Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б11-мм

Задача автоматизации распознавания CAPTCHA в контексте парсинга данных о статусе грузовых контейнеров при помощи CGAN архитектуры

Задорожный Александр Сергеевич

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
асп. каф. инф, м.н.с лаб. ТиМПИ СПб ФИЦ РАН Олисеенко В.Д.

Консультант:
м.н.с лаб. ТиМПИ СПб ФИЦ РАН Корепанова А.А.

Санкт-Петербург
2022

Оглавление

Введение

В настоящий момент существует огромное количество транспортных компаний, а объемы морских грузоперевозок в мире превышают все исторические максимумы.[1] Поэтому в данном срезе актуально множество задач, требующих автоматизации, как, например, процесс отслеживания контейнеров. В рамках этой задачи пользователю необходимо собственноручно заходить на сайт каждой компании и проверять груз по его трек-номеру.

Для автоматизации решения задачи парсинга данных о статусе контейнера, а также задач связанных со сбором актуальных каталогов компаний, представляющих услуги в области перевозки грузов, быстрого поиска подходящего перевозчика, расчётом стоимости и времени доставки был разработан сервис Cargotime.ru. Однако при сборе данных по запросу пользователя у сервиса Cargotime.ru возникают проблемы с запросом к некоторым сайтам перевозчиков. Это связано с тем, что на сайтах установлены системы, которые препятствуют автоматическому сбору информации и блокируют отслеживание грузов из-за следующих причин: предотвращение спама или хакерской атаки, защита пользовательских данных, снижение риска возникновения проблем с безопасностью.

Одним из видов защиты сайта от автоматизированного сбора данных является CAPTCHA или же Completely Automated Public Turing Tests to Tell Computers and Humans Apart, — CAPTCHA представляет собой некоторый защитный код, где необходимо написать слово, указанное на изображении, решить несложное арифметическое уравнение, собрать пазл, кликнуть на соответствующую картинку и т.д., чтобы потом можно было совершить какое-либо действие на сайте. Это подводит нас к сути проекта, т.е создание умного автоматического решателя (программы-декодера), который мог бы справиться со сложными CAPTCHA. И хотя, как было сказано ранее, они помогают в обеспе-

[1] Statista Research Department Capacity of container ships in seaborne trade from 1980 to 2021[Электронный ресурс]. (Дата обращения 15.11.2022)

чении безопасности, это в значительной мере усложняет получение открытых пользовательских данных. Стоит также отметить законность наших действий, согласно п.1 ст.7 и п.1 ст.5, федерального закона «Об информации, информационных технологиях и о защите информации»[2]: “Общедоступная информация может свободно использоваться любым лицом и передаваться одним лицом другому лицу”. В частности, это говорит, что получение данных о грузе от различных компаний для хранения в одном месте — совершенно законно. В данной работе будут рассмотрены CAPTCHA, защищающие сайт компании «YML»

Степень разработанности темы. Работа велась на базе лаборатории теоретических и междисциплинарных проблем информатики (ТиМПИ) СПб ФИЦ РАН. По задаче, решаемой в рамках текущей учебной практики, научным руководителем и другими сотрудниками лаборатории ТиМПИ была произведена формализация задачи, а также сбор первоначального набора данных.

[2]Федеральный закон “Об информации, информационных технологиях и о защите информации”[Электронный ресурс]. (Дата обращения 07.12.2022)

Дата сборки: 15 декабря 2022 г.

1. Постановка задачи

Целью работы является создание решателя CAPTCHA компании “YML” для автоматизации агрегации данных о грузовом контейнере логистическим порталом Cargotime.ru. Для её выполнения были поставлены следующие задачи.

1. Анализ отличительных признаков набора данных CAPTCHA, защищающих сайт компании «YML».

2. Очистка имеющегося датасета от случайно попавших ошибочных данных, т.к ни одна модель машинного обучения не выдаст осмысленных результатов, если ей предоставить данные с искажениями. К ним относят: поврежденные и неточные записи из таблицы или базы данных, нетипичные для конкретной задачи сведения, неинформативные (дублированные) или несогласованные (представленные в разных регистрах или форматах) данные.

3. Компаративный анализ существующих подходов к решению рассматриваемого вида CAPTCHA.

4. Анализ подходов и методов обучения нейронных сетей, а также выбор оптимального решения, которое можно было бы применить для распознавания CAPTCHA.

5. Реализация, анализ точности и времени работы автоматического решателя.

2. Обзор

2.1. Существующие аналоги

В предыдущем исследовании [11], относительно ведущих сервисов, предоставляющих услуги решения CAPTCHA, были получены следующие результаты: Capmonster.cloud, Azcaptcha не справляются с CAPTCHA компании “YML”, а ruCaptcha и Captcha.Guru тратят на задачу до 10 секунд. Помимо этого были изучены еще четыре ресурса: XEvil — бесплатное приложение, на основе оптического распознавания символов (OCR), а также платные инструменты BestCaptchaSolver, SolveCaptcha, AntiCaptcha. Подробные результаты сравнения приведенных выше сервисов вы можете изучить в таблице 1.

Сервис	Метод	Стоимость за 1000 шт.	Время на 1 cap	Решает ли “YML” CAPTCHA
XEvil	OCR	бесплатно	—	—
CapMonster	OCR	$0.3	—	—
Azcaptcha	OCR	$0.4	—	—
Captcha.Guru	OCR	$0.1	2-6 сек	+
BestCaptchaSolver	человек	$0.6	15-20 сек	+
SolveCaptcha	человек	$0.5-$1	7-10 сек	+
RuCaptcha	человек	$0.6	8-12 сек	+
AntiCaptcha	человек	$0.5	5-7 сек	+

Таблица 1: Сравнение сервисов, предоставляющих услуги решения CAPTCHA.

Из этого можно сделать два вывода. На данный момент в Интернете не существует достаточно умных бесплатных аналогов, а сайты, на которых можно решить CAPTCHA за деньги, тратят на это огромное количество времени. Все сервисы способные распознать имеющийся датасет, за единичным исключением, прибегают к помощи человека и не способны делать это самостоятельно.

Помимо этого, в работе Касацкого В.И. [11] рассматривается возможность решения CAPTCHA компании “YML” несколькими способами.

- Аналитический способ: “для этого были выполнены попытки поиска зависимостей между размером буквы, а также расстояния

между ними, но попытки не увенчались успехом”

- Средство оптического распознавания Tesseract: “Tesseract ... не смогло распознать текст ни на одной CAPTCHA”.

Единственным неопробованным способом распознать имеющийся датасет остается прибегнуть к машинному обучению. В первую очередь мною были изучены научные статьи [2, 8, 10]. В этих работах рассматриваются свёрточные нейронные сети(CNN), которые в настоящее время доминируют в области распознавания визуальных объектов. Статья [10] посвящена теоретическому обоснованию данного подхода, какие задачи может решать данная архитектура, а также каким образом можно добиться наилучших результатов при меньших усилиях. В другой работе [8] спроектировали и обучили глубокую нейронную сеть (ResNet), применяемую для распознавания объектов на фото. Идея, которая легла в ее основу, носит название “глубокое остаточное обучение” (deep residual learning). Особенность данной архитектуры заключается в добавлении остаточного блока, который прибавляет исходные входные данные обратно к выходной карте объектов, полученных путем прохождения входных данных через один или несколько сверточных слоев. Это позволяет решить проблему градиентной дисперсии и снижения точности в глубоких сетях, так как по мере увеличения количества слоев, контролируются как точность, так и скорость. Помимо этого была рассмотрена статья [2], в которой была разработана и обучена нейронная сеть “Xception” или же “Extreme inception”, работающая по совершенно иной схеме. Мы сначала применяем фильтры к каждой карте глубины, а затем, наконец, сжимает входное пространство, используя свертку 1×1, применяя ее по всей глубине. Таким образом, мы получаем архитектуру сети, которая практически идентична по точности ResNet, при этом существенно выигрывая по размерам, а значит по требуемым ресурсам как для обучения, так и для использования этой модели(таб. 2).

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	0.790	0.945

Таблица 2: Сравнение точности классификации ImageNet [2]

Однако результаты исследования [3] показывают, что для корректного и эффективного обучения нейронной сети необходимо примерно 2.3 миллиона размеченных данных. Сбор и ручная маркировка такого количества реальных CAPTCHA потребует интенсивного участия человека и повлечет за собой значительные затраты. Из чего следует вывод, что CNN отлично подходит для решения задачи распознавания CAPTCHA, однако необходимо некоторым образом упростить процесс получения достаточного количество образцов для ее обучения.

В статьях [1, 5], проведены исследования направленные на изучение возможности применения генеративно-состязательных моделей (подробнее об этом в следующем пункте) для решения CAPTCHA. Цель данных работ заключается в минимизации количества обработанных данных, требуемых для обучения решателя, а как следствие сведения к минимуму участия человека. Идея состоит в том, чтобы с помощью синтетического генератора заполнить обучающие данные неограниченным количеством синтетических CAPTCHA (аналогичных настоящим). А такого результата практически невозможно достичь, используя исключительно размеченные человеком данных.

Подводя итог, становится очевидным, что необходимо реализовать собственный решатель CAPTCHA, так так существующие аналоги не решают конкретно нашу задачу. А в качестве решателя использовать связку двух компонентов:

- GAN модель для генерации достаточного количества синтетических обучающих данных;
- Сверточная нейронная сеть, которая будет непосредственно ре-

шать CAPTCHA компании “YML”.

2.2. Generative Adversarial Network Models.

Базовая идея архитектуры GAN [4], которая лежит в основе решения, состоит из двух моделей: генеративной сети, используемой для создания синтетических примеров, и дискриминационной сети, отличающей синтезированные примеры от реальных. Здесь используется обратное распространение ошибки для обучения обеих сетей, так в течение итераций обучения генератор производит более качественные синтетические образцы, а дискриминатор становится более опытным в определении синтетических фото. GAN архитектура показала впечатляющие результаты в задачах обработки изображений [6] и естественного языка. (рис. 1)

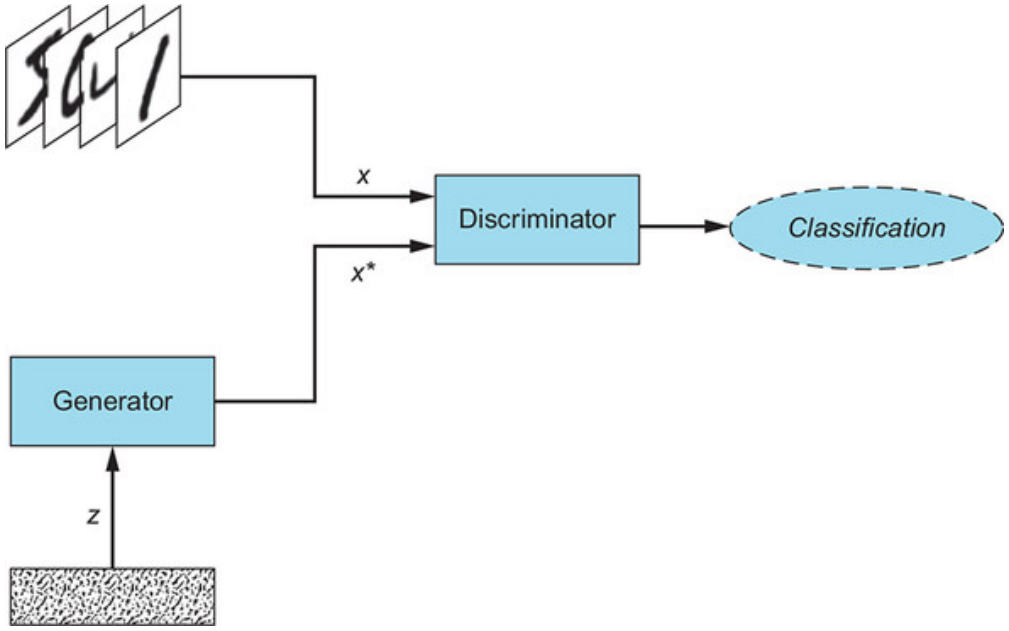

Рис. 1: GAN архитектура

Однако стоит отметить, что хоть мы и можем контролировать область примеров, которые наша GAN научилась имитировать, мы не можем указать какие-либо характеристики выборок этих данных. Например, DCGAN, обученная на датасете MNIST, может синтезировать реалистично выглядящие рукописные цифры, но мы никак не можем

контролировать, будет ли она производить, скажем, цифру 7 или цифру 9 в любой момент времени.

CGAN — одно из первых усовершенствований GAN, которое сделало возможным целевое генерирование данных, и, возможно, самое влиятельное. Условная GAN [9], представляет собой генеративно-состязательную сеть, генератор и дискриминатор которой настраиваются во время обучения с использованием некоторой дополнительной информации. Эта вспомогательная информация теоретически может быть чем угодно, например, меткой класса, набором тегов или даже письменным описанием. Мы будем использовать расшифровки CAPTCHA в качестве условной информации. (рис. 2)

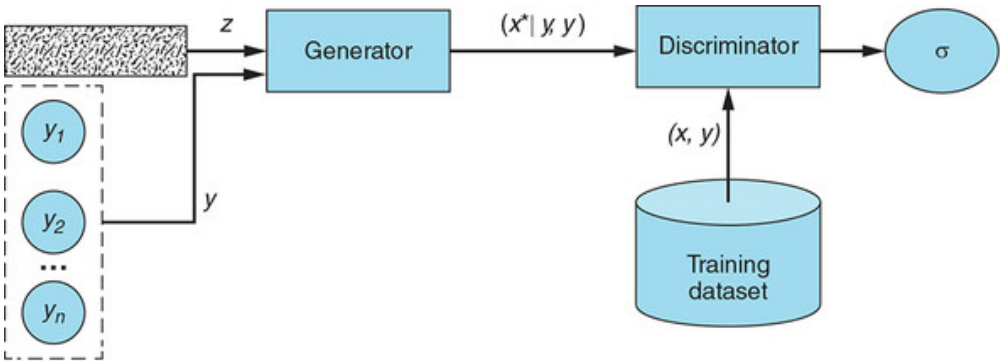

Рис. 2: CGAN архитектура

Во время обучения CGAN генератор учится создавать реалистичные примеры для каждой метки в обучающем наборе данных, а дискриминатор учится отличать поддельные пары пример-метка от реальных пар пример-метка. Соответственно, чтобы обмануть дискриминатор, генератору CGAN недостаточно выдавать реалистично выглядящие данные. Генерируемые примеры также должны соответствовать своим меткам. После того как генератор полностью обучен, это позволяет нам указать, что мы хотим, чтобы CGAN синтезировал, передав ему желаемую метку.

Изображения 1 и 2 были взяты из [7]

2.3. Используемые технологии

1. **Python=3.9.12** — интерпретируемый, интерактивный, объектно-ориентированный язык программирования. Он включает в себя модули, динамические типы данных очень высокого уровня и классы.

2. **Pillow=9.2.0 и OpenCV=4.6.0.66** — библиотеки, которые позволяют просто и удобно работать с обработкой изображений в языке Python. Последняя из которых применяет алгоритмы компьютерного зрения для распознавания и описания объектов на фотографии.

3. **TensorFlow=2.6.0** — открытая библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов. Важными преимуществами являются качественная документация, параллелизм, множество обученных моделей, а также большое комьюнити.

3. Реализация

3.1. Исследование набора данных.

Перед началом работы над решателем CAPTCHA необходимо исследовать исходный датасет, в том числе очистить его от неправильных или случайно попавших туда данных. Для того чтобы понять, откуда могут возникать ошибочные файлы, разберем детальнее процесс формирования датасета. После обработки некоторого действия пользователя на сайте Cargotime.ru генерируется запрос на получение информации с сайта компании "YML" и одновременно записывается в файл с расшифровкой. Затем человеку предлагается собственноручно решить CAPTCHA. В случае правильного ответа расшифровка перезаписывается, и пользователь получает интересующую его информацию. В результате такого сбора датасета часть файлов содержит некорректные сведения, и из 180 000 изначальных записей мы имеем лишь 18020 пар изображение + расшифровка (рис. 3).

Как вы могли бы заметить, набор CAPTCHA обладает огромной вариативностью: перечеркнутые линии, сильный шум, наложенный поверх узор, разная цветовая гамма, переменное расстояние между символами, а также разная ширина каждой отдельной буквы — все это сильно затрудняет дальнейшую обработку изображений.

Исходя из вышесказанного, принято решение разделить имеющиеся данные на 30 кластеров, так как по результатам эксперимента именно такое число является оптимальным. Для кластеризации было использовано решение, полученное в результате работы над темой предыдущим студентом [11]. В его основе лежат модель VGG16 совместно с алгоритмом кластеризации KMeans, преимуществом которых является скорость и простота.

Финальный шаг, который необходимо предпринять для успешного обучения синтетического генератора — это вручную вырезать для всех кластеров по 16 образцов каждого символа, хотя для более эффективного обучения может потребоваться больше. Так как наш датасет состоит

из 34 символов ⇒ общее количество возможных выходов возрастает до 34^4. А благодаря нашему преобразованию это число становится равным количеству различных символов кластера, т.е 34. Иначе обучить CGAN модель на таком количестве исходных данных просто невозможно.

Рис. 3: CAPTCHA компании “YML”

3.2. Реализация CGAN архитектуры.

Рассмотрим детальнее процесс обучения нашей модели. На рисунке 4 показаны три основных этапа построения CGAN. Каждый из этапов описан следующим образом.

- **Генератор CAPTCHA.** Чтобы немного формализовать определение, назовем условие, а в нашем случае комбинацию текстовых символов, которые мы хотим преобразовать в изображение — y. А необработанный вход (шум), представленный в виде вектора случайных чисел, которые Генератор использует в качестве отправной точки для синтеза поддельных примеров — z. Тогда их комбинация и дает нам условный G(z, y) = x*|y (читается как “x* с учетом или при условии y”). Цель этого поддельного примера — выглядеть (в глазах Дискриминатора) как можно ближе к реальному примеру для данного ярлыка. В нашем случае наилучшего результата удалось добиться при помощи свёрточной нейронной сети, состоящей из трех свёрточных слоев и функцией активации выходного слоя — Sigmoid. Стоит отметить, что генератор обнов-

ляет свои параметры, минимизируя следующую функцию потерь:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \qquad (1)$$

- **Дискриминатор CAPTCHA.** Дискриминатор получает на вход как настоящие примеры с метками (x, y), так и поддельные примеры с меткой, которая использовалась для их генерации (x*|y, y). На синтетических примерах он учится распознавать фальшивые изображения. Метки ему необходимы, чтобы научиться не только различать сгенерированные данные, но и проверять действительно ли нарисовано то, что мы изначально хотели. Цель дискриминатора заключается в приобретении навыка отбрасывания всех поддельных и не соответствующих им метке примеров. Для этого будем использовать нейронную сеть, состоящую из 3 сверточных слоев с той же LeakyRelu и одного полносвязного слоя, применив ту же функцию активации для BackPropogation(1), что и для Генератора. В качестве ответа Дискриминатор выдает одно число — вероятность того, что входные данные являются реальной совпадающей парой, которая вычисляется при помощи сигмоидальной функции активации.

- **Обучение.** Мы используем оптимизатор Adam со скоростью обучения 0,0003 для обучения нашего синтезатора CAPTCHA. Общая цель обучения соответствует общему подходу GAN [9], однако с некоторыми небольшими изменениями:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x*|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \qquad (2)$$

где генератор G, пытается минимизировать разницу между сгенерированными изображение+метка и реальными, в то время как дискриминатор D, пытается максимизировать ее. Необходимо помнить, что во время обновления параметров одного из них нужно фиксировать параметры другого, иначе мы не получим желаемо-

го результата.

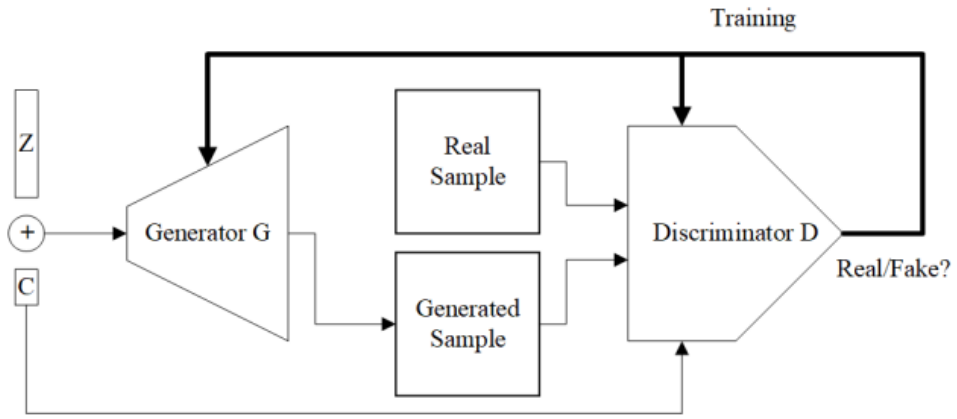

Рис. 4: Процесс обучения CGAN

Пример. Рассмотрим в качестве примера один из кластеров. Как вы можете заметить ниже(рис. 5 и 6), данные CAPTCHA практически не различимы. Это позволяет нам воссоздать неограниченное количество экземпляров для дальнейшего обучения, тем самым сократив огромное количество времени на сбор данных.

Рис. 5: Оригинальная CAPTCHA компании “YML”

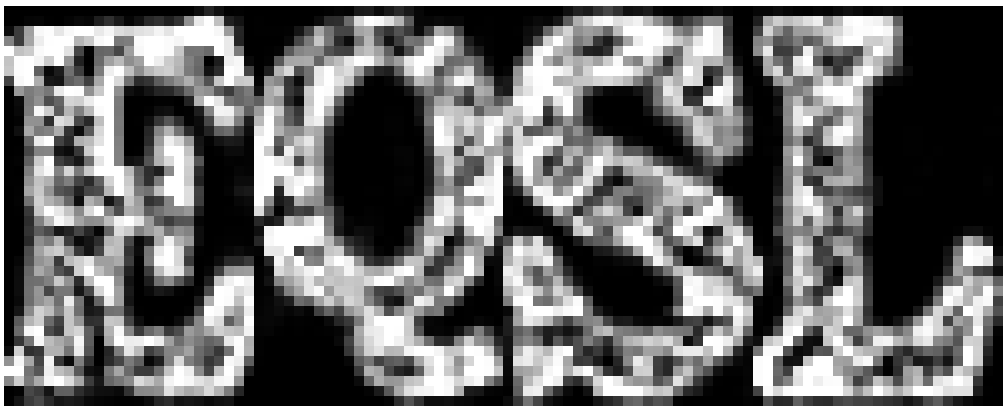

Рис. 6: CAPTCHA сгенерированная нейронной сетью

3.3. Создание решателя CAPTCHA компании “YML”.

Давайте взглянем немного подробнее на реализацию решателя CAPTCHA. Условно данный процесс можно разделить на три этапа.

1. **Подготовка обучающих данных.** В первую очередь стоит отметить, что сгенерированных образцов, обладающих схожим внешним видом, однако имеющих как статическую ширину символов, так и одинаковое расстояние между друг другом, недостаточно для обучения решателя CAPTCHA. Как было сказано в пункте 3.1, оригинальные CAPTCHA различаются не только узором и шрифтом, но и шириной и расстоянием между символами. В связи с чем был применен алгоритм, который позволяет при помощи биномиального распределения, выбрать n чисел суммы m. Применяя данный подход к полученным ранее синтетическим CAPTCHA — мы получаем изображения размером $40 \times 96 \times 1$, содержащие нужное нам количество стилистически неотличимых от оригинала символов, при этом находящихся на случайном расстоянии друг от друга.

2. **Выбор способа хранения.** После того как мы научились создавать достаточно неотличимые от оригинала CAPTCHA, необходимо разобраться с тем как хранить только что полученные

изображения. Существует два ответа на поставленный вопрос: во время обучения непрерывно генерировать новые образцы или же перед началом сгенерировать достаточное количество поддельных CAPTCHA. Преимуществом первого подхода является использование совершенно разных образцов для каждой эпохи обучения, тем самым понижая возможность переобучения. Однако иной подход значительно выигрывает в скорости, так как все ресурсы компьютера уходят исключительно на обучение. В работе было решено применять именно второй способ из-за огромного выигрыша в скорости. При этом использовано:

- Для обучения — 5000 оригинальных изображений, полученных путем клонирования нескольких сотен исходных экземпляров, совместно с 40000 синтетическими CAPTCHA
- Для валидации — 2000 оригинальных и 10000 сгенерированных изображений.

3. **Поиск оптимальной архитектуры.** Последний шаг, который необходимо совершить — это выбрать архитектуру свёрточной нейронной сети либо из уже существующих, либо создать собственную. Сначала были протестированы: AlexNet, VGG19, DenseNet, Unet. Однако на данных компании "YML" вышеперечисленные модели за различное количество эпох не смогли достичь точности выше 10%, несмотря на то что на синтетических CAPTCHA показатели спустя нескольких первых эпох стремились к 100%. Далее была рассмотрена модель XCeption (рис. 7), которая справилась с задачей намного лучше (см. пункт 4). Данные сначала проходят через входной поток, затем через средний поток (повторяясь в нем восемь раз) и, наконец, через выходной поток.

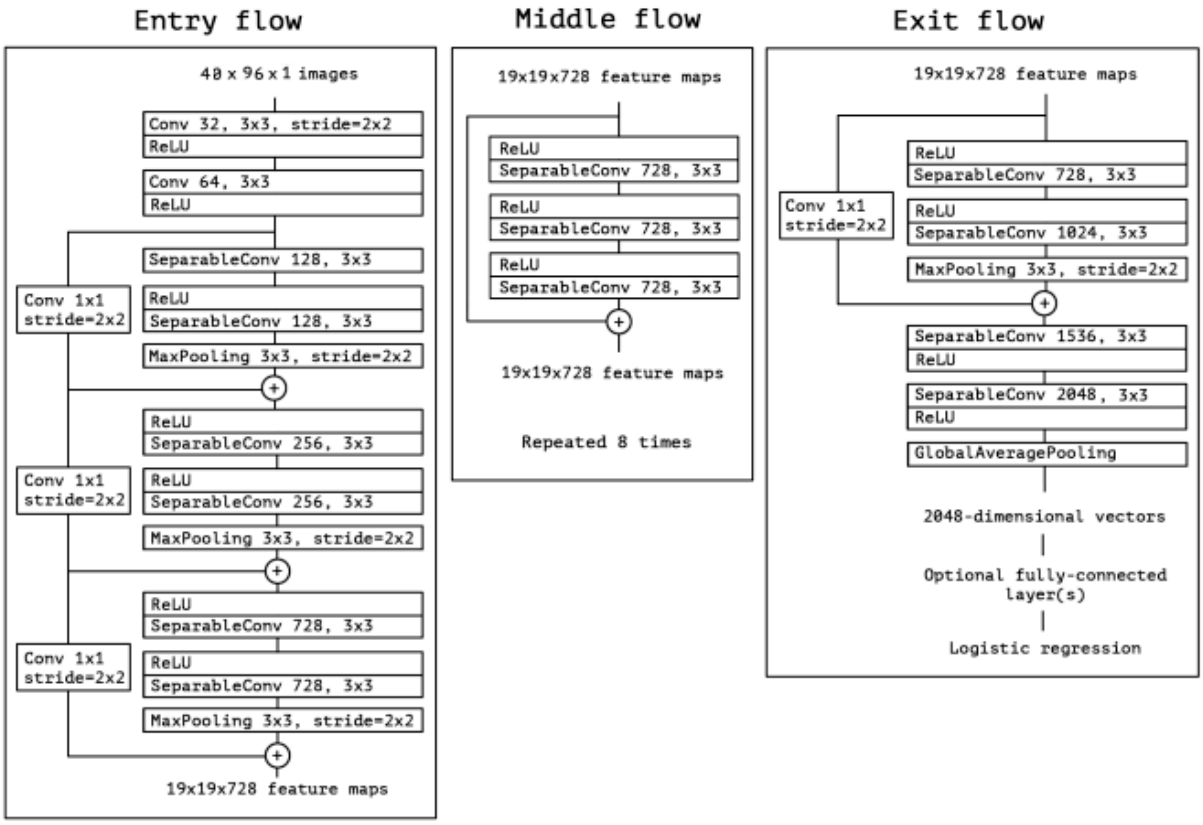

Рис. 7: Xception архитектура [2]

Однако несмотря на то, что в пункте 2.1 было отмечено преимущество XCeption перед ResNet, наилучших результатов удалось добиться именно при помощи модели ResNet-34. Рассмотрим процесс обучения более детально. Сеть состоит из сверточных слоев и остаточных блоков, идущими друг за другом, но у каждого такого блока есть свой “обходной путь”, который напрямую связывает вход с выходом(рис. 8).

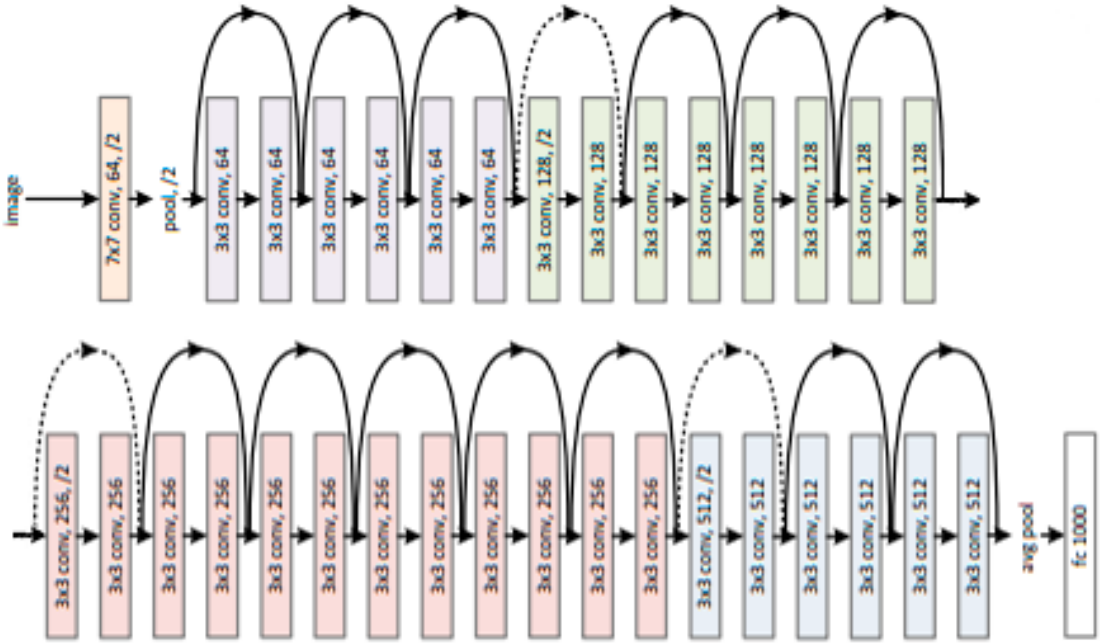

Рис. 8: ResNet-34 архитектура [8]

То есть слои, составляющие функцию, должны аппроксимировать не всю функцию целиком, а лишь ее остаток. В таком случае нам достаточно обучить сеть воспроизводить не весь сигнал, а лишь отличия входных значений для получения требуемой функции. В ходе исследования было получено, что наилучшие показатели точности на данных компании “YML” (см. пункт 4) достигаются за 30 эпох при использовании оптимизатора Adam с шагом обучения 10^{-6} и функцией активации Categorical Cross-Entropy:

$$H(x) = \sum_{i=1}^{N} y_i \log(\hat{y}_i), y_i = p(x_i) \tag{3}$$

https://github.com/Alexander-Zadorozhnyy/CGAN$_Y$$ML_C$$apthcaSolver$ - GitHub репозиторий с реализованным приложением.

4. Тестирование

В рамках данной работы был произведен анализ результатов реализованных нейронных сетей на датасете компании “YML”.

- **Условия эксперимента.** Характеристики машины на которой проводилось тестирование: процессор Intel(R) Core(TM) i5-10300H, видеокарта NVIDIA GeForce GTX 1650 Ti, размер оперативной памяти 8GB.

- **Метрики.** В рамках данного эксперимента замерялись точность распознавания 100 сгенерированных (G.Accuracy %) и 100 оригинальных (O.Accuracy %) образцов, а также скорость распознавания 5000 произвольных CAPTCHA (Speed sec).

- **Результаты.** Все результаты, которых удалось достичь за время работы над проектом, вы можете изучить в приведенной ниже таблице (таб. 3)

Таблица 3: Наивысшие показатели решателя CAPTCHA в зависимости от выбранной архитектуры.

Architecture	G.Accuracy %	O.Accuracy %	Speed sec
AlexNet	99.22	3.11	320.43 ± 0.30
VGG19	98.74	5.28	301.15 ± 0.10
DenseNet	99.33	7.14	334.27 ± 0.04
Unet	97.81	9.34	320.84 ± 0.06
Xception	98.13	58.01	430.23 ± 0.20
ResNet-34	100.00	63.10	419.10 ± 0.20

Заключение

Целью работы являлось создание автоматического решателя, который упростит процедуру получения данных о контейнере на сайте Cargotime, убрав необходимость вручную вводить CAPTCHA. Поставленная цель была достигнута не полностью. Для реализации проекта были решены следующие задачи:

- Исследован и обработан набор данных CAPTCHA;
- Рассмотрены существующие в открытом доступе сервисы, предоставляющие возможность решения CAPTCHA;
- Проанализированы созданные ранее модели, которые можно было бы адаптировать для конкретно нашего варианта CAPTCHA;
- Реализованы синтаксический генератор и дискриминатор для одного из кластеров;
- Обучена свёрточная нейронная сеть, решающая один из кластеров “YML” CAPTCHA.

Направления будущей работы. Требуется повысить точность реализованного в данной работе решателя. А также в дальнейшем планируется применить полученные результаты ко всем оставшимся кластерам, добавив тем самым возможность решать CAPTCHA компании “YML” любого вида.

Список литературы

[1] Alec Radford Luke Metz Soumith Chintala. Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks // Proc. ICLR. — Vol. 1.

[2] Chollet François. Xception: Deep Learning with Depthwise Separable Convolutions // Proc. Google, Inc. — Vol. 1.

[3] George D. Lehrach W. Kansky K. Lĺćzaro-Gredilla M. Laan C. Marthi B. Lou X. Meng Z. Liu Y., Wang H. A generative vision model that trains with high data efficiency and breaks text-based captchas. // Proc. Science. — Vol. 1.

[4] Goodfellow I. J. Pougetabadie J. Mirza M.-Xu B. Wardefarley D. Ozair S. Courville A., Bengio Y. Generative adversarial networks. Advances in Neural Information Processing Systems 3 // Proc. Université de Montréal. — Vol. 1.

[5] Guixin Ye Zhanyong Tang Dingyi Fang Zhanxing Zhu Yansong Feng Pengfei Xu Xiaojiang Chen Zheng Wang. Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. // Proc. ACM SIGSAC Conference. — Vol. 1.

[6] Isola P. Zhu J.-Y. Zhou T., Efros A. A. Image-to-image translation with conditional adversarial networks // Proc. CVPR 2017. — Vol. 1.

[7] Jakub Langr Vladimir Bok. GANs in Action: Deep learning with Generative Adversarial Networks. — US, 2019.

[8] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. Deep Residual Learning for Image Recognition // Proc. ILSVRC COCO 2015. — Vol. 1.

[9] Mehdi Mirza Simon Osindero. Conditional Generative Adversarial Nets.

[10] St´ephane Mallat Ecole Normale Sup´erieure. Understanding Deep Convolutional Networks // Proc. Centre national de la recherche scientifique. — Vol. 1.

[11] Касацкий Всеслав Игоревич. Автоматизация распознавания CAPTCHA в задаче парсинга данных о статусе грузовых контейнеров // Proc. Отчёт по учебной практике. — Vol. 1.