AutoML

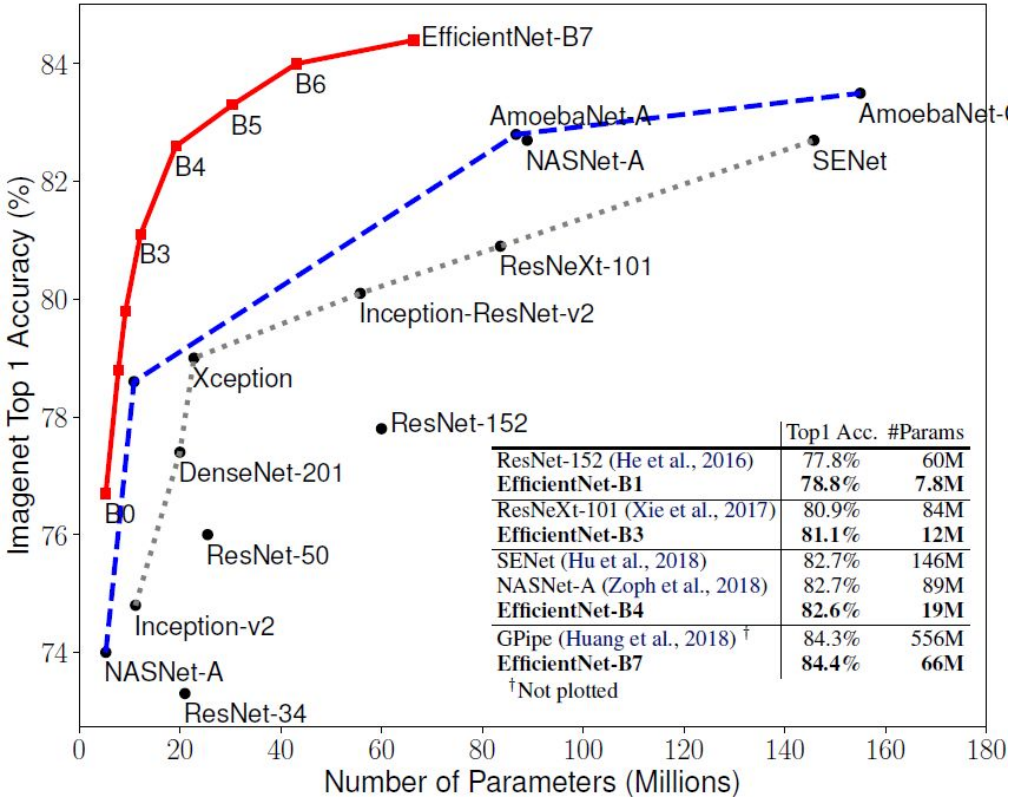
EfficientNet

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan [1] Quoc V. Le [1]

EfficientNet

На момент исследования EfficientNets превзошли другие CNN сети. В частности, EfficientNet-B7 с точностью 84,4% в 8 раза меньше и в 6,1 раз быстрее чем GPipe. EfficientNet-B1 в 7,6 раз меньше и в 5,7 раза быстрее, чем ResNet-152.

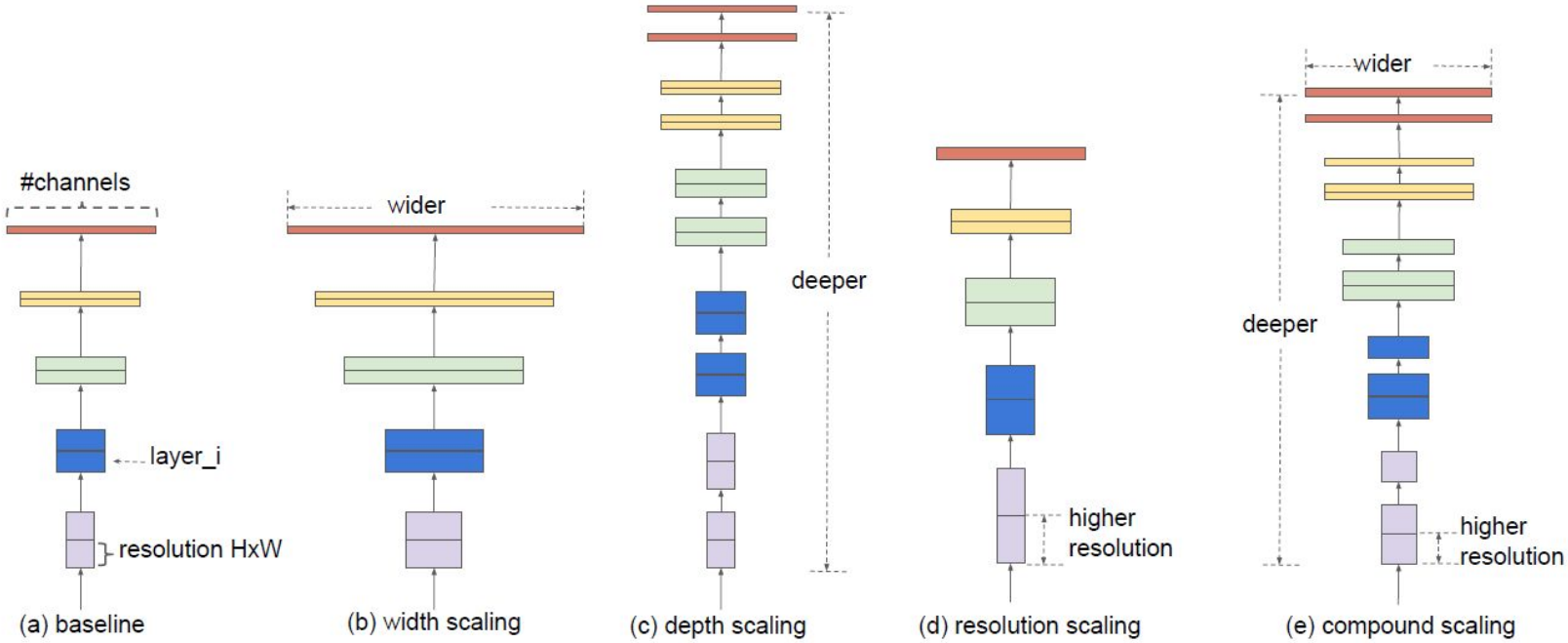


	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

EfficientNet

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks



Масштабирование модели: (a) базовая сетка; (b)-(d) обычное масштабирование, которое увеличивает только одно измерение ширины, глубины или разрешения сети. (e) — метод комплексного масштабирования равномерно масштабирует все три измерения с фиксированным соотношением.

EfficientNet

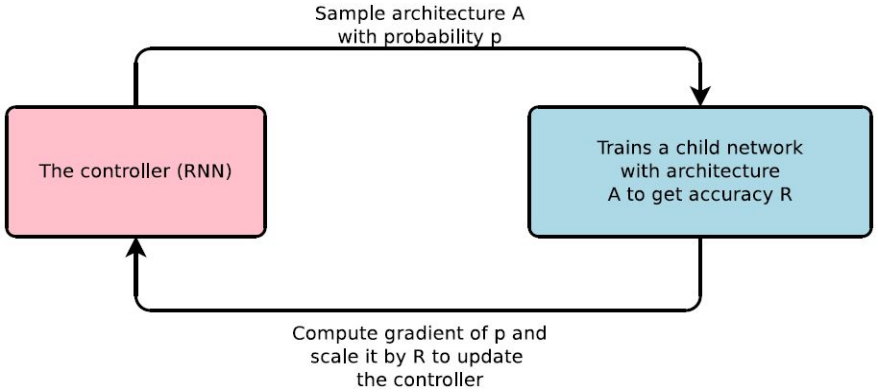
Предложенный метод составного масштабирования использует коэффициент ϕ для равномерного масштабирования ширины, глубины и разрешения сети:

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

где α, ß, ɣ являются константами, которые можно определить поиском по небольшой сетке. Коэффициент ϕ определяется доступными вычислительными ресурсами для масштабирования модели, в то время как α, ß, ɣ определяют, как назначить эти ресурсы ширине, глубине и разрешению сети соответственно. Примечательно, что FLOPS обычной операции свертки пропорциональны d, w^2, r^2, т.е. удвоение глубины сети приведет к удвоению FLOPS, но удвоение ширины или разрешения сети увеличит FLOPS в четыре раза.

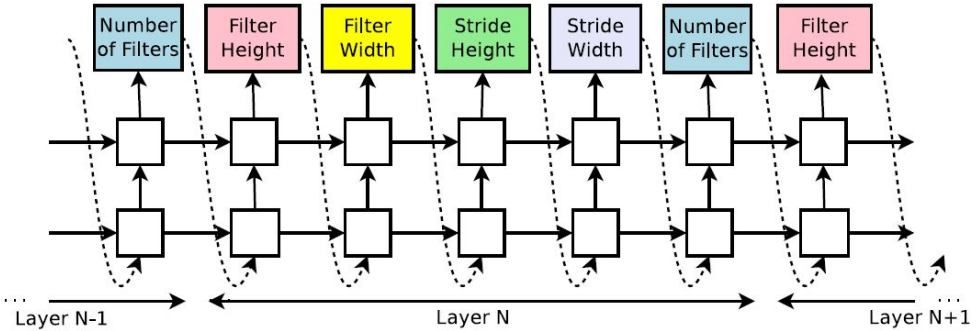
Поиск архитектуры NN с помощью RL

Neural Architecture Search with Reinforcement Learning, Barret Zoph, Quoc V. Le, 2017



$$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R]$$

$$\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}\left[\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R\right]$$



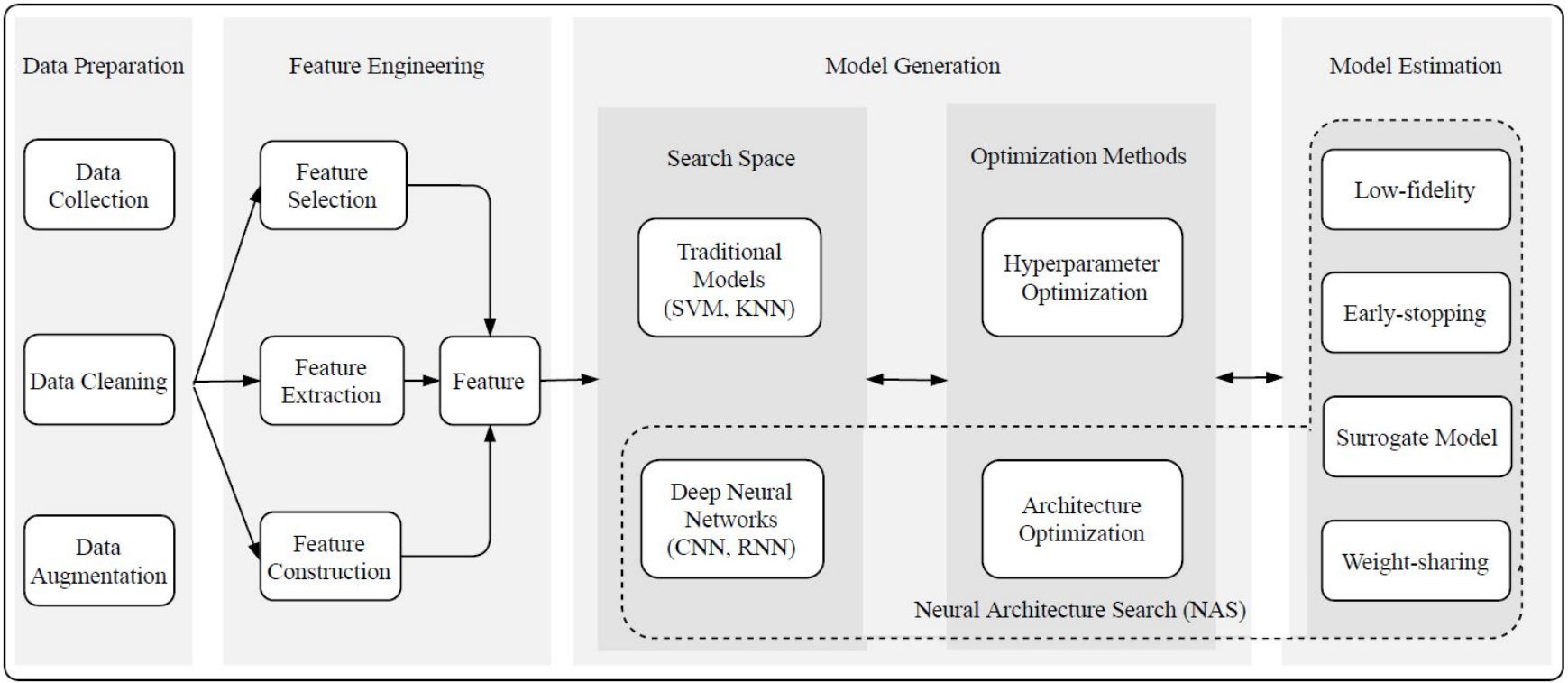
Обзор AutoML

AutoML: A Survey of the State-of-the-Art

Xin He, Kaiyong Zhao, Xiaowen Chu*

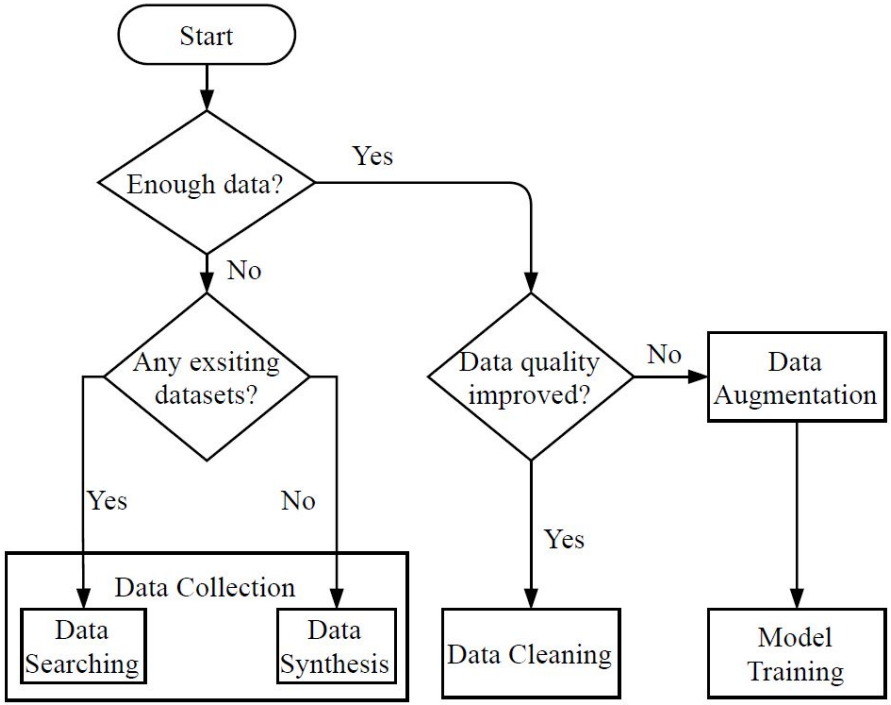
Department of Computer Science, Hong Kong Baptist University

Обзор AutoML



AutoML включает подготовку данных, формирование признаков, создание и оценку модели.

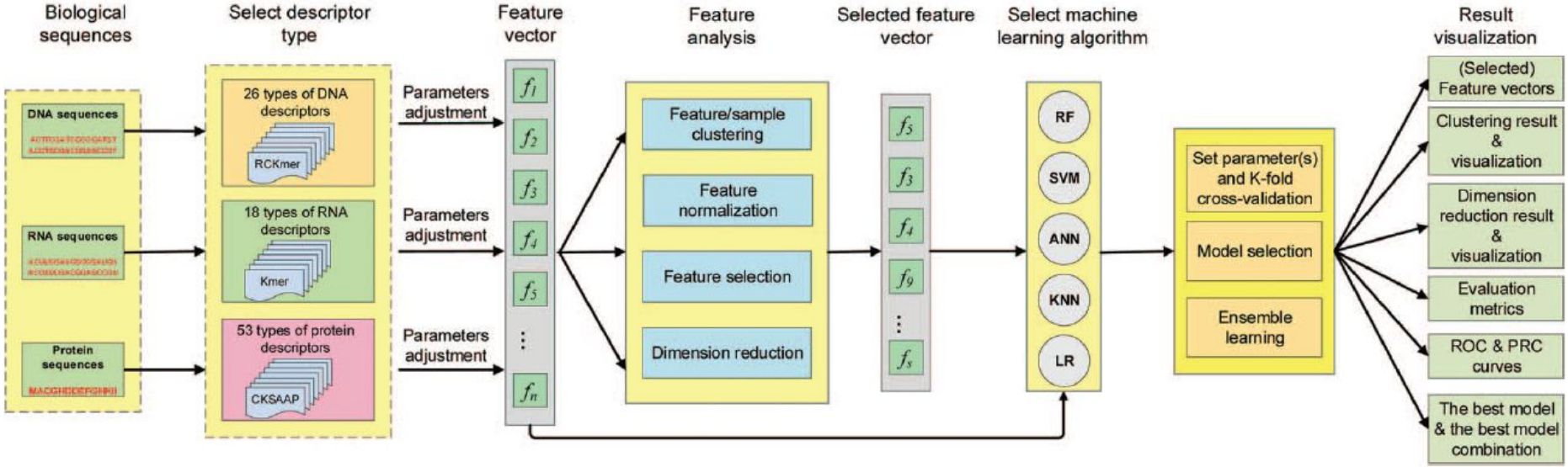
AutoML: Подготовка данных (Data Preparation)



Data Synthesis

Один из наиболее часто используемых методов генерации данных. Для ряда задач, таких как автономное вождение, невозможно протестировать и настроить модель в реальном мире на этапе исследования из-за угроз безопасности. Поэтому практический подход к генерации данных заключается в использовании симулятора данных, который соответствует реальному миру.
Еще одним новым методом получения синтетических данных являются генеративно-состязательные сети (GAN), которые можно использовать для генерации изображений, табличных и текстовых данных. Боулз продемонстрировал возможность использования GAN для создания медицинских изображений для задач сегментации мозга. В случае текстовых данных применение GAN к тексту оказалось трудным, поскольку обычный подход заключается в использовании обучения с подкреплением для обновления градиента генератора, но текст дискретен, и, следовательно, градиент не может распространяться от дискриминатора к генератору. Чтобы решить эту проблему, Донахью использовал автокодировщик для кодирования предложений в плавное представление предложений, чтобы устранить барьер обучения с подкреплением. Парк применил GAN для синтеза таблиц, которые статистически похожи на исходную таблицу, но не вызывают утечки информации. Аналогичным образом GAN применяется для создания табличных данных, таких как медицинские или образовательные записи.

Формирование признаков (Feature Engineering): iLearn



Блок-схема вычислительных методов на основе алгоритмов машинного обучения для анализа биологических последовательностей

Дескрипторы iLearn

Descriptor groups	Descriptor	Dimension	DNA or RNA
Nucleic acid composition	Nucleic acid composition (NAC)	4	DNA/RNA
	Enhanced nucleic acid composition (ENAC)	–	DNA/ RNA
	Di-nucleotide composition (DNC)	16	DNA/RNA
	Tri-nucleotide composition (TNC)	64	DNA/RNA
	k-spaced nucleic acid pairs (CKSNNP)	$(k+1)$x16	DNA/RNA
	Basic kmer (Kmer)	–	DNA/RNA
	Reverse compliment kmer (RCKmer)	–	DNA/RNA
	Accumulated nucleotide frequency (ANF)	–	DNA/RNA
	Nucleotide chemical property (NCP)	–	DNA/RNA
Binary	Binary (binary)	–	DNA/RNA
Position-specific tendencies of trinucleotide	Position-specific trinucleotide propensity based on single-strand (PSTNPss)		DNA/RNA
	Position-specific trinucleotide propensity based on single-strand (PSTNPds)		DNA
Electron-ion interaction pseudopotentials	Electron-ion interaction pseudopotentials value (EIIP)		DNA
	Electron-ion interaction pseudopotentials of trinucleotide (PseEIIP)		DNA
Autocorrelation and cross-covariance	Dinucleotide-based auto covariance (DAC)		DNA/RNA
	Dinucleotide-based cross covariance (DCC)		DNA/RNA
	Dinucleotide-based auto-cross covariance (DACC)		DNA/RNA
	Trinucleotide-based auto covariance (TAC)		DNA
	Trinucleotide-based cross covariance (TCC)		DNA
	Trinucleotide-based auto-cross covariance (TACC)		DNA
Pseudo nucleic acid composition	Pseudo dinucleotide composition (PseDNC)		DNA/RNA
	Pseudo k-tupler composition (PseKNC)		DNA/RNA
	Parallel correlation pseudo dinucleotide composition (PCPseDNC)		DNA/RNA
	Parallel correlation pseudo trinucleotide composition (PCPseTNC)		DNA
	Series correlation pseudo dinucleotide composition (SCPseDNC)		DNA/RNA
	Series correlation pseudo trinucleotide composition (SCPseTNC)		DNA

Дескрипторы BioAutoML, часть 1

Descriptor groups	Descriptor	Dimension	Biological Sequence
Other descriptors	Basic k-mer	4^k or 20^k	DNA/RNA/Protein
	Customized k-mer	4^k or 20^k	DNA/RNA/Protein
	NAC	4	DNA/RNA
	DNC	16	DNA/RNA
	TNC	64	DNA/RNA
	ORF Features or Coding Features	10	DNA/RNA
	Fickett score	2	DNA/RNA
	PseKNC	-	DNA/RNA
	ANF	L	DNA/RNA/Protein
	kGap	$4^X \cdot 4^Y$ or $20^X \cdot 20^Y$	DNA/RNA/Protein
	AAC	20	Protein
	DPC	400	Protein
	TPC	8000	Protein

Обычные дескрипторы, рассчитанные с помощью MathFeature для последовательностей ДНК, РНК и белков.

Дескрипторы BioAutoML, часть 2

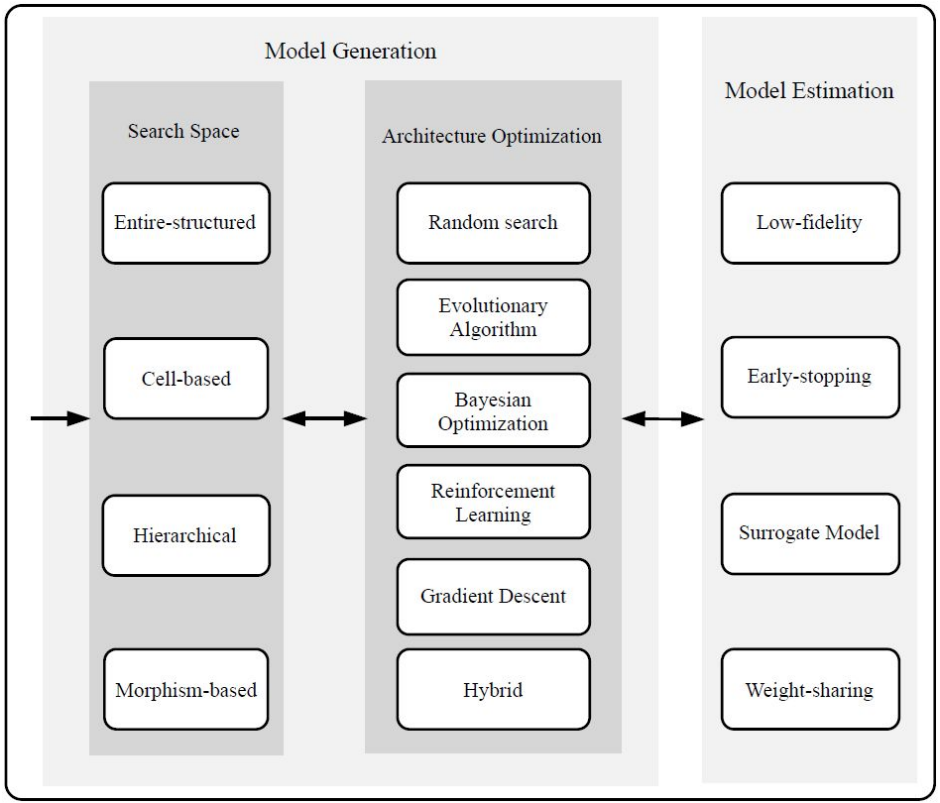
Descriptor groups	Descriptor	Dimension	Biological Sequence
Numerical Mapping	Binary	$L \cdot 4$	DNA/RNA
	Z-curve	$L \cdot 3$	DNA/RNA
	Real	L	DNA/RNA
	Integer	L	DNA/RNA/Protein
	EIIP	L	DNA/RNA/Protein
	Complex Number	L	DNA/RNA
	Atomic Number	L	DNA/RNA
Fourier Transform	Binary + Fourier	19	DNA/RNA
	Z-curve + Fourier	19	DNA/RNA
	Real + Fourier	19	DNA/RNA
	Integer + Fourier	19	DNA/RNA/Protein
	EIIP + Fourier	19	DNA/RNA/Protein
	Complex Number + Fourier	19	DNA/RNA
	Atomic Number + Fourier	19	DNA/RNA
Chaos Game	Chaos Game Representation	$L \cdot 2$	DNA/RNA
	Chaos Game Signal (with Fourier)	19	DNA/RNA
Entropy	Shannon	k	DNA/RNA/Protein
	Tsallis	k	DNA/RNA/Protein
Graphs	Complex Networks (with threshold)	$12 \cdot t$	DNA/RNA/Protein
	Complex Networks (without threshold)	$26 \cdot k$	DNA/RNA/Protein

Математические дескрипторы, рассчитанные с помощью Math Feature для последовательностей ДНК, РНК и белков.

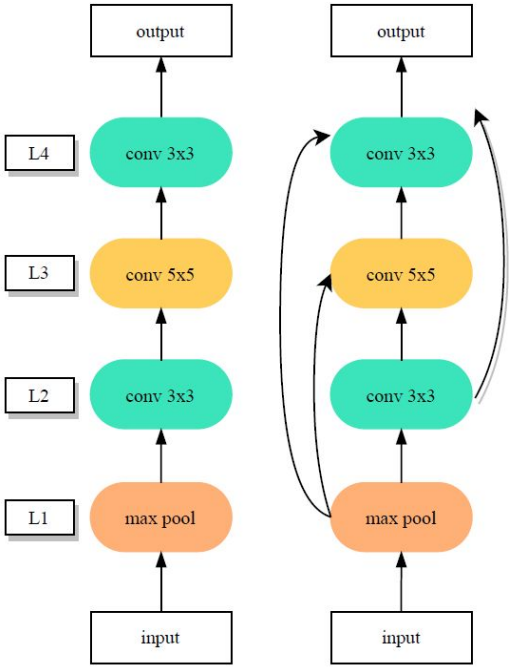
Сравнение количества дескрипторов

Package	Mathematical Descriptors	Conventional Descriptors	Number of Descriptors Calculated
MathFeature	20	17	37
PROFEAT	0	2	2
PseAAC	0	2	2
propy	0	5	5
PseKNC-General	0	5	5
SPiCE	0	4	4
ProtrWeb	0	5	5
ProFET	2	3	5
Pse-in-One	0	5	5
repDNA	0	5	5
Rcpi	0	3	3
repRNA	0	5	5
BioSeq-Analysis	0	9	9
iFeature	1	4	5
PyBioMed	0	7	7
Seq2Feature	0	0	0
PyFeat	1	8	9
iLearn	2	13	15

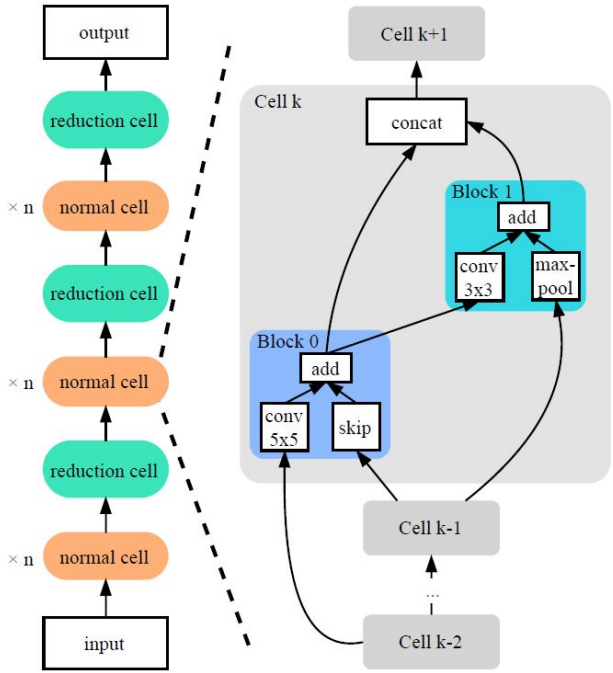
AutoML: Построение модели



Entire-structured и Cell-based Search Space



Два упрощенных примера цельноструктурированных архитектур. Каждый слой определяется отдельной операцией, такой как свертка и max pool. Операция skip-connection в примере справа, позволяет строить более глубокие и сложные архитектуры..



(Слева) Пример cell-based модели из 3-х мотивов (motifs), каждый содержит n нормальных клеток и одну редукционную. (Справа) Пример нормальной ячейки, которая содержит два блока по два узла в каждом. Узел задается операцией и входными данными.

Оценка размера Search Space

Размер Cell-based Search Space меньше чем Entire-structured. Чтобы проиллюстрировать это, предположим, что существует M предопределенных операций-кандидатов. Количество слоев как для целых структур, так и для структур на основе ячеек равно L, а количество блоков в ячейке равно B. Тогда количество возможных целых структур может быть выражено как:

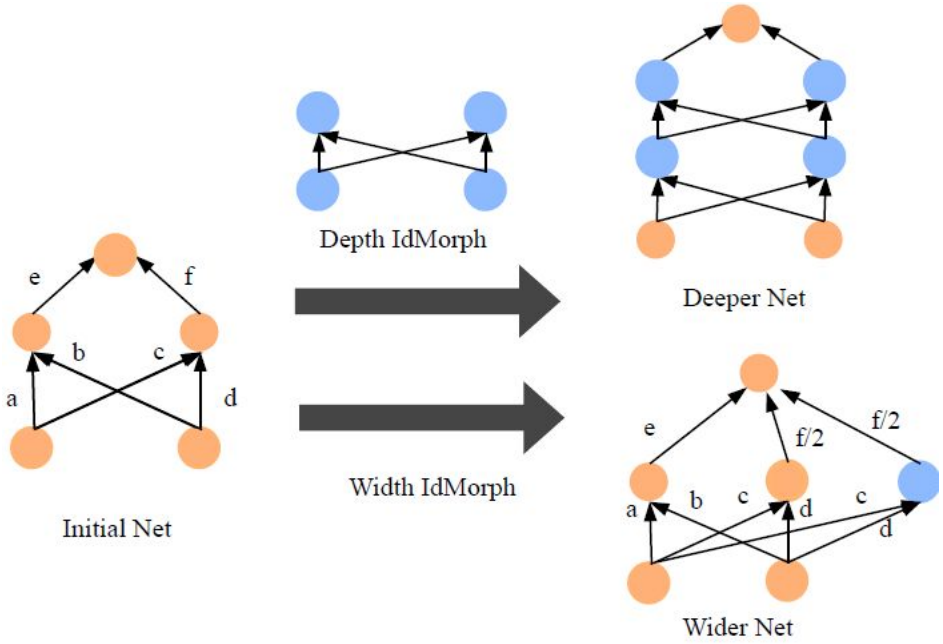
Nentire = M**L * 2**(L*(L-1))/2)

Число возможных ячеек равно (M**B * (B +2)!)**2. Однако, поскольку существует два типа ячеек (т. е. нормальные и уменьшенные ячейки), окончательный размер пространства поиска на основе ячеек рассчитывается как:

Ncell = (M**B * (B + 2)!)**4

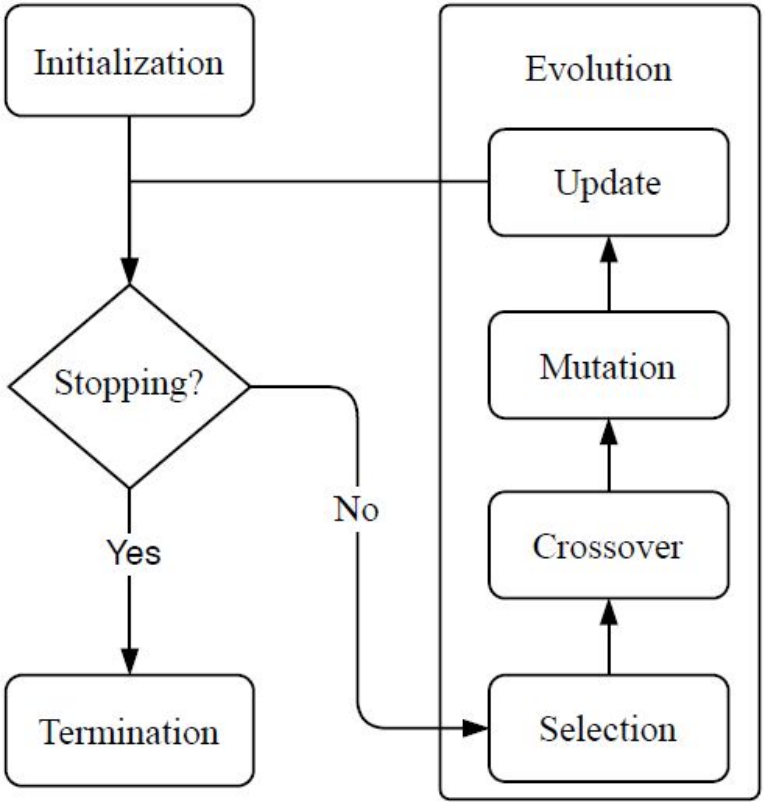
Очевидно, что сложность поиска всей структуры растет экспоненциально с увеличением количества слоев. Для интуитивного сравнения мы возьмем значения M = 5, L = 10, B = 3, тогда **Nentire = 3.44 * 10^20** намного больше, чем **Ncell = 5.06 * 10^16**

Morphism-based Search Space

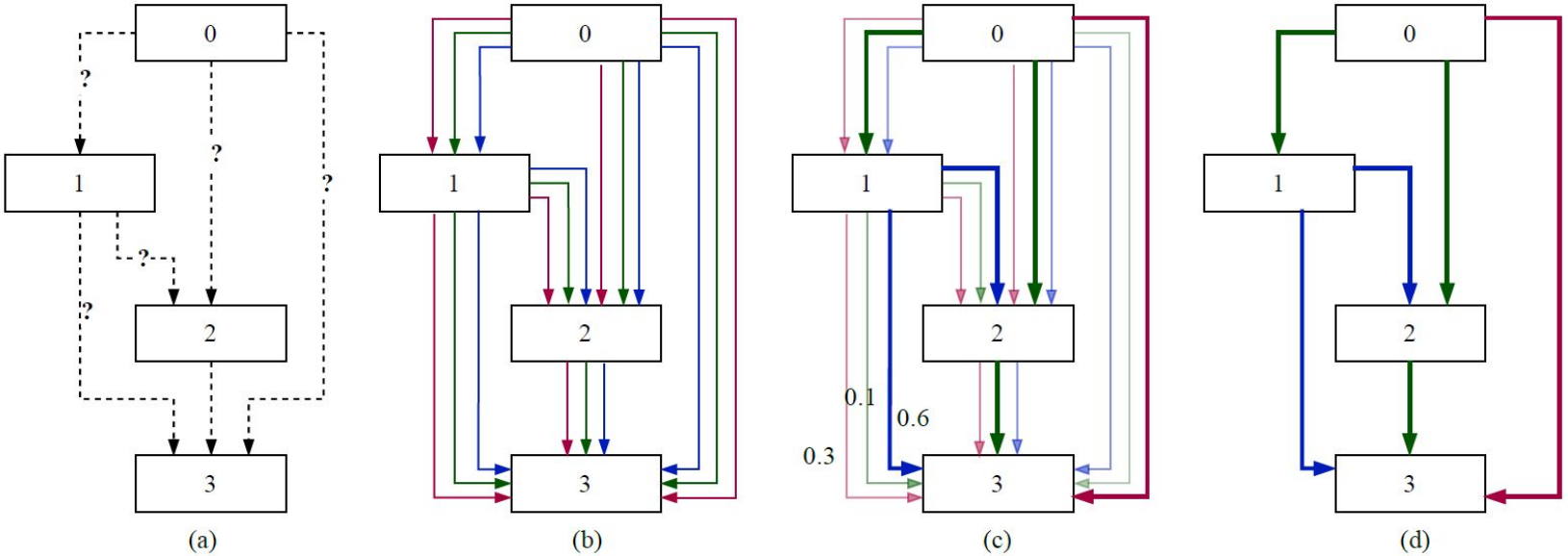


IdMorph - операция тождественного морфизма. Значение на каждом ребре - параметр модели (вес)

Оптимизация архитектуры: Эволюционный алгоритм



Оптимизация архитектуры: градиентный спуск



(a) Данные могут передаваться только от узлов более низкого уровня к узлам более высокого уровня, а операции на ребрах изначально неизвестны. (б) Начальная операция на каждом ребре представляет собой смесь операций-кандидатов, каждая из которых имеет одинаковый вес. (c) Веса операции тренируются и находятся в диапазоне от 0 до 1. (d) Окончательная архитектура строится путем сохранения операции с максимальным весовым значением на каждом ребре

Оптимизация архитектуры: градиентный спуск

$$\overline{o}_{i,j}(x) = \sum_{k=1}^{K} \frac{\exp\left(\alpha_{i,j}^{k}\right)}{\sum_{l=1}^{K} \exp\left(\alpha_{i,j}^{l}\right)} o^{k}(x)$$

где o(x) указывает операцию, выполненную на входном векторе x, $\alpha^{k}_{i,k}$ указывает вес, операции o^{k} между парой узлов (i; j), а K — количество заранее определенных операций-кандидатов. В таком виде задача поиска архитектур трансформируется в совместную оптимизацию α архитектуры и весов этой архитектуры θ. Эти два типа параметров оптимизируются поочередно, что указывает на проблему двухуровневой оптимизации. В частности, α и θ оптимизируются с помощью validation и training наборов соответственно.

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}\left(\theta^{*}, \alpha\right) \\ \text{s.t.} \quad & \theta^{*} = \operatorname{argmin}_{\theta} \mathcal{L}_{train}(\theta, \alpha) \end{aligned}$$

$$\min_{\alpha,\theta} \left[\mathcal{L}_{train}\left(\theta^{*}, \alpha\right) + \lambda \mathcal{L}_{val}\left(\theta^{*}, \alpha\right)\right]$$

Оптимизация архитектуры: SMBO

Algorithm Framework 1: Sequential Model-Based Optimization (SMBO)

$\mathbf{R}$ keeps track of all target algorithm runs performed so far and their performances (*i.e.*, SMBO's training data $\{([\boldsymbol{\theta}_1, \boldsymbol{x}_1], o_1), \ldots, ([\boldsymbol{\theta}_n, \boldsymbol{x}_n], o_n)\}$), $\mathcal{M}$ is SMBO's model, $\vec{\boldsymbol{\Theta}}_{new}$ is a list of promising configurations, and t_{fit} and t_{select} are the runtimes required to fit the model and select configurations, respectively.

Input : Target algorithm A with parameter configuration space $\boldsymbol{\Theta}$; instance set Π; cost metric $\hat{c}$

Output : Optimized (incumbent) parameter configuration, $\boldsymbol{\theta}_{inc}$

$[\mathbf{R}, \boldsymbol{\theta}_{inc}] \leftarrow \textit{Initialize}(\boldsymbol{\Theta}, \Pi)$

repeat

  $[\mathcal{M}, t_{fit}] \leftarrow \textit{FitModel}(\mathbf{R})$

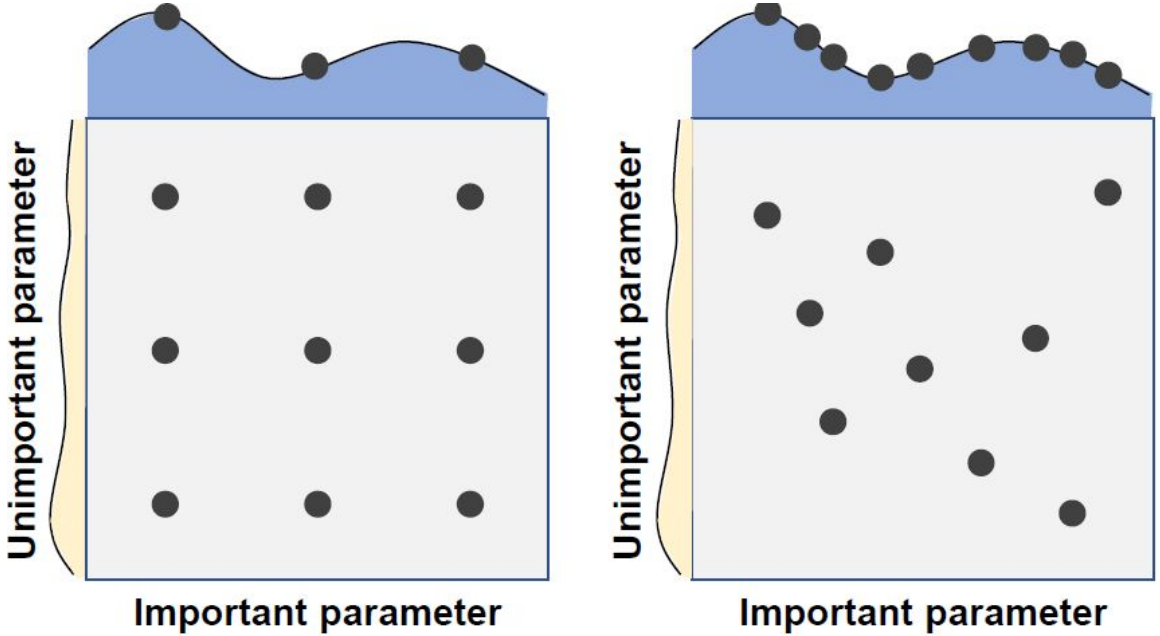
  $[\vec{\boldsymbol{\Theta}}_{new}, t_{select}] \leftarrow \textit{SelectConfigurations}(\mathcal{M}, \boldsymbol{\theta}_{inc}, \boldsymbol{\Theta})$

  $[\mathbf{R}, \boldsymbol{\theta}_{inc}] \leftarrow \textit{Intensify}(\vec{\boldsymbol{\Theta}}_{new}, \boldsymbol{\theta}_{inc}, \mathcal{M}, \mathbf{R}, t_{fit} + t_{select}, \Pi, \hat{c})$

until *total time budget for configuration exhausted*

return $\boldsymbol{\theta}_{inc}$

Оптимизация архитектуры: Random Search



Примеры поиска по сетке (слева) и случайного поиска (справа) в девяти испытаниях по оптимизации двумерной пространственной функции $f(x,y) = g(x) + h(y) \approx g(x)$. Параметр в $g(x)$ (голубая часть) относительно важен, а параметр в $h(y)$ (светло-желтая часть) не важен. При поиске по сетке девять испытаний охватывают только три важных значения параметра; однако случайный поиск может исследовать девять различных значений g. Следовательно, случайный поиск с большей вероятностью позволит найти оптимальную комбинацию параметров, чем поиск по сетке.

Model Evaluation

- … SMASH использует вспомогательную гиперсеть для генерации весов для случайно выбранных архитектур. Аналогичным образом, Чжан и др. предложил представление вычислительного графа и использовал гиперсеть графов (GHN) для более быстрого и точного прогнозирования весов для всех возможных архитектур, чем обычные гиперсети. Однако благодаря тщательному экспериментальному анализу, проведенному для понимания механизма стратегии распределения веса, Бендер и др. показал, что гиперсеть не требуется для поиска оптимальной архитектуры. Они предложили dropout для путей, чтобы облегчить проблему весовой связи. Во время обучения суперсети каждый путь суперсети отбрасывается случайным образом с постепенно увеличивающейся вероятностью….
- … Сеть шаблонов с самооценкой (SETN) предлагает оценщик для прогнозирования вероятности того, что каждая архитектура будет иметь меньшие потери при проверке. **Результаты экспериментов показывают, что SETN потенциально может найти архитектуру с более высокой производительностью, чем методы на основе RS**.

NAS-Bench-101, NAS-Bench-201

AutoML в диагностике заболеваний

Laukamp, K. R. *et al.* Fully automated detection and segmentation of meningiomas using deep learning on routine multiparametric MRI. *Eur. Radiol.* 29, 124–132 (2019).

A machine learning model for detecting invasive ductal carcinoma with Google Cloud AutoML Visio, Yan Zeng, Jinmiao Zhang, Computers in biology and medicine 122, 103861, 2020

Zhang, H. et al. Deep learning model for the automated detection and histopathological prediction of meningioma. Neuroinformatics 19, 393–402 (2021).

Papoutsoglou, G. *et al.* Automated machine learning optimizes and accelerates predictive modeling from COVID-19 high throughput datasets. *Sci. Rep.* 11, 15107 (2021).

Ikemura, K. *et al.* using automated machine learning to predict the mortality of patients with COVID-19: Prediction model development study. *J. Med. Internet Res.* 23, e23458 (2021).

Karaglani, M., Gourlia, K., Tsamardinos, I. & Chatzaki, E. Accurate blood-based diagnostic biosignatures for alzheimer's disease via Automated Machine Learning. *J. Clin. Med.* 9, E3016 (2020).

Ou, C. *et al.* Automated Machine Learning model development for intracranial aneurysm treatment outcome prediction: A feasibility study. *Front. Neurol.* 12, 735142 (2021).

Touma, S., Antaki, F. & Duval, R. Development of a code-free machine learning model for the classification of cataract surgery phases. *Sci. Rep.* 12, 2398 (2022).

AutoML от Сбера: LAMA

LightAutoML (LAMA) – мощный open-source AutoML фреймворк за которым стоит одна из сильнейших по экспертизе DS команд из Sber AI Lab. Суперсила LAMA – это бленды и настраиваемые эксперименты. В то же время LAMA скорее скальпель для профессионалов,. Давно не было обновления, надеюсь, увидим его в ближайшее время.