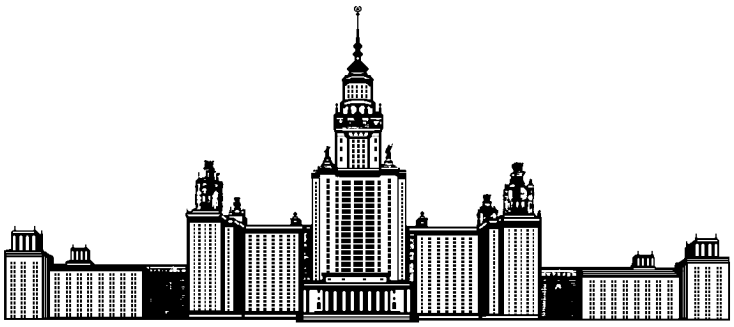
Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования



Доклад на тему

Архитектуры свёрточных нейронных сетей ч.1

Выполнил:
студент 3 курса 317 группы
Григорьев Илья Андреевич

Москва, 2021

Аннотация

Данный доклад является расширенным конспектом лекции по Deep Learning на тему архитектуры сверточных нейронных сетей от Александра Геннадьевича Дьяконова. Он будет полезен для читателей, знакомых с основами глубокого обучения, которые желают узнать, как развивались нейронные сети в сфере классификации и распознования изображений. Также доклад будет интересен тем, кто хочет разобраться в архитектуре и ключевых особенностях моделей, оказавших сильное влияние на современный Deep Learning и навсегда вошедших в историю данной области науки.

Содержание

1 Введение

Нейронная сеть (Neural Network) – это математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей. В машинном обучении нейронные сети это широко распространенное семейство алгоритмов, которое показывает наилучшее качество на множестве современных задач. Сверточные нейронные сети – это подкласс семейства моделей, в основе которого лежит использование сверточных слоев. Эта архитектура нацелена на эффективное распознавание образов, а идея для ее создания была так же взята из биологии. Связано это с особенностями работы зрительной коры головного мозга, что и объясняет особый успех сверточных нейронных сетей именно в задачах компьютерного зрения.

Долгое время они не были популярны из-за высоких вычислительных затрат, но сейчас практически все успешные методы решения задачи распознавания изображений так или иначе связаны с использованием сверточных нейронный сетей.

Целью этого доклада является знакомство читателя с некоторыми архитектурами сверточных нейронных сетей, которые оказали сильное влияние на дальнейшее развитие методов, касающихся обработки изображений. Помимо их строения будут разобраны техники и нововведения, которые позволили добиться улучшения результата.

2 Основные понятия

Рассматривается задача обучения с учителем (supervised learning). Т.е. нужно предсказать целевую переменную (target), по заданному входу x.

Всякая нейронная сеть строится путем комбинирования различных модулей, называемых слоями (layers). Каждый слой принимает на вход многомерный вектор и преобразовывает его. При этом меняются не только значения, но и размер вектора. Правильнее всего сказать, что нейронная сеть – это алгоритм, задающий последовательные преобразования признакового пространства.

Ниже даны определения основных слоев нейронной сети.

1. Полносвязный (Fully-Connected/Linear) – применяет умножение одномерного входа на матрицу весов. Гиперпараметры: количество выходных нейронов. Как правило в сверточных сетях применяется в самом конце для классификации.

2. Сверточный(Convolutional) – применение операции свёртки, суть которой в том, что каждый фрагмент входа умножается на ядро(kernel) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения. Каждое ядро порождает карту признаков (feature map), которая является двумерным вектором. Основные гиперпараметры: количество выходных каналов(channels), размер ядра, шаг ядра(stride), padding. Главная задача сверточных слоев – извлечение признаков

из изображений.

3. Дропаут (Dropout) – в процессе обучения обнуляет определенную долю значений одномерного входа. Гиперпараметры: число входных элементов, доля выброса (dropout rate). Так же есть DropConnect, идея которого заключается в занулении не элементов вектора, а определенных связей между слоями. Эти подходы нужны для борьбы с переобучением нейронной сети. Случайное исключение нейронов или связей в процессе обучения заставляет сеть учить как можно больше своих элементов, а не надеяться на какой-то хорошо обученный один. Важно отметить, что работа дропаут слоя в процессе обучения и инференса различается.

4. Пулинг (Pooling) – уплотнение двумерного входа, при этом группа пикселей уплотняется до одного, проходя нелинейное преобразование. Пулинг применяется не ко всему тензору как операция свертки, а отдельно к каждому каналу. Основная задача пулинга – уменьшить пространственные размерности карт признаков. Частые примеры: average-pooling (среднее элементов), max-pooling (максимум). Гиперпараметры: размер ядра, шаг ядра (обычно равен его размеру).

5. Батч-нормализация (Batch-Normalization) – нормирование входного вектора без изменения размера. Слой батч-нормализации имеет обучаемые параметры. Его главная цель – ускорить процесс обучения и сделать его более стабильным за счет центрирования и масштабирования данных. Известно, что градиентные методы оптимизации сходятся быстрее, когда линии уровня ровные и имеют форму окружностей.

6. Слой активации (Activation) – нелинейная функция, примененная поэлементно к входу слоя. Для разных задач лучше подходят свои нелинейности. Самые часто используемые нелинейные функции в слое активации: ReLU, Leaky ReLU, ELU, SELU, сигмоида, гиперболический тангенс.

7. Софтмакс слой (Softmax output) – обобщение логистической функции для многомерного случая, не меняет размер входа. Выдает вероятности принадлежности к классам.

Вход сверточной нейросети представляет собой тензор (входное изображение). Размеры этого тензора равны ширине, высоте и числу каналов входного изображения. Самый распространенный тип изображений RGB имеет три канала. Многие архитектуры обучались на датасете ImageNet. С 2010 года ведётся проект ILSVRC (англ. ImageNet Large Scale Visual Recognition Challenge — Кампания по широкомасштабному распознаванию образов в ImageNet), в рамках которого различные программные продукты ежегодно соревнуются в классификации и распознавании объектов и сцен в базе данных ImageNet. Часто на этом датасете смотрят на ошибки типа top-1 (класс верно угадан) и top-5 (одно из 5 предположений верно).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Рис. 1: Соответствие карт признаков второго слоя с ядрами третьего у LeNet5

3 Архитектуры

3.1 LeNet-5

Одна из первых сверточных архитектур, разработанная в 1998-м году. Отказ от использования полносвязных сетей позволил сильно уменьшить количество параметров. При этом свертки отлично справлялись с выделением признаков, концетрируя свое внимание на определенных участках изображения, а не на отдельных пикселях, которые сильно коррелированы. Правда тогда эта сеть не получила особого признания, но можно считать ее прародителем всех современных архитектур сверточных нейросетей.

На вход подается черно-белое изображение размером 32 на 32 пикселя. Сеть состоит из 7 слоев:

1. Сверточный слой. Количество каналов – 6, размер ядра–5×5, шаг–1.
2. Пулинг слой. Размер ядра – 2×2.
3. Сверточный слой. Количество каналов – 16, размер ядра–5×5, шаг–1. Однако некоторые соединения опущены согласно рис. 1. Это сделано для того, чтобы убрать симметричность в сети и уменьшить количество параметров.
4. Пулинг, аналогичный второму слою.
5. Сверточный слой. Количество каналов – 120, размер ядра–5×5.
6. Полносвязный слой, состоящий из 84 нейронов.
7. Полносвязный слой из 10 нейронов, после которого идет Softmax.

Строение сети и размеры данных при проходе через нее представлены на рис. 2.

Впервые представлен принцип, который станет популярным в будущем. Сеть как бы делится на две половины: половина, выделяющая признаки (в основном состоит из сверток и пулинга) и половина, служащая классификатором выделенных признаков (используются полносвязные слои). В качестве функций активации в LeNet-5 используется гиперболический тангенс, а в качестве пулинга – average-pooling.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Рис. 2: Архитектура LeNet, tanh – гиперболический тангенс

3.2 AlexNet

В 2012-м Алексей Крижевский опубликовал AlexNet, углублённую и расширенную версию LeNet, которая с большим отрывом победила в соревновании ImageNet (с количеством ошибок 15.3% против 26.2% у второго места). Кардинальных отличий внесено не было. Сеть стала значительно больше. На рисунке 3 можно увидеть ее строение более подробно.

После каждого сверточного и полносвязного слоя применяется функция активации Relu, а не гиперболический тангенс или сигмоида. Это позволило увеличить скорость в 6 раз. Кроме выигрыша в скорости вычисления, Relu обладает тем свойством, что на положительной полуоси градиент вообще не видоизменяется, в то время как у сигмоиды или гиперболического тангенса он умножается на число меньшее единицы, что провоцирует проблему затухания. Вообще проблема затухающих градиентов особенно сильно проявляется в глубоких сетях. Обучаясь методом обратного распространения ошибки, градиентам тяжело доходить до начальных слоев, если они постоянно умножаются на числа меньшие единицы по модулю.

В AlexNet в качестве пулинга используется max-pooling, а не average-pooling, как это было в прошлой архитектуре. Причем размер шага меньше размера ядра.

Перед первым и вторым полносвязными слоями стоят слои дропаута (с долей выброса 0.5), которые решают проблему переобучения, но удваивают время обучения сети.

Сеть имеет два вытянутых параллельных участка. Это было сделано, чтобы обучать нейросеть параллельно на двух видеокартах Nvidia Geforce GTX 580, что в итоге заняло 1 неделю. Использовался стохастический градиентный спуск (SGD) со скоростью обучения (learning rate) 0.01, импульсом (momentum) 0.9 и распадом весовых коэффициентов (weight decay) 0.0005. Применялась техника аугментации данных (data augmentation) – искусственное размноже-

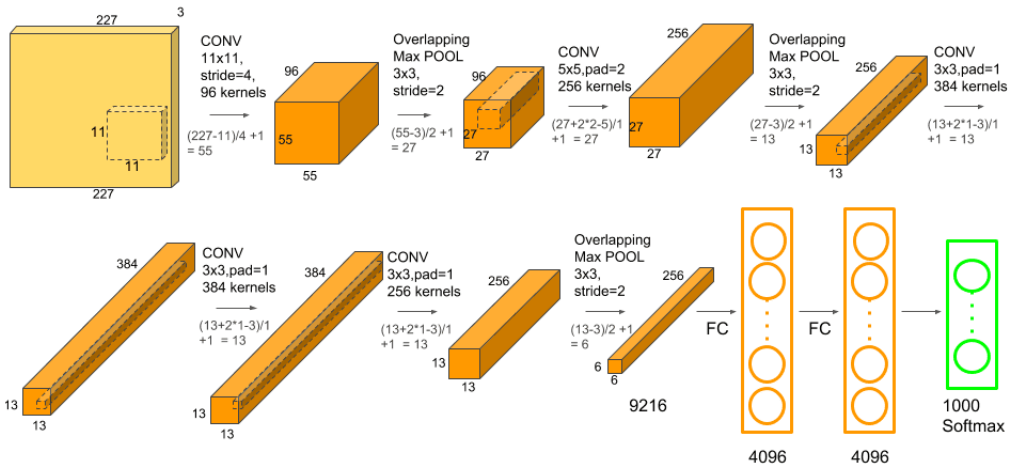


Рис. 3: Архитектура AlexNet

ние обучающей выборки за счет некоторых преобразований. Сеть обучалась по батчам размера 128 и имела 60 миллионов параметров, что намного больше чем предыдущая сеть.

Через год после публикации AlexNet все участники конкурса ImageNet стали использовать сверточные нейронные сети для решения задачи классификации. AlexNet была первой успешной реализацией сверточных нейронных сетей и открыла новую эру исследований.

3.3 VGG

VGG16 — модель, предложенная K. Simonyan и A. Zisserman из Оксфордского университета в статье “Very Deep Convolutional Networks for Large-Scale Image Recognition”. Модель достигает точности 92.7% — топ-5, при тестировании на ImageNet в задаче распознавания объектов на изображении.

VGG использует свертки с маленьким размером ядра(3×3), что противоречит заложенным в предыдущих моделях принципам. Обоснованно такое решение возможностью того, что несколько свёрток 3x3, объединённых в последовательность, могут эмулировать более крупные рецептивные поля, например, 5x5 или 7x7, при этом число обучаемых параметров получается меньше. Эти идеи позднее будут использованы в архитектурах Inception и ResNet. Строение VGG16 представлено на рисунке 4.

По краям входа сверточного слоя добавляются нулевые пиксели (padding) таким образом, чтобы пространственное разрешение сохранялось после свертки, то есть дополнение равно 1 для 3x3 сверточных слоев.

Уменьшение размера изображения осуществляется только через max-pooling с размером ядра 2 и таким же шагом. Таким образом, изображение умень-

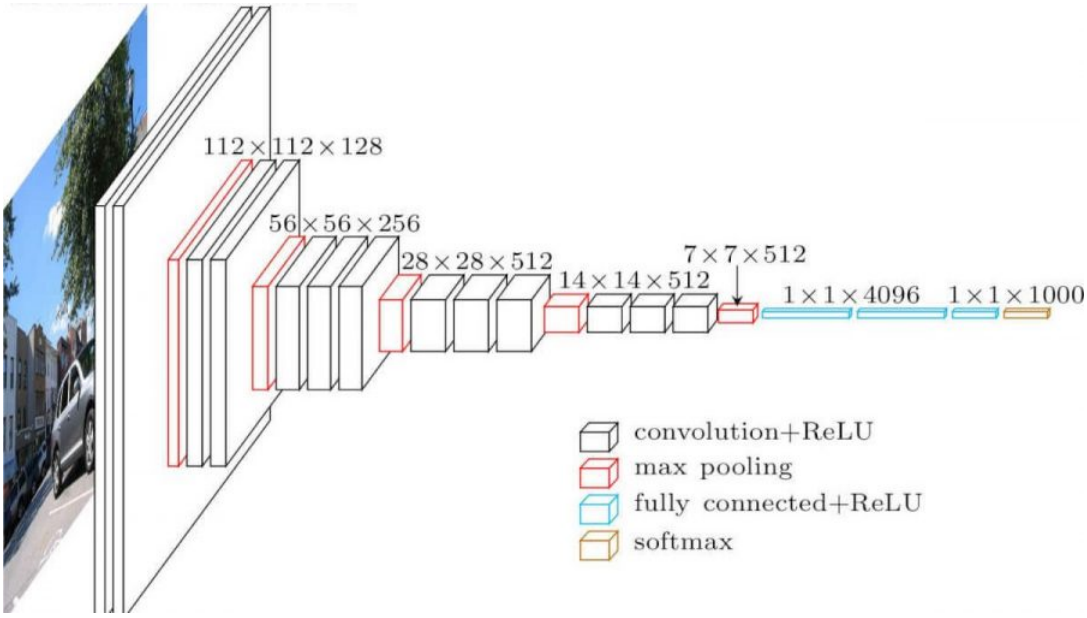


Рис. 4: Архитектура VGG16

шается ровно в 2 раза после каждого такого пулинга.

Классификатор сети состоит из трех полносвязных слоев с Softmax на выходе.

Функции активации (Relu) следуют за каждым сверточным и полносвязным слоем.

Есть несколько конфигураций этой сети. Все они представлены на рис. 5.

Размер сети получился очень большой – более 130 миллионов параметров у VGG16 с весом более 530 МБ. Из-за этого сеть сложно хранить и долго обучать (3 недели на 4-х видеокартах NVIDIA TITAN BLACK). В VGG применялся дропаут и обучение по батчам размера 256. В числе положительных качеств стоит простота реализации и хорошее качество в различных задачах.

3.4 GoogLeNet

Кристиан Жегеди (Christian Szegedy) из Google озаботился снижением объёма вычислений в глубоких нейросетях, и в результате создал GoogLeNet — первую архитектуру Inception. Она выиграла ImageNet recognition challenge в 2014-м году с результатом 6.67% top 5 error.

Главным новшеством модели стало появление модулей Inception, которые представлены на рисунке 6. Так же Google впервые представили нейронную сеть в виде конструктора, который можно собирать по блокам.

Во-первых, данные в блоке идут по нескольким параллельным тропам, которые после конкатенируются. Это позволяет частично переложить ответственность выбора наилучшего строения слоев на саму сеть. В результате обучения наиболее полезные пути станут вносить больший вклад в ответ.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Рис. 5: Конфигурации VGG. Наиболее популярны VGG16 и VGG19 – колонки D и E.

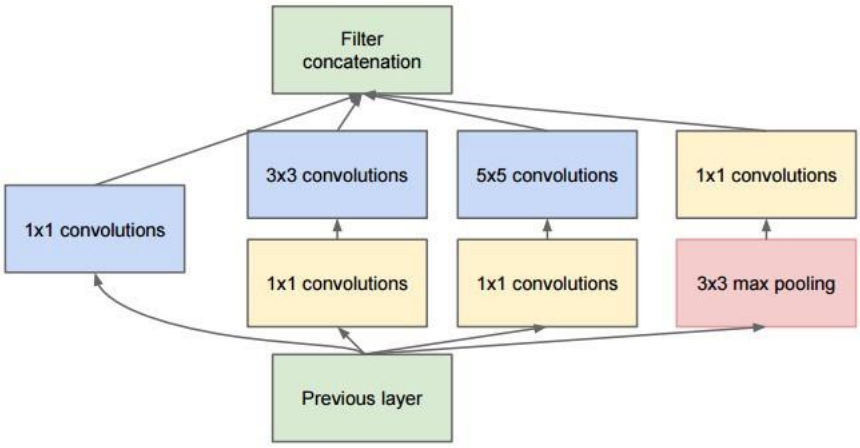


Рис. 6: Модуль Inception

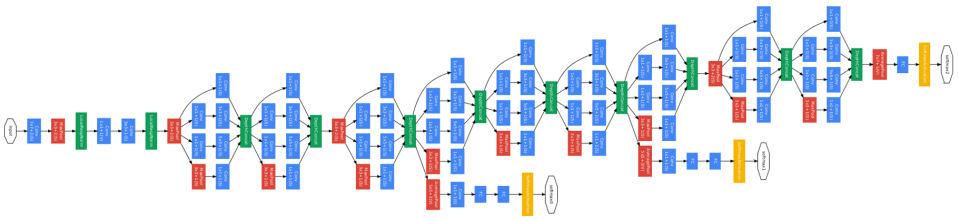


Рис. 7: GoogLeNet aka Inseption v1

Вторая, тоже очень важная идея – использование сверток с единичным размером ядра. По сути, это линейная комбинация карт признаков. Дело в том, что эти карты часто бывают коррелированы между собой, поэтому такой подход показал большую эффективность как способ уменьшить число каналов, сохранив пространственные размеры. Такие свертки ставят непосредственно перед обычными сверточными слоями, что позволяет значительно снизить количество параметров и время работы.

Авторы решили отказаться от использования полносвязных слоев. Вместо этого на последнем уровне ставиться слой Global Average Pooling, который просто усредняет каждую карту признаков до одного одного числа (т.е. на выходе получится одномерный вектор). После этого ставится один полносвязный слой и слой Softmax.

Итак, сначала в сети идут несколько обычных слоев выделения признаков, а затем просто повторяются Inception модули.

В процессе тренировки используются два дополнительных выхода на более ранних слоях (с весом 0.3 к общей ошибке) для борьбы с затуханием градиента (хотя позже выяснилось что этот трюк помогал скорее как средство регуляризации). Это было необходимо, потому что сеть очень глубокая – 22 слоя. Для применения сети все пути, ведущие к вспомогательным выходам, удаляются из архитектуры.

Сеть показала отличное качество, при этом используя совсем небольшое количество параметров (около 5 миллионов) за счет применения 1×1 сверток. GoogLeNet можно назвать настоящим прорывом, потому что улучшение качества было достигнуто не за счет огромного числа параметров, а за счет массы интересных архитектурных новшеств.

3.5 Inception v2 & v3

Спустя год выходит новая статья, где авторы формулируют основные принципы построения эффективных сетей. Новые идеи таковы:

1. Стоит избегать 1×1 сверток с сильным уровнем сжатия, особенно на ранних слоях.

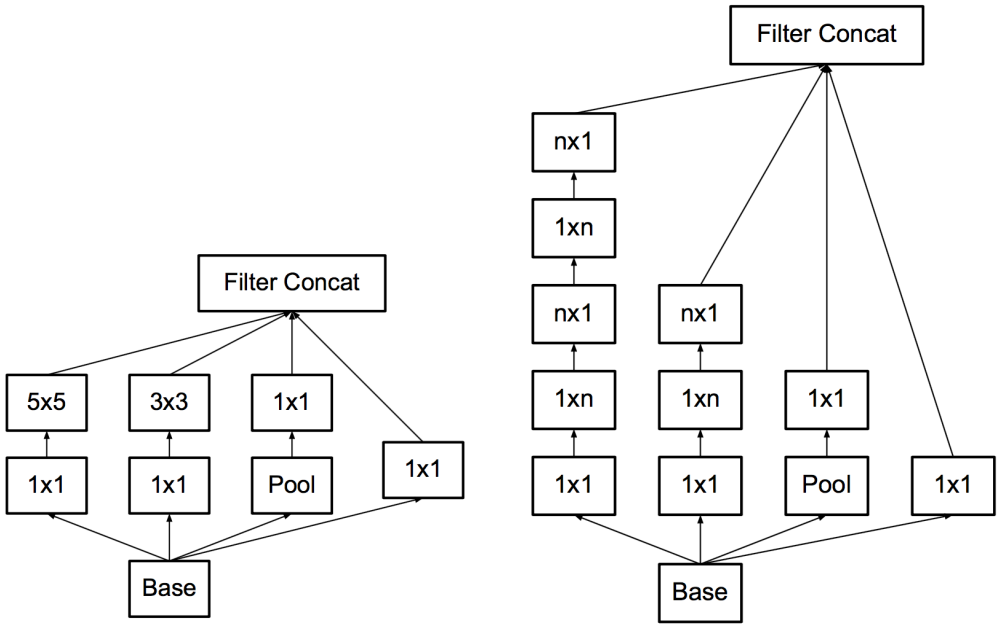


Рис. 8: Новые модули Inseption

2. При росте количества ресурсов следует соблюдать баланс между глубиной и шириной сети.

3. Можно заменить одну свертку с большими ядрами на несколько сверток с маленькими практически без потери качества. Предположительно это связано с сильной корреляцией между соседними блоками. Т. е. можно заменить свертку 5×5 на две свертки 3×3 (при этом эксперементально подтверждается, что лучше продолжать ставить нелинейности между слоями).

   Более того, можно заменить любую свертку $n \times n$ на $(1 \times n + n \times 1)$. Авторы добились хороших результатов при n равном 7. Новые модули представлены на рисунке 8.

   В результате использовать такие свертки становится очень дешево.

4. На дополнительных выходах добавили батч-нормализацию как средство регуляризации.

5. Техника label-smoothing. Обычно в качестве целевой переменной был вектор, у которого на правильном классе стоит 1, на остальных 0. Но softmax достигает 1 только на бесконечности, так что градиентный спуск будет менять параметры модели, даже если она уже уверена в правильном ответе. Это ведет к переобучению. Поэтому авторы предлагают смешать изначальный вектор с распределением, пропорциональным распределению классов по датасету. В итоге на истинной метке будет значение чуть

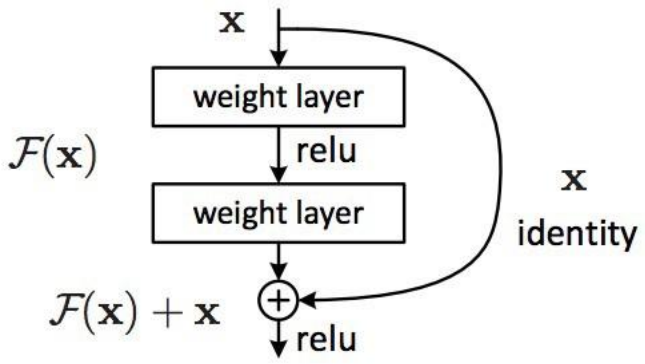


Рис. 9: Блок ResNet

меньше одного, а на остальных чуть больше нуля.

Основная архитектура называется Inception-v2, а версия, где дополнительные классификаторы работают с батч-нормализацией – Inception-v3. Inception-v3 достигает 4.2% top5 classification error на Imagenet.

3.6 ResNet

В 2015 году люди в Microsoft Research Asia, задумались над проблемой тренировки очень глубоких сетей. Они представили свою архитектуру, которая опередила Inception v3 (на ансамбле моделей).

Идея очень проста: надо пустить данные параллельно некоторому модулю через тождественный слой (identity layer), после чего просуммировать, как представлено на рисунке 9. Градиент через прокидывание (skip/shortcut connections) легко посчитать, он умножается на единичную матрицу. Размерность тензора может поменяться при прохождении через Residual block, в таком случае нельзя просто сложить тензор, прошедший через преобразования, и тензор, прошедший через тождественный слой. В такой ситуации на тождественном пути все же делаются свертки и батч-нормализация.

Стоит отметить, что прокидывание входа через всего один слой не дает существенной выгоды.

В результате стало возможным обучать сети глубиной более 100 слоев, что открыло новую эру архитектур.

Для начала была построена глубокая версия VGG с применением своей техники. Пример такой модели приведен на рисунке 10. Качество улучшилось по сравнению с оригиналом.

Далее авторы продолжили углублять сеть, поэтому для уменьшения вычислительных затрат применяется техника bottleneck согласно рисунку 11. Размерность понижается перед дорогим сверточным слоем и возвращается в изначальное состояние после него. Количество параметров в таком случае сильно уменьшается.

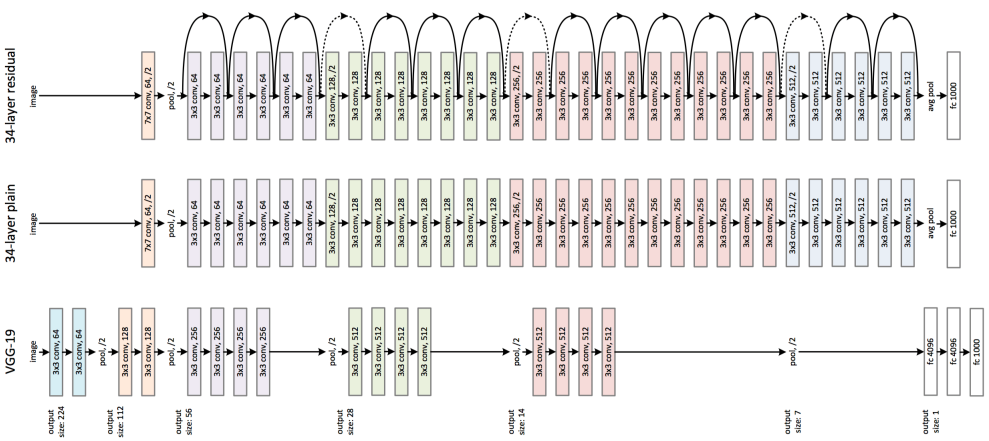


Рис. 10: ResNet аналог к VGG19

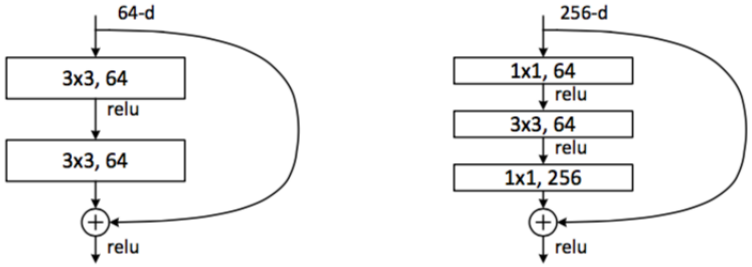


Рис. 11: Блок ResNet до и после применения bottleneck

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рис. 12: Архитектуры ResNet для ImageNet

Самая глубокая версия имела 152 слоя, ошибка на top5 classification error на ImageNet равна 4.49%. При всем этом, сложность модели была меньше, чем у VGG16.

Одно из важных предположений, указанных в статье – дело не в затухании градиента. В своих сетях они используют батч-нормализацию (после каждого сверточного слоя) и грамотную инициализацию, а экспериментальные наблюдения подтверждают, что величина градиента остается разумной и не затухает. Были поставлены эксперименты с сетями глубиной 18 и 34 слоя. Без прокидывания менее глубокая сеть работала лучше, но с прокидыванием все стало наоборот. Их теория в том, что более глубокие сети просто экспоненциально медленнее сходятся в процессе обучения, и поэтому такой же точности мы просто не успеваем дождаться с теми же вычислительными ресурсами. ResNet в свою очередь позволяет ускорить этот процесс.

Также авторы наблюдают интересный эффект: выходы 3×3 сверток после батч-нормализации имеют меньшее стандартное отклонение, если использовать прокидывание связей (рисунок 13).

В работе «Residual Networks Behave Like Ensembles of Relatively Shallow Networks» говорится о том, что ResNet архитектуры ведут себя как ансамбли небольших сетей. Для примера можно взять 3 блока с прокидыванием (рисунок 14). Их можно описать рекурсивной формулой $y_{i+1} = f_i(y_i) + y_i$. Если раскрыть эту формулу для всех i от 1 до 3, то становится очевидно, что сеть имеет множество скрытых путей, связывающих вход с выходом. Далее экспериментально подтверждается, что эти пути сильно не зависят друг от друга (удаление отдельных блоков не ведет к потерям качества), и что ошибка растет плавно при увеличении количества удаленных слоев и перестановке блоков. Такое поведение свойственно ансамблям моделей.

Важно отметить, что в процессе развития, архитектуры сверточных сетей не только привносят новые эвристики, но и становятся лучше оптимизируемыми. Поверхность функции ошибки сети с прокидыванием связей намного более гладкая, содержит меньше локальных минимумов и имеет ярко выра-

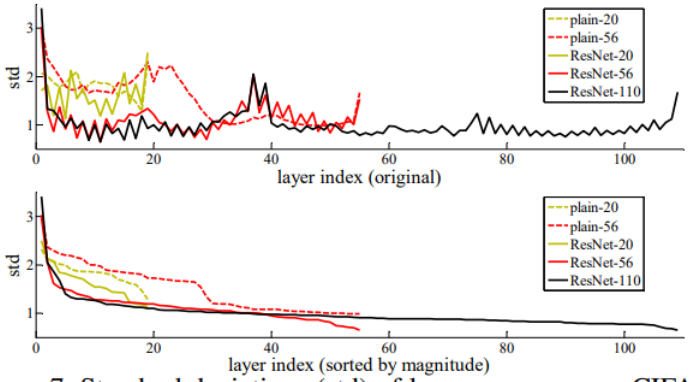


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

Рис. 13: Отклик слоев

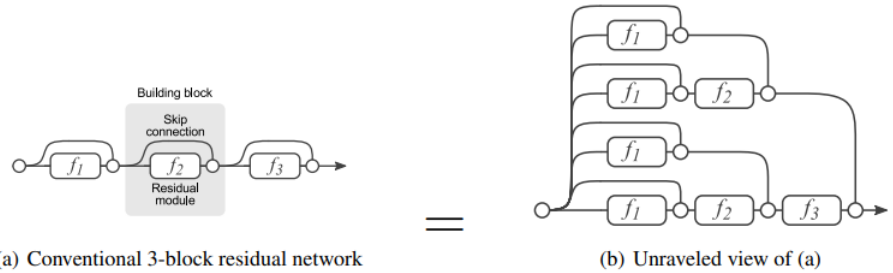


Рис. 14: Развернутый вид ResNet блоков

женный глобальный минимум (рисунок 16). Таким образом, использование прокидывания связей значительно упрощает оптимизационную задачу и делает ее более устойчивой.

3.7 Inception-v4 (Inception-ResNet)

После выхода ResNet команда из Google решили модернизировать Inception-v3 и добавили к уже существующей архитектуре прокидывание связей. Inception, но уже версии 4, снова ставит рекорд на ImageNet, при этом она имеет меньшее число параметров, чем ResNet (рисунок 20).

3.8 SENet

Последним чемпионом на ImageNet стала сеть SENet. Архитектурных новшеств придумано не было, главным нововведением стала адаптивная калибровка (рисунок 18). Данный процесс заключается в следующем: делается гло-

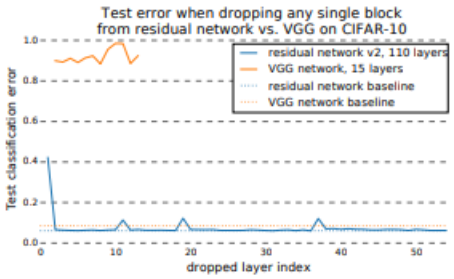


Figure 3: Deleting individual layers from VGG and a residual network on CIFAR-10. VGG performance drops to random chance when any one of its layers is deleted, but deleting individual modules from residual networks has a minimal impact on performance. Removing downsampling modules has a slightly higher impact.

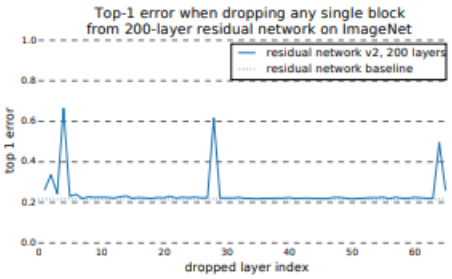
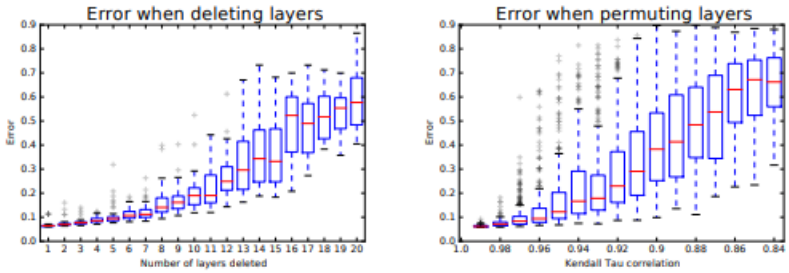


Figure 4: Results when dropping individual blocks from residual networks trained on ImageNet are similar to CIFAR results. However, downsampling layers tend to have more impact on ImageNet.



(a) (b)

Figure 5: (a) Error increases smoothly when randomly deleting several modules from a residual network. (b) Error also increases smoothly when re-ordering a residual network by shuffling building blocks. The degree of reordering is measured by the Kendall Tau correlation coefficient. These results are similar to what one would expect from ensembles.

Рис. 15: Эксперименты с перестановкой и удалением слоев ResNet.

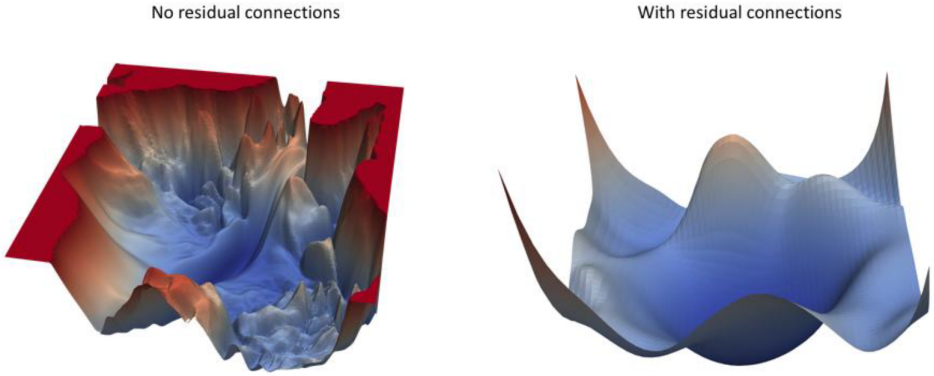


Рис. 16: Поверхности функций ошибки

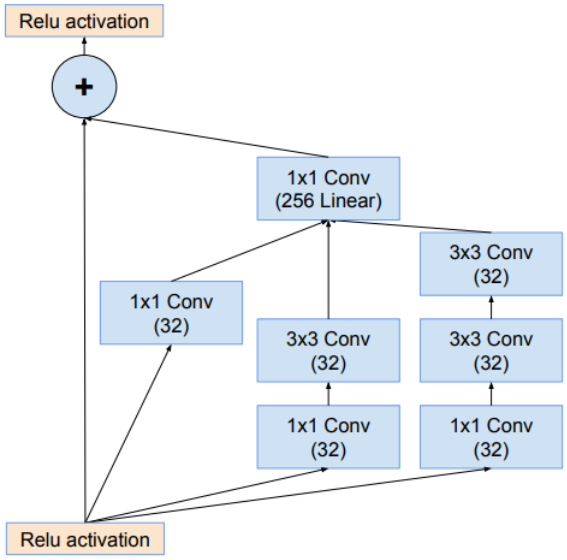


Рис. 17: Модуль Inception-ResNet

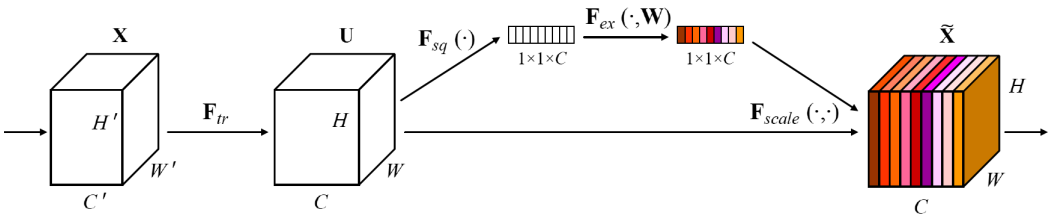


Рис. 18: Squeeze-and-Excitation block

бальный average-pooling с сохранением размерности по каналам, получается одномерный вектор. Он пропускается через отдельную небольшую нейросеть, состоящую из линейного слоя, Relu, линейного слоя и сигмоиды. На выходе получается вектор той же размерности, что и на входе, с элементами от 0 до 1. Далее каждый канал тензора карт признаков умножается на соответствующую компоненту полученного вектора. Таким образом, выполняется масштабирование каждого канала в зависимости от его значимости: полезные каналы умножаются на числа близкие к 1, а не особо важные каналы умножаются на числа близкие к 0.

Еще одна важная особенность SE-блоков заключается в том, что их можно применять к любым другим архитектурам сверточных сетей. Так, напрмер, были получены SE-Inception модуль и SE-ResNet модуль, показанные на рисунке 19.

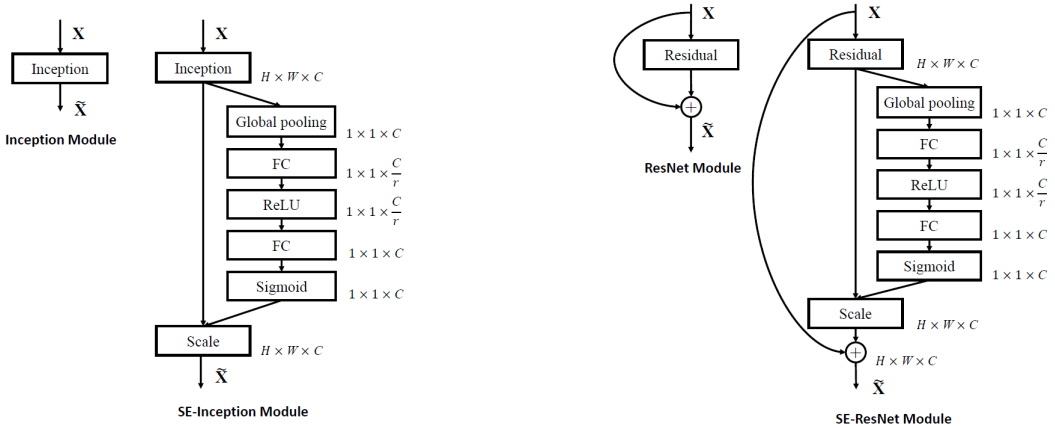


Рис. 19: SE-Inception и SE-ResNet модули

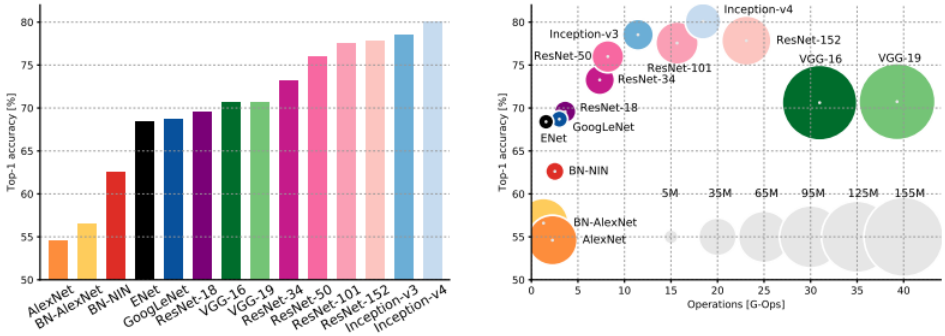


Рис. 20: Сравнение моделей по качеству и количеству операций

4 Итог

За последнее десятилетие Deep Learning прошел огромный путь развития. Нейронные архитектуры постоянно улучшаются, достигая новых высот качества. Можно наблюдать постоянное усложнение структуры моделей (что не обязательно сопровождается ростом количества параметров, рисунок 20).

В этом докладе были рассмотрены архитектуры сверточных нейронных сетей, оказавшие сильное влияние на развитие компьютерного зрения и глубокого обучения в целом.

5 Ссылки

5.1 Оригинальные статьи

1. Gradient-Based Learning Applied to Document Recognition, Yann LeCun(LeNet), 1998

2. ImageNet Classification with Deep Convolutional Neural Networks, Alex

Krizhevsky(AlexNet)

3. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, Karen Simonyan, Andrew Zisserman(VGG), 2015
4. Going deeper with convolutions(GoogleNet), 2014
5. Rethinking the Inception Architecture for Computer Vision(Inception v2/v3), 2015
6. Deep Residual Learning for Image Recognition, Kaiming He(ResNet), 2015
7. Residual Networks Behave Like Ensembles of Relatively Shallow Networks, Andreas Veit, Michael Wilber, Serge Belongie, 2016
8. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, 2016
9. AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS, Alfredo Canziani, Eugenio Culurciello, 2017
10. Squeeze-and-Excitation Networks, 2019
11. Visualizing the Loss Landscape of Neural Nets, 2018

5.2 Ссылки

1. `https://ru.wikipedia.org/wiki/ImageNet`
2. `https://cutt.ly/HuT4pde`
3. `https://neurohive.io/ru/vidy-nejrosetej/vgg16-model/`
4. `https://habr.com/ru/post/301084/`
5. `https://habr.com/ru/post/302242/`
6. `https://habr.com/ru/post/303196/`
7. `https://habr.com/ru/company/nix/blog/430524/`