https://doi.org/10.25559/SITITO.020.202403.656-665
УДК 004.032.26

Оригинальная статья

Сравнительный анализ оптимизации гиперпараметров средствами Optuna и Hyperopt для сверточных нейронных сетей

Т. А. Самойлова
ФГБОУ ВО «Смоленский государственный университет», г. Смоленск, Российская Федерация
Адрес: 214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4
tatsamoilova24@gmail.com

Аннотация

Управление процессом обучения нейросетевой модели выполняется выбором оптимальных гиперпараметров, оказывающих существенное влияние на его качество и производительность. Это влияние было доказано как теоретически, так и эмпирически многими исследованиями. Задача трудоемка, если выбирается ручной поиск. Наиболее распространенные переборные методы решения включают поиск по сетке, случайный поиск и последовательную оптимизацию на основе моделей, в которой процедура оценки целевой функции достаточно оперативна. Но все эти методы создают проблемы в приложениях со сверточными нейронными сетями, где пространство параметров настолько велико, что даже сокращенный перебор их возможных комбинаций затратен по требуемым вычислительным мощностям. Альтернативой являются инструменты автоматической настройки гиперпараметров, совместимые с фреймворками машинного обучения и использующие быстродействующие вероятностные оценки целевой функции с дополнительными механизмами. В работе сравниваются производительности обучения сверточных нейронных сетей с использованием именно таких инструментов – Python-библиотек автоматической оптимизации гиперпараметоров – Hyperopt и Optuna. Их сравнительный анализ выполнен для приложений классификации изображений. Показано, что применение библиотек дает возможность преодолеть наиболее важные проблемы оптимизации гиперпараметров таких приложений, включая большую размерность их пространства поиска и чувствительность к выбору настраиваемых гиперпараметров. Сравнивая производительность модели по точности и потерям обучения, можно прийти к выводу, что оба метода оптимизации гиперпараметров эффективны, обеспечивая точность обучения более 99% и уровень потерь менее 0.03. Реализация алгоритма оптимизации в пакете Optuna оказалась немного лучше, чем в Hyperopt, обеспечивая высокие показатели производительности и на обучающих и на тестовых данных.

Ключевые слова: сверточная нейронная сеть, оптимизация гиперпараметров, Optuna, Hyperopt

Конфликт интересов: автор заявляет об отсутствии конфликта интересов.

Для цитирования: Самойлова Т. А. Сравнительный анализ оптимизации гиперпараметров средствами Optuna и Hyperopt для сверточных нейронных сетей // Современные информационные технологии и ИТ-образование. 2024. Т. 20, № 3. С. 656-665. https://doi.org/10.25559/SITITO.020.202403.656-665

Original article

Comparative Analysis of Hyperparameter Optimization Using Optuna and Hyperopt for Convolutional Neural Networks

T. A. Samoilova
Smolensk State University, Smolensk, Russian Federation
Address: 4 Przhevalsky St., Smolensk 214000, Russian Federation
tatsamoilova24@gmail.com

Abstract

The process of training a neural network model is controlled by selecting optimal hyperparameters, which have a significant impact on its quality and performance. This impact has been confirmed both theoretically and empirically by numerous studies. If manual search is selected, the task can be labor-intensive. The most common enumeration methods include grid search, random search, and sequential model-based optimization, in which the procedure for estimating the objective function is quite fast. But all these methods create problems in applications with convolutional neural networks, where the parameter space is so large that even a shortened enumeration of their possible combinations is expensive in terms of the required computing power. An alternative is automatic hyperparameter tuning tools compatible with machine learning frameworks and using high-speed probabilistic estimates of the objective function with additional mechanisms. The paper compares the performance of training convolutional neural networks using exactly such tools – Python libraries for automatic hyperparameter optimization – Hyperopt and Optuna. Their comparative analysis is performed for image classification applications. It is shown that the use of libraries makes it possible to overcome the most important problems of hyperparameter optimization of such applications, including the large dimensionality of their search space and sensitivity to the choice of adjustable hyperparameters. Comparing the performance of the model in terms of accuracy and training loss, it can be concluded that both hyperparameter optimization methods are effective, providing training accuracy of more than 99% and a loss level of less than 0.03. The implementation of the optimization algorithm in the Optuna package turned out to be slightly better than in Hyperopt, providing high performance indicators on both training and test data.

Keywords: convolutional neural network, hyperparameter optimization, Optuna, Hyperopt

Conflict of interests: The author declares no conflict of interests.

For citation: Samoilova T.A. Comparative Analysis of Hyperparameter Optimization Using Optuna and Hyperopt for Convolutional Neural Networks. *Modern Information Technologies and IT-Education.* 2024;20(3):656-665. https://doi.org/10.25559/SITITO.020.202403.656-665

1. Введение

Сверточные нейронные сети *(Convolutional Neural Network, CNN)* широко применяются во многих областях, таких как классификация изображений [1], распознавание изображений и видео [2], обработка естественного языка [3]. Существует множество гиперпараметров, которые фиксируются до процесса обучения CNN. Они не являются неизменной частью алгоритма обучения или входных данных, а настраиваются вне процедуры обучения. Их роль заключается в управлении емкостью моделей и соответствия данным с целью повышения качества обучения. Например, есть гиперпараметры, определяющие размер партии обучающих данных, число фильтров, функции активации и сдвиги в сверточных слоях. Гиперпараметры влияют на то, как хорошо обучается модель [4], обеспечивая необходимые метрики качества обучения. Их значения варьируются в зависимости от конкретной предметной области, использующей алгоритм, что приводит к необходимости оптимизации для каждого конкретного набора входных данных. Оптимизация гиперпараметров *(Hyperparameter Optimization, HPO)* – процесс поиска правильной комбинации их значений для достижения максимальной производительности при работе с данными, обеспечивающий разумное время обучения [5]. HPO считается самой сложной частью построения моделей машинного обучения. Настройка гиперпараметров состоит из определения пространства гиперпараметров и построения функции оптимизации, которую необходимо минимизировать или максимизировать. Поскольку такой алгоритм невозможно определить одной или несколькими формулами, задача HPO фактически сводится к задаче перебора, который желательно сократить. Но критический фактор состоит в количестве различных анализируемых комбинаций гиперпараметров. Интуитивно понятно, что чем больше количество комбинаций гиперпараметров исследовано, тем больше шансов получить более эффективную модель. Но в то же время это приводит к большим вычислительным затратам, поскольку в конечном итоге мы будем обучать большое число моделей. Кроме того, важно определить, какие именно гиперпараметры оказывают большее влияние на производительность моделей глубокого обучения. Один из способов поиска подходящих гиперпараметров – ручная настройка, когда оценивается конкретный набор параметров, как он работает. Затем устанавливается другой набор параметров, и проверятся, улучшает ли он производительность. Способ хорош для простых линейных или древовидных моделей машинного обучения. Его реализация в обучении CNN обычно дает плохие результаты [6]. Необходимость автоматической HPO здесь особенно важна, поскольку пространство параметров настолько велико, что перебор всех возможных комбинаций невозможен. По умолчанию, библиотеки глубокого обучения нейросетей (в Python это TensorFlow и PyTorch) используют готовый набор значений гиперпараметров. Однако исследователи, на основании большого числа экспериментов для CNN установили [7], что увеличение размера набора обучающих данных и настройка гиперпараметров с помощью специальных алгоритмов может дополнительно улучшить производительность обучения моделей CNN. Испытания проводились в различных областях, таких как здравоохранение [8], биология [9], образование [10], промышленность [11], финансовое прогнозирование [12], сельское хозяйство [13], обнаружение сетевых вторжений [14]. Для повышения быстродействия настройку можно выполнять на графических или тензорных процессорах [15].

2. Алгоритмы подбора гиперпараметров

Известные методы подбора гиперпараметров делятся на два класса. Первый основан на методе сокращенного перебора, второй – на оптимизации. Самый простой способ сократить перебор с целью поиска подходящих гиперпараметров – разделить допустимый диапазон каждого параметра на равномерно распределенные значения, а затем просто заставить компьютер перебрать все комбинации значений. Такой перебор называется поиском по сетке [16]. Он выполняет исчерпывающий поиск посредством обучения модели глубокого обучения со всеми возможными комбинациями гиперпараметров. Затем, после оценки результатов по заданной метрике, определяет гиперпараметры, обеспечивающие наилучшую производительность. Этот метод поиска широко используется. Например, в модели создания биоматериалов [17], гиперпараметры регрессии для многослойного персептрона были выбраны с помощью поиска по сетке. В другом исследовании [18] поиск по сетке применен для настройки гиперпараметров модели LSTM *(Long Short-Term Memory).* Хотя этот метод автоматически реализует процесс перебора, он быстро теряет свою эффективность при более сложных моделях из-за увеличения количества гиперпараметров и расширения диапазона их значений [19]. Поскольку в этих случаях обучение моделей становится затратным как по времени, так и по требуемым вычислительным мощностям, перебор всех комбинаций невозможен. Другой способ сокращенного перебора с целью поиска подходящих гиперпараметров – случайный поиск [20]. Вместо систематического перебора каждой отдельной комбинации, пробуется несколько комбинаций параметров совершенно случайно. J. Bergstra [21] показал, что случайный поиск, имея те же преимущества, что и поиск по сетке, имеет гораздо большую эффективность, особенно в многомерном пространстве. Тем не менее, при случайном выборе комбинаций существует вероятность не рассмотреть лучшую комбинацию, особенно в сложных моделях CNN.

Наиболее распространенный способ оптимизации гиперпараметров – последовательная оптимизация по модели *(Sequential Model-Based Optimization, SMBO),* также известная как байесовская оптимизация [22]. Она использует подход, основанный на теореме Байеса, сопоставляющей гиперпараметры с вероятностью оптимизации целевой функции. При последовательном поиске выбирается несколько гиперпараметров, и после оценки их качества решается, где проводить следующую выборку. Методы SMBO использованы в [23], для моделей глубокого обучения, основанных на нейронных сетях. В исследовании [24, 25] этим методом оптимизации обучаются модели LSTM для прогнозирования мощности ветра. Хотя при сравнении точности и эффективности предложенного метода с традиционным методом поиска по сетке авторы доказали лучшую производительность байесовкой оптимизации, они не исследовали другие методы автоматической HPO, совместимые с фреймворками машинного обучения. Это Python – пакеты Hyperopt [26], Optuna [27]; Scikit-optimize[1] и Mango [28]. Последний предназначен для работы в распределен-

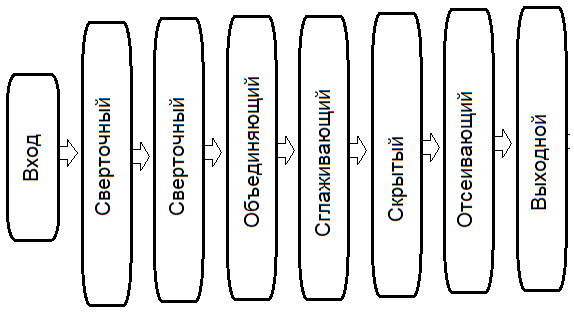
[1] Louppe G. Bayesian optimisation with scikit-optimize [Electronic resource] // PyData Amsterdam. Amsterdam, Netherlands, 2017. Available at: https://orbi.uliege.be/handle/2268/226433 (accessed 13.07.2024).

ной вычислительной среде. Пакет Hyperopt может выполнять и глобальную HPO, быстро приводя к хорошим результатам. Исследования глобальной оптимизации широко представлены работами Карпенко А.П., например [29].

В настоящем исследовании рассматриваются два метода автоматическиой HPO – Optuna и Hyperopt, оценивается их производительность по точности и потерям обучения CNN. Результаты сравниваются между собой и с результатами обучения при ручной оптимизации на обучающем и тестовом наборах. Для эксперимента использована небольшая задача распознавания изображений с набором данных MNIST *(Modified National Institute of Standards and Technology)*, состоящем из 28×28 изображений в оттенках серого рукописных цифр (0-9). Обучающий набор содержит 60000 примеров, тестовый – 10000 примеров. Размеры наборов небольшие, поэтому обучение можно провести достаточно быстро, но в более реалистичных задачах обучение может занять часы, дни или даже недели на очень быстром компьютере. Поэтому нужен такой метод оптимизации, который ищет гиперпараметры максимально эффективно. Результаты эксперимента показывают, что обе Python – библиотеки автоматичекой HPO эффективны в настройке CNN. Выбранные методы оптимизации, основанные на алгоритме TPE *(Tree-structured Parzen Estimator,* древовидная оценка Парзена) [19] с дополнительными улучшающими механизмами, обеспечивают высокие показатели производительности. Следует отметить, что применение передовых автоматических методов оптимизации в данном исследовании дает возможность преодолеть наиболее важные проблемы и трудности HPO при обучении CNN, включая большую размерность их пространства поиска и чувствительность обучения к значениям гиперпараметров.

3. Выбор архитектуры CNN

С целью упрощения эксперимента, но без ограничения общности, была выбрана архитектура CNN для распознавания и классификации изображений. Она определяется размером набора данных, доступными вычислительными ресурсами и желаемым высоким уровнем точности. С учетом этого, предлагается следующая семислойная архитектура, слои которой изображены на рисунке 1.



Р и с. 1. Слои архитектуры экспериментальной CNN

F i g. 1. Layers of the experimental CNN architecture

Источник: здесь и далее в статье все рисунки составлены автором.

Source: Hereinafter in this article all figures were drawn up by the author.

Слои 1, 2 – это сверточные слои Conv2D, сворачивающие изображение, используя фильтры определенного размера. Основные настраиваемые гиперпараметры сверточных слоев: размер пакета *(batch size)* – число выборок из набора входных данных, количество фильтров *(filters)*, число ядер *(kernel_size)*, глубина слоя *(conv_depth)*, функция активации *(activation)*, шаг *(stride)* – сдвиг фильтра на определенное число пикселей в ходе операции свертки.

Слой 3 – пулинговый (объединяющий, MaxPooling) слой, суммирующий признаки сверточного слоя. Настраиваемые гиперпараметры – тип пула и максимальный размер ядра пула.

Слой 4 – слой Flatten сглаживает выходные данные, полученные от слоя 4, и эти сглаженные выходные данные передаются в слой 5.

Слой 5 – скрытый слой Dense – это простой слой нейронов, в котором каждый нейрон получает входные данные от всех нейронов предыдущего слоя, его гиперпараметр – число нейронов *(hidden_size)*.

Слой 6 – слой отсева Dropout, который случайным образом игнорирует некоторое количество выходов слоя, предотвращая переобучение модели. Настраиваемый гиперпараметр – вероятность, при которой выходы слоя игнорируются *(dropout rate)*.

Слой 7 – это выходной полностью связанный слой Dense, имеющий 10 нейронов для 10 классов выходных данных, который использует функцию активации *softmax*. Настраиваемые гиперпараметры – шаблон связности, регуляризация веса.

Кроме гиперпараметров слоев значительное влияние на производительность и качество CNN оказывают гиперпараметры процесса обучения. Главный из них – это скорость обучения *(learning rate)*, устанавливаемая в коде обучающей программы. Скорость обучения определяет размер шага, с которым параметры модели обновляются во время обучения. Она играет решающую роль в управлении сходимостью и стабильностью процесса оптимизации. Также в коде устанавливается имя алгоритма оптимизации гиперпараметров.

4. Используемые пакеты оптимизации

На основе анализа публикаций, были выбраны две Python-библиотеки для автоматической настройки HPO.

4.1 Optuna

Позволяет реализовать HPO несколькими алгоритмами[2]. Библиотека простота как в настройке пространства поиска, так и в эффективной выборке алгоритма оптимизации целевой функции. Optuna пробует различные комбинации параметров и оценивает, как изменяется целевая функция при каждой конфигурации, то есть каждом обучении модели. Из этих испытаний фреймворк строит вероятностную модель, используемую для оценки того, какие значения параметров, скорее всего, дадут лучшие результаты. Разработчик может самостоятельно задать пространство для поиска гиперпараметров, используя базовый синтаксис Python. В пространство поиска CNN можно включить даже число сверточных слоев. Типы гиперпараметров для CNN (категориальные, целочисленные)

[2] Optuna Dashboard [Электронный ресурс]. URL: https://optuna-dashboard.readthedocs.io/en/latest/ (дата обращения: 13.07.2024).

определяются автоматически. Optuna имеет различные алгоритмы выполнения процесса HPO для вычисления целевой функции. Наиболее распространенные:

- GridSampler – поиск по сетке,
- RandomSampler – случайный поиск,
- TPESampler – использует древовидную оценку Парзена.

В эксперименте определен алгоритм TPESampler, именно он позволяет достичь главной цели оптимизации – минимизации целевой функцци – значения ошибки *(loss)* обучения CNN. Алгоритм основан на вероятностной байесовской оптимизации, но его преимущество в том, что он обрабатывает сложные гиперпараметрические связи через древовидную структуру. При этом Optuna предоставляет функцию обрезки ветвей дерева, помогающую преждевременно завершить прогоны, которые не являются оптимальными. Для этой цели промежуточные целевые значения отслеживаются, и те, которые не соответствуют предопределенным условиям, прекращаются. TPESampler также поддерживает категориальные переменные, участвующие в обучении CNN, которые традиционная байесовская оптимизация не поддерживает. Целевая функция для задачи классификации изображений – это функция минимизации потерь, которая получает значения гиперпараметров в качестве входных данных из пространства поиска и возвращает потери.

4.2 Hyperopt

В качестве второго метода настройки в эксперименте используется Python – библиотека автоматической HPO Hyperopt [30], позволяющая для CNN выполнить байесовскую оптимизацию с дополнительными механизмами. Собственно, в библиотеке реализованы три алгоритма оптимизации:

- Random Search – случайный поиск,
- TPE – байесовская оптимизация,
- Adaptive TPE – байесовская оптимизация с дополнительными механизмами.

В [28] описаны преимущества Hyperopt для TPE. В настоящем исследовании применен новый, добавленный недавно, алгоритм AdaptiveTPE с дополнительными механизмами[3] для гибкой настройки более сложных параметров CNN, таких как изменение количества слоев определенных типов, количества нейронов в слое, или даже типа слоя. Для использования Hyperopt необходимо описать целевую функцию и пространство поиска. Поиск оптимальных гиперпараметров начинается в Hyperopt с нескольких случайных комбинаций. В процессе оптимизации происходит обучение модели CNN с определенными алгоритмом значениями параметров и прогнозируется целевая функция минимизации потерь обучения. Затем ошибка прогноза оценивается и возвращается оптимизатору. Оптимизатор решает, какие значения изменять, и повторяет новый цикл оптимизации. Целевая функция работает так же, как в методе Optuna. В обоих методах она принимает несколько входных данных, а именно:

- функцию для минимизации потерь *(loss)*,
- определенное пространство поиска – набор гиперпараметров,
- алгоритм поиска,
- максимальное количество попыток запуска процесса обучения CNN.

При работе с Hyperopt и моделями CNN, представленными объектами библиотеки Keras, удобно использовать оболочку Hyperas. Она позволяет использовать всю мощь Hyperopt без необходимости детально изучать его синтаксис. Вместо этого необходимо просто определить свою модель средствами Keras, а затем использовать простую шаблонную нотацию для определения диапазонов гиперпараметров и алгоритма оптимизации целевой функции. Для хранения всех промежуточных значений гиперпараметров и Hyperopt и Optuna используют объект Trial, обеспечивающий возможность многократного запуска обучения с разными наборами гиперпараметров. Сразу после начала оптимизации целевая функция принимает объект Trial в качестве аргумента, который используется для сохранения значений гиперпараметров. Далее в результатах эксперимента попытки обучения (запуски) будут называться триалами.

5. Результаты исследования

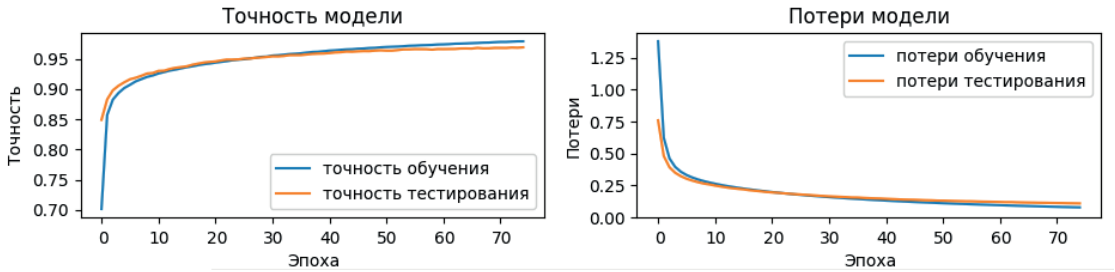
Эксперимент настоящей работы проведен на базе данных MNIST средствами библиотек Python. Исходные данные модели классификации – набор данных, содержащий обучающий набор из 60 000 изображений и тестовый набор из 10 000 изображений рукописных цифр в оттенках серого цвета. Изображения нормализованы по размеру и центрированы в фиксированном размере 28×28 пикселей, что позволяет прикладывать минимальные усилия на предварительную обработку и форматирование. Фрагмент данных представлен на рисунке 2.

Р и с. 2. Фрагмент данных MNIST

F i g. 2. MNIST data fragment

Для измерения качества обучения CNN используются метрики её производительности – точность *(accuracy)* и потери модели *(loss)*. Производительность оценивается на обучающем и тестовом наборах данных. С точки зрения оценки времени обучения, желательно получить максимальную точность и минимальные потери на небольшом числе эпох. На рисунке 3 представлены результаты обучения модели средствами библиотек TensorFlow и Keras. Установленные вручную значения основных гиперпараметров: batch_size = 32, num_epochs = 75, kernel_size = 3, conv_depth_1 = 32, conv_depth_2 = 64 , hidden_size = 512.

[3] Gray S. Levers for Improving TPE [Электронный ресурс] // Learning to Optimize. August 23, 2023. URL: https://electricbrain.io/blog/learning-to-optimize/ (дата обращения: 13.07.2024).

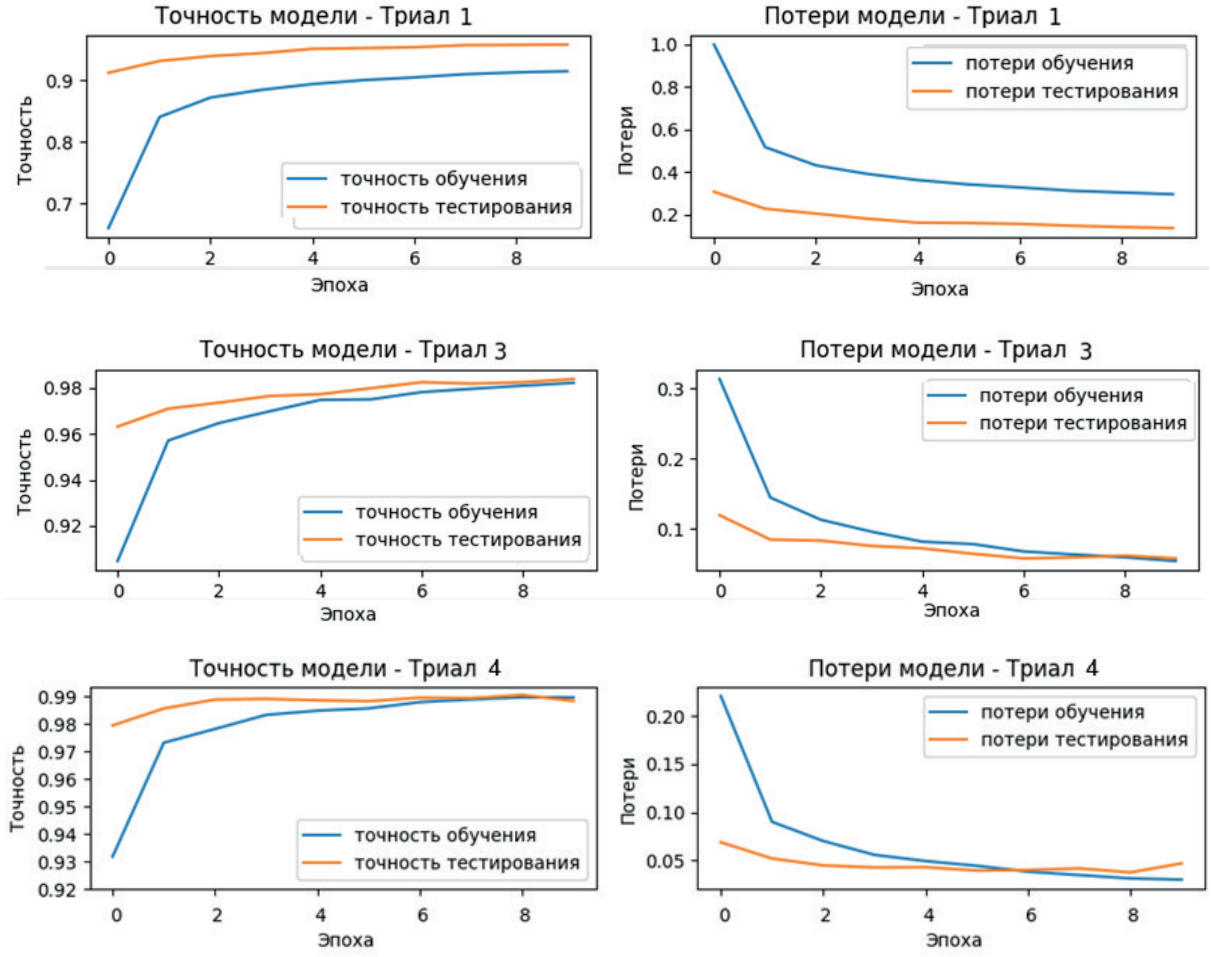


Р и с. 3. Обучение модели при ручной настройке гиперпараметров

F i g. 3. Training a model with manual hyperparameter tuning

Графики показывают, что оптимальные значения точности и потерь (0.96 и 0.1) определяются на 60-70 эпохах, что свидетельствует о большом времени обучения. Малые расхождения для обучающих и тестовых данных говорят об отсутствии необходимости переобучения модели.

В экспериментах с Optuna и Hyperopt есть функции, которые необходимо настраивать вручную. Например, алгоритм поиска гиперпараметров, максимальное число триалов целевой функции, диапазоны гиперпараметров, число эпох обучения CNN. Чтобы сохранить общую структуру CNN, количество слоев модели не оптимизировалось. На рисунке 4 представлен процесс обучения модели в среде Optuna алгоритмом оптимизации TPESampler путем построения кривых точности и потерь на протяжении 10 эпох. Верхние графики соответствуют начальному значению триала, а нижние – последнему – четвертому.



Р и с. 4. Сравнение производительности моделей в начале оптимизации (вверху), в процессе (середина) и в конце (внизу) средствами Optuna

F i g. 4. Comparison of model performance at the start of optimization (top), during (middle), and at the end (bottom) using Optuna

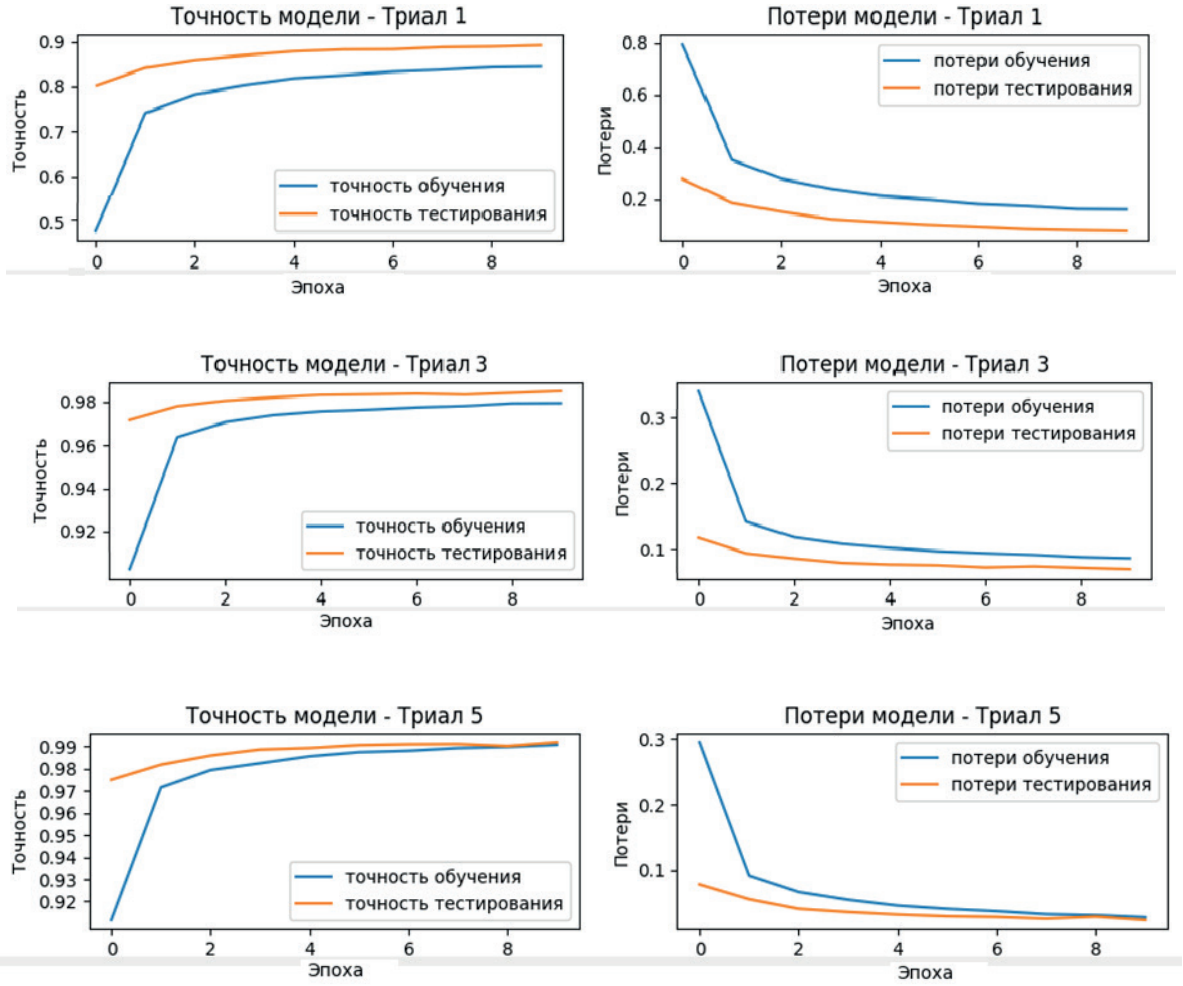
Графики показывают, что уже на четвертом триале оптимальные значения точности и потерь даже на тестовых данных уже на 5-6 эпохе достигают 0.99 и 0.02, что свидетельствует о быстром и эффективном обучении. Эффективность обучения подтверждается малым расхождением показателей производительности для обучающих и тестовых данных в последнем триале. Разница точности на обучающих и тестовых данных в первом триале для пятой эпохи составляет приблизительно 20%. Однако на последних триалах значение точности обучения и проверки имеет приблизительно постоянное значение 2%, показывая, что модель работает хорошо и не переобучается. Библиотека сама выбирает триал с оптимальными гиперпараметрами. В настоящем эксперименте это четвертый триал. Ему соответствуют следующие значения гиперпараметров: batch_size:128, filters: 64, kernel_size: 3, strides: 1, activation: relu, conv_depth_1 = 32, conv_depth_2 = 64, dropout_rate: 0.504507, hidden_size = 128, learning_rate: 0.0005067.

Процесс обучения с HPO в среде Hyrgopt выполнялся при установленном алгоритме оптимизации AdaptiveTPE. Для облегчения настройки гиперпараметров, применена библиотека Hyperas. Средствами Hyperas удобно настраивать гиперпараметры, если необходимо попробовать несколько конкретных значений. Например, так осуществлялось задание разных значений для batch_size:

batch_size={{choice([32, 64, 128])}}

Во время каждого обучения выбрано и опробовано одно из значений.

Результаты оптимизации гиперпараметров средствами Hyperopt представлены на рисунке 5.



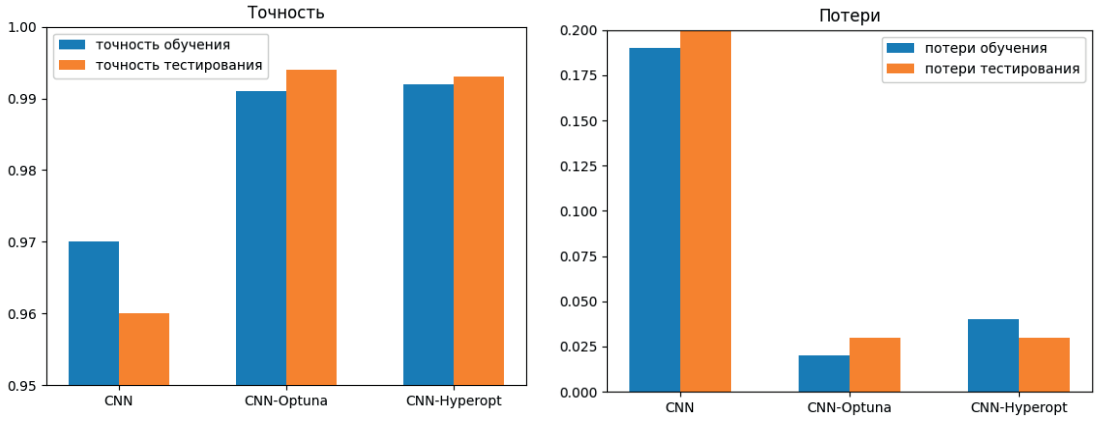
Р и с. 5. Сравнение производительности моделей в начале оптимизации (вверху), в процессе (середина) и в конце (внизу) средствами Hyperopt

F i g. 5. Comparison of model performance at the start of optimization (top), during (middle) and at the end (bottom) using Hyperopt

Из приведенных графиков видно, что уже на пятом триале оптимальная точность достаточно высока (0.99), а потери незначительно малы (0.03) и определяются на 6-7 эпохах, показывая высокую производительность обучения CNN. Разница точности на обучающих и тестовых данных в первом триале для пятой эпохи составляет приблизительно 25%. На четвертом триале с оптимальными значениями гиперпараметров разница точности обучения и проверки имеет значение.

3%. Она немного больше по сравнению с 2% в случае HPO в Optuna. Найденные оптимальные значения гиперпараметров: batch_size:64, filters: 32, kernel_size: 3, strides: 1, activation: relu, conv_depth_1 = 32, conv_depth_2 = 64, dropout_rate: 0,45328, hidden_size = 256, learning_rate: 0,00182853.

Рисунок 6 визуализирует сравнение рассмотренных средств HPO для модели CNN на обучающих и тестовых данных.



a) b)

Р и с. 6. Сравнение точности (a) и потерь (b) моделей на обучающих и тестовых данных для различных средств HPO

F i g. 6. Comparison of accuracy (a) and loss (b) of models on training and test data for different HPO tools

Основываясь на результатах разных средств HPO, можно утверждать, что обучение CNN с применением библиотек автоматической оптимизации выполняется более эффективно, чем в случае ручной настройки гиперпараметров. Обе библиотеки – Optuna и Hyperopt – работают превосходно, но Optuna немного лучше с точки зрения производительности для обучающих данных. На тестовых данных результаты обеих библиотек одинаково хороши.

6. Обсуждение и заключение

В работе представлено сравнение результатов оптимизации модели CNN путем настройки её гиперпараметров для классификации изображений средствами Python-библиотек Optuna и Hyperopt. Гиперпараметры включают размер партии обучающих данных, количество ядер и сдвиги в сверточных слоях, функции активации, число нейронов в скрытом слое, скорость обучения оптимизатора. Для каждой библиотеки выбраны алгоритмы оптимизации целевых функций: TPESampler для Optuna и AdaptiveTPE для Hyperopt. Выбраны средства определения пространства поиска в обеих библиотеках. Разработано программное обеспечение для обучения модели CNN, использующее возможности этих библиотек и включающее средства визуализации в ходе обучения. Проведен экспериментальный анализ на основании сравнения метрик производительности моделей – точности и потерь для ручной настройки гиперпараметров и их автоматической оптимизации. Построены графики и диаграммы производительности моделей на обучающем и тестовом наборах данных. Результаты демонстрируют важность автоматической HPO и показывают, что для CNN обе библиотеки являются отличным вариантом повышения их производительности.

References

[1] Zhao X., et al. A review of convolutional neural networks in computer vision. *Artificial Intelligence Review.* 2024;57(4):99. https://doi.org/10.1007/s10462-024-10721-6

[2] Wang J., Li X., Jin Y., Zhong Y., Zhang K., Zhou C. Research on Image Recognition Technology Based on Multimodal Deep Learning. In: 2024 IEEE 2nd International Conference on Image Processing and Computer Applications (ICIPCA). Shenyang, China: IEEE Press; 2024. p. 1363-1367. https://doi.org/10.1109/ICIPCA61593.2024.10709051

[3] Sruthi S., Trinath V., Jayanth V., Balaji V.P., Singh T., Mandal A. Natural Language Processing for Sentiment Analysis with Deep Learning. In: 2024 3rd International Conference for Innovation in Technology (INOCON). Bangalore, India: IEEE Press; 2024. p. 1-6. https://doi.org/10.1109/INOCON60754.2024.10511769

[4] Shekhar S., Bansode A., Salim A. A Comparative study of Hyper-Parameter Optimization Tools. In: 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE). Brisbane, Australia: IEEE Press; 2021. p. 1-6. https://doi.org/10.1109/CSDE53843.2021.9718485

[5] Du X., Xu H., Zhu F. Understanding the effect of hyperparameter optimization on machine learning models for structure design problems. *Computer-Aided Design.* 2021;135:103013. https://doi.org/10.1016/j.cad.2021.103013

[6] Yang L., Shami A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing.* 2020;415:295-316. https://doi.org/10.1016/j.neucom.2020.07.061

[7] Kiziloluk S., Sert E. COVID-CCD-Net: COVID-19 and colon cancer diagnosis system with optimized CNN hyperparameters using gradient-based optimizer. *Medical & Biological Engineering & Computing*. 2022;60(6):1595-1612. https://doi.org/10.1007/s11517-022-02553-9
[8] Raiaan M.A.K., et al. A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks. *Decision Analytics Journal*. 2024:11:100470. https://doi.org/10.1016/j.dajour.2024.100470
[9] Irmawati, Chai R., Basari, Gunawan D. Optimizing CNN Hyperparameters for Blastocyst Quality Assessment in Small Datasets. *IEEE Access*. 2022;10:88621-88631. https://doi.org/10.1109/ACCESS.2022.3196647
[10] Muhajir D., et al. Improving classification algorithm on education dataset using hyperparameter tuning. *Procedia Computer Science.* 2022;197:538-544. https://doi.org/10.1016/j.procs.2021.12.171
[11] Chernigovskaya M., et al. Hyper-parameter Optimization in the context of Smart Manufacturing: a Systematic Literature Review. *Procedia Computer Science*. 2024;232:804-812. https://doi.org/10.1016/j.procs.2024.01.080
[12] Hoque K.E., Aljamaan H. Impact of Hyperparameter Tuning on Machine Learning Models in Stock Price Forecasting. *IEEE Access*. 2021;9:163815-163830. https://doi.org/10.1109/ACCESS.2021.3134138
[13] Erkan U., Toktas A., Ustun D. Hyperparameter optimization of deep CNN classifier for plant species identification using artificial bee colony algorithm. *Journal of Ambient Intelligence and Humanized Computing*. 2023;14(7):8827-8838. https://doi.org/10.1007/s12652-021-03631-w
[14] Kilichev D., Kim W. Hyperparameter optimization for 1D-CNN-based network intrusion detection using GA and PSO. *Mathematics*. 2023;11(17):3724. https://doi.org/10.3390/math11173724
[15] Rakhimov M., Javliev S., Nasimov R. Parallel Approaches in Deep Learning: Use Parallel Computing. In: Proceedings of the 7th International Conference on Future Networks and Distributed Systems (ICFNDS '23). New York, NY, USA: Association for Computing Machinery; 2024. p. 192-201. https://doi.org/10.1145/3644713.3644738
[16] Liashchynskyi P., Liashchynskyi P. Grid search, random search, genetic algorithm: a big comparison for NAS. *arXiv:*1912.06059. 2019. https://doi.org/10.48550/arXiv.1912.06059
[17] Ogunsanya M., Isichei J., Desai S. Grid search hyperparameter tuning in additive manufacturing processes. *Manufacturing Letters*. 2023;35:1031-1042. https://doi.org/10.1016/j.mfglet.2023.08.056
[18] Kisvari A., Lin Z., Liu X. Wind power forecasting – A data-driven method along with gated recurrent neural network. *Renewable Energy*. 2021;163:1895-1909. https://doi.org/10.1016/j.renene.2020.10.119
[19] Wu J., et al. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *Journal of Electronic Science and Technology.* 2019;17(1):26-40. https://doi.org/10.11989/JEST.1674-862X.80904120
[20] Price W.L. Global optimization by controlled random search. *Journal of Optimization Theory and Applications*. 1983;40:333-348. https://doi.org/10.1007/BF00933504
[21] Bergstra J., Bengio Y. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*. 2021;13(2):281-305. Available at: https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf (accessed 13.07.2024).
[22] Snoek J., Larochelle H., Adams R.P. Practical Bayesian optimization of machine learning algorithms. In: Proceedings of the 26th International Conference on Neural Information Processing Systems – Vol. 2 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA; 2012. p. 2951-2959.
[23] Talathi S.S. Hyper-parameter optimization of deep convolutional networks for object recognition. In: 2015 IEEE International Conference on Image Processing (ICIP). Quebec City, QC, Canada: IEEE Press; 2015. p. 3982-3986. https://doi.org/10.1109/ICIP.2015.7351553
[24] Hanifi S., Lotfian S., Zare-Behtash H., Cammarano A. Offshore Wind Power Forecasting – A New Hyperparameter Optimisation Algorithm for Deep Learning Models. *Energies.* 2022;15(19):6919. https://doi.org/10.3390/en15196919
[25] Zha W., et al. Ultra-short-term power forecast method for the wind farm based on feature selection and temporal convolution network. *ISA transactions.* 2022;129(A):405-414. https://doi.org/10.1016/j.isatra.2022.01.024
[26] Komer B., Bergstra J., Eliasmith C. Hyperopt-Sklearn. In: Hutter F., Kotthoff L., Vanschoren J. (eds.) Automated Machine Learning. *The Springer Series on Challenges in Machine Learning (SSCML).* Cham: Springer; 2019. p. 97-111. https://doi.org/10.1007/978-3-030-05318-5_5
[27] Akiba T., et al. Optuna: A Next-generation Hyperparameter Optimization Framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19). New York, NY, USA: Association for Computing Machinery; 2019. p. 2623-2631. https://doi.org/10.1145/3292500.3330701
[28] Sandha S.S., Aggarwal M., Fedorov I., Srivastava M. Mango: A Python Library for Parallel Hyperparameter Tuning. In: ICASSP 2020 – 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Barcelona, Spain: IEEE Press; 2020. p. 3987-3991. https://doi.org/10.1109/ICASSP40776.2020.9054609
[29] Agasiev T.A., Karpenko A.P. Modern Techniques of Global Optimization. Review. *Informacionnye tehnologii* = Information Technologies. 2018;24(6):370-386. (In Russ., abstract in Eng.) https://doi.org/10.17587/it.24.370-386
[30] Bergstra J., et al. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery.* 2015;8(1):014008. https://doi.org/10.1088/1749-4699/8/1/014008

Поступила 13.07.2024; одобрена после рецензирования 16.09.2024; принята к публикации 02.10.2024.
Submitted 13.07.2024; approved after reviewing 16.09.2024; accepted for publication 02.10.2024.

Об авторе:

Самойлова Татьяна Аркадьевна, доцент кафедры прикладной математики и информатики физико-математического факультета, ФГБОУ ВО «Смоленский государственный университет» (214000, Российская Федерация, г. Смоленск, ул. Пржевальского, д. 4), кандидат технических наук, **ORCID: https://orcid.org/0000-0002-3712-327X,** tatsamoilova24@gmail.com

Автор прочитал и одобрил окончательный вариант рукописи.

About the author:

Tatyana A. Samoilova, Associate Professor of the Chair of Applied Mathematics and Informatics, Faculty of Physics and Mathematics, Smolensk State University (4 Przhevalsky St., Smolensk 214000, Russian Federation), Cand. Sci. (Eng.), **ORCID: https://orcid.org/0000-0002-3712-327X,** tatsamoilova24@gmail.com

The author has read and approved the final manuscript.