

# Глубокое обучение и вообще

Ульянкин Филипп

**Посиделка 8:** Современные свёрточные архитектуры, transfer learning

# Agenda

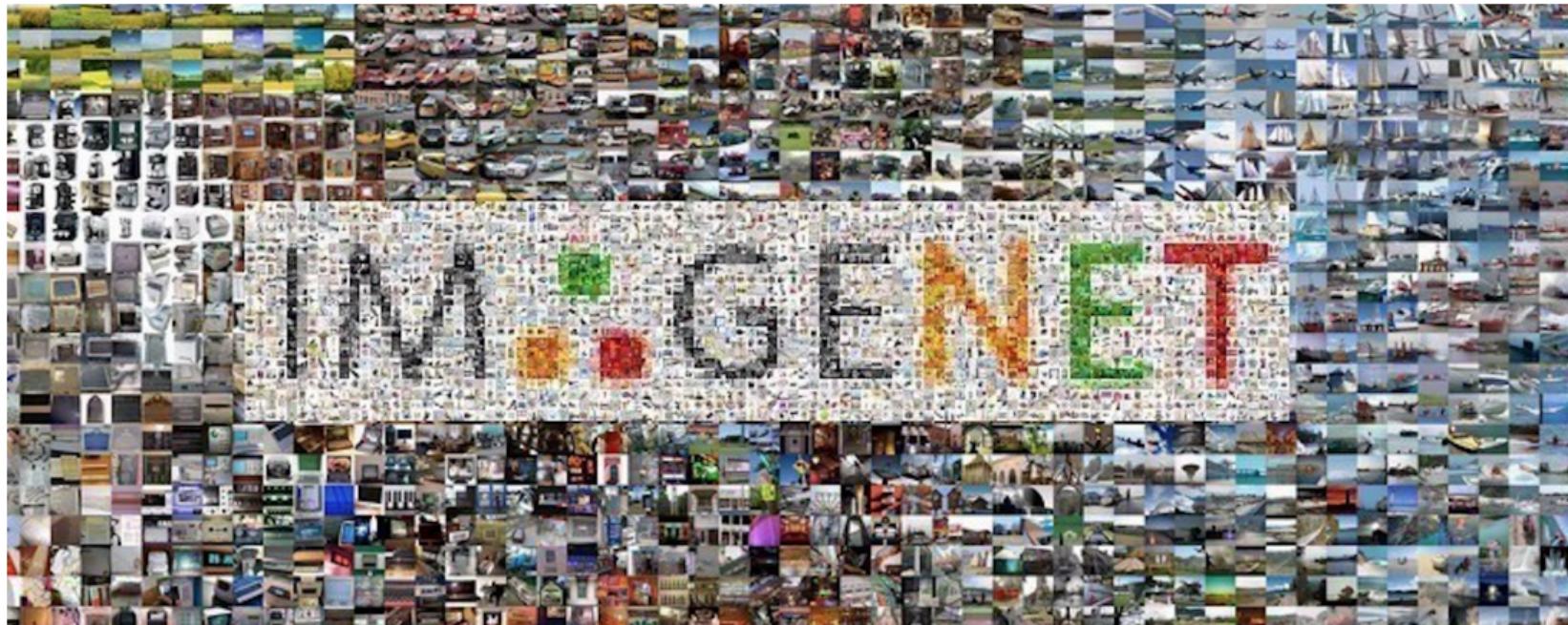
- Сказ о том, как люди ImageNet рвали
- Современные свёрточные архитектуры
- История про котейку по имени Метрика
- Перенос знаний (Transfer learning)

# Сказ о том, как люди ImageNet рвали



# ImageNet

Около 15 миллионов изображений размеченных на ~ 22 тысячи категории



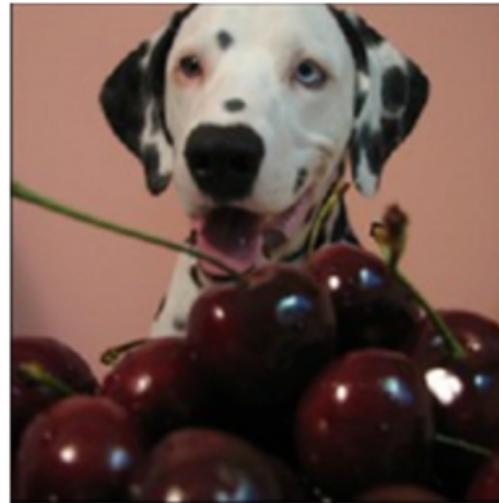
<http://www.image-net.org/>

# ImageNet

- Выборка очень большая и неоднородная, постоянно пополняется
- Соревнования на ней проводятся каждый год
- Обычно изображение требуется отнести к одному из 1000 классов
- Если один из пяти предсказанных вариантов оказался верным, то классификация считается верной
- До 2012 года лучшие алгоритмы дают ошибку в 25%
- В 2012 году на арену выходят глубокие нейронные сети

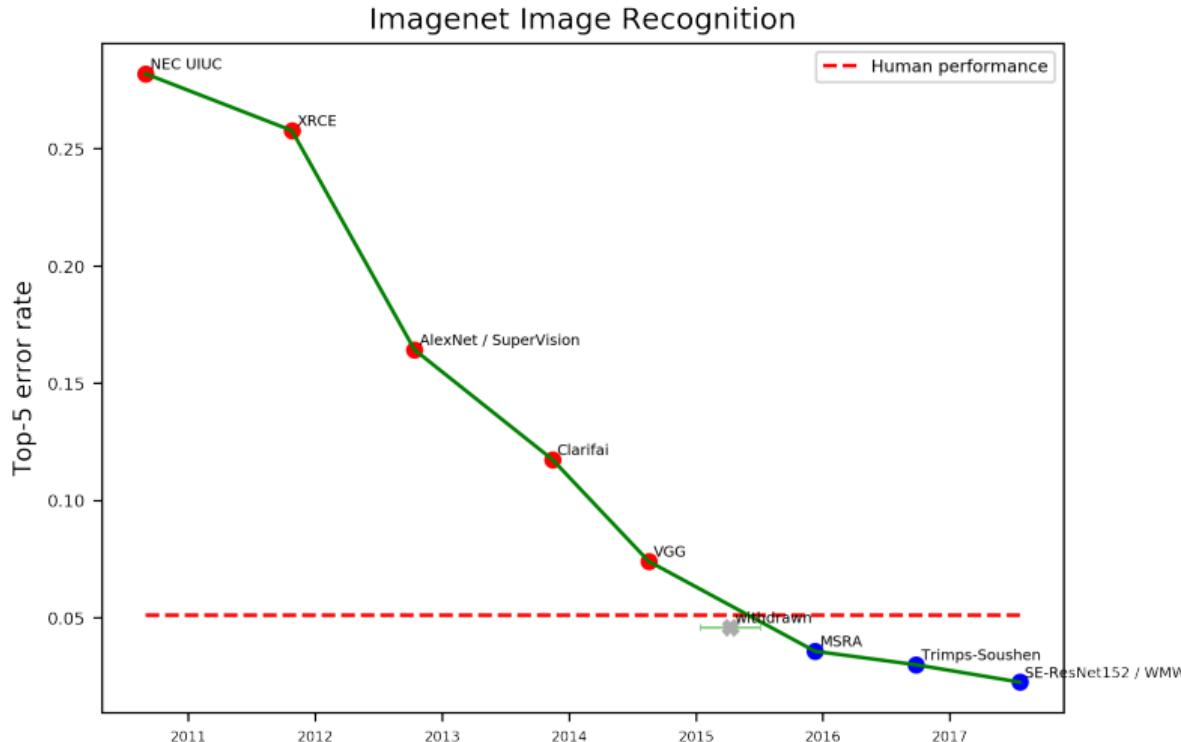
# ImageNet

- Бывают спорные изображения: тут вишня, если распознать как далматинец, будет неправильно

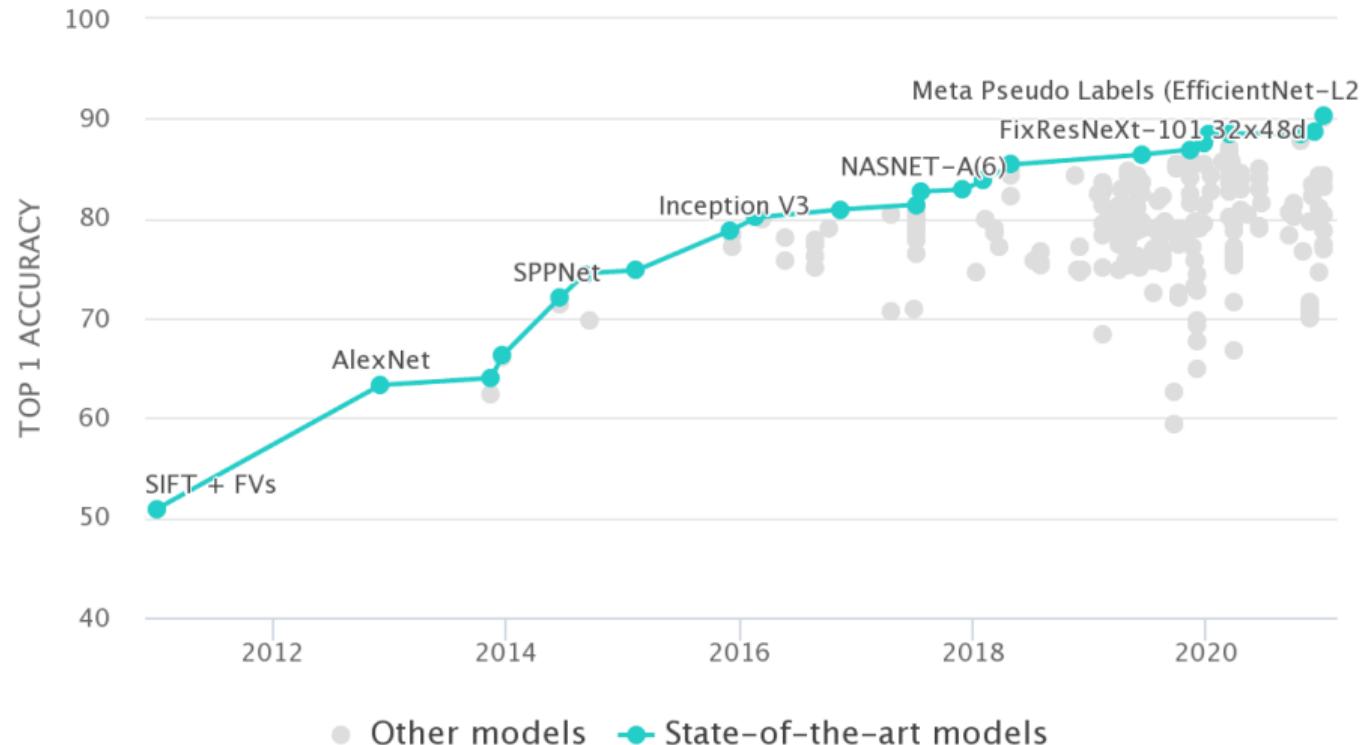


Можно попробовать сразиться с компьютером: <https://cs.stanford.edu/people/karpathy/ilsvrc/>

# Точность сетей на ImageNet (Top-5 error rate)



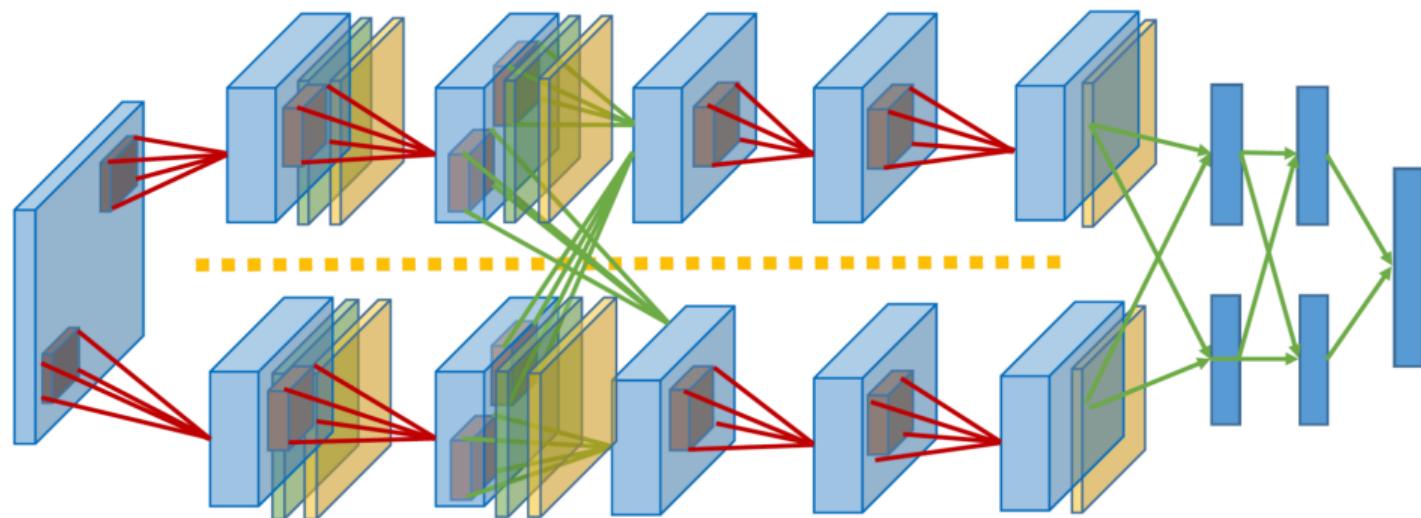
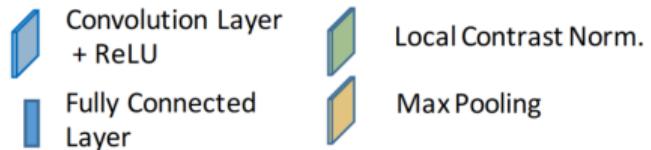
# Точность сетей на ImageNet (Top-1 accuracy)



# AlexNet



# AlexNet (2012)



<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

## AlexNet (Krizhevsky et al, 2012)

- Использовали ReLU, Dropout, кастомную нормализацию (не батчнорм), аугментацию
- Использовался градиентный спуск с инерцией (momentum), размер батча 128, изначальная скорость обучения  $1e - 2$ , делилась руками на 10, когда точность на валидации выходила на плато
- Свёртки  $11 \times 11$ ,  $5 \times 5$ ,  $3 \times 3$
- Около 60 миллионов параметров, училась 5 – 6 суток на 2 GPU
- Уронила ошибку с 26% до 16.4%
- В 2013 Zeiler и Fergus потюнили гиперпараметры AlexNet и уронили ошибку до 11.7% (ZFNet)

# Аугментация (Data augmentation)

- Увеличение, уменьшение
- Повороты, искажения, приближения
- Новые цвета, затемнения
- Смена стиля
- Нужно ли добавлять сдвиги?



[https://www.tensorflow.org/tutorials/images/data\\_augmentation](https://www.tensorflow.org/tutorials/images/data_augmentation)

# Аугментация (Data augmentation)

- Увеличение, уменьшение
- Повороты, искажения, приближения
- Новые цвета, затемнения
- Смена стиля
- Нужно ли добавлять сдвиги?  
**(нет, вместо них лучше пулинг)**



[https://www.tensorflow.org/tutorials/images/data\\_augmentation](https://www.tensorflow.org/tutorials/images/data_augmentation)

# Аугментация (Data augmentation)

- Много разных вариантов
- «Бесплатное» расширение обучающей выборки
- **Обучение:** случайно применяем к картинкам из текущего батча
- **Тест:** делаем несколько аугментаций картинки, применяем сеть к каждой и усредняем предсказания
- Для тестовых данных набор аугментаций всегда фиксирован

# ImageNet augmentations

```
import torchvision.transforms as T

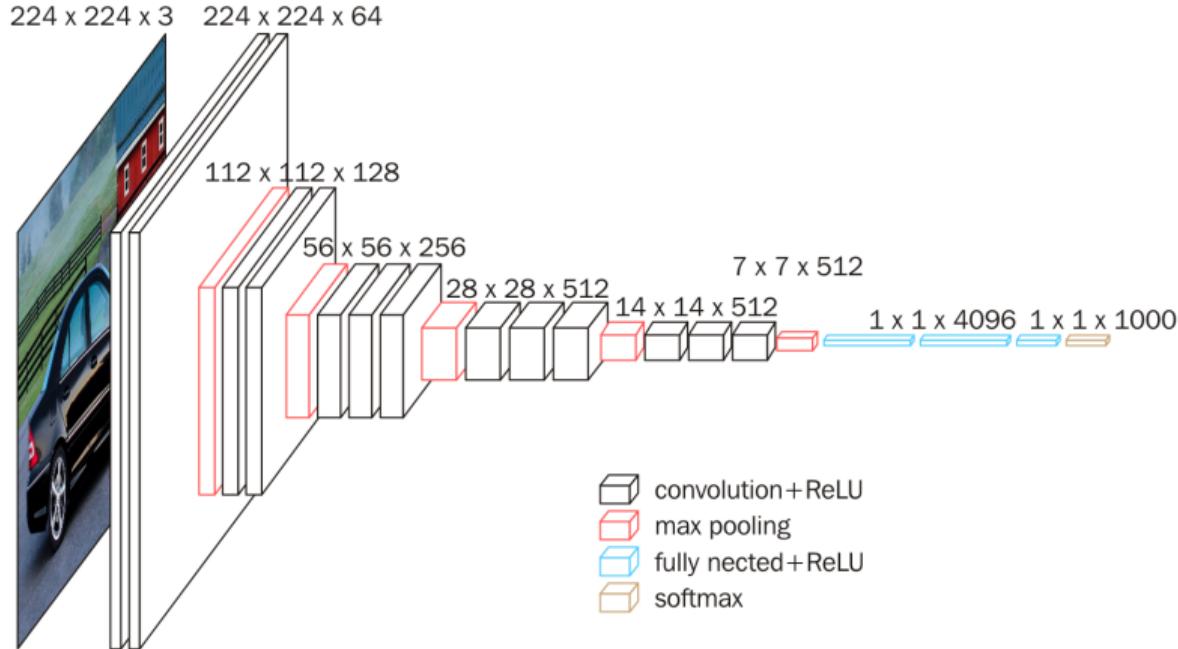
train_transform = T.Compose([
    T.RandomResizedCrop(224),
    T.RandomHorizontalFlip(),
    T.ToTensor(),
    T.Normalize(mean=[0.485, 0.456, 0.406],
               std=[0.229, 0.224, 0.225])
])

test_transform = T.Compose([
    T.Resize(256),
    T.CenterCrop(224),
    T.ToTensor(),
    T.Normalize(mean=[0.485, 0.456, 0.406],
               std=[0.229, 0.224, 0.225])
])
```

# Visual Geometry Group (VGG)



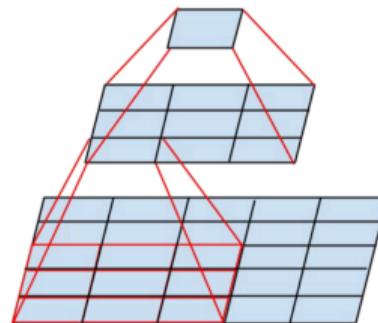
# VGG (2014)



<https://arxiv.org/pdf/1409.1556.pdf>

# VGG (2014)

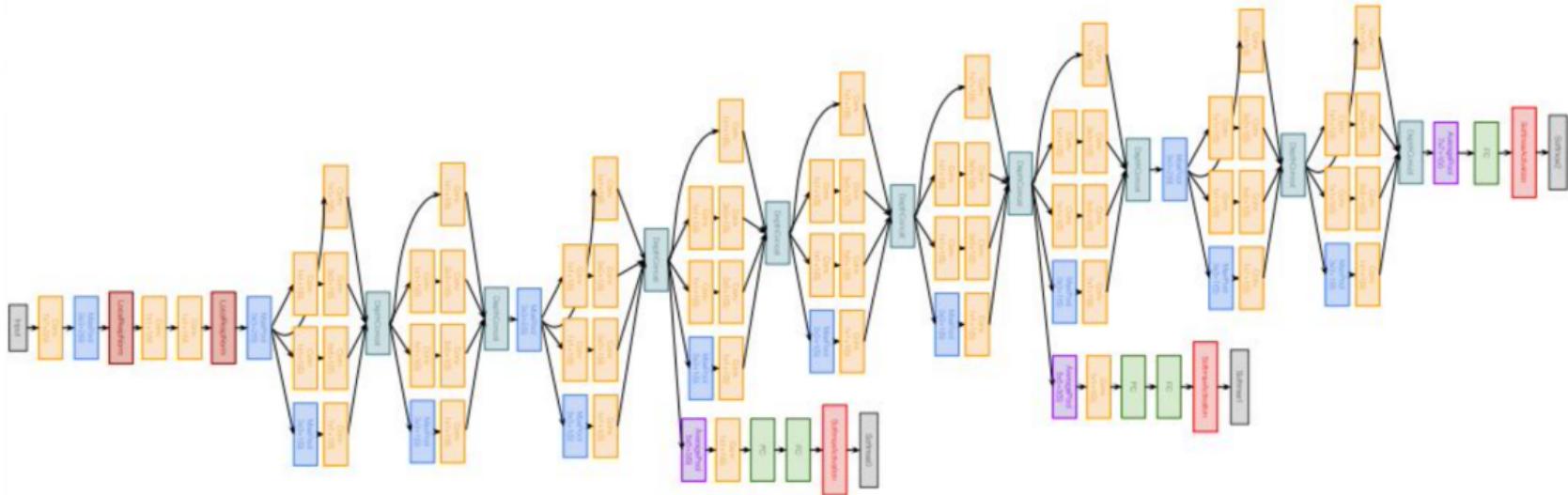
- Свёртки только  $3 \times 3$ , но намного больше фильтров
- Экономия параметров (две свёртки  $3 \times 3$  покрывают поле  $5 \times 5$  и требуют 18 параметров, а свёртка  $5 \times 5$  требует 25 параметров)
- Дропаут для первых двух полно связных слоёв, Momentum, хитрая инициализация
- 138 миллионов параметров, училась 2 – 3 недели на 4 GPU
- Ошибка упала до 7%



# GoogleLeNet aka Inception



# GoogleLeNet aka Inception V1



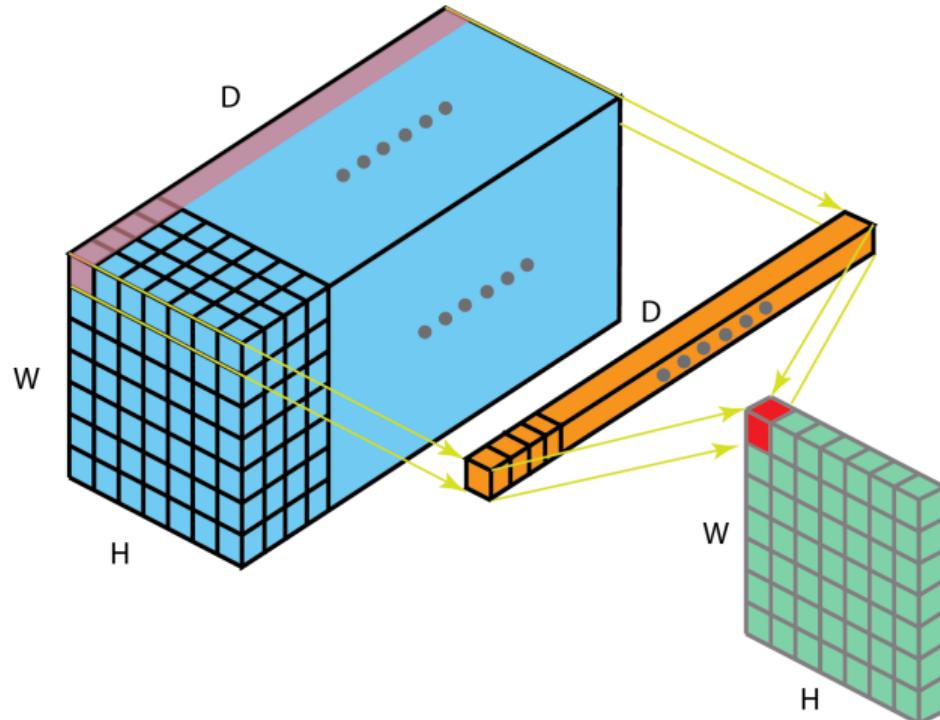
<https://arxiv.org/abs/1409.4842>

# GoogleLeNet aka Inception V1 (2014)

- Уменьшаем свёртки, саму **сетку собираем из специальных inception-блоков** (9 блоков)
- используются **свёртки  $1 \times 1$**  для уменьшения числа каналов перед "тяжёлыми" свёртками
- **Несколько дополнительных классификаторов на разных уровнях**, идея в том, что такие классификаторы позволяют «протолкнуть» градиенты к ранним слоям и тем самым уменьшить эффект затухания градиента
- Параметров в 10 раз меньше, чем в AlexNet, ошибка упала до 6.7%

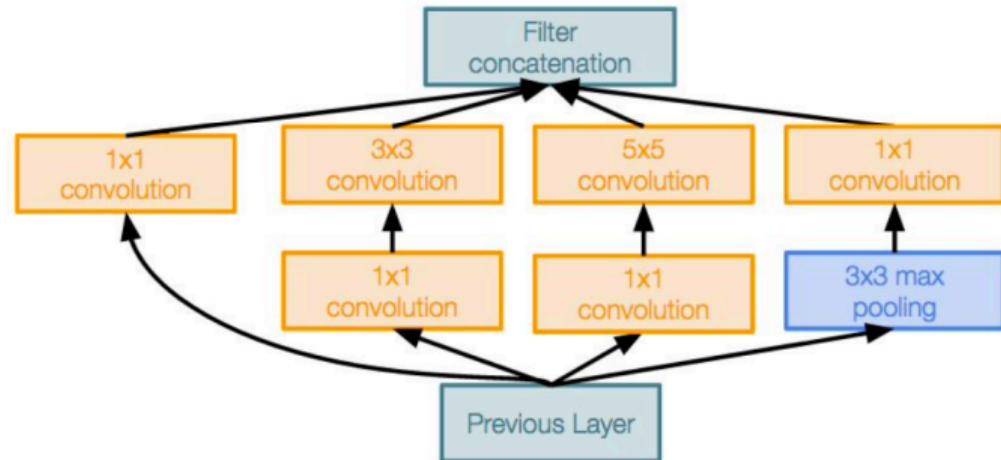
# Свёртка $1 \times 1$

- Позволяет сократить размерность по числу каналов
- Представляет из себя полносвязный слой по фильтрам
- В Inception мы делаем много разных свёрток, а потом свёрткой  $1 \times 1$  выбираем из них лучшее, агрессивно понижая размерность



# GoogleLeNet aka Inception V1 (2014)

- На каждом слое используется ни одна свёртка, а несколько разных, что помогает реагировать на сигналы разного масштаба
- Свёртка  $1 \times 1$  делает вычисления легче



<https://arxiv.org/abs/1409.4842>

# GoogleLeNet aka Inception V1

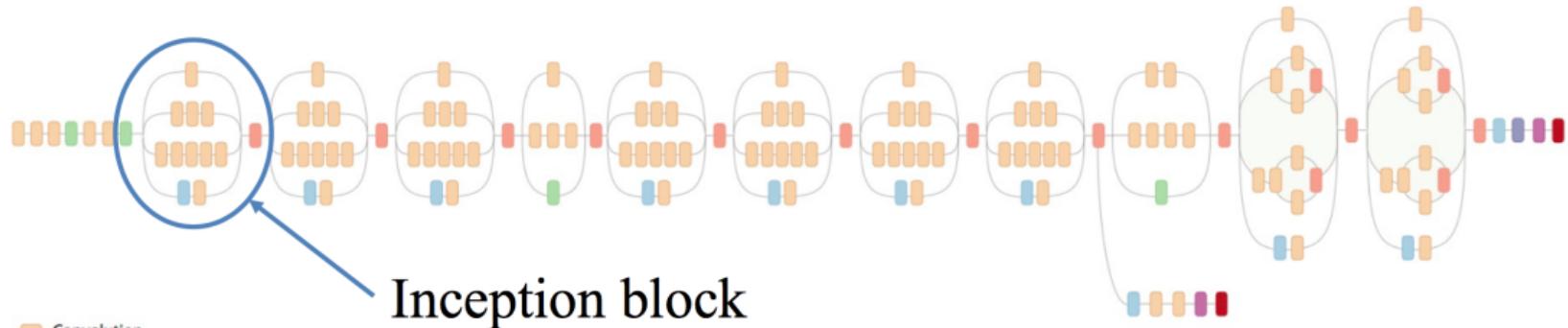
## 10 Acknowledgements

We would like to thank Sanjeev Arora and Aditya Bhaskara for fruitful discussions on [2]. Also we are indebted to the DistBelief [4] team for their support especially to Rajat Monga, Jon Shlens, Alex Krizhevsky, Jeff Dean, Ilya Sutskever and Andrea Frome. We would also like to thank to Tom Duerig and Ning Ye for their help on photometric distortions. Also our work would not have been possible without the support of Chuck Rosenberg and Hartwig Adam.

## References

- [1] Know your meme: We need to go deeper. <http://knowyourmeme.com/memes/we-need-to-go-deeper>. Accessed: 2014-09-15.
- [2] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. *CoRR*, abs/1310.6343, 2013.
- [3] Ümit V. Çatalyürek, Cevdet Aykanat, and Bora Uçar. On two-dimensional sparse matrix partitioning: Models, methods, and a recipe. *SIAM J. Sci. Comput.*, 32(2):656–683, February 2010.

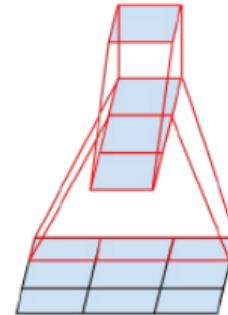
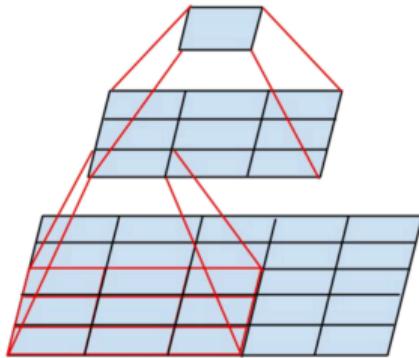
# Inception V3 (2015)



- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax

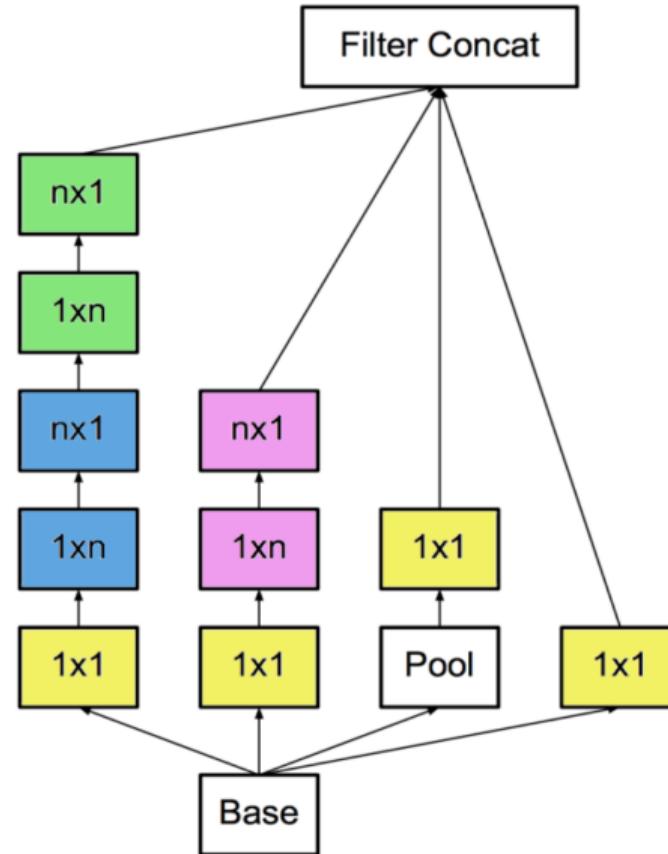
<https://arxiv.org/abs/1512.00567>

## Свёртка $n \times 1$ и $1 \times n$



- В VGG заменяли большие свёртки на последовательные  $3 \times 3$
- Пойдём дальше и заменим их на последовательные  $3 \times 1$  и  $1 \times 3$
- Экономим ещё больше параметров, для каждой свёртки 6 вместо 9

# Inception V3 (2015)

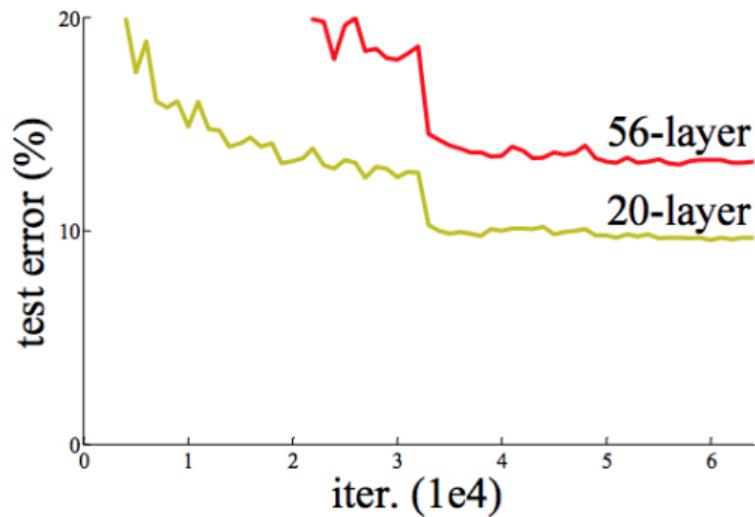
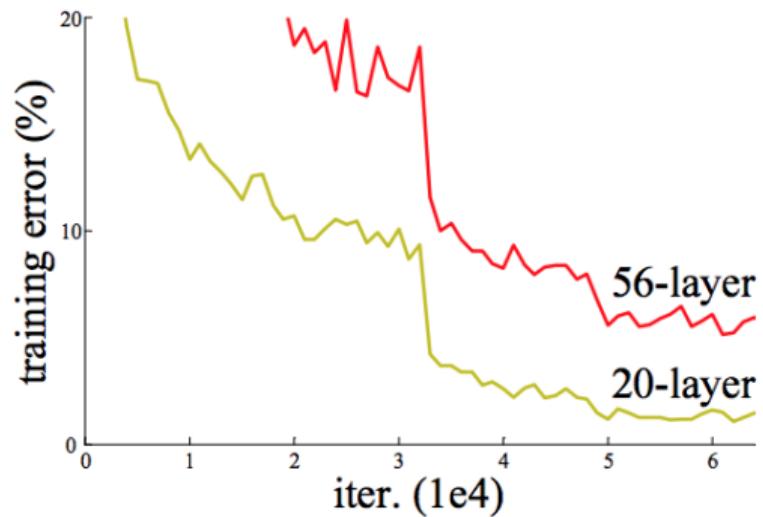


## Inception V3 (2015)

- Сетка собирается из 11 inception layers, добавили BatchNorm (то же самое без него было в Inception V2)
- В результате исследований сформулировали **принципы обучения глубоких свёрточных сетей**:
  1. Избегайте representation bottlenecks, нельзя резко снижать размерность слоя, это надо делать плавно: от начала к концу
  2. Свёртку нужно разбивать на более мелкие части для экономии ресурсов и увеличения размера сетки
  3. Нельзя резко увеличивать глубину, забивая на ширину, надо растить сбалансированно
- Итоговое качество 4.2% ошибок, ансамбль из 4-х моделей дал 3.8%.

# ResNet

# ResNet (Microsoft) (2015)



<https://arxiv.org/abs/1512.03385>

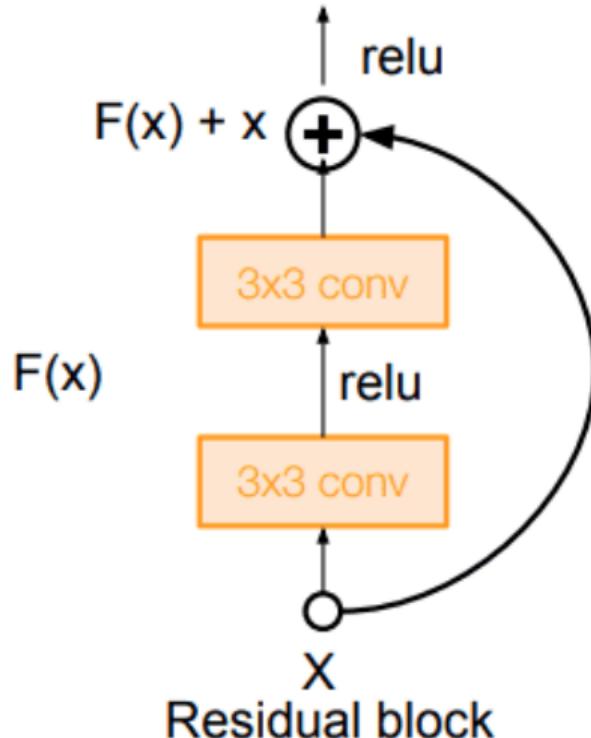
## ResNet (Microsoft) (2015)

- Добавление слоёв в свёрточную сеть ухудшает качество даже на обучении
- Хотя возможностей для переобучения больше, сеть почему-то не может ими воспользоваться
- **Интерпретация эффекта:** слои инициализированы шумом, если какой-то один слой не натренирован, он убивает работу сети, через него не проходит полезный сигнал
- Чем больше слоёв, тем более ярко выражен этот эффект

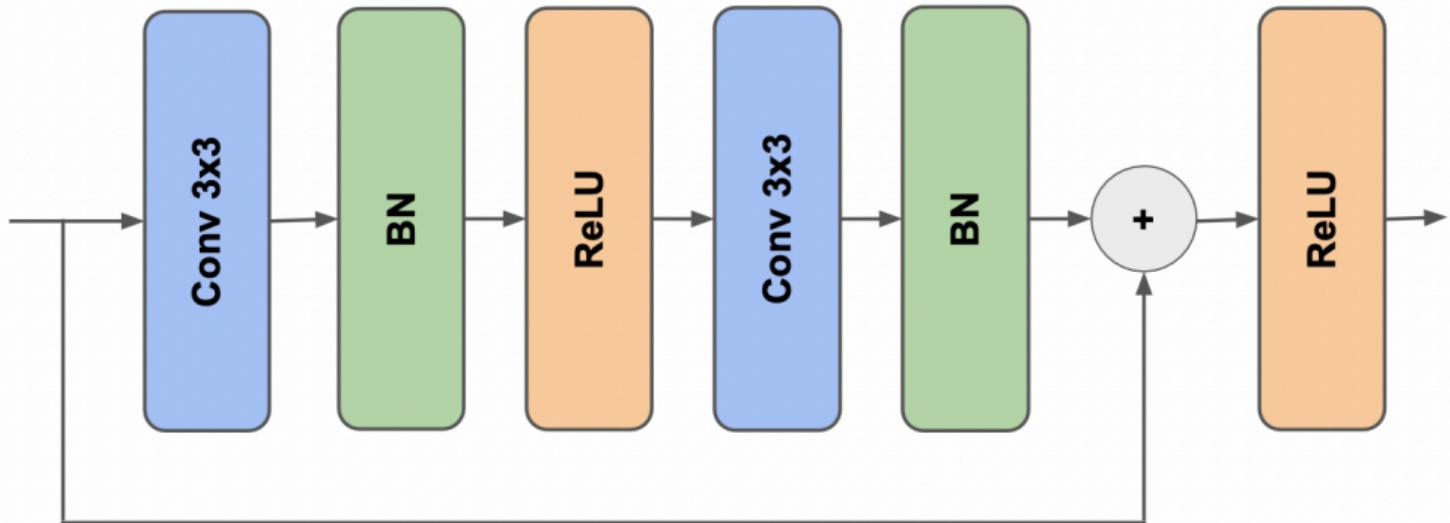
# ResNet (Microsoft) (2015)

- **Решение проблемы:** Будем посыпать вход на выход и давать слою возможность немного его подправить (residual слой)
- Идея чем-то похожа на бустинг, сеть сама решает когда заканчивать подправлять выходы (грубо говоря, сама выбирает глубину)

# ResNet (Microsoft) (2015)

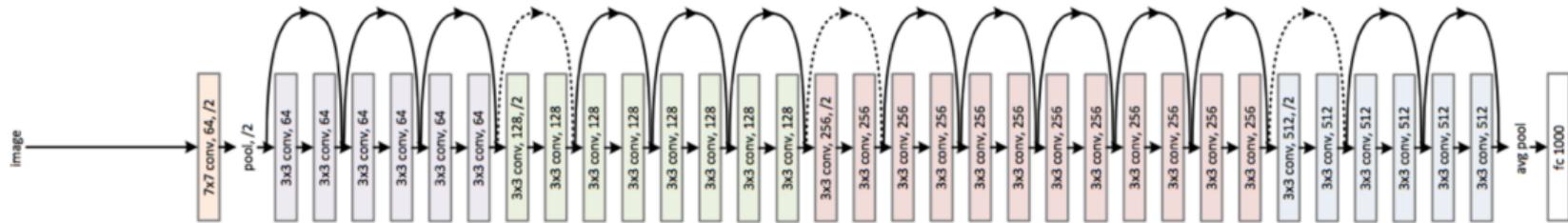


# Residual Block



<https://arxiv.org/abs/1512.03385>

# ResNet (Microsoft) (2015)



- 152 слоя, ошибка составила 3.6%

<https://arxiv.org/abs/1512.03385>

## ResNet (Microsoft) (2015)

- **Идея:** более глубокие уровни должны улавливать разницу между новым и тем, что было раньше
- Ключевым элементом архитектуры является связь, которая пропускает несколько слоёв, передавая результат предыдущего слоя
- Используется инициализация Ксавье, батч нормализация после каждого свёрточного слоя
- Обучение с помощью Momentum SGD, стартовая скорость 0.1, уменьшается в 10 раз при выходе ошибки на тесте на плато, размер батча 256

<https://arxiv.org/abs/1512.03385>

# ResNet (Microsoft) (2015)

MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: “*Ultra-deep*” (quote Yann) **152-layer** nets
  - ImageNet Detection: **16%** better than 2nd
  - ImageNet Localization: **27%** better than 2nd
  - COCO Detection: **11%** better than 2nd
  - COCO Segmentation: **12%** better than 2nd

<https://arxiv.org/abs/1512.03385>

# Why skip-connections ?

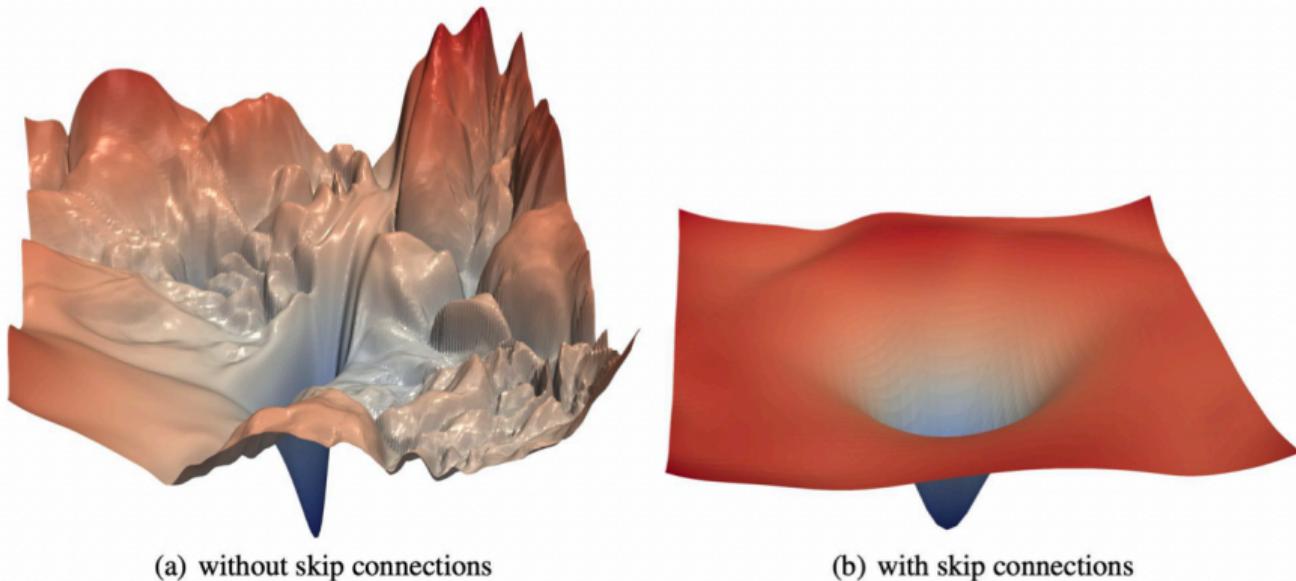


Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

<https://arxiv.org/abs/1712.09913>

# Что было дальше?



# Inception-Resnet (2016)

- Совместили две идеи вместе
- Судя по всему, виды слоёв перебирали автоматически, но не палятся
- Ансамблем Google поставил новый рекорд 3.08%

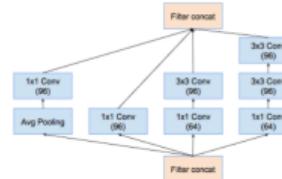


Figure 4. The schema for  $35 \times 35$  grid modules of the pure Inception-v4 network. This is the Inception-A block of Figure 9.

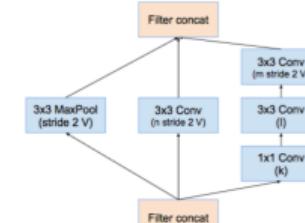


Figure 7. The schema for  $35 \times 35$  to  $17 \times 17$  reduction module. Different variants of this blocks (with various number of filters) are used in Figure 9 and 15 in each of the new Inception(-v4, -ResNet-v1, -ResNet-v2) variants presented in this paper. The k, l, m, n numbers represent filter bank sizes which can be looked up in Table 1.

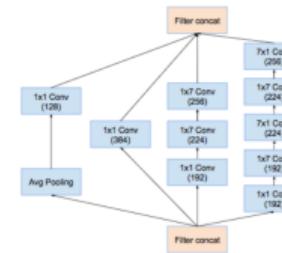


Figure 5. The schema for  $17 \times 17$  grid modules of the pure Inception-v4 network. This is the Inception-B block of Figure 9.

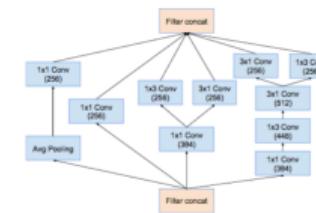


Figure 6. The schema for  $8 \times 8$  grid modules of the pure Inception-v4 network. This is the Inception-C block of Figure 9.

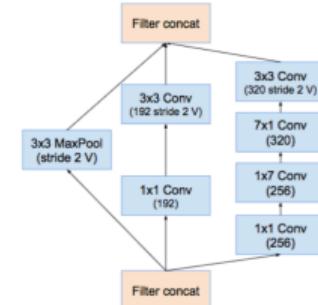
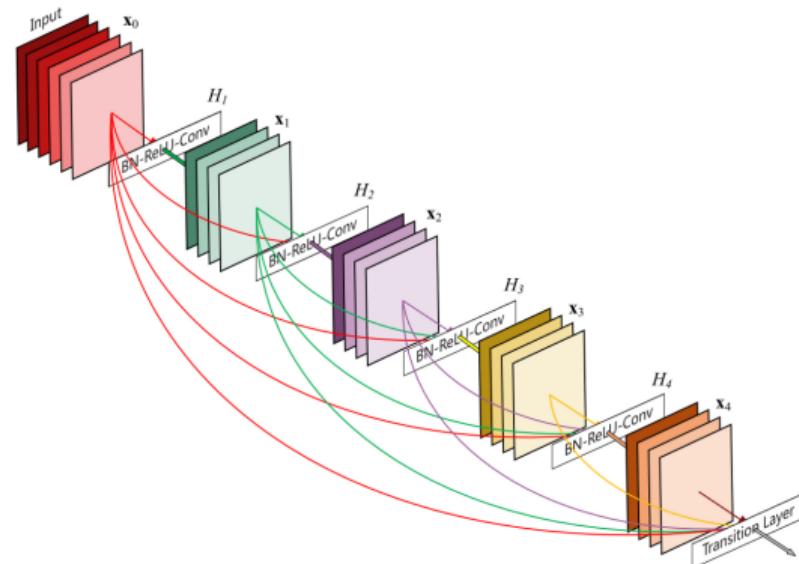


Figure 8. The schema for  $17 \times 17$  to  $8 \times 8$  grid-reduction module. This is the reduction module used by the pure Inception-v4 network in Figure 9.

# DenseNet (2018)

- Давайте всё соединим со всем
- Вместо сложения в skip-connection будем делать конкатенацию
- У каждого слоя берут мало фильтров
- Параметров меньше, чем у ResNet



<https://arxiv.org/abs/1608.06993>

## Хотим компактности

Нейросеть должна быть компактной и влезать на разные устройства (телефоны, IoT)

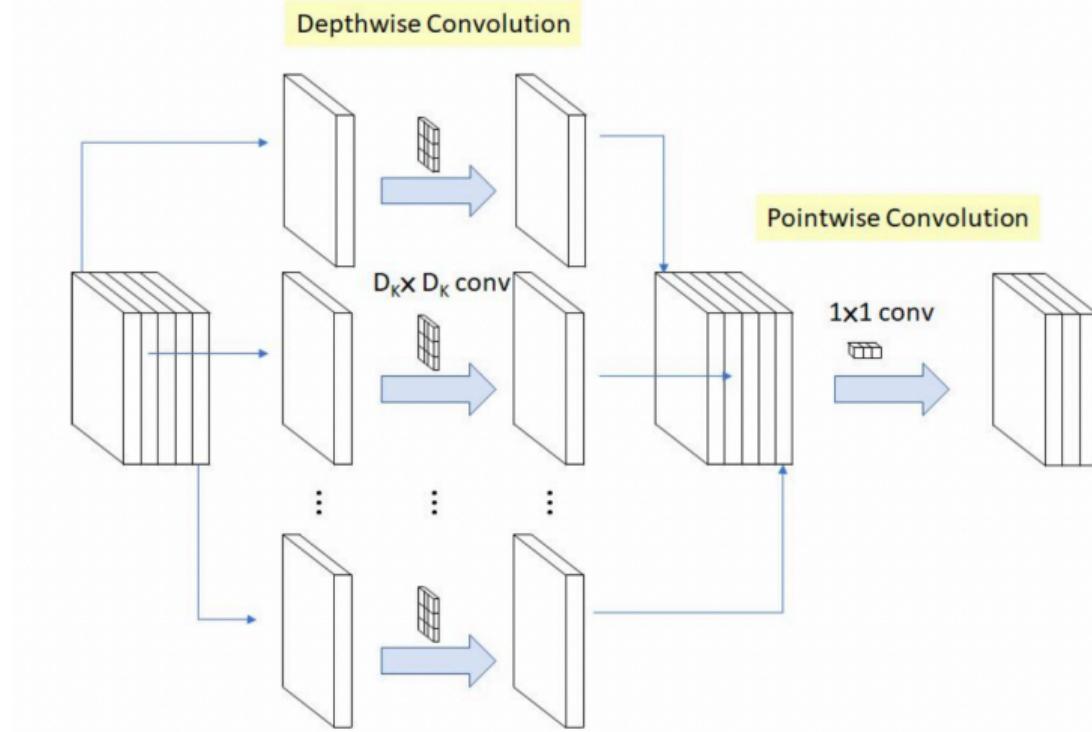
Год	Архитектура	Примерный вес
2012	AlexNet	240 MB
2014	VGG-19	549 MB
2015	ResNet-50	98 MB
2015	ResNet-152	232 MB
2015	Inception v3	92 MB
2016	Inception-Resnet	215 MB

Хотелось бы 20 MB максимум

Веса указаны примерно: <https://habr.com/ru/company/mipt/blog/450732/>

# MobileNet (2017)

- Легкая архитектура для мобильных устройств
- Комбинирует depthwise и pointwise convolution
- Есть просадка по качеству, но сильно уменьшаются требования по времени и памяти



<https://arxiv.org/abs/1704.04861>

# Несвёрточные архитектуры

- Текущие SOTA (state-of-the-art) модели носят не свёрточные
- Современные SOTA модели - трансформеры
- Свёртки всё ещё часто используются, они быстрее работают и проще обучаются
- Например, в январе 2022 вышла ConvNext, которая работает не хуже трансформера

<https://arxiv.org/abs/2201.03545>

# История про Метрику

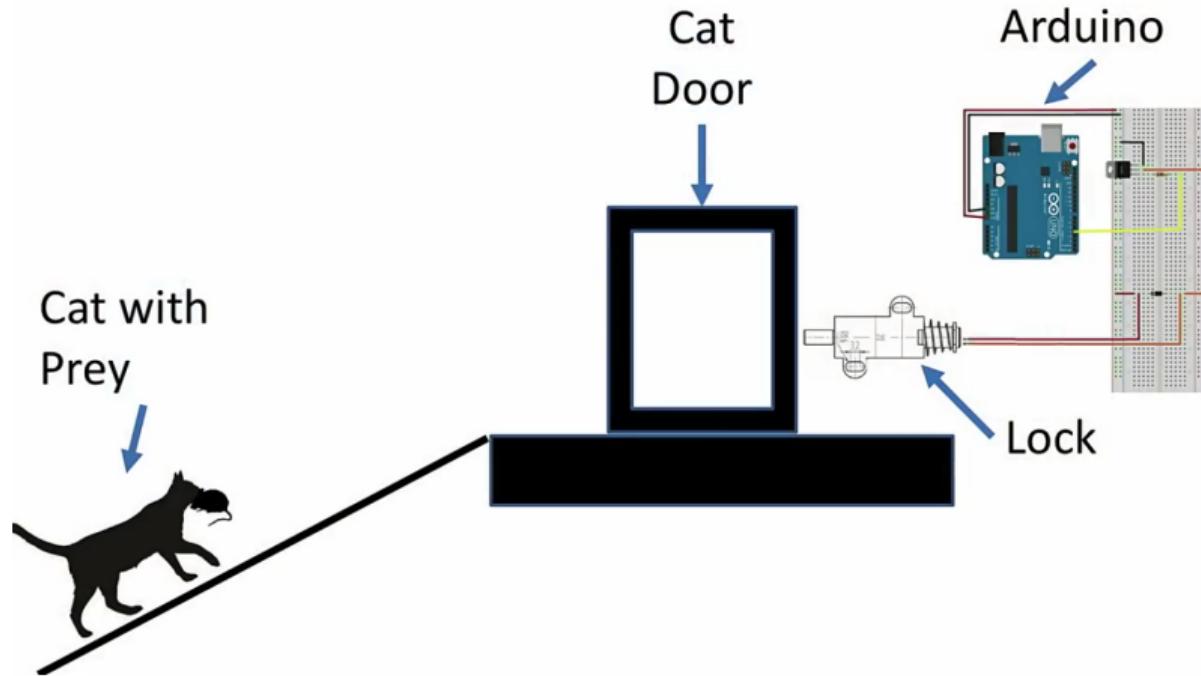


# Метрика это кот



<https://www.youtube.com/watch?v=1A-Nf3QIJjM>





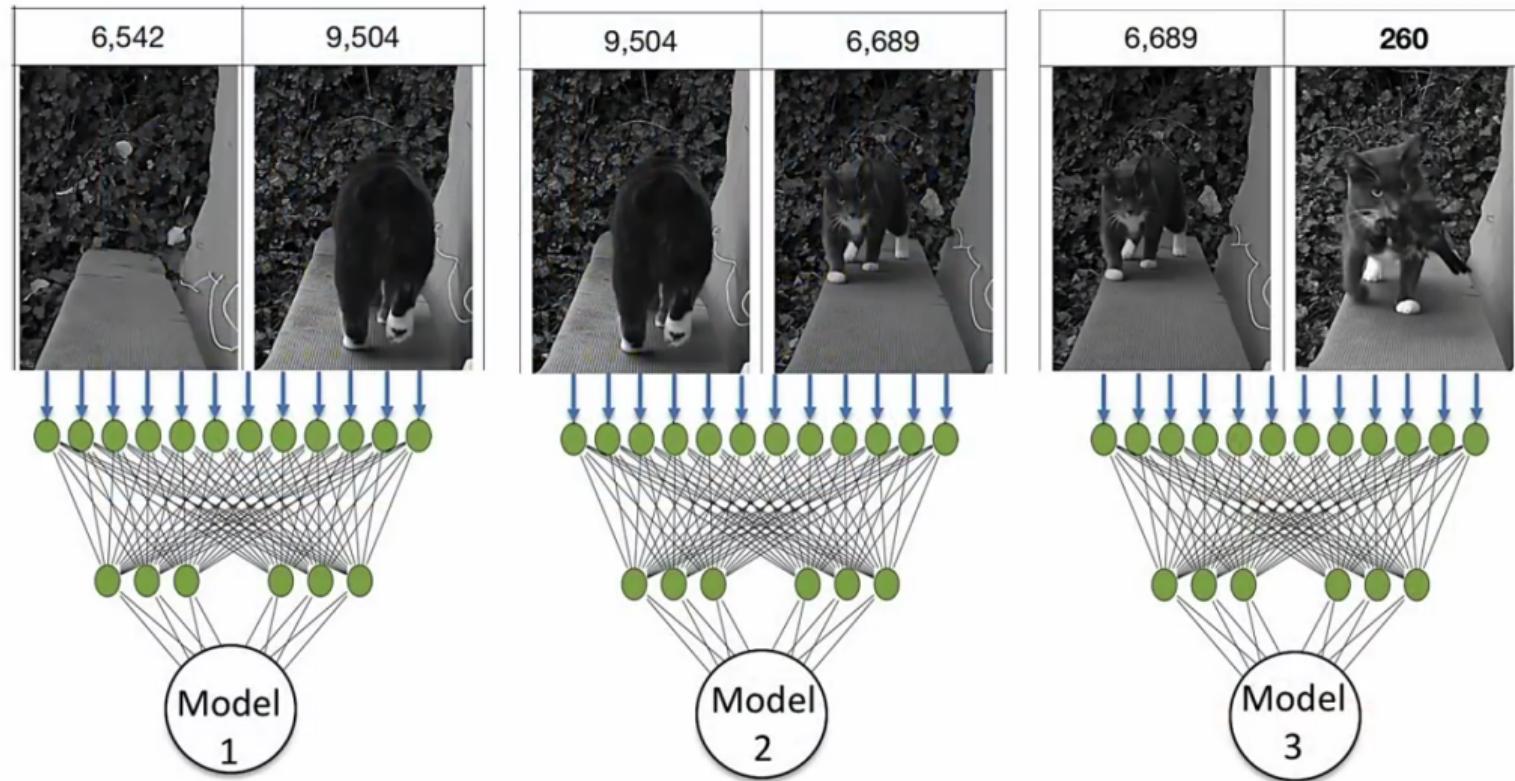




VS.



Image Type	No Cat	Cat not on approach	Cat on approach	Cat with prey
Count of Images	6,542	9,504	6,689	260
Example				



# CRITTER DETECTED!

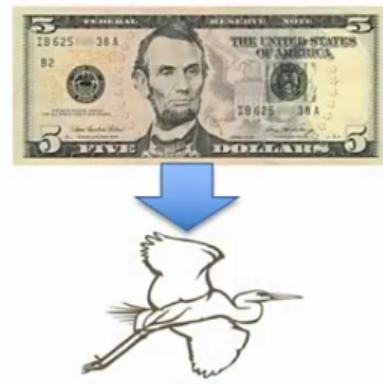
Step 1:  
Lock the door!



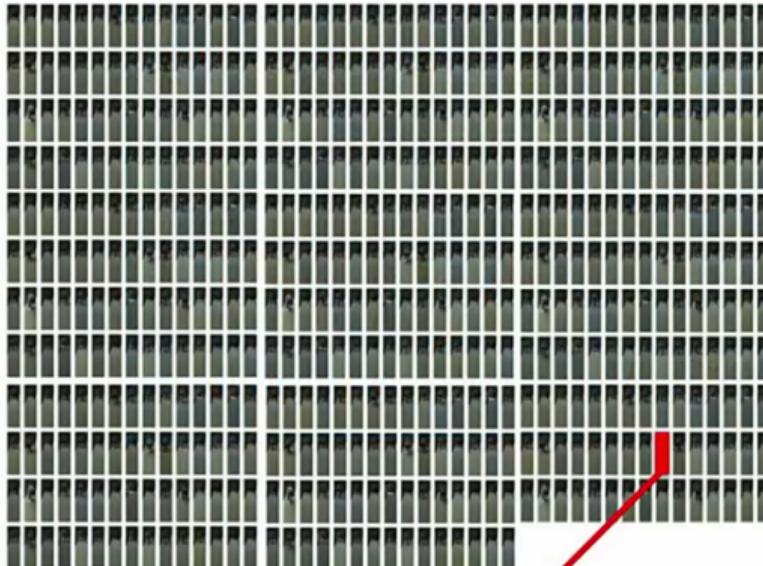
Step 2:  
Text me pics



Step 3:  
Donate blood  
money

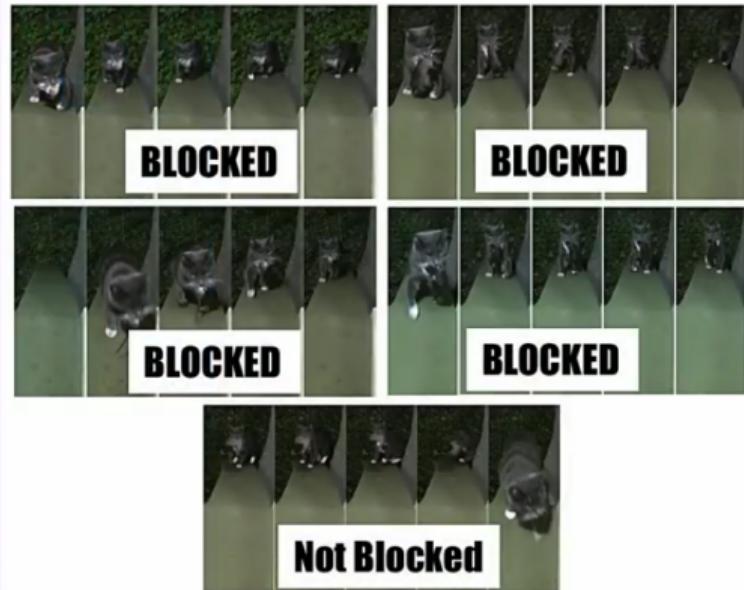


## Innocent Entries



1 Unfair Lockout

## Entries with Prey



4/5 Critters Blocked

# Как он это сделал?

- В выборке было всего лишь 260 фотографий кота с добычей, неужели этого хватило для обучения сетки?
- На самом деле сетку с нуля никто не учил, делался **transfer learning** (перенос знаний)

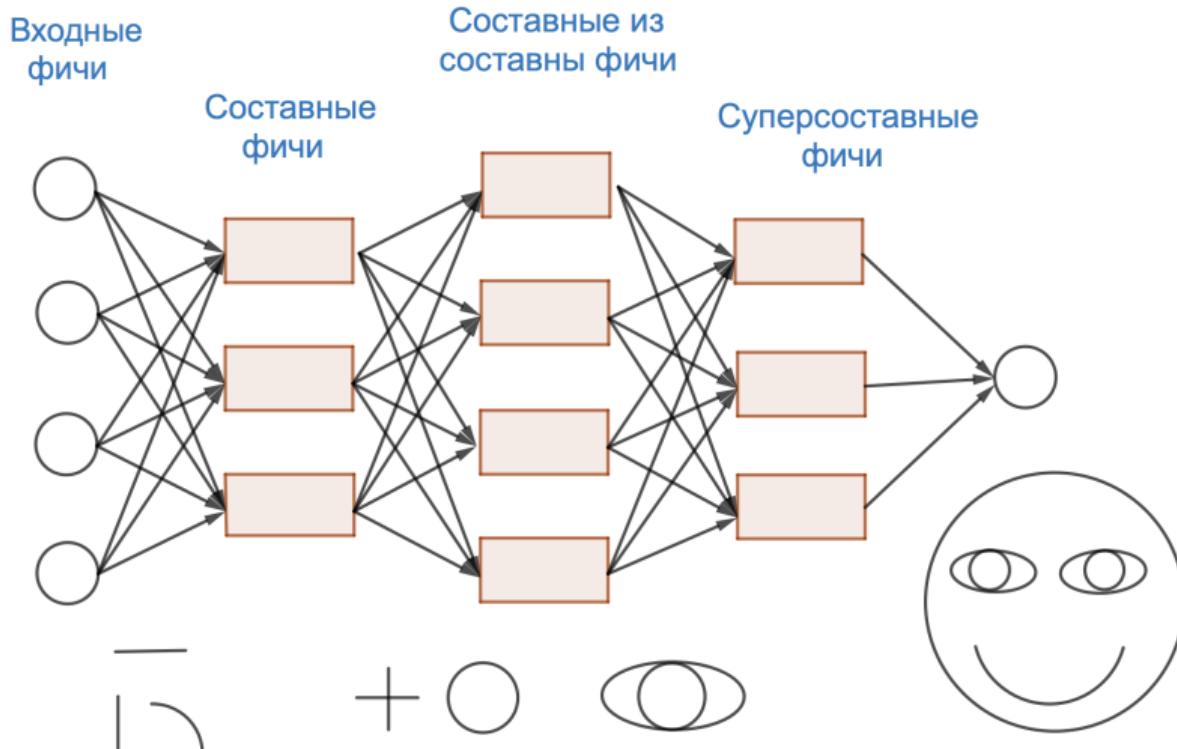
# Transfer learning (перенос знаний)



# Transfer learning

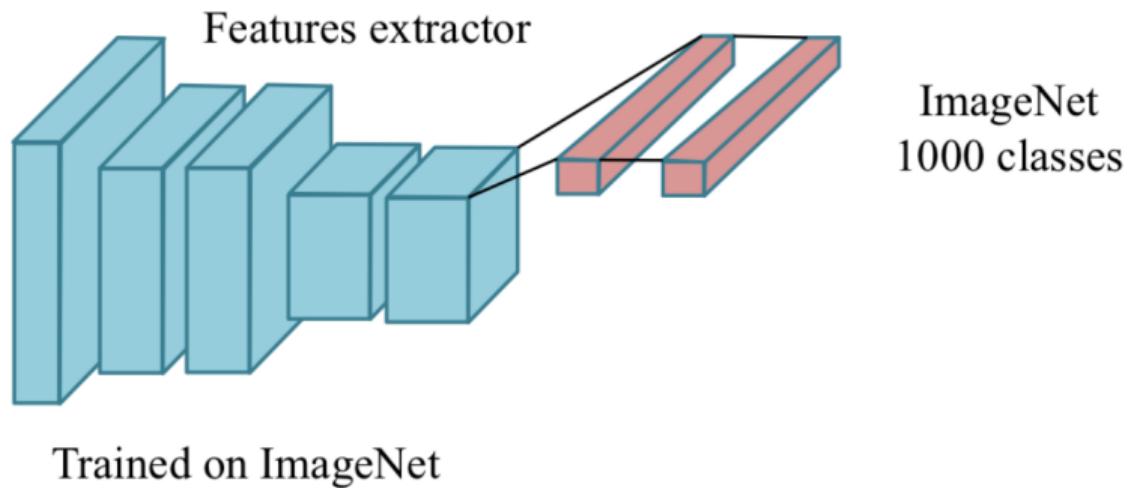
- На практике свёрточные сети с нуля обучаются только большие технологические компании
- Это происходит из-за ограниченности ресурсов
- Уже обученные архитектуры пытаются адаптироваться под новые задачи, это называется **transfer learning** (перенос знаний)

# Что выучивают нейросети



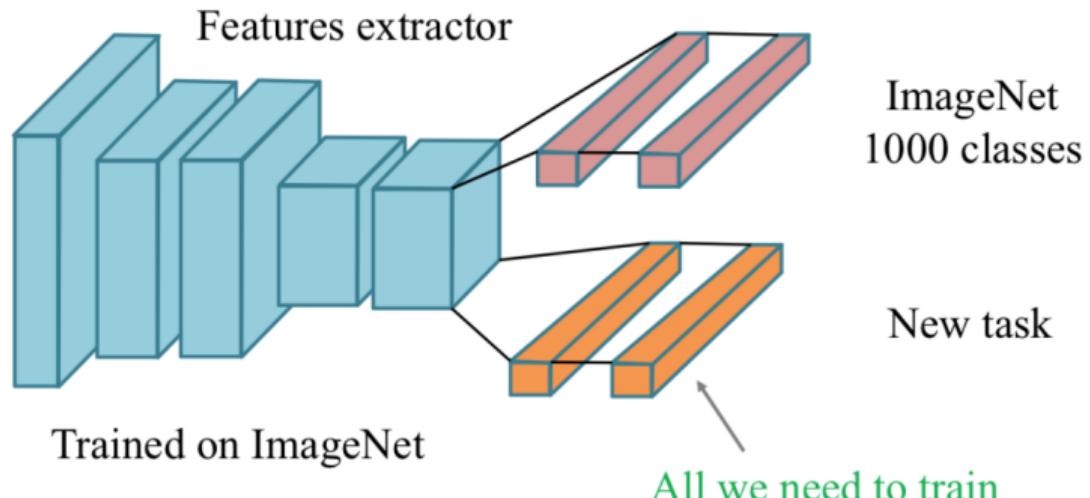
# Transfer learning

- Глубокие сети извлекают из изображений сложные фичи, но для их обучения нужно много данных...



# Transfer learning

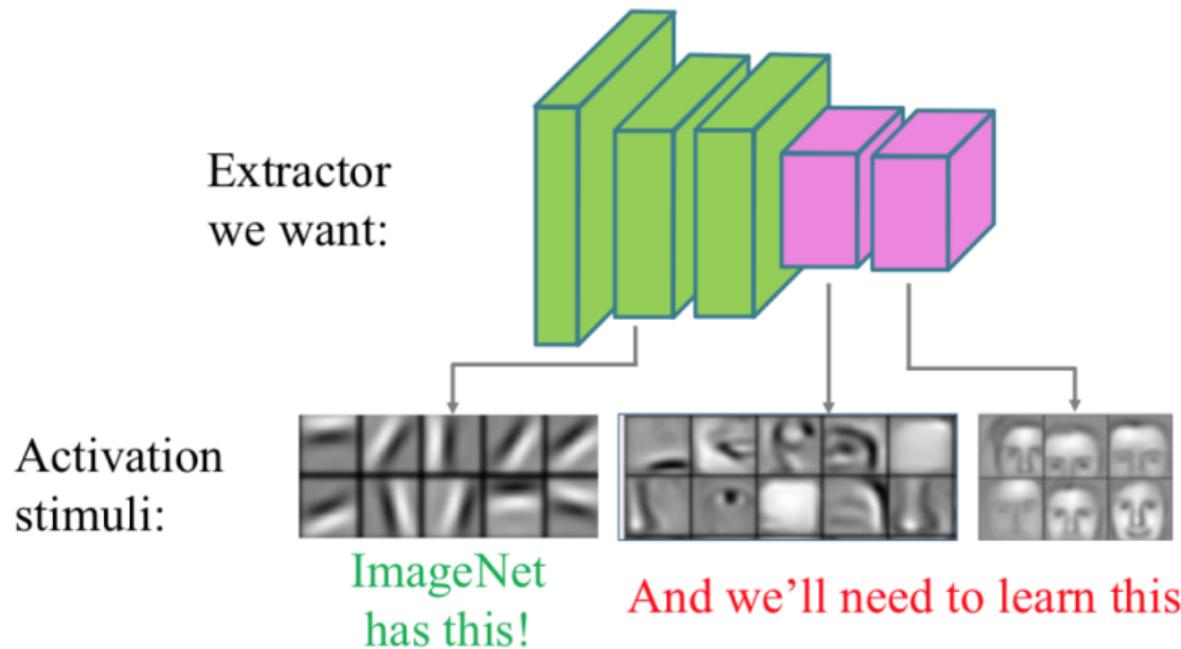
- Глубокие сети извлекают из изображений сложные фичи, но для их обучения нужно много данных...
- Давайте повторно использовать уже предобученную сеть!



# Transfer learning

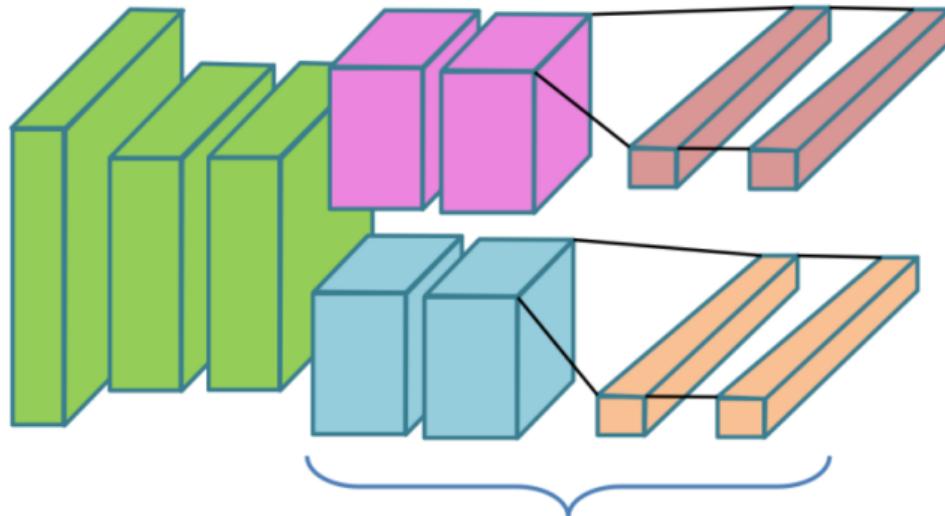
- Нужно меньше данных для обучения, так как нас интересуют лишь последние слои
- Как правило, на первых слоях фильтры похожие для всех задач
- Чем сильнее новая задача отличается от исходной, тем больше слоёв нужно переучивать
- Например, если мы хотим распознавать эмоции, в датасете для нашей сетки должны были быть человеческие лица

# Transfer learning



# Transfer learning

ImageNet features extractor



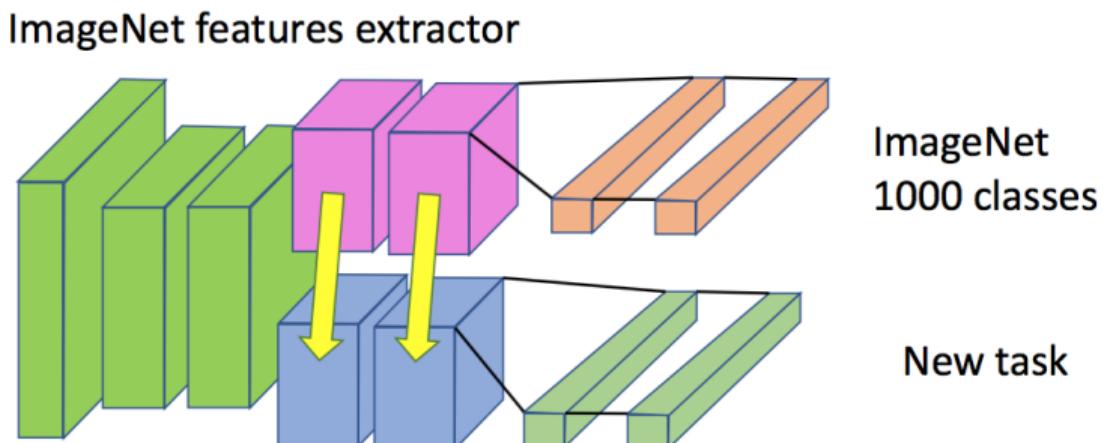
ImageNet  
1000 classes

New task

All we need to train

# Finetuning

- Можно инициализировать веса переучиваемых слоёв весами с ImageNet
- Это называется **finetuning**, так как инициализация неслучайная

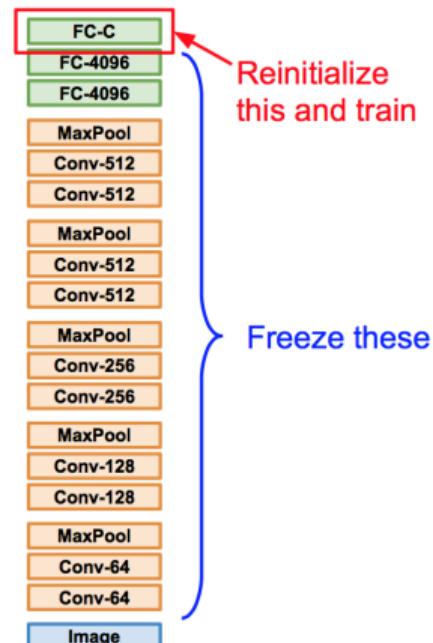


# Ещё раз, ещё раз

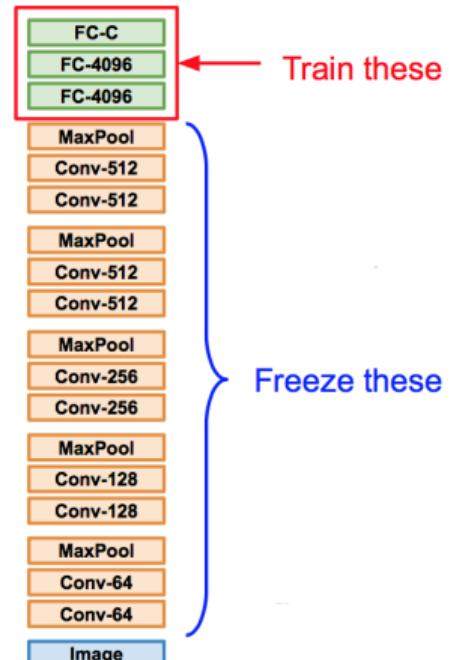
## 1. Обучаем на ImageNet



## 2. Если задача похожа: transfer learning



## 3. Задача немного отличается: finetuning

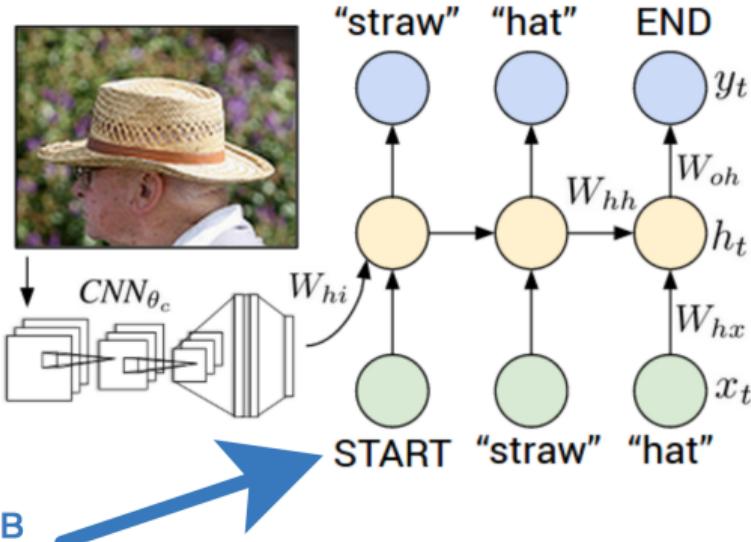


# Transfer learning

- Чем больше датасет, тем больше слоёв можно доучивать
- При finetuning выставляйте более низкую скорость обучения
- Для начала попробуйте 0.1 от оригинальной

# Transfer learning

Предобучена  
на ImageNet



Вектора для слов  
предобучены как w2v



Брать предобученные сети — нормальная, распространённая практика, а  
не маргинальные кейсы

<https://arxiv.org/pdf/1412.2306.pdf>

# Transfer learning

