МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ОПТИКИ АТМОСФЕРЫ СО РАН им. В.Е. ЗУЕВА

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ИССЛЕДОВАНИИ СЛОЖНЫХ СТРУКТУР

МАТЕРИАЛЫ
ДВЕНАДЦАТОЙ КОНФЕРЕНЦИИ С МЕЖДУНАРОДНЫМ УЧАСТИЕМ
4–8 июня 2018 г.

Мероприятие проведено при финансовой поддержке
Российского фонда фундаментальных исследований (проект № 18-07-20033)

Томск
Издательский Дом Томского государственного университета
2018

IN SITU CLASSIFICATION OF RAW AUDIO SAMPLES BY CONVOLUTIONAL NEURAL NETWORKS IN ELECTRONIC BEEHIVE MONITORING

V.A. Kulyukin, S. Mukherjee, Yu.B. Burkatovskaya

Department of Computer Science, Utah State University, Logan, Utah, USA
School of Computer Science & Robotics, Tomsk Polytechnic University, Tomsk, Russia
Institute of Applied Mathematics and Computer Science, Tomsk State University, Tomsk, Russia
vladimir.kulyukin@usu.edu, tracey@tpu.ru

Electronic beehive monitoring (EBM) helps extract critical information on colony behavior and phenologywithout invasive beehive inspections and transportation costs. Audio beehive monitoring is an important component of EBM that continues to attract considerable research and development effort, because it has the potential to automate the identification of various stressors for honeybee colonies and the monitoring of hive health. In this investigation, several convolutional neural networks are developed to classify audio samples captured from microphones deployed above Langstroth beehives' landing pads. We show that convolutional neural networks (ConvNets) can successfully classify raw audio methods and perform on par with standard machine learning (ML) methods traditionally used in audio classification. To ensure the replicability of our findings reported in this article, we have made public our source code and our data set of 9,110 manually labeled audio samples obtained from live honeybee colonies. The links to our source code and data will be available in the journal version of this article.

On our data set, ConvNets performed on par with the four standard ML methods traditionally used in audio classification: logistic regression, KNN, random forests, and support vector machines. All ConvNets and standard ML methods achieved a validation accuracy above 99%. The main trade-off between the ConvNets and the standard ML methods is between manual feature engineering and training time. On an Intel Core i7-4770@3.40GHz processor with 15.5 GiB of RAM and running 64-bit Ubuntu 14.04 LTS it took a total of 59.69 hours to train four ConvNets and only 5 minutes to train the four standard ML models. On the other hand, we spent 80 man hours to complete the feature engineering for the four standard ML methods.

The executed experiments indicate that it is possible to build ConvNets that classify raw audio sampleson par with ConvNets that classify the Fourier spectrograms of the same samples. Specifically, on our data set, a ConvNet trained to classify raw audio samples achieved a validation accuracy of 99.93%, which was slightly higher than the validation accuracy of 99.13% achieved by a ConvNet trained to classify Fourier spectrograms.

Our trained raw audio ConvNet model was persisted on the sdcard of a raspberry pi 3 model B v1.2. The raspberry pi was powered with a fully charged Anker Astro E7 26800mAh portable battery. Two hundred 30-second raw audio samples from the audio data set were placed in a local folder on the raspberry pi. A Python script was written to run every 15 minutes to load the persisted trained ConvNet into memory, load an audio sample from the local folder, split it into non-overlapping 2-second segments, and then classify each 2-second audio segment with the loaded ConvNet. The fully charged battery supported this audio classification for 40 hours during which 162 30-second samples were processed. Thus, it took the system, on average, 13.66 seconds to process one 30-second audio sample. The script was then modified to process four 30-second audio files once every 60 minutes. The objective was to estimate whether a batch approach to in situ audio classification would result in better power efficiency, because the persisted ConvNet would be loaded into memory only once per every 4 30-second audio samples. With the batch approach, the fully charged Anker battery supported audio classification for 43 hours during which 172 30-second audio samples were processed. Thus, it took the system 37.68 seconds, on average, to classify a batch of 4 30-second audio samples.

Our investigation suggests that ConvNets can be put to productive use in electronic beehive monitoring not only because they perform on par with standard ML methods and require no feature engineering but also because they can operate in situ on low voltage computational devices such as the raspberry pi computer.

СОЗДАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

Ч. Гу, М.Л. Громов

Национальный исследовательский Томский государственный университет, Томск, Россия
chongyugu@gmail.com, maxim.leo.gromov@gmail.com

В данной работе мы хотим создать систему автоматического управления игрушечным гоночным автомобилем. Подобного рода задачи всегда были сложными и требовали специализированных подходов к их решению. Это обусловлено несколькими основными моментами. Во-первых, сложностью самих объектов управления. Во-вторых, существованием большого количества неопределенностей и нелинейных характеристик, которые трудно точно описать и учесть. В-третьих, требования к задачам управления часто являются многоуровне-

выми и многоцелевыми. По этим причинам поиск новых методов создания систем автоматического управления является актуальной задачей. Один из возможных подходов к решению этой задачи – это применение нейронных сетей [1].

В работах [2, 3] описываются примеры решения задачи автоматического управления автомобилем на основе искусственных нейронных сетей. Эти решения основаны на распознавании изображений средствами библиотеки OpenCV, и ориентированы на изучение автомобилем маршрута движения (то есть, маршрут заранее не известен). В результате, движение такого автомобиля никак нельзя описать как движение гоночного автомобиля.

В нашей работе мы исходим из того факта, что в гоночных дисциплинах автоспорта трек заранее известен гонщику, и на движение автомобиля оказывают влияние возмущения локального характера. То есть, можно рассчитать допустимую (исключающую движение с проскальзыванием колёс) максимальную скорость движения автомобиля в каждой точке трека. Затем, во время движения автомобиля по треку, система управления должна стараться придерживаться максимальной расчётной скорости, учитывая только локальные возмущения. Таким образом, мы предполагаем два режима работы системы управления автомобилем: медленный режим знакомства автомобиля с треком и гоночный режим движения по треку. Обе задачи предполагается решить с применением искусственных нейронных сетей. В качестве инструмента работы с нейронными сетями будет использоваться инструмент Google TensorFlow [4]

Основным новшеством в данной работе будет использование микроконтроллера в качестве вычислительного узла, на котором будет реализована нейронная сеть. Причём мы видим два возможных способа реализовать нейронную сеть на микроконтроллере.

Первый из них – адаптировать существующие Python/C++ реализации нейронных сетей для использования в микроконтроллере. Второй – перевести нейронную сеть в одно из известных математических представлений (например, в матричное представление) и реализовывать на микроконтроллере это представление. Какой из двух способов окажется более эффективным нам заранее не известно и это является предметом нашего дальнейшего исследования.

Литература

1. *Б. Лу.* Искусственная нейронная сеть и контроль нейронной сети. [11.2005] URL: https://wenku.baidu.com/view/6cf2ed9105087632311212a1.html (дата обращения: 23.03.2018).
2. Autonomous RC car using Raspberry Pi and Neural Networks // Vimal Vignesh, [07.2016] URL: http://www.multunus.com/blog/2016/07/autonomous-rc-car-using-raspberry-pi-and-neural-networks/ (дата обращения: 15.12.2017).
3. OpenCV Python Neural Network Autonomous RC Car. TensorFlow [Электронный ресурс] // URL: https://www.tensorflow.org/ (дата обращения: 26.02.2018).