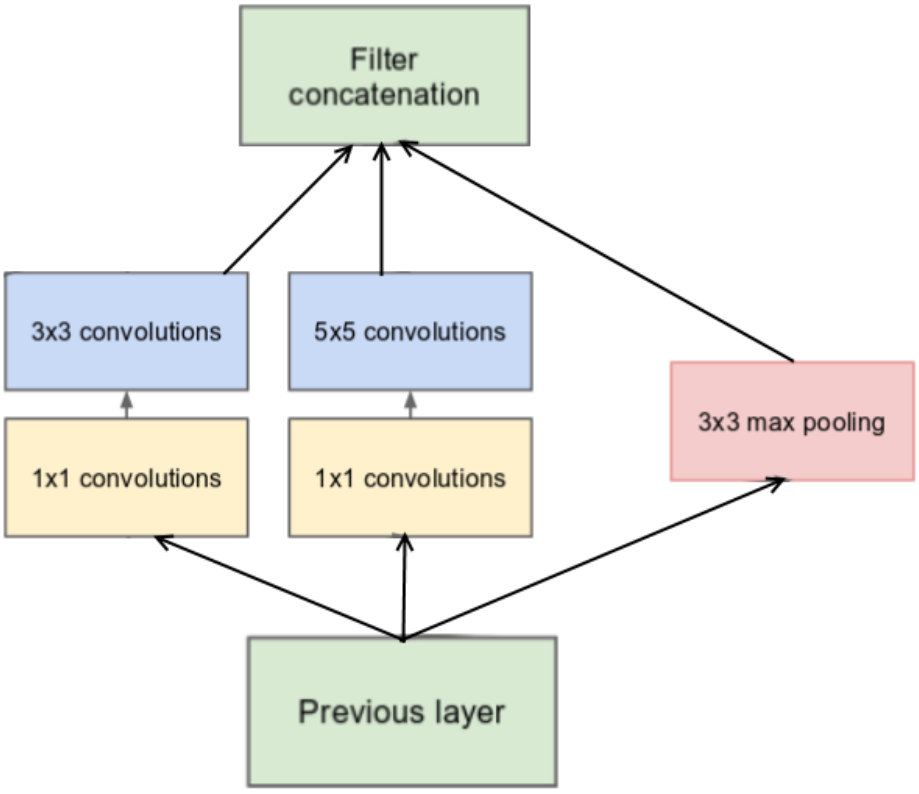
Особенностью строения глубоких CNN является соединение конволюционных слоев 1x1, 3x3, 5x5, MaxPooling и AveragePooling слоев в группы, называющиеся Inception блоки. Результаты работы каждого слоя конкатенируются и передаются следующему Inception блоку. В нашей сети 7 Inception блоков. Они обозначены и идут в порядке:

**3a => 3b => 3c => 4a => 4e => 5a => 5b**

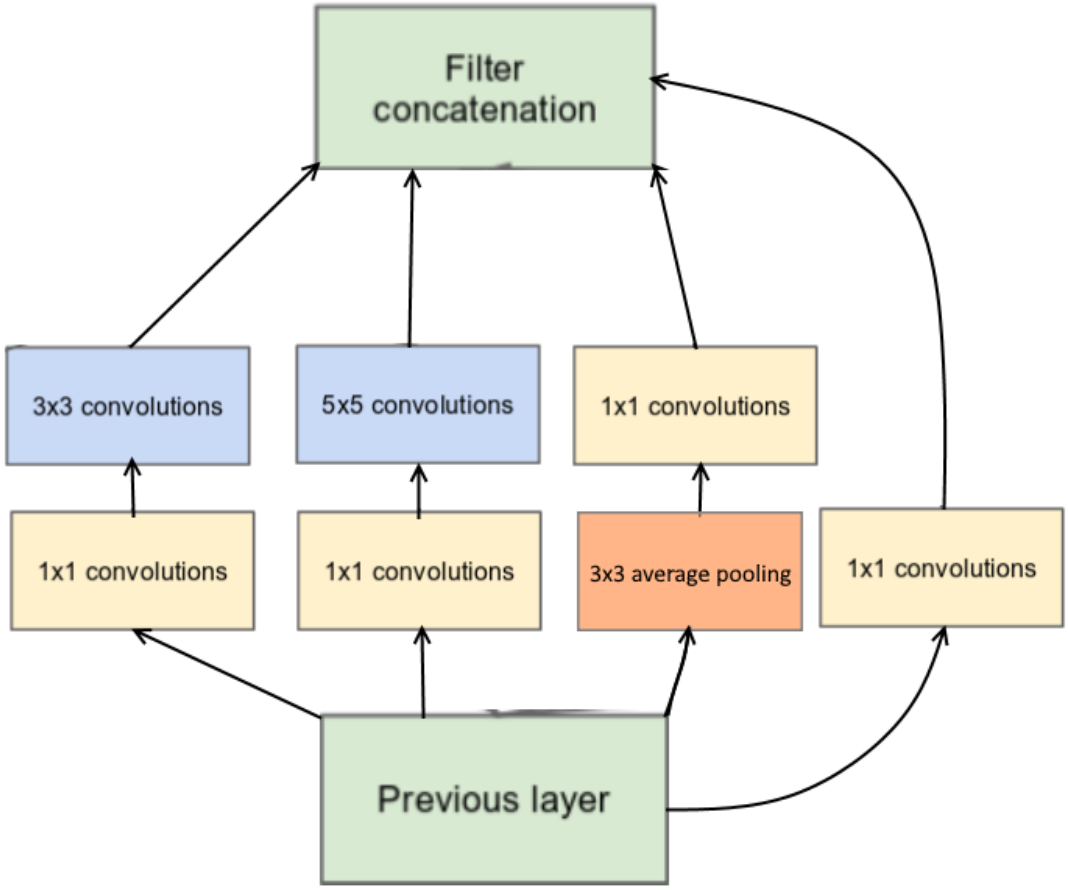
3a и 3b


Filter concatenation
1x1 convolutions
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer

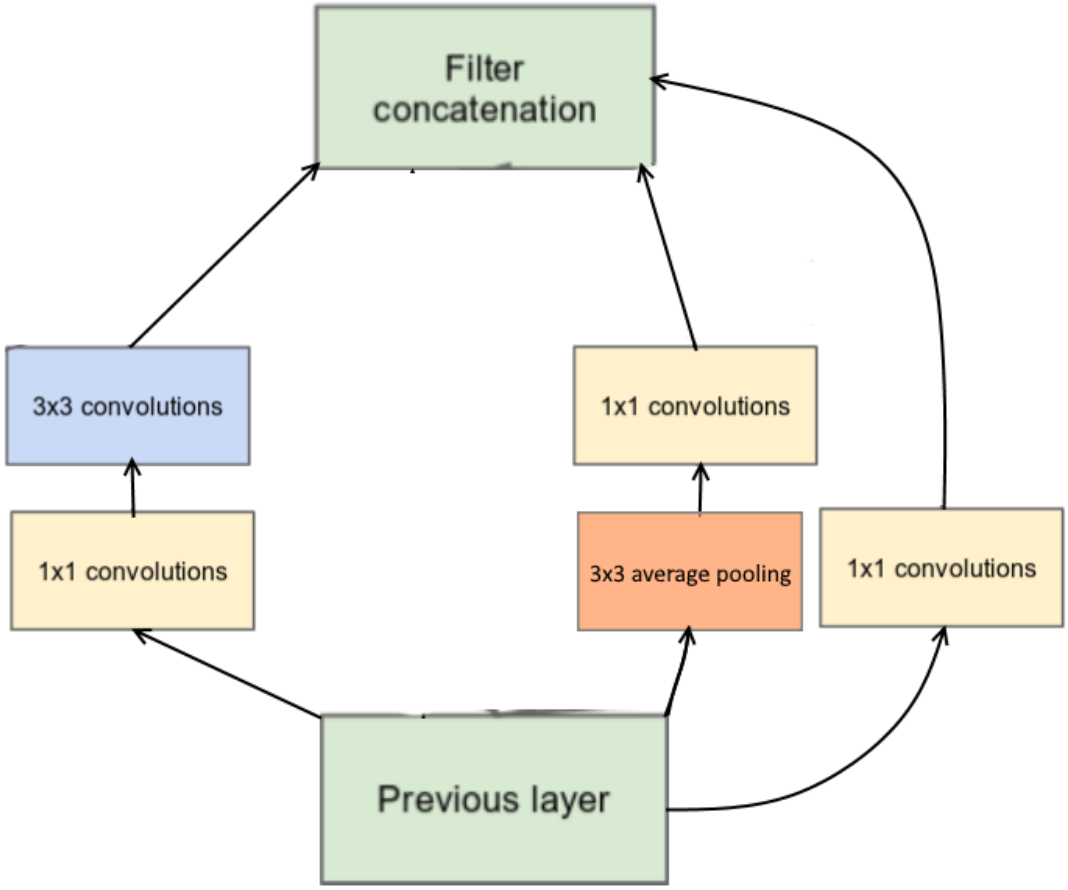
3c и 4e

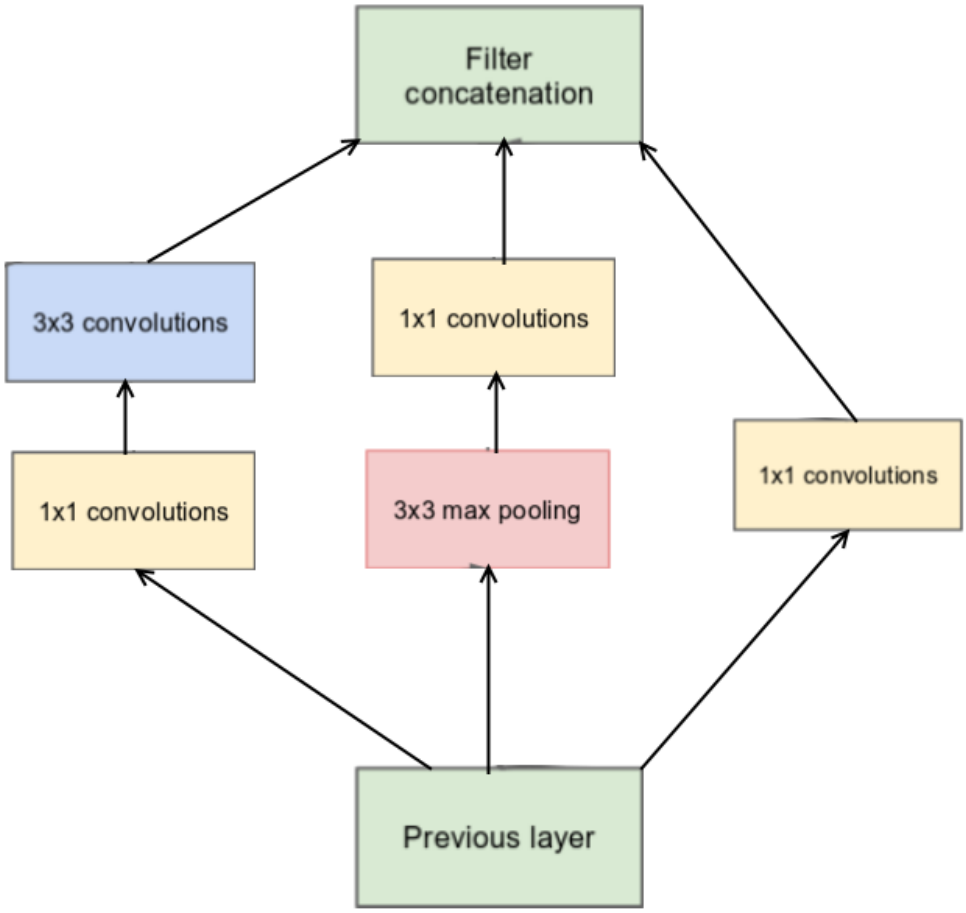

Filter concatenation
3x3 convolutions
5x5 convolutions
3x3 max pooling
1x1 convolutions
1x1 convolutions
Previous layer

4a


Filter concatenation
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 average pooling
1x1 convolutions
Previous layer

5a


Filter concatenation
3x3 convolutions
1x1 convolutions
1x1 convolutions
3x3 average pooling
1x1 convolutions
Previous layer

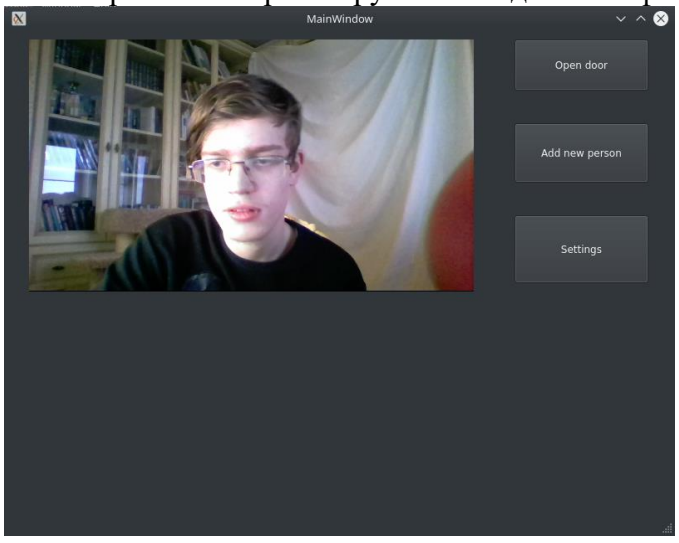



Обе нейронных сети написаны на Python.

**Графический интерфейс**

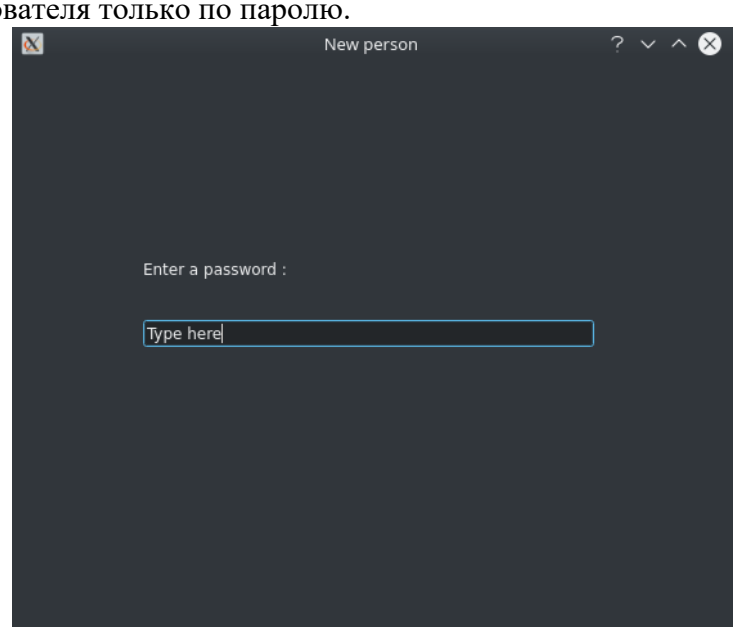
Система работает на микрокомпьютере Raspberry Pi 3B. Графический интерфейс создан в Qt Creator, на языке C++.

На экране присутствуют 3 кнопки и выводится изображение с камеры. Кнопка «Add new person» отвечает за добавление нового человека в систему. Открывает диалоговое окно. Сначала происходит авторизация пользователя паролем. Если авторизация прошла успешно, камера захватывает 5 фотографий пользователя подряд с пробелом между фотографиями в 200 мс. Далее вторая нейронная сеть делает из каждого снимка 128-мерный вектор и загружает в отдельный файл.

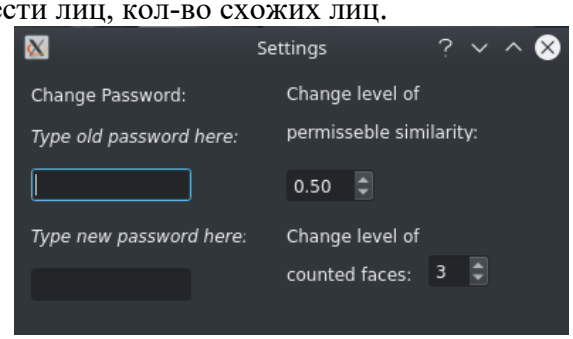



Кнопка “Open door” отвечает за открытие двери. При её нажатии камера делает снимок текущего кадра. Далее первая нейронная сеть оценивает, настоящее это лицо или

поддельное. Если лицо поддельное, то система просит повторить попытку входа. Если же лицо настоящее, то оно обрабатывается следующей нейронной сетью, которая сравнивает его с каждым авторизированным лицом. Уровень схожести лиц и кол-во схожих лиц для входа можно менять в настройках. После 3 неудачных попыток входа система впустит пользователя только по паролю.




Кнопка «Settings» открывает настройки. В них можно поменять пароль, уровень схожести лиц, кол-во схожих лиц.




Взаимодействие с системой реализуется через сенсорный экран.

## Тестирование

После того как наша модель умного замка была собрана, а программы написаны, мы приступили к тестированию нашего замка. Целью тестирования было определение правильности и стабильности работы системы, т.е. одновременная проверка и комплекса нейронных сетей и графического интерфейса.

## Итоги работы

По результатам тестирования наша нейронная сеть справилась со всеми тестами. Во время продолжительных тестов (~1 неделя) процент ошибочных определений не превысил 5% от общей выборки попыток войти в дверь. Все ошибки были связаны с недопуском известного человека. Отметим, что это происходило из-за плохого качества полученной фотографии, а не из-за программной ошибки. Проблем с графическим интерфейсом также не возникло.

## Литература и ссылки на источники


- [1]https://www.pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/ - Статья Адриана
- [2]https://keras.io/api/layers/ - Документация про слои нейронной сети
- [3]https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202 - Статья про глубокие нейронные сети

• [4]**https://towardsdatascience.com/difference-between-local-response-normalization-and-batch-normalization-272308c034ac** - Статья про LRN