# Подход к автоматическому повышению устойчивости моделей МО к внешним воздействиям на примере задачи биометрической идентификации диктора по голосу

Евгений Ильюшин, Дмитрий Намиот

***Аннотация*—Технологии искусственного интеллекта (далее ИИ), которые получили импульс в развитии в последние годы благодаря появлению значительного количества данных и вычислительных ресурсов, в свою очередь очень сильно повлияли на многие области человеческой жизни, а для некоторых из них стали неотъемлемой частью. Стоит отдельно отметить появление разнообразных средств биометрической идентификации пользователей, которые на сегодняшний день нашли свое применение как в системах общего назначения, так и в критически важных. К классу таких средств можно отнести биометрическую идентификацию пользователя по отпечатку пальца, лицу, голосу, радужной оболочке глаза, геометрии руки и т.д. Мы ежедневно используем данные средства в нашей повседневной жизни, когда пользуемся смартфонами, персональными компьютерами или взаимодействуем с банками, которые стали широко применять средства биометрической идентификации при взаимодействии с клиентами. Таким образом, вопрос надежности средств биометрической идентификации пользователей становиться безусловно важным и требующим должного внимания со стороны специалистов по информационной безопасности, а поскольку в основе данных систем как правило лежат модели машинного обучения (далее МО), то вопрос их устойчивости к внешним воздействиям играет ключевую роль. В данной работе мы представим подход к автоматическому повышению устойчивости моделей МО к внешним воздействиям, на примере задачи идентификации диктора по голосу.**

***Ключевые слова*—состязательные атаки, безопасность, глубокие нейронные сети, идентификация диктора, порождающие состязательные нейронные сети.**

## I. Введение

Важность повышения устойчивости моделей МО к внешним воздействиям, которые применяются в системах биометрической идентификации диктора по голосу, не вызывает сомнений. На сегодняшний день с помощью данной технологии банки идентифицируют пользователей в своих системах, пользователи, в свою очередь, могут выполнять перевод денежных средств по телефону, системы «умного дома» или голосовые помощники используют команды, которые должны принадлежать конкретному пользователю и т.д. Конечно злоумышленники пытаются использовать данные технологии в своих целях проводя атаки на данные системы с целью вызвать ошибку идентификации, что в свою очередь позволит выполнить определенные действия от лица целевого пользователя системы. В последние годы значительно увеличилась активность исследователей в области алгоритмов идентификации диктора по голосу и атак на них, а на самой крупной в мире конференции «Conference of the International Speech Communication Association (INTERSPEECH)» посвященной речевым технологиям стали проводить соревнования среди наиболее успешных алгоритмов идентификации и верификации диктора по голосу.

В связи с высокой актуальностью проблемы практически ежедневно публикуются новые работы на тему атак в целом на системы биометрической идентификации, а так же новые атаки на голосовые системы идентификации дикторов в частности, а так же методы защиты от них. Но до сих пор отсутствуют стандарты и общепризнанные подходы к безопасной разработке и защите систем данного класса. Таким образом целью данной работы является формализация проблемы устойчивости моделей МО к внешним воздействиям и описание подхода, который на примере задачи идентификации диктора по голосу продемострирует то, как можно защищать модели МО. Под моделями МО в данной работы мы будем понимать обученные искуственные нейронные сети (далее ИНС).

## II. Задача распознования диктора по голосу

Задачу распознавания диктора по голосу можно разделить на два класса: задача идентификации и задача верификации [1]. В случае идентификации мы используя аудиоматериал диктора, проходящего идентификацию, извлекаем признаки характерные для его голоса и затем, используя вычисленный вектор признаков ищем в нашей базе дикторов, наиболее похожего на исходного. В такой постановке, мы используем закрытое множество дикторов. В том случае, если множество дикторов открыто, появляется еще один возможный исход, когда голос исходного диктора не принадлежит ни одному из дикторов, хранящихся в базе. С другой стороны, верификация диктора предполагает проверку, действительно

Manuscript received May 19th, 2021
Евгений Ильюшин – МГУ им. М.В. Ломоносова, (email: eugene.ilyushin@gmail.com)
Дмитрий Намиот – МГУ им. М.В. Ломоносова, (email: dnamiot@gmail.com)

ли голос диктора, принадлежит ему. Хотя верификация диктора отличается от идентификации на этапе проверки, большинство современных систем верификации диктора обучаются с целью классифицировать дикторов из тренировочной выборки. Другими словами, эти модели обучаются используя в качестве целевой функции кросс-энтропию на уникальном наборе дикторов (то есть аналогично сценарию идентификации диктора). Формально, если $x \in \mathbb{R}^D$ обозначает аудиосэмпл с меткой диктора y, то обучение модели идентификации диктора обычно выполняется путем минимизации эмпирического риска (ERM):

$$\underset{\theta}{\operatorname{argmin}} \mathbb{E}_{(x,y)\sim D}[L(x, y, \theta)], \qquad \text{(II.1)}$$

где $L(\cdot)$ – функция потерь, а $\theta$ – параметры обучения модели.

Промежуточное представление обученной модели есть не что иное как признаковое описание голоса диктора или биометрический идентификатор, который суть вектор, далее полученные индентиифкаторы как правило погружают в метрическое пространство, в рамках которого и выполняется дальнейшая, уже метрическая классификация или в нашем случае идентификация.

## III. Обзор существующих атак на модели МО

Наиболее популярными в настоящее время являются состязательные атаки. Их смысл заключается в том, что мы добавляем к исходному аудио материалу небольшой шум, так чтоб заставить классификатор ошибиться.

$$\tilde{x} = x + \eta \text{ при условии что } \|f\|_p < \epsilon \qquad \text{(III.1)}$$

Другими словами если $x$ имеет истинную метку $y$, то злоумышленник вынуждает классификатор выдать $\tilde{y} \neq y$ для $\tilde{x}$.

Атаки бывают двух типов: целевые и нецелевые. Целевые подразумевают что мы добавляем шум так, чтобы классификатор идентифицировал атакующие данные как принадлежащие какому-то конкретному диктору из множества хранящегося в системе, а не целевые направлены на то чтобы заставить классификатор ошибиться, но при этом не важно в качестве какого диктора будет классифицирован исходный.

Еще одна классификация атак подразумевает три модели проведения: модель «белого», «серого» и «черного» ящиков. Модель «белого» ящика предполагает, что злоумышленник знает архитектуру модели, параметры, функцию потерь и градиенты. Это наиболее часто встречающаяся модель атаки в литературе. Модели «серого» и «черного» ящика предполагаю что наши знания о модели либо минимальны, либо вообще отсутствуют. В данной работе мы сосредоточимся на атаках «белого» ящика.

Помимо вышеперечисленных классификаций, атаки можно разделить на два подможества исходя из вектора атаки, первый из них направлен на градиент, а второй на уменьшение апостериорного значения истинного выходного класса и увеличения апостериорного значения для неправильного класса.

*1) Метод быстрого градиента:* В англоязычной литературе называется «Fast Gradient Sign Method», далее FGSM. Один из первых методов [2] создания состязательных примеров Goodfellow и д.р. предложили в 2015 году. Его суть заключается в том что мы, используя только знак градиента, перемещаемся в направлении градиента для увеличения значения функции потерь:

$$\tilde{x} = x - \alpha sign(\nabla_x L(x, y_t, \theta)) \text{ , где } \alpha > 0 \qquad \text{(III.2)}$$

Эта атака целевая, так как двигаясь в сторону антиградиента, мы увеличиваем вероятность детектирования целевого класса. С помощью параметра $\alpha$ мы контролируем уровень «шума», добавляемый к исходному примеру [3]. На практике стало ясно, что данный метод редко приводит к успешной атаке, так слишком маленькое значение $\alpha$ дает маленькое возмущение, а слишком большое сильно искажает исходный пример, что может нарушить его семантические свойства. Такой вид атаки еще называют атака $l_\infty$, так ее цель - минимизация расстояния Чебышева.

*2) Метод итеративного быстрого градиента:* В англоязычной литературе называется «Iterative Gradient Sign Method», далее IGSM. Подход, который является улучшенной версией FGSM. Его идея заключается в том что мы итеративно добавляем небольшой шум к исходному примеру, до тех пор пока не получим успешный состязательный пример или не достигнем максимального числа итераций. На каждой итерации IGSM обрезает текущий состязательный пример так, чтобы он находился в $\epsilon$ окрестности $x$ согласно расстоянию $l_\infty$. Формально это выглядит следующим образом:

$$\tilde{x_0} = x, \tilde{x_{n+1}} = Clip_{x,\epsilon}(\tilde{x_n} - \alpha \cdot sign(\nabla L(x_n, y_t, \theta))), \qquad \text{(III.3)}$$

где $\theta$ – параметры модели, $y_t$ – целевая метка.

*3) Проецируемый градиентный спуск:* В англоязычной литературе называется «Projected Gradient Descent», далее PGD [4]. Обобщенная версия IGSM, но в отличии от IGSM тут используется не знак градиента а его значение.

$$\tilde{x_0} = x, \tilde{x_{n+1}} = Clip_{x,\epsilon}(\tilde{x_n} - \alpha \cdot \nabla L(x_n, y_t, \theta)) \qquad \text{(III.4)}$$

*4) Алгоритм Карлини – Вагнера:* В англоязычной литературе называется $CW_2$, $CW_\infty$ и $CW_0$ [5]. Интуитивно, атака пытается максимизировать апостериорную вероятность класса, который не является истинным классом x, но имеет наиболее вероятный апостериорный класс среди всех неправильных классов. В качестве функции расстояния может использоваться $l_2$, $l_\infty$, или $l_0$.

Формально алгоритм описывается следующим образом:

$$\text{minimize}\{\|\tilde{x} - x\|_p^2 + c \cdot g(\tilde{x})\}, \qquad \text{(III.5)}$$

где c – параметр, с помощью которого выполняется регуляризация влияния слагаемого $g(\cdot)$, а $g(\cdot)$ – функция потерь, которая выглядит следующим образом:

$$\max\{Z(\tilde{x})_t - \max_{j \neq t}(Z(\tilde{x})_j) + \sigma\}, \qquad \text{(III.6)}$$

где $Z(\cdot)$ – вектор, содержащий апостериорные вероятности для всех классов; t обозначает элемент, соответствующий истинному классу y; $\sigma$ - параметр, регулирующий влияние степени уверенности.

*5) Якобианские карты значимости:* Особое внимание стоит обратить на способ добавления шума к исходному вектору. В случае с FGSM атакой мы добавляем одинаковый шум ко всем элементам вектора признаков, никак не дифференцируя признаки по значимости для принятия решения алгоритмом. В таком случае сложно создать атакующий вектор, но сам процесс мутации существующего достаточно быстрый и простой. Частично данный недостаток пытается минимизировать IGSM, итеративно добавляю минимальное возмущения, до тех пор пока вектор не выйдет за границу класса. Кардинально данную проблему пытались решить авторы метода якобианских карт значимости (анг. Jacobian-based Saliency Map Approach, далее JSMA) [6]. Данный метод основан на вычисление матрицы Якоби, используя которую мы можем определить какой элемент или элементы входного вектора вносят наибольший вклад при принятии решения моделью и далее меняем значения только этих элементов. Недостатком данного метода заключается в его вычислительной сложности.

*6) Порождающие состязательные ИНС:* В англоязычной литературе называется «Generative Adversarial Networks», далее GAN. Данный метод весьма эффективен, так как позволяет порождать искусственным образом данные, которые не отличимы от реальных, а значит распределения синтетических данных и реальных не различимы. Архитектура GAN состоят из двух частей: генератор $G$ и дискриминатор $D$. Цель дискриминатора понять, что данные, которые порождает генератор синтетические, а цель генератора обмануть дискриминатор. Формально, данный метод можно представить следующим образом:

$$G = G(z, \theta_g) : Z \to X \tag{III.7}$$

$$D = D(x, \theta_d) : X \to [0, 1] \tag{III.8}$$

$$L(D, G) = \mathbb{E}_{x\ p_{data}(x)}[\log D(x)] + \mathbb{E}_{x\ p_{gen}(x)}[log(1 - D(x))], \tag{III.9}$$

где $p_{data}$ – реальное распределение данных, $p_{gen}$ – распределение порождаемое генератором. Таким образом алгоритм обучения GAN достаточно простой, на каждом шаге мы то фиксируем веса G и обновляем веса D, то наоборот. Таким образом в ходе такой минимаксной игры мы решаем задачу оптимизации III.9 в ходе решения которой $p_{gen}$ сходиться к $p_{data}$. Формальное доказательство этого факта представлено в работе [7].

## IV. Обзор существующих методов и средств повышения устойчивости моделей МО

*1) Состязательное обучение:* Исходя из понимания, которое у нас сформировалось в процессе изучения существующих алгоритмов проведения атак, наиболее очевидным способом защиты моделей МО является состязательное обучение, смысл которого состоит в том что мы расширяем обучающую выборку с помощью примеров, полученных с помощью атакующих алгоритмов [4]. В таком случае задачу классификации можно переформулировать следующим образом:

$$\underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{(x,y)\sim D}[\max_{\eta:\|\eta\|_P<\epsilon} L(x + \eta, y, \theta)] \tag{IV.1}$$

Задача максимизации решается с помощью алгоритма атаки, используемого во время состязательного обучения, а минимизация – это стандартный ERM (II.1), используемый для обучения модели, параметризованной с помощью $\theta$. Мы по отдельности применяем как одношаговые алгоритмы FGSM (III.2), так и алгоритмы PGD (III.4) для решения внутренней задачи максимизации. Примечательно, что общее обучение проводится как на исходных данных, так и на полученных с помощью состязательных атак. Общая функция потерь определяется как:

$$L_{AT}(x, \tilde{x}, y, \theta) = (1 - w_{AT}) \cdot L(x, y, \theta) + w_{AT} \cdot L(\tilde{x}, y, \theta), \tag{IV.2}$$

где $w_AT$ – вес состязательного обучения.

*2) Состязательная регуляризация по Липшицу:* В англоязычной литературе называется «Adversarial Lipschitz Regularization», далее ALR. Этот подход к повышению устойчивости основан на использовании функции, которая не очень чувствительна к небольшому изменению входных данных. Другими словами, апостериорное распределение не должно резко меняться, если входное возмущение находится в пределах максимально допустимого порога. В работе [8] автор представил стратегию обучения, использующую состязательную технику Липшицевой регуляризации. Подобно регуляризации, основанной на гладкости локального распределения в Virtual Adversarial Training (VAT) [9], ALR вводит термин регуляризации, определяемый с помощью константы Липшица:

$$\|f\|_L = \sup_{x,\tilde{x}\in X, d_x(x,\tilde{x})>0} \frac{d_y(f(x), f(y))}{d_X(x, \tilde{x})}, \tag{IV.3}$$

где $f(\cdot)$ интересующая функция (реализованная нейронной сетью), которая отображает входное метрическое пространство $(X, d_x)$ в выходное метрическое пространство $(Y, d_y)$. В случае классификации дикторов авторы выбрали $f(\cdot)$ в качестве окончательного лог-апостериорного выхода сети, пример $f(x) = log(p(y|x, \theta)$, $l_1$ норма как $d_y$ и $l_2$ норма как $d_x$. Состязательное возмущение $\eta = \epsilon\eta_k$ в $\tilde{x} = x + \eta$ аппроксимируется степенными итерациями:

$$\eta_{i+1} = \frac{\nabla_{\eta_i} \cdot d_Y(f(x), f(x + \xi\eta_i))}{\|\nabla_{\eta_i} \cdot d_Y(f(x), f(x + \xi\eta_i))\|_2}, \tag{IV.4}$$

где $\eta_0$ инициализируется случайным образом, а $\xi$ – еще один гиперпараметр. Регуляризирующий оператор, используемый в обучении:

$$L_{ALR} = \left[\frac{d_Y(f(x), f(\tilde{x}))}{d_X(x, \tilde{x})} - K\right]_+, \tag{IV.5}$$

где $K$ – искомая константа Липшица.

## V. Предлагаемый подход

Эффективность предлагаемого подхода для автоматического повышения устойчивости моделей МО, продемонстрирована на примере задачи биометрической идентификации диктора по голосу. В качестве базовой технологии было решено использовать фаззинг. Впервые термин «fuzz» был введен Бартоном Миллером [10] в конце 1980-х и употреблялся в контексте описания подхода для тестирования и верификации программного обеспечения, который был противоположностью формальным. Основная идея данного подхода заключалась в генерации случайных последовательностей символов и использование их в качестве входных данных для тестируемой программы. В качестве тестируемых программ были выбраны утилиты входящие в состав нескольких дистрибутивов UNIX. В результате было обнаружено значительное количество ошибок, а также в ходе анализа ошибок были выявлены характерные проблемы, связанные с несоблюдением программистами определенных правил, которые и приводили к появлению этих ошибок. В последствии, такие лидеры индустрии как Google [11], Microsoft и прочие, стали применять фаззинг для всестороннего тестирования своего программного обеспечения, а взрывной рост технологий, связанных с машинным обучением, позволил создать интеллектуальные инструменты, благодаря которым увеличился охват областей, в которых имеет смысл применять.

### *A. Фаззинг ИНС*

*Искусственные нейронные сети* – это подход к созданию алгоритмов искусственного интеллекта с помощью машинного обучения. В ИНС иерархическая структура взаимосвязанных узлов, которые организованы в слои и обучаются на тренировочных данных для решения какой-то конкретной прикладной задачи. В нейронной сети отдельные нейроны выполняют простые вычисления. Способность достаточно точно приближать сложные функциональные зависимости – это результат простых вычислений, выполняемых нейронами и делающих свой небольшой вклад в общий результат. Простой расчет, выполняемый нейроном, состоит из двух шагов: во-первых, вычисление взвешенной суммы заданных входных значений и свободных членов, во-вторых, применения нелинейного преобразования к взвешенной сумме. За исключением последнего слоя нейронов, выходное значение нейрона становится входом одного или нескольких следующих нейронов. Выходные данные последнего слоя нейронов соответствуют значениям, вычисленным нейронной сетью. На этапе обучения для целевой функции подбираются только веса входных данных и их свободные члены. Таким образом, можно считать, что на логику принятия решения ИНС оказывают веса и свободные члены.

Из-за взаимосвязи между поведением всей нейронной сети и функцией, которую вычисляют ее отдельные нейроны, анализ правильности выводов, которые делает ИНС является нетривиальным. Помимо обученных весов и свободных членов, есть еще один очень важный аспект нейронных сетей, который влияет на то, насколько хорошо нейронная сеть сможет решить задачу – это архитектура сети. Архитектура нейронной сети описывает, сколько нейронов используется и как они связаны. В многослойной архитектуре отдельный нейрон использует некоторые или все выходные значения нейронов нижнего уровня для вычисления своего выходного значения, которое будет служить еще одним входом для следующего слоя ИНС. На примере ИНС, разработанных для распознавания изображений, упрощенно мы можем представить что первые слои детектируются простые элементы изображения (углы, линии и т.д.), а более глубокие может обнаруживать такие объекты, как небо, деревья или текст.

*1) Тестирование покрытия:* Проблема связанная с зависимостями общего результата от вычислений минимальных блоков программы не новая. Классическое программирование использует фундаментальные и в основном тривиальные утверждения для построения более крупных программных структур, которые решают сложные проблемы. Было доказано, что формальная верификация результата является вычислительно сложной задачей, а иногда и не разрешимой. Поэтому разработчики ПО придумали несколько способов решения указанной проблемы. Существуют подходы, ограничивающие степени свободы ПО, позволяющие проверить правильность кода. Другие подходы удовлетворяются тем, что предоставляют не доказательства, а свидетельства того, что программный артефакт ведет себя, как указано. Одним из таких подходов является подход к оценке качества программного обеспечения с помощью тестовых примеров. Тестовый пример состоит из набора входных данных для программы и ожидаемого значения, которое программа должна вернуть в соответствии со своей спецификацией. Если программа возвращает ожидаемое значение, тест проходит успешно, в противном случае он терпит неудачу. Этот подход позволяет инженеру-программисту локализовать проблемное пространство ввода и проверить правильность работы ПО на нем.

Важно иметь возможность качественно оценить, насколько хорошо эти тестовые примеры покрывают проблемное пространство. Поэтому инженеры-программисты ввели так называемые показатели покрытия тестами. Самый простой критерий оценки качества – это покрытие операторов. Он оценивает долю всех операторов в коде ПО, которые были вызваны хотя бы один раз при запуске тестовых примеров, относительно всех остальных операторов, присутствующих в тестируемой программе. Более сложные метрики покрытия, такие как покрытие ветвлений, требуют, чтобы тестовые примеры оценивали каждое решение, относящееся к потоку управления, на истинное или ложное.

Такие показатели тестового покрытия используются в отрасли для подтверждения того, что программное обеспечение было хорошо протестировано и поэтому считается безопасным с высокой вероятностью [4].

В работе [12] авторы представили базовый алгоритм тестирования покрытия ИНС и используемый критерий оценки качества, называемый покрытием нейронов, в нескольких последующих работах было проведено исследование того, какие еще критерии могут быть использованы. Среди наиболее интересных можно выделить работу [13], в которой рассмотрены критерии, которые не уделяют равного внимания отдельным нейронам. Поскольку важность и сложность функций варьируются на

разных слоях, нейроны первого слоя не следует считать такими же важными, как нейроны из последних слоев. В работе представлены следующие показатели покрытия:

- *Neuron Coverage (NC)* – покрытие нейронов, которая учитывает количество нейронов, получивших на вход положительные значение, а на выходе после нелинейного преобразования получили значения, превосходящее определенный порог;
- *k-multisection Neuron Coverage (KMNC)* – для этого вычисляется, для каждого нейрона, диапазон значений, наблюдаемых в процессе тренировки, делят этот интервал на k секций и определяют была ли ”затронута”каждая из секций;
- *Neuron Boundary Coverage (NBC)* – в отличии от KMNC, которое учитывает равномерно распределенные значения по одинаковым по размеру k-секциям, NBC учитывает только верхний и нижний диапазоны значений. NBC рассчитывает на тестовой выборке сколько раз выход нейронов попадал в верхний и нижний диапазоны значений. Верхний диапазон определяется как область выходных значений в интервале $[high_n, \infty]$. Нижний диапазон определяется как область выходных значений в интервале $[low_n, \infty]$. Нейрон считается полностью покрытым, если наблюдаются выходные значения, входящие в верхний и нижний диапазоны;
- *Strong Neuron Activation Coverage (SNAC)* – очень похоже на NBC. Соответствует проценту нейронов для наблюдаемых выходных значений, пренадлежащих вернему диапазону;
- *Top-k Neuron Coverage (TKNC)* – это показатель охвата, который рассчитывается для слоя. На каждом слое определяется top-k наиболее активных нейронов с наивысшими значениями. Одиночный нейрон считается покрытым согласно TKNC, если он попал на тестовой выборке в top-k хотя бы один раз. Общее покрытие соответствует проценту нейронов, которые были покрыты TKNC;
- *Top-k Neuron Patterns* – если посмотреть на комбинации k наиболее активных нейронов на разных $l$ слоях, можно получить паттерн активации, состоящий из $l \cdot k$ нейронов. Результирующее покрытие – это процент наблюдаемых паттернов активации от общего числа возможных паттернов активации;
- *Sign-Sign Coverage, Value-Sign Coverage, Sign-Value Coverage, Value-Value Coverage* – в работе [14] предложен другой набор критериев покрытия. Авторы основывают свое критерии на ранее упомянутом понятии признаков, на которые нейронная сеть реагирует на разных уровнях. Поскольку обычно неизвестно, какие признаки значимы и какие нейроны соответствуют их извлечению, авторы определили набор признаков слоя как набор всех возможных подмножеств нейронов слоя. Они определяют понятие смены знака для признака следующим образом: если для двух входов $x_1$ и $x_2$ знаки выходных значений всех нейронов признака различны в $x_2$, по сравнению со знаками выходов в $x_1$, то считается что $x_1$ наблюдает смену знака. Аналогично определяется изменение значения, но учитывается не изменение знака, а «значительное» изменение значения нейронов. Это понятие «значительного» изменения значения должно быть закодировано с использованием знаний предметной области при разработке архитектуры ИНС.

*2) Данные:* В качестве исходного набора данных для обучения и оценки качества моделей биометрической идентификации диктора по голосу, было решено использовать подмножество *train-clean-100* из открытого набора LibriSpeech [15], который состоит из 1000 часов английской речи. Набор данных был разбит на тренировочный и тестовый в соотношении 80/20 соответственно.

Далее были проведены эксперименты с различными промежуточными представлениями сигнала, а именно:

- спектрограммы;
- мел-спектрограммы;
- мел-кепстральные коэффеценты.

В результате проведенных экспериментов, используя спектрограммы и мел-спектрограммы, удалось обучить модель за одинаковое количество эпох без потери в качестве. Так как для получения спектрограмм требуется меньше вычислений, было решено использовать их, в качестве представления сигнала для обучения моделей и оценки их качества. Для извлечения спектрограмм к исходному акустическому сигналу, из которого выбирались случайным образом 3 секунды, было применено дискретное преобразование Фурье (ДПФ). Далее используя абсолютные значения полученных коэффициентов ДПФ получены спектрограммы. Для тех треков, которые были короче 3-х секунд, расширялось длинна за счет исходного сигнала. Таких записей было чуть более 1000 (рисунок 1).

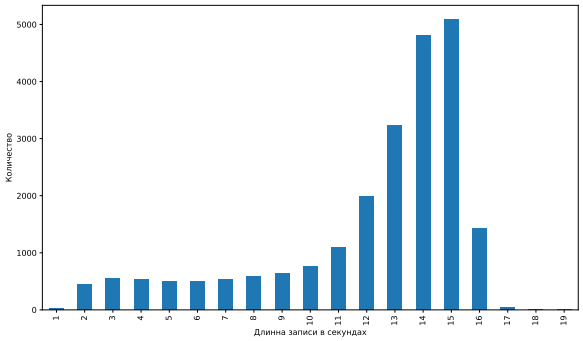


Рис. 1. Распределение длительности треков.

*3) Используемая архитектура ИНС:* В качестве исходных архитектур ИНС для моделей идентификации, которые согласно исследованиям показали приемлемые результаты в схожих задачах, были выбраны следующие:

- ResNet 34 [16]
- ResNeXt 50 [17]
- VGG 16 [18]

Далее предложенные архитектуры были оценены по следующим критериям:

1) скорость обучения – требуемое количество эпох, необходимое для достижения заданных критериев качества;

2) точность классификации, которая рассчитывалась по следующей формуле:

$$accuracy(y, \tilde{y}) = \frac{\sum_{i=0}^{N} 1(y_i = \tilde{y}_i)}{N}, \quad \text{(V.1)}$$

где $y_i$ – истинная метка класса для i-го примера из выборки, $\tilde{y}_i$ – предсказанная метка класса для i-го примера, а $N$ – общее количество элементов в выборке;

3) EER – Equal Error Rate (EER) [19]:

$$EER = \frac{FRR + FAR}{2}, \quad \text{(V.2)}$$

где $FRR$ – ошибки первого рода, $FAR$ – ошибки второго рода. Чем ниже значение EER, тем выше точность системы идентификации.

Таблица I
Результаты сравнительного анализа моделей идентификации диктора по голосу.

| Модель | Параметры | Эпохи | Точность | EER |
|---|---|---|---|---|
| ResNet 34 | 45 776 320 | 50 | 0.96 | 0.08 |
| ResNeXt 50 | 24 219 710 | 70 | 0.95 | 0.05 |
| VGG 16 | 15 104 955 | 70 | 0.93 | 0.09 |

В результате экспериментов лучшие результаты получены для *ResNet34*. Для ее обучения и достижения нужных характеристик качества требуется не более чем 50 эпох, так же она обладает лучшей обобщающей способностью, а архитектура более устойчива к небольшим изменениям входных данных.

*4) Критерии оценки качества:* Основным критерием оценки качества работы моделей биометрической идентификации диктора по голосу считается Equal Error Rate (EER) V.2. Модель идентификации обучается на образцах голоса нескольких человек. На этапе обучения для каждого человека создается биометрический идентификатор. Образец, который будет идентифицирован, сопоставляется с каждым известным идентификатором, путем вычисления метрики сходства или функции расстояния между двумя идентификаторами. Система сопоставляет идентификатор человеку с наиболее похожим биометрическим идентификатором. Чтобы предотвратить успешное прохождение идентификации злоумышленником (в данном случае все идентификаторы лиц, не известных системе), значение метрики сходства должно превышать определенный порог. Если этот порог не достигнут, идентификатор отклоняется.

### *B. Подход к фаззингу модели идентификации диктора*

Решая задачу фаззинга ИНС нам с одной стороны нужно максимизировать покрытие нейронов, выбрав одну из стратегий, а с другой определить текущие границы классов и тем самым выделить те области метрического пространства, где модель ошибочно принимает не верные решения.

Процесс фаззинга состоит из следующих шагов:

1) инициализация массива seeds, который на первом шаге содержит первый элемент из тренировочной выборки. Seed – это пример, используя который мы порождаем новые, состязательные примеры на текущем шаге фаззинга;
2) выбор seed с использованием алгоритма выборки Томпсона[20];
3) мутация текущего seed с помощью атакующих алгоритмов FGSM, PGM;
4) подача на вход модели и получение вектора предсказаний;
5) расчет метрики покрытия;
6) если покрытие увеличилось, а предсказанный класс равен исходному, добавляем seed в текущем состоянии в массив seeds;
7) если предсказанный класс и исходный разные, добавляем в массив атакующих примеров, сохраняем информацию о граничных значениях и процесс мутации для данного seed прекращается.

При расчете метрики покрытия учитывались те нейроны, которые реже были активированы на предыдущих итерациях.

### *C. Автоматическое повышение устойчивости акустической модели*

После этапа фаззинга у нас собрана информация об успешности проведенных атак, включая атакующие примеры и методы, с помощью которых они были созданы, а также получены граничные значения для классов. Используя данную информацию система запускает процесс переобучения модели, с целью повысить ее устойчивость к атакам. Зная граничные значения для классов, система автоматически изменяет существующие границы, так чтоб подпространство разных классов не пересекались. А используя синтетические данные, расширяется множество исходных классов, путем добавления нового, к которому относятся синтетические данные.

Далее система возвращается на этап оценки качества и устойчивости модели к внешним воздействиям. Цикл продолжается до тех пор пока модель не достигнет требуемых характеристик на используемом наборе данных, либо пока не будет достигнуто заданное количество итераций.

### *D. Результаты экспериментов*

На первом шаге была проведена оценка качества исходной модели с помощью выбранных метрик оценки качества, а так же проведена кластеризация. В процессе кластеризации сначала был применен метод главных компонент, для понижения размерности признаковых описаний, далее применялся метод стохастического вложения соседей с t-распределением. *Исходная модель демострировала точность 95% и ERR 0.08% на тестовом наборе данных.*

Затем были сгенерированны атакующие примеры, на основе тестовой выборки. Для каждого исходного примера создавалось два атакующих, один с помощью FGSM, другой с помощью PGM алгоритмов для разных $\epsilon$. После чего была вновь проведена оценка качества уже на атакующих примерах, после чего результаты были визуализированы. Результаты экспериментов для одной итерации

Таблица II
Результаты проведения атак с разным значением $\epsilon$.

| $\epsilon$ | Точность | | | EER | | |
|---|---|---|---|---|---|---|
| | Исходная | FGSM | PGM | Исходная | FGSM | PGM |
| 0.001 | 0.95 | 0.25 | 0.16 | 0.08 | 0.13 | 0.14 |
| 0.002 | 0.95 | 0.13 | 0.05 | 0.08 | 0.16 | 0.15 |
| 0.003 | 0.95 | 0.09 | 0.03 | 0.08 | 0.18 | 0.16 |
| 0.004 | 0.95 | 0.08 | 0.03 | 0.08 | 0.19 | 0.17 |
| 0.005 | 0.95 | 0.06 | 0.03 | 0.08 | 0.20 | 0.17 |

фаззинга представлены в таблице II. Все эксперименты на предмет оценки качества и устойчивости проводились на тестовом наборе данных.

Далее были проведены не целевые атаки как направленные на имитацию спектрограмм, так и на имитацию акустического сигнала с помощью GAN. Эксперимент состоял из 20 итераций для каждого типа атаки, в рамкой которых порождалось по 100 атакующих примеров. Для атак на спектрограммы в среднем на каждом шаге успешными были 14 из 100. Основными дикторами, спектрограммы которых удавалось подделать были дикторы со следующими идентификаторами: 1, 147, 27, 156, 31, 163, 164, 38, 167, 168, 166, 171, 51, 58, 61, 62, 193, 207, 208, 104, 239, 124. В случае атак на акустический сигнал в среднем успешными были 9 из 100. Идентификаторы дикторов, голоса которых удалось сымитировать были следующие: 140, 146, 156, 162, 39, 179, 58, 62, 191, 71, 208, 210, 219, 225, 230, 104, 235, 108, 117. Успешные атакующие примеры далее были включены в обучающую выборку в качестве отдельного класса.

Сохранив информацию об проведенных экспериментах, система запускает процесс фаззинга, цель которого, с одной стороны увеличить процент покрытия нейроном модели, а с другой максимизировать число найденных ошибок классификации, что в итоге позволяет сформировать представление о том, для каких видов атак модель уязвима, где проходят границы классов и какие из подпространств следует покрыть на следующем шаге обучения модели. Seed для следующего шага фаззинга выбирался с помощью алгоритма выборки Топсона, который с одной стороны позволяет выбирать следующий seed, который вероятней всего позволит максимизировать покрытие, а с другой привести к ошибке классификации. Стоит отметить что со временем возможность увеличить покрытие у seed вырождается, поэтому число мутаций было ограниченно до 3.

В фаззере использовались эпохи с фиксированным количеством проходов. Всего эпох было 5, ввиду вычислительной сложности процесса фаззинга. В качестве исходного набора данных, использовался тренировочный. Результаты представлены в таблице III. В начале цикла фаззинга, когда исходная модель была атакована, точность уменьшилась до 54%, далее после одного цикла фаззинга и переобучения модели, точность увеличилась до 85% при $\epsilon = 0.001$. Стоит отметить что точность на исходном наборе данных данных, без атакующих примеров, тоже уменьшилась на значения для разных эпсилон из диапазона от 7 до 10 процентов.

Таблица III
Результаты фаззинга.

| Эпоха | Покрытие | Точность | | | |
|---|---|---|---|---|---|
| | | $\epsilon = 0.001$ | | $\epsilon = 0.002$ | |
| | | до | после | до | после |
| 1 | 0.64 | 0.35 | 0.85 | 0.23 | 0.84 |
| 2 | 0.71 | 0.72 | 0.87 | 0.64 | 0.83 |
| 3 | 0.75 | 0.64 | 0.82 | 0.70 | 0.79 |
| 4 | 0.79 | 0.80 | 0.85 | 0.72 | 0.85 |
| 5 | 0.85 | 0.81 | 0.87 | 0.75 | 0.87 |

## VI. Заключение

В работе сформулирована задача идентификации диктора по голосу, затем рассморены сщуствующие атаки на модели МО и методы повышения устойчивости. Далее описан предложенный подход к автоматическому повышению устойчивости моделей МО к внешним воздействиям с применение технологии фаззинга. После чего представлены результаты проведенного эксперимента. Стоит отметить что эксперимент проводился на нецелевых атаках. Полученные результаты могут служить основой для дальнейших исследований и разработки технологий автоматического повышения устойчивости моеделей МО для других типов данных.

## Библиография


[1] Z. Bai and X.-L. Zhang, “Speaker recognition based on deep learning: An overview.” [Online]. Available: http://arxiv.org/abs/2012.00931

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples.” [Online]. Available: http://arxiv.org/abs/1412.6572

[3] X. Cao and N. Z. Gong, “Mitigating evasion attacks to deep neural networks via region-based classification.” [Online]. Available: http://arxiv.org/abs/1709.05583

[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks.” [Online]. Available: http://arxiv.org/abs/1706.06083

[5] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks.” [Online]. Available: http://arxiv.org/abs/1608.04644

[6] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings.” [Online]. Available: http://arxiv.org/abs/1511.07528

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks.” [Online]. Available: http://arxiv.org/abs/1406.2661

[8] D. Terjék, “Adversarial lipschitz regularization.” [Online]. Available: http://arxiv.org/abs/1907.05681

[9] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, “Distributional smoothing with virtual adversarial training.” [Online]. Available: http://arxiv.org/abs/1507.00677

[10] B. P. Miller, L. Fredriksen, and B. So, “An empirical study of the reliability of UNIX utilities,” vol. 33, no. 12, pp. 32–44. [Online]. Available: https://dl.acm.org/doi/10.1145/96267.96279

[11] A. Abhishek and N. Cris. Fuzzing for security. [Online]. Available: https://blog.chromium.org/2012/04/fuzzing-for-security.html

[12] K. Pei, Y. Cao, J. Yang, and S. Jana, “DeepXplore: Automated whitebox testing of deep learning systems,” pp. 1–18. [Online]. Available: http://arxiv.org/abs/1705.06640

[13] N. Wenzler, “Not all neurons are created equal: Towards a feature level deep neural network test coverage metric.” [Online]. Available: http://www.cs.toronto.edu/~chechik/courses19/csc2125/project/nils-final.pdf

[14] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, “Testing deep neural networks.” [Online]. Available: http://arxiv.org/abs/1803.04792

[15] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: an ASR corpus based on public domain audio books,” in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pp. 5206–5210.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition.” [Online]. Available: http://arxiv.org/abs/1512.03385

[17] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks.” [Online]. Available: http://arxiv.org/abs/1611.05431

[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition.” [Online]. Available: http://arxiv.org/abs/1409.1556

[19] J. Oglesby, “What’s in a number? moving beyond the equal error rate,” vol. 17, no. 1, pp. 193–208. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/016763939500017I

[20] E. Gangan, M. Kudus, and E. Ilyushin, “Survey of multiarmed bandit algorithms applied to recommendation systems,” vol. 9, no. 4, pp. 12–27. [Online]. Available: http://injoit.org/index.php/j1/article/view/1093

# An approach to the automatic enhancement of the robustness of ML models to external influences on the example of the problem of biometric speaker identification by voice

Eugene Ilyushin, Dmitry Namiot

***Abstract*—AI technologies, which have received an impetus in development in recent years due to the emergence of a significant amount of data and computing resources, have greatly influenced many areas of human life, and for some of them, they have become an integral part.**
**It is worth noting separately the emergence of various means of biometric user identification, which today have found their application in general-purpose systems and critical ones. Such tools include biometric user identification by fingerprint, face, voice, iris, hand geometry, etc. We use these tools daily in our everyday life when we use smartphones, personal computers, or interact with banks, which have begun to use biometric identification tools when interacting with customers widely.**
**Thus, the reliability of biometric identification of users becomes undoubtedly important and requires due attention from information security specialists. Since these systems are usually based on ML models, their resistance to external influences plays a key role. In this paper, we presented an approach to automatically increasing the stability of ML models to external influences, using the example of the speaker identification problem by voice.**

***Keywords*—adversarial attack, deep neural network, speaker identification, generative adversarial networks**

## References

[1] Z. Bai and X.-L. Zhang, "Speaker recognition based on deep learning: An overview." [Online]. Available: http://arxiv.org/abs/2012.00931
[2] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples." [Online]. Available: http://arxiv.org/abs/1412.6572
[3] X. Cao and N. Z. Gong, "Mitigating evasion attacks to deep neural networks via region-based classification." [Online]. Available: http://arxiv.org/abs/1709.05583
[4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks." [Online]. Available: http://arxiv.org/abs/1706.06083
[5] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks." [Online]. Available: http://arxiv.org/abs/1608.04644
[6] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings." [Online]. Available: http://arxiv.org/abs/1511.07528
[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks." [Online]. Available: http://arxiv.org/abs/1406.2661
[8] D. Terjék, "Adversarial lipschitz regularization." [Online]. Available: http://arxiv.org/abs/1907.05681
[9] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training." [Online]. Available: http://arxiv.org/abs/1507.00677
[10] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of UNIX utilities," vol. 33, no. 12, pp. 32–44. [Online]. Available: https://dl.acm.org/doi/10.1145/96267.96279
[11] A. Abhishek and N. Cris. Fuzzing for security. [Online]. Available: https://blog.chromium.org/2012/04/fuzzing-for-security.html
[12] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated whitebox testing of deep learning systems," pp. 1–18. [Online]. Available: http://arxiv.org/abs/1705.06640
[13] N. Wenzler, "Not all neurons are created equal: Towards a feature level deep neural network test coverage metric." [Online]. Available: http://www.cs.toronto.edu/~chechik/courses19/csc2125/project/nils-final.pdf
[14] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, "Testing deep neural networks." [Online]. Available: http://arxiv.org/abs/1803.04792
[15] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an ASR corpus based on public domain audio books," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, pp. 5206–5210.
[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." [Online]. Available: http://arxiv.org/abs/1512.03385
[17] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks." [Online]. Available: http://arxiv.org/abs/1611.05431
[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." [Online]. Available: http://arxiv.org/abs/1409.1556
[19] J. Oglesby, "What's in a number? moving beyond the equal error rate," vol. 17, no. 1, pp. 193–208. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/016763939500017I
[20] E. Gangan, M. Kudus, and E. Ilyushin, "Survey of multiarmed bandit algorithms applied to recommendation systems," vol. 9, no. 4, pp. 12–27. [Online]. Available: http://injoit.org/index.php/j1/article/view/1093