https://www.scilook.eu/index.php/slif/article/view/slif21-01-027
DOI: 10.30888/2415-7538.2021-21-01-027

УДК 004.93

FACE RECOGNITION SYSTEM FOR SINGLE-BOARD COMPUTERS BASED ON MACHINE LEARNING METHODS

СИСТЕМА РАСПОЗНАВАНИЯ ЛИЦ ДЛЯ ОДНОПЛАТНЫХ КОМПЬЮТЕРОВ НА ОСНОВЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ

Kolmakov N.P. / Колмаков Н.П.
student / студент
Rusakov A.M. / Русаков А.М.
senior lecturer / ст. преподаватель
SPIN: 1066-8077
MIREA-Russian Technological University, Moscow, PR-t Vernadsky, 78
МИРЭА – Российский технологический университет,
Москва, Пр-т Вернадского, 78

Аннотация. *В работе рассматривается результат обучения архитектур моделей машинного обучения: GoogLeNet, ResNet50 и VGG19. Так же в рамках работы будет расмотрена визуализация обучения каждой архитектуры и сравнение их результативности путём сопостовления точности ответа на определённых количестве изображений. Датасет был заготовлен заранее и ссылку на него можно найти в конце статьи.*

Ключевые слова: *нейронные сети, python, googlenet, vgg19, resnet50, одноплатный компьютер, jetsonnano, визуализация, распознование лиц.*

Вступление

В эпоху бурного развития технологий каждый из нас чувствует себя в безопасности, если знает, что за ним присматривает тот, кто сможет помочь в критический момент или рассказать о том, что произошло. Для решения этой задачи обычно использует видеокамеры, которые выполняют функцию слежения в многолюдных и важных местах, уже практически на всех критически-важных местах. Однако, недостаточно чтобы камеры только наблюдали, что происходит в поле их зрения, а зачастую необходимо аналитически оценивать обстановку и случае необходимости предпринять какие-либо действия: например, вызвать полицию, пожарных или кого бы то ни было в зависимости от ситуации. Для подобной цели уже существует целое направление в Computer Science, оно называется Computer Vision [1]. За последние годы совершен колоссальный прорыв в данном направлении, с видеорядом снятым на камеру можно извлечь много информации, начиная с классификации объектов до его полной идентификации. Но как понять, например, что нежелательный человек находится в помещении и об надо предупредить администрацию? Обычно, в подобном случае камеры отправляют весь поток данных на сервер, где в свою очередь происходит обработка и последующие действия. У данной методики есть два небольших, но очень существенных недостатка - связь с сервером и дороговизна всей системы видеонаблюдения. Если пропадёт электричество, то от камеры не будет никакого смысла. Что если, каждая камера, будет сама иметь «мозги», и сможет работать автономно. Таким образом при отключении электропитания видеокамера сможет сама независимо от обстоятельств выполнить необходимые действия. В этой статье будет предпринята попытка решить

данные проблемы. Результаты, полученные в последующей части повествования можно применить не только к тому случаю, который описан в статье, но и в любых необходимых читателю целях.

В качестве центра вычисленной системы, которой планируется дооснастить видеокамеры, будет использован одноплатный компьютер от компании Nvidia - Jetson Nano. Основным преимуществом перед ближайшим конкурентом, Raspberry Pi, является «заточенность» под работу с видео средствами нейронных сетей.

Таблица 1.

Технические характеристики JetsonNano.

GPU	NVIDIA Maxwell архитектура с 128 NVIDIA CUDA® ядрами
CPU	Четырёх ядерный ARM Cortex-A57 MPCore процессор
Оперативная память	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
Встроенная память	16 Гб eMMC 5.1 + 64 Гб SD
Кодирование видео	250MP/sec 1x 4K @ 30 (HEVC) 2x 1080p @ 60 (HEVC) 4x 1080p @ 30 (HEVC) 4x 720p @ 60 (HEVC) 9x 720p @ 30 (HEVC)
Декодирование видео	500MP/sec 1x 4K @ 60 (HEVC) 2x 4K @ 30 (HEVC) 4x 1080p @ 60 (HEVC) 8x 1080p @ 30 (HEVC) 9x 720p @ 60 (HEVC)
Слот для камеры	12 линий (3x4 or 4x2) MIPI CSI-2 D-PHY 1.1 (1.5 Gb/s каждая)
Подключение к сети	Gigabit Ethernet, M.2 Key E
Порты для подключения к дисплеям	HDMI 2.0 and eDP 1.4
USB	4x USB 3.0, USB 2.0 Micro-B

На данный момент Jetson Nano от фирмы NVidia [2] является, по нашему мнению, оптимальным решением. Так как одноплатный компьютере обладает необходимыми свойствами — энергоэффективность процессора ARM Cortex-A57 MPCore и встроенные графический процессор на архитектуре Maxwell со 128 ядрами CUDA, что позволяет достаточно быстро обрабатывать видео

высокого разрешения. При необходимости, также можно подключить различные датчики, что позволит более полно контролировать ситуацию в пределах видимости камеры.

Помимо самого Jetson Nano, потребуется минимальный набор для полноценной работы с фото и видео фрагментами, в различных обстановках. Для фиксации видео понадобится камера, в рамках этой статьи используется камера Raspberry Pi 8MP, благодаря которой "были" сделаны все записи.

В случаи неполадки с электросетей потребуется источник питания, для этих целей рекомендую выбрать платформу T208 [3] с шестью отсеками для аккумуляторов формфактора 18650. Все отсеки были заполнены аккумуляторами ёмкостью 3200 mAh каждый. Суммарная ёмкость составляет 18600 mAh позволяет проработать нашей камере до тридцати часов.

Таблица 2.

Технические характеристики ИБП для Jetson Nano – T208

Потребляемая мощность	5Vdc ±5%, ≥4A
Выход ИБП	5,1 В ±5% Макс. 8A
ИБП ток заряда	3A через разъем DC
Напряжение клеммной батареи	4,24 В
Порог перезарядки	4,1 В

В рамках данной статьи проблематично составить датасет с минимально-достаточным количеством данных, при использовании обычной камеры, поэтому, в роли тестовых данных будут использоваться фотографии людей, которые довольно-таки известны, чьих изображений более чем достаточно для обучения последующих архитектур нейронных сетей. Позвольте представить вам список этих людей:

1. Бенедикт Камбербэтч
2. Эмма Ватсон
3. Джонни Депп
4. Леонардо ДиКаприо
5. Марго Робби
6. Скарлет Джоханнсон

Для создания датасета тестовых данных с лицами была написаны консольная утилита, которая может найти необходимое количество фотографий актёров в поисковых системах и произвести необходимую обработку фотографий для дальнейшей обработки средствами нейронных вычислений, данный инструмент и полученный датасет можно получить здесь [4].

Представленные дальнейшие архитектуры были разработаны с помощью фреймворка PyTorch от компании Facebook [5]. Одним из основных преимуществ данной архитектуры является тот факт, что для обучение сети используется только необходимое количество видеопамяти в отличии от Tensorflow и Keras [6], которые в свою очередь резервируют всю доступную видеопамять.

Сверхточная нейронная сеть VGG19

Сверхточная нейронная сеть VGG19 выиграла в соревновании ImageNet Large Scale Visual Recognition Challenge (ILSVRC) от ImageNet [7]. В этом соревновании задействовано большое количество изображений из тысячи категорий. Но есть одна уникальная вещь, связанная с VGG19 — вместе огромного количества гипер-параметров, было принято решение использовать свёрточные слои с размером фильтра 3 x 3 с шагом 1, так же использовали заполнение, а в слое полной субдискритизации (maxpool) используется фильтр размером 2x2 с шагом равным двум. Именно такая последовательность слоёв соблюдается на протяжении всей архитектуры. Завершает всю композицию два полносвязный слоя, выход из которых является функция активации Softmax. Число 19 в названии обозначает количество слоёв, у которых есть веса. Эта нейронная сеть содержит около 138 миллионов параметров, а сам файл с сохранёнными параметрами весит около 500 Мб.

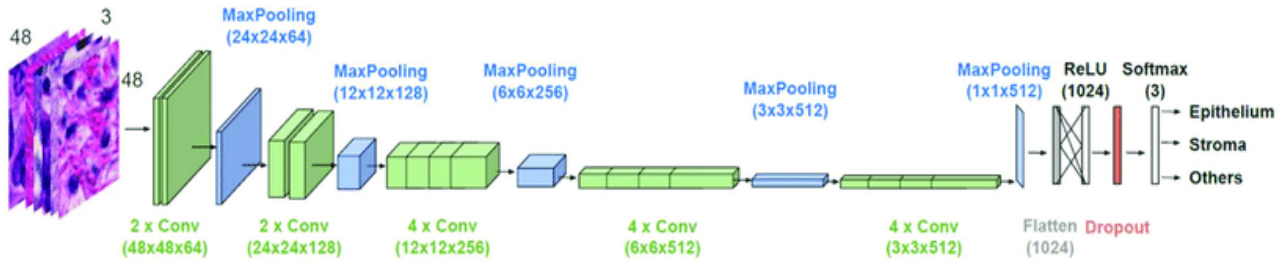

Рисунок 1. – Архитектура VGG19 источник

Источник: https://www.researchgate.net/figure/Architecture-of-the-employed-CNN-model-VGG19-is-used-for-transfer-learning-Parameter_fig2_336194252

Теперь необходимо оценить качество модели для этого нужно составить матрицу смежности.

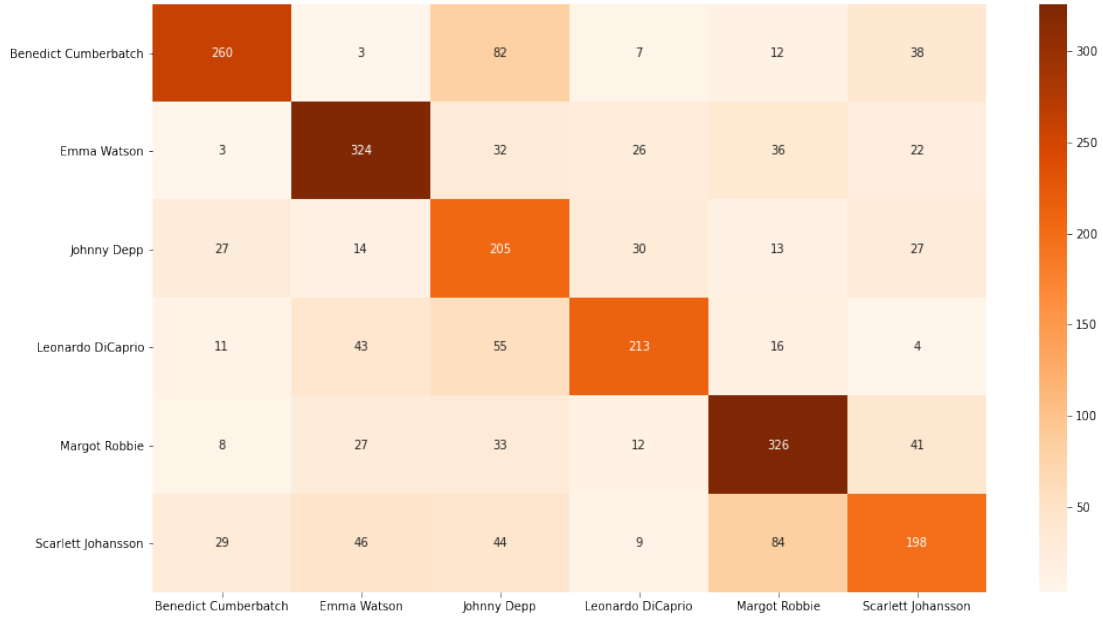

Рисунок 2. – Матрица смежности сверточной нейронной сети VGG19

Авторская разработка

Архитектура VGG 19 была обучена в течении, 30 эпох, с использованием оптимизатора SGD, размер пакета передаваемых данных за итерацию составлял 16, а размер обучающего шага был равен рекомендуемым, в описании самого оптимизатора, 0.0001. Ниже представлены график, которой целиком показывает зависимость каждого параметра от количества итераций.

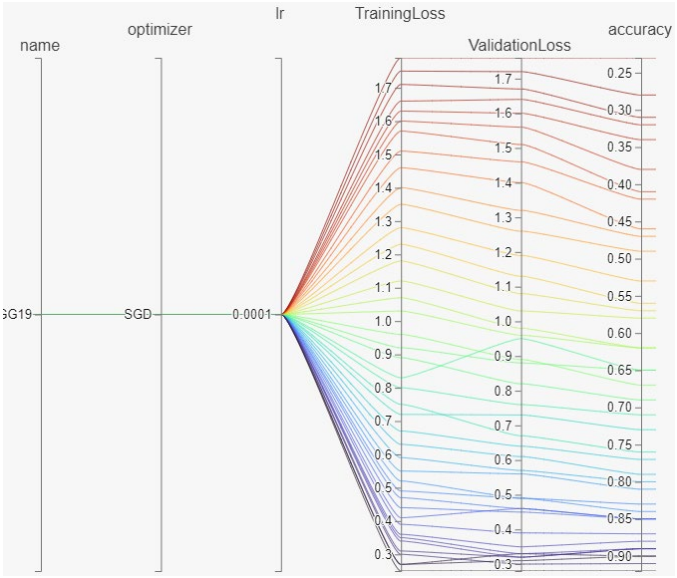

Рисунок 3 – HParams для VGG19

Авторская разработка

Более наглядную работу архитектуры можно продемонстрировать с помощью веб-интерфейса созданного с помощью фреймворка streamlit [8] для Python.

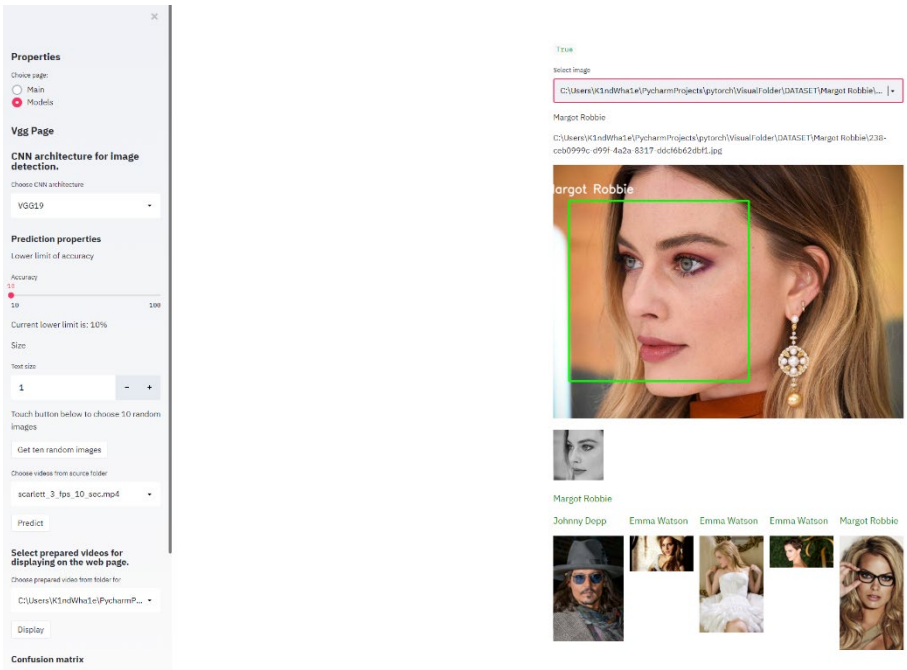

Рисунок 4 - Пример работы архитектуры VGG19

Авторская разработка

На рис 4, представлены результаты работы нейронной сети, на статических изображениях, результат, который получила сеть. Надписи над фотографиями, зелёный цвет обозначает, что предсказанный результат совпадает с действительностью, а красный — ошибка в предсказании. Так же, в левом нижнем углу есть кнопка "predict", нажав на которую начнётся анализ заранее записанного видеофайла.

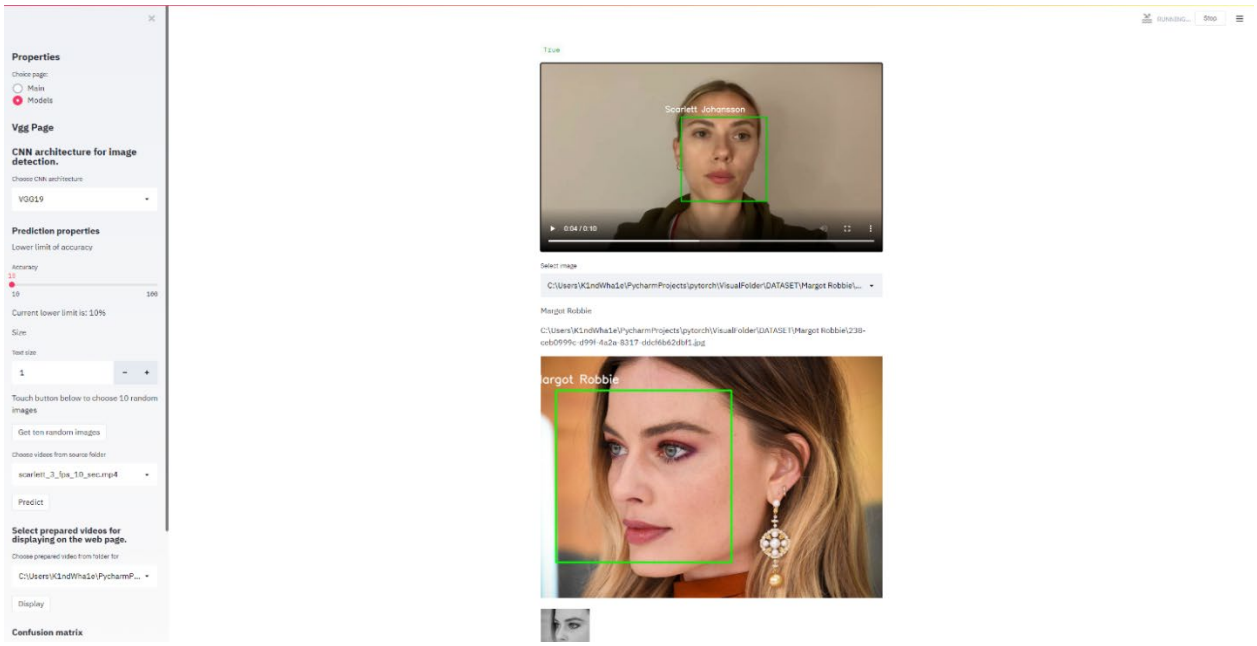

Рисунок 5 — Пример обработки видео-фрагмента с помощью архитектуры сверточной нейронной VGG19.

Авторская разработка

Сверточная нейронная сеть VGG 19 — продемонстрировала довольно-таки неплохой результат, это можно утверждать взглянув на матрицу смежности для это сети. Возможно, это не лучший результат и оставшиеся 2 архитектуры проявят себя лучше.

Архитектура GoogLeNet

Архитектура GoogLeNet — это название не так просто, как могло бы показаться, это отсылка к родоначальнику нейронных архитектур Yann Lecuns и его нейронной сети LeNet 5.

Все слои свёртки, включая те, что входят в состав Inception модулей, используют выпрямленную линейную активацию. Размер подаваемого изображения на вход нейронной сети 224x224 в цветовом канале RGB со средней вычитающей. "3x3 уменьшение" и "5x5 уменьшение" обозначают количество фильтров 1x1 в восстановительном слое, используемых перед свёртками 3x3 и 5x5. Количество фильтров 1x1 в проекционном слое можно увидеть после встроенного слоя субдискретизации в колонке "pool proj" (pool projection). Все эти слои уменьшения/проекции тоже используют выпрямительную линейную активацию.

Нейронная сеть была разработана с учётом вычислительной эффективности и практичности, таким образом расчёт может производиться на

одном устройстве, даже на тех, которые имеют ограничения в вычислительных ресурсах и небольшим объёмом памяти. GoogLeNet — состоит из 22 глубоко-связных слоёв, которые принимают участие в расчёте параметров или же 27 с учётом субдискретизированных слоёв. Общее количество, используемых для построения сети, составляет около 100, однако это число зависит от используемой системы инфраструктуры машинного обучения. Использование (adverage pooling) среднего пула перед классификатором обусловленно тем, что в реализации используется дополнительный линейный слой. Эта концепция позволяет легко адаптировать и тонко настроить сеть для разных наборов классов. При создании архитектуры разработчики обнаружили, что использование среднего пула, вместо полно-связного слоя улучшает точность, приблизительно, на 0.6%, однако использования слоя «dropout» остаётся необходимым даже после замены полно-связанного слоя. Учитывая, что глубина сети очень большая, то могла бы возникнуть проблема с обратным распределением градиентов через все слои архитектуры. Для устранения эффекта переобучения. Интересным решением, со стороны исследователей, было добавление вспомогательных классификаторов (auxiliary classifiers). Эти классификаторы принимают форму небольших свёрточных сетей, помещённые поверх выходных данных модулей Inception 4(a) и 4(b). В процессе обучения, потери основной сети и вспомогательных слоёв складываются, а во время получения результатов эти слои можно отбросить.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Рисунок 6 - GoogLeNet в таблице

Источники: https://arxiv.org/pdf/1409.4842.pdf

Подробная структура сети, включая вспомогательные классификаторы, выглядит следующим образом:

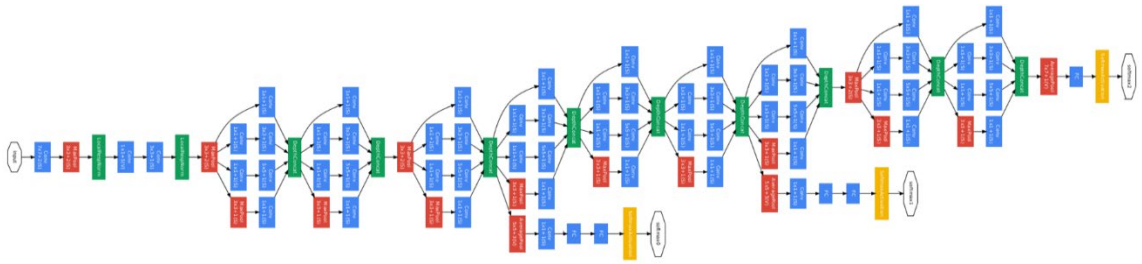

Рисунок 7 - Архитектура GoogLeNet.

Источники: https://arxiv.org/pdf/1409.4842.pdf

Пришло время проверить работоспособность GoogLeNet, на рисунке ниже (номер его) представлена матрица смежности:

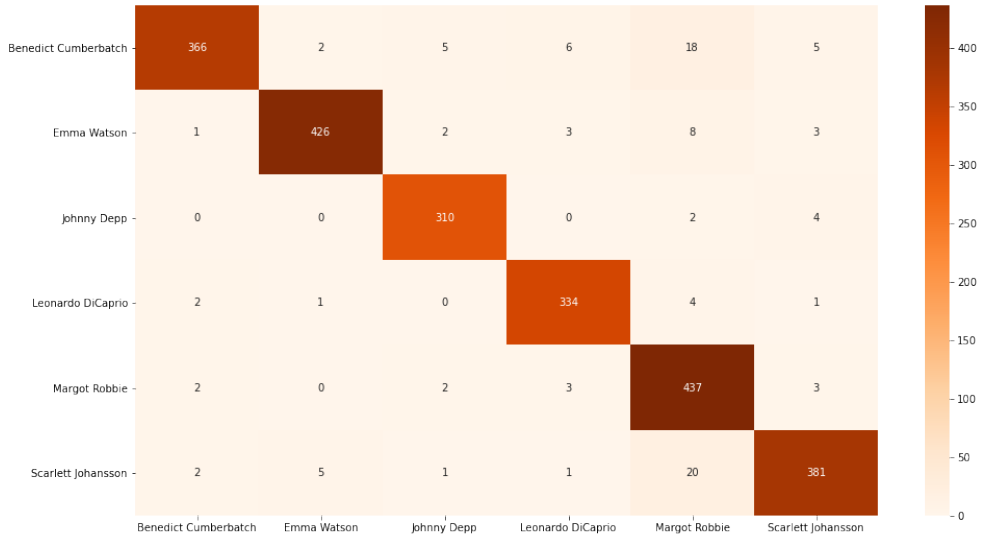

Рисунок 8 – Матрица смежности для архитектуры GoogLeNet.

Авторская разработка

Результаты очень хорошие и заметно отличаются от VGG 19, здесь очевидное преимущество на стороне нейронной сети от компании Google. Чтобы сделать более ли менее объективный выбор архитектуры для дальнейшего использования, нам понадобится ещё, которая может посостязаться с GoogLeNet или покажет схожий результат результат.

Архитектура ResNet50

Архитектура ResNet — была попыткой улучшить уже существующую архитектуру VGG19, в итоге получилось две вариации архитектуры, в первой увеличили количество слоёв, а во второй, помимо увеличения слоёв, добавили новое соединение.

Разработчики решили протестировать две версии архитектуры простую/остаточную и проверить какая из версий проявит себя лучше. Поскольку одна архитектура основана на другой, поэтому свойства присущие обеим архитектурам будут описан в простой ResNet.

Философия этой вариации ResNet позаимствована у VGG. У всех свёрточных слоёв размер фильтра 3x3 и следуют двум простым правилам проектирования:

1. Для одного и того же размера карты выходных объектов, слои имеют одинаковое количество фильтров.

2. В дальнейшем размер карт уменьшится в двое, количество фильтров удваивается чтобы сохранить сложность вычисления на каждый слой.

Далее выполняется понижающая дискретизацию, непосредственно, свёрточными слоями, имеющими шаг 2. Нейронную сеть завершает средняя глобальная операция подвыборки (субдискретизации) пространственных данных и полносвязный слой с функцией активацией softmax. Общее количество слоёв, у которых есть веса - 34, показано на рисунке 9, блок-схема посередине.

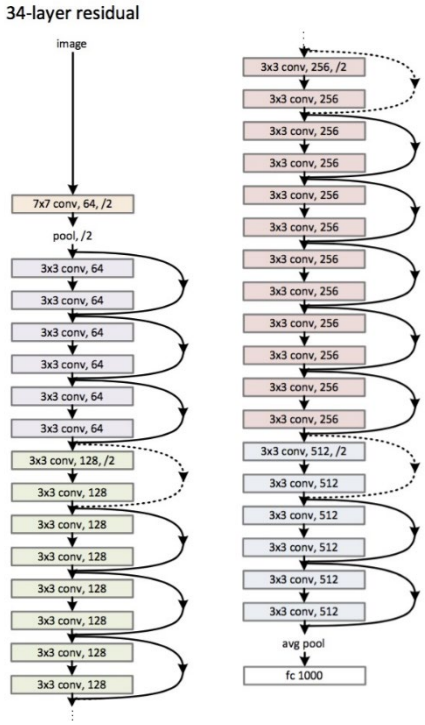

Рисунок 9 – Структура архитектуры ResNet50.

Источники: https://arxiv.org/pdf/1512.03385.pdf

Также стоить отметить, что даже простая реализация имеет меньшее количество фильтров и меньшую сложность для вычислений, чем сети семейства VGG. Для вычисления этих 34 слоёв требуется около 3.6 миллиарда FLOPs, что является 18% от количества операций требуемых для VGG-19 (19.6 миллиардов).

Остаточная сеть — базируется на выше изложенной архитектуре, с "небольшим" изменением в виде быстрых соединений, рисунок 9. Быстродействующие идентификаторы $y = F(x,\{W_i\}) + x$ могу использоваться только тогда, когда размерность выходного значения и входного идентичны. При увеличении размерности нужно рассмотреть два варианта:

Ускоренный переход, по-прежнему, выполняет сопоставление идентификаторов с дополнительными нулевыми записями, необходимыми для увеличения размерности. Эта функция не включает в себя дополнительных параметров.

Ускоренный переход $y = F(x, \{W_i\}) + W_s x$ используется для сопоставления размеров.

Для обеих вариантов, когда переход идёт через карты объектов двух размеров, они выполняются с шагом 2.

На вход первому слою ResNet подаётся изображение размером 224x224 в цветовом канале RGB. Для обучение использовались следующие: функция оптимизации SGD, с шагом обучение, подобранным путём перебора, равного 0.0001. Размер пакетной валидации равен 16, иначе, при других значениях не хватало вычислительной памяти видеоадаптера. Чтобы продемонстрировать результат работы ResNet50 была составлена матрица смежности взгляните на рисунке 10.

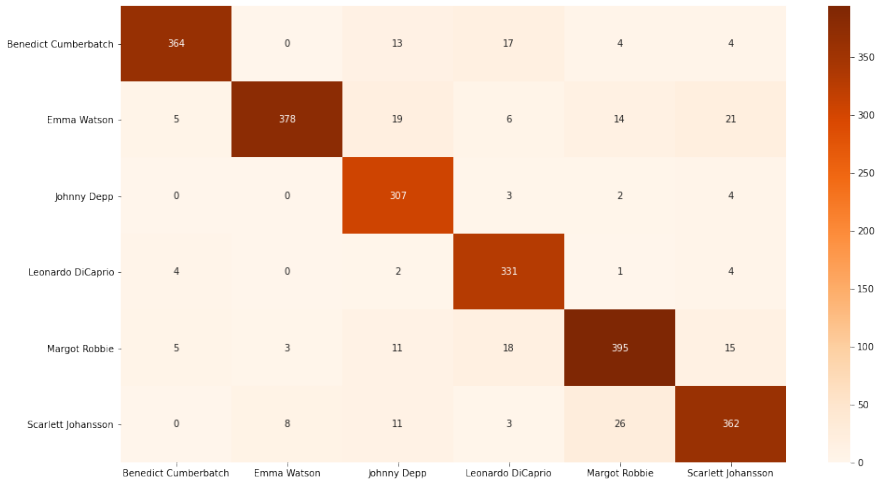

Рисунок 10 - Матрица смежности для архитектуры ResNet50.

Авторская разработка

Второй пример работы с конкретными данными в веб-приложении, можно увидеть на скриншоте ниже.

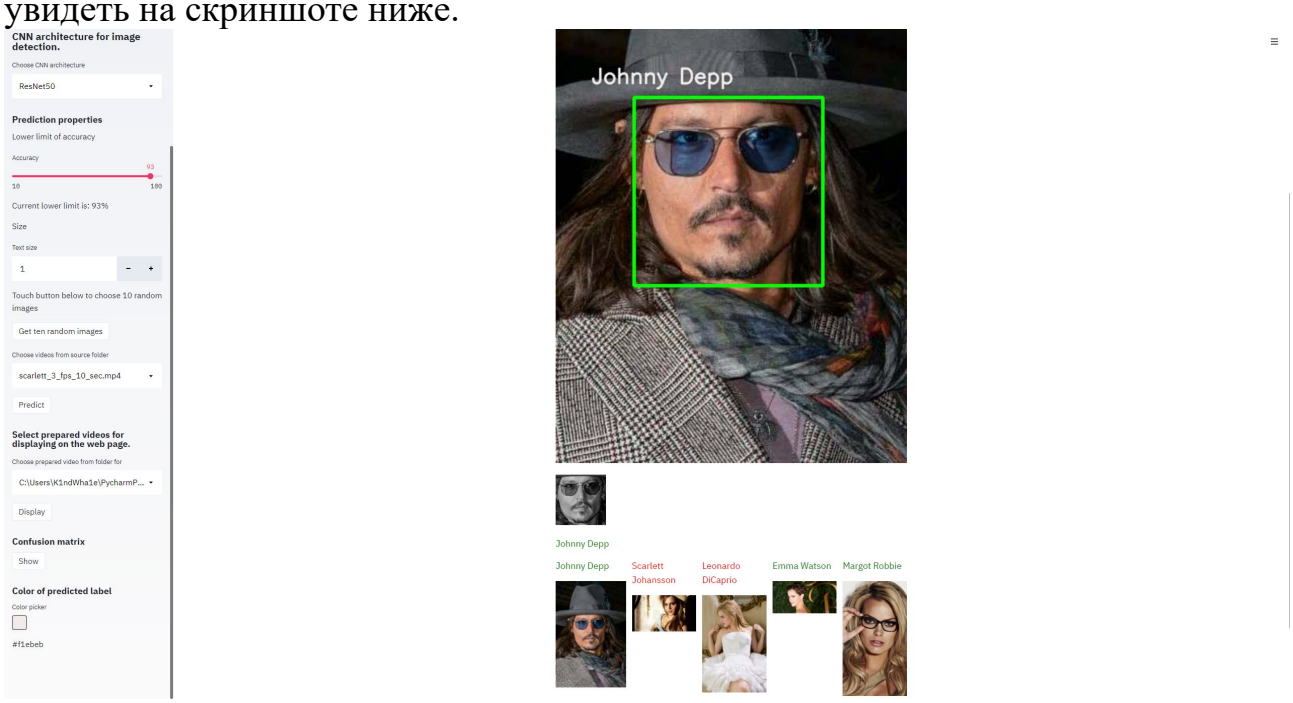

Рисунок 11 - Пример работы архитектуры ResNet50 в веб-приложении.

Авторская разработка

Сравнение архитектур

Для определения подходящей архитектуры были подобраны необходимы факторы и занесены в таблицу 3:

Таблица 3.

Сравнение архитектур нейронных сетей.

Название архитектуры	Количество слоёв	Результат [с минимальной точностью/количество изображений]			Итоговый вес файла с параметрами
-	-	30%/100	75%/100	95%/100	-
VGG19	19	89 %	88 %	73 %	≈ 545 Мб
GoogLeNet	37	90 %	86 %	80 %	≈ 41 Мб
ResNet50	50	90 %	89 %	83 %	≈ 92 Мб

Авторская разработка

Чтобы более наглядно оценить точность работы той или иной архитектуры, необходимо построить несколько графиков, что уже умеет делать представленная в данной статье программа. Попробуем поэкспериментировать с точностью распознавания:

Выставим точность на отметку 10%:

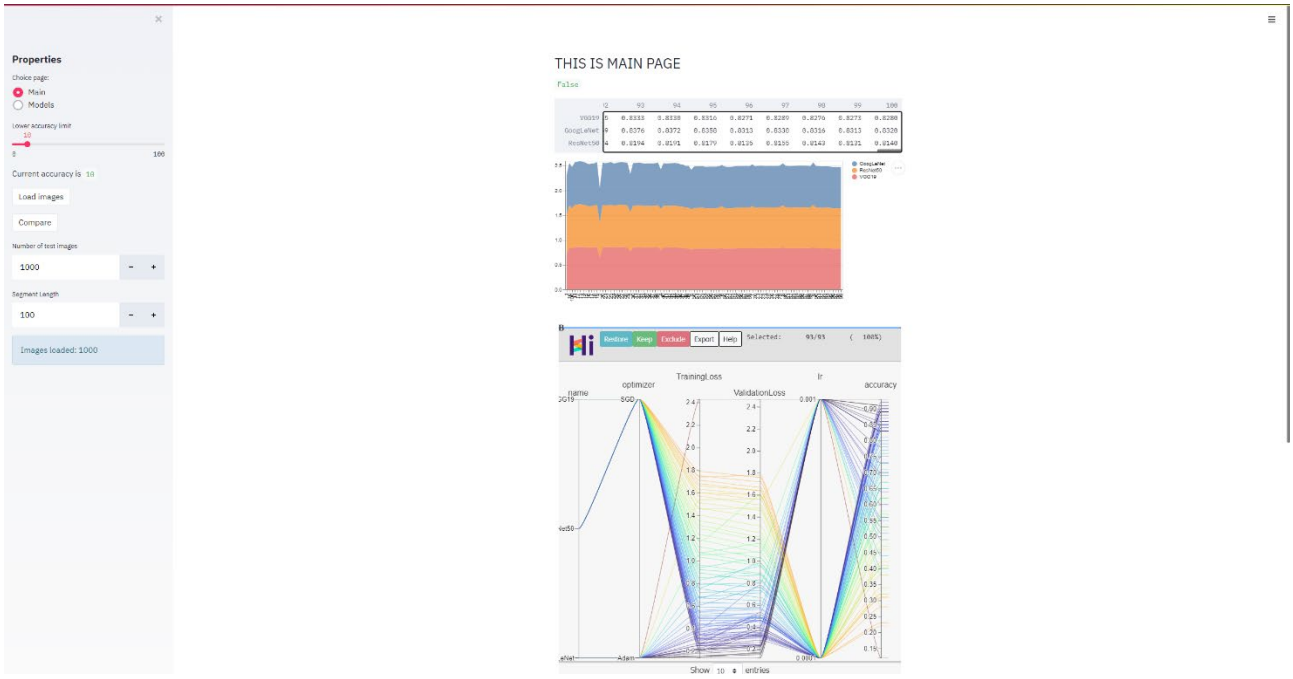

Рисунок 12 - Сравнение архитектур VGG19, ResNet50 и GoogLeNet, с выставленной точностью = 10%

Авторская разработка

Первый график показывает процент точности распознавания на каждом участке, каждый участок разбит на сто участков по десять изображений в каждом, каждый последующий сегмент включает в себя результат предыдущего.

В этом эксперименте все три архитектуры проявили себя достойно с результативностью > 80%. Лучший результат у GoogLeNet = 83,2%, второй результат у VGG19 = 82,8%, третий результат ResNet50 = 81,4%

Теперь, выставим точность на отметку 77%

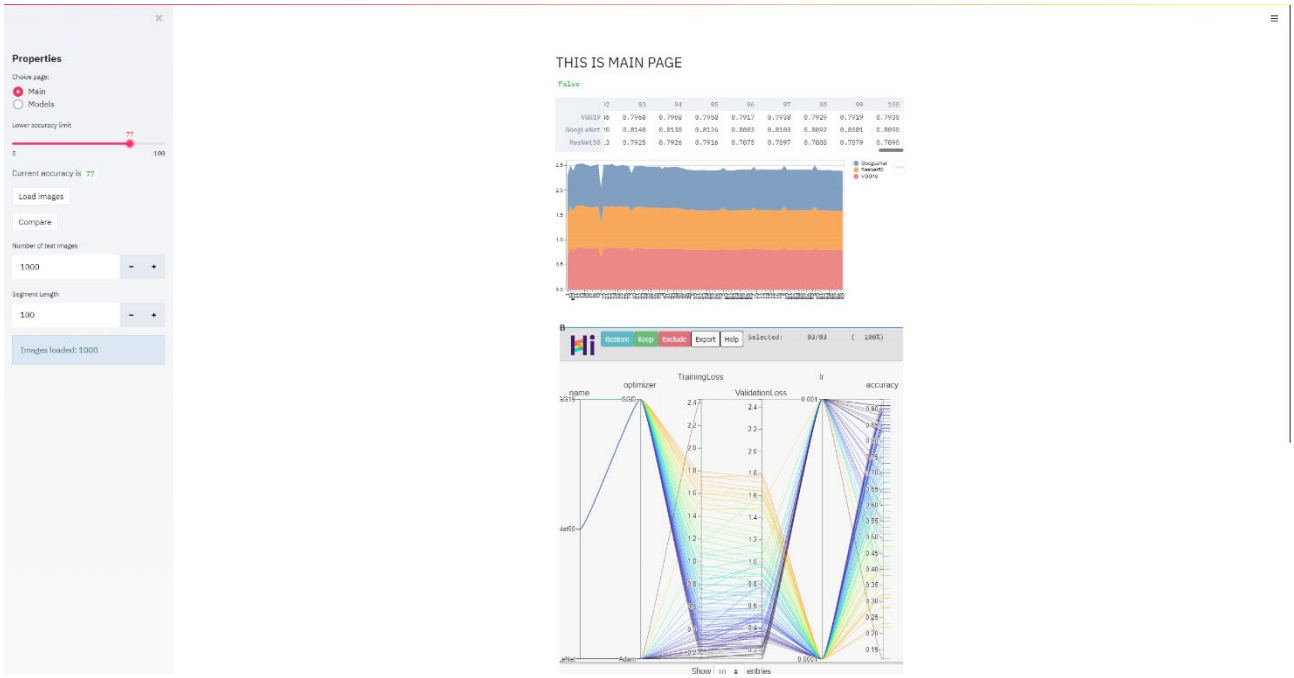

Рисунок 13 - Сравнение архитектур VGG19, ResNet50 и GoogLeNet, с выставленной точностью = 77%.

Авторская разработка

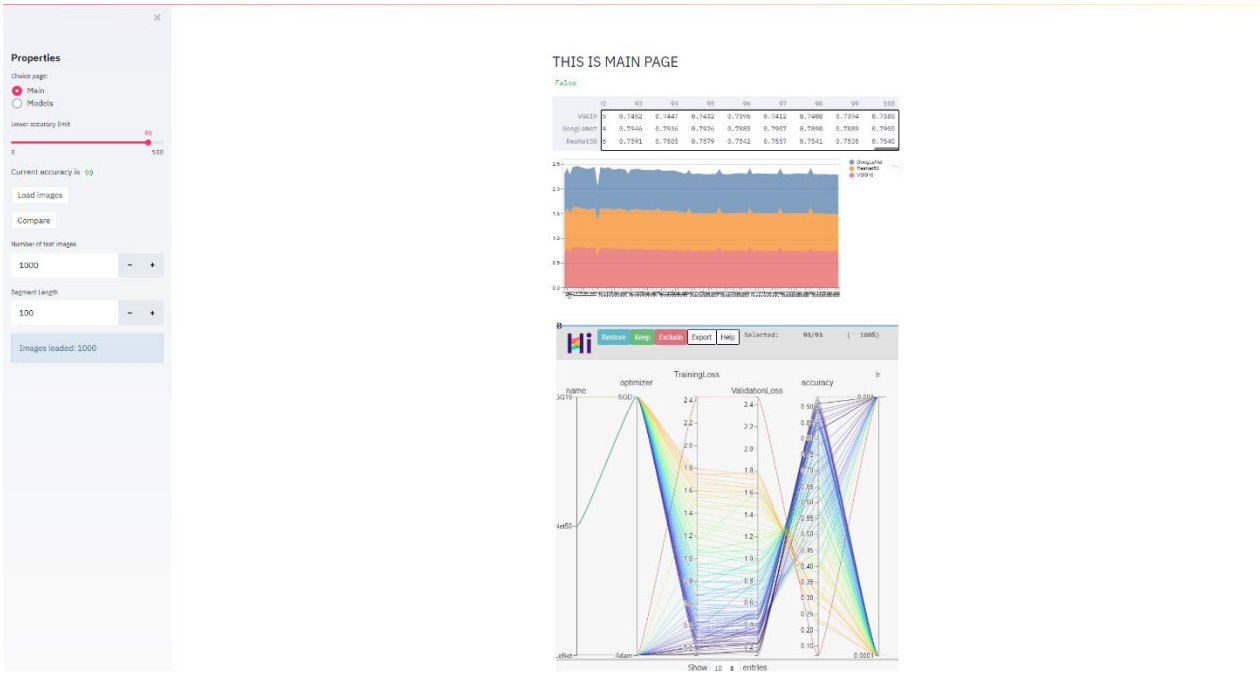

Рисунок 14 – Сравнение архитектур VGG19, ResNet50 и GoogLeNet, с выставленной точностью = 90%.

Авторская разработка

Выбор новой точности, которая очень сильно отличается от предыдущей, так как на других значениях результат меняется незаметно. Посмотрим на рисунок 13, результат, который архитектуры показали в этот раз. Первый результат попрежнему сохраняет GoogLeNet = 80,9%, второй результат у VGG19 = 79,3%, а третий результат у ResNet50 = 78,9% сейчас. Хоть и результат заметно уменьшился до промежутка [78,9%; 80,9%], но он попрежнему остаётся достойным. Проведём последний эксперимент с нижней границей точности равной 90%.

Точность и всех архитектур, без исключения, вновь стала меньше. Как видно из рисунка 14, первый результат у GoogLeNet = 79%, второй результат у ResNet50 = 75,4%, третий результат у VGG19 = 73,8%.

Выводы

Результаты выполненного эксперимента, показывают один интересный факт, чтобы получить приемлемый результат — необязательно выкрутить ползунок с минимальной точностью на максимум, для достижения конкурентноспособных результатов, необходимо провести ряд подобающих опытов, которые были произведены по ходу статьи.

Разработанное ПО для этой статьи, и однопалатного компьютера обеспечивает необходимый минимум для контроля ситуации в местах со средней или низкой активностью.

Для анализа видеоряда необязательно использовать все три архитектуры, если Вам необходимо в кратчайшие сроки развернуть похожую систему, то рекомендую воспользоваться GoogLeNet, так как на обучение данной архитектуры уходит меньшее количество времени и достигается приличная точность по сравнению с остальными архитектурами из данной статьи. Несмотря на впечатляющие показатель GoogLeNet не нужно забывать про VGG19 и ResNet50, полный набор всех трёх архитектур повышает вероятность на получение более точного результата.

Литература:

1. Voulodimos A. et al. Deep learning for computer vision: A brief review //Computational intelligence and neuroscience. – 2018. – Т. 2018.
2. NVIDIA JETSON NANO // URL: https://www.nvidia.com/ru-ru/autonomous-machines/embedded-systems/jetson-nano/ (дата обращения: 10.04.2021).
3. NVIDIA Jetson Nano T208 Power Management Board with 6-cell 18650 UPS // URL: https://a.aliexpress.com/_Afk2ef (дата обращения: 10.04.2021).
4. Used Image’s DataSet // URL: https://vk.cc/c1b2TC (дата обращения: 10.04.2021).
5. PYTORCH DOCUMENTATION // URL: https://pytorch.org/ (дата обращения: 10.04.2021).
6. Keras & TensorFlow // URL: https://keras.io/ (дата обращения: 10.04.2021).
7. ImageNet image database // URL: http://image-net.org/ (дата обращения: 10.04.2021).
8. STREAMLIT DOCUMENTATION // URL: https://streamlit.io/ (дата обращения: 10.04.2021).

References.

1. Voulodimos A. et al. Deep learning for computer vision: A brief review //Computational intelligence and neuroscience. – 2018. – Т. 2018.
2. NVIDIA JETSON NANO. (n.d.). Retrieved April 10, 2021, from https://www.nvidia.com/ru-ru/autonomous-machines/embedded-systems/jetson-nano/
3. NVIDIA Jetson Nano T208 Power Management Board with 6-cell 18650 UPS (n.d.). Retrieved April 10, 2021, from https://a.aliexpress.com/_Afk2ef
4. Used Image's DataSet Retrieved April 10, 2021, from https://vk.cc/c1b2TC
5. PYTORCH DOCUMENTATION. (n.d.). Retrieved April 10, 2021, from https://pytorch.org/
6. Keras & TensorFlow (n.d.). Retrieved April 10, 2021, from https://keras.io/
7. ImageNet image database (n.d.). Retrieved April 10, 2021, from http://image-net.org/
8. STREAMLIT DOCUMENTATION (n.d.). Retrieved April 10, 2021, from https://streamlit.io/

Abstract. *The paper considers the result of learning machine learning model architectures: GoogLeNet, ResNet50, and VGG19. Also in the framework of the work, the visualization of the training of each architecture and the comparison of their performance will be examined by comparing the accuracy of the response on a certain number of images. The dataset was prepared in advance and a link to it can be found at the end of the article.*

Key words: *neural networks, python, googlenet, vgg19, resnet50, single-board computer, jetsonnano, visualization, face recognition.*

Статья отправлена:21.04.2021 г.