# SqueezeNet

## Мотивация

1. Сети стали слишком большими, distributed learning зачастую упирается в пересылку градиентов, так как весов стало сотни миллионов.

2. Меньший вес модели нужен для более быстрого доставления модели к клиенту (например, пересылать 4 GB сеть на клиент для self driving car это очень неприятно)

3. Большие сети нельзя запустить на слабых процессорах типа мобильных, или FPGA.

## Идеи SqueezeNet

1. Заменить 3x3 фильтры на 1x1, так как 1x1 фильтр содержит в 9 раз меньше параметров.

2. Уменьшить количество входных каналов для сверток 3x3, так как количество параметров (без bias) для свертки 3x3 это $(number\ of\ input\ channels) \cdot (number\ of\ filters) \cdot (3 \cdot 3)$.

3. Делать downsampling ближе к "концу"сети, но более агрессивный. Действительно, max pooling или свертки со $stride > 1$ снижают размер активационных карт. Но так как в статье цель снизить количество параметров, то, возможно, перспективно делать такой downsample начиная с какого-то момента, а первичные слои оставить полными.

## Основной модуль статьи

Основной модуль статьи - модуль Fire, он воплощает в себе все идеи выше.

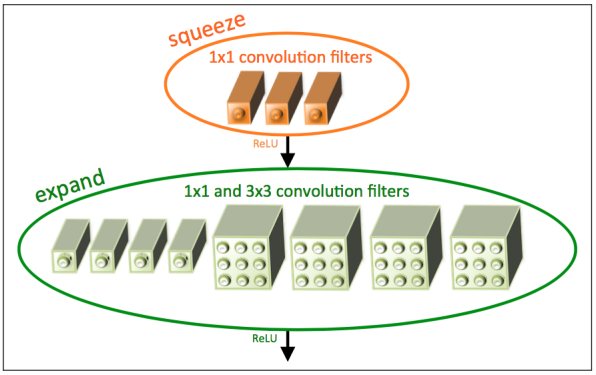



Figure 1: Microarchitectural view: Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

У него 3 основных параметра:

$$s_{1x1} - \textit{количество сверток 1x1 в squeeze модуле},$$

$$e_{1x1} - \textit{количество сверток 1x1 в expand модуле},$$

$$e_{3x3} - \textit{количество сверток 3x3 в expand модуле}$$

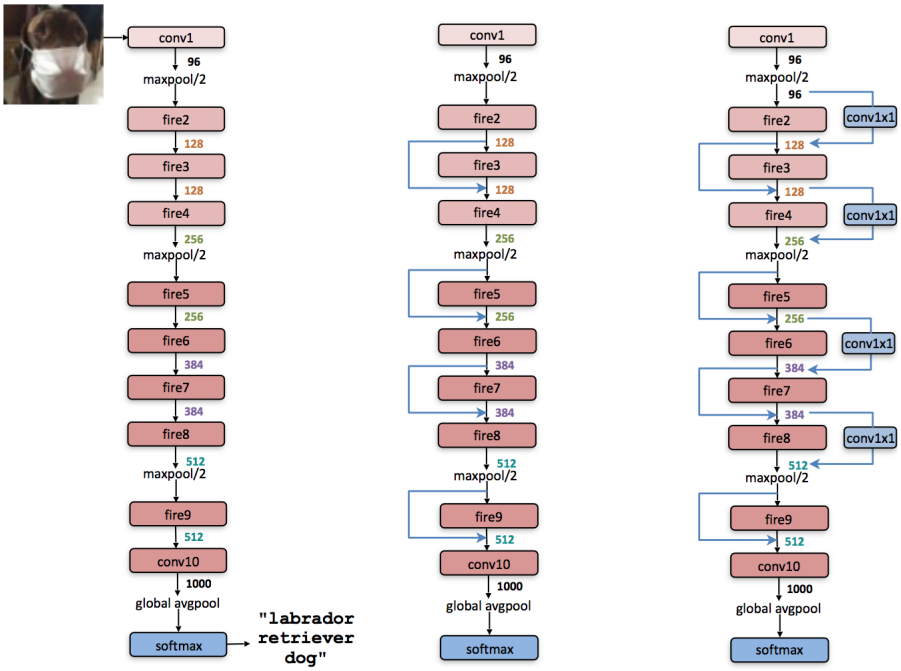



Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

## Реализация

Авторы реализовали три подхода:

1. SqueezeNet. Сеть, состоящая из модулей Fire и max pooling'ов.

2. SqueezeNet + Skip Connections. Resnet like skip connections там, где размерность выхода совпадает с размерностью входа.

3. SqueezeNet + complex Skip Connections. Тут они взяли подход №2 и добавили skip connections с свертками 1x1, чтобы выровнять входную и выходную размерности.

Поговорим пока про первый подход. Таблица размерностей SqueezeNet:

Table 1: SqueezeNet architectural dimensions. (The formatting of this table was inspired by the Inception2 paper (Ioffe & Szegedy, 2015).)

| layer name/type | output size | filter size / stride (if not a fire layer) | depth | $s_{1x1}$ (#1x1 squeeze) | $e_{1x1}$ (#1x1 expand) | $e_{3x3}$ (#3x3 expand) | $s_{1x1}$ sparsity | $e_{1x1}$ sparsity | $e_{3x3}$ sparsity | # bits | #parameter before pruning | #parameter after pruning |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| input image | 224x224x3 | | | | | | | | | | - | - |
| conv1 | 111x111x96 | 7x7/2 (x96) | 1 | | | | 100% (7x7) | | | 6bit | 14,208 | 14,208 |
| maxpool1 | 55x55x96 | 3x3/2 | 0 | | | | | | | | | |
| fire2 | 55x55x128 | | 2 | 16 | 64 | 64 | 100% | 100% | **33%** | 6bit | 11,920 | 5,746 |
| fire3 | 55x55x128 | | 2 | 16 | 64 | 64 | 100% | 100% | **33%** | 6bit | 12,432 | 6,258 |
| fire4 | 55x55x256 | | 2 | 32 | 128 | 128 | 100% | 100% | **33%** | 6bit | 45,344 | 20,646 |
| maxpool4 | 27x27x256 | 3x3/2 | 0 | | | | | | | | | |
| fire5 | 27x27x256 | | 2 | 32 | 128 | 128 | 100% | 100% | **33%** | 6bit | 49,440 | 24,742 |
| fire6 | 27x27x384 | | 2 | 48 | 192 | 192 | 100% | **50%** | **33%** | 6bit | 104,880 | 44,700 |
| fire7 | 27x27x384 | | 2 | 48 | 192 | 192 | **50%** | 100% | **33%** | 6bit | 111,024 | 46,236 |
| fire8 | 27x27x512 | | 2 | 64 | 256 | 256 | 100% | **50%** | **33%** | 6bit | 188,992 | 77,581 |
| maxpool8 | 13x12x512 | 3x3/2 | 0 | | | | | | | | | |
| fire9 | 13x13x512 | | 2 | 64 | 256 | 256 | **50%** | 100% | **30%** | 6bit | 197,184 | 77,581 |
| conv10 | 13x13x1000 | 1x1/1 (x1000) | 1 | | | | **20%** (3x3) | | | 6bit | 513,000 | 103,400 |
| avgpool10 | 1x1x1000 | 13x13/1 | 0 | | | | | | | | | |
| | activations | | | parameters | | | compression info | | | | 1,248,424 (total) | **421,098** (total) |

Здесь pruning - это процесс, придуманный Han et al. в 2015, который зануляет некоторые веса в модели, которые ниже определенного порога, таким образом превращая матрицы в разреженные.

Также стоит оговорится о технике квантизации под названием Deep Compression, который к момент выхода этой статьи был главным методом "ужимания"сетей.

Результаты SqueezeNet:

Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

| CNN architecture | Compression Approach | Data Type | Original → Compressed Model Size | Reduction in Model Size vs. AlexNet | Top-1 ImageNet Accuracy | Top-5 ImageNet Accuracy |
|---|---|---|---|---|---|---|
| AlexNet | None (baseline) | 32 bit | 240MB | 1x | 57.2% | 80.3% |
| AlexNet | SVD (Denton et al., 2014) | 32 bit | 240MB → 48MB | 5x | 56.0% | 79.4% |
| AlexNet | Network Pruning (Han et al., 2015b) | 32 bit | 240MB → 27MB | 9x | 57.2% | 80.3% |
| AlexNet | Deep Compression (Han et al., 2015a) | 5-8 bit | 240MB → 6.9MB | 35x | 57.2% | 80.3% |
| SqueezeNet (ours) | None | 32 bit | 4.8MB | **50x** | 57.5% | 80.3% |
| SqueezeNet (ours) | Deep Compression | 8 bit | 4.8MB → 0.66MB | **363x** | 57.5% | 80.3% |
| SqueezeNet (ours) | Deep Compression | 6 bit | 4.8MB → 0.47MB | **510x** | 57.5% | 80.3% |

SqueezeNet даже без техники Deep Compression имеет размер в 50 раз меньше, чем у AlexNet, прародителя современных сверточных сетей, при этом качество такое же. Применяя Deep Compression авторы получили уменьшение в 510 раз. Модель весит всего 0.47 MB.

## Skip Connections?

А как же Skip Connections? Помогают ли они? Вот результаты:

Table 3: SqueezeNet accuracy and model size using different macroarchitecture configurations

| Architecture | Top-1 Accuracy | Top-5 Accuracy | Model Size |
|---|---|---|---|
| Vanilla SqueezeNet | 57.5% | 80.3% | 4.8MB |
| SqueezeNet + Simple Bypass | **60.4%** | **82.5%** | 4.8MB |
| SqueezeNet + Complex Bypass | 58.8% | 82.0% | 7.7MB |

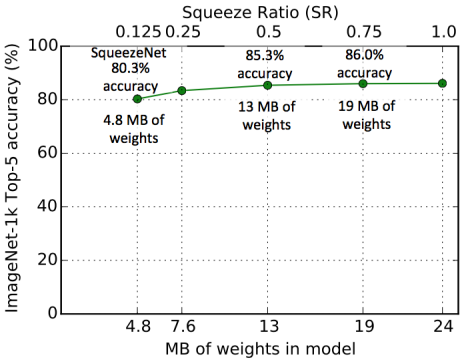
Resnet like Skip Connections действительно помогли, увеличивая качество на 2.9 абсолютных процента, не увеличивая количество параметров. Что интересно, обобщенные Skip Connections, которые, кажется, не должны были работать хуже, чем обычные, работаю хуже, при этом увеличивается количество параметров.

## Изучение гиперпараметров

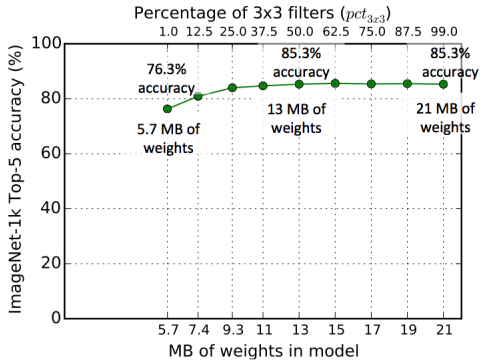
Немаловажно, что авторы изучают влияние некоторых параметров сети на размер и качество.

**Определение.** *Squeeze Ratio (SR)* это отношение количества фильтров в squeeze блоке модуля Fire к количеству фильтров в expand блоке.
Изучается влияние вышеопределенного SR и общего отношения количества сверток 3x3 к 1x1.

Интересно, что на втором графике отношения количества сверток к качеству мы видим, что примерно на 50% качество выходит на плато.




(a) Exploring the impact of the squeeze ratio ($SR$) on model size and accuracy.




(b) Exploring the impact of the ratio of 3x3 filters in expand layers ($pct_{3x3}$) on model size and accuracy.

### Итоги

Авторы начали большую работу над эффективным использованием параметров в сверточных сетях, и эту работу дальше продолжат очень много исследователей и добьются отличных результатов.

## MobileNet

### Мотивация

1. Сосредоточиться не только на маленьком размере, но и низком энергопотреблении и маленьком lattency нейросетей.
2. Улучшить результаты SqueezeNet, не увеличивая сеть.

### Идея

Главная идея статьи - Depthwise Separable Convolution.

1. Стандартая свертка.

   *Свертка* $\mathbb{K}$*, stride=1, padding=1 математически выражается так:*

   $$G_{k,l,n} = \sum_{i,j,m} \mathbb{K}_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m},$$

   где F - карта активаций.

   Стандартная свертка $C \cdot C$ имеет $(number\ of\ input\ channels) \cdot (number\ of\ filters) \cdot (C \cdot C)$ параметров.

   А количество операций будет:

   $$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F,$$

   где $D_F$ - размерность карты активации, M - ширина, N - высота, $D_K$ - размерность свертки.

2. Depthwise separable convolution Depthwise separable convolution состоит из двух слоев: depthwise convolution и pointwise convolution (1x1 свертка).

   *Depthwise separable convolution с 1 фильтром на канал математически выражается так:*

$$G_{k,l,m} = \sum_{i,j} \mathbb{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m}$$

То есть она лишается одной пространственной размерности. Чтобы саггрегировать по этой пространственной размерности используется свертка 1x1.

Количество операций будет:

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F,$$

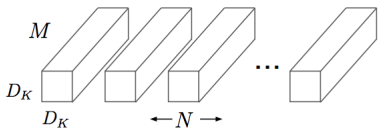
где $D_F$ - размерность карты активации, M - ширина, N - высота, $D_K$ - размерность свертки.

В итоге получается выигрыш в количестве операций:

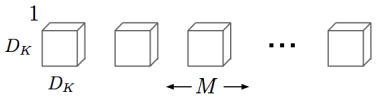
$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} =$$

$$= \frac{1}{N} + \frac{1}{D_K^2}$$

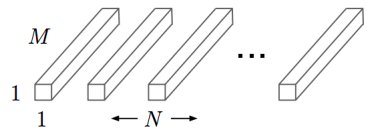
Иллюстрация:




(a) Standard Convolution Filters




(b) Depthwise Convolutional Filters




(c) $1 \times 1$ Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Также авторы вносят два параметра $\alpha \in (0, 1]$ - Width Multiplier, который отвечает за размер M и N на каждом слое, равномерно уменьшая их, и $\rho \in (0, 1]$ - Resolution Multiplier, который отвечает за размер активационных карт на каждом слое, тоже равномерно уменьшая их. Регулирование этих двух параметров позволяет достичь оптимального баланса в отношении скорость / качество для определенной задачи.

## Результаты

Table 8. MobileNet Comparison to Popular Models

| Model | ImageNet Accuracy | Million Mult-Adds | Million Parameters |
|---|---|---|---|
| 1.0 MobileNet-224 | 70.6% | 569 | 4.2 |
| GoogleNet | 69.8% | 1550 | 6.8 |
| VGG 16 | 71.5% | 15300 | 138 |

Table 9. Smaller MobileNet Comparison to Popular Models

| Model | ImageNet Accuracy | Million Mult-Adds | Million Parameters |
|---|---|---|---|
| 0.50 MobileNet-160 | 60.2% | 76 | 1.32 |
| Squeezenet | 57.5% | 1700 | 1.25 |
| AlexNet | 57.2% | 720 | 60 |

По первому графику видим, что они побеждают GoogleNet, в своё время очень мощную сеть, уменьшая одновременно и количество параметров и количество вычислений. По второму видим, что они побеждают SqueezeNet, при этом делая на несколько порядков меньше вычислений.

## Итоги

Такие факторизованные свертки очень хорошо заработали, при чем авторы перенесли результаты на другие задачи CV. Теперь эта модель занимает почетное место бейзлайн модели в почти любой задаче CV.

# Список источников


- SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE
- MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications