Designing network design spaces

Докладчик: Набатова Дарья
Рецензент-исследователь: Кокорина Юлия
Хакер: Воробьев Николай

Подбор архитектуры нейронной сети

- Ручной подбор
- NAS

NAS

- **Пространство поиска** - определение набора операций (свертка, pooling) - определяется вручную
- **Алгоритм поиска** - выбор дочерних сетей
- **Стратегия оценки -** способ оценить производительность модели

Предлагаемый способ

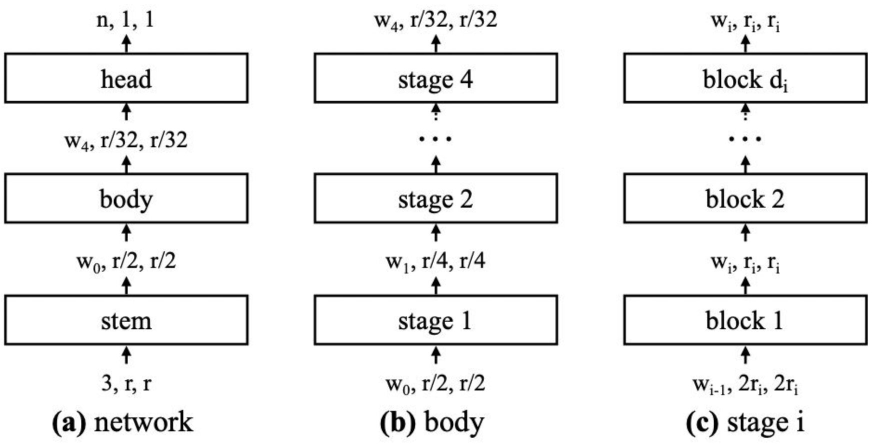
Стремимся найти не конкретную нейронную сеть, а пространство, параметризующее несколько нейронных сетей

Сети должны быть простыми и хорошо работать, как и при ручном проектировании, но при этом используются полуавтоматические процедуры из NAS.

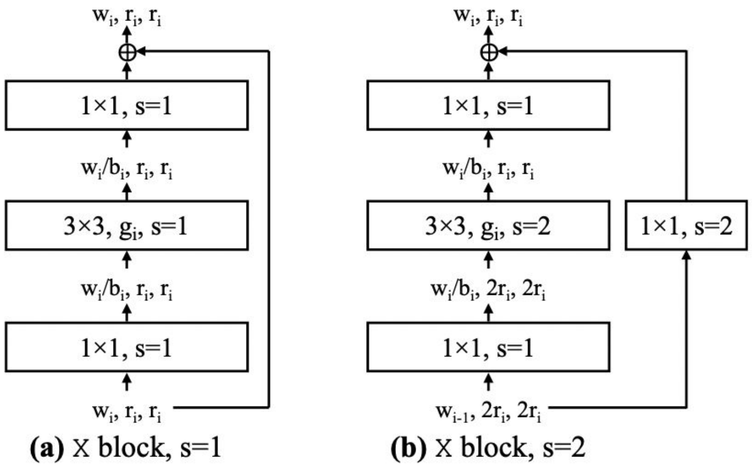
Сравнение проектируемых пространств

- выбираем и обучаем n моделей, строим их распределение
- строим EDF ошибок для обобщения качества пространства проектирования

AnyNet



General network structure



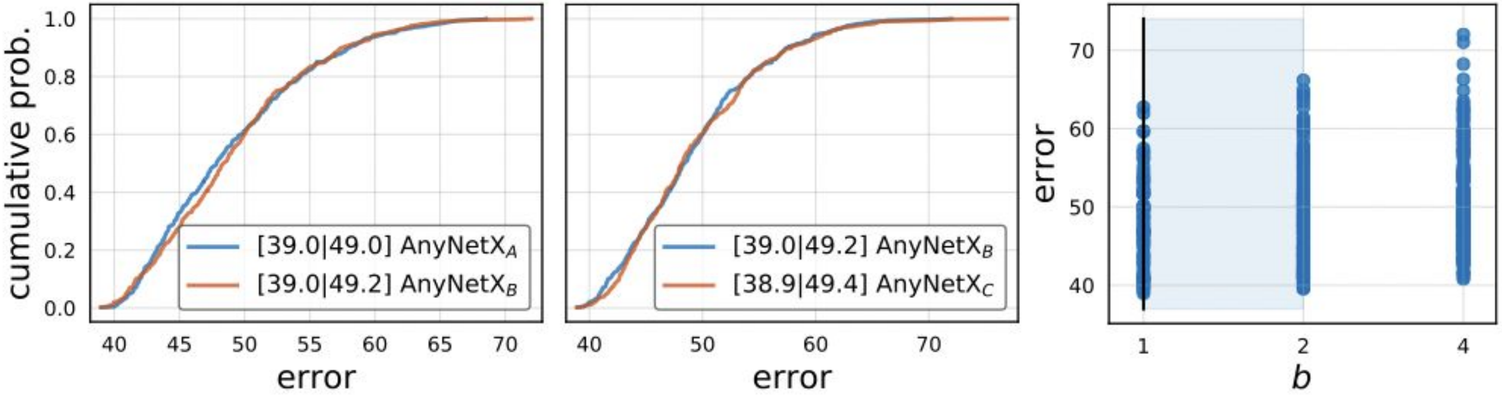
Block structure

Цели

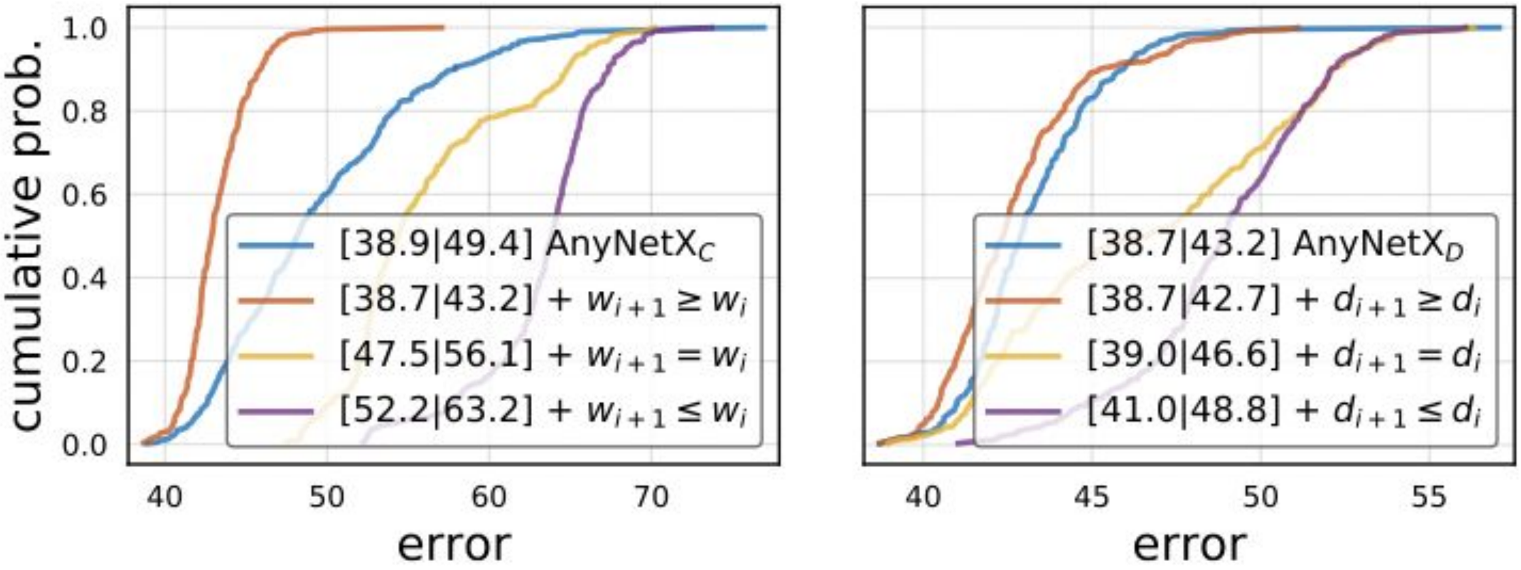
1. Упростить структуру пространства
2. Улучшить интерпретируемость пространства
3. Улучшить или сохранить качество
4. Сохранить разнообразие моделей

Алгоритм

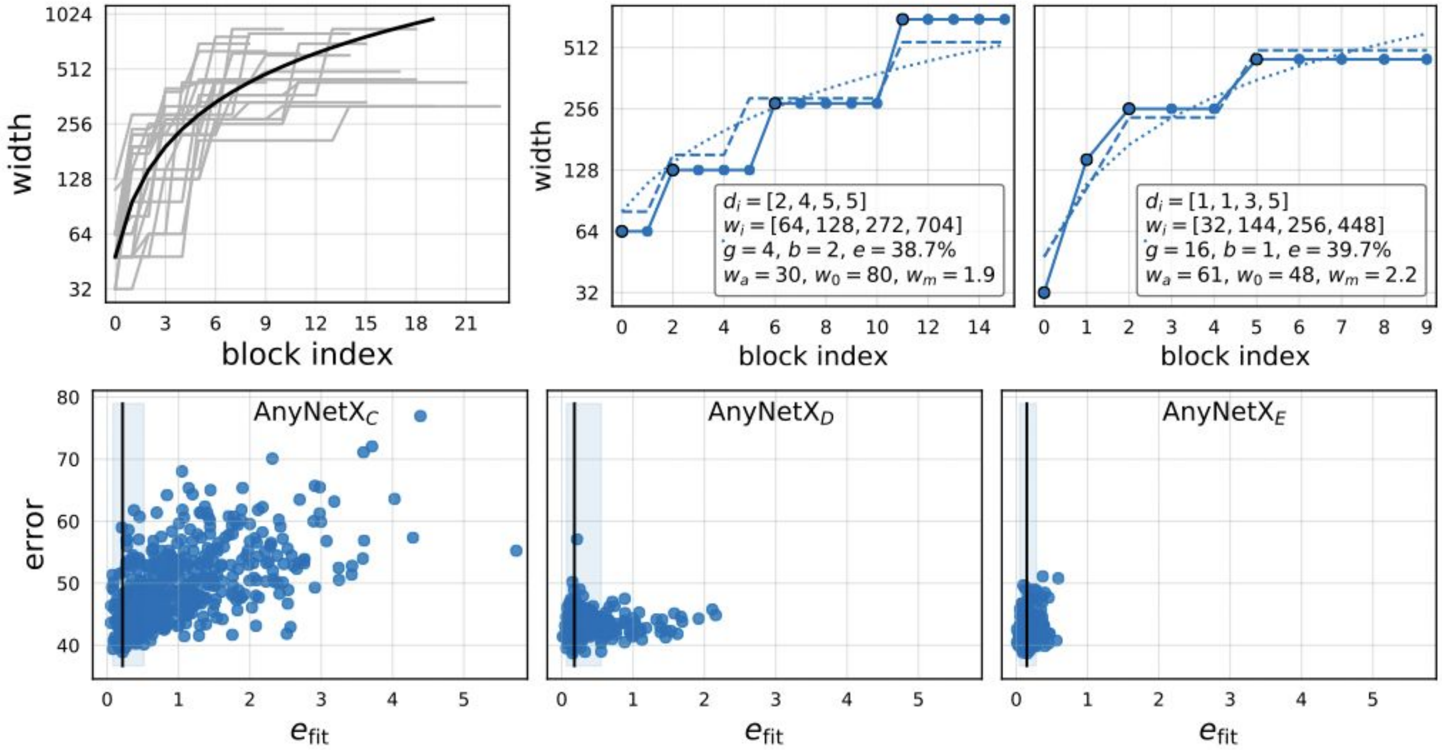
1. AnyNetXa = AnyNetX
2. AnyNetXb = AnyNetXa with bottleneck ratio bi = b
3. AnyNetXc = AnyNetXb with group width gi = g
4. AnyNetXd = AnyNetXc with increasing widths wi
5. AnyNetXe = AnyNetXd with increasing stage depths di



cumulative prob.
1.0
0.8
0.6
0.4
0.2
0.0
40
45
50
55
60
65
70
error
[39.0|49.0] AnyNetX$_A$
[39.0|49.2] AnyNetX$_B$
40
50
60
70
error
[39.0|49.2] AnyNetX$_B$
[38.9|49.4] AnyNetX$_C$
error
70
60
50
40
1
2
4
b



cumulative prob.
1.0
0.8
0.6
0.4
0.2
0.0
40
50
60
70
error
[38.9|49.4] AnyNetX$_C$
[38.7|43.2] + $w_{i+1} \geq w_i$
[47.5|56.1] + $w_{i+1} = w_i$
[52.2|63.2] + $w_{i+1} \leq w_i$
40
45
50
55
error
[38.7|43.2] AnyNetX$_D$
[38.7|42.7] + $d_{i+1} \geq d_i$
[39.0|46.6] + $d_{i+1} = d_i$
[41.0|48.8] + $d_{i+1} \leq d_i$



width
block index
$d_i = [2, 4, 5, 5]$
$w_i = [64, 128, 272, 704]$
$g = 4$, $b = 2$, $e = 38.7\%$
$w_a = 30$, $w_0 = 80$, $w_m = 1.9$
$d_i = [1, 1, 3, 5]$
$w_i = [32, 144, 256, 448]$
$g = 16$, $b = 1$, $e = 39.7\%$
$w_a = 61$, $w_0 = 48$, $w_m = 2.2$
error
AnyNetX$_C$
AnyNetX$_D$
AnyNetX$_E$
e_{fit}

RegNet

Линейная параметризация ширины блоков:

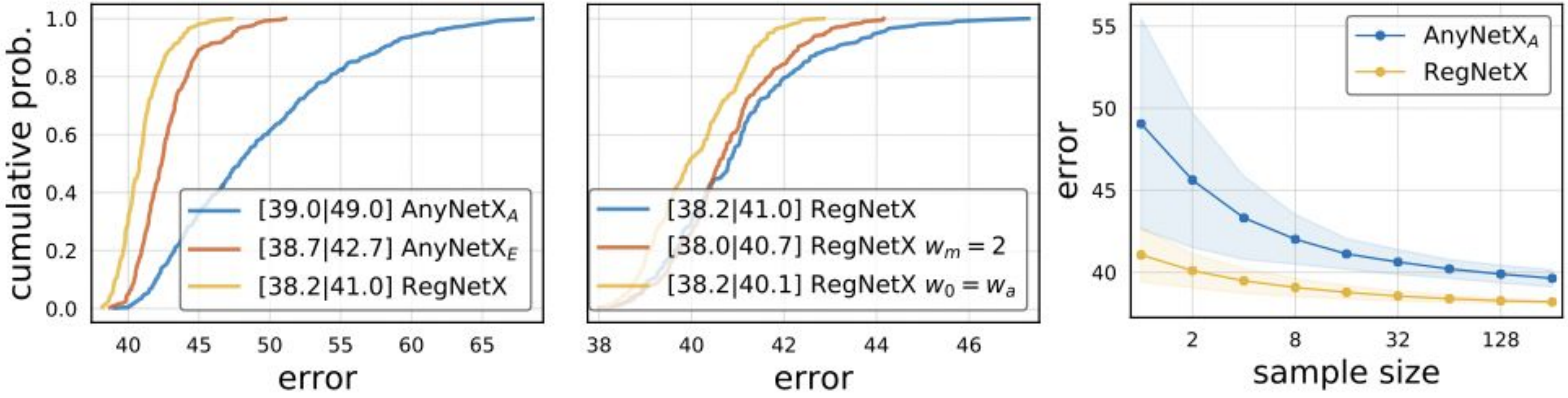
$$u_j = w_0 + w_a \cdot j \quad \text{for} \quad 0 \leq j < d$$

$$u_j = w_0 \cdot w_m^{s_j}$$

$$w_j = w_0 \cdot w_m^{\lfloor s_j \rceil}$$

Задаем структуру сети через параметры d, b, g, w0, wa, wm и уравнения выше.

Получившееся пространство назовем RegNet



cumulative prob.
error
[39.0|49.0] AnyNetX$_A$
[38.7|42.7] AnyNetX$_E$
[38.2|41.0] RegNetX
[38.2|41.0] RegNetX
[38.0|40.7] RegNetX $w_m = 2$
[38.2|40.1] RegNetX $w_0 = w_a$
AnyNetX$_A$
RegNetX
sample size

	restriction	dim.	combinations	total
AnyNetX$_A$	none	16	$(16{\cdot}128{\cdot}3{\cdot}6)^4$	$\sim 1.8{\cdot}10^{18}$
AnyNetX$_B$	$+\, b_{i+1} = b_i$	13	$(16{\cdot}128{\cdot}6)^4{\cdot}3$	$\sim 6.8{\cdot}10^{16}$
AnyNetX$_C$	$+\, g_{i+1} = g_i$	10	$(16{\cdot}128)^4{\cdot}3{\cdot}6$	$\sim 3.2{\cdot}10^{14}$
AnyNetX$_D$	$+\, w_{i+1} \geq w_i$	10	$(16{\cdot}128)^4{\cdot}3{\cdot}6/(4!)$	$\sim 1.3{\cdot}10^{13}$
AnyNetX$_E$	$+\, d_{i+1} \geq d_i$	10	$(16{\cdot}128)^4{\cdot}3{\cdot}6/(4!)^2$	$\sim 5.5{\cdot}10^{11}$
RegNet	quantized linear	6	$\sim 64^4{\cdot}6{\cdot}3$	$\sim 3.0{\cdot}10^8$

Результаты

- При усложнении структур и количества эпох для обучения, RegNet показывает лучшие результаты и не переобучается, т.е. RegNet может обобщать новые настройки
- Количество активаций увеличивается пропорционально квадратному корню из флопов
- Количество параметров увеличивается линейно

Mobile regine

	flops (B)	params (M)	top-1 error
MOBILENET [9]	0.57	4.2	29.4
MOBILENET-V2 [25]	0.59	6.9	25.3
SHUFFLENET [33]	0.52	-	26.3
SHUFFLENET-V2 [19]	0.59	-	25.1
NASNET-A [35]	0.56	5.3	26.0
AMOEBANET-C [23]	0.57	6.4	24.3
PNASNET-5 [17]	0.59	5.1	25.8
DARTS [18]	0.57	4.7	26.7
REGNETX-600MF	0.60	6.2	25.9$\pm$0.03
REGNETY-600MF	0.60	6.1	24.5$\pm$0.07

Full regine

	flops (B)	params (M)	acts (M)	batch size	infer (ms)	train (hr)	top-1 error ours±std [orig]
EFFICIENTNET-B0	0.4	5.3	6.7	256	34	11.7	**24.9**±0.03 [23.7]
REGNETY-400MF	0.4	4.3	**3.9**	1024	19	5.1	25.9±0.16
EFFICIENTNET-B1	0.7	7.8	10.9	256	52	15.6	**24.1**±0.16 [21.2]
REGNETY-600MF	0.6	6.1	**4.3**	1024	19	5.2	24.5±0.07
EFFICIENTNET-B2	1.0	9.2	13.8	256	68	18.4	**23.4**±0.06 [20.2]
REGNETY-800MF	0.8	6.3	**5.2**	1024	22	6.0	23.7±0.03
EFFICIENTNET-B3	1.8	12.0	23.8	256	114	32.1	22.5±0.05 [18.9]
REGNETY-1.6GF	1.6	11.2	**8.0**	1024	39	10.1	**22.0**±0.08
EFFICIENTNET-B4	4.2	19.0	48.5	128	240	65.1	21.2±0.06 [17.4]
REGNETY-4.0GF	4.0	20.6	**12.3**	512	68	16.8	**20.6**±0.08
EFFICIENTNET-B5	9.9	30.0	98.9	64	504	135.1	21.5±0.11 [16.7]
REGNETY-8.0GF	8.0	39.2	**18.0**	512	113	28.1	**20.1**±0.09

Рецензия

Кто авторы?

- Facebook AI Research
- Ilija Radosavovic is a PhD Student, UC Berkeley
- Raj Prateek Kosaraju is a Research Engineer at Facebook AI Research, focusing on representation learning, computer vision, and deep learning
- Ross Girshick, Kaiming He
- Piotr Dollar is a research director at Facebook AI Research (FAIR) since 2014 with a focus on deep learning and computer vision. Current deeper interests include recognition, grouping, network design, and self-supervised learning.

Предыстория

On Network Design Spaces for Visual Recognition

ICCV 2019 · Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, Piotr Dollár · Edit social preview

Over the past several years progress in designing better neural network architectures for visual recognition has been substantial. To help sustain this rate of progress, in this work we propose to reexamine the methodology for comparing network architectures. In particular, we introduce a new comparison paradigm of distribution estimates, in which network design spaces are compared by applying statistical techniques to populations of sampled models, while controlling for confounding factors like network complexity. Compared to current methodologies of comparing point and curve estimates of model families, distribution estimates paint a more complete picture of the entire design landscape. As a case study, we examine design spaces used in neural architecture search (NAS). We find significant statistical differences between recent NAS design space variants that have been largely overlooked. Furthermore, our analysis reveals that the design spaces for standard model families like ResNeXt can be comparable to the more complex ones used in recent NAS work. We hope these insights into distribution analysis will enable more robust progress toward discovering better networks for visual recognition.

Продолжение

Fast and Accurate Model Scaling

CVPR 2021 · Piotr Dollár, Mannat Singh, Ross Girshick · Edit social preview

In this work we analyze strategies for convolutional neural network scaling; that is, the process of scaling a base convolutional network to endow it with greater computational complexity and consequently representational power. Example scaling strategies may include increasing model width, depth, resolution, etc. While various scaling strategies exist, their tradeoffs are not fully understood. Existing analysis typically focuses on the interplay of accuracy and flops (floating point operations). Yet, as we demonstrate, various scaling strategies affect model parameters, activations, and consequently actual runtime quite differently. In our experiments we show the surprising result that numerous scaling strategies yield networks with similar accuracy but with widely varying properties. This leads us to propose a simple fast compound scaling strategy that encourages primarily scaling model width, while scaling depth and resolution to a lesser extent. Unlike currently popular scaling strategies, which result in about $O(s)$ increase in model activation w.r.t. scaling flops by a factor of s, the proposed fast compound scaling results in close to $O(\sqrt{s})$ increase in activations, while achieving excellent accuracy. This leads to comparable speedups on modern memory-limited hardware (e.g., GPU, TPU). More generally, we hope this work provides a framework for analyzing and selecting scaling strategies under various computational constraints.

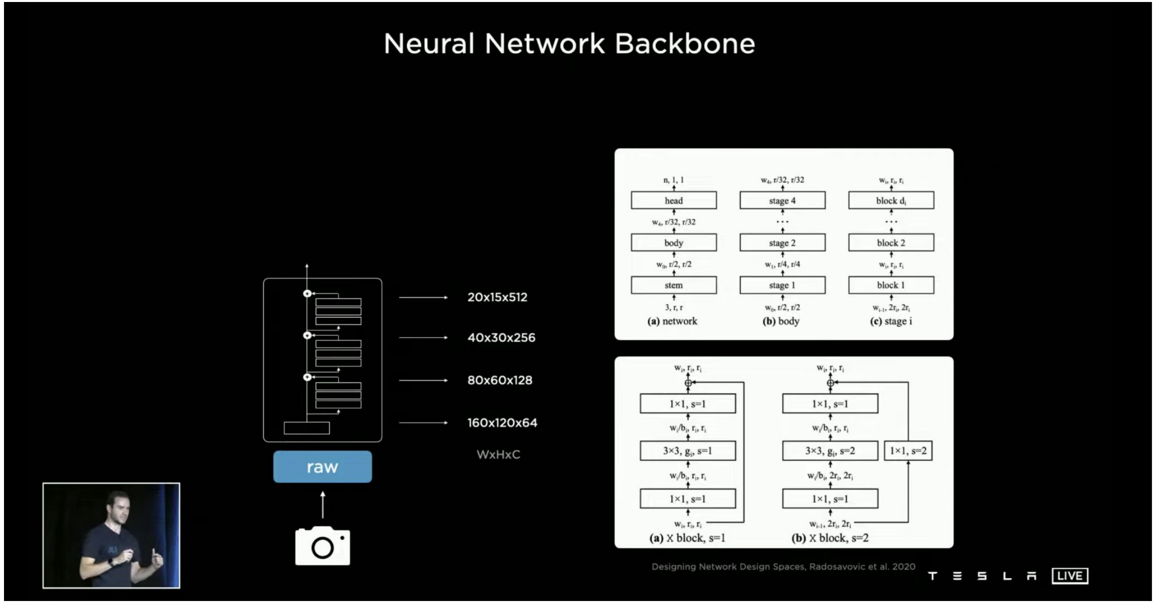
Продолжение с трансформерами

Searching the Search Space of Vision Transformer

NeurIPS 2021 · Minghao Chen, Kan Wu, Bolin Ni, Houwen Peng, Bei Liu, Jianlong Fu, Hongyang Chao, Haibin Ling · Edit social preview

Vision Transformer has shown great visual representation power in substantial vision tasks such as recognition and detection, and thus been attracting fast-growing efforts on manually designing more effective architectures. In this paper, we propose to use neural architecture search to automate this process, by searching not only the architecture but also the search space. The central idea is to gradually evolve different search dimensions guided by their E-T Error computed using a weight-sharing supernet. Moreover, we provide design guidelines of general vision transformers with extensive analysis according to the space searching process, which could promote the understanding of vision transformer. Remarkably, the searched models, named S3 (short for Searching the Search Space), from the searched space achieve superior performance to recently proposed models, such as Swin, DeiT and ViT, when evaluated on ImageNet. The effectiveness of S3 is also illustrated on object detection, semantic segmentation and visual question answering, demonstrating its generality to downstream vision and vision-language tasks. Code and models will be available at https://github.com/microsoft/Cream.

RegNet в Tesla



https://saneryee-studio.medium.com/deep-understanding-tesla-fsd-part-1-hydranet-1b46106d57

Плюсы и минусы

- \+ **Новая идея** дизайна архитектур нейросетей
- \+ **Пространство** сетей, показывающих хорошие результаты
- \+ Наличие **сравнения** с другими моделями в разных режимах(mobile, full)
- \+ Код в **открытом** доступе
- \+ Известные случаи применения в **индустрии**
- \+ Большое количество **цитирований**(более 680)

\- Модели класса RegNet уже **не** такие **конкурентоспособные**