УДК 004.032.26

УСКОРЕНИЕ ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ ДЛЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ С ПОМОЩЬЮ ТЕХНОЛОГИИ NVIDIA CUDA

А. А. Ферцев

Мордовский государственный университет им. Н. П. Огарева,
430005, Саранск, ул. Большевистская, 68

E-mail: a.fertsev@rm.volga.rt.ru

Представлена реализация нейронной сети, обучаемой по алгоритму на основе метода Левенберга-Марквардта. С помощью технологии NVIDIA CUDA обучение построенной нейронной сети ускорено почти в 9 раз. Построенная нейронная сеть применена для распознавания зашумленных изображений.

Ключевые слова: *распознавание изображений, нейронная сеть, метод Левенберга–Марквардта, графический процессор, CUDA.*

1. Введение. Одним из перспективных методов исследования земной поверхности, океана, атмосферы является технология LIDAR (англ. Light Detection and Ranging). Это технология получения и обработки информации об удалённых 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.

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

2. Модель процесса распознавания образов. Рассмотрим два пространства: пространство характеристик и тематическое пространство. В пространстве характеристик находится вся информация, прямая и косвенная, которую мы можем получить об интересующем нас 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. 1.

Александр Александрович Ферцев, аспирант, каф. прикладной математики.

Рис. 1. Модель распознавания образов

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

В рамках данной статьи мы будем рассматривать полутоновые растровые изображения. В качестве тематического пространства взят латинский алфавит и десять арабских цифр. В качестве характеристик изображения взяты средние яркости частей изображения размером 8 на 8 пикселей. Для простоты цифры и буквы начертаны одним и тем же шрифтом.

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 работе [2]. Это позволит избавиться от ограничения 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.

3. Метод Левенберга—Марквардта. Метод Левенберга—Марквардта был предложен как метод для решения задачи о наименьших квадратах. Этот метод был применен для обучения нейронных сетей, например в [3]. 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 [3] и [4] нейронную сеть можно представить в виде вектор-функции вектор-аргумента:

$$Y = Y(X, \theta), \tag{1}$$

где $X = (x_1, \ldots, x_n)$ — входные данные, $\theta = (\theta_1, \ldots, \theta_S$ — параметры сети, $Y = (y_1, \ldots, y_P)$ — выход сети. Тогда ошибка сети за одну эпоху будет выражаться формулой

$$F(Y) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{P}(y_{ij} - d_{ij})^2, \tag{2}$$

где d_{ij} — желаемый выход j–того выходного нейрона для i–того элемента обучающего множества. Пусть $E = (e_{11}, \ldots, e_{1P}, e_{N1}, \ldots, e_{NP})^\top$, где $e_{ij} = y_{ij} - d_{ij}$. Тогда формулу (1) можно переписать так:

$$F(y) = E^\top E,$$

и матрица Якоби для уравнения (2) будет иметь вид

$$J = \begin{bmatrix} \tilde{J}_1 \\ \vdots \\ \tilde{J}_N \end{bmatrix}, \quad \tilde{J}_t = \begin{bmatrix} \frac{\partial e_1(t)}{\partial \theta_1} & \frac{\partial e_1(t)}{\partial \theta_2} & \cdots & \frac{\partial e_1(t)}{\partial \theta_S} \\ \frac{\partial e_2(t)}{\partial \theta_1} & \frac{\partial e_2(t)}{\partial \theta_2} & \cdots & \frac{\partial e_2(t)}{\partial \theta_S} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_P(t)}{\partial \theta_1} & \frac{\partial e_P(t)}{\partial \theta_2} & \cdots & \frac{\partial e_P(t)}{\partial \theta_S} \end{bmatrix}. \tag{3}$$

Согласно [3], приращение параметров сети следует искать в виде решения уравнения

$$(J^\top J + \lambda I)\triangle\theta = J^\top E, \tag{4}$$

где J — матрица Якоби, полученная по формуле (3), I — единичная матрица. Однако, как отмечает Марквардт в [5], если значение λ будет достаточно велико, то влияние аппроксимированной матрицы Гессе $J^\top J$ практически равно нулю. Чтобы избежать этого, в [5] предложено заменить единичную матрицу матрицей с диагональю аппроксимированной матрицы Гессе. Тогда формула (4) будет иметь вид

$$\left(J^\top J + \lambda \mathrm{diag}(J^\top J)\right)\triangle\theta = J^\top E. \tag{5}$$

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

1) рассчитать ошибку сети за одну эпоху по формуле (2);
2) рассчитать элементы матрицы Якоби по формуле (3);
3) решить уравнение (5) 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 увеличивают в 10 раз); перейти к шагу 3;
5) принять полученные значения параметров сети, уменьшить значение параметра λ (обычно уменьшают 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.

4. 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 нейроном (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 (1) примет вид

$$Y = Y(X,\, \theta) = \sigma\left(W^{(2)}\sigma(W^{(1)}X + B^{(1)}) + B^{(2)}\right),$$

где $W^{(1)}$ — матрица весов нейронов скрытого слоя, $W^{(2)}$ — матрица весов выходного слоя, $B^{(1)}$ — веса пороговых нейронов скрытого слоя, $B^{(2)}$ — веса пороговых нейронов выходного слоя. Тогда элементы матрицы Якоби будут вычисляться по следующим формулам.

Если θ_r есть вес нейрона скрытого слоя, то есть $\theta_r \equiv w^{(1)}_{i'j'}$, тогда

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

где $\overline{x}_k$ — выход k–того нейрона скрытого слоя, $\sigma'(\cdot)$ — значение производной функции активации в точке.

Если θ_r есть вес нейрона выходного слоя, то есть $\theta_r \equiv w^{(2)}_{i'j'}$, то

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

Если θ_r есть вес порогового нейрона скрытого слоя, то есть $\theta_r \equiv b^{(1)}_{i'}$, то

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

Если θ_r есть вес порогового нейрона выходного слоя, то есть $\theta_r \equiv b^{(2)}_{i'}$, то

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

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 путём);

2) алгоритм очень чувствителен к выбору начальных значений весов;
3) при работе с нейронными сетями большого размера алгоритм потребляет очень много памяти, а необходимость на каждом шаге обращать матрицы большого размера приводит к усложнению вычислительного процесса.

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

Для устранения первого недостатка был применен метод, предложенный МакКеем (David MacKay) в работе [6]. Суть этого метода состоит в переходе от поиска точки минимума среднеквадратической ошибки, рассчитываемой по формуле (2), к поиску минимума функции, выражаемой формулой

$$F(Y) = \alpha E_w + \beta E_d,$$

где E_d — ошибка сети (см. формулу (2)), E_w — среднеквадратичная сумма параметров сети, α и β — гиперпараметры. В результате алгоритм стремится не только минимизировать ошибку сети, но и не допустить неограниченного роста её параметров. В данной статье для расчёта гиперпараметров используются формулы, предложенные Поландом (Jan Poland) в работе [7]:

$$\gamma = S - \alpha \operatorname{trace}\left([J^\top J]^{-1}\right), \quad \alpha = \frac{\gamma}{2E_w + \operatorname{trace}\left([J^\top J]^{-1}\right)}, \quad \beta = \frac{PN}{2E_d},$$

где $\operatorname{trace}(J^\top J)$ — сумма диагональных элементов матрицы, обратной к аппроксимированной матрице Гессе. 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 (Derrick Nguyen, Bernard Widrow) в работе [8]. На практике этот подход реализуется следующим образом:

– вначале 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.

Чтобы избежать потери нейронной сетью свойства генерализации и сократить число эпох обучения, был 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.

Как уже отмечалось, для уменьшения размерности входных данных, на входы нейронной сети в данной работе подаются средние яркости частей изображения размером 8 на 8 пикселей. Несмотря на это, обучение нейронной сети с помощью алгоритма на основе метода Левенберга—Марквардта является очень ресурсоемким процессом, особенно в области перемножения матриц и нахождения обратной матрицы.

В настоящее время вычисления на графических процессорах являются одним из наиболее динамично развивающихся направлений в области параллельных вычислений. Фактически, графический ускоритель с поддержкой CUDA — вычислительный кластер с множеством узлов и общей памятью. В отличие от центрального процессора графический процессор изначально предназначался для параллельных вычислений — построение трехмерных моделей, хотя и требует высокой производительности, очень хорошо распараллеливается. Вычислительная модель GPU на верхнем уровне представляет собой сетку (grid) размерности $N_1 \times N_2 \times N_3$. Каждый блок (block), в свою очередь, состоит из множества нитей (threads), производящих параллельные вычисления. Блок обладает общей памятью (shared memory), доступной всем нитям блока. Каждая нить обладает локальной (local memory) и регистровой (register memory, недоступна разработчику) памятью. Общая память и локальная память — быстрые виды памяти, однако их объём очень ограничен. Все блоки могут взаимодействовать с тремя видами памяти, общей для всего графического ускорителя: глобальной (global memory), константной (constant memory) и текстурной (texture memory) памятью. Хотя глобальная память во много раз медленнее общей памяти блока и, тем более, регистровой памяти нити, она в несколько раз быстрее оперативной памяти компьютера, и её объём позволяет хранить в ней значительные объёмы данных. На данном этапе на графический ускоритель вынесены только операции матричной алгебры как самые ресурсоёмкие: умножение матрицы на вектор, умножение матрицы на матрицу и обращение матрицы. Для умножения матриц и векторов использовались функции библиотеки CUBLAS, разработанной NVIDIA, — cublasSgemv и cublasSgemm. Для обращения матриц была разработана собственная реализация алгоритма обращения матриц на основе LU-разложения. В частности, было реализовано блочное LU-разложение. То есть, вначале матрица разбивалась на блоки и LU-разложение применялось к каждому блоку в отдельности, после чего LU-разложения блоков «собирались» в LU-разложение исходной матрицы. Полученные треугольные матрицы были использованы для поиска обратной матрицы путём решения системы уравнений $LUA^{-1} = I$, где I — единичная матрица, с помощью функции cublasStrsm библиотеки CUBLAS. Тестирование с целью выявить преимущества при использовании графического ускорителя проводилось на системе с конфигурацией, приведённой в табл. 1. Основная часть программного комплекса, реализующего построен-

Таблица 1

Характеристики тестовой платформы

Материнская плата	ASUS P5Q3
Центральный процессор	Intel Core 2 Quad Q9400 2.6 ГГц, 6 МБ L2
Оперативная память	4096 МБ DDR3, эффективная частота 1200 МГц
Графический ускоритель	NVIDIA GeForce 9600 GT, 64 CUDA-ядра, 512 МБ GDDR3
Операционная система	MS Windows 7 Enterprise x64
Библиотеки CUDA	4.0.17
Компилятор кода на GPU	MS Visual C++, 10.00.40219
Компилятор кода на CPU	Delphi Pascal Compiler, 14.0.3593

ную нейронную сеть, разработана в среде программированиz `Delphi 2010`. Часть кода, отвечающего за операции на графическом ускорителе, разработана в среде `MS Visual Studio 2010`. При реализации расчётов на центральном процессоре не использовались возможности процессора в области параллельных вычислений. Нейронной сети, построенной в данной работе, соответствует матрица Гессе размером 630×630 элементов. На рис. 2 представлено соотношение времени обучения такой нейронной сети на центральном процессоре и графическом ускорителе.

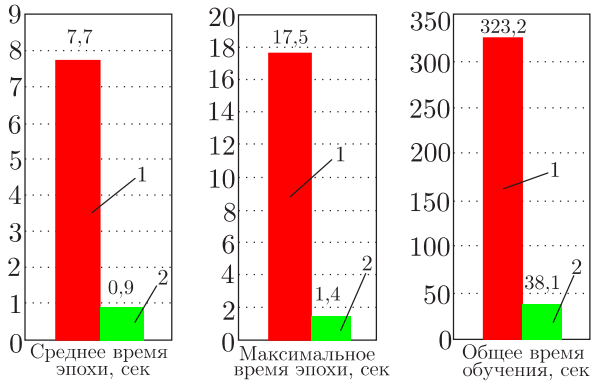

Рис. 2. Соотношение времени обучения: 1 — `CPU`; 2 — `GPU`

Таким образом, перенос ресурсоемких операций на графический ускоритель позволил сократить время одной эпохи обучения почти в 9 раз, а общее время обучения — в 8,5 раз (в работе [9] было достигнуто ускорение в 6 раз).

5. 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 на 64 пикселя. Изображения разбивались на части размером 8 на 8 пикселей и на вход нейронной сети подавались средние яркости этих частей — всего 64 яркости. Поскольку сеть училась распознавать изображения 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 англоязычной литературе чаще используют термин salt and pepper noise) [10]. Под уровнем шума понимается доля пикселей, чьи яркости были заменены на случайные яркости, распределенные равномерно, то есть уровень шума в 50 % означает, что яркости 50 % всех пикселей изображения были 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 (см. табл. 2).

Таблица 2

Состав обучающих выборок и число эпох обучения

Выборка	0	0–5	0–10	0–20	0–5–10	0–10–20
Всего элементов	36	72	72	72	108	108
Элементов с шумом 0 %	36	36	36	36	36	36
Элементов с шумом 5 %	0	36	0	0	36	0
Элементов с шумом 10 %	0	0	36	0	36	36
Элементов с шумом 20 %	0	0	0	36	0	36
Среднее число эпох	55	72	36	48	44	34

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 частности, в работе использовался порог 0,3. При таком пороге всегда можно выделить те выходные нейроны, чей выход существенно превышает выход других нейронов. Обучение останавливалось, как только ошибка нейронной сети на тестовой выборке становилась меньше 0,1. Для каждой обучающей выборки проводилось 10 итераций обучения. Результаты обучения нейронной сети на перечисленных выборках приведены в последней строчке табл. 2.

Результаты тестирования нейронной сети, обученной по каждой из приведенных обучающих выборок, приведены на рис. 3.

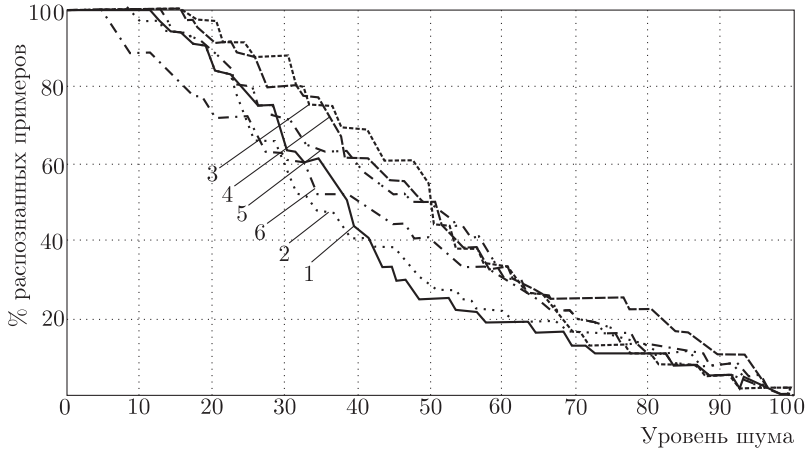

Рис. 3. Результаты тестирования: 1 — выборка 0; 2 — выборка 0–5; 3 — выборка 0–10; 4 — выборка 0–20; 5 — выборка 0–5–10; 6 — выборка 0–10–20

Таким образом, 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 %.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Mashor M. Y., Sulaiman S. N.* Recognition of Noisy Numerals using Neural Network // *IJCIM*, 2001. Vol. 9, no. 3. Pp. 158–164.
2. *Boureau Y.-L., Bach F., LeCun Y., Ponce J.* Learning mid-level features for recognition / In: *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. Pp. 2559–2566.
3. *Hagan M. T., Menhaj M.* Training feedforward networks with the Marquardt algorithm // *IEEE Transactions on Neural Networks*, 1994. Vol. 5, no. 6. Pp. 989–993.
4. *Wilamowski B. M., Chen Y., Malinowski A.* Efficient algorithm for training neural networks with one hidden layer / In: *International Joint Conference on Neural Networks (IJCNN '99)*. Vol. 3, 1999. Pp. 1725–1728.
5. *Marquardt D.* An Algorithm for Least-Squares Estimation of Nonlinear Parameters // *SIAM J. Appl. Math.*, 1963. Vol. 11, no. 2. Pp. 431–441.
6. *David J. C. MacKay* A practical Bayesian framework for backpropagation networks // *Neural Computation*, 1992. Vol. 4, no. 3. Pp. 448–472.
7. *Poland J.* On the Robustness of update strategies for the Bayesian hyperparameter alpha, available on: `http://www-alg.ist.hokudai.ac.jp/~jan/alpha.pdf`, 2001.
8. *Nguyen D., Widrow B.* Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights / In: *International Joint Conference on Neural Networks*. Vol. 3, 1990. Pp. 21–26.
9. *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. [*Izotov P. Yu., Sukhanov S. V., Golovashkin D. L.* Technology of implementation of neural network algorithm in CUDA environment at the example of handwritten digits recognition // *Komp'yuternaya optika*, 2010. Vol. 34, no. 2. Pp. 243–252].
10. *Gonzalez R. C., Woods R. E.* Digital Image Processing. Boston, MA: Addison-Wesley Publishing Company, 1992. 528 pp.

Поступила в редакцию 25/VIII/2011;
в окончательном варианте — 30/XI/2011.

MSC: 62M45; 94A08

NEURAL NETWORK TRAINING ACCELERATION USING NVIDIA CUDA TECHNOLOGY FOR IMAGE RECOGNITION

A. A. Fertsev

Mordovian State University by N. P. Ogarev,
68, Bolshevistskaya st., Saransk, 430005, Russia

E-mail: `a.fertsev@rm.volga.rt.ru`

In this paper, an implementation of neural network trained by algorithm based on Levenberg-Marquardt method is presented. Training of neural network increased by almost 9 times using NVIDIA CUDA technology. Implemented neural network is used for the recognition of noised images.

Key words: *image recognition, neural networks, Levenberg–Marquardt method, graphics processing unit, CUDA.*

Original article submitted 25/VIII/2011;
revision submitted 30/XI/2011.

Alexander A. Fertsev, Postgraduate Student, Dept. of Applied Mathematics.