УДК 004.093

Обзор технологии AutoML, имеющихся инструментов, пример применения и сравнительный анализ с классическим решением

© И. В. Низамов, Д. А. Еловенко
Иркутский национальный исследовательский технический университет,
г. Иркутск, Российская Федерация

Аннотация. В данной статье рассмотрены принципы автоматизации сквозного процесса применения машинного обучения, приведена основная терминология. Произведена сравнительная оценка сильных и слабых сторон использования AutoML. Рассмотрены наиболее зрелые и распространённые существующие решения совместимые с языком Python, позволяющие осуществлять выбор модели и поиск гиперпараметров. Приведён пошаговый пример использования автоматизированного машинного обучения для решения реальной задачи классификации изображений. А также поэтапной пример разработки нейросети, решающий аналогичную задачу. Сделано сравнение функций потерь и общей точности предсказания моделей по всем классам. Проведён анализ полученных моделей и выделены ключевые отличия архитектур нейросетей. В заключение сформированы следующие выводы и общие рекомендации: автоматизированное машинное обучение на данном этапе развития не может в полной мере гарантировать точность результата, но при решении задач, с использованием наиболее подходящих инструментов при выполнении отдельных классов задач AutoML, демонстрирует итоги, превосходящие результаты достигаемые специалистами. Наиболее рациональное применение AutoML – это поиск новых наиболее оптимальных нейросетевых архитектур и гиперпараметров для последующей ручной оптимизации с целью повышения точности.

Ключевые слова: нейросеть, нейросетевая архитектура, машинное обучение, автоматизированное машинное обучение, гиперпараметры, генетический алгоритм

Overview of AutoML Technology, Available Tools, Application Example and Comparative Analysis with the Classic Solution

© Igor V. Nizamov, Denis A. Elovenko
Irkutsk National Research Technical University,
Irkutsk, Russian Federation

Abstract. The article discusses the principles of automating the end-to-end process of using machine learning, and provides the basic terminology. The article provides a comparative assessment of the strengths and weaknesses of using AutoML. The article discusses the most mature and common existing solutions compatible with the Python language, allowing for model selection and hyperparameter search. The article provides a step-by-step example of using automated machine learning to solve a real image classification problem, as well as a step-by-step example of developing a neural network that solves a similar problem. It compares the loss functions and the overall accuracy of predicting models for all classes. The article analyzes the obtained models and highlights the key differences between neural network architectures. The article contains the following conclusions and general recommendations: automated machine learning at this stage of development cannot fully guarantee the accuracy of the result, but when solving problems, using the most appropriate tools when performing certain classes of AutoML problems; it demonstrates results that exceed the results achieved by specialists. The most rational use of AutoML is the search for new most optimal neural network architectures and hyperparameters for subsequent manual optimization in order to improve accuracy.

Keywords: neural network, neural network architecture, machine learning, automated machine learning, hyperparameters, genetic algorithm

Введение

Алгоритмы машинного обучения являются неотъемлемой частью большого числа современных информационных систем. В данной статье рассмотрены технологии автоматизированного машинного обучения, описаны современные инструменты, пример их практического применения и сравнительный анализ с классическим решением.

Автоматизированное машинное обучение (далее – AutoML) – это набор концепций и методов, используемых для автоматизации процессов формирования моделей машинного обучения, проходящих через следующие этапы:

1. Предварительная обработка данных.

2. Выделение признаков.
3. Выбор функций.
4. Выбор наиболее подходящего алгоритма.
5. Настройка параметров.

Концепции AutoML

AutoML имеет две основные концепции:

1. Поиск нейронной архитектуры – это процесс автоматизации проектирования нейронных сетей. Для этих целей используются обучающиеся или эволюционные алгоритмы. В обучении с подкреплением модели наказываются за низкую точность и вознаграждаются за высокую. Используя эту технику, модель всегда будет стремиться получить более высокую точность. Поиск нейронной архитектуры показывает наилучшие результаты для решения задач, требующих обнаружения новых архитектур [1].
2. Передача обучения – это метод, в котором используются предварительно обученные модели для переноса того, что изучила модель при применении к новому, но похожему набору данных. Это позволяет нам получить высокую точность, используя меньше времени и вычислительных мощностей. Трансферное обучение лучше всего подходит для задач, в которых наборы данных аналогичны тем, которые используются в моделях предварительного обучения [2].

Преимущества AutoML:

– возможность построения и применения моделей машинного обучения без навыков программирования, предметной экспертизы и математического анализа;

– возможность автоматизированной обработки первичных данных, самостоятельного определения решаемой задачи, выбора оптимальной нейросетевой архитектуры, оптимизации гиперпараметров алгоритма обучения и решения иных задач, вплоть до визуализации;

– возможность уменьшения смещения, дисперсии, времени разработки и тестирования модели.

Недостатки AutoML:

– новизна концепции несёт соответствующие риски и требует осторожности при применении последних версий библиотек;

– решения AutoML являются весьма ресурсоемкими и требуют привлечения облачных вычислений, т. к. время их работы на локальных компьютерах достаточно велико.

Обзор решений AutoML

Ниже рассмотрены на некоторые существующие решения, совместимые с языком Python, позволяющие осуществлять выбор модели и поиск гиперпараметров, но каждый инструмент имеет свои особенности (рис. 1).

H20.ai

H2O – это платформа машинного обучения с открытым исходным кодом. Она доступна как на R, так и на Python. Этот пакет обеспечивает поддержку методов машинного обучения с использованием быстрых алгоритмов, и с 2018 года он получил поддержку глубокого обучения. На фоне популяризации Apache Spark, H2O обзавелся интерфейсом Sparkling Water для объединения возможностей Apache Spark и H2O. H20.ai является абсолютным лидером по скорости выполнения, но продукт дает относительно среднюю точность [3].

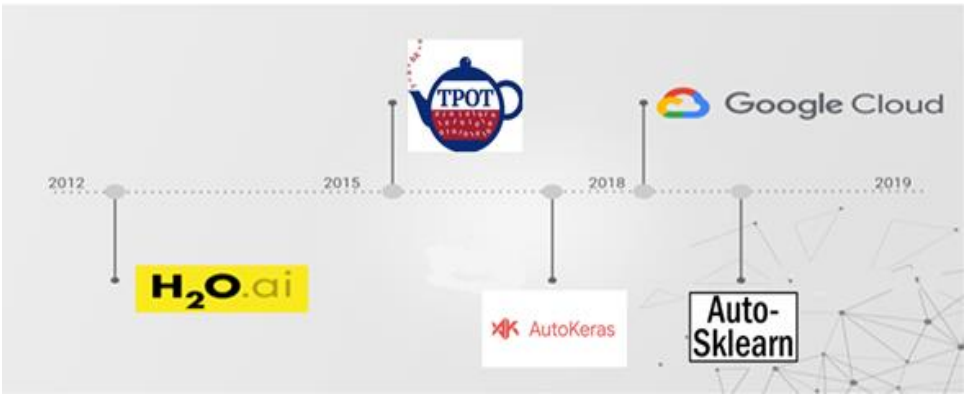

Рис. 1. Хронология выпуска инструментов AutoML

Инструмент оптимизации на основе дерева (TPOT)

Целью TPOT является автоматизация построения конвейеров ML путем объединения гибкого представления дерева конвейеров выражений с алгоритмами стохастического поиска, такими как генетическое программирование. TPOT использует основанную на Python библиотеку scikit-learn в качестве меню ML. Пакет не умеет взаимодействовать с естественным языком и категориальными признаками [4].

AutoKeras

AutoKeras – это программная библиотека, с открытым исходным кодом для автоматического машинного обучения. Она разработана DATA Lab из Техасского университета A & M и участниками сообщества. Конечная цель AutoML – предоставить легкодоступные инструменты глубокого обучения специалистам с ограниченным опытом в больших данных или машинном обучении. AutoKeras предоставляет функции для автоматического поиска архитектуры и гиперпараметров моделей глубокого обучения [5].

Cloud AutoML – Google Cloud

Cloud AutoML – это набор продуктов для машинного обучения, который позволяет разработчикам с ограниченным опытом в области ML обучать высококачественные модели, соответствующие бизнес-потребностям, используя передовые технологии обучения Google и технологию поиска нейронной архитектуры. Как утверждают разработчики, готовое для промышленной эксплуатации решение можно получить в течение рабочего дня. Обладает простым пользовательским интерфейсом и может использоваться бесплатно в течение года. Для коммерческих решений продукт является платным [6].

Auto-Sklearn

Auto-Sklearn – это автоматизированный пакет машинного обучения, основанный на scikit-learn. Умеет генерировать признаки на основе «сырых» данных. Преимущества фреймворка в бейсовском языке оптимизации, метаобучение и ансамблевое построение. Эффективно работает только с небольшими датасетами. По отзывам пользователей, Auto-Sklearn дает большую точность, но является самым медленным среди конкурентов [7].

Применение AutoML

В примере приведена задача распознавания рукописных чисел на изображениях путём классификации каждой цифры в наборе данных. Для решения задач данного класса требуется максимальная точность, то есть корректное распознавание всех цифр. Поэтому нерационально полностью полагаться на модель, полученную путём AutoML, но можно оценить точность, достигнутую при помощи тех или иных архитектур и весов, выбранных генетическими алгоритмами обучения, и использовать как основу для рабочей модели.

С подобными задачами классификации отлично справляется библиотека Keras, так как имеет мощный нейросетевой поиск параметров модели, поэтому пример будет реализован с помощью AutoKeras.

Для простоты воспроизведения использован открытый набор данных MNIST, содержащий рукописные цифры.

Для разработки нейросетей и сравнения результатов были проделаны описанные ниже шаги.

1. Установка AutoKeras.

Для работы с библиотекой AutoKeras требуется её установка в среду исполнения Google Colab.

```
!pip install autokeras
```

2. Импорт данных.

Из AutoKeras импортируются библиотека Tensor Flow, набор данных MNIST и библиотека Auto Keras.

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
import autokeras as ak
```

3. Кросвалидация.

Для получения тренировочного и тестового наборов данных производится разделение загруженных данных.

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

4. Обучение.

4.1. Для классификация изображений, инициализируем классификатор изображений с количеством испытаний 1.

```
clf = ak.ImageClassifier(max_trials=1)
```

4.2. Передаем классификатору данные для обучения, с количеством эпох = 1.

```
clf.fit(x_train, y_train, epochs=1)
```

5. Перенос обучения.

5.1. Сохранение сгенерированной модели.

```
model = clf.export_model()
print(type(model))
try:
    model.save("model_autokeras", save_format="tf")
except:
        model.save("model_autokeras.h5")
```

5.2. Загрузка сохранённой модели.

```
from tensorflow.keras.models import load_model
loaded_model = load_model("model_autokeras", custom_objects=ak.CUSTOM_OBJECTS)
predicted_y = loaded_model.predict(tf.expand_dims(x_test, -1))
```

6. Оценка модели по количеству ошибок и потерь (рис. 2).

```
loss,acc = loaded_model.evaluate(test_images,  test_labels, verbose=2)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
10000/10000 - 0s - loss: 0.4074 - accuracy: 0.9182
Restored model, accuracy: 91.82%
```

Рис. 2. Оценка автоматически полученной модели по количеству ошибок и потерь

Формирование модели ML-инженером

Для сравнения результатов работы сгенерированной модели построена модель на основе примера кода для классификации набора данных MNIST с официального сайта TensorFlow.org [8].

Импорт библиотек, кросвалидация, подготовка данных и описание архитектуры базовой модели.

```
from __future__ import absolute_import, division, print_function, unicode_literals
import os
import tensorflow as tf
from tensorflow import keras
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

train_labels = train_labels[:1000]
```

```
test_labels = test_labels[:1000]

train_images = train_images[:1000].reshape(-1, 28 * 28) / 255.0
test_images = test_images[:1000].reshape(-1, 28 * 28) / 255.0

def create_model():
        model = tf.keras.models.Sequential([
            keras.layers.Dense(512, activation='relu', input_shape=(784,)),
            keras.layers.Dropout(0.2),
            keras.layers.Dense(10, activation='softmax') ])
        model.compile(optimizer='adam',
                                    loss='sparse_categorical_crossentropy',
                                    metrics=['accuracy'])
        return model
```

Создан экземпляр базовой модели.

```
model = create_model()
```

Сохранены контрольные точки

```
checkpoint_path = "training_1/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)
```

Создан обратный вызов сохраняющий веса модели

```
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,
save_weights_only=True, verbose=1)
```

Обучена модель с новым обратным вызовом

```
model.fit(train_images,
            train_labels,
            epochs=10,
            validation_data=(test_images,test_labels),
            callbacks=[cp_callback])
```

Создан экземпляр базовой модели

```
model2 = create_model()
```

Произведена оценка модели (рис. 3)

```
loss, acc = model2.evaluate(test_images,  test_labels, verbose=2)
print("Untrained model, accuracy: {:5.2f}%".format(100*acc))
```

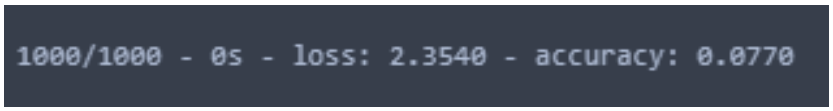

Рис. 3. Оценка модели построенной ML-инженером

Загружены веса

```
model2.load_weights(checkpoint_path)
```

Произведена повторная оценка модели (рис. 4)

```
loss,acc = model2.evaluate(test_images,  test_labels, verbose=2)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

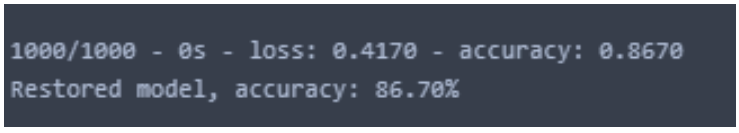

Рис. 4. Повторная оценка модели построенной ML-инженером

```
loaded_model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0
_________________________________________________________________
normalization (Normalization (None, 28, 28, 1)         3
_________________________________________________________________
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 12, 12, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 9216)              0
_________________________________________________________________
dense (Dense)                (None, 10)                92170
_________________________________________________________________
classification_head_1 (Softm (None, 10)                0
=================================================================
Total params: 110,989
Trainable params: 110,986
Non-trainable params: 3
_________________________________________________________________
```

Рис. 5. Отчёт по модели, сгенерированной AutoKeras

```
model2.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_8 (Dense)              (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

Рис. 6. Отчёт по модели, сформированной ML-инженером

Сравнительный анализ модели сформированной ML-инженером с моделью сгенерированной AutoKeras

Результаты сравнения показали, что точность модели, сгенерированной в AutoKeras, превосходит на пять процентов модель, построенную человеком. В отчётах, представленных на рис. 5–6, имеется информация – какие слои и параметры были использованы при AutoML и «классическом» обучении.

Точность сгенерированной модели достигнута благодаря использованию Conv2D, Flatten, MaxPooling2D слоев и уменьшению количества обучающих параметров [9]. Сгенерированная модель распознает рукописные цифры с точностью до 91 % и может быть использована повторно при решении аналогичных задач без дополнительных манипуляций.

Заключение

Использовать моделей, сгенерированных при помощи AutoKeras в промышленных решениях, рекомендовано после проверки опытным ML-инженером, который может правильно интерпретировать результаты работы моделей и увидеть ошибки, т. к. AutoKeras проходит финальное тестирование [10].

Начинающим ML-инженерам рекомендовано начинать знакомство с H2O.ai, т. к. данная платформа имеет множество стабильных версий и качественную документацию, также платформа H2O.ai имеет дочерний продукт Driverless AI, автоматизирующий процесс создания признаков.

Все программные продукты, перечисленные в статье, находятся в стадии активной разработки и поддержки, и любой пользователь может следить за их развитием или принимать личное участие в расширении функционала.

Список источников

1. Elsken T., Metzen J. H., Hutter F. Neural Architecture Search: A Survey, Journal of Machine Learning Research 20 (2019) 1-21.
2. Шмиг А. Погружение в свёрточные нейронные сети, передача обучения // Хабр: сетевой журнал. 2022. [Электронный ресурс]. URL: https: // habr.com/ru/post/467967/ (13.04.2022).
3. Pandey P., Deep A. Dive into H2O's AutoML // Вебсайт продукта H2O. 2022. [Электронный ресурс] URL: https://h2o.ai/blog/a-deep-dive-into-h2os-automl/?_ga=2.136276145.1846681263.1651196999-430484785.1649755677 (13.04.2022).
4. Radečić D. Machine Learning Automation with TPOT: Build, validate, and deploy fully automated machine learning models with Python, Packt Publishing (May 7, 2021).
5. Sobrecueva L., Automated Machine Learning with AutoKeras: Deep learning made accessible for everyone with just few lines of coding, Packt Publishing (May 21, 2021).
6. Sabharwal N., Agrawal A. Up and Running Google AutoML and AI Platform: Building Machine Learning and NLP Models Using AutoML and AI Platform for Production Environment, BPB Publications (November 27, 2020).
7. Masood A. Automated Machine Learning, Packt Publishing (Feb 2021) (с. 312).
8. Géron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'relly 1st Edition (Mar 2017).
9. Шмиг А. Погружение в свёрточные нейронные сети // Хабр: сетевой журнал. 2022. [Электронный ресурс]. URL: https://habr.com/ru/post/454986// (13.04.2022).
10. Колесников Е. В. AI- и ML-решения в промышленности и производстве// Джетинфо: сетевой журнал. 2022. [Электронный ресурс]. URL: https://www.jetinfo.ru/ml-v-promyshlennosti-zdes-i-sejchas/// (13.04.2022).

Информация об авторах / Information about the Authors

Игорь Владимирович Низамов,
студент группы КСМ-20-1,
Институт информационных технологий
и анализа данных,
Иркутский национальный исследовательский
технический университет,
664074, г. Иркутск, ул. Лермонтова, 83,
Российская Федерация,
igor.nizamov.off@gmail.com

Igor V. Nizamov,
Student,
Institute of Information Technology and Data Analysis,
Irkutsk National Research Technical University,
83 Lermontov St., Irkutsk 664074,
Russian Federation,
igor.nizamov.off@gmail.com

Денис Александрович Еловенко,
кандидат технических наук, доцент,
доцент кафедры конструирования
и стандартизации в машиностроении,
Институт авиамашиностроения и транспорта,
Иркутский национальный исследовательский
технический университет,
664074, г. Иркутск, ул. Лермонтова, 83,
Российская Федерация,
elovenko03@gmail.com

Denis A. Elovenko,
Cand. Sci. (Technics),
Professor of Design and Standardization
in Mechanical Engineering,
Institute of Aircraft Engineering and Transport,
Irkutsk National Research Technical University,
83 Lermontov St., Irkutsk 664074,
Russian Federation,
elovenko03@gmail.com