Московский Государственный Университет имени М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики

Научно-технический отчет

«Распознавание нейронов на снимках с микроскопа»

Москва 2014

1. Описание и цели работы

В данной работе было проведено исследование проблемы распознавания нейронов на снимках с микроскопа, а также разработана библиотека распознавания нейронов на двумерных снимках с микроскопа. В библиотеке присутствует инструментарий для анализа, машинного обучения, обработки и сбора признаков.

Библиотека может быть встроена как в консольное приложение, так и в приложение с пользовательским интерфейсом. Первый вариант удобен для разработчиков и исследователей, второй - для конечного пользователя. Также был разработан ряд вспомогательных программ для создания и обработки ручной разметки нейронов. Код всех программ, а также история разработки лежит в открытом доступе по ссылке [1], скачать можно как с сайта, так и через систему контроля версий GIT.

На вход нам был дан набор снимков с микроскопа. На изображениях вручную размечаются нейроны. По размеченным областям делается выборка нейронов и фона. По каждому объекту выборки вычисляются признаки и производится обучение линейным классификатором [6]. В качестве признаков в данной работе используются гистограммы ориентированных градиентов HOG [5]. Линейный классификатор [6] выставляет каждому изображению число с плавающей точкой, которое сравнивается с пороговым, и в случае, если число больше порога, объект классифицируется как нейрон, если меньше – как фон. Варьируя порог, получаются различные значения ошибок первого и второго рода. По тестовой выборке строится ROC-кривая [4], которая показывает соотношение этих ошибок в результате варьирования параметра классификатора.

Распознавание нейронов делается методом скользящего окна [7] разных масштабов. Для каждого изображения скользящего окна вычисляются дескрипторы, и по ним делается классификация. На снимках с микроскопа параметр классификации в распознавании будет определять количество ложных детекций и количество пропусков нейронов. Так результаты автоматической разметки на данный момент не идеальные, делается корректировка разметки и по корректированной разметке по каждому нейрону вычисляются признаки.

2. Программный комплекс для распознавания

Код комплекса находится в репозитории по адресу:
https://github.com/Ccas-Recognition [1].

В репозитории находится несколько проектов:
1) https://github.com/Ccas-Recognition/Neurons_HoG_Classificatione - библиотека распознавания нейронов на двумерных снимках с микроскопа.
2) https://github.com/Ccas-Recognition/TrainingSampleMaker - программа для разметки нейронов на снимках.
3) https://github.com/Ccas-Recognition/SamplesCombination - программа для комбинирования ручной разметки, сделанной разными людьми.
4) https://github.com/Ccas-Recognition/TrainingImages - место, где хранятся актуальные размеченные данные.
5) https://github.com/Ccas-Recognition/NeuronsRecognition - реализация распознавания нейронов с пользовательским интерфейсом.

2.1. Программа для разметки нейронов на снимках

Название исполняемого файла TrainingSampleMaker.exe.
Параметры для запуска:

1) Путь к изображению
2) -fg dir_name_fg – Путь к папке, куда будут сохраняться выборка с нейронами (папка должна существовать)
3) -bg dir_name_bg – Путь к папке, куда будут сохраняться выборка с фоном (папка должна существовать)

В папки сохраняется xml файл с описанием путей и сами изображения выборки.
В bin/run.bat - пример запуска для нескольких изображений.
В приложении opencv отсутствует UI, поэтому все взаимодействие только на горячих клавишах и мышкой:

1) Горячая клавиша h показывает/убирает помощь
2) Клавиши w, a, s, d для навигации по изображению; e, q для zoom
3) Клавиши 1, 2, 3, 4 - подбор подходящего размера окна для нейронов (1 - 24 пикселя, 2 - 36 пикселя, 3 - 48 пикселя, 4 - 60 пикселя)
4) После разметки можно нажать на Ctrl+f - будет сгенерирована выборка фона
5) Ctrl+s - делает сохранение выборки, выборка фона будет сгенерирована автоматически
6) Enter - закрыть программу и сделать сохранение
7) Esc - закрыть программу без сохранения
8) Ctrl+R - удаляет всю разметку
9) Правая кнопка мыши используется для навигации по изображению
10) Левая кнопка мыши добавляет/удаляет размеченную область. Если под курсом нет размеченной области - она добавляется, если есть – удаляется
11) Ctrl+z - Удаляет последнюю разметку

В ОС Windows, если разметка для изображения уже сделана - то она подгружается и ее можно редактировать.
clean.bat - удаляет всю разметку для всех изображений

Проект написан на MS Visual Studio 2010 с билиотекой opencv 2.4.9. Скомпилированный для x32 системе есть в папке bin.

2.2. Формат размеченной выборки

Формат разметки изображения представлен в виде xml файла, сохраненного через opencv storage библиотеки opencv [2]. В тэге images через пробел записаны пути к фрагментам разметки. Информация о расположении прямоугольника записана в названии файла.
Пример xml-файла:

```
<?xml version="1.0"?>
<opencv_storage>
<images>
  "6_rnd12_fg_1002_1360_36_36.jpg"
  "6_rnd12_fg_1015_1231_24_24.jpg"
</images>
</opencv_storage>
```

В данном примере разметка состоит из двух прямоугольников 6_rnd12_fg_1002_1360_36_36.jpg и 6_rnd12_fg_1015_1231_24_24.jpg. Название объекта содержит в себе свойства, которые записываются через символ нижнее подчеркивание. Возьмем первый прямоугольник с названием 6_rnd12_fg_1002_1360_36_36 и распишем его свойства:

1) Название снимка с микроскопа без расширения (в примере «6»)
2) Универсальный идентификатор, уникальный для каждого снимка (в примере «rnd12»)
3) Метка класса: нейрон помечается как fg, фон – через bg (в примере «fg»)
4) Координата x, положения левого верхнего угла прямоугольника (в примере «1002»)
5) Координата y, положения левого верхнего угла прямоугольника (в примере «1360»)
6) Ширина прямоугольника (в примере «36»)
7) Высота прямоугольника (в примере «36»)

2.3. Программа для комбинирования ручной разметки

Название исполняемого файла SamplesCombination.exe.
Параметры для запуска:

1) -fg dir_name_fg – Путь к папке, в которой находятся xml файлы разметок нейронов
2) -bg dir_name_bg – Путь к папке, в которой находятся xml файлы разметок фона

На выходе будет скомбинирован xml файл разметки, в котором будут содержаться все прямоугольники. Все прямоугольники нейронов будут записаны в fg_imlist_combined.xml. Все прямоугольники фона будут записаны в bg_imlist_combined.xml.
В bin/run.bat - пример запуска для нескольких изображений.
Проект написан на MS Visual Studio 2010 с билиотекой opencv 2.4.9. Скомпилированный для x32 системе есть в папке bin.

2.4. Библиотека распознавания нейронов на двумерных снимках с микроскопа

Библиотека состоит из трех проектов:

1) Проект /vs2013/Neurons_HoG_Classification_Lib/ статической библиотеки, в которой находится весь функционал по обучению, обработки и распознаванию. Библиотека компилируется в файл Neurons_HoG_Classification_Lib.lib, интерфейс к реализовываемым в библиотеке функциям находится в файле src/ImageRecognition.h.

2) Проект /vs2013/ консольного приложение, который через командную строку обращается ко всем функциям статической библиотеки. Приложение компилируется в Neurons_HoG_Classification.exe. Состоит только из одного файла main.cpp, там же пример вызова процедур библиотеки Neurons_HoG_Classification_Lib.lib.
3) Проект vs2013/PythonProj для создания обучающей и тестовой выборки по всем размеченным изображениям нейронов и фона, а также для построения графиков ROC-кривых [4]. Проект написан на языке Python и представляет собой два скрипта: /vs2013/PythonProj/train_test_picker.py и /vs2013/PythonProj/roc_curve.py. В скрипте train_test_picker.py задается путь к директориям, в которых находятся изображения нейронов и фона. На выходе этого скрипта генерируется два текстовых файла для обучающей и тестовой выборки. Обучающая и тестовая выборка задается единым форматом: это текстовый файл, в каждой строке которого написан относительный путь к изображению нейрона или фона и через пробел записывается метка класса: 1 - нейрон, 0 – фон. Пример содержимого текстового файла такого формата:

   total_data/fg/Tv286_rnd913_fg_981_2212_48_48.jpg 1
   total_data/fg/30_rnd88_fg_1112_1064_24_24.jpg 1
   total_data/fg/27_rnd432_fg_1089_1309_24_24.jpg 1
   total_data/bg/31_rnd723_bg_961_1082_24_24.jpg 0
   total_data/bg/Tv265_rnd688_bg_3311_1889_36_36.jpg 0

   Данная выборка состоит из двух объектов-нейронов, и трех объектов фона.
   В скрипте roc_curve.py задается путь к значениям ROC-кривой и к значениям ошибок разметки. По этим значениям строится изображение кривой через библиотеку [3].

Так как программа делает множество операций, выбор и установка параметров нужной команды осуществляется через аргументы командной строки. Команды и параметры для запуска консольного приложения:

1) Обучение классификатора и сохранение модели.
   Параметры:
      -t -m model_finename -d data_set_filename -c context_filename
   Выход:
      В файл model_finename запишется модель классификатора, сделанной по обучающей выборке image_set_filename. В файле context_filename задаются параметры классификатора, лучше использовать стандартный файл context.ini, в корне библиотеки.

2) Тестирование классификатора на тестовой выборке и построение ROC-кривой [4].
   Параметры:
      -l -m model_finename -d data_set_filename
   Выход:
      В папке с исполняемым файлом создастся папка rocs, в которую запишется значения ROC-кривой predictROC.txt. Модель классификатора прочитается из текстового файла model_finename, а данные из файла image_set_filename.

3) Оптимизация параметра модели для распознавания через скользящее окно [7] и построение кривой ошибок распознавания.
   Параметры:
      -o -m model_finename -a image_set_filename

Выход:

1. В папке с исполняемым файлом создастся папка rocs, в которую запишется значения ROC-кривой [4] predictROC.txt. Модель классификатора прочитается из текстового файла model_finename, а данные из файла image_set_filename. В текстовом файле image_set_filename записан список xml файлов типа разметки, по которым будут делаться оптимизация параметра и вычисляться ошибка распознавания. В той же папке, в которой находится разметка снимка, должен находиться и сам снимок.
2. Оптимальное значение параметра модели допишется в файл модели model_finename.

4) Скользящее окно для снимка с микроскопа.
   Параметры:
   -s -m model_finename -i image_filename
   Выход:
   Сделается распознавание нейронов на изображении model_finename, результат запишется в файл model_finename_rects в той же папке.

5) Пороговая адаптивная бинаризация для выделения оболочек нейронов (нужна для сбора признаков нейронов)
   Параметры:
   -b -m model_finename -i image_filename –r output_image
   Выход:
   В файл output_image запишется бинаризация изображения входного снимка с микроскопа.

Проект написан на MS Visual Studio 2013 с билиотекой opencv 2.4.9. Скомпилированный для x32 системе есть в папке bin.

3. Результаты

На вход был дан набор из 61 снимка с микроскопа. По ним была сделана разметка и получена выборка нейронов и фона, состоящая из 4915 изображений нейронов и 42501 изображения фона. На обучение подавалось 4000 изображений нейронов и 6000 изображений фона выбранных случайным образом. Все остальные изображения шли в тестовую выборку.

По каждому объекту выборки вычисляются признаки. В текущей работе в роли признаков используется многомасштабная гистограмма градиентов HOG [5]. Изображение делилось на равные квадратные области 3x3 и 6x6. Для каждой области вычислялись гистограммы градиентов [5], состоящие из 8 значений. Значения со всех областей записывались как признаки объекта.

Обучение делается линейным классификатором [6]. Линейный классификатор выставляет каждому изображению число с плавающей точкой, которое сравнивается с пороговым значением (параметр классификатора), и в случае, если число больше порога, объект классифицируется как нейрон, если меньше – как фон. Варьируя порог, получаются различные значения ошибок первого и второго рода.

После обучения классификатора распознавание делается методом скользящего окна [7] разных масштабов. В данной работе окна принимают значения 24x24, 36x36, 48x48, 60x60 пикселей. Для каждого изображения в скользящем окне вычисляются признаки, и по ним делается классификация. На реальных снимках параметр классификации в распознавании будет определять количество ложных детекций и количество пропусков нейронов. Так же для ускорения делается процедура предобработки, при которой делается адаптивная бинаризация изображения, при которой приоисходит примерная разметка пикселей: черные пиксели – пиксели нейронов, белые пиксели – пиксели фона. В случае если в окно не попадает ни одного черного пикселя, признаки не вычисляются и оконному объекту присваивается метка фона.

3.1. Классификация

На рисунке 1 представлена ROC-кривая [4] классификации нейронов. Кривая представляет собой соотношение ошибок первого и второго рода при разных значениях параметра классификации. Параметр нормирован так, что он принимает значения в отрезке от 0 до 1. Каждой точке кривой соответствует какому-то значению параметра, абсцисса точки определяет значения ошибок типа False Positive Rate (FPR). Ордината каждой точки определяет значения ошибок True Positive Rate (TPR). Величина FPR определяет количество ситуаций, когда объект фона был классифицирован как нейрон, по отношению ко всем объектам фона. Величина TPR определяет количество ситуаций, при которой объект-нейрон был детектирован как нейрон, по отношению ко всем объектам-нейронам. На тестовой выборке оптимальное значение ошибок следующее: 10.10% ложные нейроны, 6.33% ложный фон. Точность классификации: 94.70%.

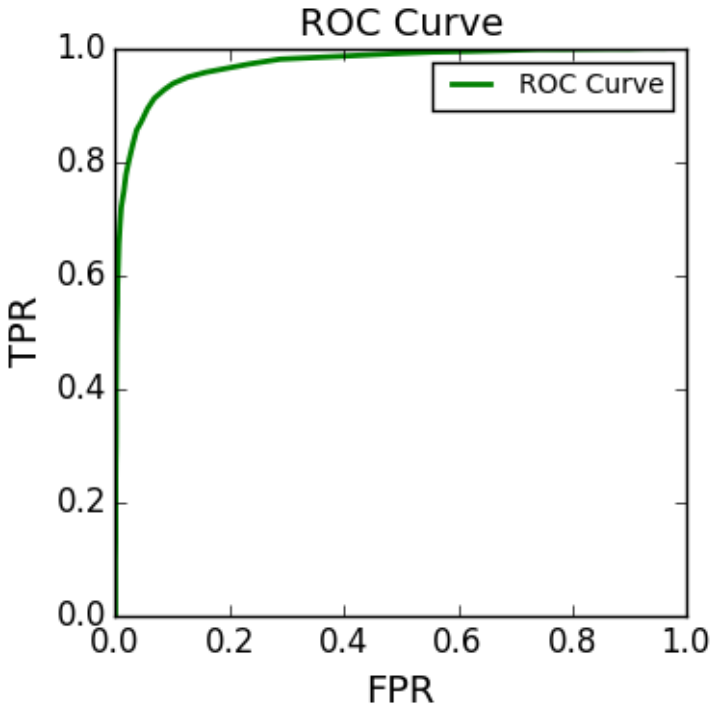

Рисунок 1. ROC-кривая классификации

3.2. Распознавание

На рисунке 2 и 3 показаны зависимости ошибок первого и второго рода от параметра классификации. На первом из них показана зависимость ложных детектирований распознавания. Величина ошибки вычислялась как отношение количества ложных детектирований по отношению к количеству правильных нейронов на всех изображениях. На втором показана зависимость пропусков распознавания. Величина определяется как отношение количества нейронов, которые не были детектированы к общему количеству нейронов на всех изображениях.

На данном этапе оптимальные результаты распознавания плохие: 66% потерь размеченных нейронов, 14% ложных детекций.

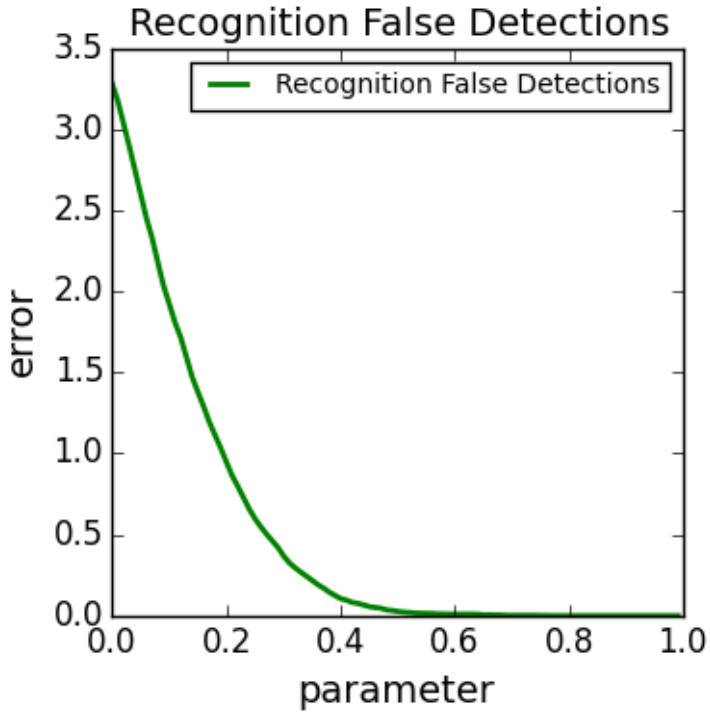

Рисунок 2. Кривая ложных детектирований распознавания.

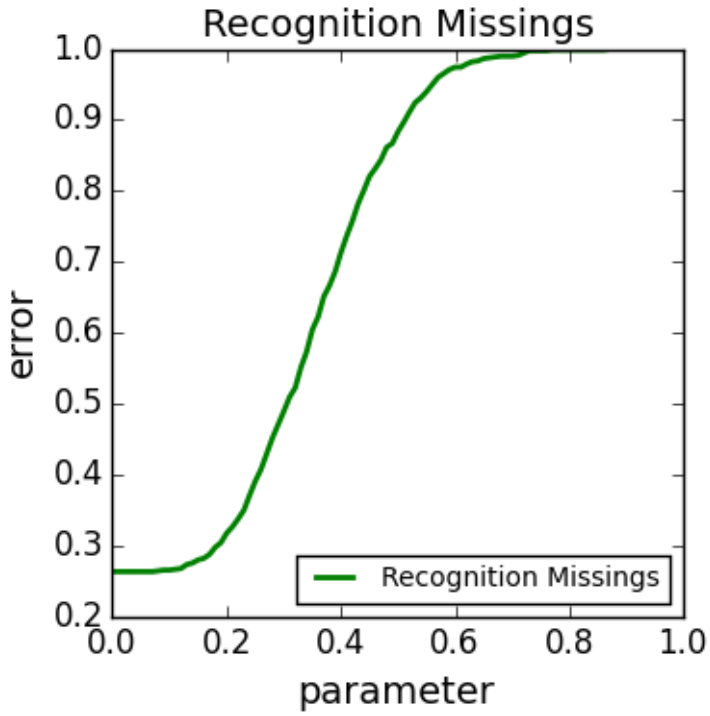

Рисунок 3. Кривая пропусков распознавания.

Пример распознавания показан на рисунке 4.

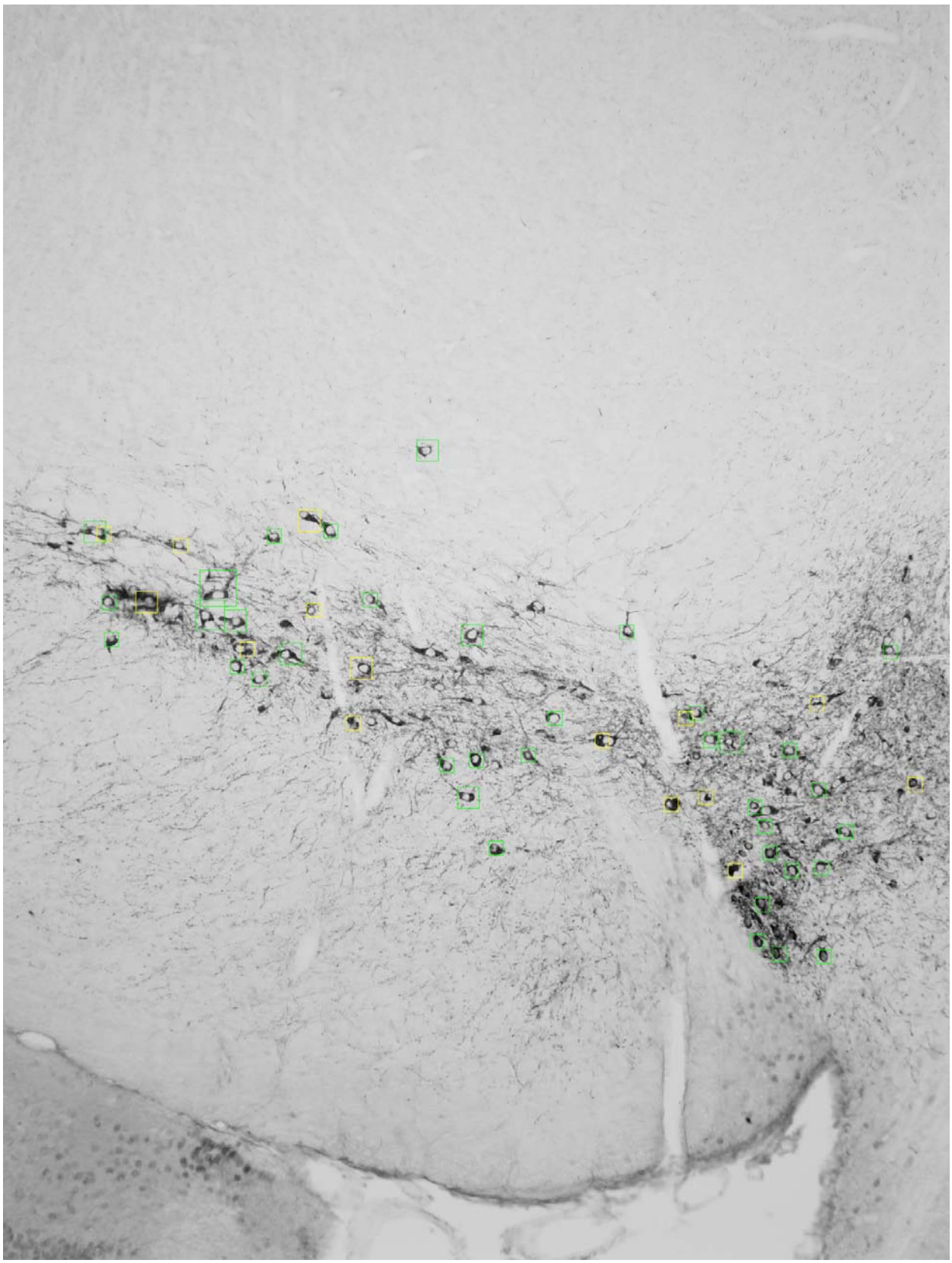

Рисунок 4. Пример распознавания нейронов. Зелеными прямоугольниками обозначены правильно размеченные нейроны, желтыми - не правильные.

4. Улучшение качества

Текущая реализация работает не очень хорошо для распознавания нейронов, однако можно существенно улучшить результаты путем обработки обучающей выборки и заменой метода классификации, при этом подход в целом останется таким же.
Причины большего числа пропусков следующие:

1) Недостаточное количество объектов фона в обучающей выборке. Проблема в том, что на одном изображении количество нейронов много меньше количества объектов фона. Так на одном изображение примерно 20-50 окон с нейронами, в то время как окон с фоном примерно 140000-250000, т.е. их количество на 4 порядка больше. А в текущей версии обучения использовалось 4000 изображений нейронов и 6000 изображений фона. Т.е. при обучении не учитывались все возможные вариации фона, а только некоторые из них. По этому в распознавании получалось большое количество ложных детекций. Однако если добавить все объекты фона, получится слишком большая выборка, которая в текущей реализации классификатора не умещается в памяти.
   Следует понять, что в объектах фона очень большое количество похожих изображений. Поэтому для решения этой проблемы следует сделать кластеризацию изображений фона на большое количество кластеров (по похожести) и обучать не по случайной выборке объектов фона, а по представителям кластеров. Это существенно улучшит качество распознавания.
   Помимо уменьшения количества объектов фона, кластеризация важна еще и тем, что она позволяет оценить качество признаков, т.е. определить, насколько хорошо они разделяют нейроны и фон. Для этого нужно выбрать метрику среди объектов не по пикселям, а по вычисленным признакам. Тем самым можно сделать кластеризацию среди всех объектов: и нейронов и фона. После можно проанализировать, как часто происходит кластеризация объектов разных классов. Так же можно визуально посмотреть такие кластеры и понять какие признаки нужно добавить, чтобы разделить проблемные кластеры.
2) Из изображений нейронов видна их специфическая структура, которая может быть описана другими признаками. Так, например, можно использовать не только гистограммы градиентов [5], но и еще признаки на основе многомасштабных сверток с лапласианом типа SIFT [8] или SURF [9], что улучшит качество распознавания.
3) Сейчас используется линейный классификатор в силу своей простоты и скорости, что вообще говоря, не является оптимальным средством для обучения. Нужно использовать более хитрые классификаторы, такие как SVM, нейронные сети, методы на основе ближайших соседей, чтобы улучшить качество классификации.
4) Так как снимки с микроскопа могут очень сильно меняться, возможно ухудшение качества при обучении на всем множестве разных типов. По этому, можно устроить обучение только по похожим снимкам. Т.е. пользователь размечает только 1-2 снимка из группы похожих снимков, по ним происходит обучение, и все остальные снимки из этой группы распознаются только что обученным классификатором. Это более надежный способ, так как в дальнейшем нейроны на новых снимках с микроскопа могут резко отличаться от исходной обучающей выборки. Более того, такой подход гибче, его можно использовать, чтобы детектировать не только нейроны, но и еще другие подобные объекты.

Литература

[1] https://github.com/Ccas-Recognition
[2] http://opencv.org/
[3] http://matplotlib.org/
[4] Pepe Margaret S. “The statistical evaluation of medical tests for classification and prediction” — New York, NY: Oxford, 2003. — ISBN 0-19-856582-8.
[5] D. G. Lowe. “Distinctive image features from scale-invariant keypoints” — IJCV, 60(2):91–110, 2004.
[6] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang. "Chih-Jen LinLIBLINEAR: A Library for Large Linear Classification" – Journal The Journal of Machine Learning Research archive, Volume 9, 6/1/2008, Pp. 1871-1874
[7] Lampert, Christoph H. "Beyond sliding windows: Object localization by efficient subwindow search" – Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on 23-28 June 2008, pp. 1 - 8.
[8] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision November 2004, Volume 60, Issue 2, pp 91-110
[9] Herbert Bay, Tinne Tuytelaars, Luc Van Gool. "SURF: Speeded Up Robust Features" - Computer Vision – ECCV 2006, Lecture Notes in Computer Science Volume 3951, 2006, pp 404-417