УДК 519.681.5
DOI: 10.15827/0236-235X.139.362-373
Дата подачи статьи: 25.07.22
2022. Т. 35. № 3. С. 362–373

Применение технологии CUDA для обучения нейронной сети Кохонена

Д.С. Латыпова [1], *аспирант, dina.latypova23@gmail.com*
Д.Н. Тумаков [1], *к.ф.-м.н., доцент, dtumakov@kpfu.ru*

[1] *Казанский (Приволжский) федеральный университет, Институт вычислительной математики и информационных технологий, г. Казань, 420008, Россия*

Проведена кластеризация данных, содержащихся в обучающих выборках – базах данных MNIST и Fashion MNIST. Для кластеризации использована нейронная сеть Кохонена с евклидовой метрикой при оценке расстояний. Для каждой рукописной цифры (MNIST) и каждого типа предметов обихода (Fashion MNIST) определено оптимальное количество кластеров (не более 50).

Обучение нейронной сети распараллелено на графическом устройстве NVidia с использованием технологии CUDA. Для каждой цифры приведены результаты, иллюстрирующие сравнение времен работы процессора и графического процессора. Как для цифр, так и для типов предметов обихода сделан вывод о 17-кратном ускорении параллельных вычислений над последовательными на игровом ноутбуке начального уровня. Тестовые выборки из тех же баз данных использованы для проверки правильности построения кластеров. Как для последовательного, так и для параллельного обучения сделан вывод о том, что векторы из тестовой выборки принадлежат правильному кластеру с вероятностью более 90 % в случае рукописных цифр. Кроме того, для каждой цифры и каждого типа предметов обихода вычислены F-меры для оценки попадания объектов в свои кластеры.

Показано, что последовательная и параллельная кластеризации дают близкие результаты. Наилучшие значения F-меры получены для цифр 0 и 1 (F-среднее значение равно 0,974), в то время как худшее значение (0,903) получено для цифры 9. Для данных Fashion MNIST лучшее значение F-меры (0,96) получено для типа «Брюки», а худшее (0,34) – для типа «Рубашка». Несмотря на большие разбросы для значений F-метрик двух рассмотренных баз данных, отличия результатов кластеризации минимальны. Так, для MNIST максимальное отличие F-меры составляет порядка 0,01, а для Fashion MNIST – около 0,04.

Ключевые слова: *нейронная сеть Кохонена, кластеризация, распараллеливание, CUDA, MNIST.*

Распознавание образа – это автоматизированный процесс поиска точного соответствия и закономерностей данных, который тесно связан с машинным обучением. Распознавание изображений с использованием искусственного интеллекта широко применяется в повседневной жизни. Понимание рукописного ввода значительно облегчает взаимодействие между компьютером и человеком. Однако при распознавании изображений существует проблема с категоризацией данных – данные в неподходящей категории означают неверную информацию, что, в свою очередь, может привести к серьезным ошибкам.

Одним из решений этой проблемы является кластерный анализ данных. Распознавание образов выступает в качестве основного шага для обеспечения кластеризации, поскольку во время этого процесса анализируются структура и векторное значение каждого символа в наборе данных. Например, в работе [1] предложен алгоритм консенсусной кластеризации для набора рукописных цифр из БД MNIST. Так называемая надежная непрерывная кластеризация рассмотрена в [2]. В работе [3] предложен алгоритм для улучшения эффективности классификации при распознавании рукописных цифр на наборе данных MNIST.

Для кластеризации рукописных цифр часто применяются подходы c-means [4] и k-means [5, 6]. В работе [7] рукописные письма были сгруппированы с использованием нейронной сети Кохонена. Различные категории алгоритмов кластеризации рассмотрены в [8] – это кластеризации на основе разделов, плотности, сетки, а также иерархическая кластеризация. Одним из самых популярных алгоритмов кластеризации является k-means [9]. При использовании этого метода на большинстве данных достигается точность примерно 90 %. В случае больших размерностей также может быть применена кластеризация подпространства [10, 11].

Обучение нейронной сети на больших данных занимает довольно много времени, поэтому все чаще применяется распараллелива-

ние сетевого обучения с использованием технологии CUDA. Основная причина – единообразие операций, выполняемых во время обучения. В [12] технология CUDA использована для идентификации текста с помощью нейронных сетей, полученное ускорение составило 15 раз. В [13] технология CUDA применена для реализации самоорганизующихся карт на основе модели распределенных вычислений MapReduce. Во время работы ядра оптимизированы для доступа к общей памяти, а также ее эффективного использования. Было получено десятикратное ускорение. В [14] алгоритм Batch-SOM использован для кластеризации, а технология CUDA – для распараллеливания, выполненного на уровне обучающей выборки. В результате получено 11-кратное ускорение.

В [15] отмечено, что при реализации OpenCL снижается производительность по сравнению с технологией CUDA. Кроме того, при использовании различных комбинаций OpenCL и CUDA для двух разных видеокарт продемонстрирована возможность масштабирования реализации алгоритма SOM на разных графических устройствах. В [16] получено также ускорение алгоритма SOM на графических процессорах. В [17] рассмотрен алгоритм HPSOM, который был распараллелен с использованием технологий CUDA и MPI; показано, что с его помощью можно не только добиться хороших результатов с точки зрения производительности, но и оптимизировать затраты с точки зрения памяти.

В настоящей работе описана кластеризация рукописных цифр из БД MNIST [18] и типов предметов обихода из расширения MNIST – Fashion MNIST. Кластеризация выполнена с использованием нейронной сети Кохонена. При этом для каждого объекта количество кластеров, полученное предложенным алгоритмом, разное, но не превышает 50. Для оценки расстояния между изображениями цифр использована евклидова норма, хотя для оценки расстояний возможно использование и других метрик [15, 19]. Обучение сети Кохонена проведено на центральном и графическом процессорах с применением технологии CUDA.

Для каждой цифры приведены диаграммы, которые показывают время, затраченное на обучение. Проведено сравнение времени обучения на центральном и графическом процессорах. Получен средний коэффициент ускорения обучения нейронной сети с использованием технологии CUDA. Тестовая выборка позволила оценить точность кластеризации, вычислить F-меру для каждой цифры. Также кластеризация проведена для БД Fashion MNIST, которая, как и база MNIST, содержит обучающий и тестовый наборы соответственно из 60 000 и 10 000 примеров. Каждый пример представляет собой изображение в оттенках серого, связанное с меткой, которая берется из 10 классов. Получены оценки ускорения и точности кластеризации для обеих баз.

Материалы и методы

Целью кластеризации является разделение набора объектов на группы, называемые кластерами. Объекты группируются таким образом, чтобы в пределах одного кластера они были более похожи друг на друга, чем на объекты из других кластеров. Рассмотрим два набора данных: рукописные цифры и подборку изображений десяти типов предметов обихода.

Наборы данных MNIST и Fashion MNIST. MNIST – это БД рукописных цифр, которая содержит 60 000 изображений цифр в обучающей выборке и 10 000 изображений в тестовой. В таблице 1 приведена информация о том, сколько изображений каждой цифры содержится в обучающей и тестовой выборках. Каждое изображение из БД представлено в двух форматах: в виде метки и вектора значений пикселей, а также нормировано по размеру. Размер каждого изображения составляет 28 на 28 пикселей. При этом значение пикселя изменяется от 0 до 255, где 0 соответствует черному пикселю, а 255 белому.

Таблица 1

Количество изображений каждой цифры в обучающей и тестовой выборках из набора данных MNIST

Table 1

The number of each digit images in the training and test samples from the MNIST dataset

Цифра	Обучающая выборка	Тестовая выборка
0	5 923	980
1	6 742	1 135
2	5 958	1 032
3	6 131	1 010
4	5 842	982
5	5 421	892
6	5 918	958
7	6 265	1 028
8	5 851	974
9	5 949	1 009

БД MNIST и ее расширения (Fashion MNIST и другие) широко используются для тестирования различных подходов к распознаванию образов [20, 21], кластеризации [22] и других алгоритмов [23–25]. Многие алгоритмы проверяются на этой БД. Например, методы k-ближайшего соседа на основе MNIST дают ошибку 5 %, многослойные персептроны в зависимости от методов обучения и количества слоев – около 2–5 %, сверточные нейронные сети – чуть менее 1 %, а иерархические нейронные сети даже улучшают точность [26, 27].

Необходимо показать универсальность используемых нейронных сетей, то есть продемонстрировать способность предлагаемого алгоритма группировать не только рукописные цифры, но и другие изображения. Для этой цели выбран набор данных Fashion MNIST.

Набор модных списков содержит данные размером 28 на 28 пикселей и десять составляющих: футболка/топ, брюки, пуловер, платье, пальто, сандалии, рубашка, кроссовки, сумка, ботильоны. В обучающей выборке содержится 60 000 изображений, в тестовой – 10 000.

Нейронная сеть Кохонена. Эта сеть представляет собой неконтролируемый тип обучения и состоит из одного слоя настраиваемых весов. Веса нейронной сети изменяются таким образом, что векторы, принадлежащие одному и тому же кластеру, приводят в движение один и тот же выходной нейрон [28]. Архитектура нейронной сети Кохонена приведена на рисунке 1.

Выходы нейронной сети z_j вычисляются по формуле $z_j = \Sigma_i w_{ij} x_i$, где x_i – входные значения нейронной сети Кохонена. Изображения в наборе данных MNIST имеют размер 28 на 28.

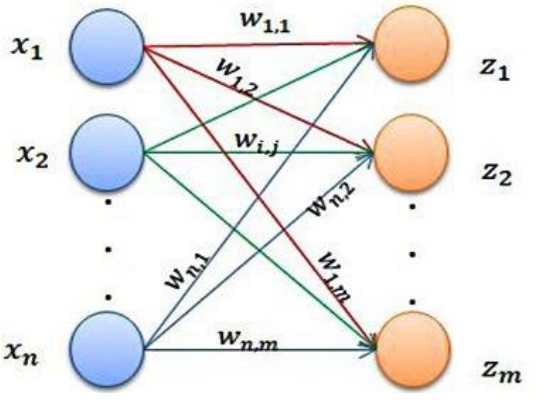

Рис. 1. Архитектура нейронной сети Кохонена

Fig. 1. The architecture of the Kohonen neural network

Следовательно, в нейронной сети Кохонена количество нейронов равно 784. Веса w_{ij} являются центрами кластеров и участвуют в обучении сети. Обучение нейронной сети Кохонена можно описать следующим алгоритмом.

Шаг 1. Нормируются входные векторы x^p.

Шаг 2. Из векторов обучающей выборки случайным образом выбираются значения весов w_{ij}. Это связано с тем, что в случае неравномерного распределения (если веса будут заполняться случайным образом) веса могут располагаться далеко от входных векторов и поэтому не принимать участия в обучении.

Шаг 3. Вычисляются расстояния между входными векторами x^p с координатами $x^p{}_i$ и весами w_{ij} по формуле $D_j = \left(\sum_{i=1}^{784} \left(x_i^p - w_{ij} \right)^2 \right)^{1/2}$.

Выбирается вектор-победитель – такой вектор w_j, у которого расстояние D_j до входного вектора x^p наименьшее.

Шаг 4. Изменяются координаты вектора-победителя, выбранного на предыдущем шаге, в соответствии с формулой $w_l = w_l + \theta(x^p - w_l)$, где θ – скорость обучения.

Шаг 5. Шаги 3 и 4 повторяются для всех векторов x^p.

Значение θ определяется по формуле $\theta = \alpha\theta$, где $\alpha < 1$. Если $\theta > \varepsilon$, то выполняется переход к шагу 3, и снова, проходя через все входные векторы, веса корректируются.

Технология CUDA. Процесс обучения нейронной сети очень длительный, для каждой цифры необходимо обучить отдельную нейронную сеть Кохонена. Для распараллеливания выберем технологию CUDA [29]. Логически графический процессор с поддержкой CUDA можно рассматривать как набор многоядерных процессоров. Основными вычислительными блоками таких видеочипов являются мультипроцессоры нескольких тысяч регистров общей памяти, текстурных и постоянных кэшей. Также стоит отметить одно из ключевых преимуществ технологии CUDA – отсутствие типов данных и принципов вычислений, характерных исключительно для обработки вершин и пикселей при построении кадра.

С помощью технологии CUDA могут быть достигнуты значительные ускорения (в несколько десятков раз) на обычных компьютерах при вычислении математических и физических задач [30–32]. Реализация нейронной сети с использованием комбинации CUDA и OpenMP и параллелизма центральных процессоров на уровне языка программирования и ап-

паратно-программного интерфейса создает еще один уровень прироста производительности.

Результаты

Кластеры рукописных цифр. Проблема с распознаванием рукописных цифр искусственным интеллектом заключается в том, что разные люди могут написать одно и то же разными способами. На примере БД MNIST можно видеть, что существует множество методов для записи определенной цифры. Каждая цифра имеет свои специфические особенности написания, например, цифру 1 можно записать с наклоном или без него. Каждая цифра имеет разное количество таких признаков. По этой причине количество кластеров для каждой цифры может быть разным.

Количество кластеров, на которые необходимо разделить входные изображения, неизвестно. Определим их оптимальное количество для каждой цифры. Для автоматической кластеризации будем использовать следующий алгоритм [33].

Шаг 1. На основе евклидовой метрики вычислим квадраты расстояний между всеми векторами в обучающей выборке. Из полученных значений создадим матрицу

$$d^2\left(x^i, x^j\right) = \left\|x^i - x^j\right\|^2 = \sum_{l=1}^{n} (x_l^i - x_l^j)^2 .$$

Шаг 2. Определим максимальное значение матрицы, которая была построена на предыдущем шаге. Этот элемент является максимальным расстоянием между всеми векторами ($\max d^2(x^i, x^j)$).

Шаг 3. Определим допустимое расстояние между векторами, расположенными в одном кластере, как фиксированный процент от максимального расстояния между векторами.

Шаг 4. Выберем произвольный i-й столбец матрицы (вектор x^i). Отметим все элементы столбца со значениями, меньшими допустимых, как элементы одного и того же кластера (строки под номером j). Проигнорируем i-й столбец, а также все j-е столбцы.

Шаг 5. Если матрица еще не пустая, то перейдем к шагу 4.

Таким образом, использовав автоматическую кластеризацию, для каждой цифры получим следующее количество кластеров: для 0 – 39, 1 – 49, 2 – 30, 3 – 47, 4 – 28, 5 – 33, 6 – 49, 7 – 29, 8 – 38, 9 – 28.

В таблице 2 показано оптимальное количество кластеров для БД Fashion MNIST. Легко заметить, что типы «Рубашка» и особенно «Сумка» содержат мало кластеров. Это связано с простотой изображений. Все изображения типа объектов «Сумка» обрабатываются как простые прямоугольники. Это дает один кластер, который представляет собой прямоугольник на плоскости.

Таблица 2

Оптимальное количество кластеров для каждого предмета обихода из Fashion MNIST

Table 2

An opptimal number of clusters for each Fashion MNIST

Тип предмета обихода	Количество кластеров
Футболка/Топ	25
Брюки	37
Свитер	29
Платье	37
Пальто	44
Сандалии	46
Рубашка	13
Кроссовки	49
Сумка	1
Туфли	40

После определения оптимального количества кластеров для каждой цифры используем нейронную сеть Кохонена для построения самих кластеров.

Реализация нейронной сети Кохонена. Оптимальные параметры для обучения нейронной сети Кохонена получены экспериментально: $\alpha = 0{,}96$, $\varepsilon = 0{,}02$, $\theta = 0{,}6$. Ниже представлены основные этапы обучения этой сети, которые являются самыми длительными по времени.

Первый шаг влияет на определение расстояния между весами и вектором из обучающей выборки. Представим функцию для вычисления расстояния и выбора минимального расстояния на процессоре ($m = 39$):

```
void DistanceSeq(int clustersCount, int
numOfVector, float* x[], float* w[], int*
clst){
float minimum = FLOAT_MAX, tmp;
int num = 0;
for (int i = 0; i < clustersCount; i++){
      tmp = 0.0f;
      for (int j = 0; j < N; j++){
         float r = w[i][j] - x[numOfVec-
tor][j];
         tmp += r * r;
      }
      if (tmp < minimum){
         min = tmp;
```

```
            num = i;
        }
}
*clst = num;
}
```

Эта часть определяет расстояние от вектора из обучающей выборки x[numOfVector] до каждого из кластеров *w*[*i*]. Количество кластеров обозначено как clustersCount, и это значение варьируется от 28 до 49 в зависимости от цифры, для которой выполняется обучение. Так как проводится перебор всех пикселей изображения по всем кластерам, сложность этой функции равна 0 (количество кластеров * *N*), где *N* равно 784.

Второй шаг влияет на прохождение всех векторов и изменяет веса нейронов. В обучающей части второй этап включает передачу всех векторов и изменение весов нейронов. Последовательный алгоритм обучения нейронной сети Кохонена выглядит следующим образом:

```
void Training_Sequential(int clst, int vec-
torsCount, float* w[], float* x[])
{
float h = 0.6f;
float rt = 0.96f;
float minimumh = 0.002f;
int distMin;
do
{
      for (int k = 0; k < vectorsCount;
k++){
          DistanceSeq(clst, k, x, w,
&distMin);
          for (int i = 0; i < N; i++)
             w[distMin][i] =
w[distMin][i]+h*(x[k][i]-w[distMin][i]);
      }
      h *= rt;
} while (h > minh);
}
```

В части обучения прохождение осуществляется по всем векторам из обучающей выборки. Переменная vectorsCount обозначает количество векторов в выборке, а distMin содержит номер кластера, к которому относится текущее векторное изображение.

Реализация алгоритма обучения нейронной сети Кохонена с помощью технологии CUDA. Алгоритм обучения нейронной сети Кохонена состоит из двух длительных этапов, которые были описаны выше. В обучающих образцах содержится около 6 000 изображений, размерность входного вектора равна 784, а количество кластеров для каждой цифры варьируется от 28 до 49. Как правило, обучение сети занимает всего несколько десятков итераций. В рассматриваемом случае (выбранные параметры указаны в предыдущем разделе) алгоритм обучения содержит 84 итерации. Существуют следующие уровни распараллеливания [34]: фазы обучения, обучающей выборки, слоя, нейрона и весов.

Выбор уровня распараллеливания зависит от количества нейронов, вычислительных узлов и особенностей компьютерной архитектуры искусственной нейронной сети. В настоящей работе распараллеливание осуществлено на уровне обучающей выборки, что связано с большим количеством входных векторов. Для распараллеливания выберем необходимое количество блоков (чтобы покрыть количество входных векторов), причем каждый блок содержит 256 потоков. Каждый поток отвечает за один вектор в обучающей выборке. Таким образом, убран цикл по всем изображениям, что значительно ускорит работу алгоритма. Функция Distance вычисляет расстояние между векторами на графическом процессоре. Представим ее код:

```
__device__ void Distance(int vector, float*
x[], float* w[], int* clst){
float mn = FLOAT_MAX, tmp;
int numb = 0;
for (int i = 0; i < CLUSTERCOUNT; i++){
   tmp = 0.0;
   for (int j = 0; j < N; j++){
      float r = w[i][j] - x[vector][j];
      tmp += r * r;
   }
   if (tmp < mn){
      mn = tmp;
      numb = i;
   }
}
*clst = numb;
}
```

Представим код функции графического процессора, которая обучает сеть Кохонена:

```
__global__ void TrainingPar(float* w[],
float* x[], float h){
int thread = blockIdx.x * blockDim.x +
threadIdx.x;
int distMin;
if (thread < vectorsCount)
   do{
      Distance(thread, patternArr, w,
&distMin);
      for (int i = 0; i < N; i++){
         w[distMin][i] += h * (x[thread][i]
- w[distMin][i]);
      }
      __syncthreads();
      h *= Rate;
   } while (h > minH);
}
```

Здесь Rate означает параметр θ. Обратим внимание, что в цикле *for* изменение веса *w* может выполняться одновременно несколькими потоками. Однако из-за большого размера выборки и, следовательно, частых изменений весов этот «конфликт» записи новых значений

в массив не является критичным и практически не влияет на результат.

Компьютер, на котором выполнены вычисления (в среде Visual Studio 2017), имеет следующие характеристики: операционная система – Windows 10 Pro x64, модель GPU – NVidia GeForce GTX 1050TI, объем GDDR – 4 GB, модель CPU – Intel Core i7-7700HQ 2.80 GHz, объем RAM – 24 GB.

На рисунке 2 представлена гистограмма, демонстрирующая время обучения сети Кохонена на CPU и GPU для каждой цифры отдельно.

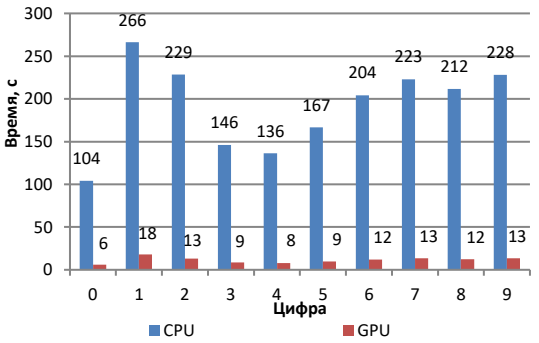

Рис. 2. Время обучения нейронной сети Кохонена на CPU и GPU

Fig. 2. Training time of the Kohonen neural network on CPU and GPU

На основе данных о времени выполнения алгоритма для каждой цифры можно рассчитать ускорение, полученное с использованием технологии CUDA. Коэффициент ускорения вычислен как отношение времени выполнения алгоритма на процессоре к времени выполнения алгоритма на графическом процессоре. Результаты представлены в таблице 3.

Таблица 3

Ускорение работы на графическом процессоре при кластеризации

Table 3

GPU acceleration with clustering

Цифра	Ускорение
0	17,2
1	14,8
2	17,4
3	16,9
4	17,5
5	17,5
6	17,2
7	16,5
8	17,2
9	17,0

Таким образом, на основе анализа результатов (рис. 2 и табл. 3) можно сделать вывод, что обучение нейронной сети Кохонена ускоряется в среднем в 16,9 раза. Примерно такое же ускорение достигается и для Fashion MNIST.

Проверим правильность построения кластеров на тестовом образце. Для тестового образца из БД рукописных цифр MNIST проанализируем процент векторов – изображений цифр, которые находятся в требуемом кластере.

Для этого нужно определить расстояние от каждого вектора цифр до всех полученных кластеров для каждой из цифр. Если вектор изображения находится ближе всего к кластеру, который относится к группе кластеров для того же рисунка, считаем, что вектор попал в нужный кластер.

На рисунке 3 представлены процентные доли получения изображений из тестовой выборки в кластеры их цифр. Также имеет смысл проверить, построены ли кластеры отдельно на CPU и GPU.

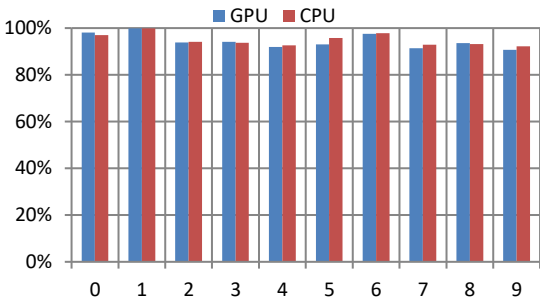

Рис. 3. Процент изображений цифр, которые попадают из тестовой выборки в свой кластер

Fig. 3. The percentage of images of numbers that fall from the test sample into their cluster

Сравнение точности определения типа предметов обихода на центральном и графическом процессорах показано на рисунке 4. Отметим, что в этом случае характеристики последовательного и параллельного алгоритмов очень близки. Как видно на рисунках 3 и 4, кластеры, построенные с помощью CPU и GPU, дают примерно одинаковый результат. Однако это не одно и то же. Кроме того, для анализа точности результатов кластеризации нейронной сети Кохонена вычислим F-меру (табл. 4).

Также вычислим F-меру для набора данных Fashion MNIST (табл. 5).

Класс «Рубашка» распознается очень плохо, в то время как классы «Брюки» и «Платье»,

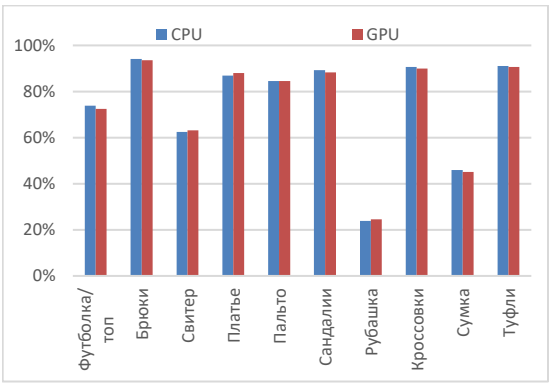

Рис. 4. Процент изображений преметов обихода, попадающих из тестируемого образца в свой кластер

Fig. 4. The percentage of images of clothes that fall from the tested sample into their cluster

а также классы, представляющие типы обуви, распознаются довольно хорошо. В среднем их распознавание нейронной сетью Кохонена составляет 74 %. Распознавание объектов из Fashion MNIST оказалось хуже, чем объектов из MNIST. Это связано с тем, что центры кластеров для этого набора оказались менее четкими, следовательно, при построении кластеров могут возникать ошибки.

Обсуждение

Основная проблема при распознавании образов заключается в том, что одни и те же цифры записываются по-разному. В результате либо значительно увеличивается количество кластеров, либо растет их размер. С другой стороны, существует проблема, связанная с тем, что разные цифры записываются одинаковым образом. В случае незначительного количества кластеров ожидается, что из-за их большого размера будут пересечения кластеров с разными цифрами [35].

Анализируя рисунок 3 и таблицу 5, можно сделать вывод, что нейронная сеть Кохонена может быть использована для распознавания шаблонов рукописных цифр, поскольку большой процент изображений цифр из тестовой выборки попадает в требуемый кластер. Отметим, что цифра 9 признана наихудшей, результат ее распознавания – чуть более 90 %. Цифра 1 распознается нейронной сетью Кохонена почти в 100 % случаев.

Алгоритм, реализованный на CUDA, показывает стабильные результаты кластеризации с большим количеством кластеров. Об этом свидетельствуют гистограммы на рисунке 4. Таким образом, для каждой из цифр 0, 1, 3, 6 и 8 количество кластеров варьируется примерно от 40 до 50 и процент цифр из тестовой выборки в ее кластере примерно одинаков. Для цифр 4, 5, 7 и 9 количество кластеров составляет около 30, и для них точность кластеризации для центрального процессора выше. Таким образом, увеличение количества кластеров дает улучшение результатов кластеризации на графическом процессоре.

Полученные кластеры могут быть использованы для распознавания изображений, например, в качестве этапа в ансамбле нейронных сетей или в иерархической нейронной сети. В этих случаях число кластеров может быть произвольным. В случае же использова-

Таблица 4

F-мера для кластеров, построенных на GPU и CPU на наборе данных MNIST

Table 4

F-measure for GPU and CPU clusters for the MNIST dataset

Цифра	Recall		Precision		F-means	
	GPU	CPU	GPU	CPU	GPU	CPU
0	0,97	0,97	0,98	0,97	0,97	0,97
1	0,95	0,96	0,99	0,99	0,97	0,98
2	0,96	0,97	0,94	0,94	0,95	0,95
3	0,93	0,96	0,94	0,94	0,94	0,95
4	0,94	0,94	0,92	0,93	0,93	0,93
5	0,95	0,94	0,93	0,96	0,94	0,95
6	0,96	0,97	0,98	0,98	0,97	0,97
7	0,95	0,95	0,91	0,93	0,93	0,94
8	0,94	0,95	0,94	0,93	0,94	0,94
9	0,89	0,89	0,90	0,92	0,90	0,91

Таблица 5

F-мера для кластеров, построенных на GPU и CPU на наборе данных Fashion MNIST

Table 5

F-measure for GPU and CPU clusters on the Fashion MNIST dataset

Тип предмета обихода	Recall		Precision		F-means	
	GPU	CPU	GPU	CPU	GPU	CPU
Футболка / Топ	0,73	0,75	0,72	0,74	0,72	0,74
Брюки	0,96	0,97	0,93	0,94	0,96	0,97
Свитер	0,66	0,65	0,63	0,62	0,64	0,63
Платье	0,78	0,74	0,91	0,87	0,84	0,80
Пальто	0,53	0,5	0,88	0,85	0,66	0,63
Сандалии	0,75	0,77	0,87	0,89	0,81	0,83
Рубашка	0,55	0,52	0,25	0,24	0,34	0,33
Кроссовки	0,80	0,83	0,88	0,91	0,83	0,86
Сумка	0,94	0,99	0,43	0,46	0,61	0,63
Туфли	0,89	0,9	0,90	0,91	0,89	0,9

ния нейронной сети Хопфилда существует ограничение на количество запоминаемых объектов [36] и для изображений MNIST оно ограничено 50.

Заключение

С использованием нейронной сети Кохонена проведена кластеризация рукописных цифр из БД MNIST и типов предметов обихода из БД Fashion MNIST. Для каждой цифры определено оптимальное количество кластеров. Сделан вывод, что изображения из тестовой выборки принадлежат правильному кластеру с вероятностью более 90 % для каждой цифры. Наилучшая кластеризация получена для цифр 0 и 1 (F-мера равна 0,97), в то время как цифра 9 была наихудшей кластеризованной – F-мера равна 0,903. Для данных Fashion MNIST лучшее значение F-меры получено для типа «Брюки» (значение равно 0,96), а худшее – для типа «Рубашка» (значение равно 0,34).

Алгоритм обучения нейронной сети Кохонена распараллелен на графическом процессоре с использованием технологии CUDA. Распараллеливание выполнено на уровне обучающей выборки. Для обеих БД алгоритм ускорен примерно в 17 раз. Несмотря на большой разброс значений F-метрик двух рассмотренных БД, отличия результатов кластеризации минимальны. Так, для MNIST максимальное отличие F-мер составляет порядка 0,01, а для Fashion MNIST – около 0,04. Таким образом, предложенный подход без потери точности кластеризации может быть использован для больших объемов данных.

Работа выполнена за счет средств Программы стратегического академического лидерства Казанского (Приволжского) федерального университета («ПРИОРИТЕТ-2030»).

Литература

1. Rexy M., Lavanya K. Handwritten digit recognition of MNIST data using consensus clustering. IJRTE, 2019, vol. 7, no. 6, pp. 1969–1973.

2. Nhery S., Ksantini R., Kaaniche M.B., Bouhoula A. A novel handwritten digits recognition method based on subclass low variances guided support vector machine. Proc. XIII Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications. VISAPP, 2018, vol. 4, pp. 28–36. DOI: 10.5220/0006611100280036.

3. Shal S.A., Koltun V. Robust continuous clustering. Proceedings of the National Academy of Sciences, 2017, vol. 114, no. 37, pp. 9814–9817. DOI: 10.1073/pnas.1700770114.

4. Miri E., Razavi S.M., Sadri J. Performance optimization of neural networks in handwritten digit recognition using intelligent fuzzy c-means clustering. Proc. ICCKE, 2011, pp. 150–155. DOI: 10.1109/ICCKE.2011.6413342.

5. Pourmohammad S., Soosahabi R., Maida A.S. An efficient character recognition scheme based on k-means clustering. ICMSAO, 2013, pp. 1–6. DOI: 10.1109/ICMSAO.2013.6552640.

6. Li B.Y. An experiment of k-means initialization strategies on handwritten digits dataset. Intelligent Information Management, 2018, vol. 10, pp. 43–48. DOI: 10.4236/iim.2018.102003.

7. Munggaran L.C., Widodo S., Cipta A.M. Handwritten pattern recognition using Kohonen neural network based on pixel character. IJACSA, 2014, vol. 5, no. 11, pp. 1–6. DOI: 10.14569/IJACSA.2014.051101.

8. Fahad A., Alshatri N., Tari Z., Alamari A., Zomaya A. et al. A survey of clustering algorithms for big data: taxonomy and empirical analysis. IEEE Transactions on Emerging Topics in Computing, 2014, vol. 2, no. 3, pp. 267–279. DOI: 10.1109/TETC.2014.2330519.

9. Bi Y., Wang P., Guo X. K-means clustering optimizing deep stacked sparse autoencoder. Sensing and Imaging, 2019, vol. 20, art. 6. DOI: 10.1007/s11220-019-0227-1.

10. Chen Y., Li C.G., You C. Stochastic sparse subspace clustering. Proc. CVPR, 2020, pp. 4155–4164.

11. Zhang S., You C., Vida R., Li C.G. Learning a self-expressive network for subspace clustering. Proc. CVPR, 2021, pp. 12393–12403.

12. Jang H., Park A., Jung K. Neural network implementation using CUDA and Open MP. Proc. DICTA, 2008, pp. 155–161. DOI: 10.1109/DICTA.2008.82.

13. Wittek P., Daranyi S. A GPU-accelerated algorithm for Self-Organizing Maps in a distributed environment. Proc. ESANN, 2012, pp. 609–614.

14. Daneshpajouh H., Delisle P., Boisson J.C., Krajecki M., Zakaria N. Parallel batch Self-Organizing Map on graphics processing unit using CUDA. In: High Performance Computing, CARLA, 2018, pp. 87–100. DOI: 10.1007/978-3-319-73353-1_6.

15. McConnell S., Sturgeon R., Henry G., Mayne A., Hurley R. Scalability of Self-Organizing Maps on a GPU cluster using OpenCL and CUDA. J. of Physics: Conf. Ser., 2012, vol. 341, art. 012018. DOI: 10.1088/1742-6596/341/1/012018.

16. Takatsuka M., Bui M. Parallel batch training of the Self-Organizing Map using OpenCL. Proc. ICONIP, 2010, pp. 470–476. DOI: 10.1007/978-3-642-17534-3_58.

17. Liu Y., Sun J., Yao Q., Wang S., Zheng K., Liu Y. A scalable heterogeneous parallel SOM based on MPI/CUDA. PMLR, 2018, vol. 95, pp. 264–279.

18. LeCun Y., Cortes C., Burges C.J.C. The MNIST Database Handwritten Digits. URL: http://yann.lecun.com/exdb/mnist/ (дата обращения: 17.01.2022).

19. Xu Y., Zhang W. On a clustering method for handwritten digit recognition. Proc. III Int. Conf. on Intelligent Networks and Intelligent Systems, 2010, pp. 112–115. DOI: 10.1109/ICINIS.2010.130.

20. Cohen G., Afshar S., Tapson J., Schaik A. EMNIST: Extending MNIST to handwritten letters. IJCNN, 2017, pp. 2921–2926. DOI: 10.1109/IJCNN.2017.7966217.

21. Baldominos A., Saez Y., Isasi P. A Survey of handwritten character recognition with MNIST and EMNIST. Applied Sciences, 2019, vol. 9, no. 15, art. 3169. DOI: 10.3390/app9153169.

22. Agarap A.F., Azcarraga A.P. Improving k-means clustering performance with disentangled internal representations. IJCNN, 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207192.

23. Cheng K., Tahir R., Eric L.K., Li M. An analysis of generative adversarial networks and variants for image synthesis on MNIST dataset. Multimedia Tools and Applications, 2020, vol. 79, no. 19-20, pp. 13725–13752. DOI: 10.1007/s11042-019-08600-2.

24. Kossen J., Farquhar S., Gal Y., Rainforth T. Active testing: Sample-efficient model evaluation. PMLR, 2021, vol. 139, pp. 5753–5763.

25. Zhang R., Chang P.C. Robustness against adversary models on MNIST by deep-Q reinforcement learning based parallel-GANs. Proc. APSIPA ASC, 2021, pp. 1590–1597.

26. Kayumov Z., Tumakov D., Mosin S. Hierarchical convolutional neural network for handwritten digits recognition. Procedia Computer Science, 2020, vol. 171, pp. 1927–1934. DOI: 10.1016/j.procs.2020.04.206.

27. Kayumov Z., Tumakov D., Mosin S. Combined convolutional and perceptron neural networks for handwritten digits recognition. Proc. DSPA, 2020, pp. 1–5. DOI: 10.1109/DSPA48919.2020.9213301.

28. Murtagh F., Hernández-Pajares M. The Kohonen Self-Organizing Map method: An Assessment. J. of Classification, 1995, vol. 12, no. 2, pp. 165–190. DOI: 10.1007/BF03040854.

29. Storti D., Yurtoglu M. CUDA for Engineers: An Introduction to High-performance Parallel Computing. Addison-Wesley Professional Publ., 2015, 352 p.

30. Shirokanev A.S., Andriyanov N.A., Ilyasova N.Y. Development of vector algorithm using CUDA technology for three-dimensional retinal laser coagulation process modeling. Computer Optics, 2021, vol. 45, pp. 427–437. DOI: 10.18287/2412-6179-CO-828.

31. Giniyatova D., Tumakov D., Markina A. Solving the problem of electromagnetic wave diffraction by a flat screen using CUDA. Lobachevskii J. of Mathematics, 2021, vol. 42, no. 6, pp. 1335–1344. DOI: 10.1134/S1995080221060081.

32. Imankulov T., Daribayev B., Mukhambetzhanov S. Comparative analysis of parallel algorithms for solving oil recovery problem using CUDA and OpenCL. Int. J. of Nonlinear Analysis and Applications, 2021, vol. 12, no. 1, pp. 351–364.

33. Сеньковская И.С., Сараев П.В. Автоматическая кластеризация в анализе данных на основе самоорганизующихся карт Кохонена. Вестн. МГТУ им. Г.И. Носова. 2011. С. 278–279.

34. Nordstrom T. Designing parallel computers for Self-Organizing Maps. Proc. IV Swedish Workshop on Computer System Architecture, Linkoping, 1992, pp. 1–10.

35. Латыпова Д.С., Тумаков Д.Н. Определение основных кластеров рукописных цифр // DSPA. 2020. С. 620–625.

36. Latypova D., Tumakov D. Peculiarities of image recognition by the Hopfield neural network. Proc. IEMAICLOUD, 2021, vol. 273, pp. 34–47. DOI: 10.1007/978-3-030-92905-3_4.

Software & Systems
DOI: 10.15827/0236-235X.139.362-373

Received 25.07.22
2022, vol. 35, no. 3, pp. 362–373

Applying CUDA technology for training the Kohonen neural network

D.S. Latypova [1], *Postgraduate Student, dina.latypova23@gmail.com*
D.N. Tumakov [1], *Ph.D. (Physics and Mathematics), Associate Professor, dtumakov@kpfu.ru*

[1] *Kazan Federal University, Institute of Computational Mathematics and Information Technology, Kazan, 420008, Russian Federation*

Abstract. The paper presents clustering of the data from in the training samples of the MNIST and Fashion MNIST databases. For clustering, the authors use a Kohonen neural network with a Euclidean metric for estimating distances. The optimal number of clusters (no more than 50) is determined for each handwritten digit (MNIST) and type of clothing (Fashion MNIST).

Neural network training is parallelized on a NVidia graphics device using CUDA technology. There are the results for each digit illustrating the comparison of the processor and GPU operating time. For both the digits and clothing types, there is a conclusion about a 17-fold acceleration on an entry-level gaming laptop. Test samples of the same databases are used to verify the cluster construction correctness. For both sequential and parallel learning, it is concluded that the vectors from the test sample belong to the correct cluster with a probability of more than 90 % in the case of handwritten digits. In addition, there are calculated F-measures for each digit and type of clothing to evaluate clusters.

It is shown that sequential and parallel clustering give similar results. The best values of the F-measure are obtained for the digits 0 and 1 (F-mean is 0.974), while the worst value is obtained for the digit 9 (F-mean is 0.903). For the Fashion MNIST data, the best value for the F-measure was obtained for trousers (F-average value is 0.96), and the worst value was for a shirt (F-average value is 0.34). Despite the large variations for the F-metric values of the considered two databases, the differences in the clustering results are minimal. Thus, the maximum difference of the F-measure is about 0.01 for the MNIST and about 0.04 for the Fashion MNIST.

Keywords: Kohonen neural network, clustering, parallelization, CUDA, MNIST.

Acknowledgements. *This paper has been supported by the Kazan Federal University Strategic Academic Leadership Program ("PRIORITY-2030").*

References

1. Rexy M., Lavanya K. Handwritten digit recognition of MNIST data using consensus clustering. *IJRTE*, 2019, vol. 7, no. 6, pp. 1969–1973.

2. Nhery S., Ksantini R., Kaaniche M.B., Bouhoula A. A novel handwritten digits recognition method based on subclass low variances guided support vector machine. *Proc. XIII Int. Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications. VISAPP*, 2018, vol. 4, pp. 28–36. DOI: 10.5220/0006611100280036.

3. Shal S.A., Koltun V. Robust continuous clustering. *Proceedings of the National Academy of Sciences*, 2017, vol. 114, no. 37, pp. 9814–9817. DOI: 10.1073/pnas.1700770114.

4. Miri E., Razavi S.M., Sadri J. Performance optimization of neural networks in handwritten digit recognition using intelligent fuzzy c-means clustering. *Proc. ICCKE*, 2011, pp. 150–155. DOI: 10.1109/ICCKE.2011.6413342.

5. Pourmohammad S., Soosahabi R., Maida A.S. An efficient character recognition scheme based on k-means clustering. *ICMSAO*, 2013, pp. 1–6. DOI: 10.1109/ICMSAO.2013.6552640.

6. Li B.Y. An experiment of k-means initialization strategies on handwritten digits dataset. *Intelligent Information Management*, 2018, vol. 10, pp. 43–48. DOI: 10.4236/iim.2018.102003.

7. Munggaran L.C., Widodo S., Cipta A.M. Handwritten pattern recognition using Kohonen neural network based on pixel character. *IJACSA*, 2014, vol. 5, no. 11, pp. 1–6. DOI: 10.14569/IJACSA.2014.051101.

8. Fahad A., Alshatri N., Tari Z., Alamari A., Zomaya A. et al. A survey of clustering algorithms for big data: taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2014, vol. 2, no. 3, pp. 267–279. DOI: 10.1109/TETC.2014.2330519.

9. Bi Y., Wang P., Guo X. K-means clustering optimizing deep stacked sparse autoencoder. *Sensing and Imaging*, 2019, vol. 20, art. 6. DOI: 10.1007/s11220-019-0227-1.

10. Chen Y., Li C.G., You C. Stochastic sparse subspace clustering. *Proc. CVPR*, 2020, pp. 4155–4164.

11. Zhang S., You C., Vida R., Li C.G. Learning a self-expressive network for subspace clustering. *Proc. CVPR*, 2021, pp. 12393–12403.

12. Jang H., Park A., Jung K. Neural network implementation using CUDA and Open MP. *Proc. DICTA*, 2008, pp. 155–161. DOI: 10.1109/DICTA.2008.82.

13. Wittek P., Daranyi S. A GPU-accelerated algorithm for Self-Organizing Maps in a distributed environment. *Proc. ESANN*, 2012, pp. 609–614.

14. Daneshpajouh H., Delisle P., Boisson J.C., Krajecki M., Zakaria N. Parallel batch Self-Organizing Map on graphics processing unit using CUDA. In: *High Performance Computing, CARLA*, 2018, pp. 87–100. DOI: 10.1007/978-3-319-73353-1_6.

15. McConnell S., Sturgeon R., Henry G., Mayne A., Hurley R. Scalability of Self-Organizing Maps on a GPU cluster using OpenCL and CUDA. *J. of Physics: Conf. Ser.*, 2012, vol. 341, art. 012018. DOI: 10.1088/1742-6596/341/1/012018.

16. Takatsuka M., Bui M. Parallel batch training of the Self-Organizing Map using OpenCL. *Proc. ICONIP*, 2010, pp. 470–476. DOI: 10.1007/978-3-642-17534-3_58.

17. Liu Y., Sun J., Yao Q., Wang S., Zheng K., Liu Y. A scalable heterogeneous parallel SOM based on MPI/CUDA. *PMLR*, 2018, vol. 95, pp. 264–279.

18. LeCun Y., Cortes C., Burges C.J.C. *The MNIST Database Handwritten Digits*. Available at: http://yann.lecun.com/exdb/mnist/ (accessed January 17, 2022).

19. Xu Y., Zhang W. On a clustering method for handwritten digit recognition. *Proc. III Int. Conf. on Intelligent Networks and Intelligent Systems*, 2010, pp. 112–115. DOI: 10.1109/ICINIS.2010.130.

20. Cohen G., Afshar S., Tapson J., Schaik A. EMNIST: Extending MNIST to handwritten letters. *IJCNN*, 2017, pp. 2921–2926. DOI: 10.1109/IJCNN.2017.7966217.

21. Baldominos A., Saez Y., Isasi P. A Survey of handwritten character recognition with MNIST and EMNIST. *Applied Sciences*, 2019, vol. 9, no. 15, art. 3169. DOI: 10.3390/app9153169.

22. Agarap A.F., Azcarraga A.P. Improving k-means clustering performance with disentangled internal representations. *IJCNN*, 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207192.

23. Cheng K., Tahir R., Eric L.K., Li M. An analysis of generative adversarial networks and variants for image synthesis on MNIST dataset. *Multimedia Tools and Applications*, 2020, vol. 79, no. 19-20, pp. 13725–13752. DOI: 10.1007/s11042-019-08600-2.

24. Kossen J., Farquhar S., Gal Y., Rainforth T. Active testing: Sample-efficient model evaluation. *PMLR*, 2021, vol. 139, pp. 5753–5763.

25. Zhang R., Chang P.C. Robustness against adversary models on MNIST by deep-Q reinforcement learning based parallel-GANs. *Proc. APSIPA ASC*, 2021, pp. 1590–1597.

26. Kayumov Z., Tumakov D., Mosin S. Hierarchical convolutional neural network for handwritten digits recognition. *Procedia Computer Science*, 2020, vol. 171, pp. 1927–1934. DOI: 10.1016/j.procs.2020.04.206.

27. Kayumov Z., Tumakov D., Mosin S. Combined convolutional and perceptron neural networks for handwritten digits recognition. *Proc. DSPA*, 2020, pp. 1–5. DOI: 10.1109/DSPA48919.2020.9213301.

28. Murtagh F., Hernández-Pajares M. The Kohonen Self-Organizing Map method: An Assessment. *J. of Classification*, 1995, vol. 12, no. 2, pp. 165–190. DOI: 10.1007/BF03040854.

29. Storti D., Yurtoglu M. *CUDA for Engineers: An Introduction to High-performance Parallel Computing*. Addison-Wesley Professional Publ., 2015, 352 p.

30. Shirokanev A.S., Andriyanov N.A., Ilyasova N.Y. Development of vector algorithm using CUDA technology for three-dimensional retinal laser coagulation process modeling. *Computer Optics*, 2021, vol. 45, pp. 427–437. DOI: 10.18287/2412-6179-CO-828.

31. Giniyatova D., Tumakov D., Markina A. Solving the problem of electromagnetic wave diffraction by a flat screen using CUDA. *Lobachevskii J. of Mathematics*, 2021, vol. 42, no. 6, pp. 1335–1344. DOI: 10.1134/S1995080221060081.

32. Imankulov T., Daribayev B., Mukhambetzhanov S. Comparative analysis of parallel algorithms for solving oil recovery problem using CUDA and OpenCL. *Int. J. of Nonlinear Analysis and Applications*, 2021, vol. 12, no. 1, pp. 351–364.

33. Senkovskaya I.S., Saraev P.V. Automatic clustering in data analysis based on Kohonen Self-Organizing Maps. *Vestn. of NMSTU*, 2011, pp. 278–279 (in Russ.).

34. Nordstrom T. Designing parallel computers for Self-Organizing Maps. *Proc. IV Swedish Workshop on Computer System Architecture*, Linkoping, 1992, pp. 1–10.

35. Latypova D., Tumakov D. Identification of base clusters of handwritten digits. *Proc. DSPA*, 2020, pp. 620–625 (in Russ.).

36. Latypova D., Tumakov D. Peculiarities of image recognition by the Hopfield neural network. *Proc. IEMAICLOUD*, 2021, vol. 273, pp. 34–47. DOI: 10.1007/978-3-030-92905-3_4.

Для цитирования

Латыпова Д.С., Тумаков Д.Н. Применение технологии CUDA для обучения нейронной сети Кохонена // Программные продукты и системы. 2022. Т. 35. № 3. С. 362–373. DOI: 10.15827/0236-235X.139.362-373.

For citation

Latypova D.S., Tumakov D.N. Applying CUDA technology for training the Kohonen neural network. *Software & Systems*, 2022, vol. 35, no. 3, pp. 362–373 (in Russ.). DOI: 10.15827/0236-235X.139.362-373.