DOI 10.34185/1562-9945-4-135-2021-02
УДК 004.852

В.В. Герасимов, Д.И. Дружинин, Н.В. Карпенко

ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ ДЛЯ СИСТЕМЫ КОНТРОЛЯ УРОВНЯ ТОПЛИВА ТРАНСПОРТНЫХ СРЕДСТВ

Аннотация. Рассмотрено машинное обучение глубоких многоуровневых нейронных сетей для решения задачи улучшения системы контроля уровня топлива транспортных средств. В работе изложены особенности подготовки и трансформации исходных данных для использования их в модели обучения, описана работа с библиотекой DeepLearning4j и ее преимущества для загрузки модели обучения, показан процесс обучения нейронной сети. В результате выполненной работы разработана модель обучения для определения уровня топлива в баках транспортных средств, описаны особенности предложенной модели.
Ключевые слова: нейронная сеть, глубокое обучение, модель обучения, Deeplearning4j, DataVec, csv, уровень топлива, DenseLayer, SOFTMAX, TANH.

Постановка проблемы. Все предприятия, имеющие автотранспорт, сталкиваются с проблемой больших затрат на содержание своего автопарка. Топливо — одна из самых накладных статей расходов на транспортное средство, а с учетом того, что цены топливо на практически постоянно идут вверх, контроль расхода топлива на сегодня является актуальным вопросом.

Контроль топлива можно осуществлять различными способами: расчётным методом, с помощью штатного датчика уровня топлива (ДУТ) в баке, с помощью емкостного ДУТ, с помощью ультразвукового ДУТ, с помощью датчика расхода топлива и т.д. [1, 2]. Если помимо расхода топлива нужно ещё контролировать и заправку транспортного средства (ТС) (фиксировать время и объем заправки), то круг возможных способов решения задачи сужается к использованию различных ДУТ. В тоже время при работе технологического, крупногабаритного и грузового транспорта уровень топлива в баках может сильно колебаться, что накладывает значительный шум на измеренные данные, поэтому для более качественного контроля расхода топлива следует очистить показания ДУТ от шума, обусловленного работой самого ТС. Таким образом, целью данной работы является создание модели обучения на основе ре-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

альных зашумленных данных и обучение нейронной сети для определения поведения уровня топлива, а именно для определения времени и объема заправки ТС, расхода/перерасхода/слива топлива.

Основная часть. Во время измерения уровня топлива с помощью различных ДУТ на данные влияют очень много внешних факторов, которые могут создавать помехи при измерении и искажают реальный уровень топлива. В системах контроля и учета топлива применяют различные алгоритмы и способы обработки данных с целью избавления их от зашумления. В некоторых системах применяется первичная фильтрация путем исключения показаний, выходящих за пределы допустимых значений, резких скачков и отклонений, усреднение по скользящему окну [3]. Проводятся исследования по применению более сложных фильтров, чем простое усреднение, например, в работе [2] авторы предлагают использовать для обработки данных фильтр Калмана. Однако, следует учесть тот факт, что помехи случайны, имеют разную структуру и их очень сложно полностью удалить с помощью классических алгоритмов подавления шума. Поэтому мы будем использовать искусственный интеллект, а именно нейронную сеть, для поиска закономерностей, определения шума и исправления искаженных данных. Для исправления искаженных данных сначала надо определить, какие именно данные искажены, по сути провести классификацию данных.

Deeplearning4J или DL4J — это библиотека Java/Scala для глубокого обучения [4]. Кроме того, это также целое семейство других библиотек, которые упрощают использование моделей глубокого обучения с помощью языков программирования Java/Scala.

При работе с нейронными сетями большая часть данных, обычно около 80%, используется для обучения и поэтому называется обучающей выборкой. Остальные данные, обычно около 20%, используются для оценки качества модели и называются набором тестов. При выборе данных для набора тестов следует отметить, что этот набор должен состоять из репрезентативного подмножества всех данных. Это необходимо для правильной проверки информативности модели.

Для определения поведения уровня топлива в определенный момент времени будем использовать скользящее окно — 20 значений уровня топлива до и после текущего значения.

Сначала необходимо выполнить трансформацию исходных данных об уровне топлива для использования их в модели обучения. Исходные данные

представляют собой значение уровня топлива в каждый момент времени. Для обучающей модели мы будем брать все значения уровня топлива в рамках скользящего окна (для формирования контекста поведения), основным значением будет величина уровня топлива в данный момент времени. Для определения времени, когда происходит реальный расход топлива, будем использовать дополнительные значения скорости ТС. Для проверки качества нашей модели будем использовать данные о заправках ТС, которые получены другими способами (внесены вручную, получены с помощью другой системы контроля и т.д.).

Для описания поведения уровня топлива будем использовать 4 возможных класса. Так как данные для обучения возможны только в числовом формате, будем использовать условные числовые значения: 0 – наблюдается расход топлива; 1 – осуществляется заправка ТС топливом; 2 – уровень топлива не изменяется (ТС простаивает); 3 – данные искажены помехами.

После обработки данных будем использовать DataVec [5] для их загрузки в модель. Как и все другие методы статистического машинного обучения, глубокое обучение работает только с числовыми данными, а DataVec — это библиотека, которая помогает нам загружать, анализировать и преобразовывать наши данные в необходимый формат.

Библиотека DataVec включает дополнительные подклассы FileSplit, RecordReader и TransformProccess.

Мы передаем экземпляру RecordReader папку, в которой находится наш обучающий набор, как экземпляр класса File.

```
var numLinesToSkip = 0
var delimiter = ','
val recordReader: RecordReader = new
CSVRecordReader(numLinesToSkip, delimiter)
recordReader.initialize(new FileSplit(new
File("/home/denis/Documents/trainData1.txt")))
```

Наши данные отформатированы в CSV, поэтому мы создаем экземпляр CSVRecordReader и инициализируем его ранее созданным экземпляром FileSplit. Затем экземпляр RecordReader входные данные, полученные от экземпляра FileSplit, делит на один или несколько примеров.

Следующим шагом будет векторизация данных. Обычно глубокое обучение учитывает не все данные обучения на каждом этапе обучения, а только их часть. Этот процесс называется мини-дозированием (mini-batches). Каждый

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

полный проход данных называется эпохой. Желательно, чтобы для каждой эпохи мини-партии состояли из разных примеров, поскольку это помогает в лучшем обучении модели [6].

Размер мини-пакета, так называемый размер пакета, определяет, сколько примеров модель видит на каждом этапе обучения, и, таким образом, также значительно влияет на режим обучения. Чем больше размер пакета, тем меньше мини-партии и, следовательно, меньше шагов обучения в каждую эпоху. Однако в то же время модель может видеть больше данных и может лучше находить закономерности. Экспериментально мы определили, что обучение дает лучший результат при размере партии равной 100: `val batchSize = 100`.

Затем создается экземпляр DataSetIterator, который считывает преобразованные данные из экземпляра RecordReader, векторизует их и заботится о создании мини-партий. Мы используем подход классификации [7], чтобы указать, в каком столбце находится наш класс, то есть целевое значение, и сколько существует возможных классов. Мы будем использовать этот DataSetIterator для обучения нашей модели.

```
val labelIndex = 22
val numClasses = 4
val iterator: DataSetIterator = new
RecordReaderDataSetIterator(recordReader, batchSize,
labelIndex, numClasses)
val allData: DataSet = iterator.next()
allData.shuffle()
```

Разделяем данные на обучающие и тестовые

```
val testAndTrain: SplitTestAndTrain =
allData.splitTestAndTrain(0.25)
```

В нашем случае используется трехслойная модель нейронной сети. Сначала устанавливаем случайное начальное число весов(seed), поскольку веса модели будут инициализироваться случайным образом — в нашем случае с использованием схемы инициализации XAVIER [8]. Затем функция активации TANH [9] устанавливается по умолчанию для каждого слоя. Мы используем алгоритм оптимизации Adam [10] со скоростью обучения 0,1.

```
val conf: MultiLayerConfiguration = new
NeuralNetConfiguration.Builder()
.seed(seed)
.activation(Activation.TANH)
.weightInit(WeightInit.XAVIER)
.updater(new Sgd(0.1))
```

```
.l2(1e-4)
.list()
.layer(new DenseLayer.Builder().nIn(numInputs).nOut(numInputs)
.build())
.layer(new DenseLayer.Builder().nIn(numInputs).nOut(numInputs)
 .build())
.layer( new
OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKE
LIHOOD)
.activation(Activation.SOFTMAX) //Override the global TANH
activation with softmax for this layer
.nIn(numInputs).nOut(outputNum).build())
.build()
```

В нашем случае архитектура модели состоит из 2 базовых мультиперцептронных слоев DenseLayer [11], каждый с 22 модулями ввода-вывода, и одного слоя OutputLayer с 4 выходами. Сам выходной слой OutputLayer также является DenseLayer, но разница заключается в функции потерь (`LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD`), которая вычисляет отклонение между результатом обучения модели и целевым классом. По результатам этого отклонения параметры модели затем корректируются, чтобы обеспечить лучшие результаты. Выходной слой OutputLayer устанавливает функцию активации, отличную от TANH по умолчанию, и вместо нее использует активацию SOFTMAX [9]. Это гарантирует, что выходные данные модели можно интерпретировать как распределение вероятностей, то есть каждое значение находится между 0 и 1, а сумма всех значений равна точно 1.

Все выбранные здесь значения параметров являются результатом экспериментов с количеством слоев, количеством входов для слоёв (кроме первого) и выходов на каждом слое (кроме последнего), а также изменением функции активации. Так функция активации TANH показала лучшие результаты, чем более часто используемая функция активации RELU [9].

По сравнению с усилиями, необходимыми для создания хорошей архитектуры модели, обучение является довольно простым. Сначала мы создаем новую модель из ранее определенной конфигурации, инициализируем ее, а затем обучаем, используя обучающий набор для 1000 эпох, т.е. обучающий набор повторяется более 1000 раз во время обучения. Через некоторое время обучение заканчивается, и обученную модель можно использовать.

```
val model: MultiLayerNetwork = new MultiLayerNetwork(conf)
model.init()
```


ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

```
//record score once every 100 iterations
model.setListeners(new ScoreIterationListener(100))
for(i <- 0 to 1000) {
 model.fit(trainingData)
}
```

Фактическое обучение всегда вращается вокруг вызова метода подгонки. DL4J предлагает несколько вариантов. Мы перебираем итератор и передаем следующую мини-партию методу fit. Каждый вызов итератора — эпоха.

После обучения модели нужно также оценить, насколько хорошо она обобщает данные, которые еще не использовала. Для этого используется тестовый набор. Чтобы использовать тестовый набор, мы загружаем его так же, как и обучающий. В этом случае размер пакета не имеет значения для результата, поэтому для простоты мы будем использовать тот же размер, что и для обучающей выборки.

```
val eval: Evaluation = new Evaluation(4)
   val output: INDArray = model.output(testData.getFeatures)
```

Матрица неточностей [12] показывает, каким образом модель классификации сбивается с толку, когда делает прогнозы. По матрице неточностей мы видим, что искусственный интеллект смог выделить 4 класса поведения (рис. 1). Успешный результат находиться на перекрестке колонки и ряда определенного класса, остальные значения — ошибочные.

```
==========================Confusion Matrix==========================
  0  1  2  3
-------------
 16  0  0  4 | 0 = 0
  0  1  5  1 | 1 = 1
  0  0 39  0 | 2 = 2
  2  0  1  6 | 3 = 3

Confusion matrix format: Actual (rowClass) predicted as (columnClass) N times
```

Рисунок 1 – Результат тестирования модели нейронной сети – Матрица неточностей (Confusion Matrix)

Объект оценки можно использовать для доступа к другим метрикам [12, 13], которые по умолчанию не отображаются в сводке. В нашем случае следует особо отметить коэффициент корреляции Мэтьюза (F1) [13], поскольку он также учитывает неравномерное распределение в данных и, таким образом, показывает то, что, несмотря на хорошую точность, предсказательная сила этой модели все еще несколько ограничена (рис. 2).

```
========================Evaluation Metrics========================
 # of classes:    4
 Accuracy:        0.8267
 Precision:       0.7008
 Recall:          0.7050
 F1 Score:        0.7027
Precision, recall & F1: macro-averaged (equally weighted avg. of 4 classes)
```

Рисунок 2 – Результат тестирования модели – Метрики оценки (Evaluation Metrics): Accuracy — доля от общего числа верных прогнозов; Precision — доля правильно идентифицированных положительных случаев; Recall — доля действительно положительных случаев, которые правильно идентифицированы

После обучения модель нейронной сети записываем в указанный файл, вызывая метод saveFile. По умолчанию состояние Updater также сохраняется для того, чтобы продолжить обучение сохраненной модели позже.

Результаты работы разработанной нейронной сети показаны на рис. 3 — пунктиром отображаются исходные данные, сплошной линией показаны скорректированные данные.

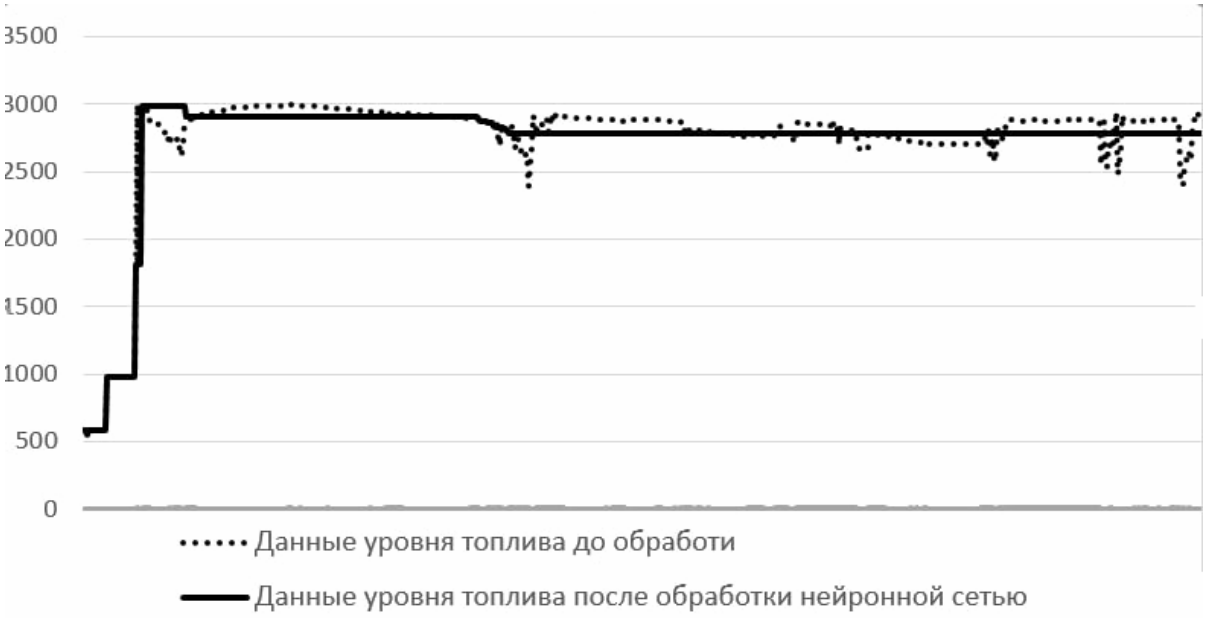

Рисунок 3 – График исходных и скорректированных данных

Выводы. В данной работе описана созданная модель обучения и процесс обучения нейронной сети через использование библиотеки Deeplearning4J, которая предоставляет методы для более быстрого глубокого обучения. Пред-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

ставлен процесс обработки и трансформации совокупности сырых данных в векторизованный числовой вид для использования их в модели обучения. Получена и обучена нейронная сеть для определения состояния уровня топлива в текущий момент времени. Классифицировано поведение модели на четыре основных класса. Хотя пока ошибка определения поведения уровня топлива не сведена к нулю, мы сохранили состояния нейронной сети и в будущем сможем переобучить и развить нашу нейронную сеть для получения лучших результатов.

ЛИТЕРАТУРА / ЛІТЕРАТУРА

1. Контроль топлива — Системы GPS мониторинг транспорта и контроля топлива [Электронный ресурс]. Режим доступа: https://gps-monitoring.com.ua/kontrol-topliva/

2. Алексеев Н. Ю., Кудрявцев А. А., Асмолов Г. И., Лобов О. П. Реализация фильтра Калмана при обработке данных от датчика уровня топлива с использованием дополнительной информации от навигационно-связного терминала // Международный журнал перспективных исследований, т.8, № 3, 2018, с. 9-23.

3. Контроль уровня топлива глонасс АвтоГРАФ [Электронный ресурс]. Режим доступа: http://snavi.ru/toplivo.html

4. Deeplearning4j [Электронный ресурс]. Режим доступа: https://deeplearning4j.org

5. GitHub - deeplearning4j/DataVec: ETL Library for Machine Learning - data pipelines, data munging and wrangling [Электронный ресурс]. Режим доступа: https://github.com/deeplearning4j/DataVec

6. Паттерсон Дж., Гибсон А. Глубокое обучение с точки зрения практика / пер. с анг. А. А. Слинкина. — М.: ДМК Пресс, 2018. — 418 с.

7. Asiri S. Machine Learning Classifiers [Электронный ресурс]. Режим доступа: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

8. Joshi P. Understanding Xavier Initialization In Deep Neural Networks [Электронный ресурс]. Режим доступа: https://prateekvjoshi.com/2016/03/29/understanding-xavier-initialization-in-deep-neural-networks/

9. Sharma S. Activation Functions in Neural Networks [Электронный ресурс]. Режим доступа: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

10. Brownlee J. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning [Электронный ресурс]. Режим доступа:

https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
11. Multilayer Network [Электронный ресурс]. Режим доступа: https://deeplearning4j.konduit.ai/models/multilayernetwork
12. Brownlee J. What is a Confusion Matrix in Machine Learning [Электронный ресурс]. Режим доступа: https://machinelearningmastery.com/confusion-matrix-machine-learning/
13. Chicco D., Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21, 6 (2020). https://doi.org/10.1186/s12864-019-6413-7

REFERENCES

1. Fuel control - GPS systems for vehicle monitoring and fuel control [Electronic resource]. Access mode: https://gps-monitoring.com.ua/kontrol-topliva/
2. Alekseev N. Yu., Kudryavtsev A. A., Asmolov G. I., Lobov O. P. Implementation of the Kalman filter for data processing from the fuel level sensor with the use of additional information from the navigation communication terminal // International Journal of Advanced Studies, Vol. 8, No 3, 2018, P. 9-23.
3. FUEL LEVEL CONTROL GLONASS AutoGRAPH [Electronic resource]. Access mode: http://snavi.ru/toplivo.html
4. Deeplearning4j [Electronic resource]. Access mode: https://deeplearning4j.org
5. GitHub - deeplearning4j/DataVec: ETL Library for Machine Learning - data pipelines, data munging and wrangling [Electronic resource]. Access mode: https://github.com/deeplearning4j/DataVec
6. Patterson J., Gibson A. Deep Learning: A Practitioner's Approach 1st Edition. O'Reilly Media, 2017, 532 p.
7. Asiri S. Machine Learning Classifiers [Electronic resource]. Access mode: https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623
8. Joshi P. Understanding Xavier Initialization In Deep Neural Networks [Electronic resource]. Access mode: https://prateekvjoshi.com/2016/03/29/understanding-xavier-initialization-in-deep-neural-networks/
9. Sharma S. Activation Functions in Neural Networks [Electronic resource]. Access mode: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
10. Brownlee J. Gentle Introduction to the Adam Optimization Algorithm for Deep Learning [Electronic resource]. Access mode:

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
11. Multilayer Network [Electronic resource]. Access mode: https://deeplearning4j.konduit.ai/models/multilayernetwork
12. Brownlee J. What is a Confusion Matrix in Machine Learning [Electronic resource]. Access mode: https://machinelearningmastery.com/confusion-matrix-machine-learning/
Chicco D., Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 21, 6 (2020). https://doi.org/10.1186/s12864-019-6413-7

Received 22.03.2021.
Accepted 24.03.2021.

Навчання нейронної мережі для системи контролю рівня палива транспортних засобів

Розглянуто машинне навчання глибоких багаторівневих нейронних мереж для вирішення задачі покращення системи контролю рівня палива транспортних засобів. В роботі викладено особливості підготовки і трансформації вихідних даних для використання їх в моделі навчання, описана робота з бібліотекою DeepLearning4j і її переваги для завантаження моделі навчання, показано процес навчання нейронної мережі. В результаті виконаної роботи розроблено модель навчання для визначення рівня палива в баках транспортних засобів, описано особливості запропонованої моделі.

Neural network training for vehicle fuel level control

The goal of the paper is to create a training model based on real raw noisy data and train a neural network to determine the behavior of the fuel level, namely, to determine the time and volume of vehicle refueling, fuel consumption / excessive consumption / drainage.

Various algorithms and data processing methods are used in fuel control and metering systems to get rid of noise. In some systems, primary filtering is used by excluding readings that are out of range, sharp jumps and deviations, and averaging over a sliding window. Research is being carried out on the use of more complex filters than simple averaging – by example, the Kalman filter for data processing.

When measuring the fuel level using various fuel level sensor the data is influenced by many external factors that can interfere with the measurement and distort the real fuel level. Since these interferences are random and have a different structure, it is very difficult to completely remove them using classical noise suppression algorithms. Therefore, we use artificial intelligence, namely a neural network, to find patterns, detect noise and correct distorted data. To correct distorted data, you first need to determine which data is distorted, classify the data.

In the course of the work, the raw data on the fuel level were transformed for use in the neural network training model. To describe the behavior of the fuel level, we use 4 possible classes: fuel consumption is observed, the vehicle is refueled, the fuel level does not change (the vehicle is idle), the data is distorted by noise. Also, in the process of work, additional tools of the

DeepLearning4 library were used to load data training and training a neural network. A multilayer neural network model is used, namely a three-layer neural network, as well as used various training parameters provided by the DeepLearning4j library, which were obtained because of experiments.

After training the neural network was used on test data, because of which the Confusion Matrix and Evaluation Metrics were obtained.

In conclusion, finding a good model takes a lot of ideas and a lot of experimentation, also need to correctly process and transform the raw data to get the correct data for training. So far, a neural network has been trained to determine the state of the fuel level at a point in time and classify the behavior into four main labels (classes). Although we have not reduced the error in determining the behavior of the fuel level to zero, we have saved the states of the neural network, and in the future we will be able to retrain and evolve our neural network to obtain better results.

Герасимов Володимир Володимирович – доцент кафедри електронних обчислювальних машин ДНУ.

Карпенко Надія Валеріївна – доцент кафедри електронних обчислювальних машин ДНУ.

Дружинін Денис Ігорович – аспірант кафедри електронних обчислювальних машин ДНУ.

Герасимов Владимир Владимирович - доцент кафедры электронных вычислительных машин ДНУ.

Карпенко Надежда Валерьевна - доцент кафедры электронных вычислительных машин ДНУ.

Дружинин Денис Игоревич - аспирант кафедры электронных вычислительных машин ДНУ.

Gerasimov Vladimir Vladimirovich — Associate Professor of Computer Systems Engineering Department of DNU

Karpenko Nadija Valeriivna — Associate Professor of Computer Systems Engineering Department of DNU

Druzhynin Denys Igorovich — Postgraduate of Computer Systems Engineering Department of DNU.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)