

Направление: Нелинейная цифровая обработка сигналов.

Тема: Поиск нейросетевых архитектур. Часть 2. AutoKeras.
Neural architecture search (NAS). Part 2. AutoKeras.

Авторы: к.т.н., старший инженер Huawei Антонов Лев
старший инженер Huawei Власов Роман.

Auto-Keras: An Efficient Neural Architecture Search System



Auto-Keras and AutoML: A Getting Started Guide

by Adrian Rosebrock on January 7, 2019



[Click here to download the source code to this post](#)



Так как все предыдущие реализации *NAS*, *NASNet* и *PNAS* были очень ресурсозатратными, то был предложен новый метод поиска структур, основанный на *Байесовской оптимизации и Гауссовских процессах*.

Метод основан на последовательном преобразовании структуры, направляемой Байесовской оптимизацией. Для данного метода разработана очень качественная высокоуровневая реализация, поддерживаемая сообществом, и поддерживающая последние версии языка **Python** и библиотек, на основе которых она построена.

Haifeng Jin, Qingquan Song, Xia Hu “*Auto-Keras: An Efficient Neural Architecture Search System*”, Department of Computer Science and Engineering, Texas A&M University. 2018

- <https://arxiv.org/abs/1806.10282>
- <https://autokeras.com>
- <https://github.com/keras-team/autokeras>

<https://www.pyimagesearch.com/2019/01/07/auto-keras-and-automl-a-getting-started-guide/>

Auto-Keras: An Efficient Neural Architecture Search System

2 PROBLEM STATEMENT

The general neural architecture search problem we studied in this paper is defined as: Given a neural architecture search space \mathcal{F} , the input data D divided into D_{train} and D_{val} , and the cost function $Cost(\cdot)$, we aim at finding an optimal neural network $f^* \in \mathcal{F}$, which could achieve the lowest cost on dataset D . The definition is equivalent to finding f^* satisfying:

$$f^* = \operatorname{argmin}_{f \in \mathcal{F}} Cost(f(\theta^*), D_{val}), \quad (1)$$

$$\theta^* = \operatorname{argmin}_{\theta} \mathcal{L}(f(\theta), D_{train}). \quad (2)$$

where $Cost(\cdot, \cdot)$ is the evaluation metric function, e.g., accuracy, mean squared error, θ^* is the learned parameter of f .

The search space \mathcal{F} covers all the neural architectures, which can be morphed from the initial architectures. The details of the morph operations are introduced in 3.3. Notably, the operations can change the number of filters in a convolutional layer, which makes \mathcal{F} larger than methods with fixed layer width [24].

Основная идея предлагаемого метода состоит в том, чтобы исследовать пространство поиска посредством преобразования нейронных архитектур на основе **алгоритма байесовской оптимизации (ВО)**. Традиционная байесовская оптимизация состоит из цикла из трех этапов: **обновление, генерация и наблюдение**.

В контексте NAS наш предложенный алгоритм байесовской оптимизации итеративно проводит:

- (1) **Обновление**: обучение основной модели гауссовского процесса на существующих архитектурам и узнать их **score**;
- (2) **Генерация**: генерация следующей архитектуры для наблюдения путем оптимизации “тонко определенной” функции сбора данных (*acquisition function*);
- (3) **Наблюдение**: получить **score**, путем обучения сгенерированной нейронной архитектуры.

Общая проблема поиска нейронной архитектуры, которую мы изучали в этой статье, определяется следующим образом: Учитывая пространство поиска нейронной архитектуры, входные данные D разделены на D_{train} и D_{val} и функцию потерь $Cost(\cdot)$. Эта функция нацелена на поиск оптимальной нейронной сети

где **Cost** - метрическая функция оценки, например, точность, среднеквадратичная ошибка, θ^* - изученный параметр f . \mathcal{F} – все возможное пространство нейросетей, в которое может перейти начальная архитектура.

3. NETWORK MORPHISM GUIDED BY BAYESIAN OPTIMIZATION

Преобразование сетей, направляемое Байесовской оптимизацией.

Auto-Keras: An Efficient Neural Architecture Search System

Существует несколько основных проблем при разработке метода преобразования нейронных архитектур с помощью байесовской оптимизации.

Проблема №1:

3.1 Edit-Distance Neural Network Kernel for Gaussian Process

Представление расстояния (дистанции редактирования) нейронной сети в ядре Гауссовского процесса.

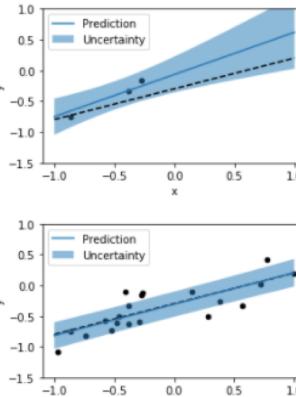
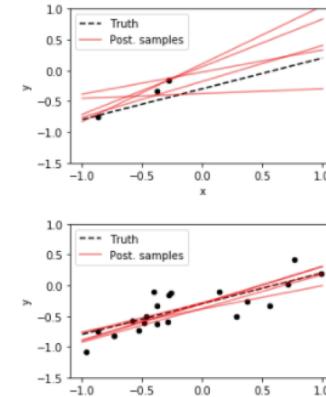
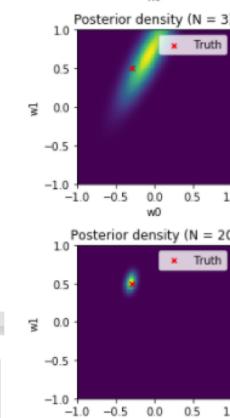
Линейная Байесовская регрессия

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$
$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(t - y(\mathbf{x}, \mathbf{w}))^2\right)$$

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^N \mathcal{N}(t_i|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1})$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$
$$\boxed{\mathbf{m}_N = \beta \mathbf{S}_N \boldsymbol{\Phi}^T \mathbf{t}}$$
$$\boxed{\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}}$$



Корреляционная составляющая / ядро Гауссовского процесса. Оценивает “близость” координат \mathbf{x} относительно друг друга в Евклидовом пространстве.

Auto-Keras: An Efficient Neural Architecture Search System

Существует несколько основных проблем при разработке метода преобразования нейронных архитектур с помощью байесовской оптимизации.

Проблема №1:

3.1 Edit-Distance Neural Network Kernel for Gaussian Process

Представление расстояния (дистанции редактирования) нейронной сети в ядре Гауссовского процесса.

Линейная Байесовская регрессия

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(t - y(\mathbf{x}, \mathbf{w}))^2\right)$$

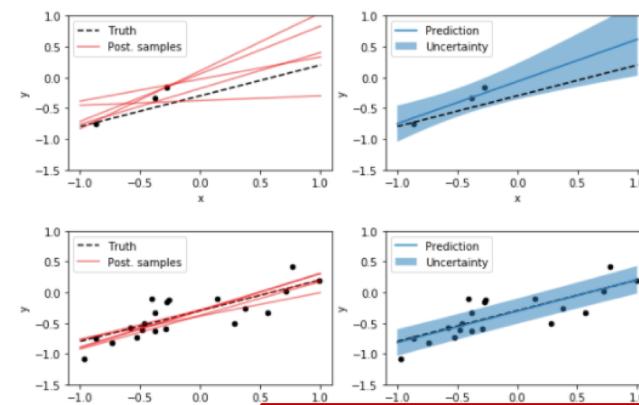
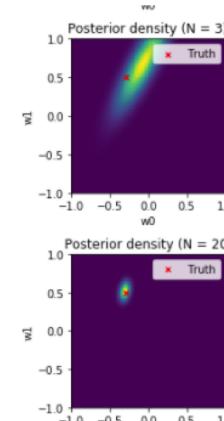
$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^N \mathcal{N}(t_i | \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i), \beta^{-1})$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

$$\mathbf{m}_N = \beta \mathbf{S}_N \boldsymbol{\Phi}^T \mathbf{t}$$

$$\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}$$

Корреляционная составляющая / ядро Гауссовского процесса. Оценивает “близость” координат \mathbf{x} относительно друг друга в Евклидовом пространстве.



Пространство NAS не является Евклидовым! И следовательно, не удовлетворяет условиям традиционного GP.

Auto-Keras: An Efficient Neural Architecture Search System

Существует несколько основных проблем при разработке метода преобразования нейронных архитектур с помощью байесовской оптимизации.

Проблема №1:

3.1 Edit-Distance Neural Network Kernel for Gaussian Process

Представление расстояния (дистанции редактирования) нейронной сети в ядре Гауссовского процесса.

NAS не является евклидовым пространством, и оно не удовлетворяет условиям традиционного гауссовского процесса (GP). Непосредственная векторизация нейронной архитектуры нецелесообразна из-за неопределенного количества слоев и параметров, которые она может содержать. Поскольку гауссовский процесс - это метод ядра, а не векторизация нейронной архитектуры, мы предлагаем решить эту проблему путем разработки функции ядра нейронной сети.

Линейная Байесовская регрессия

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x})$$

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

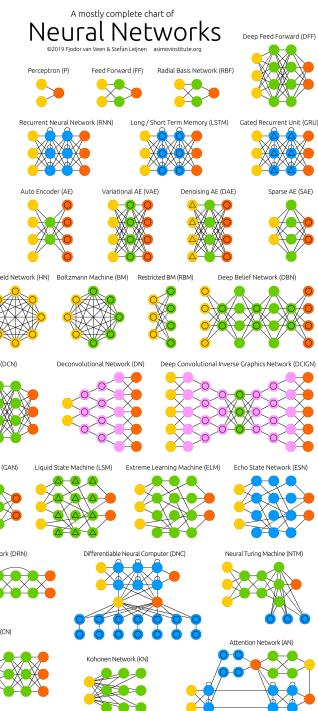
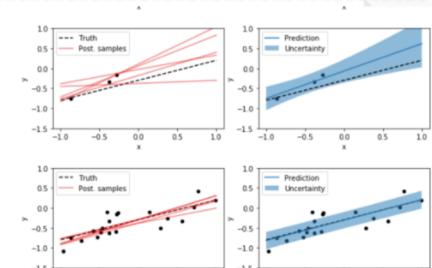
$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|\mathbf{y}(\mathbf{x}, \mathbf{w}), \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}(t - y(\mathbf{x}, \mathbf{w}))^2\right)$$

$$p(t|\mathbf{X}, \mathbf{w}, \beta) = \prod_{i=1}^N \mathcal{N}(t_i | \mathbf{y}^T \Phi(\mathbf{x}_i), \beta^{-1})$$

$$\Phi = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

$\mathbf{m}_N = \beta \mathbf{S}_N \Phi^T \mathbf{t}$
 $\mathbf{S}_N^{-1} = \alpha \mathbf{I} + \beta \Phi^T \Phi$

Корреляционная составляющая / ядро Гауссовского процесса. Оценивает “близость” координат \mathbf{x} относительно друг друга в Евклидовом пространстве.



Auto-Keras: An Efficient Neural Architecture Search System

Существует несколько основных проблем при разработке метода преобразования нейронных архитектур с помощью байесовской оптимизации.

Проблема №1:

Предположим, имеется 2 сети f_a и f_b . В Предлагаем ядро с дистанцией редактирования для нейронных сетей. **Расстояние редактирования здесь означает, сколько операций необходимо для преобразования одной нейронной сети в другую.** Функция ядра определяется как:

$$\kappa(f_a, f_b) = e^{-\rho^2(d(f_a, f_b))}, \quad (3)$$

- Где $d(f_a, f_b)$ определяет расстояние редактирования между двумя сетями f_a и f_b , для преобразования одной сети в другую.
- ρ – это функция преобразования расстояния из одного пространства в другое на основе теоремы Бургейна.

$$d(f_a, f_b) = D_l(L_a, L_b) + \lambda D_s(S_a, S_b), \quad (4)$$

Где D_l – расстояние редактирования слоев или минимальное количество изменений для того чтобы структура слоев одной сети превратилась в другую, игнорируя skip-connections.

$L_a = \{l_a^{(1)}, l_a^{(2)}, \dots\}$ and $L_b = \{l_b^{(1)}, l_b^{(2)}, \dots\}$ - наборы слоев для нейронных сетей f_a и f_b .

D_s – расстояние редактирования skip-connections для преобразования набора пропущенных соединений одной сети в набор другой.

$S_a = \{s_a^{(1)}, s_a^{(2)}, \dots\}$ and $S_b = \{s_b^{(1)}, s_b^{(2)}, \dots\}$ - наборы skip-connections для моделей f_a и f_b .

λ – коэффициент баланса между расстояниями преобразования слоев и пропущенных соединений.

Auto-Keras: An Efficient Neural Architecture Search System

- Расчет D_l .

если $|L_a| < |L_b|$ (количество слоев), то расстояние редактирования слоев для f_a и f_b равно

$$D_l(L_a, L_b) = \min \sum_{i=1}^{|L_a|} d_l(l_a^{(i)}, \varphi_l(l_a^{(i)})) + \left| |L_b| - |L_a| \right|, \quad (5)$$

$\varphi_l : L_a \rightarrow L_b$ – это инъективная функция отображения слоев L_a в L_b , удовлетворяющая следующему условию $\forall i < j, \varphi_l(l_a^{(i)}) < \varphi_l(l_a^{(j)})$

если слои отсортированы в топологическом порядке.

$d_l(\dots)$ – расстояние редактирования (расширения) слоя до другого слоя:

$$d_l(l_a, l_b) = \frac{|w(l_a) - w(l_b)|}{\max[w(l_a), w(l_b)]}, \quad (6)$$

где $w(l)$ – это ширина слоя l .

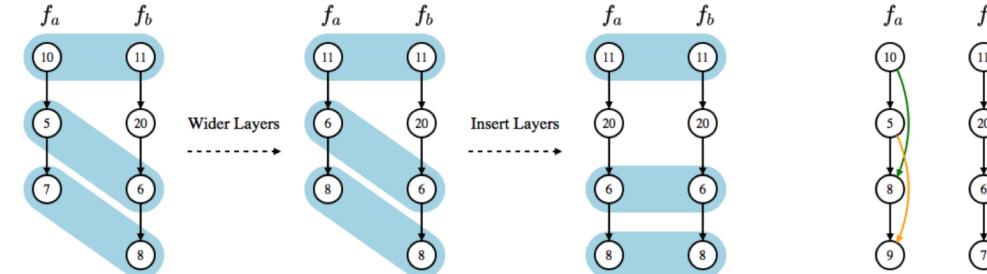


Figure 1: Neural Network Kernel. Given two neural networks f_a , f_b , and matchings between the similar layers, the figure shows how the layers of f_a can be changed to the same as f_b . Similarly, the skip-connections in f_a also need to be changed to the same as f_b according to a given matching.

<https://arxiv.org/abs/1806.10282>

Auto-Keras: An Efficient Neural Architecture Search System

- Расчет D_s .

D_s – это сумма расстояний редактирования между согласованными парами skip-connections двух нейронных сетей.

Как показано на рисунке 1 skip-connections одного цвета – это согласованные пары. Аналогично $D_l(,), D_s(,)$ определяется следующим образом

$$D_s(S_a, S_b) = \min \sum_{i=1}^{|S_a|} d_s(s_a^{(i)}, \varphi_s(s_a^{(i)})) + |S_b| - |S_a|, \quad (8)$$

Где мы полагаем, что $|S_a| < |S_b|$.

($|S_a| - |S_b|$) – измеряет общее расстояние редактирования для несопоставленных skip-connections. Так как каждая из несопоставленных пропущенных связей в S_b требует вставки нового соединения в f_a , функция отображения $\varphi_s : S_a \rightarrow S_b$ – это инъективная функция; $d_s()$ – расстояние редактирования для двух согласованных пропущенных соединений.

$$d_s(s_a, s_b) = \frac{|u(s_a) - u(s_b)| + |\delta(s_a) - \delta(s_b)|}{\max[u(s_a), u(s_b)] + \max[\delta(s_a), \delta(s_b)]}, \quad (9)$$

Где $u(s)$ – топологический ранг слоя начала skip-соединения. $\delta(s)$ – количество слоев между началом и концом skip-соединения.

Проблема минимизации в уравнении (8) схожа с проблемой согласования двудольного графа, где f_a и f_b – это два непересекающихся набора графа, каждое пропускаемое соединение – это узел в своем соответствующем наборе. Расстояние редактирования между двумя пропущенными соединениями – это вес ребра между ними. Весовая задача сопоставления двудольных графов решается с помощью венгерского алгоритма (алгоритм Куна-Мункressа).

Auto-Keras: An Efficient Neural Architecture Search System

- **Проблема №2. Optimization for Tree Structured Space.**
- Второй проблемой использования Байесовской оптимизации для управления преобразованием нейросетей является функция сбора данных (acquisition function). Традиционные функции сбора данных определены в Евклидовом пространстве.
- Подобные методы оптимизации не применимы для древовидного поиска через преобразования сетей.
- Чтобы оптимизировать нашу acquisition function, нам нужен метод оптимизации нашей функции в древовидной структуре (то есть acquisition function исследует древовидное пространство, где узлы – это варианты сетей, унаследованные от узла родителя, а ребра – это последовательность действий для преобразований одной вершину в другую).
- Предлагается новый метод оптимизации *acquisition function*. *Upper-confidence bound (UCB)* для acquisition function выбирается .

$$\alpha(f) = \mu(y_f) - \beta\sigma(y_f), \quad (10)$$

$y_f = Cost(f, D)$, - значение функции потерь архитектуры на датасете.

β – коэффициент баланса,

$\mu(y_f)$ и $\sigma(y_f)$ – апостериорные средние и отклонение от y_f .

Новый метод имеет 2 важных свойства:

Есть явный коэффициент β – между разведкой и эксплуатацией.

$\alpha(f)$ – напрямую сравнивается со значением функции потерь $c^{(i)}$ в истории поиска

Это позволяет оценить минимальную возможную потерю (cost), данную нейронной $\mathcal{H} = \{(f^{(i)}, \theta^{(i)}, c^{(i)})\}$.

$\hat{f} = argmin_f \alpha(f)$ - это сгенерированная нейронная архитектура для наблюдения.

Древовидный поиск определяется следующим образом. В ходе оптимизации $\alpha(f)$, f следует получать из $f^{(i)}$ и O . Где f - наблюдаемая архитектура в истории поиска, O – последовательность операций для преобразования архитектуры в новую.

Преобразование f в \hat{f} определяется как $\hat{f} \leftarrow M(f, O)$, $M(f, O)$ – функция преобразования f с помощью O .

<https://arxiv.org/abs/1806.10282>

Auto-Keras: An Efficient Neural Architecture Search System

- **Проблема №2. Optimization for Tree Structured Space.**

- Самый большой дефект преобразования сети – это увеличение размеров архитектуры, вместо ее уменьшения. Использование преобразования сети в NAS может привести к очень большой архитектуре без достаточного изучения малых архитектур. Однако, при поиске в древовидном пространстве архитектур, мы расширяем не только конечные листья, но и внутренние узлы, что означает, что меньшие архитектуры, найденные на ранней стадии могут быть выбраны несколько раз, чтобы перейти к более сравнительно небольшим архитектурам.

Algorithm 1 Optimize Acquisition Function

```
1: Input:  $\mathcal{H}, r, T_{low}$ 
2:  $T \leftarrow 1, Q \leftarrow PriorityQueue()$ 
3:  $c_{min} \leftarrow \text{lowest } c \text{ in } \mathcal{H}$ 
4: for  $(f, \theta_f, c) \in \mathcal{H}$  do
5:    $Q.Push(f)$ 
6: end for
7: while  $Q \neq \emptyset$  and  $T > T_{low}$  do
8:    $T \leftarrow T \times r, f \leftarrow Q.Pop()$ 
9:   for  $o \in \Omega(f)$  do
10:     $f' \leftarrow M(f, \{o\})$ 
11:    if  $e^{\frac{c_{min}-\alpha(f')}{T}} > Rand()$  then
12:       $Q.Push(f')$ 
13:    end if
14:    if  $c_{min} > \alpha(f')$  then
15:       $c_{min} \leftarrow \alpha(f'), f_{min} \leftarrow f'$ 
16:    end if
17:  end for
18: end while
19: Return The nearest ancestor of  $f_{min}$  in  $\mathcal{H}$ , the operation sequence to reach  $f_{min}$ 
```

- предлагается новый метод, основанный на поиске A* и имитируемом отжиге. A* широко используется для поиска в древовидной структуре. Он поддерживает приоритетную очередь узлов и продолжает расширять лучший узел в очереди. Поскольку A* всегда использует лучший узел, вводится моделируемый отжиг, чтобы сбалансировать разведку и эксплуатацию, не выбирая предполагаемую лучшую архитектуру с вероятностью.



www.huawei.com

Copyright © Huawei Technologies Co., Ltd. 2020. All rights reserved.

All logos and images displayed in this document are the sole property of their respective copyright holders. No endorsement, partnership, or affiliation is suggested or implied. The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.