УДК 004

Создание демонстрационной системы распознавания рукописных символов в приложении Brain Simulator

Лопатин Дмитрий Константинович
Приамурский государственный университет им. Шолом-Алейхема
студент

Аннотация
В статье рассмотрено создание демонстрационной системы распознавания рукописных символов в приложении Brain Simulator**.** Наглядно показано как обучается и тестируется нейронная сеть.
Ключевые слова: нейронная сеть, распознавание, Brain Simulator, рукописного.

Creation of demonstration system of recognition of hand-written characters in Brain Simulator application

Lopatin Dmitry Konstantinovich
Sholom Aleichem State University
student

Abstract
In article creation of demonstration system of recognition of hand-written characters in Brain Simulator application is considered. It is visually shown as the neural network studies and tested.
Keywords: neural network, recognition, Brain Simulator, hand-written.

Проблема распознавания образов, возникшая первоначально в связи с необходимостью решать задачи зрительного анализа, сегодня поставила перед создателями автоматизированных систем новую крупномасштабную задачу – ввод огромных объемов информации с бумажных документов в компьютер. С каждым годом электронный документооборот растёт. Многие документы представляют огромную важность для человечества, но, как известно, в прошлом, многие документы были написаны «вручную», когда ещё не было печатных машинок или компьютеров. Бумага не самый надёжный носитель с течением времени. В электронном виде данные удастся сохранить надолго. Поэтому применение технологии распознания рукописного текста для оцифровки данных является актуальным. Искусственные нейронные сети представляют собой математическую модель функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Теоретические основы программирования таких нейронных сетей, описываются во многих работах [1, 2, 3, 4]

Для создания нейронной сети воспользуемся приложением Brain Simulator.

Проект Brain Simulator представляет собой приложение, библиотеку и фреймворк для эффективной разработки и тестирования нейронных моделей. Его возможности позволяют:

- реализовать свою конкретную модель или алгоритм в качестве узла, который можно использовать в более крупной архитектуре;
- имитировать его поведение в различных средах (называемых мирами (worlds), которые предоставляют источники данных и выходы/исполнительные механизмы);
- эффективно настраивать параметры алгоритма;
- визуализировать свои данные (входы, выходы, внутреннюю память) с различными вариантами рендеринга;
- внедрите свои собственные визуализаторы (называемые наблюдателями) и миры, если необходимо;
- создавать проекты со сложными архитектурами, сохранять, загружать и экспортировать их состояния.

Для построения нашей сети воспользуемся средой MNISTWorld. Эта среда предоставляет инструменты для работы с базой данных MNIST. База данных MNIST (сокращение от "Mixed National Institute of Standards and Technology") – объёмная база данных образцов рукописного написания цифр. База данных является стандартом, предложенным Национальным институтом стандартов и технологий США с целью калибровки и сопоставления методов распознавания изображений с помощью машинного обучения, в первую очередь на основе нейронных сетей. Данные состоят из заранее подготовленных примеров изображений, на основе которых проводится обучение и тестирование систем. В ней содержится 60000 изображений для обучения и 10000 изображений для тестирования.

Данные для обработки предоставляются в виде пары: изображение – значение.

При построении будут использоваться следующие узлы (nodes):

1. NeuralNetworkGroup – обязательный элемент для размещения последовательностей слоёв и узлов, работающих с нейронными сетями.
2. HiddenLayer – скрытый слой. Наиболее часто используемый слой в нейронных сетях. Он принимает входные данные и подает на другой слой.
3. OutputLayer – сыходной слой. Выходной уровень принимает цель как входную информацию и автоматически масштабирует нейроны, чтобы соответствовать цели.
4. CSharpNode – программируемый пользователем узел с настраиваемым количеством входов и выходов.

Построение первого уровня сети изображено на рисунке 1.

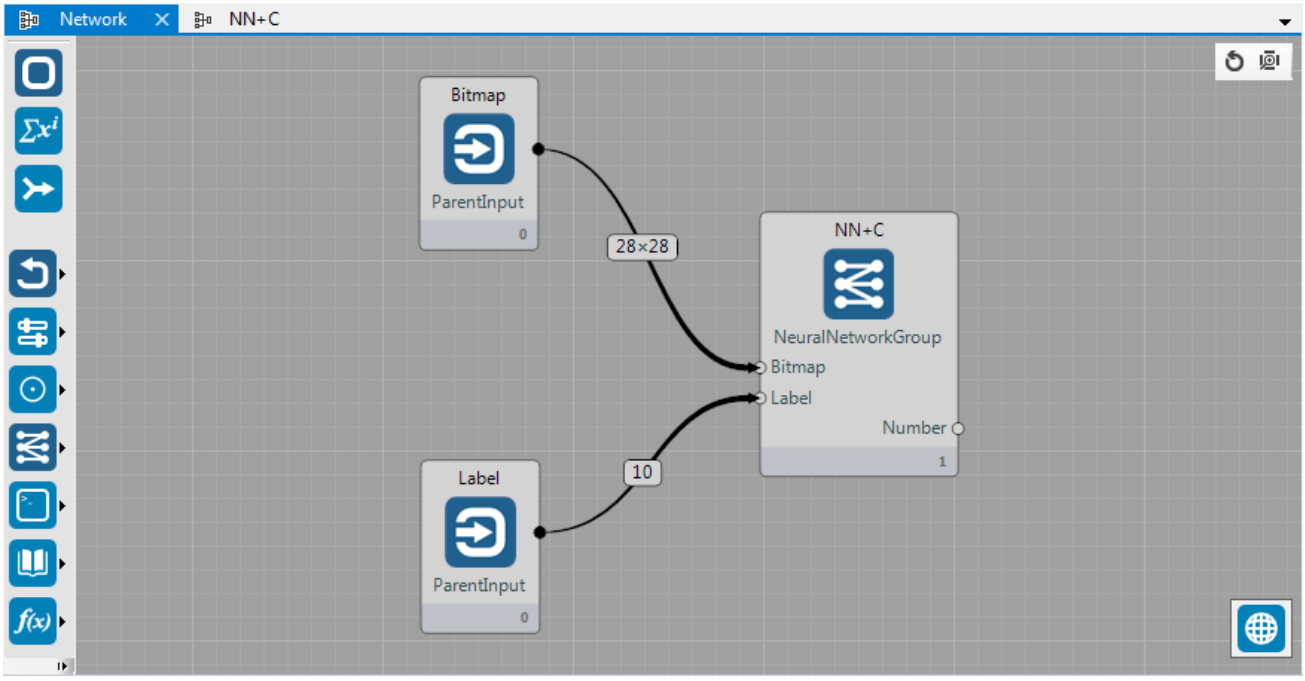

Рисунок 1. Общий вид сети

Здесь мы видим следующие элементы:

1. Bitmap – является источником изображений.
2. Label – является источником значений соответствующим изображениям, полученным от узла Bitmap.
3. NeuralNetworkGroup – является группой элементов. На вход получает пару изображение-значение и обрабатывает её внутри группы. Опционально может иметь выход.

Структура вычислительной группы элементов изображена на рисунке 2.

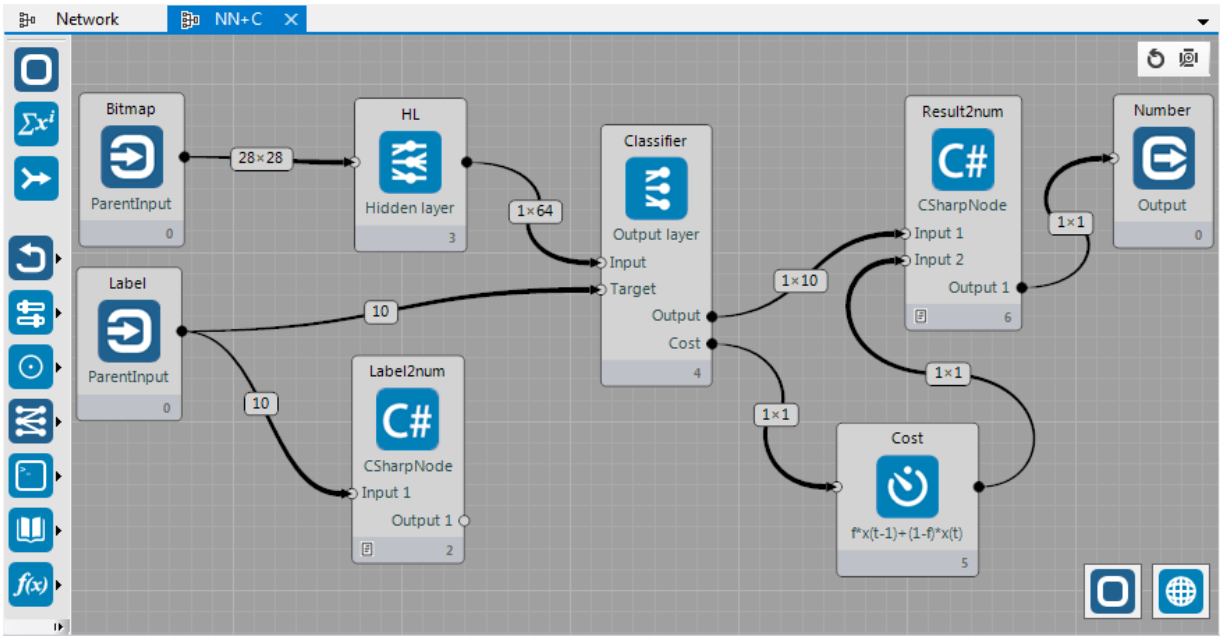

Рисунок 2. Структура вычислительной группы

Вычислительная группа состоит из следующих элементов:

1. Bitmap, Label и Output – наследуемые узлы от NeuralNetworkGroup с уровня выше.
2. Hidden layer (HL) – принимает изображение от узла Input и пропускает через указанное в параметрах количество нейронов. Выход от нейронов направляется в Outpul layer.
3. Outpul layer (Classifier) – принимает выход от узла Hidden layer и значение изображения от узла Label. В этом узле происходит перерасчёт весов связей нейронов и происходит попытка классификации входящего изображения, а также происходит вычисление точности классификации.
4. Label2num – узел типа CSharpNode. Программный код этого узла преобразовывает значение изображения в понятное человеку число. Код элемента приведён на рисунке 3.

Network NN+C Result2num Label2num

```
using System;
using GoodAI.Core.Utils;
using GoodAI.Core.Nodes;
using GoodAI.Modules.Scripting;

namespace Runtime
{
    public class Script
    {
        public static void Init(MyCSharpNode owner)
        {
            MyLog.DEBUG.WriteLine("Init called");
        }

        public static void Execute(MyCSharpNode owner)
        {
            MyLog.DEBUG.WriteLine("Execute called");

            float[] input = owner.GetInput(0).Host;
            float[] output = owner.GetOutput(0).Host;

            int i;
            for (i=0;i<=9;i++)
            {
                if (input[i]==1.00){break;}
            }
            output[0] = i;
        }
    }
}
```

Рисунок 3. Код узла Label2num

5. Result2num – узел типа CSharpNode. Программный код этого узла выводит значение изображения, с максимальной вероятностью совпадающее с входящим изображением, при этом вероятность рассчитывается Классификатором. Код элемента приведён на рисунке 4.4.

Network NN+C Result2num Label2num

```
1  using System;
2  using GoodAI.Core.Utils;
3  using GoodAI.Core.Nodes;
4  using GoodAI.Modules.Scripting;
5
6  namespace Runtime
7  {
8      public class Script
9      {
10         public static void Init(MyCSharpNode owner)
11         {
12             MyLog.DEBUG.WriteLine("Init called");
13         }
14
15         public static void Execute(MyCSharpNode owner)
16         {
17             MyLog.DEBUG.WriteLine("Execute called");
18
19             float[] input = owner.GetInput(0).Host;
20             float[] input2 = owner.GetInput(1).Host;
21             float[] output = owner.GetOutput(0).Host;
22
23             int i;
24             int im=-1;
25             float acm=0;
26             for (i=0;i<=9;i++)
27             {
28                 if (input[i]>=acm)
29                 {
30                     acm=input[i];
31                     im=i;
32                 }
33             }
34             output[0] = im;
35         }
36     }
37 }
```

Рисунок Код узла Result2num

Запускаем обучение. Сразу после начала обучения классификатор начинает отдавать первые результаты (Рисунок 5).

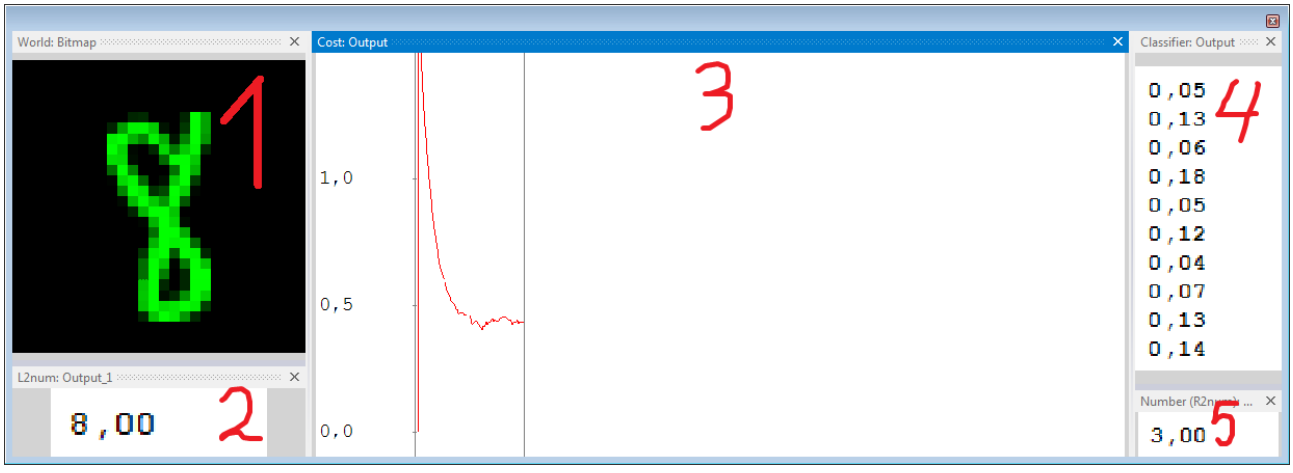

Рисунок 5. Визуализация процесса обучения и тестирования сети

Рассмотрим элементы визуализации:

1. Распознаваемое в данный момент изображение.
2. Значение распознаваемого изображения.
3. График погрешности распознавания изображения.

4. Список, каждая строка которого отражает вероятность того, что распознаваемое изображение соответствует цифре от 0 до 9.
5. Распознанное сетью значение изображения.

Обучение было остановлено после 100 изображений. На визуализации видно, что погрешность распознавания довольно велика, поэтому продолжим обучение.

После обучения почти на двух тысячах изображений мы видим, что погрешность распознавания уменьшилась более чем вдвое (Рисунок 6). Продолжим обучение.

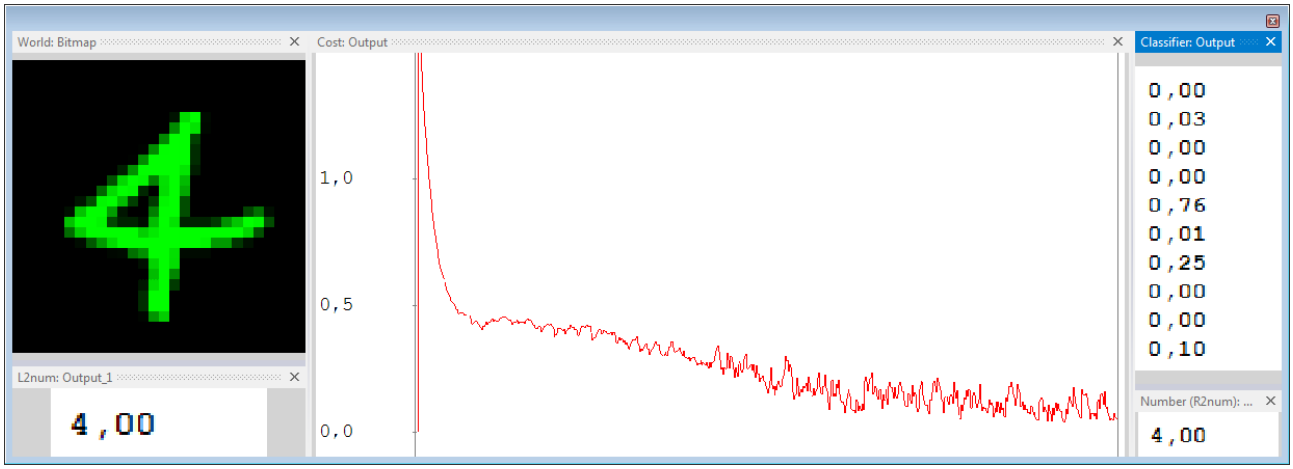

Рисунок 6. Визуализация процесса обучения и тестирования сети после тренировки на двух тысячах изображений

После обучения на двадцати тысячах изображений график погрешности стал более плавным – вся нейронная сеть стала ошибаться намного реже, что свидетельствует о том, что система распознаёт образы точнее (Рисунок 7).

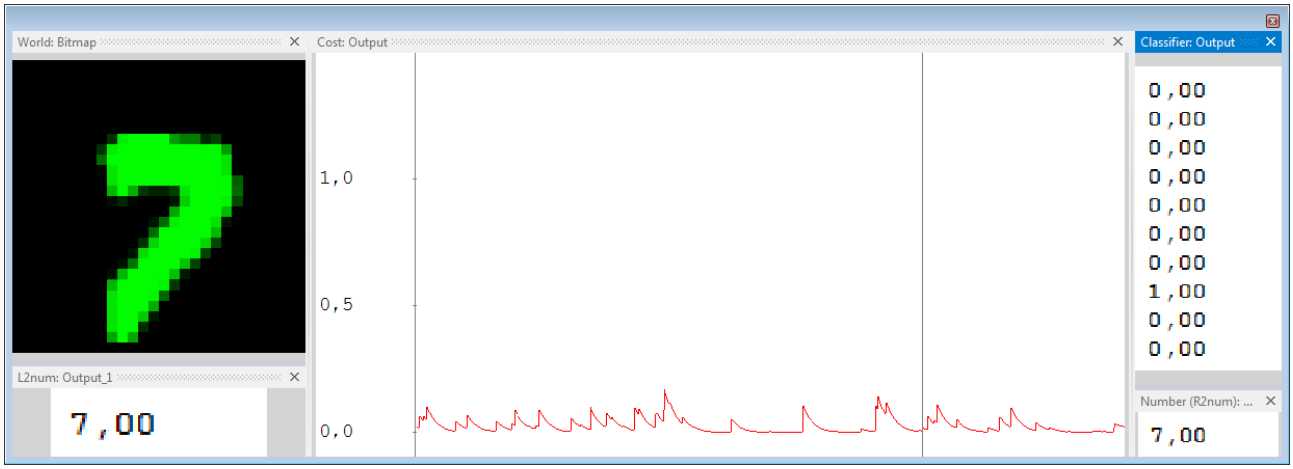

Рисунок 7. Визуализация процесса обучения и тестирования сети после тренировки на двадцати тысячах изображений

Приступим к тестированию сети. Для начала заменим тренировочный набор отправляемых в сеть данных на набор для тестирования (Рисунок 8).

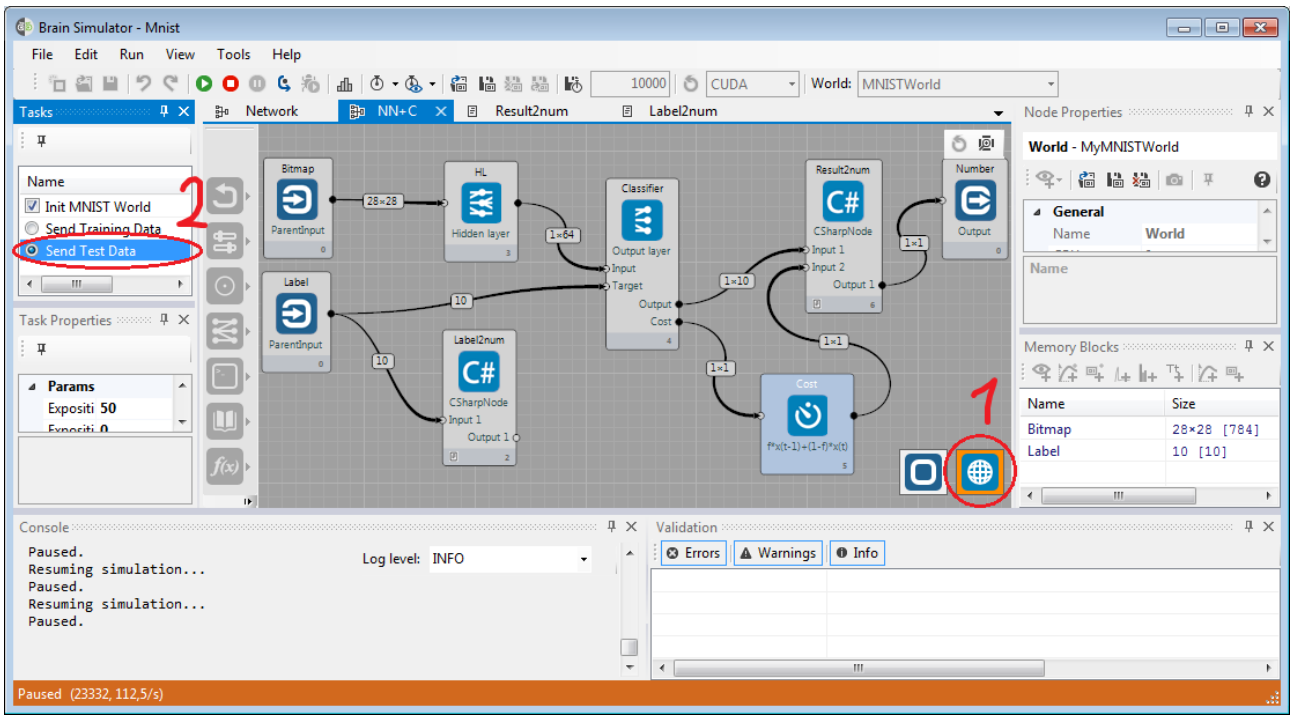

Рисунок 8. Переключение набора данных

Также отключим обновление весов в узлах Hidden layer и Output layer (Рисунки 4.9 и 4.10).

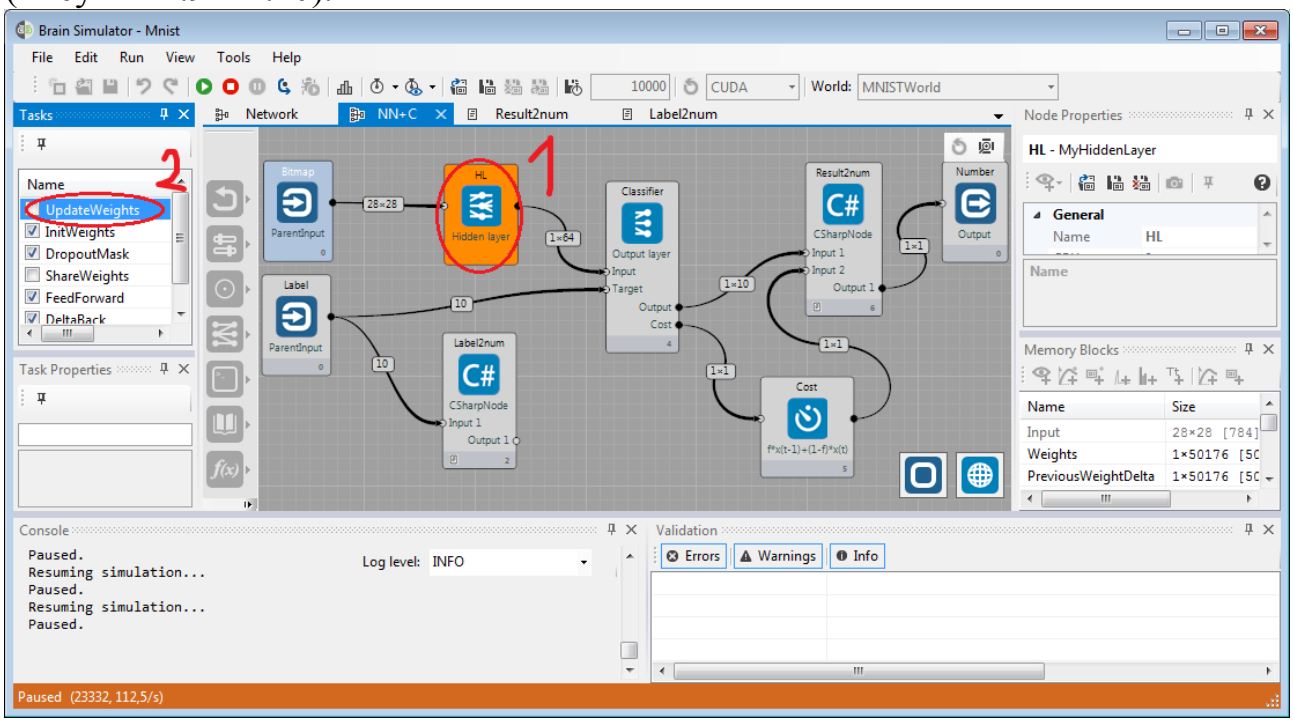

Рисунок 9. Отключение обновления весов на узле Hidden layer

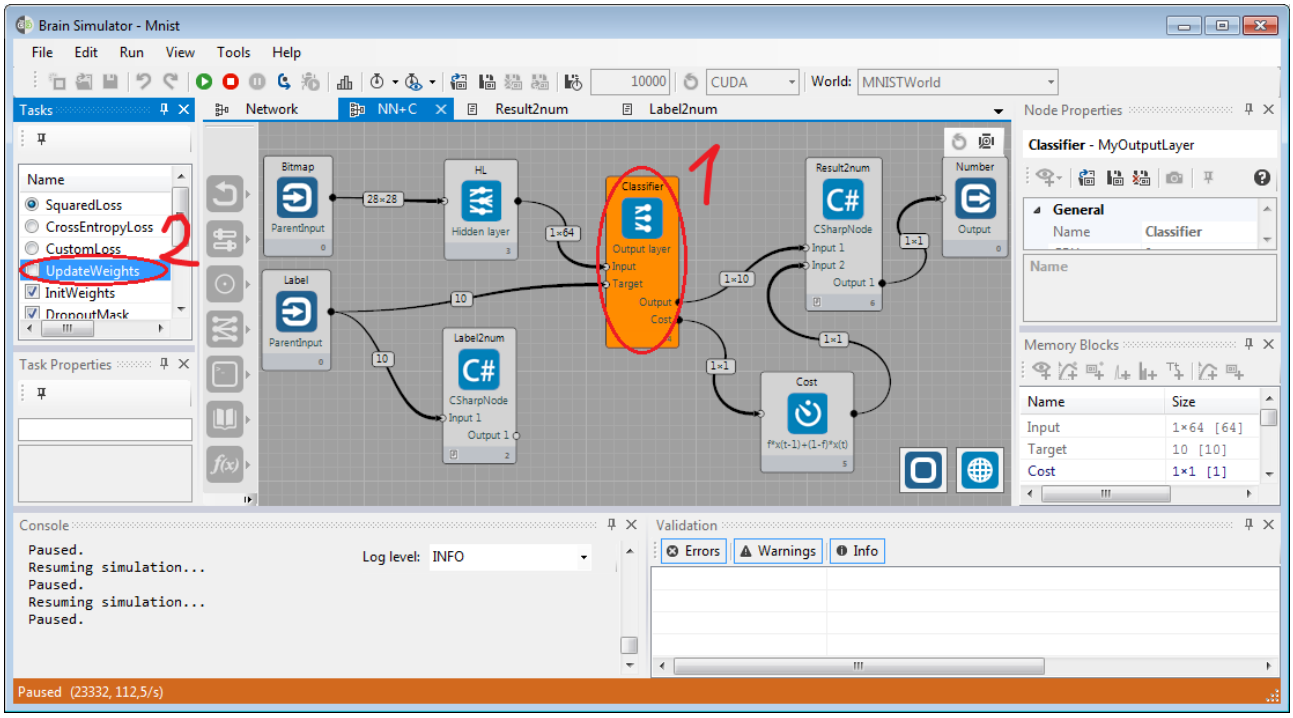

Рисунок 10. Отключение обновления весов на узле Output layer

Смена набора позволит проверить нейронную сеть на изображениях, с которыми она не сталкивалась при обучении, а отключение обновления весов предотвратит обучение сети на новых данных.

Теперь снова запустим сеть, но уже с другим набором данных. На рисунке 4.11 можно заметить, что система распознала в изображении как цифру 3 (62%) так и цифру 5 (69%).

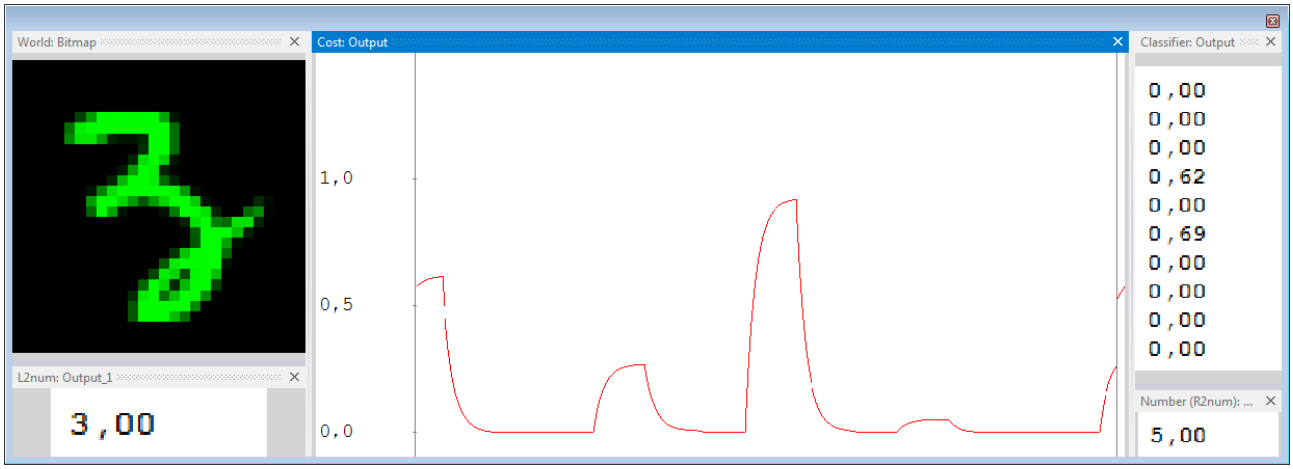

Рисунок 11. Некорректное распознавание цифры 3

На рисунке 4.12 мы видим, что система распознала в изображении в большей степени цифру 8 (73%), чем цифру 9 (18%).

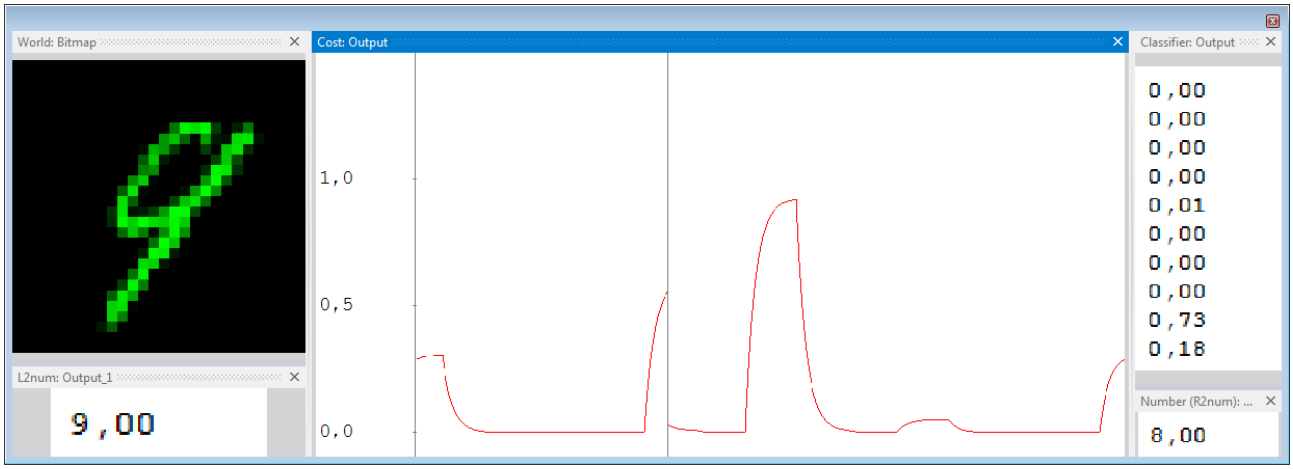

Рисунок 12. Некорректное распознавание цифры 9

На рисунке 4.13 мы видим, «неуверенное», но корректное распознавание цифры 9 в изображении.

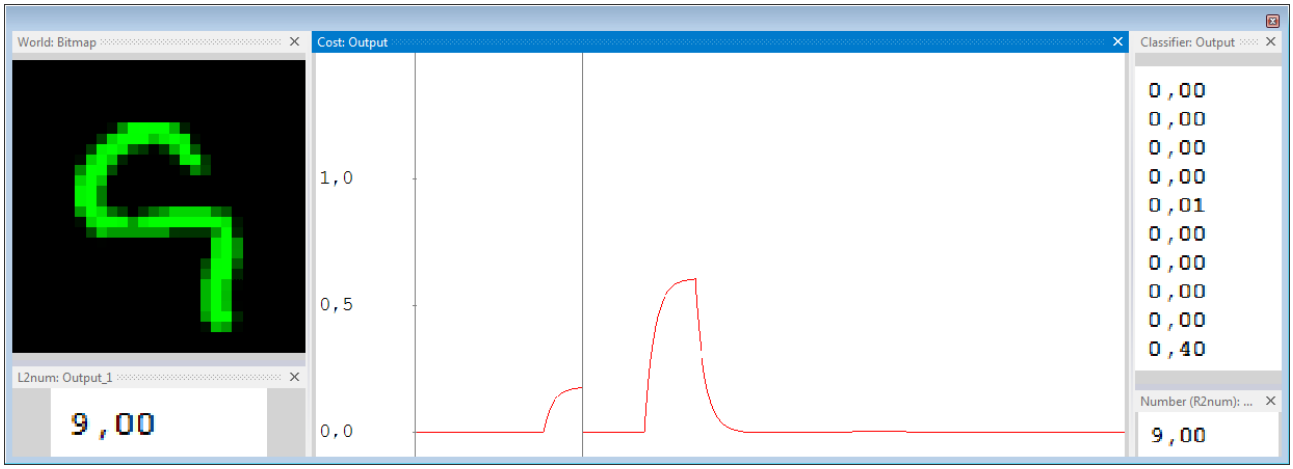

Рисунок 13. «Неуверенное» распознавание цифры 9

По результатам тестирования было установлено, что система обучена и корректно распознаёт большую часть неизвестных ранее данных.

Библиографический список

1. Борисова И.А. Методы решения задач распознавания образов комбинированного типа: дис. на соискание ученой степени канд. тех. наук. Новосибирск. 2008. 121 с.
2. Горский Н.Д. Распознавание рукописного текста: от теории к практике / Н.Д.Горский, В.А. Анисимов, Л.М.Горская. СПб.: Политехника, 1997. 126 с.
3. Демин А.А. Интеллектуальная интерактивная обучающая система «Электронная пропись» // Современная техника и технологии: Сборник трудов 14-ой Международной научно-практической конференции. Томск: Томский политехнический университет, 2008. С. 284–285.
4. Зиатдинов А.М., Емекеев А.А. Методы искусственного интеллекта при

распознавании объектов и образов // Ученые записки Альметьевского государственного нефтяного института. 2013. Том 11. №1. С. 180-184.
5. Миронов И.С., Скурлаев С.В. Распознавание образов при помощи нейронной сети URL: http://confonline.susu.ru/index.php?option=com_content&view=article&id=57:2011-05-06-04-36-21&catid=16:-2----&Itemid=18
6. Luo F-L., Unbehauen R. Applied Neural Networks for Signal Processing. Cambridge University Press. 1998.
7. Principe J.C., Euliano N.R., Lefebvre W.C. Neural and Adaptive Systems. Fundamentals Through Simulations. New York. John Wiley & Sons Inc. 2000.