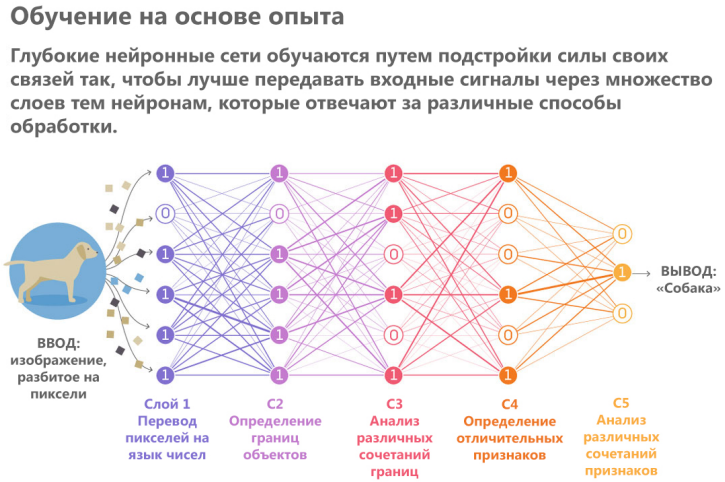
NAS: Neural Architecture Search

Анищенко Илья

Март 2020

Содержание

- Модель поиска нейронных архитектур - NAS
- Эксперимент с поиском CNN для набора картинок CIFAR-10
- Улучшение поиска нейронных архитектур - ENAS
- Два подхода к составлению CNN в улучшенном методе
- Эксперимент с методом ENAS на наборе картинок CIFAR-10
- Вопросы



Постановка проблемы

Нейронные сети хорошо показывают себя в задачах обработки и классификации изображений, обработки языка. Но при этом конструирование оптимальных нейросетей остается непростой задачей, требующей достаточного кол-ва времени на обучение большого множества моделей, в попытках найти самую оптимальную

Идея решения проблемы

Создать алгоритм, который сможет описывать архитектуру нейросети для конкретной задачи машинного обучения

Концепция поиска нейронной архитектуры

Схема работы

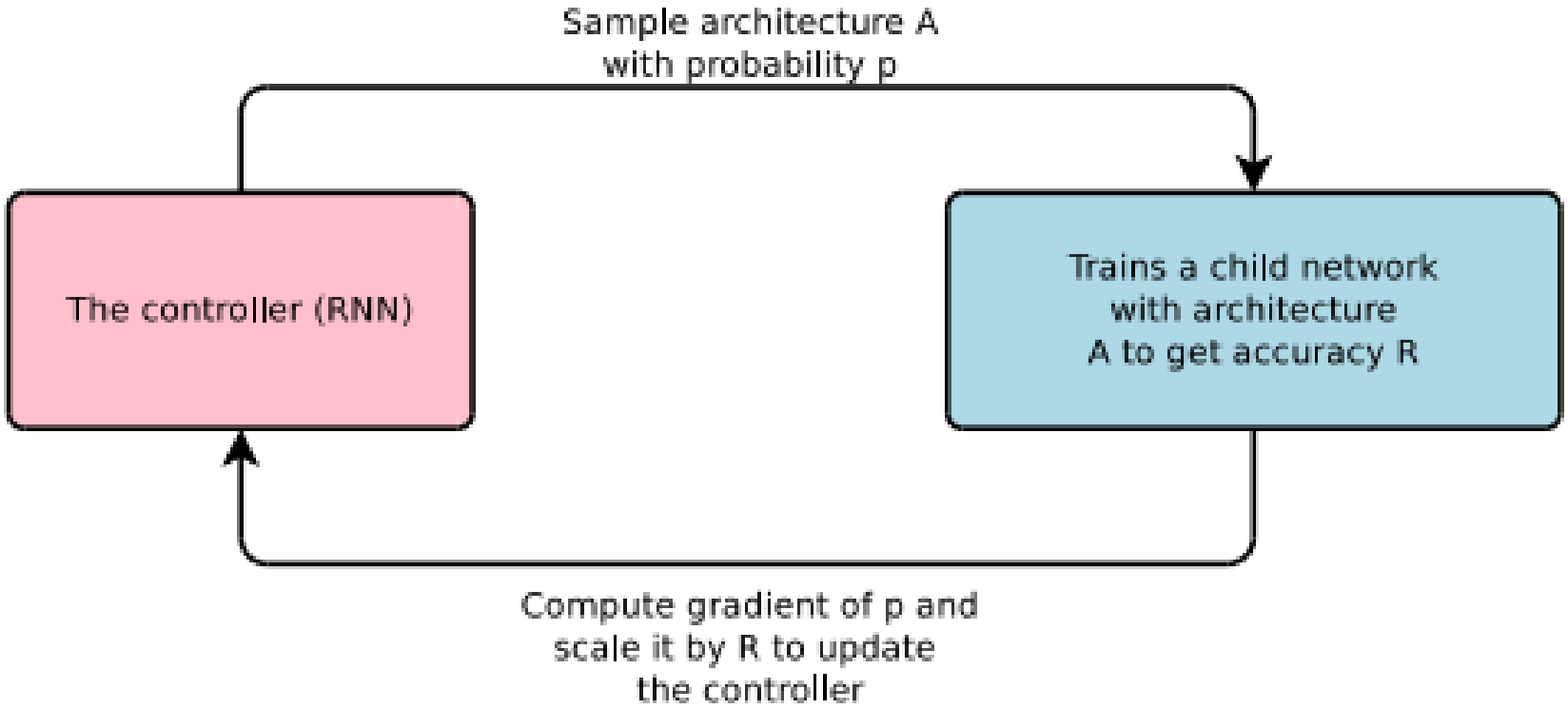


Figure 1: An overview of Neural Architecture Search.

Концепция поиска нейронной архитектуры

Neural Architecture Search

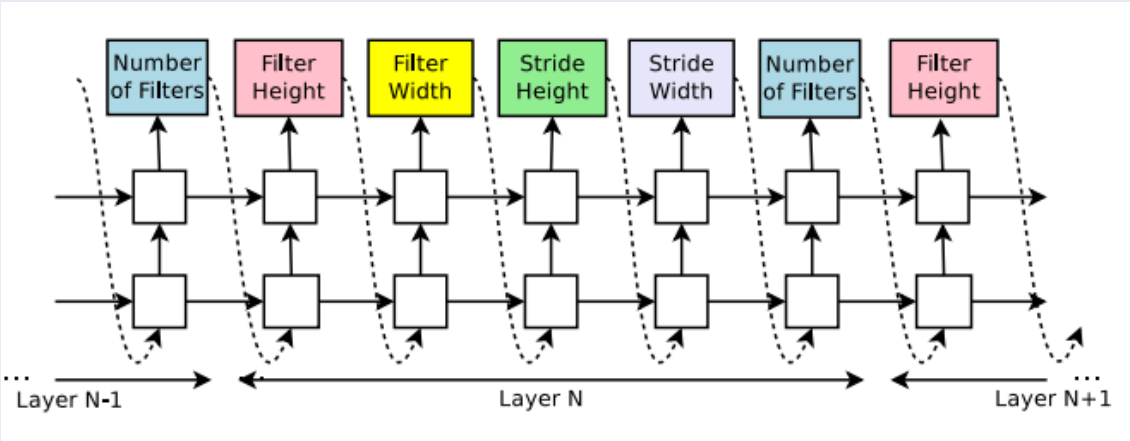
- Архитектура нейросети может быть записана как строка произвольной длины
- Можно взять RNN (controller), каждый блок которого будет отвечать за определенный параметр искомой архитектуры(кол-во фильтров в слое сверточной нейросети, с какими слоями соединен текущий слой (skip connections))
- Обучаем controller с помощбю RL, считая что вознаграждение это точность полученной архитектуры на валидационной выборке

Концепция поиска нейронной архитектуры

попытка объяснить это через мем

Мем

Генерация описания дочерних CNN с использованием RNN controller



- RNN контроллер - LSTM для необходимости учитывания долговременных зависимостей
- каждый выход - с классификатором softmax, после выход подается на след временной шаг
- каждый блок предсказывает характеристики конкретно слоя (в данном примере: ширина, высота фильтров, ширина и высота шага, кол-во фильтров)

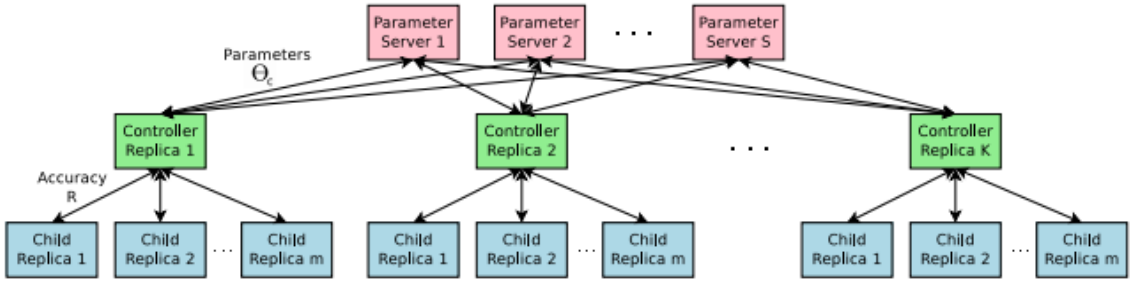
Обучение с подкреплением

- Характеристики, которые предсказывает контроллер, можно рассматривать как список действий $a_{1:T}$, для получения архитектуры сверточной сети.
- Точность полученной сети на тестовых данных будем понимать как полученное вознаграждение.
- Используя RL и полученные значения наград будем использовать для обучения контроллера
- А конкретно - будем его заставлять максимизировать ожидаемое вознаграждение $J(\theta_c)$: $E_{P(a_{1:T},\theta_c)}[R]$

Обучение с подкреплением

- Поскольку R не дифференцируема, мы переходим к policy gradient method с итеративным изменением θ_c и используем REINFORCE: $\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}[\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R]$
- через эмпирическое приближение получим следующий градиент функционала: $\nabla_{\theta_c} J(\theta_c) = \frac{1}{m} \sum^{(} m)_{(} k = 1) \nabla_{\theta_c} \log P(a_t | a_{(t-1)}; \theta_c)(R_k$ - b), где
- m - размер батча, созданных контроллером архитектур
- R_k - вознаграждение, которое получено от k-ой архитектуры
- T - кол-во гиперпараметров, которые контроллер должен определить
- P - вероятностная модель, которую моделирует контроллер
- b - бейзлайн, который был добавлен для уменьшения дисперсии оценки градиента, его значение не зависит от текущего действия и является скользящим средним по точностям предыдущих архитектур.

Распараллеливание обучения сверточных нейросетей

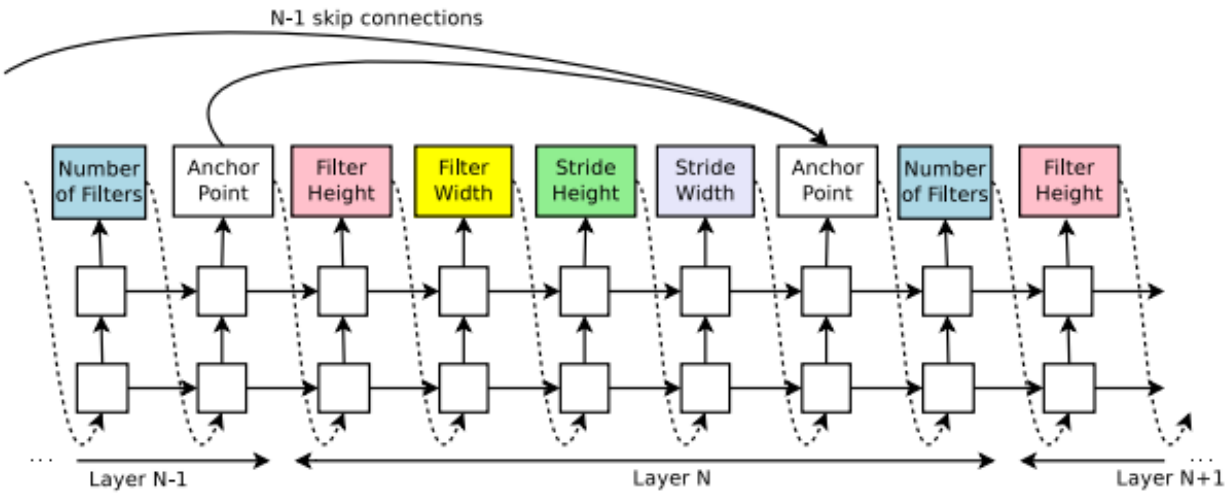


- При обучении m сверточных моделей от одного контроллера появляется проблема в необходимости большого кол-ва времени на подсчет всех этих моделей, поэтому было придумано следующее решение - распределенное обучение и рассинхронное обновление параметров
- У нас имеется S серверов для общих параметров всех K копий контроллеров
- каждая копия контроллера запускает на параллельное обучение m сверточных нейросетей
- Дале подсчитанный градиент функционала от результатов m архитектур он отправляет на сервера для изменения параметров во всех копиях контроллера

Усложнение дочерних CNN

- В рассмотренном нами ранее способе генерации сверточных сетей отсутствуют пропускные соединения слоев (skip connection), которые используются в современных архитектурах и вносят свой полезнвый вклад.
- Для их реализации в нашей моделе описания CNN мы добавляем некоторые точки привязки на блок описания каждого слоя N, которые содержат в себе сигмоиду. Она является функцией от текущего скрытого состояния контроллера и всех предыдущих скрытых состояний с предыдущих $N-1$ слоя.
- P(слой j будет входящим в слой i) = $\text{sigmoid}(v^T \tanh(W_{prev} h_j + W_{curr} h_i))$
- Матрицы W_{prev}, W_{curr}, v - обучаемые параметры

Усложнение дочерних CNN



- Теперь наша модель описания выглядит следующим образом
- Так же у нас теперь может приходить много слоев на вход к текущему слою N, в таком случаем мы их объединяем по размерности глубины.
- Для добавления различных слоев, помимо сверток, в блок можно добавить еще один временный шаг, в котором контроллер определяет вид текущего слоя, а после уже предсказывает его параметры

Поиск лучшей модели для классификации картинок из набора CIFAR-10

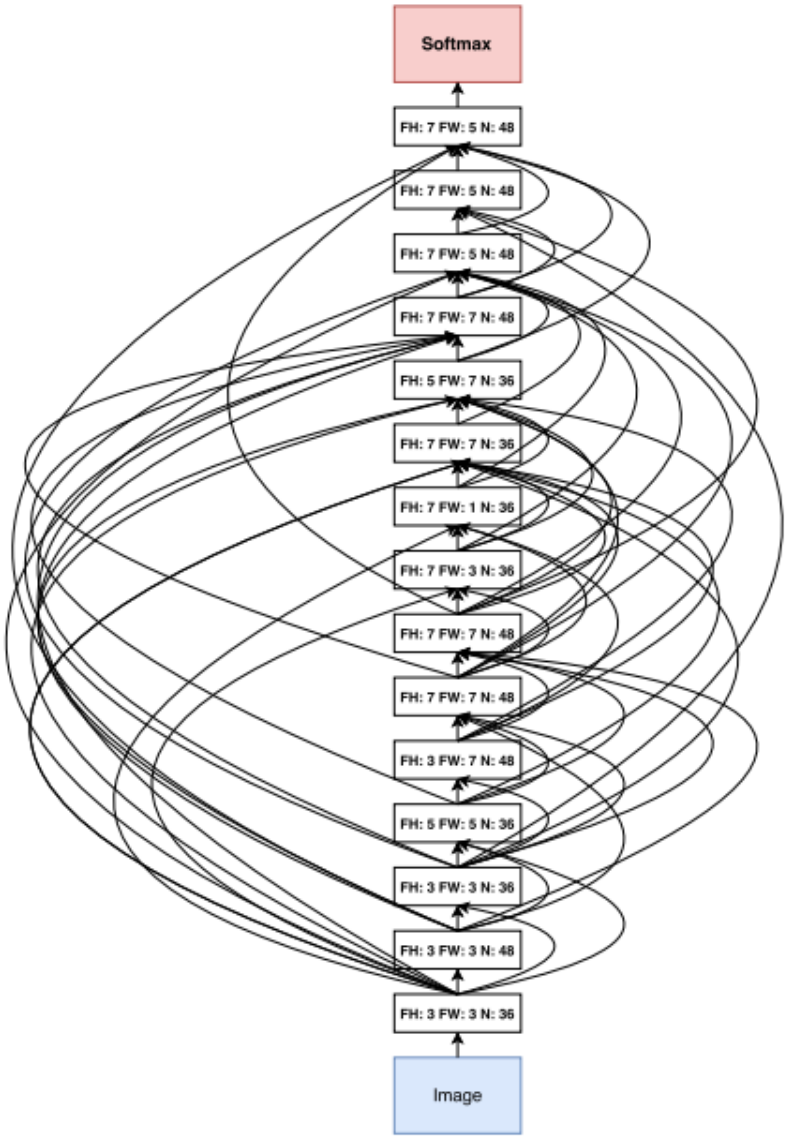
- Датасет: набор картинок CIFAR-10 с предобработкой, вырезанием с изображения случайного кадра размером 32x32, горизонтальные флипы
- Пространство поиска: сверточные слои, батч нормализация и пропуский соендинений между слоями.
- Детали обучения: контроллер - LSTM с 35 скрытыми слоями, для оптимизации был взят Adam. Для распараллеливания было взято 20 серверов под параметры.
- Главное - было необходимо 800 GPU, чтобы была возможность задействовать 100 копий контроллера, и каждый при этом обучал по 8 сверточных нейростей в батче.

Результаты

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.

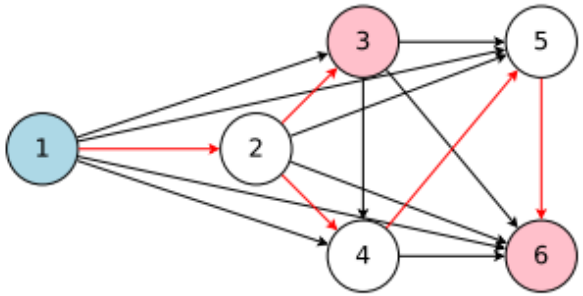
Найденная сеть из 15 слоев



ENAS - эффективный поиск нейронной архитектуры

- Основной минус самого метода поиска нейронной архитектуры - действительно большие ресурсы, необходимые для решения поставленной задачи (в эксперименте выше было задействовано 800 GPU).
- Поэтому позже были попытки оптимизировать данный алгоритм и сделать его менее ресурсно затратным
- Основной вектор в направлении улучшения - стараться избавиться от обучения дочерних нейросетей с нуля и до полной сходимости.

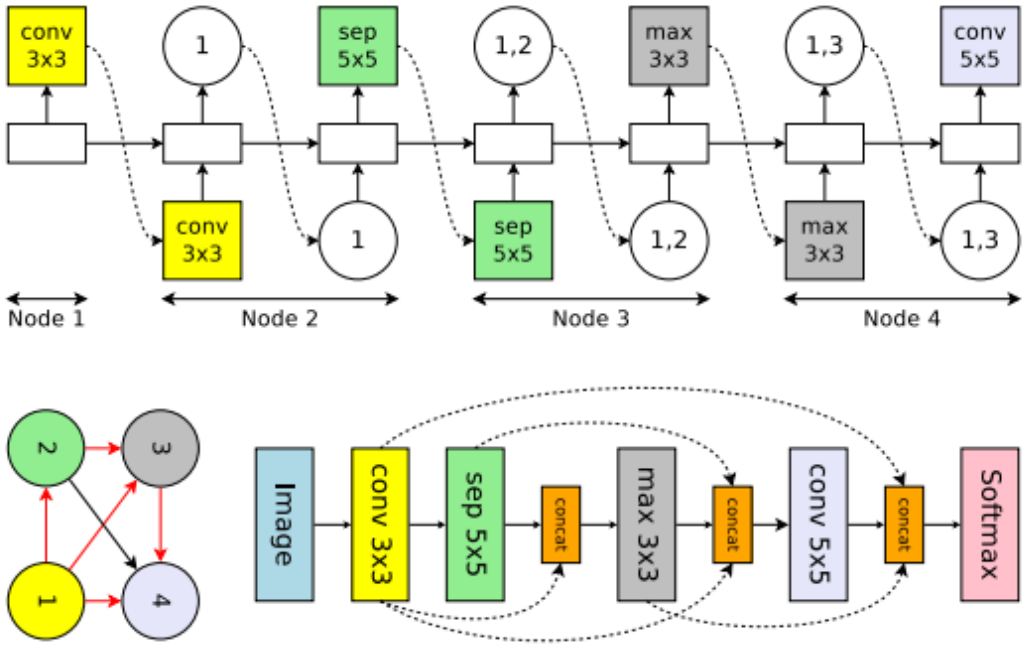
Направленные ациклические графы



- В данном методе все дочерние нейросети интерпретируются как подграфы более крупного графа. Другими словами, пространство поиска в данной задаче представляется как направленный ациклический граф
- узлы графа - локальные вычисления (один или некоторое множество слоев)
- ребра - поток перехода информации между частями модели.
- в данном методе помимо отпимизации параметров θ_c контроллера мы еще учитываем оптимизацию значений весов w каждой рассмотренноей, дочерней модели, так как на следующей итерации эти значения мы можем использовать.

Два подхода к построению CNN модели

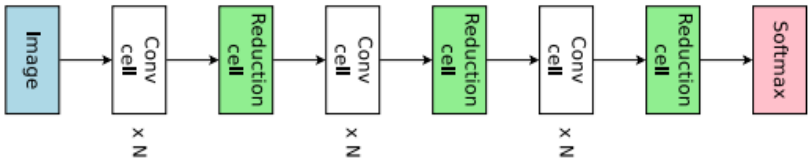
Macro search

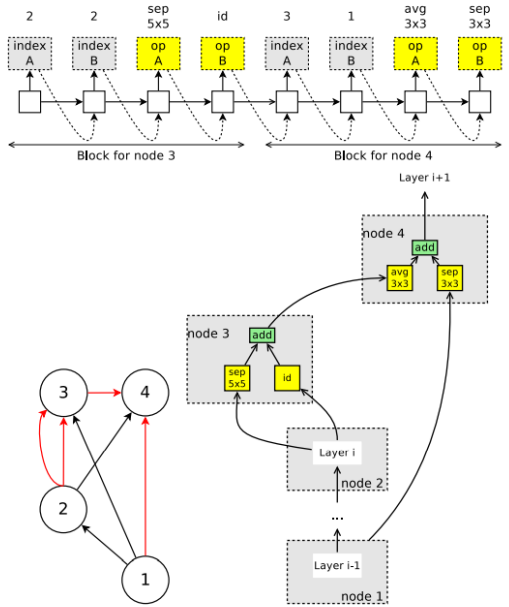


- Строит сеть стандартным образом, т.е. на каждом шаге контроллер делает два выбора:
- Какие из предыдущих слоев будут входными в новый слой
- какую операцию надо добавить к новому слою

Два подхода к построению CNN модели

Micro search





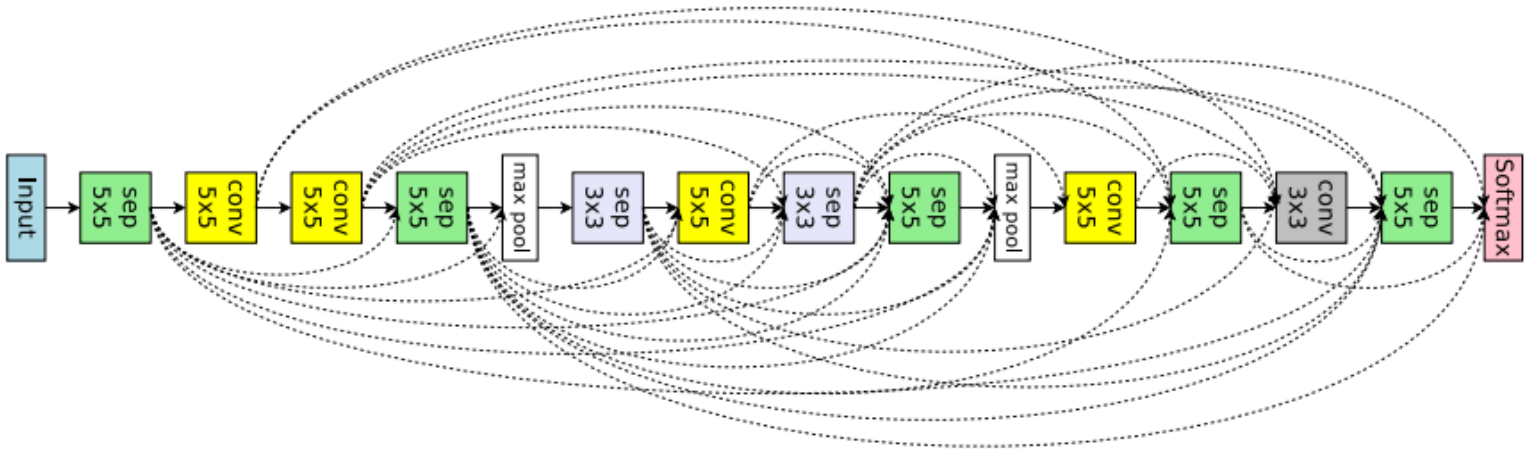
- Строит сеть блоками, где каждый блок: N ячеек со свертками и одна сжимающая ячейка
- Для каждого блока выбирается пара предыдущих блоков, которая будет входными данными в текущий блок,и пара операций, которые к ним применятся

Эксперимент с ENAS для поиска сверточной нейросети для классификации картинок CIFAR-10

Efficient Neural Architecture Search via Parameter Sharing

Method	GPUs	Times (days)	Params (million)	Error (%)
DenseNet-BC (Huang et al., 2016)	–	–	25.6	3.46
DenseNet + Shake-Shake (Gastaldi, 2016)	–	–	26.2	2.86
DenseNet + CutOut (DeVries & Taylor, 2017)	–	–	26.2	**2.56**
Budgeted Super Nets (Veniat & Denoyer, 2017)	–	–	–	9.21
ConvFabrics (Saxena & Verbeek, 2016)	–	–	21.2	7.43
Macro NAS + Q-Learning (Baker et al., 2017a)	10	8-10	11.2	6.92
Net Transformation (Cai et al., 2018)	5	2	19.7	5.70
FractalNet (Larsson et al., 2017)	–	–	38.6	4.60
SMASH (Brock et al., 2018)	1	1.5	16.0	4.03
NAS (Zoph & Le, 2017)	800	21-28	7.1	4.47
NAS + more filters (Zoph & Le, 2017)	800	21-28	37.4	**3.65**
ENAS + macro search space	1	0.32	21.3	4.23
ENAS + macro search space + more channels	1	0.32	38.0	**3.87**
Hierarchical NAS (Liu et al., 2018)	200	1.5	61.3	3.63
Micro NAS + Q-Learning (Zhong et al., 2018)	32	3	–	3.60
Progressive NAS (Liu et al., 2017)	100	1.5	3.2	3.63
NASNet-A (Zoph et al., 2018)	450	3-4	3.3	3.41
NASNet-A + CutOut (Zoph et al., 2018)	450	3-4	3.3	**2.65**
ENAS + micro search space	1	0.45	4.6	3.54
ENAS + micro search space + CutOut	1	0.45	4.6	**2.89**

Модель, найденная методом ENAS + macro search space



Вопросы

- Какую задачу решает NAS? Опишите такую архитектуру для решения задачи классификации картинок.
- В чем основное преимущество ENAS перед обычным поиском нейронной архитектуры?
- Выписать формулу gradient policy для обновления параметров θ_c
- Для чего используются ориентированные ациклические графы в ENAS?

Список литературы

1. Neural Architecture Search with Reinforcement Learning; Barret Zoph, Quoc V. Le
2. Efficient Neural Architecture Search via Parameter Sharing; Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, Jeff Dean