ФИЛИАЛ МГУ имени М.В. ЛОМОНОСОВА в городе БАКУ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ

КУРСОВАЯ РАБОТА

Невербальное общение с компьютером

студента III курса группы № 218

Эфендиев Камал Вугар оглы

Научный Руководитель:
Доцент, кандидат физико-математических наук
Строгалов Александр Сергеевич

БАКУ - 2021

Содержание

1 Введение

Жестовый язык - самостоятельный язык,состоящий изкомбинации жестов, каждый из которых производится руками в сочетании с мимикой, формой или движением рта и губ, а также в сочетании с положением корпуса тела.
Использование этого языка является основным способом коммуникации людей с нарушением слуха и играет важную роль в их жизни, поскольку даёт возможность участия в жизни общества.

Различными положениями пальцев рук можно воспроизводить знаки, отдалённо напоминающие по очертанию буквы естественного языка.

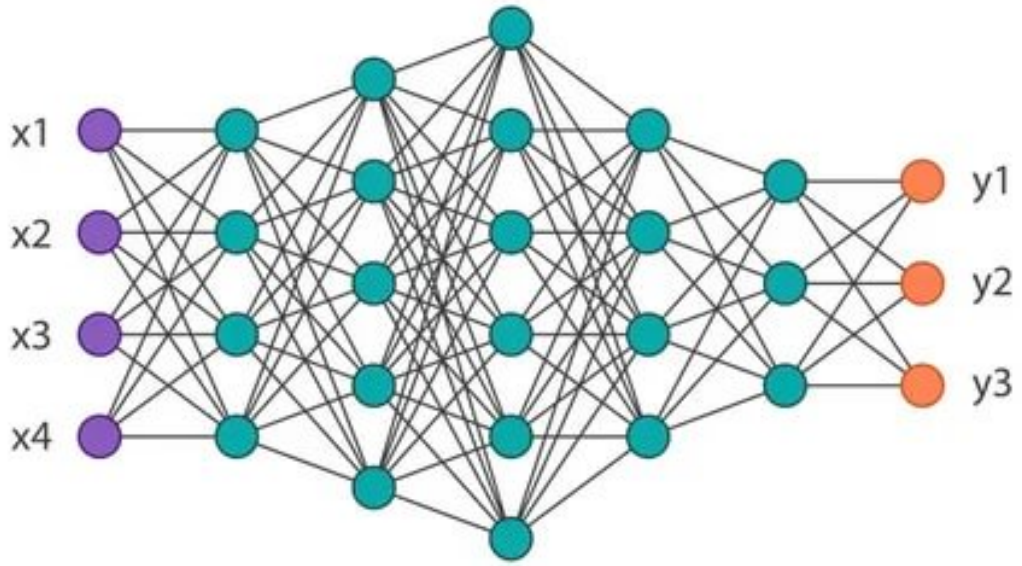
В настоящее время существуют различные технологии, выполняющие распознание жестового языка. Однако эта задача остаётся актуальной

Жестовые языки в первую очередь предназначены для общения глухих между собой, но используются не только ими. К примеру, родители ребенка с полной или частичной, но сильной потерей слуха,

вынуждены тоже учить жестовый язык, чтобы в свою очередь научить ему ребенка и общаться с ним.

Для обучения жестовому языку можно воспользоваться помощью учителя, но это может быть не всегда удобно, а кроме того, требует финансовых затрат. Также в интернете есть много видео, посвященных обучению тому или иному жестовому языку, но они не дают ученику обратной связи и не позволяют оценить корректность выполнения жестов.

Наибольший интерес представляют технологии искусственных нейронных сетей, поскольку имеют высокую популярность и широкое распространение в повседневной жизни.



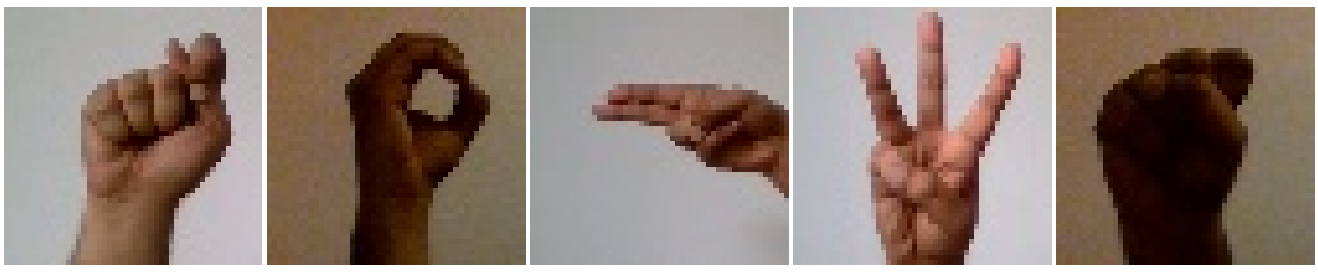
2 Метод

Модель является результатом изменения и дообучения такой State-of-Art модели как VGG19. Она была обучена на дополненном вручную наборе данных снимков рук которые изображают знаки языка жестов, находящемся в свободном доступе на Kaggle. Для реализации моделей использовалась библиотека нейронной сети Keras с TensorFlowbackend. Обучение проводилось на GPU Tesla V100 16Gb.

3 Обзор нейронной сети

3.1 Вход

На вход нейроонной сети идут изображения размера 50 на 50, в трех каналах, предварительно рассортированые по папкам. В нейронную сеть на вход подается тензор следующей размерности: (количество входов) x (высота) x (ширина) x (количество входных каналов).

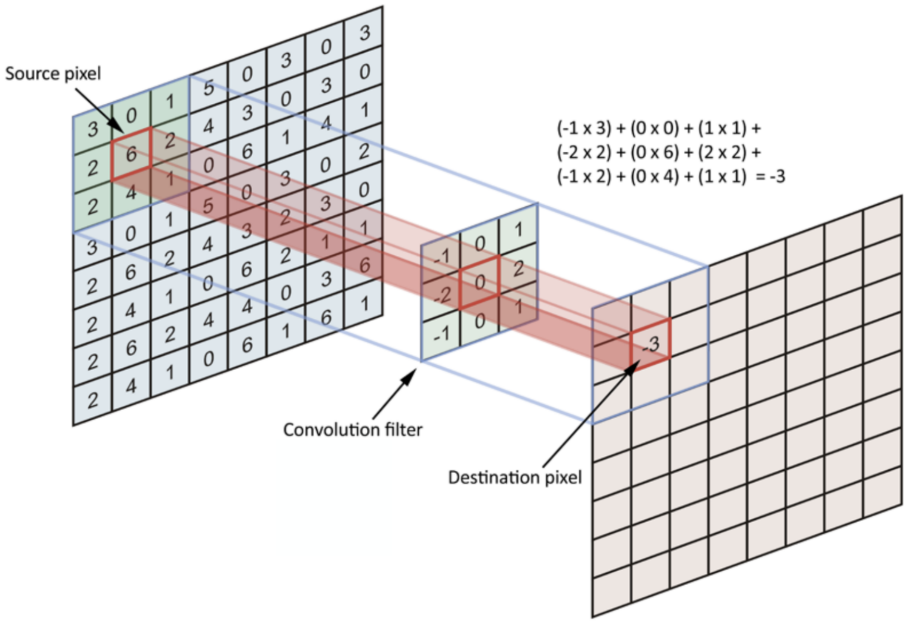


Данные:
Обучающая выборка: 9435 фотографий
Тестовая выборка: 1500 фотографий
Валидационная выборка: 165 фотографий

3.2 Свёрточные слои

Сверточный слой представляет из себя набор карт (другое название – карты признаков, в обиходе это обычные матрицы), у каждой карты есть синаптическое ядро (в разных источниках его называют по-разному: сканирующее ядро или фильтр).

Пример свертки:



Как производится свертка: ядро поочередно скалярно умножается на подматрицы такого же размера исходного изображения.

Ядро представляет из себя фильтр или окно, которое скользит по всей области предыдущей карты и находит определенные признаки объектов. Например, если сеть обучали на множестве лиц, то одно из ядер могло бы в процессе обучения выдавать наибольший сигнал в области глаза, рта, брови или носа, другое ядро могло бы выявлять другие признаки. Размер ядра обычно берут в пределах от 3x3 до 7x7. Если размер ядра маленький, то оно не сможет выделить какие-либо признаки, если слишком большое, то увеличивается количество связей между нейронами. Также размер ядра выбирается таким, чтобы размер карт сверточного слоя был четным, это позволяет не терять информацию при уменьшении размерности в подвыборочном слое

Ядро представляет собой систему разделяемых весов или синапсов, это одна из главных особенно-

стей сверточной нейросети. В обычной многослойной сети очень много связей между нейронами, то есть синапсов, что весьма замедляет процесс детектирования.

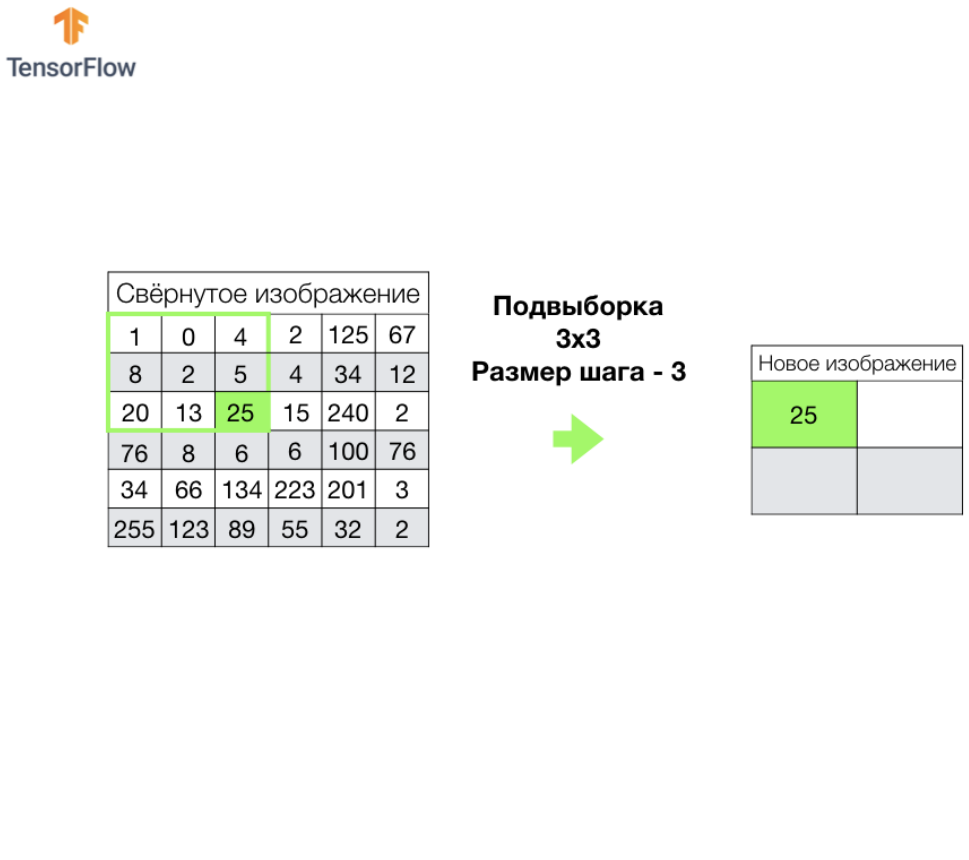
В сверточной сети – наоборот, общие веса позволяет сократить число связей и позволить находить один и тот же признак по всей области изображения.

Хоть обычные полносвязные нейронные сети с прямой связью и могут использоваться для классификации данных, они обычно непрактичны для больших входных данных, таких как изображения с высоким разрешением. Это потребует очень большого количества нейронов, даже для неглубокой архитектуры из-за большого входного размера изображений, где каждый пиксель является входным элементом. Например, полносвязный слой для изображения размером 50 x 50 (маленькое изображение) имеет 2500 весов для каждого нейрона во втором слое. Вместо этого свертка уменьшает количество параметров, позволяя сети быть глубже. Использование регуляризованных весов по меньшему количеству параметров позволяет избежать проблем с исчезающими и разрывными градиентами, которые наблюдаются при обратном распространении в традиционных нейронных сетях. Кроме того, сверточные нейронные сети идеально подходят для данных с топологией в виде сетки (например, изображений), поскольку пространственные отношения учитываются во время свертки и объединения.

3.3 Слои подвыборки

Подвыборочный слой также, как и сверточный имеет карты, но их количество совпадает с предыдущим (сверточным) слоем, их 6. Цель слоя – уменьшение размерности карт предыдущего слоя. Если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не переобучаться.

Пример максимальной подвыборки:



В процессе сканирования ядром подвыборочного слоя (фильтром) карты предыдущего слоя, сканирующее ядро не пересекается в отличие от сверточного слоя. Обычно, каждая карта имеет ядро размером 2x2, что позволяет уменьшить предыдущие карты сверточного слоя в 2 раза. Вся карта признаков разде-

ляется на ячейки 2x2 элемента, из которых выбираются максимальные по значению.
Обычно в подвыборочном слое применяется функция активации RelU. Операция подвыборки (или MaxPooling – выбор максимального) в соответствии с рисунком 6.

3.4 Классификатор

После всех преобразований данные подаются на вход в сглаживающий слой, а оттуда в обычную полносвязную нейронную сеть - классификатор (многослойный перцептрон), которая моделирует сложную нелинейную функцию и ,оптимизируя которую, улучшается качество распознавания. Каждый полносвязный слой в классификаторе извлекает какие-либо характерные черты, и на этой основе сеть делает прогноз. Этот процесс называется прямое распространение. После прямого распространения рассчитывается функция потерь. После расчета функции потерь происходит изменение весов методом обратного распространения. Этот процесс повторяется до тех пор, пока сеть не достигнет оптимальной производительности.

3.5 Функция Активации

Функция активации определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения. Она проверяет произведенное нейроном значение и решает, игнорировать его или оставить активным.

Рассмотрим только те функции, которые мы будем использовать:

ReLu :

$$R(x) = \max(x, 0)$$

Понятно, что ReLu возвращает значение x, если x положительно, и 0 в противном случае. ReLu нелинейна и комбинация ReLu тоже нелинейна.На самом деле, такая функция является хорошим аппроксиматором, так как любая функция может быть аппроксимирована комбинацией ReLu. Следующий пункт — разреженность активации. Представим большую нейронную сеть с множеством нейронов. Использование сигмоиды или гиперболического тангенса будет влечь за собой активацию всех нейронов аналоговым способом. Это означает, что почти все активации должны быть обработаны для описания выхода сети. Другими словами, активация плотная, а это затратно. В идеале мы хотим, чтобы некоторые нейроны не были активированы, это сделало бы активации разреженными и эффективными.
ReLu позволяет это сделать. Представим сеть со случайно инициализированными весами (или нормализированными), в которой примерно 50% активаций равны 0 из-за характеристик ReLu (возвращает 0 для отрицательных значений x). В такой сети включается меньшее количество нейронов (разреженная активация), а сама сеть становится легче. Отлично, кажется, что ReLu подходит нам по всем параметрам. Но ничто не безупречно, в том числе и ReLu.
Из-за того, что часть ReLu представляет из себя горизонтальную линию (для отрицательных значений X), градиент на этой части равен 0. Из-за равенства нулю градиента, веса не будут корректироваться во время спуска. Это означает, что пребывающие в таком состоянии нейроны не будут реагировать на изменения в ошибке/входных данных (просто потому, что градиент равен нулю, ничего не будет меняться). Такое

явление называется проблемой умирающего ReLu (Dying ReLu problem). Из-за этой проблемы некоторые нейроны просто выключатся и не будут отвечать, делая значительную часть нейросети пассивной. Однако существуют вариации ReLu, которые помогают эту проблему избежать. Например, имеет смысл заменить горизонтальную часть функции на линейную. Если выражение для линейной функции задается выражением y = 0.01x для области x < 0, линия слегка отклоняется от горизонтального положения. Существует и другие способы избежать нулевого градиента. Основная идея здесь — сделать градиент неравным нулю и постепенно восстанавливать его во время тренировки.

ReLu менее требовательно к вычислительным ресурсам, чем гиперболический тангенс или сигмоида, так как производит более простые математические операции. Поэтому имеет смысл использовать ReLu при создании глубоких нейронных сетей.

Softmax :

$$Y_i = \frac{e^i}{\sum_{j=1}^{N} e^j}$$

для i-ого нейрона

Softmax — это обобщение логистической функции для многомерного случая. Функция преобразует вектор **X** размерности **N** в вектор **Y** той же размерности, где каждая координата $\mathbf{Y}_i$ полученного вектора представлена вещественным числом в интервале [0,1] и сумма координат равна 1.

Функция Softmax применяется в машинном обучении для задач классификации, когда количество возможных классов больше двух (для двух классов используется логистическая функция). Координаты $\mathbf{Y}_i$ полученного вектора при этом трактуются как вероятности того, что объект принадлежит к классу **i**.

3.6 Оптимизатор

Оптимизаторы определяют оптимальный набор параметров модели, таких как вес и смещение, чтобы при решении конкретной задачи модель выдавала наилучшие результаты. Мы будем использовать оптимизатор Adam.

Адам **-** это алгоритм оптимизации, который можно использовать вместо классической процедуры стохастического градиентного спуска для итеративного обновления весов сети на основе обучающих данных.

Преимущества данного оптимизатора:

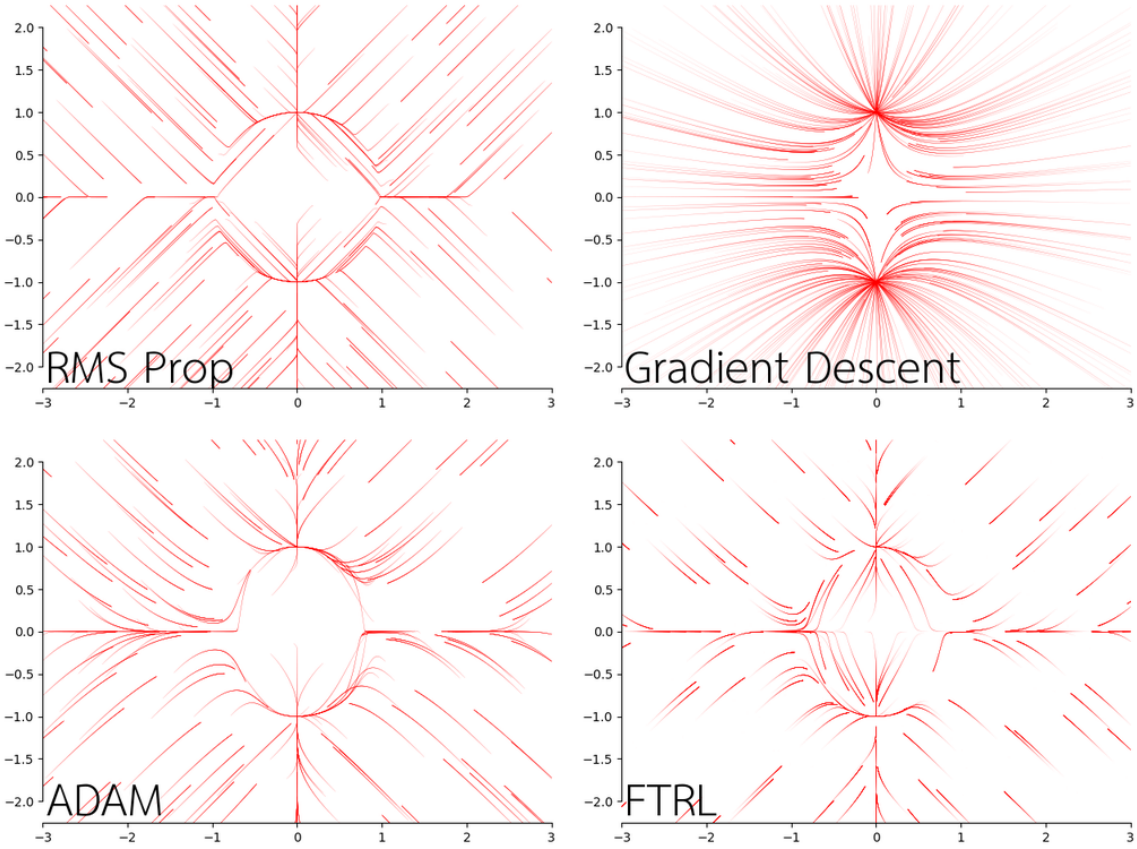
Просто для реализации.

Вычислительно эффективен.

Маленькие требования к памяти.

Инвариант к изменению масштаба градиента по диагонали.

Сравнение различных оптимизаторов:



Подходит для нестационарных целей.

Подходит для задач с очень шумными / или редкими градиентами.

Гиперпараметры обычно требуют небольшой настройки.

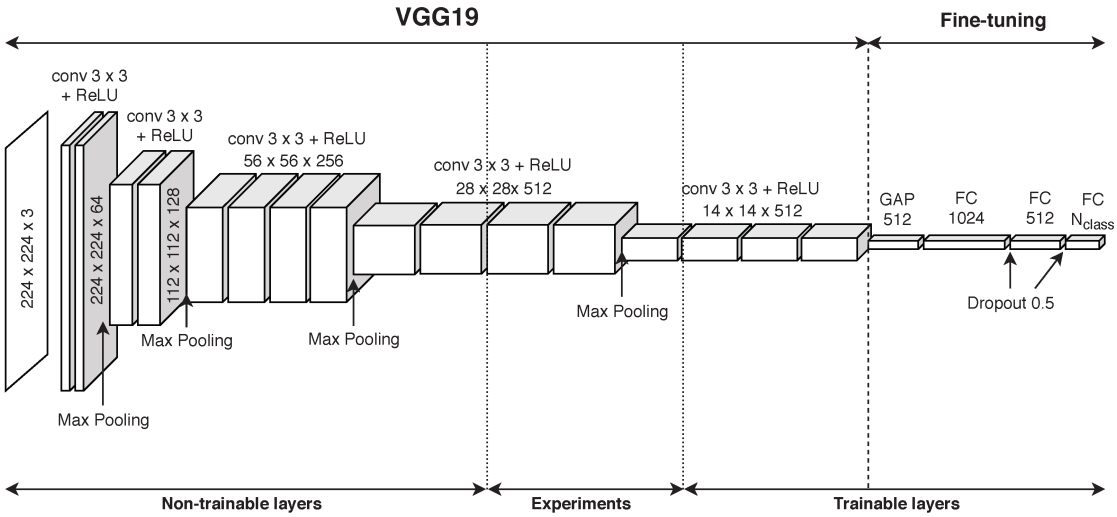
Гиперпараметры имеют интуитивно понятную интерпретацию.

4 VGG19

На вход подается полноцветное изображение с тремя каналами: красный, зеленый, сиий. Затем, оно последовательно проходит через сверточные слои свертки, каждый имеет по 64 фильтра с размером ядер 3x3. Далее, операция максимальной подвыборки уменьшает вдвое линейные размеры карт признаков и результат обрабатывается двумя последовательными слоями свертки со 128 фильтрами с тем же размером ядер 3x3. И так далее. Потом через три сверточных слоя все проходит, потом еще через три. Результирующий тензор размером 7x7x512 подается на полносвязную нейроннуюс сеть с 4096 нейронами двух скрытых слоев и 1000 нейронами выходного слоя. Число 1000 соответствует количеству классов, на которые эта сеть была обучена.

Обучение происходит в 25 эпох, изображения подаются пакетами по 90 фотографий.

Архитектура:



5 Метрики используемые для оценок

Для оценки мы будем использовать 2 метрики: точность(precision) и полнота(recall).

Точность можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а полнота показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.
Матрица ошибок - это матрица в которой первый элемент первой строки это TP, второй элемент первой строки FP, первый элемент второй строки FN, второй элмент второй строки TN.

TP(True Positive) - модель определила, что изображение принадлежит классу и это верно.
FP(False Positive) - модель определила, что изображение принадлежит классу, но это неверно.
FN(False Negative) - модель определила, что изображение не принадлежит классу, но это неверно.
TN(True Negative) - модель определила, что изображение не принадлежит классу и это верно.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

6 Заключение

На 11100 фотографий получилось:
Precision: 96%
Recall: 96%

Список литературы

[1] https://www.kaggle.com/paultimothymooney/interpret-sign-language-with-deep-learning

[2] https://ru-keras.com/convolutional-layers/

[3] https://ru-keras.com/pooling-layers/.

[4] https://ru-keras.com/core-layers/

[5] https://neerc.ifmo.ru/wiki/index.php?title=Batch-normalization

[6] https://www.pyimagesearch.com/2020/03/16/detecting-covid-19-in-x-ray-images-with-keras-tensorflow-and-deep-learning/

[7] https://www.kaggle.com/manavjain99/resnet50

[8] https://www.kaggle.com/brussell757/american-sign-language-classification

[9] https://www.pdfdrive.com/deep-learning-for-computer-vision-with-python-e176309953.html

[10] https://habr.com/ru/post/309508/