Поиск нейронных архитектур(NAS)

Дмитрий Осин @xaosina

>>> Содержание

Содержание:

Эволюция машинного обучения:

1. Аналитик генерирует новые признаки и обрабатывает данные, запускает классическую модель **по своему усмотрению**
2. Аналитик генерирует новые признаки и обрабатывает данные, запускает классическую модель **с перебором гиперпараметров**
3. Аналитик обрабатывает данные, запускает **нейронную сеть** по своему усмотрению
4. Аналитик обрабатывает данные, запускает нейронную сеть **с поиском архитектуры**

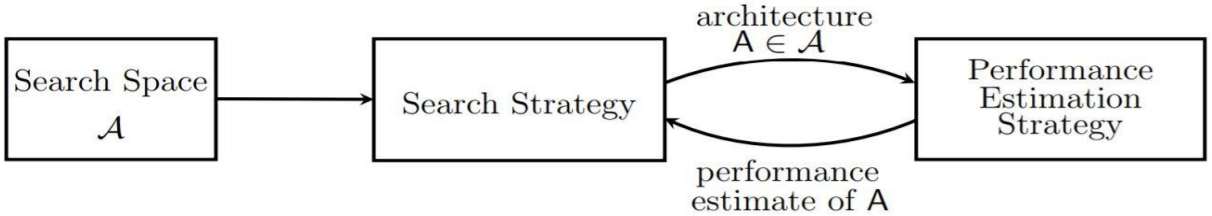
Table 1
Overview of NAS approaches, their performance, and the general methodology employed: Evolutionary Algorithms (EA), Reinforcement Learning (RL), Gradient-Based (GB), Weight-Sharing (WS) and Prediction (Pred). † entails object detection. ‡ entails instance segmentation. Otherwise, reported results are for Image Classification. Results correspond to the results reported in their respective original paper, even when subsequent papers report higher performance or results generated using more comparable computational resources.

Reference	Technique							Top-1 Acc (CIFAR-10) (%)	Params (CIFAR-10)	Top-1 Acc (ImageNet) (%)	Params (ImageNet)
	Evolutionary algorithms	Reinforcement Learning	Gradient based	Prediction	Weight sharing	Sample based	One shot				
Zoph & Le [16] (2017)		✓				✓		96.35	37.4M	n/a	n/a
MnasNet [19] (2019)		✓				✓		n/a	n/a	76.7M	5.2M
ENAS [30] (2018)		✓			✓		✓	97.11	4.6M	n/a	n/a
SNAS [31] (2018)		✓			✓		✓	97.02	2.9M	72.7	4.3M
CAS [38] (2019)		✓			✓		✓	n/a	n/a	n/a	n/a
CNAS [32] (2020)		✓			✓		✓	97.40	3.7M	75.40	5.3M
BigNAS-S [39] (2020)					✓		✓	n/a	n/a	76.5	4.5M
BigNAS-M [39] (2020)					✓		✓	n/a	n/a	78.9	5.5M
BigNAS-L [39] (2020)					✓		✓	n/a	n/a	79.5	6.4M
BigNAS-XL [39] (2020)					✓		✓	n/a	n/a	80.9	9.5M
ProxylessNAS-R [40] (2018)		✓			✓		✓	97.70	5.8M	74.6	n/a
ProxylessNAS-G [40] (2018)			✓		✓		✓	97.92	5.7M	74.2	n/a
NASP [41] (2020)			✓		✓		✓	97.56	7.4M	73.7	9.5M
TuNAS [15] (2020)		✓			✓		✓	n/a	n/a	75.0	n/a
AttentiveNAS (largest) [42] (2021)					✓		✓	n/a	n/a	80.7	n/a
RNSGA-Net1 [43] (2022)	✓							96.11	3.01	n/a	n/a
Stage-Wise NAS [44] (2020)					✓	✓		95.68	7.27M	n/a	n/a
PAD-NAS [45] (2022)	✓				✓		✓	n/a	n/a	76.1	4.7M
GLiT-Tiny [46] (2021)	✓				✓		✓	n/a	n/a	76.3	7.2M
GLiT-Small [46] (2021)	✓				✓		✓	n/a	n/a	80.5	24.6M
GLiT-Base [46] (2021)	✓				✓		✓	n/a	n/a	82.3	96.1M
NEAS [47] (2021)	✓				✓		✓	n/a	n/a	80.0	n/a
BONAS [48] (2020)	✓				✓			97.57	3.3M	74.6	4.8M
DARTS [26] (2019)			✓		✓		✓	97.24	3.3M	73.3	4.7M
I-DARTS [49] (2019)			✓		✓		✓	97.63	3.8M	75.7	n/a
Wu et al. [50] (2020)			✓		✓		✓	97.50	3.5M	75.33	5.7M
E-DNAS [51] (2020)			✓		✓		✓	n/a	n/a	76.9	5.9M
ISTA-NAS [52] (2020)			✓		✓		✓	97.64	3.37M	76.0	5.65M
BMTAS [53] (2020)			✓		✓		✓	n/a	n/a	n/a	n/a
SMASH [54] (2018)			✓		✓		✓	94.47	4.6M	61.38	16.2M
unsupervised: DARTS [55] (2020)			✓		✓		✓	97.44	3.6M	n/a	n/a
FairNAS [56] (2021)	✓				✓		✓	98.2	n/a	77.5	5.9M
PI-NAS [57] (2021)			✓		✓		✓	n/a	n/a	81.60	27.1M
EnTranNAS [58] (2021)			✓		✓		✓	97.78	7.68M	75.70	7.2M
EnTranNAS-DST [58] (2021)			✓		✓		✓	97.52	3.20M	76.20	7.0M
DOTS [59] (2021)			✓		✓		✓	97.51	3.5M	76.0	5.3M
Shapley-NAS [2] (2022)					✓		✓	97.57	3.6M	76.1	5.4M
P-DARTS [27] (2019)			✓		✓		✓	97.50	3.5M	75.9	5.4M
PC-DARTS [60] (2019)			✓		✓		✓	97.43	3.6M	75.8	5.3M
FP-DARTS [61] (2023)			✓		✓		✓	97.50	3.4M	76.3	5.3M
R-DARTS(ℓ2) [62] (2020)			✓		✓		✓	97.49	n/a	n/a	n/a
Landmark Regularization:NAO [69] (2021)			✓		✓		✓	n/a	n/a	68.89	4.49M
TAS [70] (2019)			✓		✓		✓	94.00	n/a	76.20	n/a
NetAdaptV2 [71] (2021)			✓		✓		✓	n/a	n/a	77.0	n/a
FBNet-C [72] (2019)			✓		✓		✓	n/a	n/a	74.9	5.5M
PNAS [73] (2019)				✓				96.59	3.2M	74.2	5.1M
PNAS-Large [73] (2019)				✓				n/a	n/a	82.9	86.1M
NAO [74] (2018)			✓				✓	96.82	10.6M	74.3	11.35M
NAO with pseudo morphological operations [75] (2022)			✓				✓	97.35	n/a	n/a	n/a
GBDT-NAS [76] (2020)				✓				n/a	n/a	76.6	5.7M
ReNAS [77] (2021)				✓				n/a	n/a	n/a	n/a
MdeNAS [28] (2019)				✓				97.45	3.61M	74.5	6.1M
NASWOT [29] (2021)				✓				n/a	n/a	n/a	n/a
NASBOT [78] (2018)				✓				91.31	n/a	n/a	n/a
Auto-Keras [79] (2019)				✓				96.40	n/a	n/a	n/a
BayesNAS [80] (2019)				✓				97.59	3.4M	73.5	3.9M
BANANAS [81] (2019)				✓				n/a	n/a	n/a	n/a
MnasNet [19] (2019) †		✓						23.0		4.9M	
DetNAS [82] (2019) †	✓				✓		✓	42.0		n/a	
SpineNET-49S [83] (2020) †		✓				✓		41.5		12M	
SpineNET-190 [83] (2020) †		✓				✓		52.1		163.6M	
NATS [84] (2019) †			✓		✓		✓	38.4		n/a	
NAS-FPN (AmoebaNet Backbone) [85] (2019) †		✓				✓		48.4		166.5M	
Auto-FPN [86] (2019) †			✓		✓		✓	44.3		n/a	
NAS-FCOS [87] (2020) †		✓				✓		46.1		89.4M	
OPANAS [88] (2021) †	✓				✓		✓	41.6		29.8M	
								mIOU (cityscapes)			
DPC [89] (2018) ‡								82.7			
Auto-DeepLab [90] (2019) ‡			✓		✓		✓	82.1			
DCNAS [91] (2021) ‡			✓		✓		✓	84.3			
EDNAS [51] (2020) ‡	✓							n/a			

Источник

>>> Общий подход

Общий подход к NAS



- Представление DAG
- Блоки операций
- Мета-архитектура

- Обучение с подкреплением
- Дифференцируемый поиск
- Эволюционный поиск

- Полное обучение
- Частичное обучение
- Разделяемые веса

>>> NAS с использованием RL

Neural architecture search with reinforcement learning.

1. Модели строятся послойно при помощи - Reinforcement Learning
2. В качестве награды - качество на валидации после обучения с нуля

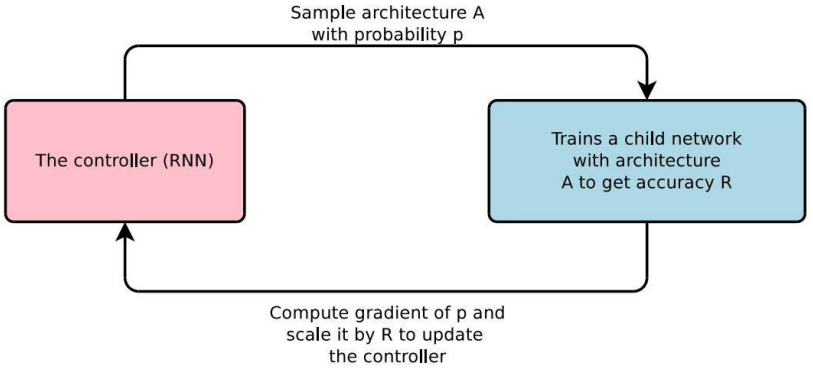
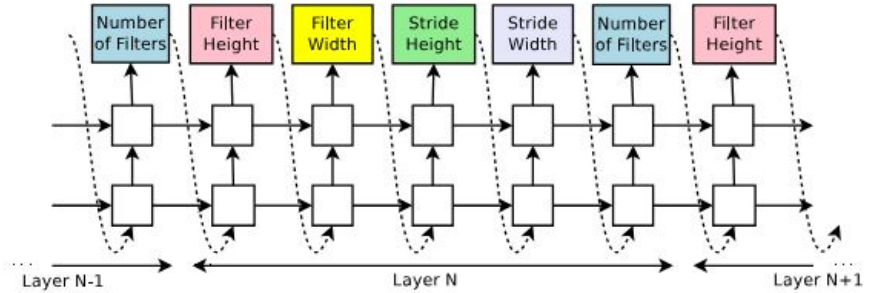


Figure 1: An overview of Neural Architecture Search.

Источник

Neural architecture search with reinforcement learning.



Как контроллер(RNN) семплирует простую сверточную сеть.

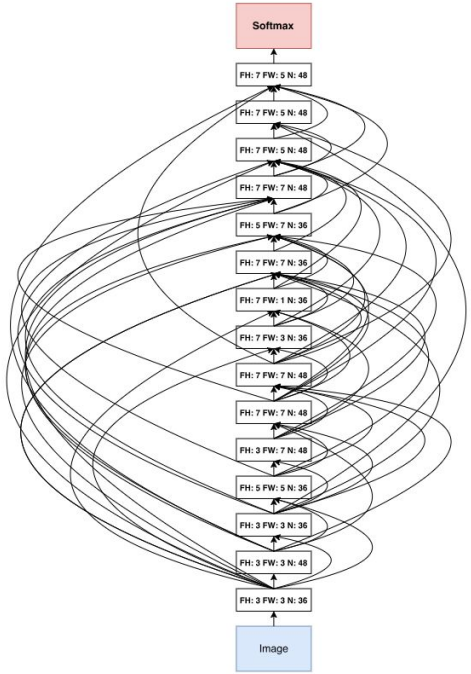
По очереди предсказываются ширина, высота и другие параметры свертки для одного слоя, после чего предсказываем параметры следующего. Каждое предсказание - многоклассовая классификация

Источник

Лучше чем SOTA(на тот момент). CIFAR-10

Model	Depth	Parameters	Error rate (%)
Network in Network (Lin et al., 2013)	-	-	8.81
All-CNN (Springenberg et al., 2014)	-	-	7.25
Deeply Supervised Net (Lee et al., 2015)	-	-	7.97
Highway Network (Srivastava et al., 2015)	-	-	7.72
Scalable Bayesian Optimization (Snoek et al., 2015)	-	-	6.37
FractalNet (Larsson et al., 2016)	21	38.6M	5.22
with Dropout/Drop-path	21	38.6M	4.60
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
Wide ResNet (Zagoruyko & Komodakis, 2016)	16	11.0M	4.81
	28	36.5M	4.17
ResNet (pre-activation) (He et al., 2016b)	164	1.7M	5.46
	1001	10.2M	4.62
DenseNet ($L = 40, k = 12$) Huang et al. (2016a)	40	1.0M	5.24
DenseNet($L = 100, k = 12$) Huang et al. (2016a)	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) Huang et al. (2016a)	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) Huang et al. (2016b)	190	25.6M	3.46
Neural Architecture Search v1 no stride or pooling	15	4.2M	5.50
Neural Architecture Search v2 predicting strides	20	2.5M	6.01
Neural Architecture Search v3 max pooling	39	7.1M	4.47
Neural Architecture Search v3 max pooling + more filters	39	37.4M	3.65

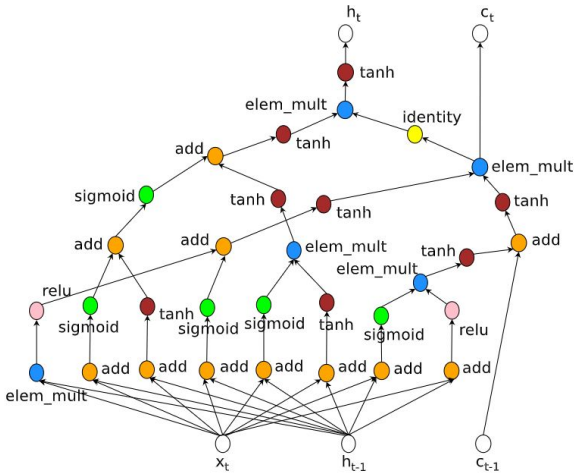
Пример архитектуры



Источник

Лучше чем SOTA(на тот момент).
Penn Treebank dataset - языковое моделирование

Model	Parameters	Test Perplexity
Mikolov & Zweig (2012) - KN-5	2M‡	141.2
Mikolov & Zweig (2012) - KN5 + cache	2M‡	125.7
Mikolov & Zweig (2012) - RNN	6M‡	124.7
Mikolov & Zweig (2012) - RNN-LDA	7M‡	113.7
Mikolov & Zweig (2012) - RNN-LDA + KN-5 + cache	9M‡	92.0
Pascanu et al. (2013) - Deep RNN	6M	107.5
Cheng et al. (2014) - Sum-Prod Net	5M‡	100.0
Zaremba et al. (2014) - LSTM (medium)	20M	82.7
Zaremba et al. (2014) - LSTM (large)	66M	78.4
Gal (2015) - Variational LSTM (medium, untied)	20M	79.7
Gal (2015) - Variational LSTM (medium, untied, MC)	20M	78.6
Gal (2015) - Variational LSTM (large, untied)	66M	75.2
Gal (2015) - Variational LSTM (large, untied, MC)	66M	73.4
Kim et al. (2015) - CharCNN	19M	78.9
Press & Wolf (2016) - Variational LSTM, shared embeddings	51M	73.2
Merity et al. (2016) - Zoneout + Variational LSTM (medium)	20M	80.6
Merity et al. (2016) - Pointer Sentinel-LSTM (medium)	21M	70.9
Inan et al. (2016) - VD-LSTM + REAL (large)	51M	68.5
Zilly et al. (2016) - Variational RHN, shared embeddings	24M	66.0
Neural Architecture Search with base 8	32M	67.9
Neural Architecture Search with base 8 and shared embeddings	25M	64.0
Neural Architecture Search with base 8 and shared embeddings	54M	62.4



Пример найденной ячейки, которая лучше LSTM

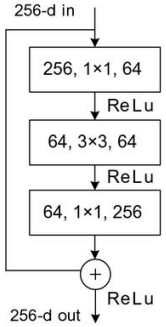
Источник

Резюме:

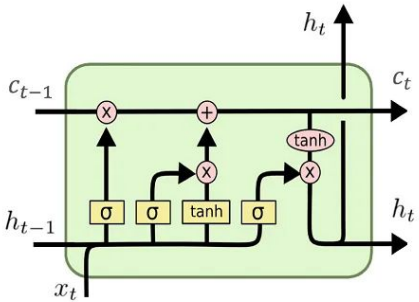
1. Одни из первых работ по NAS были основаны на методе поиска при помощи RL
2. Агент генерировал параметры сети последовательно, а в качестве награды получал качество обученной с нуля модели.
3. Уже эти работы находили архитектуры лучшего качества чем созданные человеком.
4. Но работают они очень долго и требуют очень много ресурсов

>>> Разделение весов (Weights sharing)

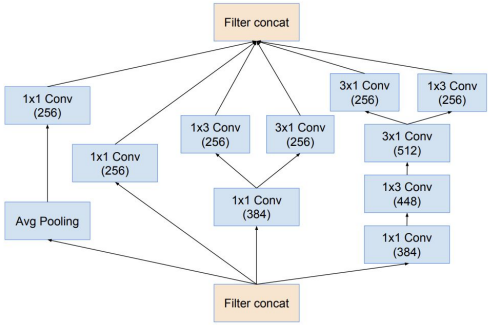
Направленный граф без циклов (DAG)



Блок ResNet



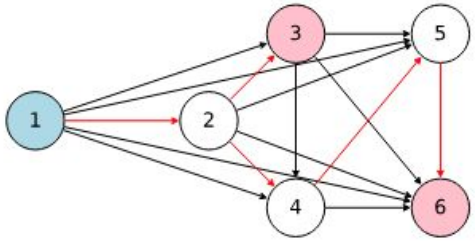
LSTM



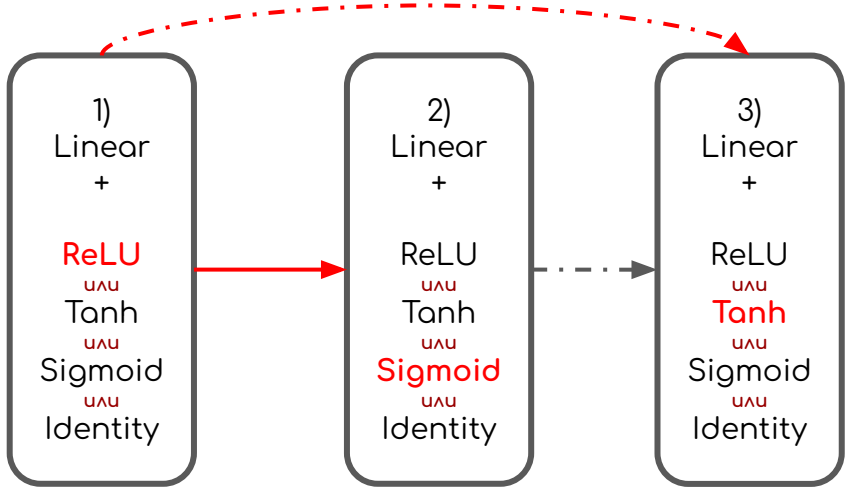
Блок Inception-v4

Supernet.

Направленный граф без циклов, в узлах которого лежат операции, а ребрами обозначается течение данных. Supernet используется для определения пространства поиска.



Граф, описывающий пространство поиска. Красные стрелки определяют модель, они определяются контроллером. Узел 1 - входной. Узлы 3 и 6 - выходные.



Источник

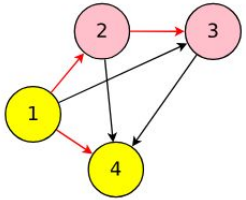
Разделение весов

С технихникой weights sharing все архитектуры являются подсетями большой модели - Supernet.
При обучении следующего кандидата мы фактически учим подмножество весов Supernet.

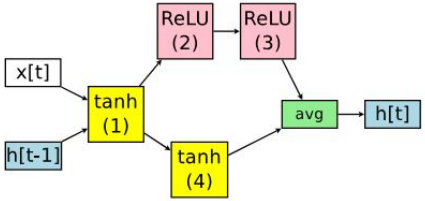
Т.е. теперь следующие кандидаты будут переиспользовать и продолжать обучать веса предыдущих.

Источник

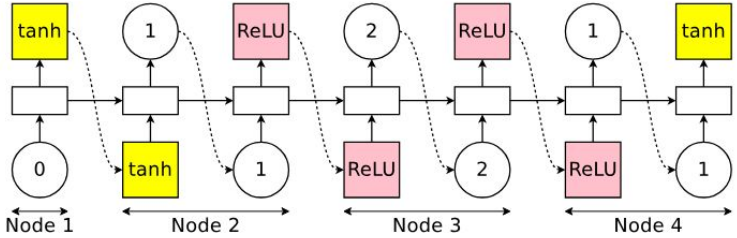
ENAS



Супернет 4 мя узлами, который определяет пространство поиска.



Пример рекурентного блока, выбранной из этого супернета



Пошаговое построение этого блока при помощи контроллера

Источник

ENAS

Итоговые вычисления архитектуры из примера проходят так:

$$h_1 = \tanh\left(\mathbf{x}_t \cdot \mathbf{W}^{(\mathbf{x})} + \mathbf{h}_{t-1} \cdot \mathbf{W}_1^{(\mathbf{h})}\right).$$
$$h_2 = \text{ReLU}(h_1 \cdot \mathbf{W}_{2,1}^{(\mathbf{h})}).$$
$$h_3 = \text{ReLU}(h_2 \cdot \mathbf{W}_{3,2}^{(\mathbf{h})})$$
$$h_4 = \tanh\left(h_1 \cdot \mathbf{W}_{4,1}^{(\mathbf{h})}\right)$$
$$\mathbf{h}_t = (h_3 + h_4)/2.$$

Источник

ENAS

1. Все модели - подграфы в Supernet
2. Модели выбираются при помощи - Reinforcement Learning
3. Используют технику - shared parameters
4. За счет 3) работают в 1000x быстрее предыдущих работ

Источник

ENAS

Еще лучше чем SOTA(на тот момент).
Penn Treebank dataset - языковое моделирование

Architecture	Additional Techniques	Params (million)	Test PPL
LSTM (Zaremba et al., 2014)	Vanilla Dropout	66	78.4
LSTM (Gal & Ghahramani, 2016)	VD	66	75.2
LSTM (Inan et al., 2017)	VD, WT	51	68.5
RHN (Zilly et al., 2017)	VD, WT	24	66.0
LSTM (Melis et al., 2017)	Hyper-parameters Search	24	59.5
LSTM (Yang et al., 2018)	VD, WT, ℓ_2, AWD, MoC	22	57.6
LSTM (Merity et al., 2017)	VD, WT, ℓ_2, AWD	24	57.3
LSTM (Yang et al., 2018)	VD, WT, ℓ_2, AWD, MoS	**22**	**56.0**
NAS (Zoph & Le, 2017)	VD, WT	54	62.4
ENAS	VD, WT, ℓ_2	**24**	**56.3**

Источник

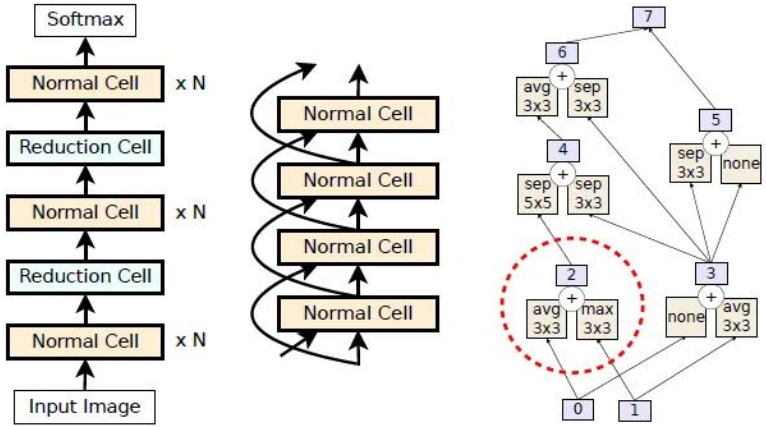
Резюме:

1. Supernet - направленный граф без циклов, который часто используется для определения пространства поиска
2. Weights sharing - все архитектуры являются подсетями большой модели (Supernet). При обучении очередного кандидата мы фактически учим подмножество весов Supernet.
3. Это позволяет ускорить поиск в 1000 раз и улучшить финальное качество.

>>> Эволюционный алгоритм

AmoebaNET

1. Внешняя структура сети фиксирована, ищем только блоки
2. Чтобы найти архитектуру, используем эволюционный алгоритм.

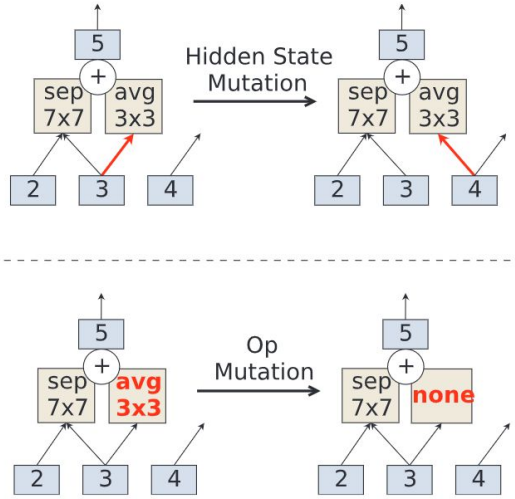


Пространство поиска

Источник

AmoebaNET

Мутации - небольшое случайное изменение архитектуры.



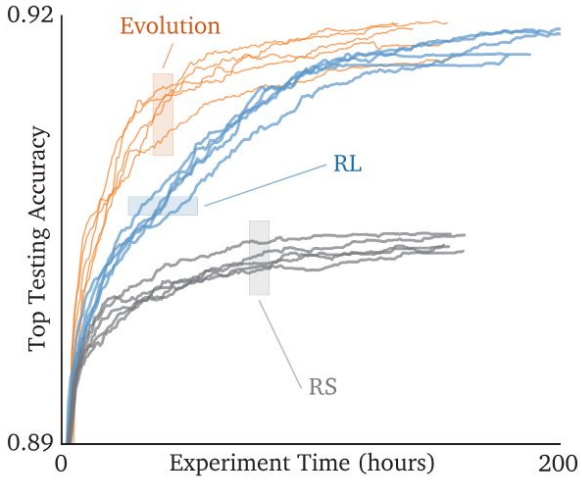
Источник

Algorithm 1 Aging Evolution (*i.e.* Regularized Evolution)

```
population ← empty queue                          ▷ The population.
history ← ∅                                       ▷ Will contain all models.
while |population| < P do                         ▷ Initialize population.
    model.arch ← RANDOMARCHITECTURE()
    model.accuracy ← TRAINANDEVAL(model.arch)
    add model to right of population
    add model to history
end while
while |history| < C do                            ▷ Evolve for C cycles.
    sample ← ∅                                    ▷ Parent candidates.
    while |sample| < S do
        candidate ← random element from population
                                                  ▷ The element stays in the population.
        add candidate to sample
    end while
    parent ← highest-accuracy model in sample
    child.arch ← MUTATE(parent.arch)
    child.accuracy ← TRAINANDEVAL(child.arch)
    add child to right of population
    add child to history
    remove dead from left of population           ▷ Oldest.
    discard dead
end while
return highest-accuracy model in history
```

Источник

AmoebaNET



Источник

AmoebaNET

ImageNet классификация.
Находит архитектуры лучше SOTA

Model	# Parameters	# Multiply-Adds	Top-1 / Top-5 Accuracy (%)
Incep-ResNet V2 (Szegedy et al. 2017)	55.8M	13.2B	80.4 / 95.3
ResNeXt-101 (Xie et al. 2017)	83.6M	31.5B	80.9 / 95.6
PolyNet (Zhang et al. 2017)	92.0M	34.7B	81.3 / 95.8
Dual-Path-Net-131 (Chen et al. 2017)	79.5M	32.0B	81.5 / 95.8
GeNet-2 (Xie and Yuille 2017)*	156M	–	72.1 / 90.4
Block-QNN-B (Zhong, Yan, and Liu 2018)*	–	–	75.7 / 92.6
Hierarchical (Liu et al. 2018b)*	64M	–	79.7 / 94.8
NASNet-A (Zoph et al. 2018)	88.9M	23.8B	82.7 / 96.2
PNASNet-5 (Liu et al. 2018a)	86.1M	25.0B	82.9 / 96.2
AmoebaNet-A (N=6, F=190)*	86.7M	23.1B	82.8 / 96.1
AmoebaNet-A (N=6, F=448)*	469M	104B	83.9 / 96.6

Источник

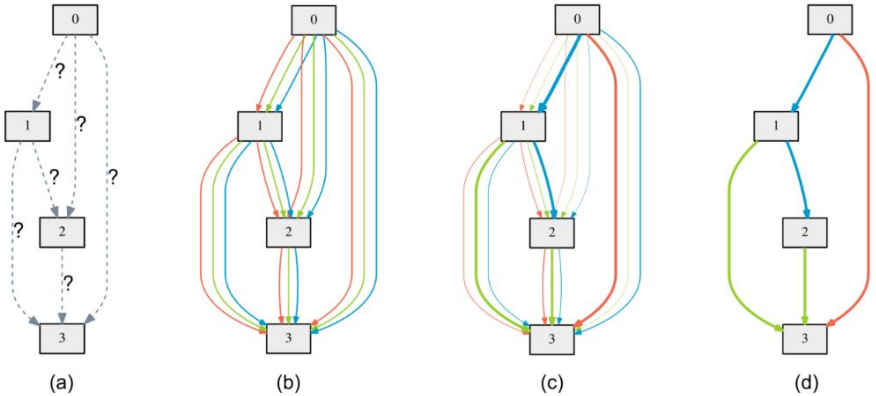
Резюме:

1. Многие работы по NAS фиксируют внешнюю структуру и ищут только повторяющиеся блоки.
2. Эволюционный алгоритм использует мутации - небольшое случайное изменение архитектуры сети для создания нового потомства.
3. При создании потомства от лучших родителей мы постепенно находим все лучшие модели

DARTS
>>>(Differentiable ARchiTecture Search)

DARTS

Задача - выбрать операции на каждой грани.



В процессе обучения учатся параметры при помощи которых будут выбраны операции и в конце выбирается конкретная сеть.

Создается супернет и делается релаксация путем смешивания операций на каждом узле, после чего весь супернет учится одновременно

Источник

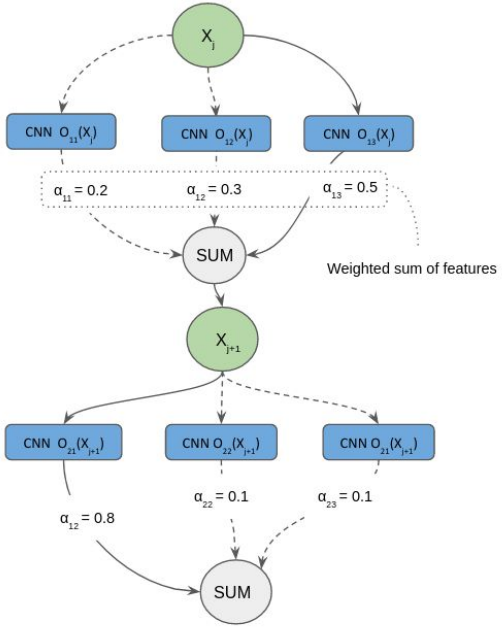
DARTS

Было:
На каждой итерации сэмплировать подсеть и обучалась только она.

Стало:
Все возможные подсети обучаются одновременно на каждой итерации.

Источник

DARTS



$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$

8 операций:

- 3x3, 5x5 sep. conv;
- 3x3, 5x5 dil. sep. conv;
- 3x3 max pooling
- 3x3 avg. pooling
- identity
- zero

Источник

DARTS

Было:
Архитектура выбиралась при помощи RL агента.

Стало:
Архитектура выбирается за счет обучаемых параметров alpha.

Причем оптимизируется супернет при помощи **двухуровневой оптимизации**

Hyperparameter optimization is a bilevel optimization problem

$$\begin{aligned} \min_{\alpha} \quad & L_{val}(w^*(\alpha), \alpha) \\ s.t. \quad & w^*(\alpha) = \operatorname*{argmin}_{w} L_{train}(w, \alpha) \end{aligned}$$

Источник

DARTS

Теперь поиск архитектуры проходит в два этапа:

1. Обучение супернета
2. Обучение выбранной из супернета архитектуры

Источник

DARTS

CIFAR-10 классификация.
Находит архитектуры лучше SOTA

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	#ops	Search Method
DenseNet-BC (Huang et al., 2017)	3.46	25.6	–	–	manual
NASNet-A + cutout (Zoph et al., 2018)	2.65	3.3	2000	13	RL
NASNet-A + cutout (Zoph et al., 2018)†	2.83	3.1	2000	13	RL
BlockQNN (Zhong et al., 2018)	3.54	39.8	96	8	RL
AmoebaNet-A (Real et al., 2018)	3.34 ± 0.06	3.2	3150	19	evolution
AmoebaNet-A + cutout (Real et al., 2018)†	3.12	3.1	3150	19	evolution
AmoebaNet-B + cutout (Real et al., 2018)	2.55 ± 0.05	2.8	3150	19	evolution
Hierarchical evolution (Liu et al., 2018b)	3.75 ± 0.12	15.7	300	6	evolution
PNAS (Liu et al., 2018a)	3.41 ± 0.09	3.2	225	8	SMBO
ENAS + cutout (Pham et al., 2018b)	2.89	4.6	0.5	6	RL
ENAS + cutout (Pham et al., 2018b)*	2.91	4.2	4	6	RL
Random search baseline‡ + cutout	3.29 ± 0.15	3.2	4	7	random
DARTS (first order) + cutout	3.00 ± 0.14	3.3	1.5	7	gradient-based
DARTS (second order) + cutout	2.76 ± 0.09	3.3	4	7	gradient-based

Источник

DARTS

Преимущества:

1. Размер пространства поиска можно значительно увеличить с относительно небольшим увеличением времени поиска.
2. SOTA качество(первая работа, не основанная на сэмплировании)

Источник

DARTS

Проблемы:
1. Обучение alpha
2. Weights coadaptation
3. Память
4. Нестабльность метода(требует нескольких запусков)

Позднее было выпущено множество статей, направленных на улучшение этого метода.

P-DARTS, PC-DARTS, DARTS+, GDAS, SDARTS, SGAS, DARTS+PT....

Источник

Позднее было выпущено огромное множество модификаций этого подхода

Algorithms	Search Space				Search Method			Search type	
	NASNet	DARTS	MobileNet	Others	RL	EA	GB	Micro	Macro
MetaQNN[2]				✓	✓				✓
SMASH[34]				✓		✓			✓
Large-Scale Evolution of ICs 2017[24]				✓		✓			✓
NOS with RL 2017[35]				✓	✓				✓
NASBOT 2018[36]				✓			✓		✓
SNAS[8]		✓					✓	✓	
BlockQNN 2018[37]				✓	✓				✓
DARTS[4]		✓					✓	✓	
Understanding One-Shot Models [38]				✓		✓			✓
ENAS[25]	✓				✓			✓	✓
Progressive NAS[39]	✓				✓			✓	
NASNet [21]	✓				✓			✓	
NAONet [40]					✓	✓		✓	
Proxylessnas[33]		✓					✓	✓	
FBNet[41]			✓				✓		✓
MNASNet[42]			✓						✓
ChamNet[43]				✓					✓
SPNAS[44]			✓				✓		✓
AmoebaNet [23]	✓					✓		✓	
GDAS[45]		✓					✓	✓	
EfficientNet[46]			✓			✓			✓
FairNAS[30]		✓					✓	✓	
PCDARTS[5]		✓					✓	✓	
RDARTS[6]		✓					✓	✓	
BayenNAS[7]		✓					✓	✓	
PDARTS[29]		✓					✓	✓	
XNAS[47]		✓					✓	✓	
DARTS+[31]		✓					✓	✓	
NAT[48]		✓					✓	✓	
SETN[49]		✓					✓	✓	
SPOSNAS[50]				✓			✓		✓
Smooth DARTS[51]			✓				✓	✓	

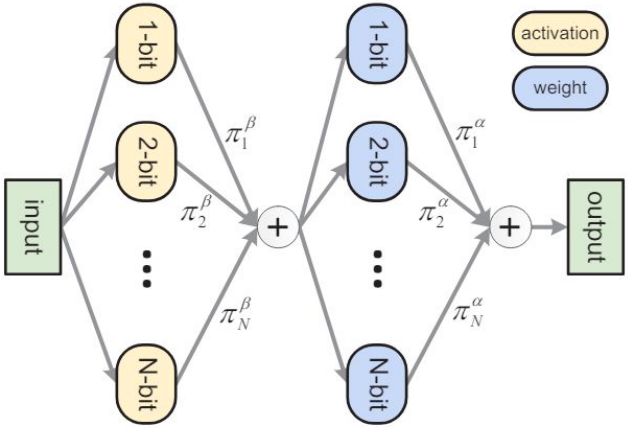
Источник

Резюме:

1. DiffNAS - один из самых популярных подходов к поиску архитектуры за счет своей эффективности и высокого качества.
2. Вместо обучения отдельных кандидатов, в DARTS обучается сразу весь Supernet.
3. У этого подхода есть много проблем с оптимизацией, и еще больше решений было предложено.

>>> Mixed-precision quantization

EdMIPS



В каждом слое при помощи DiffNAS ищутся уровень битовости для активаций и для весов

Источник

EdMIPS

$$y = f(a(x)) = \mathbf{W} * a(x),$$

Рассмотрим один сверточный слой

$$y = \sum_{i=1}^{n_f} \pi_i^\alpha f_i \left(\sum_{j=1}^{n_a} \pi_j^\beta a_j(x) \right)$$

График на предыдущем слайде описывается такой формулой

$$= \left(\sum_{i=1}^{n_f} \pi_i^\alpha Q_i(\mathbf{W}_i) \right) * \bar{a}(x) = \bar{f}(\bar{a}(x))$$

Которую можно переписать вот так. При этом, для каждого уровня битовости будет отдельные W_i веса

$$\overline{\mathbf{W}} = \sum_{i=1}^{n_f} \pi_i^\alpha Q_i(\mathbf{W}).$$

Авторы придумали для экономии ресурсов вместо этого хранить одни и те же веса и квантизировать их на каждом шаге

Источник

EdMIPS

Финальный лосс - сумма обычного и лосса для регуляризации флопсов.

$$\mathcal{L}[F] = \mathcal{R}_E[F] + \eta\mathcal{R}_C[F], \tag{6}$$

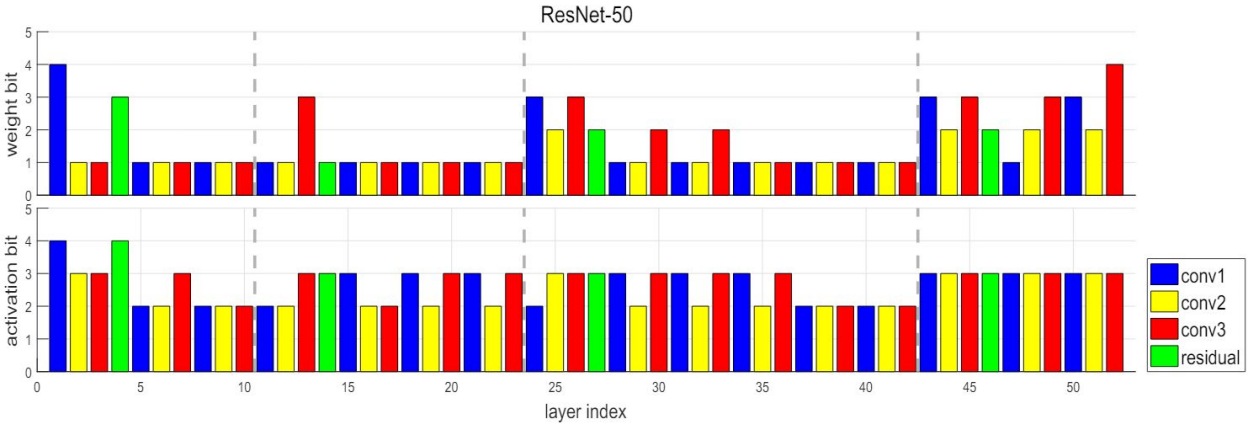
При этом R_c - сумма флопсов по всем фильтрам сети.

$$c(f) = E[b_f]E[b_a]|f|w_x h_x/s^2, \tag{12}$$

$$E[b_f] = \sum_{i=1}^{n_f} \pi_i^\alpha b_{f_i}, \quad E[b_a] = \sum_{j=1}^{n_a} \pi_j^\beta b_{a_j} \tag{13}$$

Источник

EdMIPS



Источник

EdMIPS

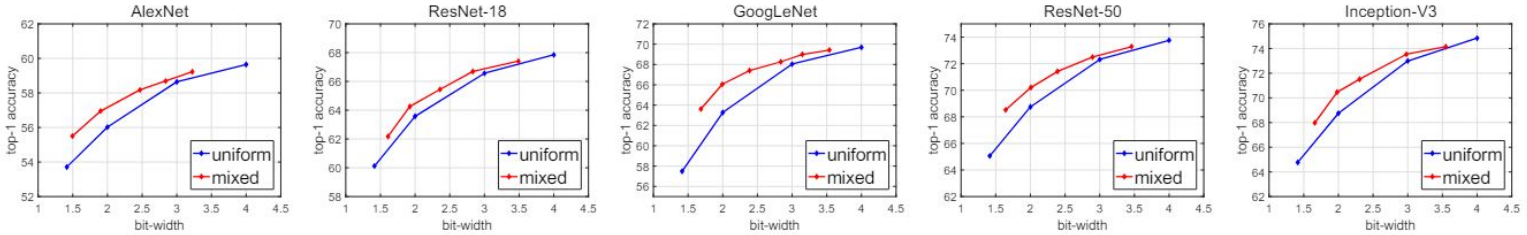


Figure 5. Comparison of the uniform HWGQ-Net and the EdMIPS network. The x-axis, indicating BitOps, is normalized to the scale of bit-width, which is actually in log-scale.

Источник

Резюме:

1. Mixed precision quantization - задача NAS
2. Ее можно решать при помощи дифференцируемого поиска
3. Первые и последние слои модели часто требуют более высокий уровень точности

>>> Рассмотрим два более узких примера NAS в реальных задачах

>>> 1. QuantNAS

QuantNAS

1. Фокус на задаче Super Resolution
2. Ищет одновременно уровень битовости и операцию.
3. Применим к любой модели Super Resolution.
4. Использует энтропийную регуляризацию
5. Использует квантизационный шум.

Источник

QuantNAS

$$L(\boldsymbol{\alpha}) = L_1(\boldsymbol{\alpha}) + \eta L_{cq}(\boldsymbol{\alpha}) + \mu(t) L_e(\boldsymbol{\alpha}),$$

$$L_{cq}(\boldsymbol{\alpha}) = \sum_{l=1}^{|S|} \sum_{i=1}^{|O^l|} \sum_{b=1}^{|B|} \alpha_{ib}^l b^2 F_{fp}(o_i^l, x_l), \qquad (8)$$

$$L_e(\boldsymbol{\alpha}) = \sum_{l=1}^{|S|} H(\boldsymbol{\alpha}_l), \qquad (9)$$

Как и в EdMIPS, есть обычный лосс и регуляризация на флопсы, но дополнительно есть энтропийный лосс для лучшей сходимости супернета.

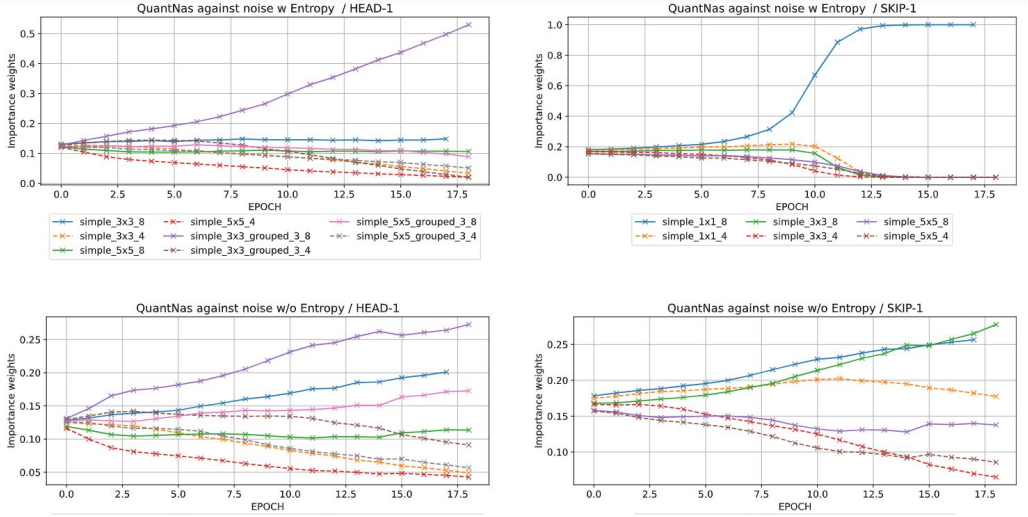
В отличии от EdMIPS, тут сохраняются одинаковый уровень битовости для весов и активаций(это более честно, тк железо иначе не умеет)

Энтропийный лосс это сумма энтропий с каждого ребра

Источник

QuantNAS

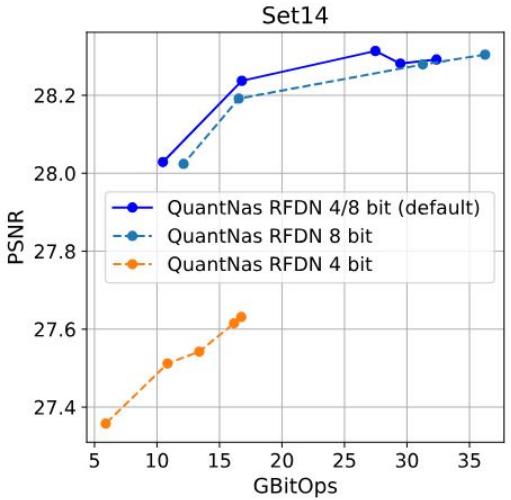
Энтропийный лосс действительно помогает



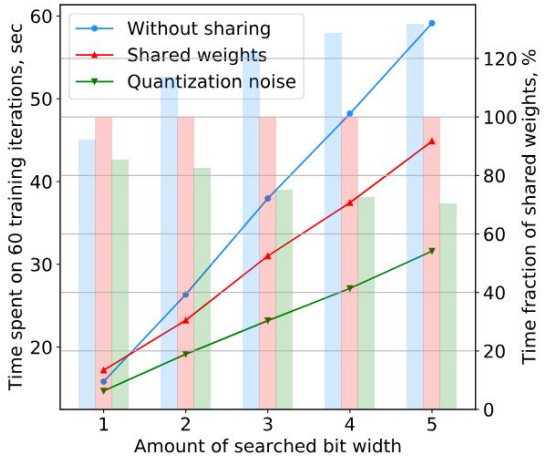
Источник

QuantNAS

В итоге получается лучше решение



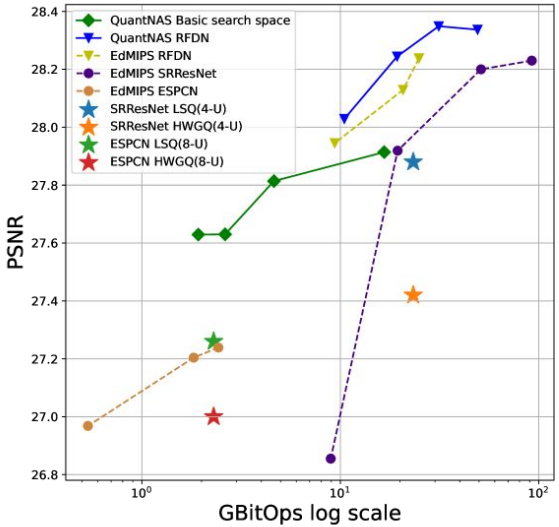
За меньшее количество времени



Источник

QuantNAS

Сравнение с другими методами.
DIV2K super resolution



Источник

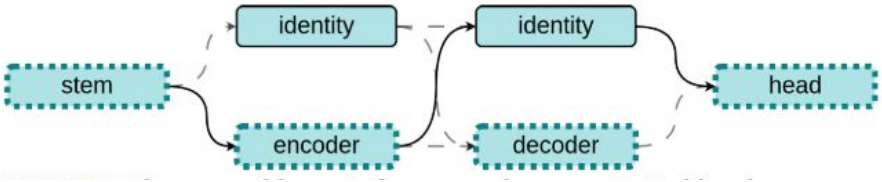
>>> 2. SeqNAS

SeqNAS

1. Фокус на классификации последовательностей.
2. Использует дистилляцию
3. Кодирует каждую архитектуру в one-hot представление.
4. Поиск архитектуры на основе байесовского семплирования.

Источник

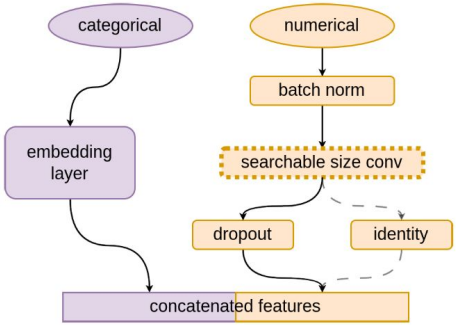
SeqNAS



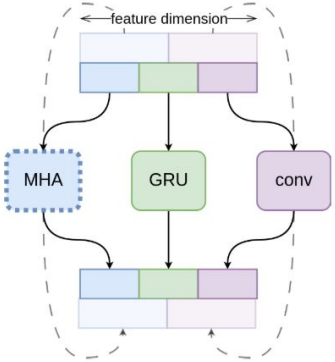
Общий вид пространства поиска. В пунктирных блоках выбираются операции, прерывистые линии означают возможность выбора соединений. Черной линией указан пример течения данных

Источник

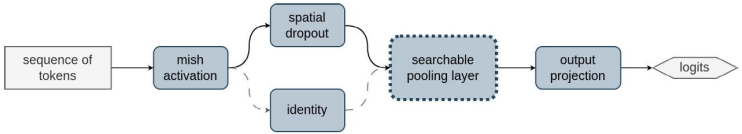
SeqNAS



Пространство поиска для Stem блока



Пространство поиска для encoder, decoder



Head блок

Источник

Алгоритм

1. Сгенерируем случайно случайный набор архитектур и обучим их на нашей задаче, оценим их качество.
2. Обучим модель предсказывать по архитектуре ее качество на задаче.
3. Далее в цикле будем повторять следующие этапы, пока не израсходуем бюджет на оптимизацию, либо пока не найдем устраивающую нас архитектуру:
    a. Сгенерируем случайно большой набор архитектур.
    b. С помощью обученной модели отберем из них подмножество *наиболее перспективных* (что значит «наиболее перспективные» — смотри Thompson sampling)
    c. Обучим эти перспективные архитектуры, оценим их качество.
    d. Дообучим нашу модель на этом подмножестве перспективных архитектур.

Также для повышения качества мы дистиллируем новые архитектуры на предсказаниях лучших из ранее обученных. Даже если обученные архитектуры-учителя хуже архитектуры-студента, которую мы обучаем, то студент все равно обучается лучше, чем если бы дистилляция не использовалось.

За счет умного выборы перспективных архитектур мы не тратим ресурсы на обучение плохих архитектур, как в случайном поиске.

Источник

SeqNAS

TABLE 2. Comparison of our method with two NAS procedures 1) AutoAttend [13], 2) TextNAS [14] and four fixed architectures 3) Gated Transformer Networks [16], and baseline models such as 4) Fixed Transformer, 5) GRU, 6) LSTM. We report MEAN and STD of the 3 best models found, for both HPO and NAS procedures. We mark the First and the Second best performing models as highlighted in this text.

Dataset	Model Search Space	SeqNAS	AutoAttend	TextNAS	GTN	Fixed TF	GRU	LSTM
	Metric / Search Method	Our	Context-Aware Weight Sharing	ENAS	HPO	HPO	HPO	HPO
AmEx	Custom[2]	**0.7911 ± 0.0004**	0.6170 ± 0.0033	0.7818 ± 0.002	0.7717 ± 0.006	0.7850 ± 0.0002	0.7718 ± 0.0005	0.7709 ± 0.0005
ABank	ROC-AUC	**0.7963 ± 0.0014**	0.6827 ± 0.0160	0.7653 ± 0.002	0.7462 ± 0.001	0.7747 ± 0.0011	0.7699 ± 0.0002	0.7451 ± 0.0032
VBank	ROC-AUC	**0.8032 ± 0.0022**	0.6533 ± 0.0408	0.7951 ± 0.001	0.7362 ± 0.001	0.7883 ± 0.0013	0.7980 ± 0.0008	0.7704 ± 0.0012
RBchurn	ROC-AUC	**0.8525 ± 0.0033**	0.7345 ± 0.0028	0.7936 ± 0.002	0.7701 ± 0.003	0.8170 ± 0.0012	0.8300 ± 0.002	0.8090 ± 0.0027
AGE	Accuracy	**0.6445 ± 0.0018**	0.6251 ± 0.0013	0.6016 ± 0.003	0.5363 ± 0.019	0.6170 ± 0.001	0.6300 ± 0.001	0.5920 ± 0.0010
TaoBao	ROC-AUC	**0.7138 ± 0.0007**	0.6352 ± 0.0023	0.7079 ± 0.002	0.6713 ± 0.001	0.7107 ± 0.0011	0.7100 ± 0.0004	0.6680 ± 0.0008

Udovichenko, Igor, et al. "SeqNAS: Neural architecture search for event sequence classification." *IEEE Access* (2024).

NAS results

Problem	Dataset	NAS method	Human arc.perf.	NAS Perf.	Diff.
Machine translation	WMT'14 En-De	Evolution	BLEU=28.8 Perplexity=4.05 Transformer *[Vasnawi et al., 2017]*	BLEU=29.0 Perplexity=3.94 *Evolved Transformer [So et al. 2019]*	+0.7% -3.0% **lower is better**
Object classification	CIFAR-10	DARTS	Accuracy = 96.54% *DenseNet-BC, [Huang et al., 2017]*	Accuracy=97.24% *[Liu et al. 2018]*	+0.7%
Semantic segmentation	Cityscapes	Evolution	mIOU=71.8% *FRRN-B, [Pohlen et al., 2017]*	mIOU=80.4% *Auto-DeepLab-L [Liu et al, 2019]*	+8.6%

NAS results

Problem	Dataset	NAS method	Human arc.perf.	NAS Perf.	Diff.
Natural language modeling	Penn Tree Bank	ENAS	Perplexity=56.0 *[Yang et al, 2018]*	Perplexity=55.8 *[Zoph et al, 2018]*	-0.3% **lower is better**
Graph NN, Classification	Citeseer	ENAS	Accuracy=73.0% *LGCN* *[Gao et al, 2018]*	Accuracy=73.8% *Auto-GNN* *[Zhou et al., 2019]*	+1%
Deep RL	Atari	ENAS	Avg. reward = 172.8 *NatureCNN* *[Mnih et al., 2015]*	Avg. reward = 181.8 *Skoltech, 2019*	+5.2%
Object detection	CoCo	DARTS	Avg.precision = 0.064 *[Law et al., 2018]*	Avg. precision=0.078 *Skoltech, 2019*	+1.4%

Резюме

1. NAS - область, направленная на решении задачи поиска оптимальной архитектуры в условиях ограниченных ресурсов.
2. В любом NAS важно определить пространство поиска, поисковый алгоритм и стратегию оценки.
3. Мы рассмотрели три подхода к поиску, каждый из которых имеет свои преимущества и недостатки:
   a. RL
   b. Эволюционный подход
   c. Дифференцируемый NAS
4. Weights sharing - популярная техника, которая позволяет радикально сократить количество вычислений.

Спасибо за внимание!

TABLE 3. Gradient based NAS innovations.

Algorithm	Main context	Solving strategies	Outcomes
DARTS [4]	Formulate NAS as gradient descent based optimization problem.	Relax the discrete search space to be continuous. The softmax is used for smoothing the operation choices, and a candidate architecture is constructed by stacking the cell for training.	Optimize the architecture parameters via gradient descent and thus dramatically reduces the high search cost of NAS.
SNAS [8]	Formulate NAS as a stochastic model. Enhance RL with a smooth sampling scheme.	Samples and optimizes candidate architectures directly with concrete optimization [70].	More efficient and less regularization biased framework (compared with DARTS)
Proxylessnas [33]	A model trained and tested on different datasets often not guaranteed to be optimal.	Directly learn the architectures for large-scale target tasks and target hardware platforms.	Latency regularization loss helps for different hardware.
GDAS [45]	Formulate NAS as gradient descent problem	Samples one sub-graph at one training iteration	Better performance with less computing resources.
FairNAS [30]	Unfair bias in supernet sometimes reduce the performance of candidate architectures	Two levels of constraints: expectation fairness and strict fairness.	It can be adopted on any search pipeline.
PCDARTS [5]	DARTS based NAS suffered from large memory and computing overhead	Sample the supernet into a subnet and partially connect to construct a candidate architecture	Edge normalization can stabilize the search process.
RDARTS [6]	DARTS does not work robustly for new problem	Add different types of regularization methods with early stops.	Generalization improves in the search process.
BayesNAS [7]	Nodes inside normal and reduction cells often disregard their predecessors and successors.	A Hierarchical automatic relevance determination (HARD) approach is used to model architecture parameters.	Compress CNN by enforcing structural sparsity without accuracy deterioration
PDARTS [29]	Bridging the Depth Gap between Search and Evaluation	Gradually increase the searched architecture during training.	Regularized search space and improve accuracy.
XNAS [47]	New optimization method for differential NAS.	Designing for wiping out inferior architectures and enhance superior ones dynamically.	Fewer hyper-parameters need to be tuned.
DARTS+ [31]	Skip connection increases for larger epochs.	Early stopping into the original DARTS [4]	Improved the performance of DARTS.
NAT [48]	New optimization method for NAS	Redundant operations are replaced by the Markov decision process (MDP).	Reduces hyper-parameters and improve the accuracy
SETN [49]	After the search, a lengthy training requires to train the hyper-parameters for evaluations.	Template network shares parameters among all candidates.	Improve the quality of the candidate architecture for evaluation
StacNAS [71]	DARTS performs poorly when the search space is changed	Calculates correlation of similar operators incurs unfavorable competition among them.	Increase the stability and performance
Smooth DARTS [51]	Stabilize the architecture search process.	Perturbation-based regularization for improving the generalizability.	Stable candidate architecture.
DOTS [72]	Operation weights cannot indicate the importance of cell topology	Decouple the Operation and Topology Search (DOTS)	Topology search space to improve accuracy.
PARSEC [73]	Search directly on large scale problems.	Probability based architecture search approach	Reduce the computing costs.
SGAS [32]	Searched architectures often fail to generalize in the final evaluation.	Divides the search procedure into sub-problems, chooses, and greedily prunes candidate operations.	State-of-the-art architectures for tasks such as image classification
GDAS-NSAS [74]	Performance of preceding candidate architecture often degraded during training of new architecture with partially share weights.	Formulate supernet training as One-Shot NAS. During training, the performance of current architecture should not degrade the performance of preceding candidate architecture.	Improve predicatively of supernet in One-Shot NAS
DropNAS [75]	Co-adaption problem and Matthew Effect	Propose a novel grouped operation dropout algorithm	Achieves promising performance
DARTS- [76]	Instability issue during architecture searching	Skip connections with a learnable architectural coefficient	Improves the robustness of DARTS.
DrNAS [77]	Formulate the DARTS as a distribution learning problem	Progressive learning scheme to search architectures in a large dataset	Improves the generalization ability and induces stochasticity in search space

Santra, Santanu, Jun-Wei Hsieh, and Chi-Fang Lin. "Gradient descent effects on differential neural architecture search: A survey." *IEEE Access* 9 (2021): 89602-89618.