**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Старший преподаватель факультета компьютерных наук департамента Программной инженерии  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В.Пантюхин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2017 г. |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | RU.17701729.502840-01 81 01-1 | | **Программа распознавания рукописных букв латинского алфавита на основе нейронной сети**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.502840-01 81 01-1-ЛУ**  Исполнитель  Студент группы 162 ПИ  \_\_\_\_\_\_\_\_\_\_\_/Кузнецов Д.С./  «\_\_\_»\_\_\_\_\_\_\_\_\_\_2017 г.  **2017** | |  |

УТВЕРЖДЕН

RU.17701729.502840-01 81 01-1-ЛУ

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Подп. и дата |  | | Инв. № дубл. |  | | Взам. Инв. № |  | | Подп. и дата |  | | Инв. № подл. | RU.17701729.502840-01 81 01-1 | | **Программа распознавания рукописных букв латинского алфавита на основе нейронной сети**  **Пояснительная записка**  **RU.17701729.502840-01 81 01-1**  **Листов 20**  **2017** |  |

**Содержание**

[1. ВВЕДЕНИЕ 3](#_Toc482820103)

[1.1. Наименование программы 3](#_Toc482820104)

[1.2. Документы на основании которых ведется разработка 3](#_Toc482820105)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc482820106)

[2.1. Функциональное назначение 4](#_Toc482820107)

[2.2. Эксплуатационное назначение 4](#_Toc482820108)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5](#_Toc482820109)

[3.1. Постановка задачи на разработку программы 5](#_Toc482820110)

[3.2. Описание алгоритмов и функционирования программы 5](#_Toc482820111)

[3.2.1. Создание пользовательского изображения 5](#_Toc482820112)

[3.2.2. Структура искусственной нейронной сети 6](#_Toc482820113)

[3.2.3. Алгоритм обучения сети 6](#_Toc482820114)

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 8](#_Toc482820115)

[4.1. Ориентировочная экономическая потребность 8](#_Toc482820116)

[4.2. Предполагаемая потребность 8](#_Toc482820117)

[5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 9](#_Toc482820118)

[Приложение 1 10](#_Toc482820119)

[Приложение 2 14](#_Toc482820120)

[Описание и функциональное назначение классов 14](#_Toc482820121)

[Приложение 3 15](#_Toc482820122)

[Описание и функциональное назначение полей, методов и свойств 15](#_Toc482820123)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 20](#_Toc482820124)

# ВВЕДЕНИЕ

## Наименование программы

«Программа распознавания рукописных букв латинского алфавита на основе нейронной сети».

## Документы на основании которых ведется разработка

Приказ №2.3-0.2/0812-01 Национального исследовательского университета «Высшая школа экономики» от 08.12.2016.

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

* 1. **Функциональное назначение**

Программа предназначена для создания изображений латинских букв и последующего распознавания латинских букв от A до F на созданных изображениях.

* 1. **Эксплуатационное назначение**

Программа предназначена для демонстрации работы алгоритмов распознавания образов латинских букв от A до F на изображении.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

## Постановка задачи на разработку программы

Разрабатываемая программа должна предоставлять пользователю следующие возможности:

1. Запуск алгоритма распознавания с помощью графического интерфейса
2. Создание пользовательских изображений в графическом интерфейсе
3. Создание нового изображения безе перезагрузки интерфейса
4. Получение результата распознавания образов латинских букв от A до F на созданном изображении в графическом интерфейсе
5. Доступ к возможностям предобученной нейронной сети для распознавания латинских букв от A до F
6. Возможность обучать ИНС на собственных данных, корректируя параметры обучения вне графического интерфейса (для опытных пользователей)
7. Возможность конвертировать изображения в вектор признаков вне графического интерфейса

## Описание алгоритмов и функционирования программы

### Создание пользовательского изображения

За создание изображения в графическом интерфейсе отвечает элемент «pictureBox» в левой части формы.

Описанный элемент имеет разрешение 280х280 пикселов, но позволяет получать изображения с разрешением 28х28 пикселов для распознавания сети. Такой результат достигнут следующим образом: при инициализации элементов форм элемент pictureBox получает значение изображения 28х28 белых пикселов. Если оставить результат без изменений, работать придется только с левым верхним углом 28х28 пикселов элемента pictureBox, поэтому изображение было растянуто по всему элементу pictureBox.

Работа по созданию изображения начинается с обработки событий перемещения указателя мыши по элементу. Зажав левую клавишу мыши, пользователь двигает указатель и рисует контур буквы. При зажатии левой кнопки мыши (ЛКМ далее) вызывается и обрабатывается событие MouseDown, связанное с pictureBox. Обработчик события получает аргументы события и сохраняет начальную позицию указателя. Далее при каждом смещении указателя пользователем вызывается событие MouseMove и, соответственно, обработчик, который, используя новые координаты указателя, создает на pictureBox линию, соединяющую старые координаты с текущими, и сохраняет текущие координаты для продолжения алгоритма. Таким образом, будет получаться ломанная линия. Однако событие изменения положения указателя вызывается достаточно часто, чтобы для глаз пользователя контур выглядит гладким.

Важно отметить - все сохраненные координаты указателя уменьшаются в 10 раз, так они переводятся из координат pictureBox в координаты самого изображения, которое хранится в элементе рисования. Само изображение, напомним, просто растягивается по pictureBox, но активными пикселами все равно остаются пикселы левого верхнего угла 28х28.

Очистка холста – это присваивание pictureBox изображения 28х28 белых пикселов.

### Структура искусственной нейронной сети

Для распознавания образов латинских букв была использована сверточная нейронная сеть [Приложение 1] с двумя полносвязными слоями.

Первые 4 слоя сети – это две группы чередующихся сверточного и субдискретизирующего слоев.

Первый сверточный слой представляет собой 32 фильтра 5х5х3, где 3 – число каналов изображения (RGB)

Второй сверточный слой представляет собой 64 фильтра 5x5x3

Первый и второй субдискретизирующие слои представляют собой тензор 2х2.

После каждого сверточного слоя использовалась ReLu функция

Выходом сверточной части нейронной сети является матрица размерности <количество обучающих примеров>x7\*7\*64 (64 карты признаков 7х7 пикселов).

Данный выход подается на вход полносвязной нейронной сети.

Первый скрытый слой состоит из 1024 нейронов, в которых используется обычная сумматорная функция и за функцию активации взята ReLu функция.

Второй скрытый слой состои из 6 нейронов, каждый из которых отвечает за свой распознаваемый класс изображений (латинские буквы от A до F). К каждому нейрону применяется нормализованная экспоненциальная функция. Значение каждого выходного нейрона – это и есть убежденность ИНС нахождения на изображении соответствующей нейрону букве.

### Алгоритм обучения сети

Не умоляя общности, рассмотрим процесс обучения с одним единственным перераспределением выборки входных изображений.

На первом этапе работы алгоритма все веса сети инициализируются достаточно малыми случайными величинами. Далее, если обучение происходит с тестированием обученности ИНС (testing==True), выбирается случайно 10% изображений из общей выборки и помечаются как тестовые данные. Остальное подается для обучения.

Обучающее множество условно разбивается на пакеты данных. Размер одного пакета определяется параметром обучения (batch\_size). По умолчанию пакет состоит из 15 векторов изображений. Пакет представляет собой матрицу 15х2352, где 2352 = 28\*28\*3. На каждом шаге алгоритма (эпохе обучения) берется новый, еще не использованный пакет и подается на вход сети. Пройдя прямое распространение (процесс описан в приложении 1), матрица порождает матрицу выходных значений, имеющую размерность (15, 6), где 6 – кол-во распознаваемых сетью образов. Для каждого изображения существует вектор (1, 6), являющийся вектором правильного ответа. Вектор состоит из одной единицы и остальных нулей. Единица стоит на месте соответствующей букве, изображенной на распознаваемой картинке.

Для оценки точности предсказания используется перекрестная энтропия [Приложение 1]. Чем больше значение функционала, тем дальше ответ нейронной сети от верного.

Далее, используя функционал ошибок, Adam оптимизатор (описан в приложении 1) корректирует веса сети.

Обучение выполняется, пока алгоритм не выполнит определенное параметром обучение число эпох. По завершении работы алгоритм тестирует сеть на тестовом множестве. Данные этого множества изображений не использовались в обучении, поэтому они могут являться показателем обученности сети.

Чтобы избежать случая, когда алгоритм распределил очень удачные или легкие для распознавания примеры в тестовую выборку, используется несколько шагов полного обучения сети. На каждом шаге данные заново перераспределяются. Количество таких шагов определяется параметром N\_ATTEMPTIONS. После завершения всех шагов алгоритм усредняет точность предсказаний на тестовых выборках по всем шагам и выдает более объективную, усредненную характеристику работы сети.

# ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

* 1. **Ориентировочная экономическая потребность**

Программа разработана в учебных целях. Она демонстрирует работу сверточных нейронных сетей в задаче распознавания образов, дает возможность начать знакомство с данным типом сетей. К проекту прилагается размеченные и обработанные входные данные для обучения: более тысячи векторов пикселов изображений латинских букв от A до F.

* 1. **Предполагаемая потребность**

На данный момент существует множество примеров решения задачи классификации образов, но немногие предоставляют возможность удобно и быстро создавать пользовательские изображения и подавать их на вход нейронной сети для проверки качества ее работы. Этим и отличается этот продукт.

* 1. **Приемущества разработки по сравнению с существующими аналогами**

1. Демонстрация работы программы, используя удобный графический интерфейс
2. Обучающая выборка, прикрепленная к продукту

# ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. Статья «Методы оптимизации нейронных сетей», Павел Садовников [Электронный ресурс]//  
   URL: <https://habrahabr.ru/post/318970/> (Дата обращения: 09.05.2017, режим доступа: свободный).
10. Статья «Convolutional Neural Networks for Visual Recognition», Stanford University [Электронный ресурс]//

URL: <http://cs231n.github.io/convolutional-networks/> (Дата обращения: 09.05.2017, режим доступа: свободный)

1. Веб-страница «Tensorflow API r1.1 Python», Tensorflow

[Электронный ресурс]//

URL: <https://www.tensorflow.org/api_docs/python/> (Дата обращения: 15.05.2017, режим доступа: свободный).

1. Веб-ресурс «Информационный центр для разработчиков», Microsoft

[Электронный ресурс]//

URL: <https://msdn.microsoft.com/ru-ru/default.aspx> (Дата обращения: 15.05.2017, режим доступа: свободный).

**Приложение 1**

**Применяемые алгоритмы и термины**

**Перекрестная энтропия**

- функционал ошибки, используемый в программе для оценки точности работы алгоритма:

, где *y* – константа, равная правильному выходному значению сети для исходных данных; – выход нейронной сети; – iый параметр сети

**Оптимизатор Adam**

Оптимизационный алгоритм, сочетающий в себе идеи Стохастического градиентного спуска с моментом и RMSProb (Root Mean Square Propagation).

**Задача:** найти такие параметры сети, при которых функционал ошибки будет принимать наименьшее возможное значение, т.е. найти глобальный минимум функционала.

С этой задачей справляется алгоритм Стохастического градиентного спуска (SGD далее), заключающийся в изменении на каждом шаге работы алгоритма обучения параметров сети следующим образом:

, где – iый параметр сети; -константа сети, отвечающая за скорость обучения; – градиент J по параметру i сети

Приведенный алгоритм имеет ряд недостатков:

При некоторых начальных значениях алгоритм сходится в локальном минимуме или происходит паралич сети, т.е. параметры перестают меняться, но глобальный минимум не достигнут.

Некоторые параметры, отвечающие за редкие признаки во входных данных, обновляются редко в отличие от часто встречаемых.

Сложности в подборе скорости обучения, которая влияет на сходимость алгоритма.

Существуют модификации приведенного алгоритма, например, SGD с моментом.

Идея заключается в накоплении изменений параметров сети. Накопленный результат будет влиять на новое изменение параметра так же как и градиент функционала:

*-*коэффициент (0, 1), отвечающий за влияние накопленного результата на изменение параметра. Чем ближе коэффициент к нулю тем больше алгоритм схож с обычным SGD.

Такая модификация позволяет избегать аномальных неровностей на поверхности функционала ошибки.

Существует другая модификация SGD, которая помогает избавиться от проблемы обновления редких параметров сети. Это RMSProp

Модификация заключается в накоплении квадрата градиента функционала по параметрам сети:

, где -градиент на шаге алгоритма t; -накопленный градиент на шаге алгоритма t

Тогда обновление параметров будет следующим:

*–* малая величина, добавленная, чтобы избежать деление на ноль.

Теперь веса, которые часто обновляются, будут иметь большее накопленное значение градиента, потому они будут обновляться с медленнее.

Оптимизатор Adam включает в себя идеи двух ранее приведенных алгоритмов.

Пусть – накопление градиентов, – накопление квадратов градиентов на tом шаге алгоритма. Тогда

Важное отличие Adam от двух других алгоритмов в начальных значениях накопителей. Если взять достаточно малые величины, то даже при больших значениях β m и v будут долго накапливаться, а если пытаться подобрать какие-то другие начальные значения, то появляются еще 2 дополнительных гиперпараметра. По этой причине авторы Adam предложили искусственно увеличивать m и v на первых шагах алгоритма:

Тогда конечная формула изменения параметров будет выглядеть так:

**Сверточные нейронные сети**



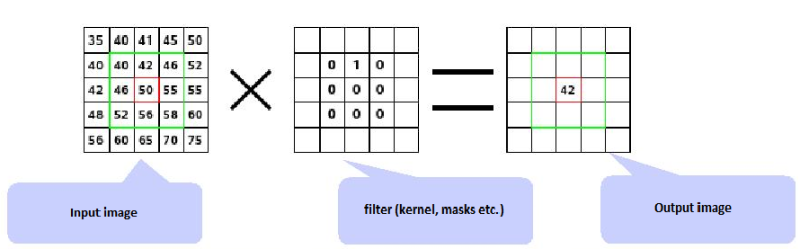
*Рисунок 1 – Архитектура сверточной нейронной сети*

Сверточные сети состоят из двух типов слоев: сверточные и субдискретизирующие.

Сверточный слой – это набор фильтров одинаковой размерности. Например 32 фильтра размерности 5х5х3.

Субдискретизирующий слой – это рамка (матрица), выбранной разработчиком размерности. Обычно 2х2.

Сверточный слой получет на вход тензор размерности соответствующей фильтрам. Например, если фильтр размерности NxMxK, то тензор должен быть размерности QxVxK. Сверточный слой применяет каждый фильтр к входному тензору для каждого участка изображения (применить фильтр значит скалярно умножить фильтр на участок тензора).



*Рисунок 2 – Пример применения фильтра*

Субдискретизирующий слой так же применяет рамку для входного тензора (применить рамку значит применить функцию, описывающую слой к участку тензора, который покрывает рамка)

Чаще всего применяется функция max(). Иными словами выбирается максимальное значение на каждом участке тензора



*Рисунок 3 – Пример субдискретизирующего слоя с функцией max()*

Обычно в сверточных нейронных сетях используют не только вышеописанные слои, но и полносвязнные слои. На них подаются значения после прохождения сверточных и субдискретизирующих слоев.

**Приложение 2**

## Описание и функциональное назначение классов

**Графический интерфейс**

|  |  |
| --- | --- |
| Класс | Назначение |
| Form1.cs | Окно графического интерфейса |

**Программа обучения и распознавания ИНС**

|  |  |
| --- | --- |
| Класс | Назначение |
| Main.py | Обучение и распознавание ИНС |

# Приложение 3

## Описание и функциональное назначение полей, методов и свойств

**Графический интерфейс**

|  |  |  |  |
| --- | --- | --- | --- |
| Класс Form1 | | | |
| **Методы** | | | |
| Имя | Модификатор доступа | Аргументы | Описание |
| pictureBox1\_MouseDown | private | Object, MouseEventArgs | Обработчик события для рисования изображений. При зажатии ЛКМ сохраняет позицию указателя |
| button1\_Click | private | object, EventArgs | Метод, очищающий холст рисования |
| button2\_Click | private | object, EventArgs | Метод, вызывающий скрипт распознавания и выводящий результат в форму |
| GetEncoder | private | ImageFormat | Метод, возвращающий описание кодека для заданного формата изображений. |
| **Поля** | | | |
| Имя | Модификатор доступа | Тип | Описание |
| p1 | private | Point | Точка, имеющая координаты последнего положения указателя |
| path\_to\_image | private | string | Строка, указывающая куда сохранять созданные изображения |
| path\_to\_tf\_server | private | string | Строка, указывающая адрес локального сервера |
| wc | Private | WebClient | Поле, позволяющее обращаться к локальному серверу по адресу |
| jpgEncoder | Private | ImageCodecInfor | Поле, хранящее информацию о кодеке изображений |
| eParameter | Private | EncoderParameter | Параметр конвертирования изображений |
| enc | Private | Encoder | Параметр качества сжатия |
| eParameters | Private | EncoderParameters | Список параметров конвертирования изображений |

**Обучение и распознавание ИНС**

|  |  |  |  |
| --- | --- | --- | --- |
| Класс Main | | | |
| **Методы** | | | |
| Имя | Модификатор доступа | Аргументы | Описание |
| params\_init | - | np.array shape, string name | Инициализирует переменную очень малыми случайными величины указанной размерности |
| conv2D\_init | - | tf.Tensor x, np.array kernel\_shape,int output channels, string name, function activation | Метод, создающий условный сверточный нейронный слой, создавая и инициализируя его параметры |
| pool2x2 | - | tf.Tensor x, string name | Метод, создающий условный субдискретизирующий слой |
| fcl\_init | - | tf.Tensor x, int in\_neurons, int out\_neurons, string name | Метод, создающий условный полносвязнный слой, создавая и инициализируя его параметры. |
| get\_one\_hot | - | string c | Метод, возвращающий кодировки символа латинского алфавита в One Hot |
| get\_letter | - | int i | Возвращает букву, соответствующую номеру в алфавите |
| Initialising\_LatinClassifierNet | - | bool restore, string path, string ckpt\_name | Инициализирует константы обучения и параметры сети |
| Learn | - | string name, tf.Tensor train\_set, tf.Tensor train\_set\_labels, int batch\_size, int N\_ATTEMPTIONS, int epochs, tf.Session sess, bool testing | Метод, выполняющий настройку параметров сети |
| start | - | string path | Метод, работающий в режиме сервера и запускающий процесс распознавания |
| **Поля** | | | |
| input\_data | - | numpy.array | Поле, хранящее векторы изображений |
| input\_label | - | numpy.array | Поле, хранящее правильные ответы для соответствующих изображений |
| N\_EXAMPLES | - | Int | Количество входных примеров изображений |
| default\_saver\_path | - | String | Путь сохранения параметров по умолчанию |
| Alphabet | - | String | Строка с распознаваемыми буквами латинского алфавита |
| sess | - | tf.Session | Поле, хранящее сессию по умолчанию |
| s | - | pandas.Series | Поле, хранящее латинские буквы в One Hot |
| x | - | Tf.placeholder | Поле, определяющее матрицу входных изображений |
| y | - | Tf.placeholder | Поле, определяющее матрицу правильных ответ для входных изображений |
| x\_image\_ | - | Tf.placholder | Поле, определяющее тензор входных изображений |
| conv1 | - | Tf.Relu | Поле, определяющее первый сверточный слой |
| pool1 | - | Tf.pool1 | Поле, определяющее первый субдискретизирующий слой |
| conv2 | - | Tf.Relu | Поле, определяющее второй сверточный слой |
| pool2 | - | Tf.pool2 | Поле, определяющее второй субдискретизирующий слой |
| vectorized\_pool2 | - | Tf.reshape | Поле, определяющее входной слой полносвязной сети |
| fcl1 | - | Tf.Relu | Поле, определяющее первый скрытый слой полносвязной сети |
| keep\_prob | - | Tf.placeholder | Поле, определяющее вероятность исключения нейронов из полносвязной сети |
| d\_fcl1 | - | Tf.dropout | Поле, определяющее первый скрытый слой с возможностью исключения мало активных нейроново |
| y\_hat | - | Tf.Relu | Поле, определяющее выходной слой полносвязной сети |
| cross\_entropy | - | Tf.softmax\_cross\_entropy | Поле, определяющее функционал ошибки |
| train\_step | - | Tf.train.AdamOptimizer | Поле, определяющее шаг обучения |
| correct\_prediction | - | Tf.equal | Поле, определяющее точность ответа сети на изображение |
| accuracy | - | Tf.reduce\_mean | Поле, определяющее точность ответа сети на изображение |
| Saver | - | Tf.train.Saver | Поле, предоставляющее возможность записывать параметры сети в файл и их восстанавливать |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |