УДК 517.9

ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ РАСПОЗНАВАНИЯ ОБРАЗА ПРЕДМЕТА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

А. А. Ферцев
(Мордовский ГУ им. Н. П. Огарева, г. Саранск)

Представлена реализация нейронной сети, обучаемой по алгоритму на основе метода Левенберга—Марквардта. С помощью технологии NVIDIA CUDA обучение построенной нейронной сети ускорено почти в 16 раз. Построенная нейронная сеть применена для распознавания зашумленных изображений.

Ключевые слова: распознавание, нейронная сеть, метод Левенберга—Марквардта, графический процессор, CUDA.

Введение

Одним из перспективных методов исследования земной поверхности, океана, атмосферы является технология LIDAR (Light Detection and Ranging). Это технология получения и обработки информации об удаленных объектах с помощью активных оптических систем, использующих явления отражения света и его рассеивания в прозрачных и полупрозрачных средах. Принцип действия LIDAR не имеет больших отличий от радара: направленный луч источника излучения отражается от целей, возвращается к источнику и улавливается высокочувствительным приемником (в случае LIDAR — светочувствительным полупроводниковым прибором); время отклика обратно пропорционально расстоянию до цели (рис. 1).

В процессе обработки и интерпретации результатов сканирования при помощи LIDAR возникает задача распознавания полученных изображений. В данной работе LIDAR рассматривается только как одно из возможных средств получения изображения, физические и конструктивные особенности этой технологии на данном этапе не рассматриваются. Жесткая привязка программного комплекса к особенностям какой-либо одной технологии получения изображения представляется нецелесообразной.

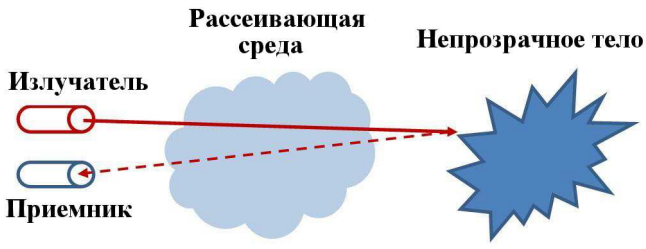

Рис. 1. Технология LIDAR

Модель процесса распознавания образов

Рассмотрим два пространства: пространство характеристик и тематическое. В пространстве характеристик находится вся информация, прямая и косвенная, которую можно получить об интересующем явлении по его изображению. В тематическом пространстве задается само изучаемое явление. Тогда задачу распознавания можно интерпретировать как построение отображения из пространства характеристик в тематическое пространство, при этом на отображение могут налагаться те или иные ограничения. Это отображение называется *классифицирующим* отображением. Образ — группа элементов тематического пространства со сходными (сходство определяется постановкой задачи) характеристиками. Схематично процесс распознавания представлен на рис. 2.

Совокупность характеристик объектов, для которых известны их образы, образует обучающее множество (набор эталонов). Основная задача распознавания заключается в том, чтобы исходя из обучающего множества определить об-

Рис. 2. Модель процесса распознавания образов

раз, которому соответствует набор конкретных характеристик, либо установить, что такого образа не существует.

В рамках данной статьи будем рассматривать полутоновые растровые изображения. В качестве тематического пространства взят латинский алфавит и десять арабских цифр. Для простоты цифры и буквы начертаны одним и тем же шрифтом. В качестве характеристик изображения взяты средние яркости частей изображения размером 8×8 (в пикселях). Выбор средних яркостей частей изображения вместо яркостей отдельных пикселей позволяет, во-первых, уменьшить размерность входных данных, что приводит к уменьшению сложности вычислений, и, во-вторых, снизить влияние шумов на результаты распознавания. Подобный подход продемонстрирован в работе [1].

В качестве направления для дальнейшего исследования перспективным представляется переход к характеристикам изображений более высокого порядка, описанным, например, в работах [2, 3]. Это позволит избавиться от ограничений в начертании отдельных символов и сделает систему распознавания более универсальной.

В настоящее время одним из самых популярных направлений при решении задач распознавания изображений является применение различных нейронных сетей. В данной статье описывается реализация распознавания изображений с помощью нейронной сети обратного распространения. В качестве алгоритма обучения нейронной сети выбран алгоритм на основе метода Левенберга—Марквардта.

Нейронная сеть с обратным распространением ошибки

Классический алгоритм обучения нейронной сети с обратным распространением ошибки впервые был предложен в 1974 г. А. И. Галушкиным [4] и П. Дж. Вербосом [5] независимо друг от друга. Этот алгоритм является модификацией классического алгоритма градиентного спуска. Цель алгоритма — поиск точки минимума ошибки нейронной сети.

В качестве ошибки нейронной сети Румельхарт [6] предложил использовать среднеквадратичную ошибку:

$$E(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{p}(y_i - d_i)^2, \qquad (1)$$

где $\boldsymbol{\theta}$ — вектор параметров сети; y_i, d_i — соответственно полученный и желаемый выходы i-го выходного нейрона сети; p — количество выходных нейронов. Тогда точку минимума функции ошибки, рассчитываемой по формуле (1), т. е. значения параметров, при которых ошибка сети минимальна, согласно [6] следует искать в направлении, противоположном направлению градиента функции ошибки в точке

$$\Delta\theta_{ij}^{(k)}(t) = -\eta\frac{\partial E(\boldsymbol{\theta})}{\partial\theta_{ij}^{(k)}},$$

где $\theta_{ij}^{(k)}(t)$ — j-й параметр i-го нейрона слоя k; $\Delta\theta_{ij}^{(k)}(t)$ — приращение этого параметра за эпоху обучения t; $\eta \in (0,\,1)$ — параметр, называемый скоростью обучения.

Как показано в [6], приращение параметров сети на каждом шаге следует искать по формуле

$$\Delta\theta_{ij}^{(k)}(t) = -\eta\delta_j^{(k)}x_i^{(k)}. \qquad (2)$$

Здесь x_i — выход i-го нейрона предыдущего слоя, а $\delta_j^{(k)}$ ищется по формуле

$$\delta_j^{(k)} = \left(\sum_r \delta_r^{(k+1)}\theta_{jr}^{(k+1)}\right)\frac{d\sigma\left(S_j^{(k)}\right)}{dS},\quad k = \overline{1,\ldots,N-1};$$

$$\delta_j^{(N)} = (y_j - d_j)\frac{d\sigma\left(S_j^{(N)}\right)}{dS},$$

где $S_j^{(k)}$ — взвешенная сумма входов j-го нейрона слоя k; σ — функция активации нейрона; N — число слоев нейронной сети; r — индекс слоя нейронной сети. То есть, вычислив приращения параметров на последнем слое сети с использованием известных желаемых выходов, можно вычислить приращения остальных параметров сети. Тогда новые параметры сети будут рассчитываться по формуле

$$\theta_{ij}^{(k)}(t) = \theta_{ij}^{(k)}(t-1) + \Delta\theta_{ij}^{(k)}(t).$$

Данный алгоритм реализует *онлайн-обучение* — параметры сети меняются после предъявления сети каждого элемента обучающего множества. Классическому алгоритму присущ целый ряд недостатков, значительно снижающих его применимость:

- скорость сходимости алгоритма зависит от параметра η, однако его увеличение может привести к практически неограниченному росту параметров сети — так называемому *параличу сети*;
- прямая зависимость от градиента приводит к тому, что в точках, где поверхность, задаваемая функцией ошибки, имеет малую *крутизну*, шаг алгоритма уменьшается, а при большой крутизне — увеличивается, хотя, очевидно, должно быть наоборот. В результате либо большое число итераций алгоритма будет произведено в окрестности точки, где градиент очень мал, либо точка экстремума будет пропущена при итерациях алгоритма, если рядом с ней градиент велик.

Для устранения этих недостатков, сначала Руменхарт в [6], а затем Минаи и Вильямс в [7] предложили эвристический подход, основанный на введении момента μ для связи текущего приращения параметров нейронной сети с приращением на предыдущем шаге. При использовании такого подхода формула (2) принимает вид

$$\Delta\theta_{ij}^{(k)}(t) = -\eta\delta_j^{(k)}x_i^{(k)} + \mu\Delta\theta_{ij}^{(k)}(t-1).$$

В результате появляется способность алгоритма преодолевать мелкие локальные минимумы функции ошибки. Тем не менее данный подход дает лишь незначительные улучшения и в целом не устраняет недостатков алгоритма.

Применение других эвристических подходов — переменной скорости обучения [8] и стохастического обучения [9] — также не привело к успеху. Несколько лучшие результаты получены с помощью искусственного увеличения ошибки нейронов из области насыщения. Этот подход описан в работах [10—12]. Наиболее значительные улучшения классического алгоритма были сделаны с помощью методов второго порядка — метода Ньютона [13], сопряженных градиентов [14] и метода Левенберга—Марквардта [15]. В частности, в работе [15] доказывается, что метод Левенберга—Марквардта является наиболее эффективным.

Метод Левенберга—Марквардта

Метод Левенберга—Марквардта был предложен Левенбергом в [16] и развит Марквардтом в [17] как метод для решения задачи о наименьших квадратах. Он был применен для обучения нейронных сетей, например, в [18, 15, 19]. Необходимо отметить, что в отличие от классического алгоритма обучения алгоритм на основе метода Левенберга—Марквардта использует *обучение по эпохам* (в англоязычной литературе используется термин *batch learning*). В этом случае ошибка сети считается за всю эпоху обучения и параметры сети изменяются, когда сети уже предъявлены все элементы обучающего множества.

Согласно [15] и [20] нейронную сеть можно представить в виде вектора-функции вектора-аргумента:

$$\mathbf{Y} = Y(\mathbf{X}, \boldsymbol{\theta}), \tag{3}$$

где $\mathbf{X} = (x_1, \ldots, x_n)$ — входные данные; $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$ — параметры сети; $\mathbf{Y} = (y_1, \ldots, y_p)$ — выход сети. Тогда ошибка сети за одну эпоху будет выражаться формулой

$$F(\mathbf{Y}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}(y_{ij} - d_{ij})^2, \tag{4}$$

где d_{ij} — желаемый выход j-го выходного нейрона для i-го элемента обучающего множества. Пусть $\mathbf{E} = (e_{11}, \ldots, e_{1p}, \ldots, e_{L1}, \ldots, e_{Lp})^T$, где L — количество элементов обучающей выборки; $e_{ij} = y_{ij} - d_{ij}$. Тогда формулу (4) можно переписать так:

$$F(\mathbf{Y}) = \mathbf{E}^T\mathbf{E},$$

и матрица Якоби для уравнения (4) будет иметь вид

$$J = \begin{pmatrix} \widetilde{J}_1 \\ \vdots \\ \widetilde{J}_i \\ \vdots \\ \widetilde{J}_L \end{pmatrix}, \qquad \widetilde{J}_i = \begin{pmatrix} \frac{\partial e_{1i}}{\partial\theta_1} & \frac{\partial e_{1i}}{\partial\theta_2} & \cdots & \frac{\partial e_{1i}}{\partial\theta_M} \\ \frac{\partial e_{2i}}{\partial\theta_1} & \frac{\partial e_{2i}}{\partial\theta_2} & \cdots & \frac{\partial e_{2i}}{\partial\theta_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{pi}}{\partial\theta_1} & \frac{\partial e_{pi}}{\partial\theta_2} & \cdots & \frac{\partial e_{pi}}{\partial\theta_M} \end{pmatrix}. \tag{5}$$

Согласно [20] приращение параметров сети следует искать в виде решения уравнения

$$\left(J^T J + \lambda I\right) \Delta\boldsymbol{\theta} = J^T \mathbf{E}, \tag{6}$$

где J — матрица Якоби, полученная по формуле (4); I — единичная матрица. Однако, как отмечает Марквардт в [17], если параметр λ будет достаточно велик, то влияние аппроксимированной матрицы Гессе $J^T J$ практически равно нулю. Чтобы избежать этого, в [17] предложено заменить единичную матрицу матрицей с диагональю аппроксимированной матрицы Гессе. Тогда формула (6) будет иметь вид

$$\left(J^T J + \lambda \operatorname{diag}\left(J^T J\right)\right) \Delta\boldsymbol{\theta} = J^T \mathbf{E}. \tag{7}$$

Таким образом, алгоритм обучения нейронной сети на основе метода Левенберга—Марквардта будет следующим:

1. Рассчитать ошибку сети за одну эпоху по формуле (4).
2. Рассчитать элементы матрицы Якоби по формуле (5).
3. Решить уравнение (7) и рассчитать ошибку сети для вновь полученных параметров сети. Если ошибка уменьшилась, перейти к шагу 5, иначе перейти к шагу 4.
4. Вернуться к прежним значениям параметров сети и увеличить регуляризующий параметр λ (обычно в 10 раз). Перейти к шагу 3.
5. Принять полученные значения параметров сети, уменьшить значение параметра λ (обычно также в 10 раз) и перейти к новой эпохе обучения.

Заметим, что нейронную сеть, обучаемую по приведенному алгоритму, правильнее называть сетью *с прямым распространением ошибки*, поскольку в ней отсутствует классическая процедура обратного распространения приращений параметров.

Применение алгоритма на основе метода Левенберга—Марквардта

В рамках данной статьи реализовано обучение нейронной сети, состоящей из трех слоев: входного (n нейронов), скрытого (m нейронов) и выходного (p нейронов). Кроме того, на каждом слое присутствует нейрон, называемый *пороговым* (в англоязычной литературе используется термин *bias* или *threshold*), выход которого, в отличие от обычного нейрона, всегда равен единице. Введение такого нейрона делает алгоритм обучения нейронной сети более гибким. Для нейронной сети с одним скрытым слоем формула (3) примет вид

$$\mathbf{Y} = Y(\mathbf{X}, \boldsymbol{\theta}) = \sigma\Big(W^{(2)}\sigma\Big(W^{(1)}\mathbf{X} + \mathbf{B}^{(1)}\Big) + \mathbf{B}^{(2)}\Big),$$

где $W^{(1)}$ и $W^{(2)}$ — матрицы весов нейронов скрытого и выходного слоев соответственно; $\mathbf{B}^{(1)}$ и $\mathbf{B}^{(2)}$ — векторы весов пороговых нейронов скрытого и выходного слоев соответственно; σ — функция активации нейрона. При этом элементы матрицы Якоби будут иметь следующий вид:

– если θ_r — вес нейрона скрытого слоя, т. е. $\theta_r \equiv w^{(1)}_{i'j'}$, то

$$\frac{\partial e_i}{\partial \theta_r} \equiv \frac{\partial e_i}{\partial w^{(1)}_{i'j'}} = \\ = w^{(2)}_{i'j'}\sigma'\left(\sigma\left(\sum_{j=1}^{n} w^{(1)}_{i'j} x_j\right)\right) x_{j'}\sigma'\left(\sum_{k=1}^{m} w^{(2)}_{ik}\bar{x}_k\right),$$

где $\bar{x}_k$ — выход k-го нейрона скрытого слоя; $\sigma'(\cdot)$ — значение производной функции активации в точке;

– если θ_r — вес нейрона выходного слоя, т. е. $\theta_r \equiv w^{(2)}_{i'j'}$, то

$$\frac{\partial e_i}{\partial \theta_r} \equiv \frac{\partial e_i}{\partial w^{(2)}_{i'j'}} = \begin{cases} \sigma'\left(\sum_{k=1}^{m} w^{(2)}_{ik}\bar{x}_k\right)\bar{x}_{j'}, & i = i'; \\ 0, & i \neq i'; \end{cases}$$

– если θ_r — вес порогового нейрона скрытого слоя, т. е. $\theta_r \equiv b^{(1)}_{i'}$, то

$$\frac{\partial e_i}{\partial \theta_r} \equiv \frac{\partial e_i}{\partial b^{(1)}_{i'}} = \\ = w^{(2)}_{i'j'}\sigma'\left(\sigma\left(\sum_{j=1}^{n} w^{(1)}_{i'j} x_j\right)\right)\sigma'\left(\sum_{k=1}^{m} w^{(2)}_{ik}\bar{x}_k\right);$$

– если θ_r — вес порогового нейрона выходного слоя, т. е. $\theta_r \equiv b^{(2)}_{i'}$, то

$$\frac{\partial e_i}{\partial \theta_r} \equiv \frac{\partial e_i}{\partial b^{(2)}_{i'}} = \begin{cases} \sigma'\left(\sum_{k=1}^{m} w^{(2)}_{ik}\bar{x}_k\right), & i = i'; \\ 0, & i \neq i'. \end{cases}$$

Стандартный алгоритм на основе метода Левенберга—Марквардта имеет ряд существенных недостатков:

1. Алгоритм плохо справляется с ситуацией, когда в обучающем множестве присутствуют элементы, сильно выделяющиеся из общей совокупности (обычно такая ситуация возникает при использовании данных, полученных опытным путем).
2. Алгоритм очень чувствителен к выбору начальных значений весов.
3. При работе с нейронными сетями большого размера требуется много памяти, а необходимость обращения матриц большого размера на каждом шаге приводит к усложнению вычислительного процесса.

Для снижения влияния этих недостатков при практической реализации алгоритма был применен ряд подходов.

Для устранения первого недостатка был использован метод, предложенный МакКеем в работах [21, 22]. Суть этого метода — в переходе от поиска точки минимума среднеквадратичной ошибки, рассчитываемой по формуле (4), к поиску минимума функции, выражаемой формулой

$$F(\mathbf{Y}) = \alpha E_w + \beta E_d, \tag{8}$$

где E_d — ошибка сети; E_w — среднеквадратичная сумма параметров сети; α и β — гиперпараметры. В результате применения алгоритма реализуется попытка не только минимизировать ошибку сети, но и не допустить неограниченного роста ее параметров. В данной статье для расчета гиперпараметров используются формулы, предложенные Поландом в работе [23]:

$$\gamma = M - \alpha \operatorname{trace}\left(\left[J^T J\right]^{-1}\right);$$
$$\alpha = \frac{\gamma}{2E_w + \operatorname{trace}\left(\left[J^T J\right]^{-1}\right)};$$
$$\beta = \frac{pn}{2E_d},$$

где $\operatorname{trace}\left(\left[J^T J\right]^{-1}\right)$ — сумма диагональных элементов матрицы, обратной к аппроксимированной матрице Гессе; M — число параметров нейронной сети. Влияние такого подхода особенно заметно при небольшом обучающем множестве.

Для уменьшения чувствительности алгоритма к начальным значениям параметров сети был применен подход, предложенный Нгуеном и Уидроу в работе [24]. На практике этот подход реализуется следующим образом.

Сначала все параметры сети инициализируются случайными значениями из отрезка $[-0{,}5, 0{,}5]$. После этого каждый параметр нормируется и умножается на коэффициент $0{,}7\sqrt[n]{m}$, где n — число нейронов текущего слоя, а m — следующего.

В результате алгоритм быстрее сходится и менее подвержен флюктуациям начальных значений параметров нейронной сети. В [24] отмечается существенное сокращение числа эпох обучения (до нескольких порядков), однако в рамках данной работы сокращение числа эпох было более скромным — около 20—30 %. Следует также отметить, что без применения данного подхода в ряде случаев алгоритм не сходился вовсе (параметр λ вырастал практически неограниченно, и обучение приходилось останавливать).

Чтобы избежать потери нейронной сетью свойства генерализации и сократить число эпох обучения, применен так называемый *метод раннего останова*. Его суть заключается в том, что обучающее множество разбивается на два: собственно обучающее множество, используемое для обучения, и тестовое множество, используемое для тестирования обучаемой сети. В процессе обучения нейронная сеть постоянно подвергается проверке с помощью тестового множества. Как только сеть сможет правильно распознать все элементы тестового множества, обучение останавливается. Некоторые авторы рекомендуют другую стратегию проверки — обучение останавливается, как только ошибка на тестовом множестве начинает возрастать. Данный подход имеет недостаток — нейронная сеть, обученная методом раннего останова, может иметь слишком большую ошибку и вследствие этого стать малопригодной для практических вычислений. Для снижения вероятности такого события в данной работе тестовое множество расширено включением зашумленных образцов. Общее решение данной проблемы — применение ансамблей нейронных сетей (bagging- и boosting-коллективов). На данном этапе ансамбли сетей не реализованы — их реализация планируется в дальнейшем.

Как уже отмечалось, для уменьшения размерности входных данных на входы нейронной сети в данной работе подаются средние яркости частей изображения размером 8×8 (в пикселях). Несмотря на это, обучение нейронной сети с по-

мощью алгоритма на основе метода Левенберга—Марквардта является очень ресурсоемким процессом, особенно при умножении матриц и нахождении обратной матрицы. Самым очевидным решением в этом случае является переход от последовательных вычислений к параллельным.

Еще недавно единственной альтернативой вычислений на одном процессоре был вычислительный кластер, состоящий из множества ядер с общей или раздельной памятью. Стоимость такого решения колеблется от десятков тысяч до сотен миллионов долларов. Существенные изменения в области параллельных вычислений внесла компания NVIDIA: 15 февраля 2007 г. она представила специальный API (Application Programming Interface — интерфейс для программирования приложений) для переноса вычислений на графический процессор (GPU) — CUDA (Compute Unified Device Architecture). В настоящее время вычисления на графических процессорах являются одним из наиболее динамично развивающихся направлений в области параллельных вычислений. Основным фактором, благодаря которому вычисления на GPU обрели столь большую популярность, является сравнительная дешевизна графических процессоров, которая дает возможность проводить параллельные вычисления с высокой скоростью на обычных GPU персональных компьютеров.

Фактически графический процессор с поддержкой CUDA — это вычислительный кластер с множеством узлов и общей памятью. В отличие от центрального процессора (CPU) графический процессор изначально предназначен для параллельных вычислений — построение трехмерных моделей, хотя и требует высокой производительности, очень хорошо распараллеливается.

Согласно [25, 26] вычислительная модель GPU на верхнем уровне представляет собой сетку размером $N1 \times N2 \times N3$ (рис. 3), состоящую из блоков. Каждый блок, в свою очередь, состоит из множества *нитей*, производящих параллельные вычисления (рис. 4).

Блок обладает общей памятью (shared memory), доступной всем нитям блока. Каждая нить обладает локальной (local memory) и регистровой (register memory), не доступной для разработчика, памятью. Общая память и локальная память — быстрые виды памяти, однако их объем очень ограничен. Все блоки могут взаимодействовать с тремя видами памяти, общими для всего графического процессора: глобальной

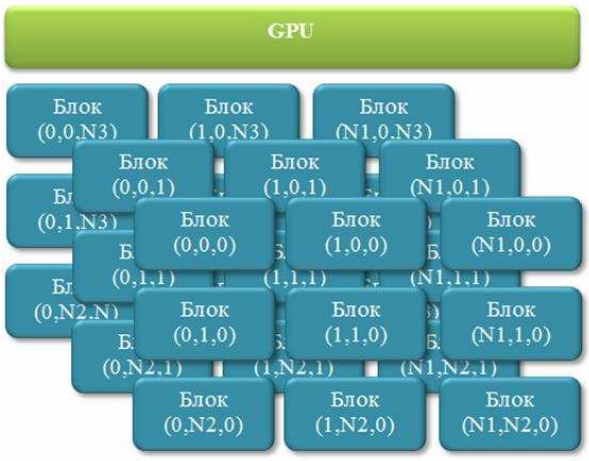

Рис. 3. Структура сетки

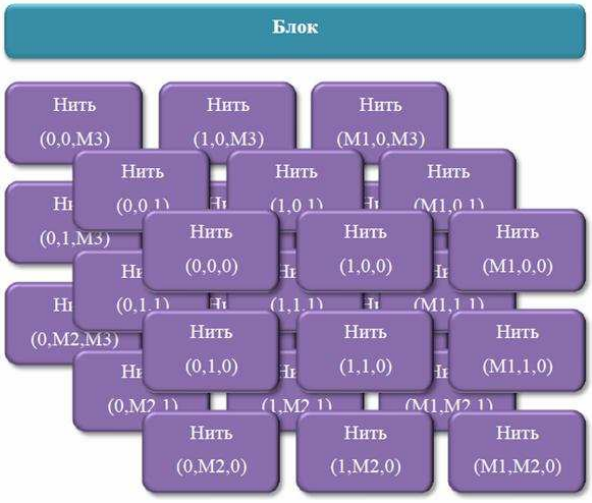

Рис. 4. Структура блока

(global memory), константной (constant memory) и текстурной (texture memory). Хотя глобальная память во много раз медленнее общей памяти блока и, тем более, регистровой памяти нити, она в несколько раз быстрее оперативной памяти компьютера; к тому же она позволяет хранить значительные объемы данных.

На данном этапе на графический процессор вынесены только операции матричной алгебры как самые ресурсоемкие — умножение матрицы на вектор, умножение матрицы на матрицу и обращение матрицы. На рис. 5 представлено соотношение времен выполнения перечисленных операций на CPU и GPU (тестирование проводилось для матрицы размером 630×630 — именно такая матрица соответствует сети, построенной в данной работе).

Перенос ресурсоемких операций на графический процессор позволил сократить время одной эпохи обучения почти в 16 раз (в работе [27] было достигнуто ускорение в 6 раз). Заметим, что с ростом размерности задачи время вычисления на графическом процессоре растет гораздо медленнее времени вычисления на центральном процессоре, особенно для хорошо распараллеливаемых задач. Например, это хорошо видно из рис. 6, где приведены графики, полученные при выполнении умножения матрицы на матрицу на CPU и GPU.

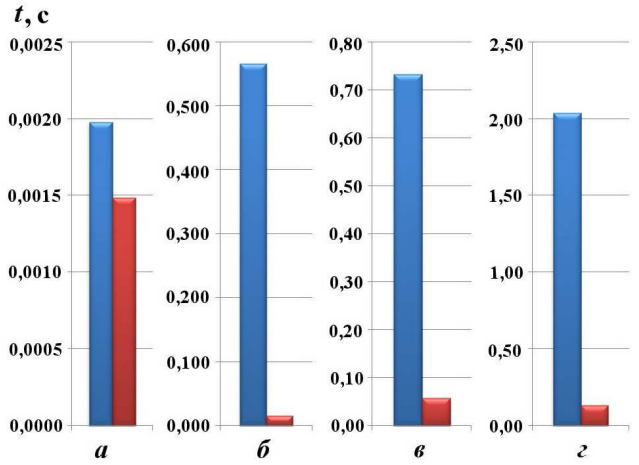

Рис. 5. Времена выполнения операций на CPU (1-й столбец) и GPU (2-й столбец): *а* — умножение матрицы на вектор; *б* — умножение матрицы на матрицу; *в* — обращение матрицы; *г* — эпоха обучения

Результаты

В рамках данной работы была построена нейронная сеть, обучаемая по алгоритму на основе метода Левенберга—Марквардта, описанному выше. Исходными данными для практической задачи послужили растровые полутоновые изображения размером 64×64 (в пикселях). Изображения разбивались на части размером 8×8 и на вход нейронной сети подавались средние яркости этих частей — всего 64 яркости. Поскольку сеть обучалась распознаванию изображений 26 латинских букв и 10 цифр, в качестве желаемых выходов сети в обучающей выборке присутствовали двоичные представления номеров эталонов изображений в выборке. Шести выходных нейронов оказалось достаточно для кодирования всех 36 эталонов. Таким образом, входной слой нейронной сети содержал 64 нейрона, скрытый — 9, а выходной — 6.

Для формирования тестового множества и тестирования сети в целом были созданы зашумленные варианты эталонных изображений. В качестве шума был выбран импульсный шум (в

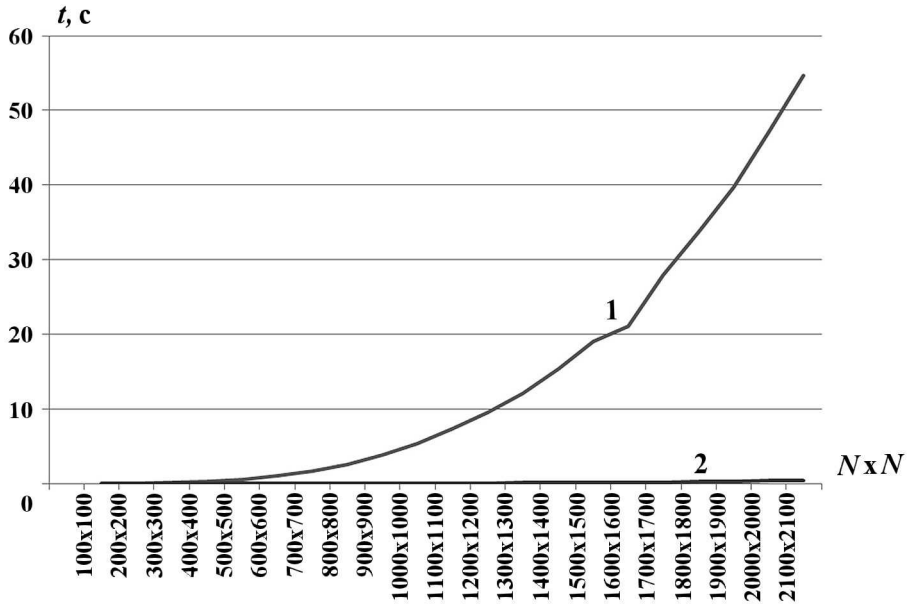

Рис. 6. Относительный рост времени выполнения умножения матрицы на матрицу на CPU (1) и GPU (2) при увеличении размерности задачи

англоязычной литературе чаще используют термин *salt and pepper noise*) [28]. Под уровнем шума понимается доля пикселей, яркости которых были заменены на случайные яркости, распределенные равномерно, т. е. уровень шума в 50 % означает, что яркости 50 % всех пикселей изображения заменены на случайные. Для каждого эталона были созданы 500 зашумленных образцов (по 5 образцов на каждый уровень шума).

Для обучения нейронной сети было сформировано 6 различных обучающих выборок:

- 0 — включает в себя только незашумленные эталоны;
- 0-5 — незашумленные эталоны и зашумленные с уровнем шума 5 %;
- 0-10 — незашумленные эталоны и зашумленные с уровнем шума 10 %;
- 0-20 — незашумленные эталоны и зашумленные с уровнем шума 20 %;
- 0-5-10 — незашумленные эталоны и зашумленные с уровнем шума 5 и 10 %;
- 0-10-20 — незашумленные эталоны и зашумленные с уровнем шума 10 и 20 %.

Под ошибкой нейронной сети подразумевается средняя доля неправильно распознанных образцов. Образец считается правильно распознанным, если среднеквадратичная ошибка нейронной сети на выходе при его распознавании не превышает наперед заданного порога. В данной работе использовался порог 0,3. При таком пороге всегда можно выделить такие выходные нейроны, у которых ошибка на выходе существенно превышает ошибки других нейронов. Результаты обучения нейронной сети на перечисленных выборках приведены в таблице.

Результаты тестирования нейронной сети, обученной по каждой из указанных выше обучающих выборок, приведены на рис. 7 (см. также цветную вкладку), где R — процент правильно распознанных тестовых примеров; NL — уровень шума.

Таблица

Количество эпох обучения нейронной сети

Выборка	Среднее число эпох
0	55
0-5	72
0-10	36
0-20	48
0-5-10	44
0-10-20	34

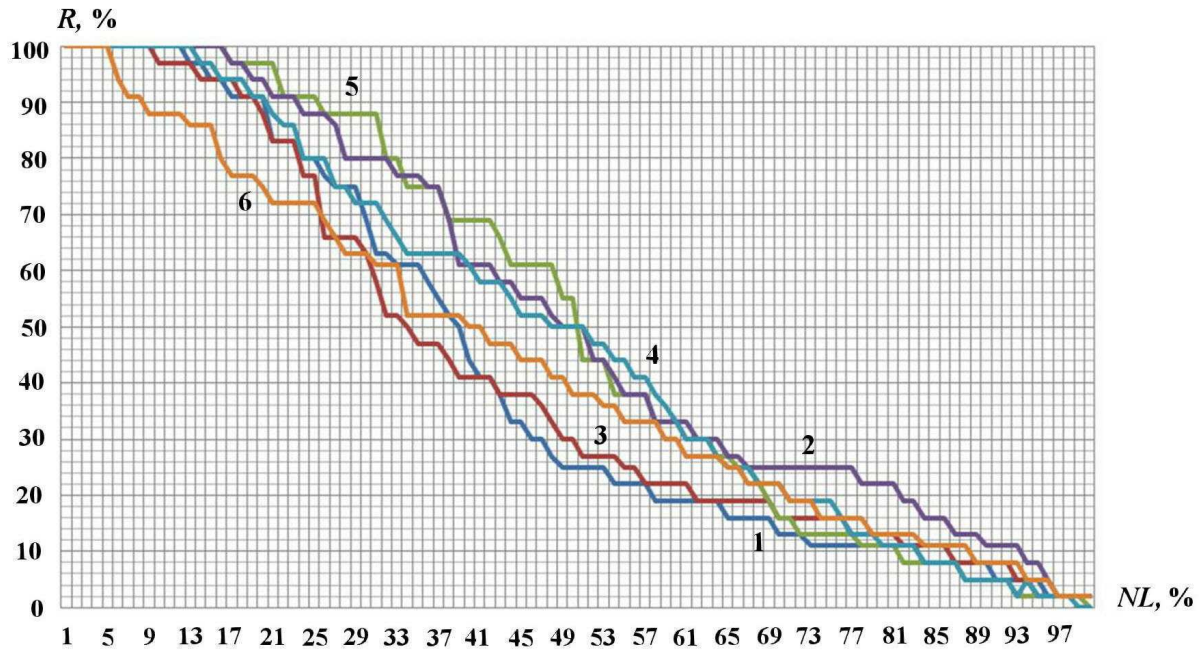

Рис. 7. Динамика тестирования нейронной сети, обученной по выборкам 0 (1), 0-5 (2), 0-10 (3), 0-20 (4), 0-5-10 (5), 0-10-20 (6)

Из графика соотношения доли правильно распознанных примеров и уровня шума видно, что нейронная сеть, обученная на выборке 0-10, при шуме до 50 % дает большее количество правильно распознанных образцов. Похожие результаты достигнуты, например, в работе [1].

Построенная нейронная сеть способна распознавать образцы с уровнем шума до 32—34 % с ошибкой не выше 20 % (доля правильно распознанных образцов не ниже 80 %). Если же повысить приемлемую ошибку до 30 % (доля правильно распознанных образцов не ниже 70 %), то сеть способна распознавать образцы с уровнем шума до 40 %.

В качестве направлений для дальнейшего развития предполагается перенос расчета всей нейронной сети на графический процессор и построение ансамбля из нескольких нейронных сетей для повышения качества распознавания.

Список литературы

1. *Mashor M. Y., Sulaiman S. N.* Recognition of noisy numerals using neural network // Int. J. of the Computer, the Internet and Management. 2001. Vol. 9, No 3. P. 158—164.
2. *Boureau Y.-L., Bach F., LeCun Y., Ponce J.* Learning mid-level features for recognition // Proc. of IEEE Conf. on Comp. Vision and Pattern Recognition. 2010. P. 2559—2566.
3. *Lowe D. G.* Distinctive image features from scale-invariant keypoints // Int. J. of Comp. Vision. 2004. Vol. 60(2). P. 91—110.
4. *Галушкин А. И.* Синтез многослойных систем распознавания образов. М.: Энергия, 1974.
5. *Werbos P. J.* Beyond regression: new tools for prediction and analysis in the behavioral sciences // PhD thesis. Harward University, 1974.
6. *Rumelhart D. E., Hinton G. E., Williams R. J.* Learning internal representations by error propagation // Parallel Distributed Processing. 1986. Vol. 1. P. 318—362.
7. *Minai A. A., Williams R. D.* Acceleration of back-propagation through learning rate and momentum adaptation // Proc. of Int. Joint Conf. on Neural Networks. San Diego, 1990. Vol. 1. P. 676—679.
8. *Jacobs R. A.* Increased rates of convergence through learning rate adaptation // Neural Networks. 1988. Vol. 1, No 4. P. 295—308.
9. *Salvetti A., Wilamowski B.* Introducing stochastic process within the backpropagation algorithm for improved convergence, presented at ANNIE'94 // Artificial Neural Networks in Engineering. St. Louis, Missouri, USA. November 13—16, 1994.
10. *Balakrishnan K., Honavar V.* Improving convergence of back propagation by handling flat-spots in the output layer // Second Int. Conf. on Artificial Neural Networks. Brighton, 1992.
11. *Parekh R., Balakrishnan K., Honavar V.* An empirical comparison of flat-spot elimination techniques in back-propagation networks // Proc. of Third Workshop on Neural Networks WNN'92. Auburn, 1992. P. 55—60.
12. *Torvik L., Wilamowski B. M.* Modification of the backpropagation algorithm for faster convergence, presented at 1993 International Simulation Technology Multiconference // Proc. of Workshop on Neural Networks WNN'93. San Francisco, November 7—10, 1993. P. 191—194.
13. *Battiti R.* First- and second-order methods for learning: between steepest descent and Newton's method // Neural Computation. 1992. Vol. 4, No 2. P. 141—166.
14. *Charalambous C.* Conjugate gradient algorithm for efficient training of artificial neural networks // IEEE Proceedings. 1992. Vol. 139, No 3. P. 301—310.
15. *Hagan M. T., Menhaj M.* Training feedforward networks with the Marquardt algorithm // IEEE Transactions on Neural Networks. 1994. Vol. 5, No 6. P. 989—993.
16. *Levenberg K.* A method for the solution of certain non-linear problems in least squares // The Quarterly of Appl. Math. 1944. Vol. 2. P. 164—168.
17. *Marquardt D.* An algorithm for least-squares estimation of nonlinear parameters // SIAM J. on Appl. Math. 1963. Vol. 11(2). P. 431—441.
18. *Andersen T. J., Wilamowski B. M.* A modified regression algorithm for fast one layer neural network training // Proc. of World Congress of Neural Networks. Washington, July 17—21, 1995. Vol. 1. P. 687—690.

19. *Xu J., Ho D. W. C., Zheng Y.* A constructive algorithm for feedforward neural networks // Proc. of Control Conference. Shanghai, China. July 20—23, 2004. Vol. 1. P. 659—664.
20. *Wilamowski B. M., Chen Y., Malinowski A.* Efficient algorithm for training neural networks with one hidden layer, dept. of EE // Proc. of Int. Joint Conf. of Neural Networks. Washington, DC, USA, 1999. Vol. 3. P. 1725—1728.
21. *MacKay D.* A practical bayesian framework for backpropagation networks // Neural Comp. 1992. Vol. 4. P. 448—472.
22. *MacKay D.* Bayesian methods for neural networks — FAQ. http://www.inference.phy.cam.ac.uk/mackay/Bayes_FAQ.html.
23. *Poland J.* On the robustness of update strategies for the bayesian hyperparameter alpha. http://www.alg.ist.hokudai.ac.jp / jan/alpha.pdf.
24. *Nguyen D., Widrow B.* Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights // Proc. of the Int. Joint Conf. on Neural Networks. Washington, DC, USA, 1990. Vol. 3. P. 21—26.
25. CUDA. Руководство по началу работы (Windows). http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc /CUDA_C_Get-ting_Started_Windows.pdf.
26. CUDA C. Руководство программиста. http://developer.download.nvidia.com/compute /DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf
27. *Изотов П. Ю., Суханов С. В., Головашкин Д. Л.* Технология реализации нейросетевого алгоритма в среде CUDA на примере распознавания рукописных цифр // Компьютерная оптика. 2010. Т. 34, № 2. С. 243—252.
28. *Gonzalez R., Woods R.* Digital Image Processing. Addison-Wesley Publishing Company, 1992.

Статья поступила в редакцию 24.01.11.