Развитие сверточных сетей. ResNeXt, Xception

Александр Боймель

23 декабря 2018 г.

Оглавление

Глава 1

Введение

Свертка изображения f с фильтром g представляется следующим выражением

$$(f * g)[m, n] = \sum_{k,l} f[m - k, n - l] \cdot g[k, l] \tag{1.1}$$

В сверточной нейронной сети делаются итеративные свертки изображения с различными фильтрами, параметры которых обучаемы (через механизм Back propogation). Можно показать, что с помощью сверточной нейронной сети из 2х слоев можно реализовать большинство эвристических методов вычисления признаков изображения (гистограммы цветов, HOG, мешки визуальных слов). Увеличивая число слоев можно выучить признаки более выского уровня.

Так как было установлено, что сверточные нейронные сети извлекают полезные и информативные признаки изображений, то их стали применять в различных задачах компьютерного зрения. За основу взяли задачу классификации (на датасете ImageNet). Для конкретной задачи берутся веса предобученной на классификацию сети для извлечения признаков, поверх чего достраивается требуемая в задаче архитектура. Поэтому получать высокое качество классификации и, хочется верить, полезные признаки важно для всех задач компьютерного зрения с использованием нейронных сетей.

Глава 2

Известные архитектуры

Для более простых архитектур можно обратиться к Лекция по CNN, miptcv

2.1 Inception

Был реализован метод *split-transform-merge*, идея которого в том, чтобы вместо выбора определенного преобразования (размера фильтра) комбинировать различные преобразования. Ниже приведены так называемые *Inception блоки*, реализующие данный принцип. Кроме того, для получения одинаковых размерностей из различных трансформаций используются свертки 1×1.

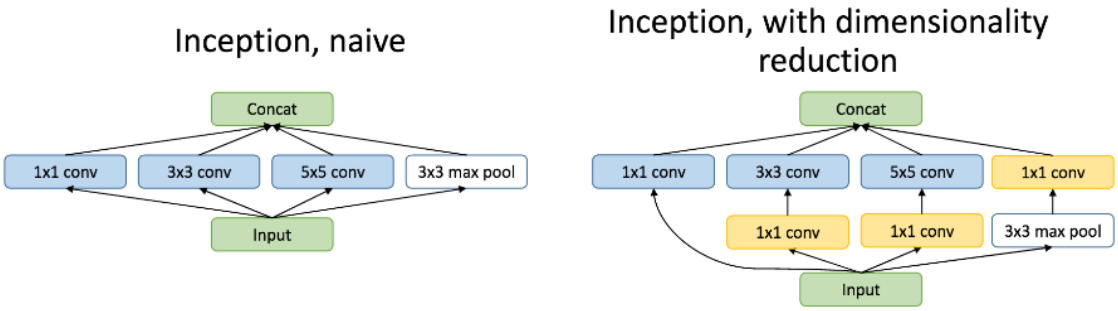


Рис. 2.1: Inception block

Объединение таких блоков дало сеть ниже. Кроме того, в этой сети добавлены дополнительные выходы-классификаторы для борьбы с затуханием градиента по мере роста глубины.

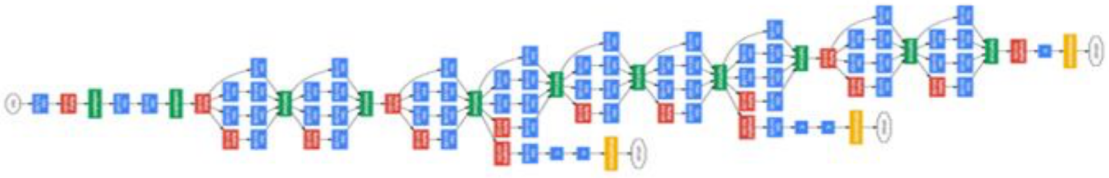


Рис. 2.2: GoogLeNet

2.2 ResNet

Появилась идея в том, чтобы при добавлении блоков учить добавку к результатам предыдущего блока. Добавление так называемых *residual connections*

уменьшает эффективную глубину слоя, помогая бороться с затуханием градиента и позволяет эффективно обучать очень глубокие сети.

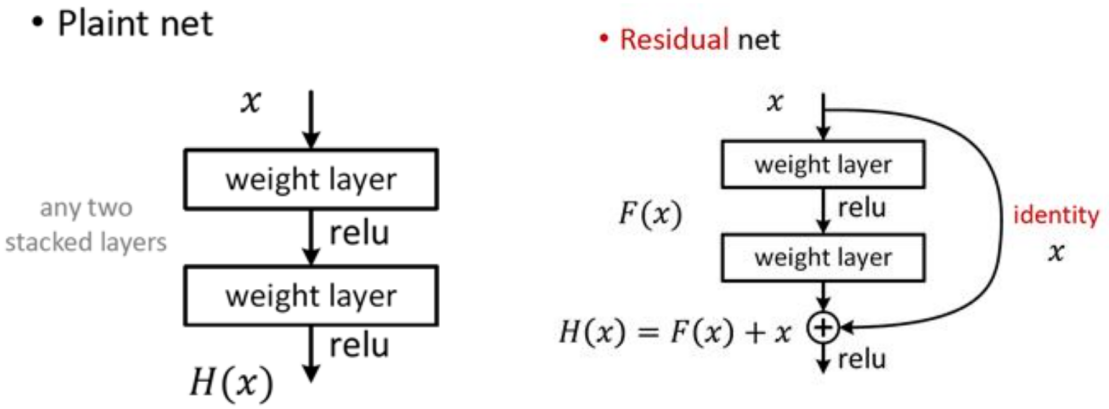


Рис. 2.3: ResNet, Residual block

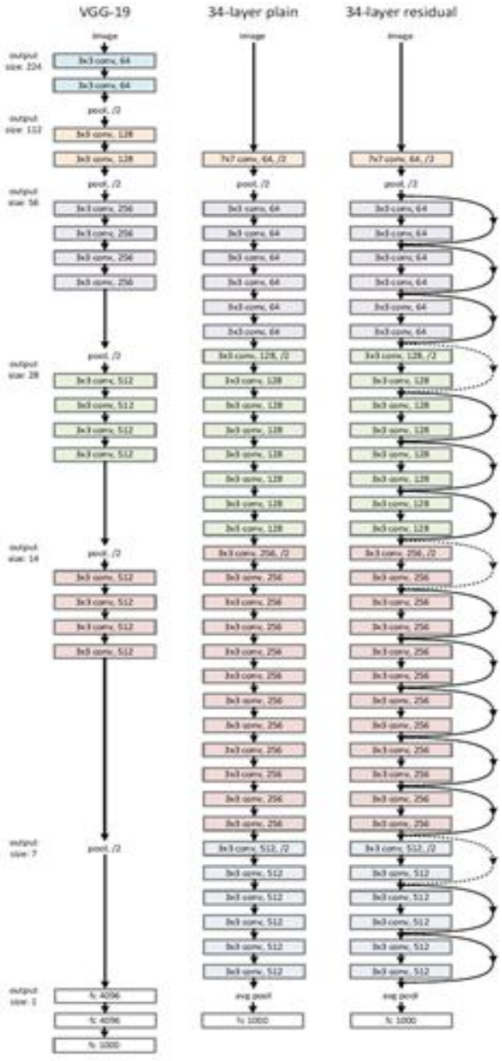


Рис. 2.4: ResNet

Глава 3

ResNeXt

Aggregated Residual Transformations for Deep Neural Networks. Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He

Практика *Inception* показывает, что метод *split-transform-merge* (см. Inception block) дает значительный прирост качества, однако в Inception сетях подбор правильных параметров блока, его внутренних слоев, является нетривиальной задачей. Авторы статьи решили улучшить исходную модель, добавили кроме глубины и ширины сети размерность *cardinality* (размер числа трансформаций). Кроме того увеличение этой размерности дает больший прирост качества по сравнению с глубиной и шириной. Ниже представлен блок слоев, заменяющий соответствующий блок в ResNet. Предложенный дизайн позволяет сделать любое количество трансформаций без ручного подбора параметров.

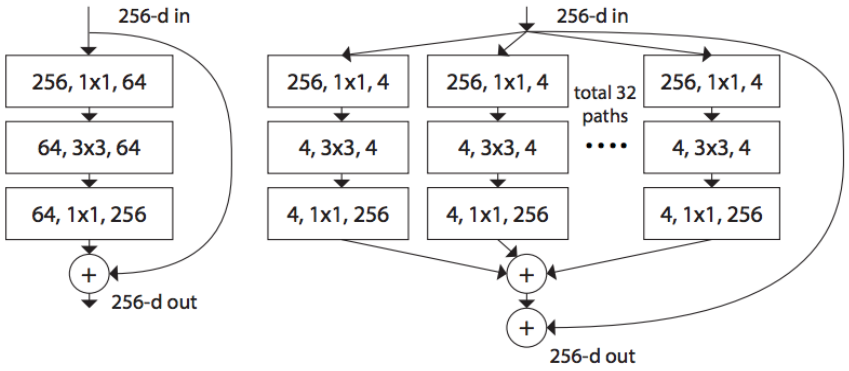


Figure 1. Left: A block of ResNet [14]. Right: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Рис. 3.1: ResNeXt block

В оригинальном ResNet каждый блок учился на $y = x + F(x)$. В ResNeXt $y = x + \sum_{i=1}^{C} \tau_i(x)$, где $\tau_i(x)$ - трансформация входа по одному из C путей блока.

Посчитаем число параметров. В левом блоке (обычный ResNet) имеем

$$N_{ResNet\_Block} = 256 \cdot 64 + 3 \cdot 3 \cdot 64 \cdot 64 + 64 \cdot 256 \approx 70k$$

В правом блоке RexNeXt (C - cardinality, число трансформаций)

$N_{ResNeXt\_Block} = C(256 \cdot d + 3 \cdot 3 \cdot d \cdot d + d \cdot 256), N_{ResNeXt\_Block} \approx 70k,$ при $d = 4, C = 32$

Следовательно, мы можем контролировать число параметров через trade-off между d и C. Авторы показывают, что увеличение C приносит большее качество, чем увеличение глубины или ширины.

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101's. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.

Рис. 3.2: Качество от изменения размерностей

Еще одно преимущество предложенной архитектуры в том, что из-за одинаковых трансформаций все пути в сети топологически эквивалентны, что позволяет делать групповые свертки, ускоряющие вычисления.

Сравнение с ResNet на ImageNet

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

Рис. 3.3: ResNet Vs ResNeXt

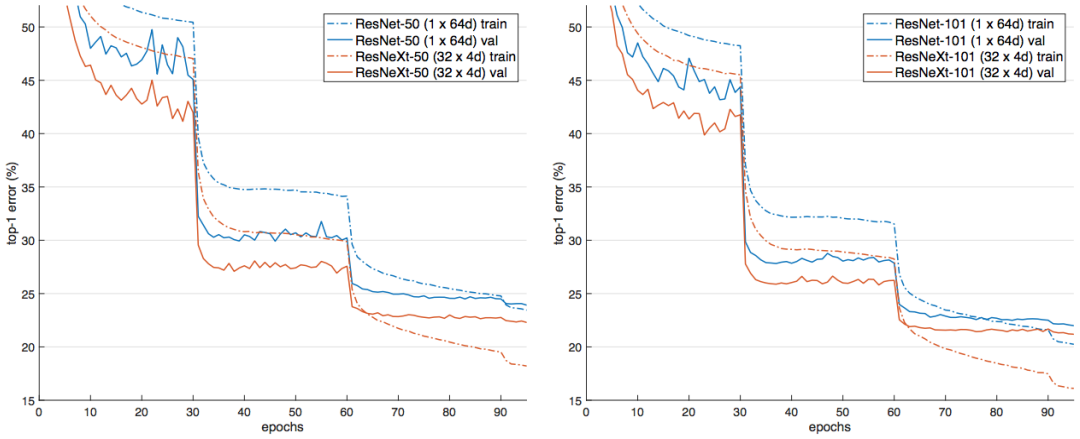


Figure 5. Training curves on ImageNet-1K. (**Left**): ResNet/ResNeXt-50 with preserved complexity (∼4.1 billion FLOPs, ∼25 million parameters); (**Right**): ResNet/ResNeXt-101 with preserved complexity (∼7.8 billion FLOPs, ∼44 million parameters).

Рис. 3.4: ResNet Vs ResNeXt

Сравнение со state-of-the-art моделями на тот момент

	224×224		320×320 / 299×299	
	top-1 err	top-5 err	top-1 err	top-5 err
ResNet-101 [14]	22.0	6.0	-	-
ResNet-200 [15]	21.7	5.8	20.1	4.8
Inception-v3 [39]	-	-	21.2	5.6
Inception-v4 [37]	-	-	20.0	5.0
Inception-ResNet-v2 [37]	-	-	19.9	4.9
ResNeXt-101 (**64** × **4d**)	20.4	5.3	**19.1**	**4.4**

Table 5. State-of-the-art models on the ImageNet-1K validation set (single-crop testing). The test size of ResNet/ResNeXt is 224×224 and 320×320 as in [15] and of the Inception models is 299×299.

Рис. 3.5: State-of-the-art ImageNet

Глава 4

Xception

Xception: Deep Learning with Depthwise Separable Convolutions, Franc¸ois Chollet, Google, Inc

Еще одна архитектура, производная от ResNet. Ее цель в том, чтобы сделать сеть более легковесной без потери качества, то есть более эффективно использовать параметры. Интересно, что автор архитектуры является автором фреймворка глубокого обучения *Keras*.

По своей сути модель продолжает идею *Inception* с ее блоками. В них мы внутри сверток 1×1 считаем кросс-канальную корреляцию, внутри сверток 3×3, 5×5 - внутри канальную. В *Inception* была гипотеза, что можно отделить эти стадии и сначала отображать в пространство меньшей размерности в свертке 1×1 (с кросс-корелляцией), а затем работать в полученном пространстве в следующих свертках. Можно упростить *Inception block* сделав одинаковые ветви. Кроме того можно построить эквивалентный блок с одной общей 1×1 сверткой, объединением всех каналов в один вектор и дальнейшие 3×3 свертки на 3 (по числу ветвей) подотрезках.

Figure 1. A canonical Inception module (Inception V3).

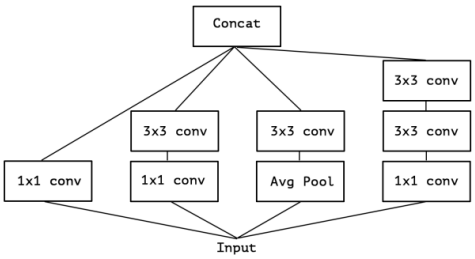


Figure 2. A simplified Inception module.

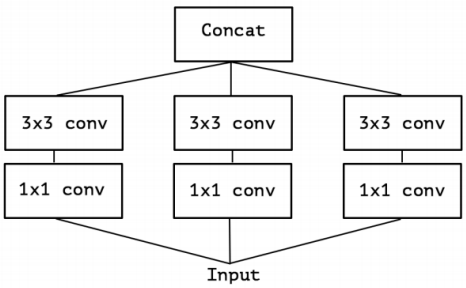


Figure 3. A strictly equivalent reformulation of the simplified Inception module.

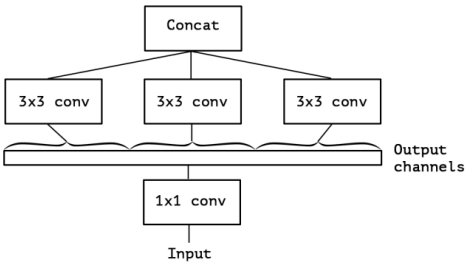
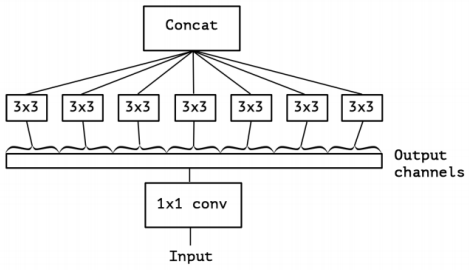


Figure 4. An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.



Xception, от Extreme Inception, усиливает гипотезу независимой последовательной обработки между и внутри каналов - каждый канал обрабатывается отдельно сверткой, как показано на рисунке выше. Метод называется *depthwise separable convolution* и эффективно реализован в *Tensorflow* и *Keras*, состоит из

отдельной *depthwise convolution* - свертки по каналам независимо, и *pointwise convolution* - свертки 1×1. В предложенной статье изменен порядок (сначала *pointwise*), но по словам автора это не имеет существенного значения.

Допустим, сворачиваем изображение с 16 каналами с 32 фильтрами 3×3. Итоговое число параметров $16 * 32 * 3 * 3 = 4608$. В предложенном подходе $16 * 32 * 1 * 1 = 512$ весов на свертку 1×1, $32 * 3 * 3 = 288$ у свертки 3×3. В итоге получаем гораздо меньше параметров.

Еще небольшое отличие в том, что между свертками блока нет нелинейностей, так как по их исследованиям это приносит больше качества.

Figure 5. The Xception architecture: the data first goes through the entry flow, then through the middle flow which is repeated eight times, and finally through the exit flow. Note that all Convolution and SeparableConvolution layers are followed by batch normalization [7] (not included in the diagram). All SeparableConvolution layers use a depth multiplier of 1 (no depth expansion).

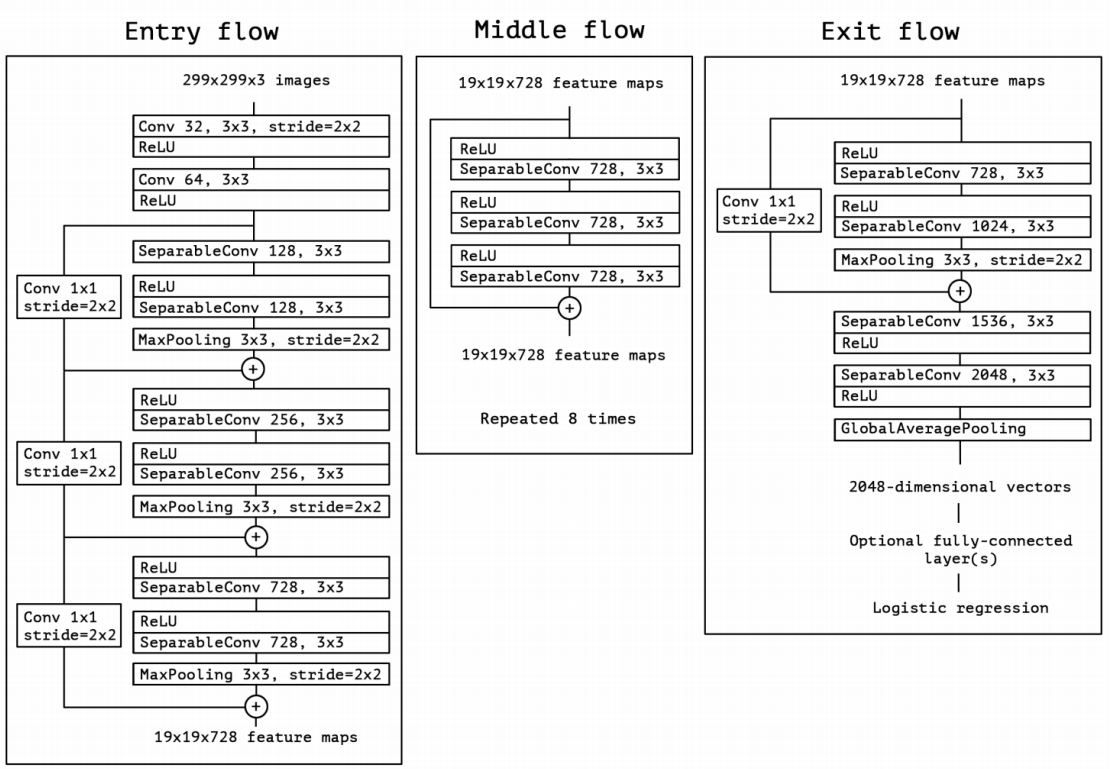


Рис. 4.2: Xception

Table 1. Classification performance comparison on ImageNet (single crop, single model). VGG-16 and ResNet-152 numbers are only included as a reminder. The version of Inception V3 being benchmarked does not include the auxiliary tower.

	Top-1 accuracy	Top-5 accuracy
VGG-16	0.715	0.901
ResNet-152	0.770	0.933
Inception V3	0.782	0.941
Xception	**0.790**	**0.945**

Рис. 4.3: Xception результаты на ImageNet

Если посмотреть на документацию Keras, можно найти следующую табличку сравнения моделей на ImageNet.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	99 MB	0.749	0.921	25,636,712	168
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88

Рис. 4.4: Сравнение моделей Keras

Видно, что модель сильно выигрывает по сравнению с аналогичными по числу параметров и размерам. Кроме того, идея Xception привела к легковесным моделям MobileNet, которые могут вычисляться прямо на смартфоне.