KARPOV.COURSES >>>
КОНСПЕКТ

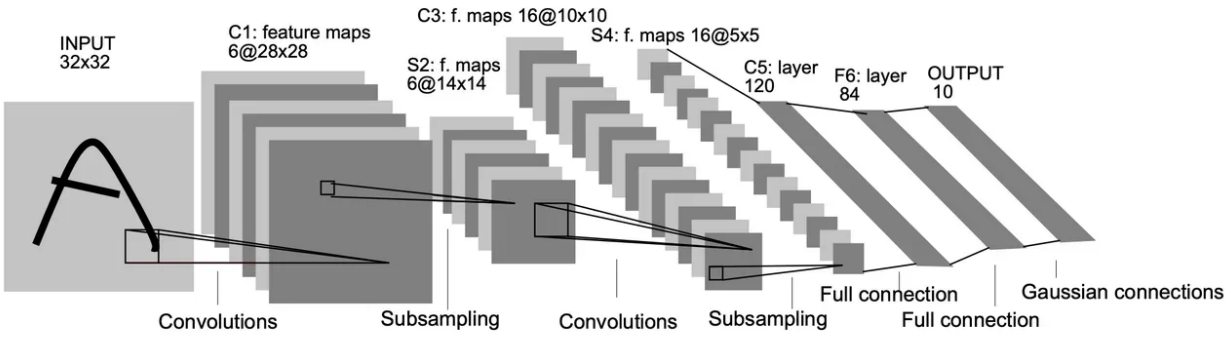
Конспект > 5 урок > Популярные архитектуры сверточных нейронных сетей. Перенос знаний

> LeNet (1998)

Давайте рассмотрим архитектуру сверточных нейронных сетей как **LeNet**, которая применялась для классификации датасета **MNIST**. Она была характерна тем, что впервые было предложено использовать **end-to-end** обучение как единую модель. До этого все это выглядело так: были модули, которые выделяют какие-то признаки (например вертикальные, горизонтальные палки и т.д.), выравнивают изображения, и делают прочие преобразования, а потом все это пытается как-то обучить классификатор.

Все это не очень хорошо работало и в 1998 году **LeNet** классифицировали **MNIST** с очень хорошим качеством - 99% на

тестовой выборке, они использовали аугментацию, и у этой нейронной сети было порядка 60 тысяч параметров, т.е. не очень большая. Соответственно тут всего несколько сверточных слоев, а если точнее 2 сверточных слоя и 3 полноценных слоя. Такая архитектура хорошо работала для датасета **MNIST**, но картинки увеличиваются, данные меняются, и появилась потребность в новой архитектуре, которая будет рассмотрена далее. Архитектуру **LeNet** можно посмотреть на картинке ниже.




```
# pragma dataset init mnist --size 1Gb


from torchvision.datasets import MNIST
mnist = MNIST('/home/jupyter/mnt/datasets/mnist', download = True)
```

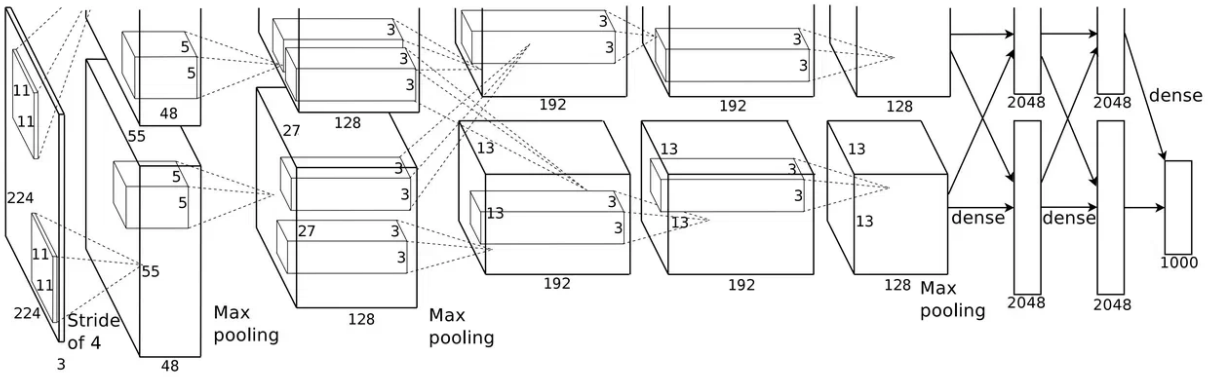
> ImageNet (2010)

В 2010 году произошел переломный момент в мире сверточных нейронных сетей. Появился датасет **ImageNet**. На него выделили деньги на сбор, разметку и т.д. ребята из Стенфорда (Стэнфордский университет). Сделали огромный датасет, больше 1 млн изображений, тысячи классов, и разметка на каждом изображении. Соответственно разметка происходила следующим образом, люди вручную размечали датасет, например, ставился вопрос есть ли на изображении кот, если есть, то люди нужно было поставить цифру 1. Таким образом были размечены изображения, и на этой основе было создано соревнование, появилась идея добавить некую геймификацию, какая модель будет лучше, у кого лучше получиться решить задачу среди исследователей. В итоге это дало очень большой

скачок, потому исследователи начали соревноваться между собой, появился популярный бенчмарк для тестирования нейронных сетей, что существенно повлияло на развитие отрасли. Соревнование называлось **ImageNet Large Scale Visual Recognition Challenge** и оно ежегодно проводилось с 2010 года., каждый год приезжали исследователи, публиковали свои статьи с придуманными архитектуры, подходами, и качество все росло и росло. Далее рассмотрим некоторые популярные архитектуры, которые были созданы для решения **ImageNet**.

> AlexNet (2012)

В 2012-м году появилась архитектура под названием **AlexNet**, названная по имени первого исследователя Алекса Крижевского. Группа исследователей была из университета Торонто, их статья называлась **ImageNet Classification with Deep Convolutional Neural Networks**. Модель их архитектуры на картинке:



Сначала были свертки 11*11, потом свертки 5*5, 3*3, затем полносвязные слои. Как можно заметить, картинка выглядит достаточно странно. Тем не менее, это осмысленная картинка, потому что модель обучали на двух GPU. Нижняя часть обучалась на одной, верхняя — на другой, а в полносвязных слоях все это конкатенировалось. Они уже использовали **relu**, **dropout**, **аугментацию**. Обучение модели заняло около недели, осуществлялось на двух GPU. Железо по меркам 2022 года можно считать слабым, но идеи были хорошие. Важно отметить, что свертки 11*11 на входе — это очень много, очень медленно. Чтобы обучить более

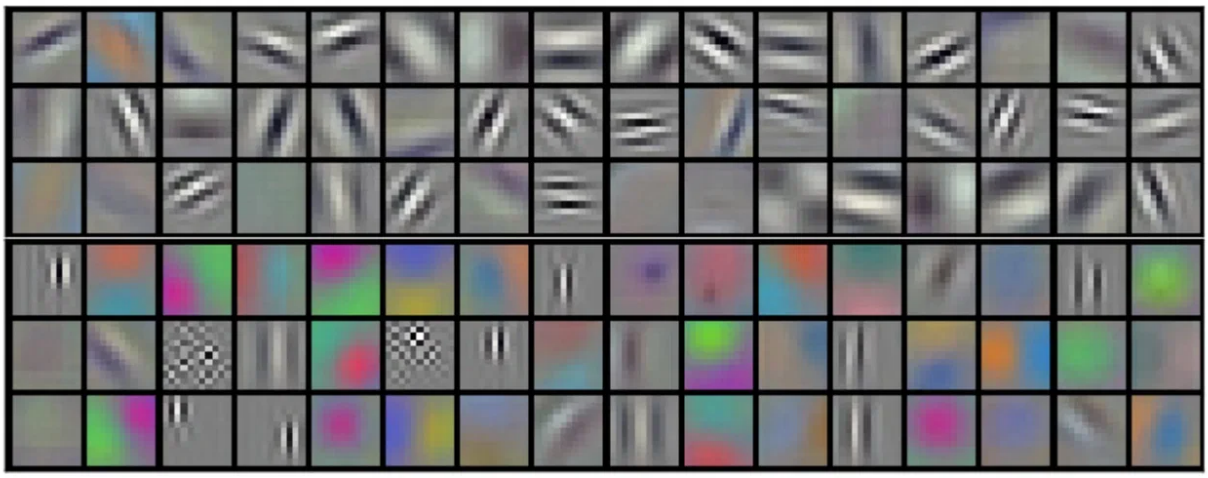
60-ти миллионов параметров, использовали градиентный спуск с инерцией. Топ5-Ошибка у нее на **ImageNet** составляет 17%.

Как мы помним, на **ImageNet** были метки такго вида: есть ли такой класс на изображении или нет. В соревновании было две метрики: Топ1-Ошибка и Топ5-Ошибка. Модель предсказывала по уверенности, т.е. ранжировала классы по уверенности, есть ли этот класс на картинке или нет. Например, у нас есть 1 тыс. классов, модель выдавала свою уверенность в том, что на картинке с наибольшей веростностью есть кот, с меньшей вероятностью есть собака, т.е. выдавала определенный вектор вероятностей.

Топ1-Ошибка — это значит, что класс действительно есть, и это был первый ответ модели.

Топ5-Ошибка — это то, что попало в топ 5 ответов. В нашем случае качество этой модели составляло 17%.

Давайте посмотрим на изображение внизу. Это картинки, которые максимизируют отклик сверточных фильтров 11*11 на первом слое. Как можно увидеть, это различные линии и какие-то осмысленные геометрические шаблоны.



> VGG (2014)

В 2014 году появляется статья **Very deep convolutional networks for large-scale image recognition**. Модель называется так, потому что авторы статьи работали в

компании Visual Geometry Group. У нее было несколько конфигураций - A, B, C, D, E и по мере роста букву она увеличивалась в размере.

На картинке снизу можно посмотреть на архитектуру этой нейроной сети, количество параметров.

Как можно заметить у самой большой модели 144 млн параметров, т.е. это в 2 раза больше чем у AlexNet. И соответственно отказались от идеи делать большие свертки, и это дало в итоге, меньше параметров, и больше нелинейности, и у нейросети появилась возможность предсказывать что-то более сложное. Модель обучалось с помощью градиентного спуска с инерцией. Для двух полносвязных слоев использовался dropout, но важно отметить, что все это не обучалось с нуля, E вариант был достаточно массивен и из-за этого модель не могла сойтись.

Поступили следующим образом, сначала обучалась модель A, она была в состоянии сойтись, потом добавлялся сверточный слой и т.д. , логика этого решения выделена жирным на картинке ниже. В итоге получилось обучить нейросеть из 19 слоев. Качество ее было лучше чем у **AlexNet** и на топ-5 составляло 8%. Причем 8% это не самый лучший вариант, потом модель показала качество немного лучше. Архитектуру можно посмотреть на картинках ниже.

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					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Далее на картинке можно посмотреть качество различных конфигураций это модели.

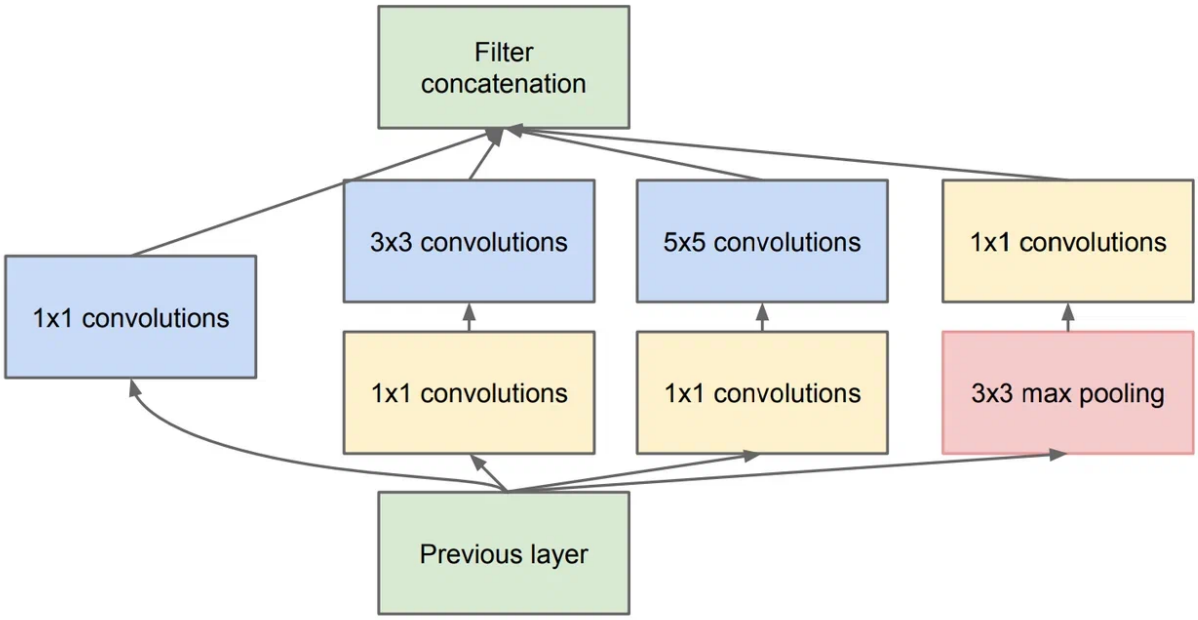
Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

> GoogLeNet (2014)

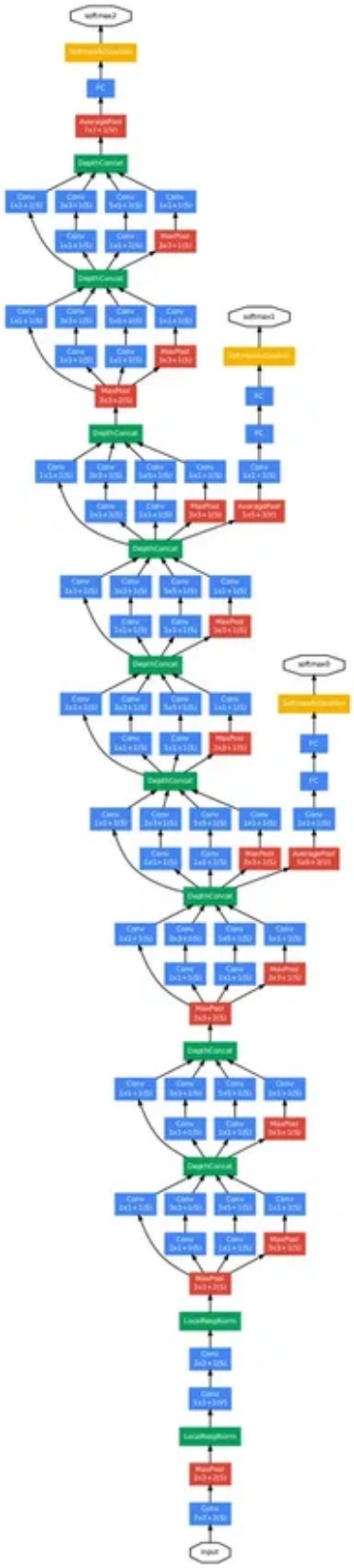
В 2014 году исследователями компании Google **в статье Going deeper with convolutions** была представлена новая архитектера нейронных сетей, которая получила название **GoogLeNet** и еще как ее называют по другому **Inception**, в честь одноименного фильма.

Что происходило внутри нейросети **Inception** можно посмотреть на картинке снизу.



Основной идее является то, что у нас есть блок Inception и в него приходит предыдущий слой, и он подается на вход сверткам 1*1, которые уменьшают количество каналов. То что мы можем назвать сверткой 1*1 - у нас есть изображение и мы берем 1 пиксель, суммируем его по всем каналам и это будет являться выходом, у изображения не меняется размер, но у него по сути происходит суммирование всех каналов для одного конкретного пикселя с какими-то весами, следующего пикселя, всех каналов с определенными весами. Соответственно мы можем сделать новое количество каналов просто просуммировав все каналы которые здесь были, и сделать условно так - было C каналов стало B каналов, которых меньше. rКак мы видим на картинке выше все результаты конкатенируются по каналам и получается новая картинка. Идея была в том, что перед большими свертками нужно уменьшить количество каналов, и это позволяет облегчить нейросеть и сделать ее немного глубже.

На картинке ниже можно посмотреть ее архитектуру.



Можно заметить на ней ответвления, которые помогают облегчить течение градиента, что позволяет модели обучаться лучше. Модель обучалась на градиентном спуске с инерцией и ее **Топ5-Ошибка** на **ImageNet** составляла **6.66%** , и ее качество лучше, чем представленная в том же году **VGG**.

> Resnet (2015)

Это очень популярная архитектура про которую можно сказать, что она поменяла мир, она идет наравне с batch-нормализацией. Статья в которой она была представлена называлась **Deep Residual Learning for Image Recognition**. Она была создана в Microsoft Research. В чем заключалась идея можем посмотреть на картинке снизу.

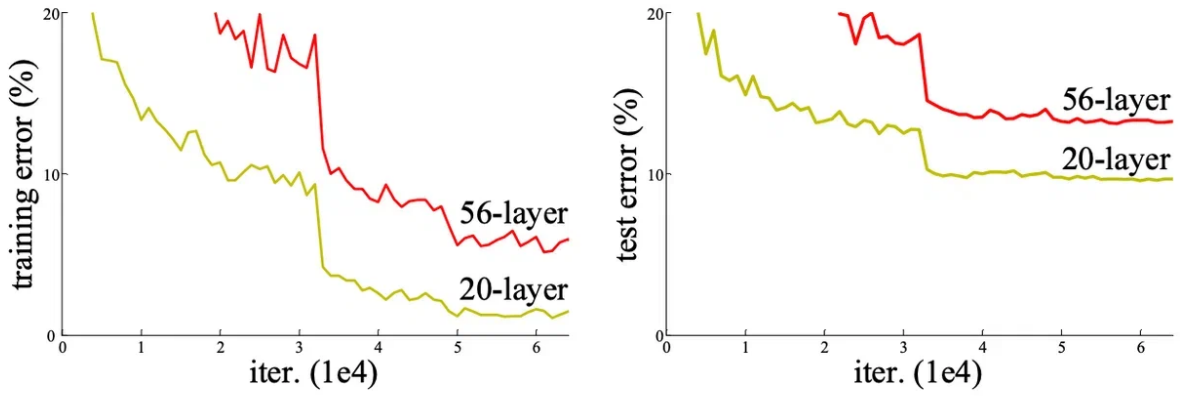


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Может показаться все то что мы видели до этого, нейронные сети все росли и росли, например VGG с 19 слоями, и что можно обучить совсем огромную нейронную сеть и качество должно быть феноменальное, и просто не хватало ресурсов на тот момент развития. Но оказалось если посмотреть на картинку вверх, слева можно увидеть, что в итоге на обучающей выборке менее глубокая

нейросеть 20 слоями, в ней значительно меньше параметров, чем в нейросети с 56 параметрами. На обучающей выборке более глубокая нейросеть не может обучиться до такого же качества, не может переобучиться, и естественно на тесте у нее тоже хуже качество, т.е. она мало того, что не может обучиться, подогнаться под обучающую выборку, так она и на тесте не может ничего показать.

Соответственно авторы заметили, что есть такой феномен, и что в определенный момент глубокие нейросети перестают побеждать менее глубокие. Т.е. получается, что сначала с увеличение глубины качество растет, а потом начинает падать обратно. И проблема в том, что нейросеть не может обучиться, не протекает градиент, она настолько глубокая, что градиент где-то застревает. Далее на картинке снизу можно посмотреть, что придумали исследовали из Microsoft.

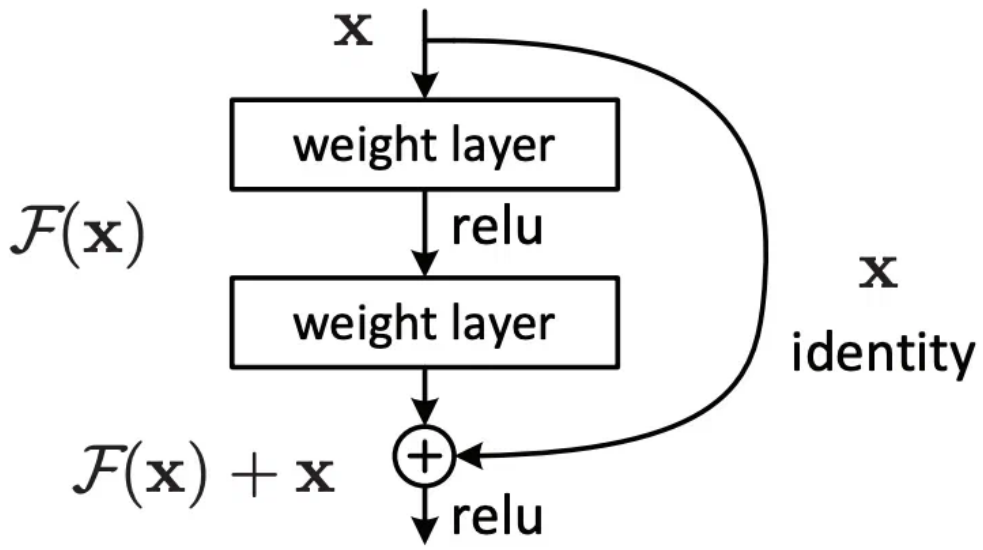
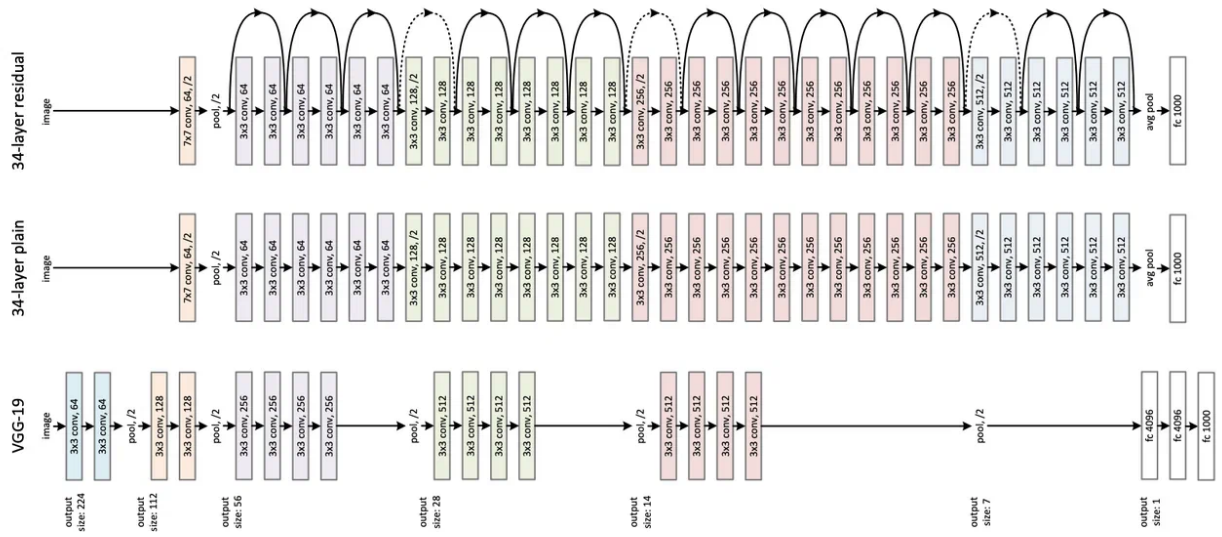


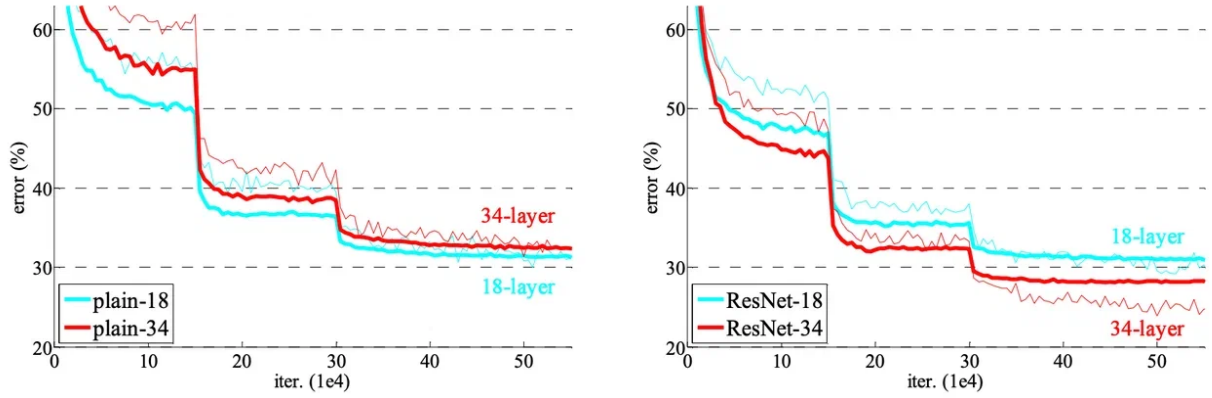
Figure 2. Residual learning: a building block.

Посмотрим на картинку выше представим, что есть некоторый блок. Он называется skip connection. У нас есть слой с параметрами, это сверточный слой, и еще один сверточный слой и между ними relu, и перед началом на вход подается x , и проходит через эти 2 слоя, и потом когда мы получаем f(x) мы прокидываем наш изначальный x вниз, и все это суммируем.

Из этого получается, что градиент может протечь свободно, x не меняется, и это дает возможность увеличивать размер нейронных сетей просто до бесконечности. Как выглядит архитектура этой нейросети можно посмотреть на картинке ниже.



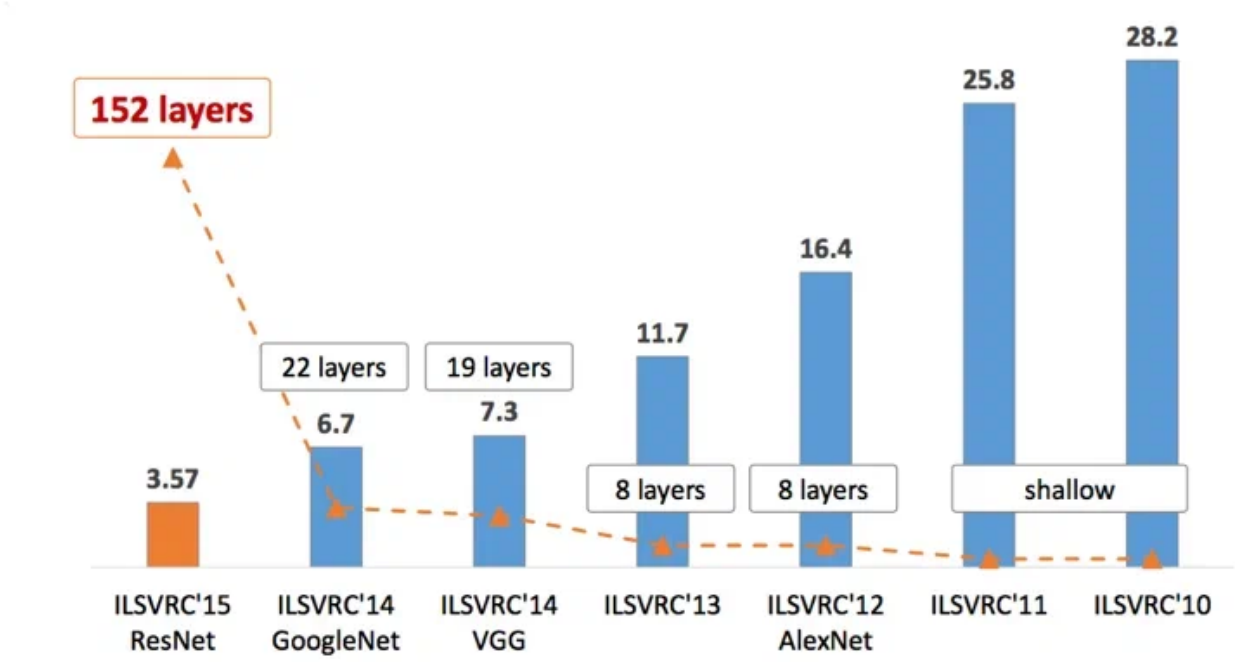
Как можно увидеть она состоит из 34 residual слоев. можно заметить, что когда мы прокидываем skip connections у нас получается, что x доходит в неизменном виде, т.е. по сути x не меняется, подается с какой-то добавкой от сверточных слоев. И получается, что каждый слой получает каждый раз свежую добавку, а не перемалывает только ту информацию, которую ему дают предыдущие слои, а он получает информацию от предыдущих слоев плюс какую то информацию со входа, и это дает существенный прирост качества. На картинки снизу можно слева можно увидеть нейросеть без skip connetions, а с права с ними.



Как можем увидеть, что с применением skip connetions качество существенно возросло. Также как указано в статье был проведен эксперимент со skip connetions на 1 тыс. слоев. Эта нейросеть смогла обучиться, она сошлась, но качество на тестовой выборке было неприемлемым, нейросеть сильно переобучилась, но была решена проблема, что огромные нейросети с какого-то момента становились хуже чем менее глубокие.

Модель обучалась с нуля при помощи градиентного спуска с инерцией, состояла из 152 слоев и ее качество на **ImageNet** было Топ5-Ошибка 4.5%. При этом начиная с **ResNet** с 2015 года, нейросети показывают лучшее качество чем человек на этом дасасете.

На картинке снизу можно посмотреть график лучшего качества на датасете **ImageNet**



С 2015 года прошло достаточно времени, появились новые архитектуры на основе рассмотренных нами ранее, на основе других новых идей, сейчас рассматриваться не будут, потому что у нас вводный курс.

> Transfer Learning

Давайте рассмотрим такой концепт как **transfer learning**. В качестве примера представим, что у нас есть датасет **ImageNet**, на который было потрачено очень много денег, времени на то чтобы с ним можно было работать, обучать модели, чтобы придумывать архитектуры. Соответственно у нас годы инкрементальных улучшений, идей, и у нас есть новая задача, например нам нужно классифицировать кошек, собак или марки автомобилей, все что угодно.

У нас появилась новая задача и не очень много новых данных, не хватает времени, и тут может возникнуть вопрос почему бы не переиспользовать различные архитектуры. Это можно сделать - оказывает можно взять модель, например с того же **ImageNet**. Она обучена на миллионе картинок, берем ее, и можем заменить в ней какие-то последние слои, например полносвязные, на новые полносвязные слои, сделав новый классификатор, и у нас наверняка поменялось число классов, нужно просто поменять на новый классификатор с нужным числом выходов, либо обучить только его, а остальную модель заморозить, либо произвести небольшие взаимодействия с моделью и классификатор дообучить.

По большей части обычно это делает так - есть сверточная нейронная сеть, первые несколько слоев жестко фиксируются, потом следующие слои немного дообучаются с маленькой длинной шага и последний классификатор обучается как нормальная полносвязная нейронная сеть.

Понятна идея, что представления с первых слоев, различные геометрические представления, пример можно посмотреть в начале конспекта, не похожи на части объектов, что-то человеческое оттуда вычленить невозможно, по большей части они одинаковы для всех фотографий, всех доменов, всех разных задач, и что и можно оставить и зафиксировать, т.е. первые несколько сверточных слоев оставить как есть, а вот дальше, последние сверточные слои, полносвязные слои можно и поменять, поучить. Чем задачи более непохожи, например была **ImageNet**, а стала классификация подводных лодок, которые все более менее одинаковы, нам придется сильно переучивать модель, соответственно больше слоев размораживать, больше слоев обучать. Соответственно чем ближе слой к началу, чем меньше мы делаем длину шага, и в конце получается самая большая длина, т.е. чем ближе к началу тем меньше доучивать. Далее это будет рассмотрено на практике.