ГБОУ города Москвы Школа № 1505 «Преображенская»

**Сборник**

**Применение нейронных сетей**

Москва,  
2018

Смирнов Артем 8 «В»

**Что такое нейронные сети и как их используют?**

Искусственные нейронные сети (ИНС) — это математические модели, а также их программное или аппаратное воплощение. Структура нейронной сети пришла в мир программирования прямиком из биологии, поэтому проще говоря, ИНС - это последовательность нейронов, соединенных между собой синапсами (место контакта между двумя нейронами). Каждый нейрон подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим нейронам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые нейроны вместе способны выполнять довольно сложные задачи. Например, машина обретает способность анализировать и даже запоминать различную информацию. Нейронные сети также способны не только анализировать входящую информацию, но и воспроизводить ее из своей памяти. Другими словами, нейросеть это машинная интерпретация мозга человека, в котором находятся миллионы нейронов, передающих информацию в виде электрических импульсов.

ИНС используются в разных практических целях. В числе задач, решение которых доверяют искусственным нейронным сетям, можно назвать следующие: распознавание текстов, системы безопасности и видеонаблюдения, автоматизация процессов распознавания образов, прогнозирование – и это далеко не все.

Если разделять применение ИНС по сферам жизни, то получится следующее:

**Экономика и бизнес:** прогнозирование курсов валют, цен на сырьё, покупательского спроса, объемов продаж и т.д., автоматический трейдинг (торговля на валютной, фондовой или товарной бирже), оценка рисков невозврата кредитов, предсказание банкротств, оценка стоимости недвижимости, оптимизация товарных и денежных потоков, считывание и распознавание чеков и документов, безопасность транзакций по пластиковым картам.

**Медицина и здравоохранение:** постановка диагноза больному, обработка медицинских изображений, мониторинг (отслеживание) состояния пациента, прогнозирование результатов применения разных методов лечения, анализ эффективности проведённого лечения.

**Авионика:** обучаемые автопилоты, беспилотные летательные аппараты.

**Связь:** сжатие видеоинформации, быстрое кодирование-декодирование.

**Интернет:** ассоциативный поиск информации, электронные секретари и автономные агенты в интернете, фильтрация и блокировка спама, автоматическая рубрикация сообщений из новостевых лент, распознавание captcha (тест для различения компьютеров и людей, используется, например, в Google).

**Автоматизация производства:** оптимизация режимов производственного процесса, контроль качества продукции, предупреждение аварийных ситуаций.

**Робототехника:** распознавание сцены, объектов и препятствий перед роботом, прокладка маршрута, управление манипуляторами («руками» роботов), поддержание равновесия робота.

**Политологические и социологические технологии:** предсказание результатов выборов, анализ опросов.

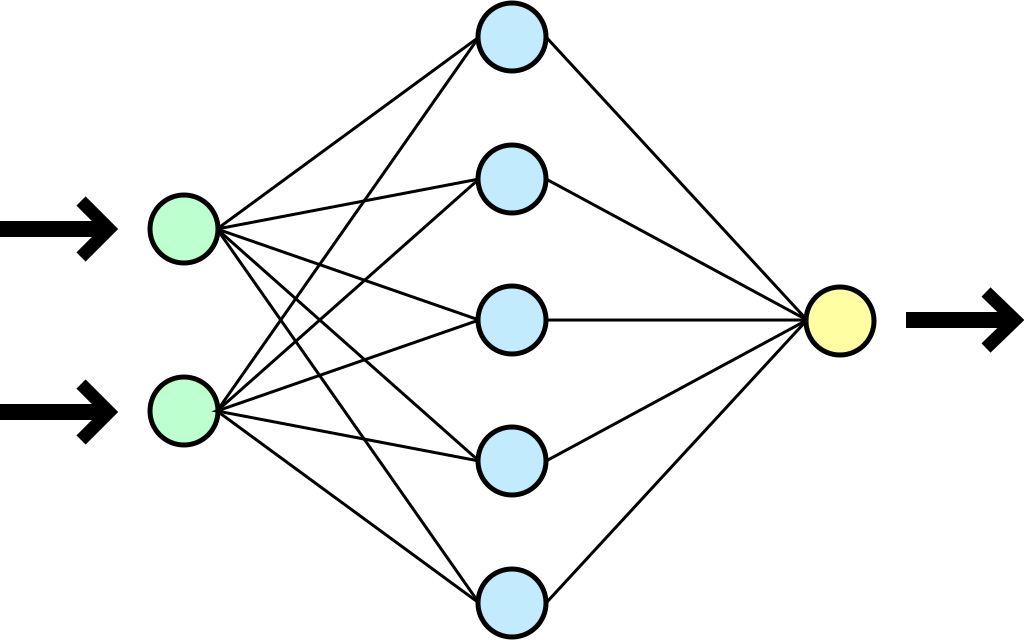
**Безопасность, охранные системы:** распознавание лиц; идентификация личности по отпечаткам пальцев, голосу, подписи или лицу; распознавание автомобильных номеров, мониторинг информационных потоков в компьютерной сети и обнаружение вторжений, обнаружение поддельных документов.

**Ввод и обработка информации:** распознавание рукописных текстов, отсканированных почтовых, платежных, финансовых и бухгалтерских документов; распознавание речевых команд, речевой ввод текста в компьютер.

**Геологоразведка:** анализ сейсмических данных, оценка ресурсов месторождений.

**Компьютерные и настольные игры:** создание нейроигроков в шашки и шахматы (подтверждённые игрой с людьми рейтинги - на уровне мастеров и международных мастеров).

**Основные термины для работы с нейронными сетями**

Нейрон — это вычислительная единица, которая получает информацию, производит над ней простые вычисления и передает ее дальше. Нейроны делятся на три основных типа: входной, скрытый и выходной. В том случае, когда нейросеть состоит из большого количества нейронов, вводят термин слоя.

Выходной слой

Входной слой

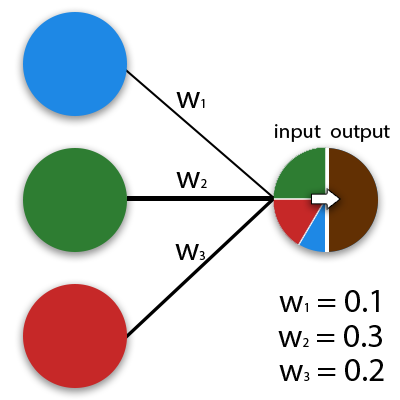
Скрытый слой

Схема многослойной нейронной сети

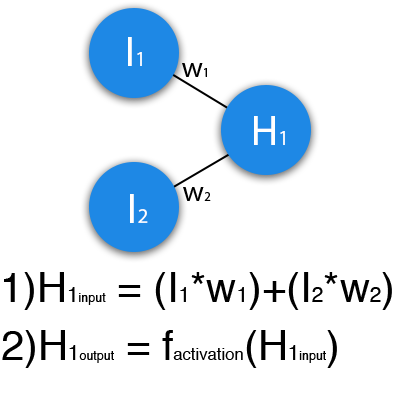
Соответственно, есть входной слой, который получает информацию, несколько скрытых слоев (обычно их не больше 3), которые ее обрабатывают и выходной слой, который выводит результат[1]. У каждого из нейронов есть 2 основных параметра: входные данные (input data) и выходные данные (output data). В случае входного нейрона: input=output. В остальных, в поле input попадает суммарная информация всех нейронов с предыдущего слоя, после чего, она нормализуется, с помощью функции активации (пока что просто представим ее f(x)) и попадает в поле output.[2]

Синапс — это связь между двумя нейронами. Синапс характеризуется своим *весом*[1] Благодаря весу, входная информация изменяется, когда передается от одного нейрона к другому.

Например, есть 3 нейрона, которые передают информацию следующему. Тогда у нас есть 3 веса, соответствующие каждому из этих нейронов. У того нейрона, у которого вес будет больше, та информация и будет доминирующей в следующем нейроне[2] .

Легче всего принцип работы нейросетей можно представить на примере смешения цветов. Синий, зеленый и красный нейрон имеют разные веса. Информация того нейрона, вес которого больше будет доминирующей в следующем нейроне. (вес обозначен буквой w) [3]

**Разбираем работу нейронной сети на примере**



На схеме изображена схема нейронной сети, где буквами I обозначены входные нейроны, буквой H — скрытый нейрон, а буквой w — веса. Из формулы видно, что входная информация — это сумма всех входных данных, умноженных на соответствующие им веса. Тогда дадим на вход значения 1 и 0. Пусть веса будут w1=0.4 и w2 = 0.7 Входные данные нейрона Н1 будут следующими: 1\*0.4+0\*0.7=0.4.[2] Теперь когда у нас есть входные данные, мы можем получить выходные данные, подставив входное значение в *функцию активации*.

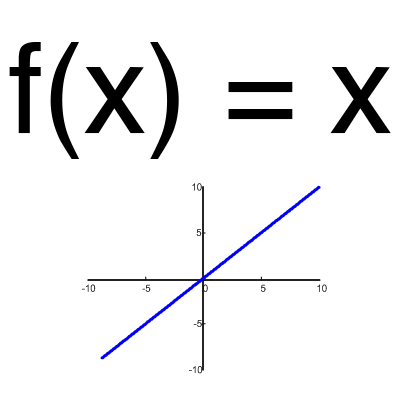
Теперь, когда у нас есть выходные данные, мы передаем их дальше. Так мы повторяем для всех слоев, пока не дойдем до выходного нейрона. Запустив такую сеть в первый раз ответ будет далек от правильного, потому что сеть не натренирована. Чтобы улучшить результаты мы будем ее тренировать, вводя ей новые значения. Но прежде, чем узнать как это делать, давайте рассмотрим еще несколько терминов и свойств нейронной сети.

*Функция активации* — функция, вычисляющая выходной сигнал искусственного нейрона. Это способ нормализации входных данных. То есть, если на входе у вас будет большое число, пропустив его через функцию активации, вы получите выход в нужном вам диапазоне[4] .

Функций активации достаточно много поэтому мы рассмотрим самые основные: *Линейная*, *Сигмоид* (который позже нам понадобится для обучения нашей собственной НС) и *Гиперболический тангенс*[4]. Главные их отличия — это диапазон значений.

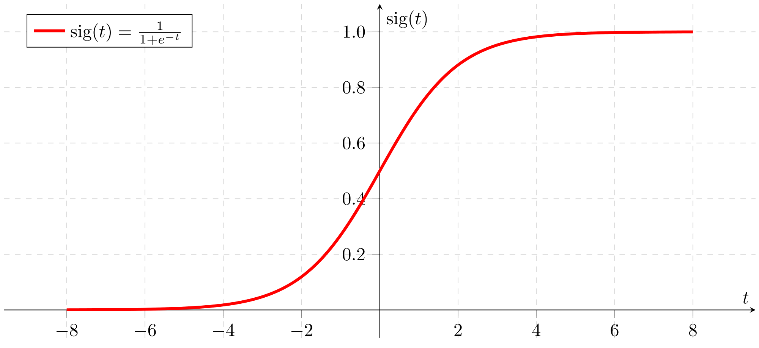
Здесь нам пригодятся знания, полученные на уроках алгебры.

**Линейная функция**

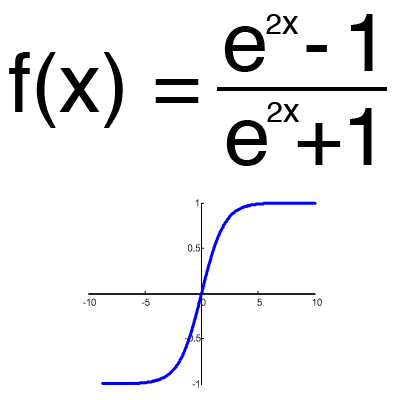


Эта функция почти никогда не используется, за исключением случаев, когда нужно протестировать нейронную сеть или передать значение без преобразований[2] .

**Сигмоид**

Это самая распространенная функция активации, ее диапазон значений [0,1]. Если в вашем случае присутствуют отрицательные значения (например, акции могут идти не только вверх, но и вниз), то вам понадобиться функция которая захватывает и отрицательные значения[4] .

**Гиперболический тангенс**

Эту функцию имеет смысл использовать только тогда, когда ваши значения могут быть и отрицательными, и положительными, так как диапазон функции [-1,1]. Использовать эту функцию только с положительными значениями нецелесообразно так как это значительно ухудшит результаты вашей нейросети[2] .

**Тренировочный сет**

Тренировочный сет — это последовательность данных, которыми оперирует нейронная сеть[5].

**Итерация**

Это своеобразный счетчик, который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет. Можно сказать, это общая численность данных сетов, которые прошла сеть[5].

**Эпоха**

При инициализации нейронной сети эта величина устанавливается в 0 и имеет верхнюю границу, задаваемый вручную. Чем больше эпоха, тем лучше натренирована сеть и соответственно, тем точнее ее результат. Эпоха увеличивается каждый раз, когда мы проходим весь набор тренировочных сетов[5].

**Ошибка**

Ошибка — это процентная величина, отражающая расхождение между ожидаемым и полученным ответами. Ошибка формируется каждую эпоху и должна идти на уменьшение. Если этого не происходит, значит, вы что-то делаете не так[5].

Есть три основных способа, помогающие вычислить ошибку: *Mean Squared Error (MSE)*, *Root MSE* и *Arctan*.

MSE

Root MSE

Artcan

Здесь нет какого-либо ограничения на использование, как в функции активации, и вы вольны выбрать любой метод, который будет приносить вам наилучший результат. Стоит лишь учитывать, что каждый метод считает ошибки по разному. У Arctan, ошибка, почти всегда, будет больше, так как он работает по принципу: чем больше разница, тем больше ошибка. Принцип подсчета ошибки во всех случаях одинаков. За каждый сет, мы считаем ошибку, отняв от идеального ответа, полученный. Далее, либо возводим в квадрат, либо вычисляем квадратный тангенс из этой разности, после чего полученное число делим на количество сетов[2] .

Итак, настал самый важный и ответственный момент в нашем сборнике – **Создание нейронной сети**



Но перед этим нам надо установить некоторые необходимые библиотеки **TensorFlow** и **Keras** (о них мы расскажем позже) для работы с НС. Для более оптимальной работы желательно использовать операционную систему Linux. В крайнем случае можно скачать *виртуальную машину* (это искусственно созданный при помощи специального ПО виртуальный компьютер, имеющий свою оперативную память, жесткий диск, процессор, а также ОС[16]). В качестве виртуальной машины подойдет VirtualBox[18].

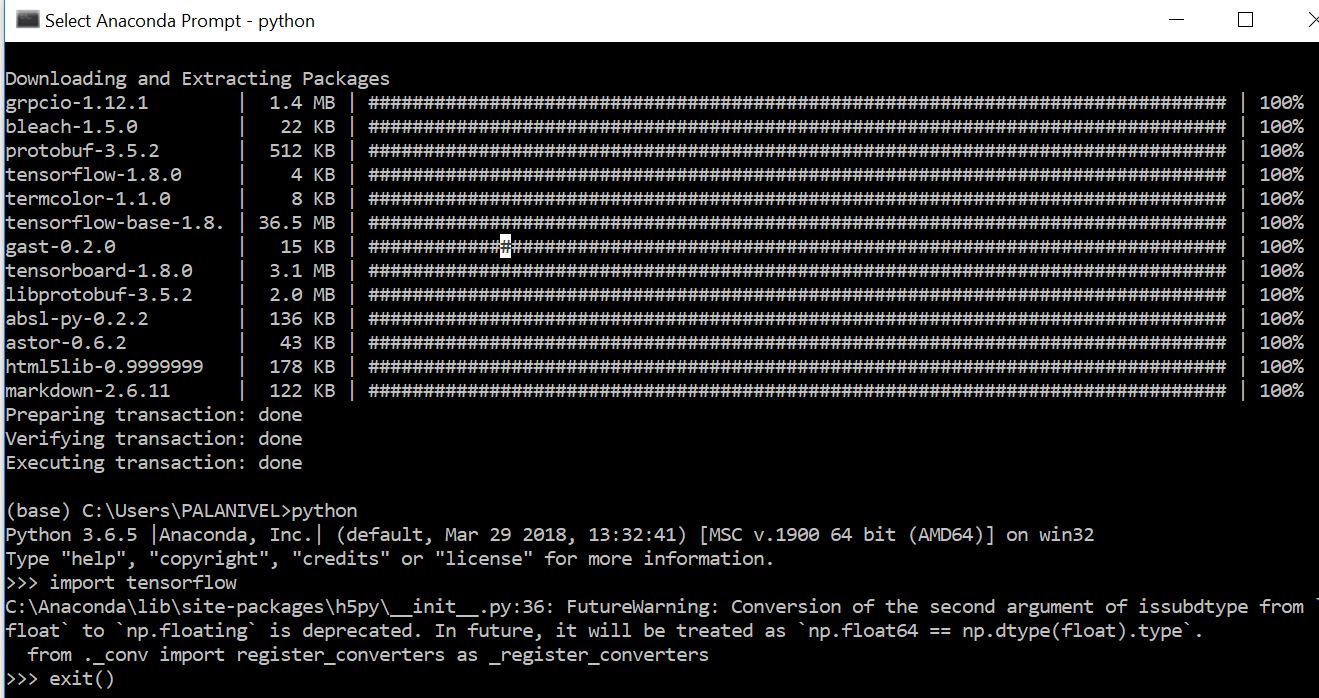
Для этого нам нужно зайти на сайт <https://www.anaconda.com/> и установить **Anaconda**, установите версию Anaconda для своей операционной системы. Выбирайте Python версии 3.6[17].

После установки открываем Anaconda Prompt. Если при запуске перед Вами подобное, то все было сделано верно.



Для установки TensorFlow используйте команду:

conda install tensorflow

У Вас после этого текст в программе должен выглядеть примерно так:

TensorFlow для CPU установится автоматически со всеми необходимыми зависимостями. Проверяем, что установка прошла успешно:

python -c "import tensorflow; print(tensorflow.\_\_version\_\_)"

1.2.1

После установки TensorFlow можно устанавливать Keras. Для Linux используйте команду:

conda install keras

По-умолчанию после установки Keras будет сконфигурирован на работу с TensorFlow. На всякий случай проверим содержимое конфигурационного файла .keras/keras.json в домашнем каталоге пользователя:

{

"floatx": "float32",

"epsilon": 1e-07,

"image\_data\_format": "channels\_last",

"backend": "tensorflow"

}

Проверяем, что все работает:

python -c "import keras; print(keras.\_\_version\_\_)"

Using TensorFlow backend.

2.0.5

Теперь, когда все готово, начнем создание нейронной сети. Мы рассмотрим задачу, связанную с диагностикой риска заболевания сахарным диабетом на основе состояния пациента. Для этого мы напишем простую нейронную сеть, которая решает важную практическую задачу.

Для этого мы будем использовать библиотеку Keras, которая представляет собой интерфейс для создания нейронных сетей. Библиотека Keras позволяет запускать нейронные сети с минимальным количеством операций. В качестве модели нейронной сети используется последовательная Sequental из модуля *keras.models* с заданием слоев *keras.layers*.

Экспериментальные данные (см. конец параграфа) представляют собой файл со значениями, разделенными запятыми, в котором каждая строка соответствует одной анкете. В наборе экспериментальных данных имеется девять параметров. Последний из них, целевой, показывает, наблюдался ли у пациента сахарный диабет или нет (1 - да или 0 - нет). Восемь остальных параметров также имеют численные значения:

1. Число беременностей (все пациенты из источника – женщины не моложе 21 года индийской народности пима).
2. Концентрация глюкозы в плазме через 2 часа после введения в пероральном глюкозотолерантном тесте.
3. Диастолическое артериальное давление (мм рт. ст.).
4. Толщина кожной складки в районе трицепса (мм).
5. Концентрация инсулина в сыворотке крови (мкЕд/мл).
6. Индекс массы тела (вес в кг/(рост в м)^2).
7. Функция, описывающая генетическую предрасположенность к диабету (diabetes pedegree).
8. Возраст (годы).

В качестве модели нейронной сети используется последовательная Sequental. Sequental это линейная последовательность слоев нейронной сети. Модель Sequental должна знать, какую форму ввода она должна ожидать. По этой причине первый слой в последовательной модели (и только первый, поскольку последующие слои могут делать автоматический вывод формы) должен получать информацию о своей входной форме. Для этого из модуля keras.models с заданием слоев keras.layers типа Dense.

1. from keras.models import Sequential  
2. from keras.layers import Dense  
3. import **numpy**

В третье строке мы импортируем NumPy — библиотеку с исходным кодом (текст компьютерной программы на каком-либо языке программирования или языке разметки, который может быть прочтён человеком) для Python

Для последующей воспроизводимости результатов зафиксируем генератор случайных чисел при помощи функции **random.seed() (random-случайность)**из библиотеки numpy. Считаем данные из датасета:

4. numpy.random.seed(2)

5. dataset = numpy.loadtxt("prima-indians-diabetes.csv", delimiter=",")

Разделим данные на матрицу признаков **X** и вектор целевой переменной **Y**(последний столбец датасета):

6. X = dataset[:,0:8]  
7. Y = dataset[:,8]

Создаем модель нейронной сети:

8. model = Sequential()

Опишем структуру модели нейронной сети. Определим входной, выходной и скрытые слои. Как мы уже написали, наша нейронная сеть будет иметь плотную (**Dense**) структуру – каждый нейрон связан со всеми нейронами следующего слоя. Выходной слой будет состоять из единственного нейрона, определяющего вероятность заболевания диабетом.

Слой добавляется к модели методом **add()**. Для входного слоя необходимо указать число признаков **input\_dim**, равное в нашем случае 8 (так как у нас имеется восемь параметров):

1. model.add(Dense(12, input\_dim=8, activation='relu'))

Для выходного слоя воспользуемся сигмоидной функцией для определения конечной вероятности риска заболевания.

1. model.add(Dense(15, activation='relu'))
2. model.add(Dense(8, activation='relu'))
3. model.add(Dense(10, activation='relu'))
4. model.add(Dense(1, activation='sigmoid'))

Перед тем, как начать тренировать модель, ее нужно *скомпилировать*.

Что это такое? Допустим, какой-нибудь программист написал программу на каком-нибудь языке программирования, например, Delphi программу. И эта программа включается только через этот Delphi. А как ее распространять? Вместе с Delphi? Для этого и есть компилирование. Оно кодирует программу с языка программирования на машинный код компьютера (в виде двоичного кода, как 101 и т. д.) в исполняющий файл exe.

Компилируем сеть при помощи метода compile():

1. model.compile(loss="binary\_crossentropy", optimizer="adam", metrics=['accuracy'])

Методу передается три параметра:

* loss – функция потерь – объект, который модель стремиться минимизировать;
* optimizer – оптимизатор, мы используем встроенный метод стохастической оптимизации adam
* metrics – список метрик оптимизации, для задач классификации используется метрику ‘accuracy’.

Для обучения нейронной сети применяем метод **fit()**:

1. model.fit(X, Y, epochs = 1000, batch\_size=10)

Параметр **epochs** – «эпохи» – количество проходов нейронной сети по всем записям датасета (выбирается исходя из того, насколько быстро модель с каждым новым проходом приближается к желаемой предсказательной точности), **batch\_size** – количество объектов выборки, берущихся за один шаг. В процессе обучения API будет выводить соответствующие строчки с величинам функции потерь и метрики для каждой из эпох.

Оценим результат обучения нейронной сети. Метод evaluate() возвращает значения функции потерь и метрики для обученной модели:

1. scores = model.evaluate(X, Y)
2. print("\n%s: %.2f%%" % (model.metrics\_names[1], scores[1]\*100))

Последняя строчка в форматированном виде выводит точность прогноза по нашей модели для заданной метрики accuracy (точность):

1. acc: 87.89%

Наконец, полный код нейронной сети с комментариями (после значка #) приведен ниже:

* 1. from keras.models import Sequential
  2. from keras.layers import Dense
  3. import numpy
  4. # задаем для воспроизводимости результатов
  5. numpy.random.seed(2)
  6. # загружаем датасет, соответствующий последним пяти годам до определение диагноза
  7. dataset = numpy.loadtxt("prima-indians-diabetes.csv", delimiter=",")
  8. # разбиваем датасет на матрицу параметров (X) и вектор целевой переменной (Y)
  9. X, Y = dataset[:,0:8], dataset[:,8]
  10. # создаем модели, добавляем слои один за другим
  11. model = Sequential()
  12. model.add(Dense(12, input\_dim=8, activation='relu')) # входной слой требует задать input\_dim
  13. model.add(Dense(15, activation='relu'))
  14. model.add(Dense(8, activation='relu'))
  15. model.add(Dense(10, activation='relu'))
  16. model.add(Dense(1, activation='sigmoid')) # сигмоида вместо relu для определения вероятности
  17. # компилируем модель, используем градиентный спуск adam
  18. model.compile(loss="binary\_crossentropy", optimizer="adam", metrics=['accuracy'])
  19. # обучаем нейронную сеть
  20. model.fit(X, Y, epochs = 1000, batch\_size=10)
  21. # оцениваем результат
  22. scores = model.evaluate(X, Y)
  23. print("\n%s: %.2f%%" % (model.metrics\_names[1], scores[1]\*100))

**Таблица со значениями, а также сам код Вы можете найти по ссылке https://github.com/jg-fisher/diabetesNeuralNetwork**

**Дорогие читатели!**

Надеемся, наш сборник расширил ваши знания в сфере программирования.

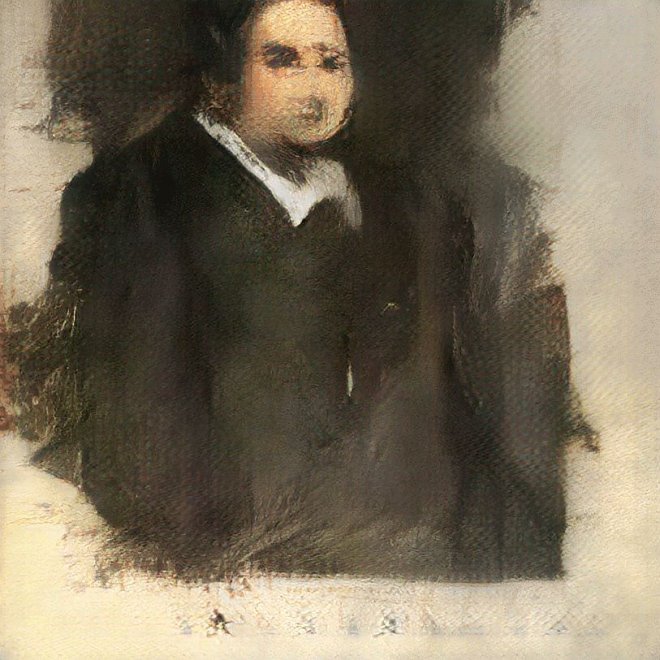
Предлагаем вашему вниманию список статей о необычных способах применения нейронных сетей.

1. Искусственный интеллект под руководством инженера из Google **самостоятельно** сделал первую кинокартину. Сам написал сюжет, сам написал все диалоги, сам выбрал сцены и определил выражения лиц актеров. Озвучка персонажей, музыка и монтаж – естественно, тоже за НС. Но получившаяся черно-белая научная фантастика отнюдь не кажется милой.

Директор фильма, нейронная сеть «Бенджамин», сделал короткометражку за 48 часов, используя тысячи часов старых фильмов, сотни тысяч

новелл и лица реальных актеров (с их разрешения). Жанр фильма sci-fi, но его с легкостью можно менять: достаточно «скормить» AI романтические комедии, хорроры а не sci-fi и нуар – и вот у вас на выходе получится совершенно другой продукт для своей целевой аудитории. Всего за 48 часов.

**Кадр из фильма. Компьютер сам подбирает эмоции героям, поэтому получаются такие странные лица**



**«Портрет Эдмонд Беллами»**

1. На престижном лондонском аукционе Christie's официально продана первая художественная картина, написанная нейронной сетью. Группа французских креативщиков под названием «Obvious» показала пример, как при помощи программного кода заработать солидные деньги с помощью нейронных сетей.

Дизайнеры рутинно «скармливали» нейросети примеры реальных картин из Wiki Commons, добиваясь, чтобы она начала рисовать нечто схожее.

Итоговый вариант специалисты аукционного дома согласились признать «картиной», а накопленный интерес помог продать этот рисунок за почти полмиллиона долларов. Картина носит название

«Портрет Эдмонд Беллами». Размеры картины, выполненной в виде печати на холсте, составляют 70 на 70 сантиметров

1. Искусственный интеллект научился создавать фотографии **несуществующих** людей в высоком разрешении. Любые морщинки, родимые пятна, с трудом заметные складки и ямочки — ничего не остается без внимания НС. И от настоящих людей не отличить.

Получить сходство с людьми на таком высоком уровне удалось благодаря разделению нейросети на две части, где одна ее часть рисовала портреты людей, основываясь на библиотеке, содержащих большое количество фотографий, что и позволило комбинировать образы для создания. Другая часть системы проверяла успех работы первой, контролируя весь процесс.

**Пример фотографии, созданных НС**

1. Нейросети также обучаемы к различным играм. Уже давно ИИ победил мирового чемпиона по игре в шахматы. Сейчас нейронная сеть побеждает суперкомпьютер в этой же настольной игре. А что насчет компьютерных игр? На этот вопрос ответ нашелся у польских исследователей, создавших платформу [VizDoom](https://arxiv.org/pdf/1605.02097.pdf) для игры в Doom (Doom - культовая компьютерная игра, созданная в 1993 году). В самом начале обучения у бота нет никаких знаний о том, что в принципе происходит в игре, — он не знает, что там есть стены, оружие и монстры, которых надо истреблять. Он начинает случайно двигаться и в какой-то момент натыкается на патроны

**Doom, 1993 год**

для оружия или аптечку; тут он понимает, что произошло нечто хорошее. В идеале постепенно он понимает, что нужно находить патроны как можно чаще, и начинает делать это целенаправленно – все как у настоящих людей, которые впервые вошли в игру и не знают, что в ней делать. ИИ использует только поток визуальной информации, считываемой с экрана монитора и выполняет параллельно две задачи. Первая: перемещение по карте, сбор предметов и оружия, анализ интерфейса. Вторая: боевые действия (прицеливание, уворот от вражеских снарядов, выбор наиболее подходящего оружия, изучение поведения противников). На обучение всем премудростям управления с видеоигрой у ИИ ушла всего одна неделя.

1. Нейронная сеть фиксирует нарушения ПДД. И эта система заработает **в Москве** уже **в конце года**, после 25 декабря. Возможность комплексов фотовидеофиксации автоматически сверять соответствие марки и номера автомобиля позволит решить сразу две важные задачи. Во-первых, будут исключены ситуации, когда камера неверно распознает номерной знак и штраф получает не тот, кто совершил нарушение. Во-вторых, легче будет вычислить мошеннические действия по подмене номеров", - сказал собеседник агентства.
2. Ученые из Нью-Йоркского университета научили нейронную сеть имитировать отпечатки пальцев, чтобы использовать их при взломе биометрических систем. В своей работе система программа использует две особенности систем идентификации по отпечаткам пальцев. Одна из них заключается в том, что из соображений эргономики большинство сканеров считывает не отпечаток пальца целиком, а любую его часть, которая первой коснётся регистрирующей поверхности. В такой системе сравнивается не весь отпечаток с образцом, а части отпечатка и соответствующие части образца. Это означает, что злоумышленник, чтобы обмануть систему, должен подобрать лишь одну десятую или сотую часть полного отпечатка. Вторая особенность состоит в том, что некоторые элементы папиллярного узора (рельефные линии на пальцах) являются более распространёнными, чем другие. В связи с этим фальшивый отпечаток, который содержит много часто встречающихся элементов, сработает с большей вероятностью.
3. Американский профессор создал цифрового композитора, который **самостоятельно** сочиняет оригинальную современную классическую музыку. Имя программы - Emily Howell. Ее первый альбом был выпущен в феврале 2009 года под названием **From Darkness, Light**, второй альбом **Breathless** был выпущен в декабре 2012 года.

**Есть одноименный YouTube канал со всеми композициями, написанными Emily** **Howell**

Плейлист с музыкой <https://www.youtube.com/playlist?list=PLJzS9_7dvWfaeOTh_36NKjaFOkWmSGptZ>

Список литературы

1. Кольцов С.Н. Нейронные сети <https://linis.hse.ru/data/2014/09/02/1313428617/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%207.pdf>
2. [Арнис](https://habr.com/users/Arnis71/) - Android разработчик Нейронные сети для начинающих.

<https://habr.com/post/312450/>

1. Владислав Режепа Простыми словами о сложном: что такое нейронные сети?

<https://gagadget.com/another/27575-prostyimi-slovami-o-slozhnom-chto-takoe-nejronnyie-seti/>

1. Функции активации в нейронных сетях

<http://www.aiportal.ru/articles/neural-networks/activation-function.html>

1. Как работает нейронная сеть

<https://ideas-garden.com/blog/poiskovye-sistemy/kak-rabotaet-nejronnaya-set/>

1. GitHub <https://github.com/jg-fisher/diabetesNeuralNetwork>
2. Habr https://habr.com/company/pochtoy/blog/413867/
3. Техкульт <https://www.techcult.ru/technology/5924-kartina-sozdannaya-nejronnoj-setyu>
4. [ВЕРСИЯ - ИНФО](https://versiya.info/) <https://versiya.info/tehnika-i-tehnologii/96076>
5. Красная весна, Скопина Ольга <https://rossaprimavera.ru/article/729591e2>
6. Искусственный интеллект научили играть в Doom «Медуза»

https://meduza.io/feature/2016/09/23/iskusstvennyy-intellekt-nauchili-igrat-v-doom

1. Тасс <https://tass.ru/obschestvo/5835510>
2. Оружие России <http://www.arms-expo.ru/news/tekhnologii_bezopasnosti/obman_sistemy_identifikatsii_po_otpechatkam_paltsev_s_pomoshchyu_neyronnykh_setey/>
3. The Times <https://www.thetimes.co.uk/article/emily-howell-the-virtual-composer-making-waves-in-the-computer-world-xcfbhldvl3p>
4. [Laewerwyn](https://laewerwyn.livejournal.com/)<https://laewerwyn.livejournal.com/9025.html>
5. Курс по анонимности и безопасности в сети <https://book.cyberyozh.com/ru/virtualnaya-mashina-i-virtualnaya-operatsionnaya-sistema/>
6. Созыкин Андрей Владимирович Установка Keras с TensorFlow в Anaconda <https://www.asozykin.ru/deep_learning/2017/09/07/Keras-Installation-TensorFlow.html>
7. Лучшие виртуальные машины для windows: ставьте, если очень хочется посмотреть на другие ос

<https://smartbobr.ru/programmy/virtualnye-mashiny-dlya-windows/>