УДК 004.8

РАСПОЗНАВАНИЕ CAPTCHA НА ОСНОВЕ ГЕНЕРАТИВНО-СОСТЯЗАТЕЛЬНОЙ СЕТИ

Парфенов Анатолий Сергеевич[1], Сычёв Пётр Павлович[2]

[1]Студент;
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: steamfenov@ya.ru.

[2]Доцент;
ГБОУ ВО МО «Университет «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: sychov@dubna.ru.

Распознавание CAPTCHA, безусловно, не является новой темой исследования. За последнее десятилетие исследователи продемонстрировали различные способы автоматического распознавания текстовых CAPTCHA. Однако в таких способах настройка распознавания требует большого участия экспертов и несет трудоемкий процесс сбора и маркировки данных. В этой статье представлен общий, малозатратный, но эффективный подход к автоматическому решению текстовых CAPTCHA на основе глубокого обучения. Данный подход основан на архитектуре генеративно-состязательной сети, что значительно сократит количество реальных необходимых CAPTCHA.

Ключевые слова: CAPTCHA, распознавание, искусственные нейронные сети, глубокое обучение, генеративно-состязательная сеть.

CAPTCHA RECOGNITION BASED ON A GENERATIVE-ADVERSARIAL NETWORK

Parfenov Anatoliy[1], Sychov Peter[2]

[1]Student;
Dubna State University,
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: steamfenov@ya.ru.

[2]Associate professor;
Dubna State University,
Institute of the system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: sychov@dubna.ru.

CAPTCHA recognition is certainly not a new research topic. Over the past decade, researchers have demonstrated various ways to automatically recognize text-based CAPTCHAs. However, in such methods, the recognition setup requires a large participation of experts and carries a laborious process of collecting and marking data. This article presents a general, low-cost, but effective approach to automatically solving text-based CAPTCHAs based on deep learning. This approach is based on the architecture of a generative-competitive network, which will significantly reduce the number of real required CAPTCHAs.

Keywords: CAPTCHA, recognition, artificial neural networks, deep learning, transfer learning, generative adversarial network.

Введение

Текстовые *CAPTCHA* (*Completely Automated Public Turing Test to Tell Computers and Humans Apart)* широко используются для того, чтобы отличить человека от автоматизированных компьютерных программ [1]. Хотя и были предложены многочисленные альтернативы текстовым *CAPTCHA*, многие веб-сайты и приложения, такие как *Google, Microsoft, eBay*, Википедия, Яндекс, по-прежнему используют текстовые *CAPTCHA* в качестве механизма безопасности и аутентификации.

В последнее время было предложено достаточно много общих распознаваний *CAPTCHA*. Однако эти методы требуют очень большого числа примеров с метками и нацелены только на текстовые *CAPTCHA* с относительно простыми функциями безопасности. Тут и приходит на помощь алгоритм генеративно-состязательной сети, которая продемонстрировала впечатляющую производительность в задачах трансляции изображений [2, 3]. Данный алгоритм не только значительно сокращает человеческое участие и усилия, необходимые для построения хорошего решателя *CAPTCHA*, но и дает значительно лучшую производительность при решении широкого спектра современных схем *CAPTCHA*.

Способы защиты текстовой CAPTCHA

В данной статье будут рассмотрены шесть широко используемых функций безопасности (рис. 1).

Функции анти-сегментации затрудняют программе разделить символы друг от друга:

- перечеркивающие линии;
- накладывание символов друг на друга;
- сложный задний фон.

Функции анти-распознавания увеличивает сложность распознавания символов:

- полые и заполненные символы;
- повернутые, искаженные и размытые символы;
- разные размеры, шрифты и цвета символов.

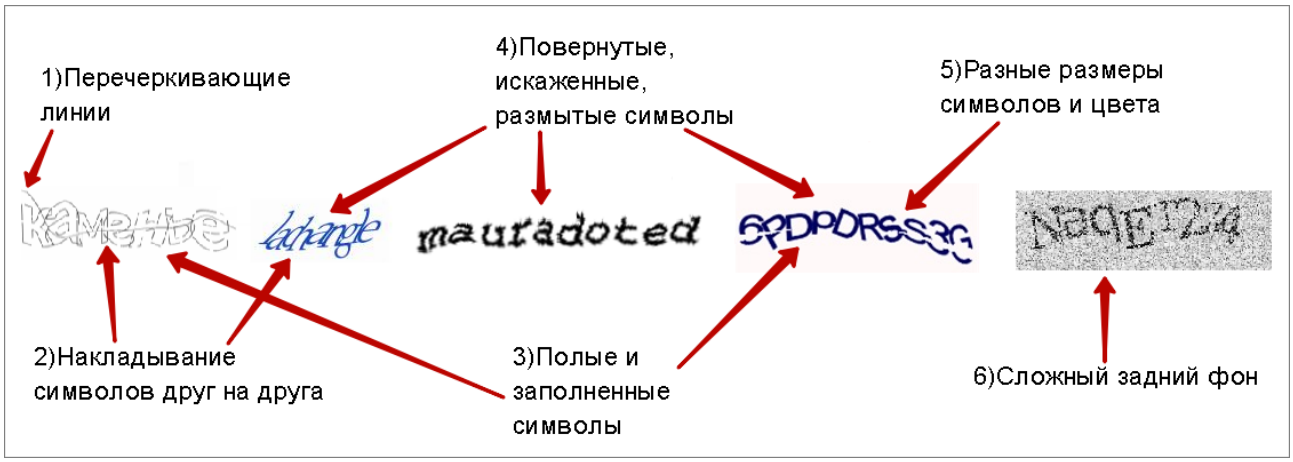

Рис. 1. Примеры с различными защитами текстовой CAPTCHA

Сверточные нейронные сети

Для решения задачи распознавания изображений применяются сверточные нейронные сети, сети прямого распространения без обратных связей. Сверточная нейронная сеть представляет собой последовательность слоев, где каждый слой преобразует один объем входных данных в другой с помощью дифференцируемой функции. Для построения сверточных нейросетей используют следующие типы слоев:

- сверточный слой (*convolutional layer*);

- слой субдискретизации или пулинг (*subsampling or pooling layer*);
- полносвязный слой (*fully-connected layer*).

А также входной и выходной слои. На вход подаются изображения в многомерном виде: ширина, высота и глубина, где глубина может быть представлена как три цветовых канала *RGB* (англ. red, green, blue – красный, зеленый, синий), так и один цветовой канал в оттенках серого. На выходном слое нейронная сеть выдает информацию в виде числа или вектора в зависимости от количества нейронов, которое непосредственно определяется типом решаемой задачи.

Сверточный слой – это основной блок сверточной нейронной сети. Сверточный слой выполняет математическую операцию свертки. Свертка – это особый вид линейной операции над двумя функциями вещественного аргумента. Для выполнения свертки необходимо иметь значения следующих параметров: размер ядра (фильтра) свертки (*kernel size*) и шаг (*stride*). А также необходимо определить необходимость применения нулевого паддинга (*zero padding*).

Размер ядра свертки определяет то, какие особенности (*features*) будут детектироваться: маленький размер ядра определяет поиск маленьких локальных особенностей, большой размер – большие *features* или целые объекты. Шаг контролирует то, каким образом фильтр «свертывается» вокруг определенного набора данных: чем выше значение шага, тем сильнее «свертывается» (уменьшается) набор данных.

На рис. 2 представлен пример двумерной операции свертки, выполняемый сверточным слоем.

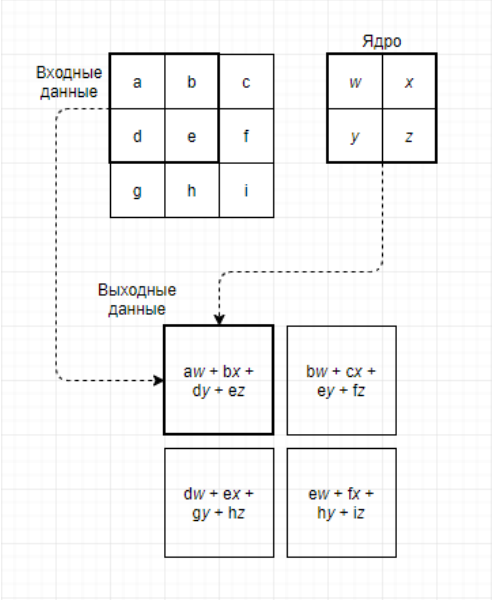

Рис. 2. Пример двумерной операции свертки

Слой субдискретизации, выполняющий функцию пулинга, позволяет уменьшать объемы данных путем замены выходов сети в некоторой точке сводной статистикой близлежащих выходов. Существует два наиболее популярных в использовании алгоритма пулинга:

- *max*-пулинг, возвращающий максимальный выход в прямоугольной области;
- *average*-пулинг, усредняющий значения прямоугольной области в выходное значение.

На вход слоя пулинга поступают данные в четырехмерном формате: ширина (*W*), высота (*H*), глубина (*depth*), количество элементов (*N*). Операция пулинга имеет ядро (*kernel*) заданного размер, а также имеет шаг (*stride*), с которым ядро пулинга применяется к входным данным.

На рис. 3 изображен пример операции *max*-пулинга с ядром размера 2 на 2 и шагом 2.

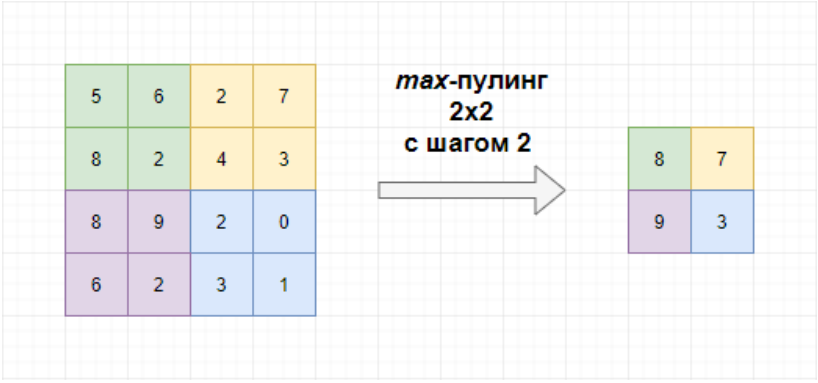

Рис. 3. Пример операции max-пулинга ядром 2x2 и шагом 2

Полносвязный слой – слой, состоящий их простых нейронов, в котором каждый нейрон связан с каждым нейроном следующего слоя. Для сверточных нейросетей характерно использование нескольких полносвязных слоев в конце нейронной сети, а последний полносвязный слой является «выходным», определяющем результат работы.

В качестве функции активации для нейронов сверточных нейронных сетей применяется блок линейной ректификации *ReLU* (*rectified linear unit*) (1):

$$f(x) = max(0, x). \quad (1)$$

Более подробное описание архитектур сверточных сетей можно найти в [4].

Алгоритмы обучения нейронных сетей

В связи с тем, что обучение нейронных сетей связано с вычислениями, требующими большие вычислительные мощности, одним из наиболее важных критериев является скорость обучения нейронных сетей.

Существует множество алгоритмов обучения нейронных сетей, большинство их, которых включают оптимизацию, то есть нахождения минимума или максимума функции *f(x)* при итеративном изменении *x*. При обучении нейросетей рассматривается задача нахождения минимума. Функция, для которой идет поиск минимума, называется функцией потерь (*loss function*) или целевой функцией. Для нахождения минимума чаще всего применяется метод градиентного спуска (*gradient descent*), а большинство современных оптимизаторов, применяемых для обучения нейронных сетей, являются производными этого метода. У классического метода градиентного спуска есть модификация, названная стохастическим градиентным спуском *SGD* (*stochastic gradient descent)*.

Существует большое количество различных оптимизаторов, подробнее можно почитать в [6], в данной работе применялся *Adam* (*adaptive moment estimation*) [5].

Генеративно-состязательная сеть

Генеративно-состязательная сеть – *GAN* (*generative adversarial network*) [2], состоит из двух моделей: генеративной сети *G* для создания синтетических примеров и дискриминирующей сети *D*, чтобы отличить синтезированные примеры от реальных. Процедура обучения для *G* - максимизировать вероятность того, что *D* допустит ошибку. Эта структура соответствует минмаксной игре для двух игроков.

Состязательная модель наиболее проста в применении, когда обе модели представляют собой многослойные персептроны. Чтобы узнать распределение генератора ρg по данным *x*, нужно определить априорные переменные входного шума $\rho z(z)$, а затем составить сопоставление в пространстве данных как $G(z; \theta g)$, где *G* – это дифференцируемая функция, представленная многослойным персептроном с параметром θg. Также нужно определить второй многослойный персептрон $D(x; \theta d)$, который выводит один скаляр. $D(x)$ представляет вероятность того, что *x* пришел из данных, а не из ρg. Также нужно обучить *D*, чтобы максимизировать вероятность присвоения

правильной метки обучающим примерам, и образцам из *G*. Также нужно одновременно обучать *G*, чтобы минимизировать (2).

$$log\big(1 - D(G(z))\big) \tag{2}$$

Другими словами, *D* и *G* играют в следующую минмаксную игру для двух игроков со значением функции *V*(*G*, *D*) (3).

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim\rho_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim\rho_x(x)}[\log(1 - D(G(z)))] \tag{3}$$

На практике уравнение (3) может не обеспечивать достаточного градиента, чтобы *G* мог хорошо обучаться. В начале обучения, когда *G* еще плох, *D* может отделять образцы от данных обучения с высокой достоверностью, потому что они явно отличаются друг от друга. В этом случае (2) насыщается. Вместо того, чтобы обучать *G* минимизировать (2), можно обучить *G* максимизировать (4).

$$\log\big(D(G(z))\big) \tag{4}$$

Эта целевая функция приводит к той же фиксированной точке динамики *G* и *D*, но обеспечивает гораздо более сильные градиенты в начале обучения.

На рис. 4 изображено как генеративные противоборствующие сети обучаются путем одновременного обновления дискриминационного распределения (*D*, синяя пунктирная линия), чтобы *D* отличал выборки от распределения, генерирующего данные (черная пунктирная линия), от образцов генеративного распределения *ρg* (*G*, зеленая сплошная линия). Нижняя горизонтальная линия – это область, из которой выбирается *z*, в этом данном случае равномерно. Горизонтальная линия выше является частью областью *x*. Стрелки вверх показывают, как сопоставляется *x* = *G*(*z*), накладывая неравномерное распределение *ρg* на преобразованные выборки. *G* сокращается в регионах с высокой плотностью и расширяется в регионах с низкой плотностью *ρg*. Состязательная пара (рис. 4, a), *ρg* похож на *ρdata*, а *D* – частично точный классификатор. Во внутреннем цикле алгоритм *D* обучен различать выборки из данных, сходящихся к (5) (рис. 4, b).

$$D^*(x) = \frac{\rho_{data}(x)}{\rho_{data}(x) + \rho_g(x)}. \tag{5}$$

После обновления *G*, градиент *D* направил *G*(*z*) в область, в которой с большей вероятностью будут классифицированы как данные обучения (рис. 4, c). После нескольких шагов обучения, *G* и *D* достигнут точки, в которой оба не могут улучшиться, потому что *ρg* = *ρdata* (рис. 4, d).

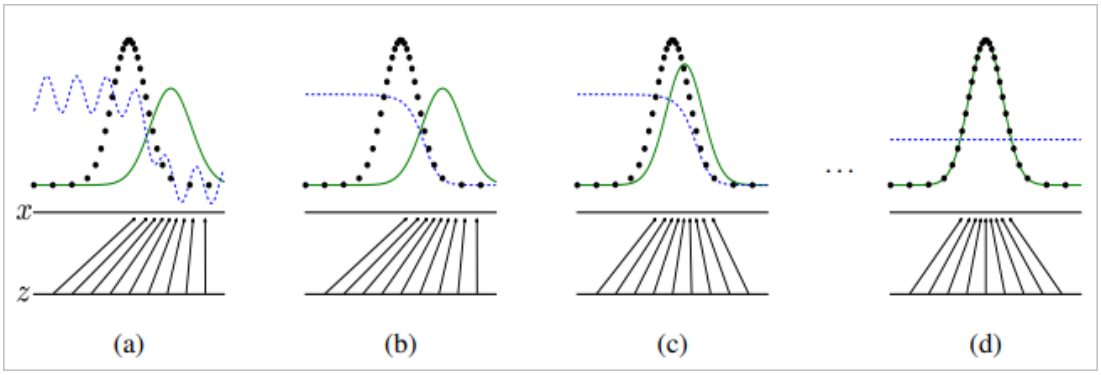

Рис. 4. Генеративная состязательная сеть обучаются путем одновременного обновления

Описание подхода распознавания CAPTCHA

На рис. 5 отображена общая схема данной работы. Сначала используется небольшой набор реальных *CAPTCHA* целевой схемы, чтобы научиться синтезировать *CAPTCHA*. Затем синтезатор используется для автоматической генерации синтетических *CAPTCHA* с элементами защиты и без, далее эти примеры нужны для обучения модели предварительной обработки, которая будет удалять функций безопасности, например, шумный фон и перечеркивающие линии, из входного изображения

CAPTCHA. В тоже время, синтетические *CAPTCHA* (без функций безопасности) используются для обучения решателя [8].

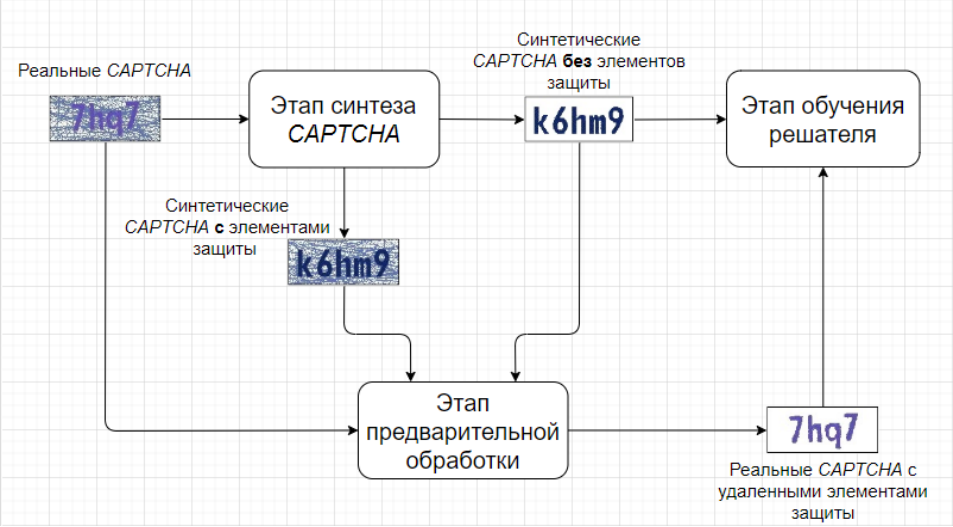

Рис. 5. Модель распознавания CAPTCHA на основе GAN

Этап синтеза *CAPTCHA*. Первым шагом является создание *CAPTCHA*, которые визуально похожи на целевые *CAPTCHA* (рис. 6). Синтезатор на основе *GAN* состоит из двух частей: генератора *CAPTCHA*, который пытается создать *CAPTCHA*, максимально похожие на целевые *CAPTCHA*, и дискриминатора, который пытается идентифицировать синтетические и реальные *CAPTCHA*. Этот процесс завершается, когда дискриминатор не может идентифицировать большую часть синтетических *CAPTCHA*. После окончания обучения можно использовать обученный синтезатор *CAPTCHA*, чтобы автоматически генерировать неограниченное количество *CAPTCHA*, для которых известны ответы.

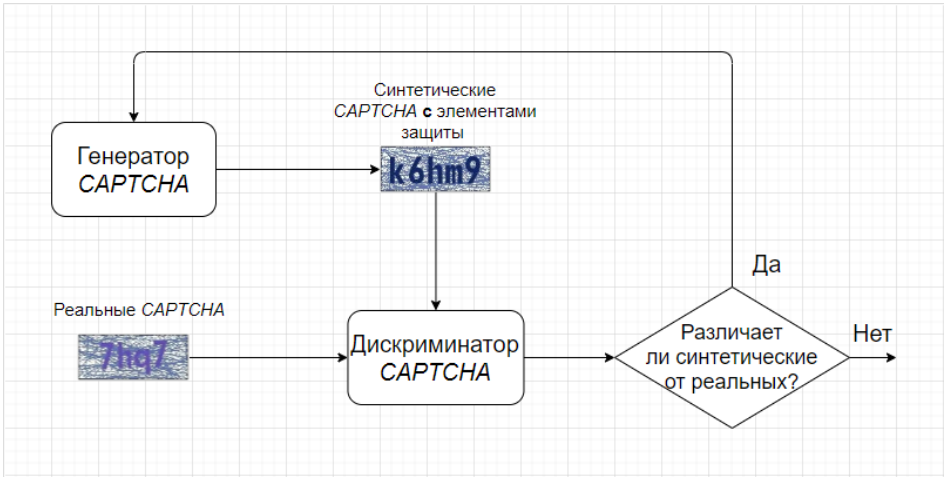

Рис. 6. Модель синтезатора CAPTCHA на основе GAN

Этап предварительной обработки. Перед тем как передать пример *CAPTCHA* решателю, генератор работает на уровне пикселей, и пытается изменить некоторые пиксели входного *CAPTCHA*-изображения, например, для удаления фонового шума. В отличие от этого, дискриминатор пытается отличить предварительно обработанные *CAPTCHA* от чистых, которые производятся синтезатором *CAPTCHA*. (рис. 7). Модель предварительной обработки основана определенной *GAN* называемой *Pix2Pix* [3]. Она обучается на синтетических *CAPTCHA* которые не имеют защитных элементов. Затем обученную модель можно использовать для очистки *CAPTCHA* от защитных функций. Целью обучения является изучение генератором функций безопасности для дальнейшего удаления их, а также стандартизации стиля шрифта. Обучение заканчивается, когда дискриминатор не может определить более 5% сгенерированных *CAPTCHA* из чистых аналогов.

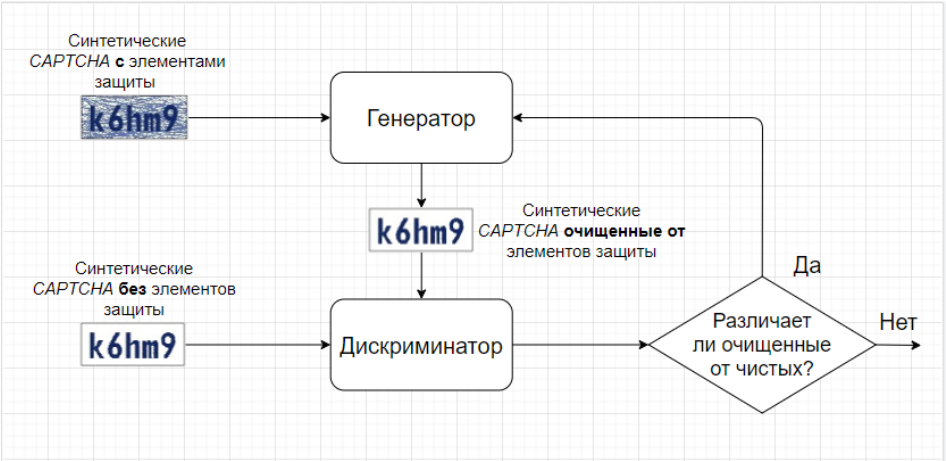

Рис. 7. Процесс обучения модели предварительной обработки на основе GAN

Этап обучения решателя. При наличии синтезатора *CAPTCHA* и модели предварительной обработки можно генерировать большое количество синтетических *CAPTCHA* вместе с их ответами и использовать этот набор данных для обучения решателя для целевой схемы *CAPTCHA*. Решатель основан на классической *CNN* под названием *LeNet*-5 [9]. Обученный решатель берет предварительно обработанное изображение и выводит соответствующие символы. Изначально предполагалось распознавать по одному символу, но введя несколько дополнительных слоев (2 сверточных и 3 пулинговых), получилось расширить способность для распознавания нескольких символов. Структура данного решателя, имеет пять сверточных слоев, пять пулинговых слоев, за которыми следуют два полносвязанных слоя (рис. 8). Выходной слой данного решателя состоит из нескольких нейронов, по одному нейрону для каждого символа *CAPTCHA*. Например, если схема *CAPTCHA* использует *n* символов, выходной слой будет состоять из *n* нейронов, где каждый нейрон соответствует символу. Решатель обучается для каждой схемы *CAPTCHA* отдельно. Если количество символов в *CAPTCHA* не является фиксированным, то базовый решатель обучается для каждого возможного количества символов.

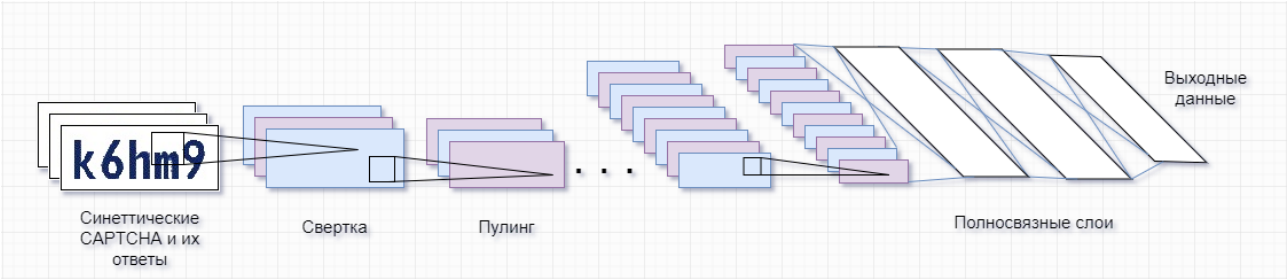

Рис. 8. Модель решателя CAPTCHA на основе CNN

Подготовка данных

Используется два набора *CAPTCHA*: один для обучения, другой для тестирования. Для обучения и тестирования синтезатора на основе *GAN* используется в общей сложности 1000 настоящих *CAPTCHA*, выбранных на определенном веб-сайте. Некоторые примеры (рис. 9). Из 1000 настоящих *CAPTCHA* 500 используется для обучения и оставшиеся 500 для тестирования. Это нужно для того, чтобы быть уверенным, что решатель действительно работает, а не просто запомнил ответы.

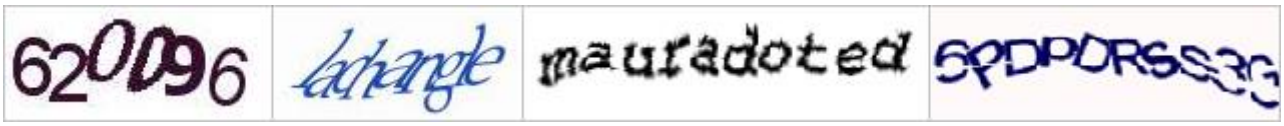

Рис. 9. Пример изображений ebay, Google, Wikipedia, Microsoft

Чтобы сгенерировать синтетические *CAPTCHA* для конкретной схемы, сначала инициализируем параметры функции безопасности. Затем используется исходные параметры для генерации первой партии синтетических *CAPTCHA*, которые используются вместе с 500 реальными *CAPTCHA* для автоматической тренировки синтезатора. После того, как обучился синтезатор, он используется для генерации синтетических образцов для обучения модели предварительной обработки и решателя. Для обучения модели предварительной обработки используются примерно 20000 синтетических примеров, а для обучения решателя используются примерно 200000 синтетических примеров.

Результаты распознавания

В таблице 1, показано на сколько успешно справляется данный решатель с распознаванием *CAPTCHA*, а также среднее время на решение одной *CAPTCHA*. Все тесты проводились на реальных *CAPTCHA*, которые не используются для обучения решателя. Стоит отметить, что данный решатель справился всего с 5% *CAPTCHA* от *Google*. Однако 5% выше 1% порога, для которого решатель *CAPTCHA* считается неэффективным [7].

Табл. 1. Общий показатель успешности и время выполнения решателя для каждой схемы CAPTCHA

Веб-сайт	Примеры	Распознанных *CAPTCHA*	Время на одну *CAPTCHA*
ebay		72%	29мс
Википедия		70%	47мс
Microsoft		67%	46мс
Aliexpres		61%	33мс
Яндекс		44%	50мс
Amazon		34%	42мс
Google		5%	40мс

Заключение

В данном исследовании был проведен пример построения решателя на основе генеративно-состязательной сети для распознавания текстовых *CAPTHCA*. Данный решатель автоматически обучается на предоставленных примеров и может работать с широким спектром схем *CAPTHCA*. В отличие от предыдущих работ на основе машинного обучения, данный подход требует значительно меньше реальных *CAPTHCA* для построения решателя. Это достигается, сначала обучением синтезатора *CAPTHCA*, для того чтобы автоматически генерировать синтетические примеры для обучения решателя. Основным преимуществом данной работы является то, что она требует меньшего участия человека при распознавании новой схемы *CAPTHCA*. Это означает, что решатель можно будет легко скорректировать, чтобы дообучить новыми схемами *CAPTHCA*.

Список литературы

1. Von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. CAPTCHA: Using Hard AI Problems for Security. Springer Berlin Heidelberg, 2003.
2. Goodfellow, I. J., Pougetabadie, J., Mirza, M., Xu, B., Wardefarley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. Advances in Neural Information Processing Systems 3, 2014.
3. Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks, arXiv:1611.07004, 2016.
4. Кадурин А. Глубокое обучение. Погружение в мир нейронных сетей / А. Кадурин, Е. Архангельская, С. Николенко. — СПБ.: Питер, 2018. — С. 480.
5. Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. Computer Science, 2014.
6. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. — М.: ДМК-Пресс, 2017.
7. Bursztein, E., Martin, M., and Mitchell, J. Text-based captcha strengths and weaknesses. In CCS, 2011.
8. Ye, G.; Tang, Z.; Fang, D.; Zhu, Z.; Feng, Y.; Xu, P.; Chen, X.; Wang, Z. Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. In Proceedings of the ACM Conference on Computer and Communications Security, 2018.
9. Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 1998.