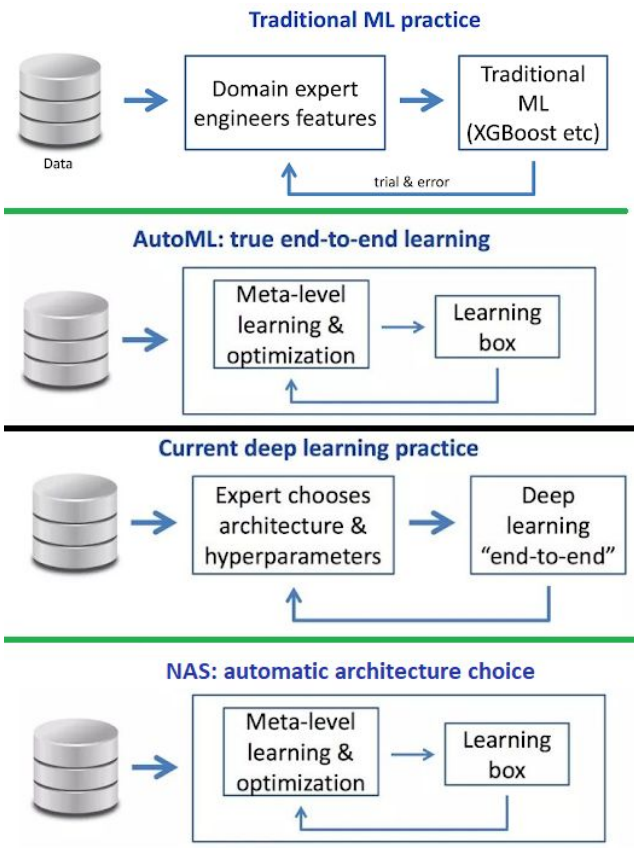
Поиск нейронных архитектур(NAS)

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**

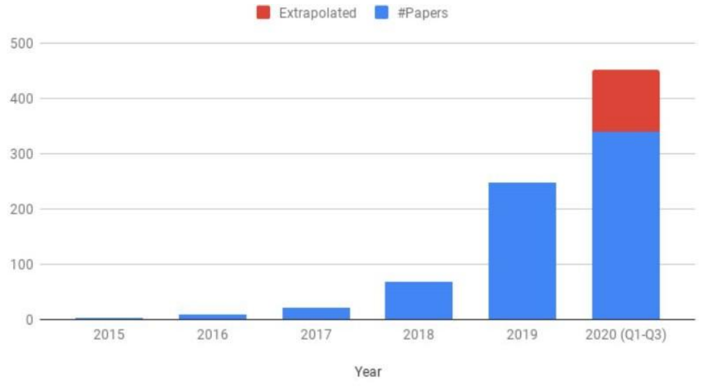
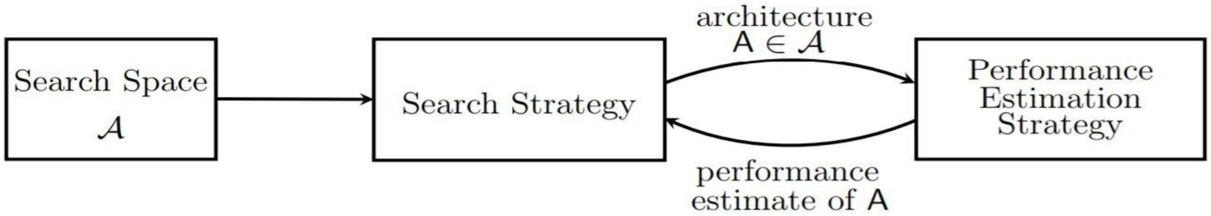


Figure 1: NAS papers per year based on the literature list on `automl.org`. Extrapolation for 2020 based on the first 9 months of the year.

Lindauer, M., & Hutter, F. (2020). Best practices for scientific research on neural architecture search. *Journal of Machine Learning Research*, *21*(243), 1-18.

>>> 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**

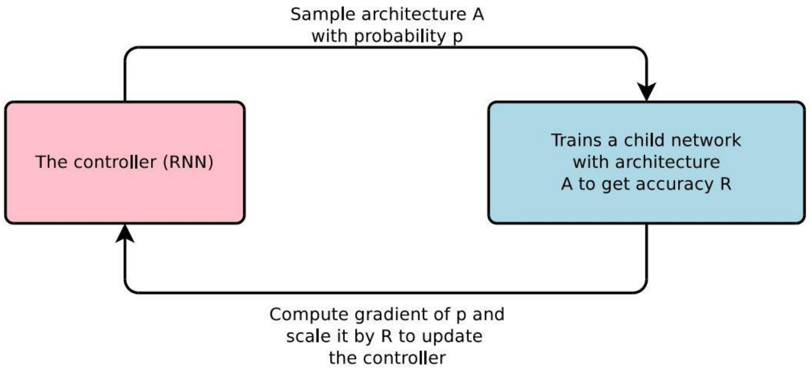


Figure 1: An overview of Neural Architecture Search.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Neural architecture search with reinforcement learning.

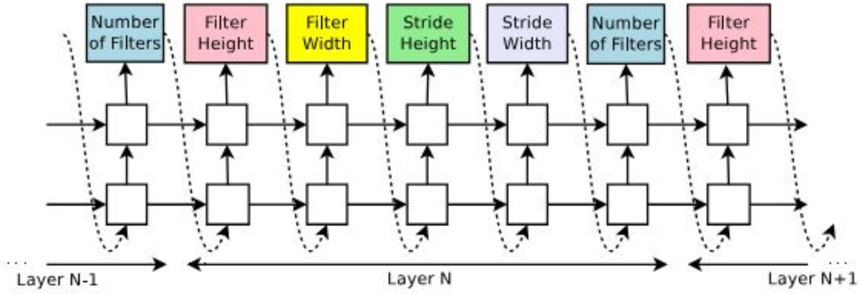


Figure 2: How our controller recurrent neural network samples a simple convolutional network. It predicts filter height, filter width, stride height, stride width, and number of filters for one layer and repeats. Every prediction is carried out by a softmax classifier and then fed into the next time step as input.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Results: After the controller trains 12,800 architectures, we find the architecture that achieves the best validation accuracy. We then run a small grid search over learning rate, weight decay, batchnorm epsilon and what epoch to decay the learning rate. The best model from this grid search is then run until convergence and we then compute the test accuracy of such model and summarize the results in Table 1. As can be seen from the table, Neural Architecture Search can design several promising architectures that perform as well as some of the best models on this dataset.

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

Table 1: Performance of Neural Architecture Search and other state-of-the-art models on CIFAR-10.

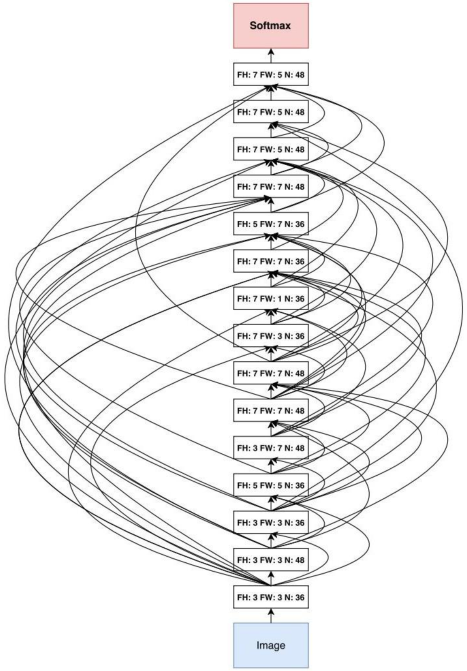
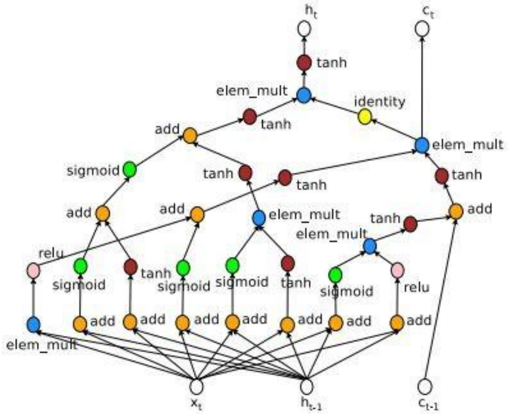


Figure 7: Convolutional architecture discovered by our method, when the search space does not have strides or pooling layers. FH is filter height, FW is filter width and N is number of filters. Note that the skip connections are not residual connections. If one layer has many input layers then all input layers are concatenated in the depth dimension.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

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

Table 2: Single model perplexity on the test set of the Penn Treebank language modeling task. Parameter numbers with ‡ are estimates with reference to Merity et al. (2016).



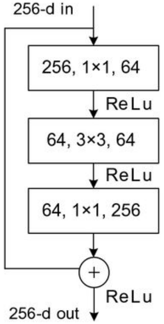
NAS нашел RNN ячейку лучше чем LSTM.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

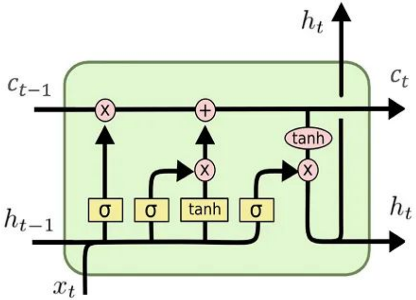
>>>

Разделение весов (Weights sharing)

Направленный граф без циклов (DAG)



(a) ResNet-50 bottleneck



LSTM
(Long-Short Term Memory)

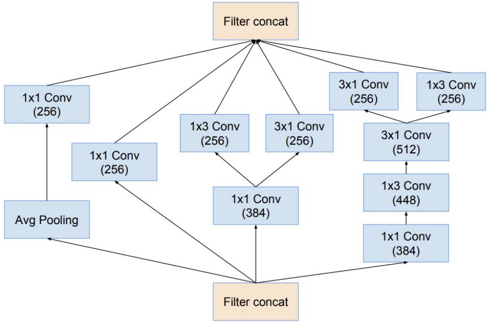


Figure 6. The schema for 8×8 grid modules of the pure Inception-v4 network. This is the Inception-C block of Figure 9.

Supernet

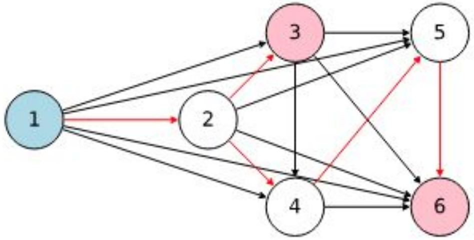
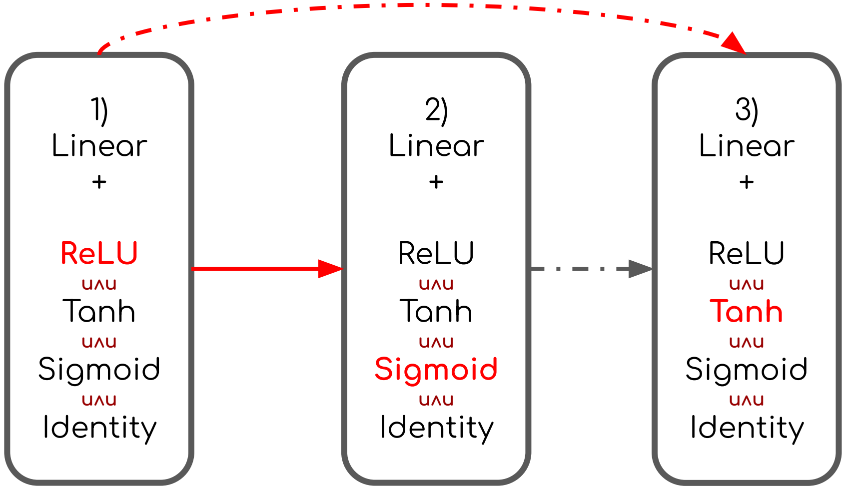


Figure 2. The graph represents the entire search space while the red arrows define a model in the search space, which is decided by a controller. Here, node 1 is the input to the model whereas nodes 3 and 6 are the model's outputs.



Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* (pp. 4095-4104). PMLR.

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.

Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* (pp. 4095-4104). PMLR.

ENAS

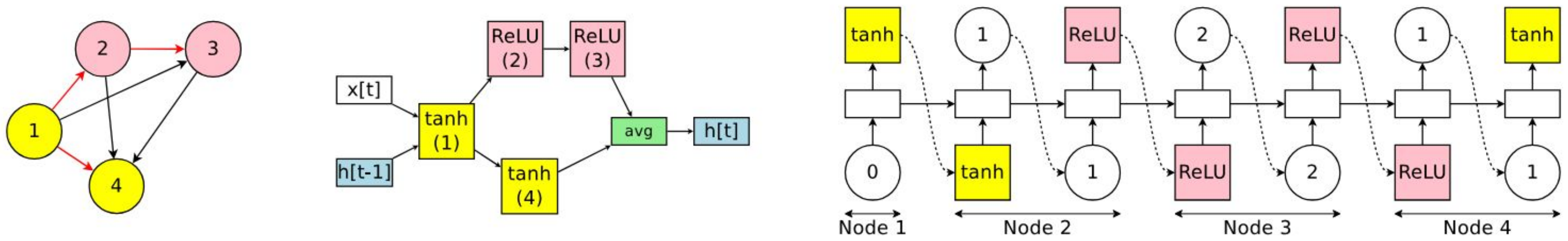


Figure 1. An example of a recurrent cell in our search space with 4 computational nodes. *Left:* The computational DAG that corresponds to the recurrent cell. The red edges represent the flow of information in the graph. *Middle:* The recurrent cell. *Right:* The outputs of the controller RNN that result in the cell in the middle and the DAG on the left. Note that nodes 3 and 4 are never sampled by the RNN, so their results are averaged and are treated as the cell's output.

Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* (pp. 4095-4104). PMLR.

ENAS

$$h_1 = \tanh\left(\mathbf{x}_t \cdot \mathbf{W}^{(\mathbf{x})} + \mathbf{h}_{t-1} \cdot \mathbf{W}_1^{(\mathbf{h})}\right).$$

$$h_2 = \mathrm{ReLU}(h_1 \cdot \mathbf{W}_{2,1}^{(\mathbf{h})}).$$

$$h_3 = \mathrm{ReLU}(h_2 \cdot \mathbf{W}_{3,2}^{(\mathbf{h})})$$

$$h_4 = \tanh\left(h_1 \cdot \mathbf{W}_{4,1}^{(\mathbf{h})}\right)$$

$$\mathbf{h}_t = (h_3 + h_4)/2.$$

Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* (pp. 4095-4104). PMLR.

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

Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* (pp. 4095-4104). PMLR.

ENAS

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**	$\boxed{\mathbf{56.3}}$

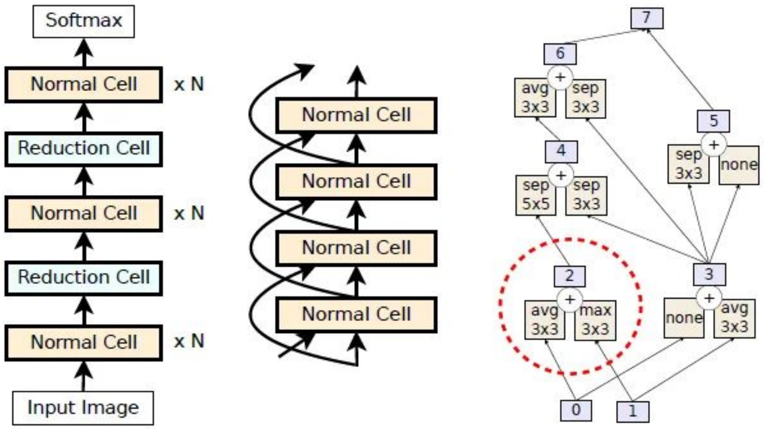
Table 1. Test perplexity on Penn Treebank of ENAS and other baselines. Abbreviations: RHN is *Recurrent Highway Network*, VD is *Variational Dropout*; WT is *Weight Tying*; ℓ_2 is *Weight Penalty*; AWD is *Averaged Weight Drop*; MoC is *Mixture of Contexts*; MoS is *Mixture of Softmaxes*.

Pham, H., Guan, M., Zoph, B., Le, Q., & Dean, J. (2018, July). Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning* (pp. 4095-4104). PMLR.

>>> Эволюционный алгоритм

AmoebaNET

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



NASNet Search Space

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, July). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, No. 01, pp. 4780-4789).

```
Algorithm 1 Aging 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
```

Aging Evolution

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, July). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, No. 01, pp. 4780-4789).

AmoebaNET

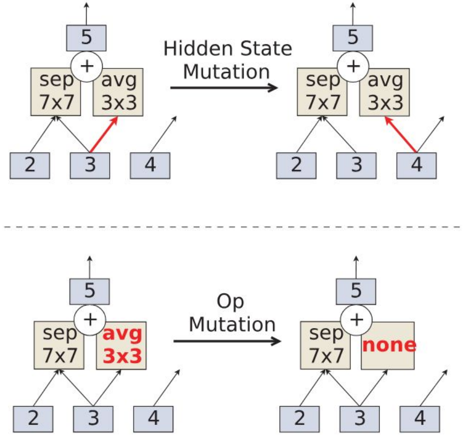
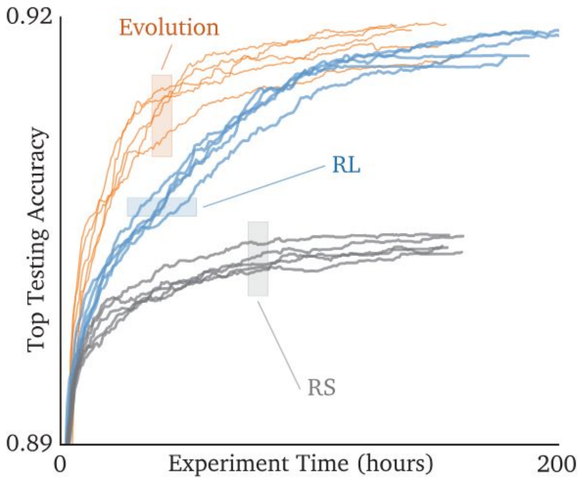


Figure 3: Illustration of the two mutation types.

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, July). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, No. 01, pp. 4780-4789).

AmoebaNET



Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, July). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, No. 01, pp. 4780-4789).

AmoebaNET

Table 2: ImageNet classification results for AmoebaNet-A compared to hand-designs (top rows) and other automated methods (middle rows). The evolved AmoebaNet-A architecture (bottom rows) reaches the current state of the art (SOTA) at similar model sizes and sets a new SOTA at a larger size. All evolution-based approaches are marked with a *. We omitted Squeeze-and-Excite-Net because it was not benchmarked on the same ImageNet dataset version.

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

Real, E., Aggarwal, A., Huang, Y., & Le, Q. V. (2019, July). Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, No. 01, pp. 4780-4789).

DARTS
>>>(Differentiable ARchiTecture Search)

DARTS

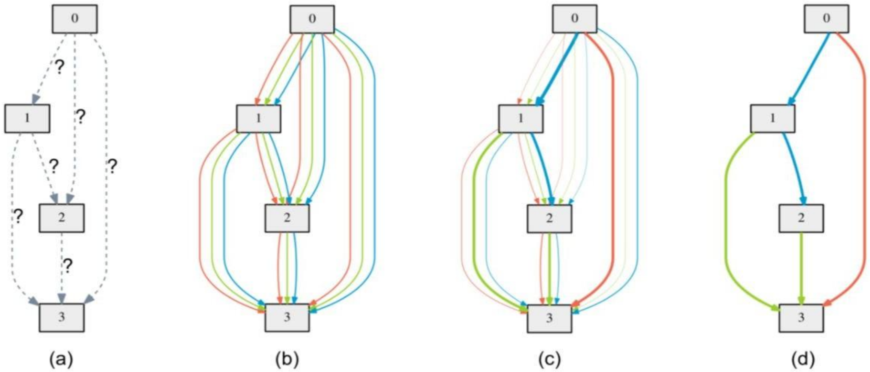


Figure 1: An overview of DARTS: (a) Operations on the edges are initially unknown. (b) Continuous relaxation of the search space by placing a mixture of candidate operations on each edge. (c) Joint optimization of the mixing probabilities and the network weights by solving a bilevel optimization problem. (d) Inducing the final architecture from the learned mixing probabilities.

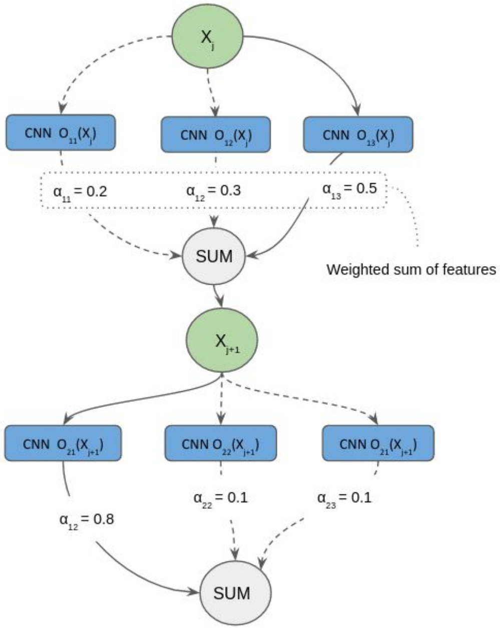
DARTS

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

Стало:
Все возможные подсети обучаются одновременно на каждой итерации.

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

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

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

DARTS

Было:
Архитектура выбиралась при помощи RL агента.

Стало:
Архитектура выбирается за счет обучаемых параметров alpha.

Hyperparameter optimization is a bilevel optimization problem

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

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

DARTS

Теперь поиск архитектуры проходит в два этапа:

1. Обучение супернета
2. Обучение выбранной из супернета архитектуры

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

DARTS

Table 1: Comparison with state-of-the-art image classifiers on CIFAR-10 (lower error rate is better). Note the search cost for DARTS does not include the selection cost (1 GPU day) or the final evaluation cost by training the selected architecture from scratch (1.5 GPU days).

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

* Obtained by repeating ENAS for 8 times using the code publicly released by the authors. The cell for final evaluation is chosen according to the same selection protocol as for DARTS.

† Obtained by training the corresponding architectures using our setup.

‡ Best architecture among 24 samples according to the validation error after 100 training epochs.

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

DARTS

Преимущества:

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

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

DARTS

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

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

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

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

TABLE 2. NAS-based methods with search spaces, search strategies, and search types.

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]			✓				✓	✓	

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.

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.

>>> Mixed-precision quantization

EdMIPS

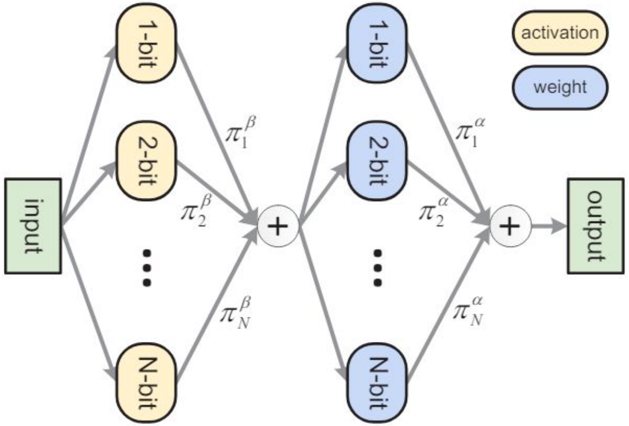


Figure 1. Differentiable architecture of the proposed mixed-precision network search module.

Cai, Zhaowei, and Nuno Vasconcelos. "Rethinking differentiable search for mixed-precision neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

EdMIPS

Deep networks implement many filtering operators,

$$y = f(a(x)) = \mathbf{W} * a(x), \qquad (1)$$

$$y = \sum_{i=1}^{n_f} \pi_i^\alpha f_i\left(\bar{a}(x)\right) = \sum_{i=1}^{n_f} \pi_i^\alpha \left(Q_i(\mathbf{W}_i) * \bar{a}(x)\right)$$
$$= \left(\sum_{i=1}^{n_f} \pi_i^\alpha Q_i(\mathbf{W}_i)\right) * \bar{a}(x) = \bar{f}(\bar{a}(x)) \qquad (14)$$

$$\overline{\mathbf{W}} = \sum_{i=1}^{n_f} \pi_i^\alpha Q_i(\mathbf{W}). \qquad (16)$$

Cai, Zhaowei, and Nuno Vasconcelos. "Rethinking differentiable search for mixed-precision neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

EdMIPS

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

$$\mathcal{R}_C[F] = \sum_{f\in F} c(f). \tag{7}$$

where $c(f)$ is the cost of filter f.

$$\pi_i^\alpha = \frac{\exp(\alpha_i)}{\sum_k \exp(\alpha_k)}, \quad \pi_j^\beta = \frac{\exp(\beta_j)}{\sum_k \exp(\beta_k)}. \tag{11}$$

This leads to the architecture of Figure 1. The complexity measure of (9) is finally defined as

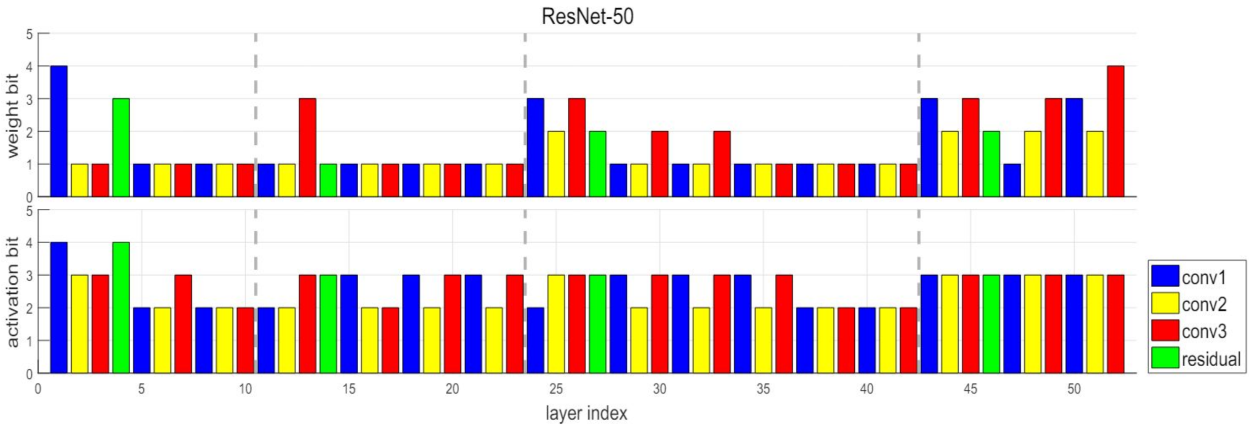
$$c(f) = E[b_f]E[b_a]|f|w_x h_x/s^2, \tag{12}$$

where

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

Cai, Zhaowei, and Nuno Vasconcelos. "Rethinking differentiable search for mixed-precision neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

EdMIPS



Cai, Zhaowei, and Nuno Vasconcelos. "Rethinking differentiable search for mixed-precision neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

EdMIPS

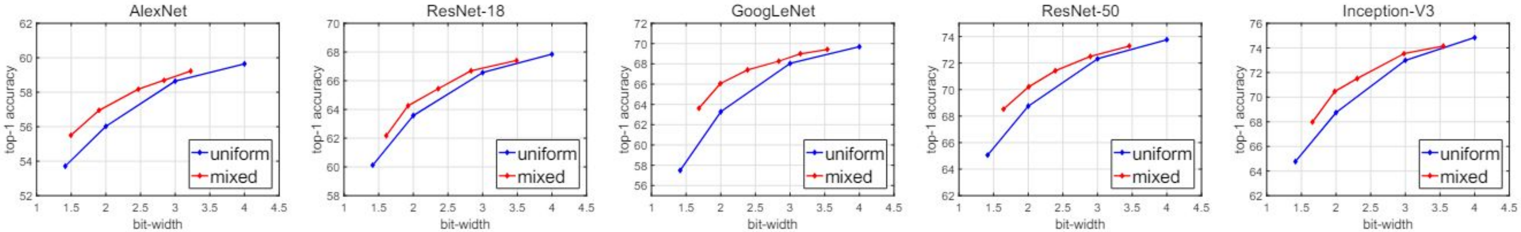


Figure 5. Comparison of the uniform HWGQ-Net and the EdMIPS network. The x-axis, indicating BitOps, is normalized to the scale of bit-width, which is actually in log-scale.

Cai, Zhaowei, and Nuno Vasconcelos. "Rethinking differentiable search for mixed-precision neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

>>> 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.

Shvetsov Egor, Dmitry Osin et al. "QuantNAS for super resolution: searching for efficient quantization-friendly architectures against quantization noise." arXiv preprint arXiv:2208.14839 (2022).

QuantNAS

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

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

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

Shvetsov Egor, Dmitry Osin et al. "QuantNAS for super resolution: searching for efficient quantization-friendly architectures against quantization noise." arXiv preprint arXiv:2208.14839 (2022).

QuantNAS

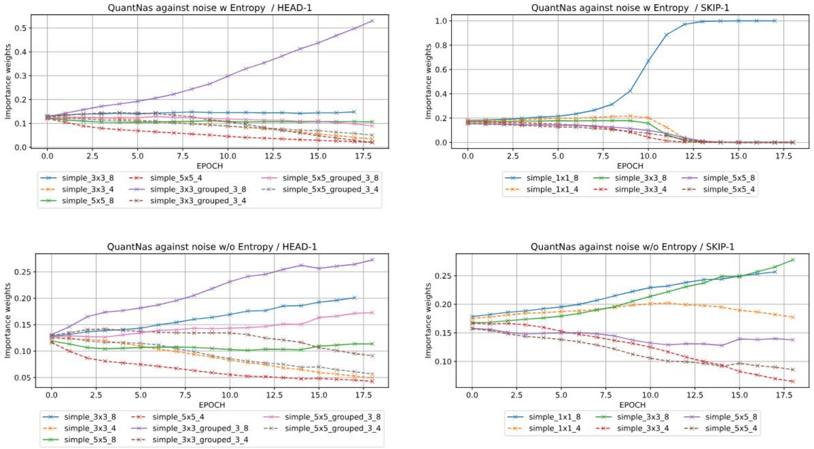
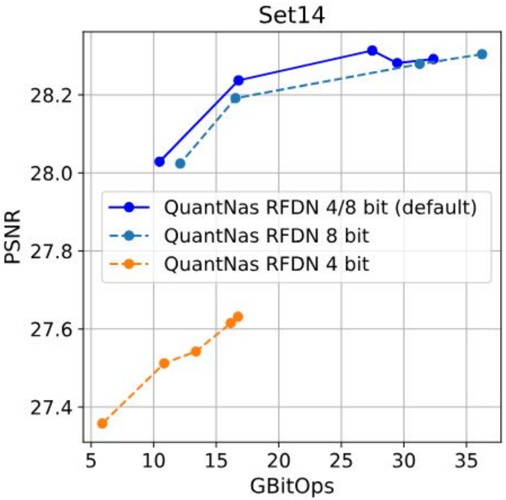
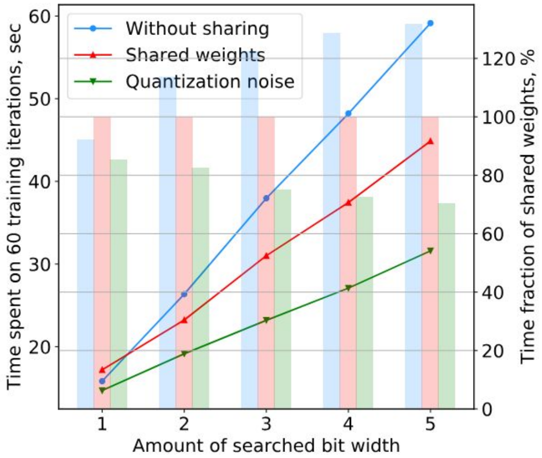


Fig. 9. Dynamics of importance weights for different operations through epochs for QuantNAS. For 8 and 4 bits, we use solid and dashed lines, respectively. Usage of entropy sparsification (top) allows for selecting a single most relevant block with high importance c.t. variants without entropy sparsification (bottom).

Shvetsov Egor, Dmitry Osin et al. "QuantNAS for super resolution: searching for efficient quantization-friendly architectures against quantization noise." arXiv preprint arXiv:2208.14839 (2022).

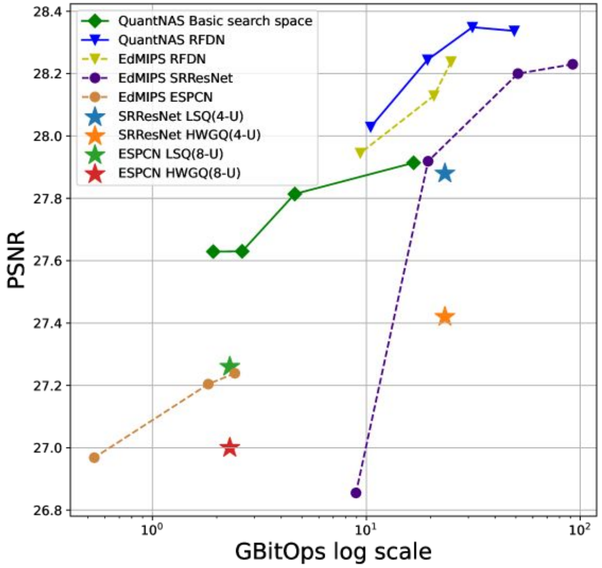
QuantNAS





Shvetsov Egor, Dmitry Osin et al. "QuantNAS for super resolution: searching for efficient quantization-friendly architectures against quantization noise." arXiv preprint arXiv:2208.14839 (2022).

QuantNAS



Shvetsov Egor, Dmitry Osin et al. "QuantNAS for super resolution: searching for efficient quantization-friendly architectures against quantization noise." arXiv preprint arXiv:2208.14839 (2022).

>>> 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.

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

SeqNAS

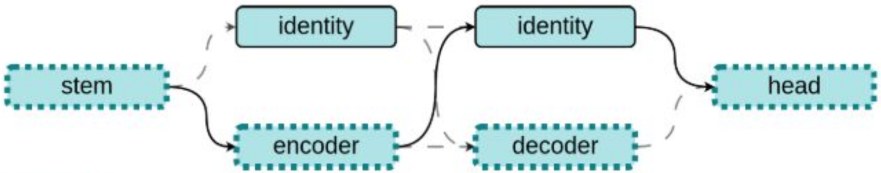


FIGURE 2. **The general layout of our search space. Dotted borders indicate that blocks contain searchable operations. Dashed lines indicate that connections between nodes are searchable. The solid line is an example of selected architecture.**

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

SeqNAS

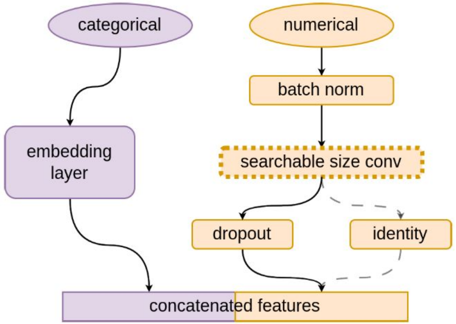


FIGURE 3. **Searchable part of Stem block is depicted with dashed and dotted lines. Convolutional layers with different kernels and the presence of dropout are selected at each search step. A solid line is an example of a selected path.**

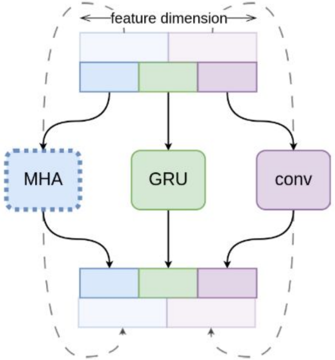


FIGURE 5. **Encoder Layer with searchable MHA, GRU and conv operations. A combination of one, two, or three operations can be selected during each search step. Different combinations are selected on different layers. Incoming features are divided into several selected operations. An example combination with MHA, GRU and conv operations is depicted with solid lines, and an example combination with MHA and conv operations is depicted with dashed lines. Dotted border around MHA indicates that it has a searchable number of heads.**

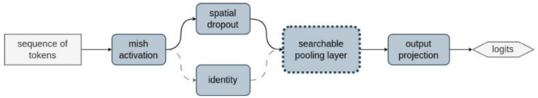


FIGURE 4. **There are two searchable pooling layers in Head Block: Max pooling and Average pooling. The type of a pooling layer and the presence or absence of spatial dropout are determined by the search procedure. A solid line is an example of a selected path.**

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

Algorithm 1 *Predictor − model* — Model Score Predictor With Parameters θ, N_{init} — Initial Number of Architectures to Train, N_{iter} — Number of Architectures to Sample During Each Iteration, $L_{candidates}$ — Number of Architectures to Train During Each Iteration, *Ensemble* — Ensmbling Function, M — Number of Iterations, $\hat{X}, S$ — Predicted Scores and Corresponding Uncertainties

$K_1 \leftarrow Sample(N_{init})$, sample random architectures from the search space.
$TrainedArches \leftarrow Train(K_1)$. Train all architectures in K_1 and obtain their scores, X is a set of scores from all trained models, X_i are scores for a current iteration.
$ArchFeatures \leftarrow AVec(TrainedArches)$, encode architectures into features.
$\min_{\theta}(MAE(Predictor - model(ArchFeatures; \theta), X))$, train a score predictor model.
for $i = 1, 2, \ldots, M$ **do**
  $K_{1+i} \leftarrow Sample(N_{iter})$, sample random architectures from the search space such that $K_{i+1} \cap TrainedArches = \emptyset$.
  $\hat{X}, S = Predictor - model(K_{i+1}; \theta)$, predict scores and score uncertainties.
  Select $L_{candidates}$ architectures from K_{1+i} with **Thomson sampling** using obtained uncertainties S.
  $T \leftarrow topK(TrainedArches)$, select the best performing teacher models from already trained models and obtain an ensemble of teachers $Ensemble(T)$.
  Train all models in $L_{candidates}$ with distillation loss and $Ensemble(T)$ and obtain actual scores X_i.
  $TrainedArches \leftarrow TrainedArches \cup L_{candidates}$.
  $X \leftarrow X \cup X_i$.
  $ArchFeatures = AVec(TrainedArches)$, encode architectures into features.
  Update a score predictor model $\theta \leftarrow \arg\min_{\theta} L(\theta)$, where $L(\theta) = MAE(Predictor - model(ArchFeatures; \theta), X)$.
end for
Select the best architecture from *TrainedArches* according to some performance metric.

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

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%

Спасибо за внимание!