

Xception: Deep Learning with Depthwise Separable Convolutions

François Chollet

Google, Inc.

fchollet@google.com

Abstract

我々はInceptionモジュールの解釈として、Regular convolutionとDepthwise separable convolutionの中間に存在する手法であるという解釈を提唱する

この解釈の観点から、Depthwise separable convolutionはInceptionモジュール内で最大数のタワーを設計した形と理解できる

この考え方に基づき、我々はInceptionネットワークにおいてInceptionモジュールをDepthwise separable convolutionに置き換えた新規のディープニューラルネットワークを開発した

Xceptionと名付けられた我々の提案するネットワークはInception V3と比較し、ImageNetデータセットを用いた画像分類性能の評価で僅かな改善を、より大規模なデータセット(3億5000万画像、17000クラス)を用いた画像分類性能の評価で大きな改善を達成した

XceptionはInception V3とほぼ同数のパラメータを有するため、性能の改善は計算能力の増加によるものではなく、モデルパラメータのより効率的な使用に起因する

論文の構成

- Abstract
- 背景と目的** {
 - 1. Introduction
 - 1.1. The Inception hypothesis
 - 1.2. The continuum between convolutions and separable convolution
- 関連研究** {
 - 2. Prior work
- 提案手法** {
 - 3. The Xception architecture
- データと解析手法** {
 - 4. Experimental evaluation
 - 4.1. The JFT dataset
 - 4.2. Optimization configuration
 - 4.3. Regularization configuration
 - 4.4. Training infrastructure
 - 4.5. Comparison with Inception V3
 - 4.5.1 Classification performance
 - 4.5.2 Size and speed
 - 4.6. Effect of the residual connections
 - 4.7. Effect of an intermediate activation after pointwise convolution
- 結果** {
 - 5. Future directions
- 結論** {
 - 6. Conclusions

論文の目的と結論の整合性の確認

目的 (Introductionの最後の段落)

While Inception modules are conceptually similar to convolutions (they are convolutional feature extractors), they empirically appear to be capable of learning richer representations with less parameters. **How do they work, and how do they differ from regular convolutions? What design strategies come after Inception?**

Inceptionモジュールは概念的には畳み込みに似ているが、経験的にはInceptionモジュールは畳み込みに比べてより複雑な特徴をより少ないパラメータで学習できる。では**Inceptionモジュールはどのように機能し、どういった点で通常の畳み込みと異なるのか。そしてInceptionをさらに発展させる手法はどのようなものか。**

結論(6. Conclusionsの部分)

We showed how convolutions and depthwise separable convolutions lie at both extremes of a discrete spectrum, with Inception modules being an intermediate point in between. This observation has led to us to propose replacing Inception modules with depthwise separable convolutions in neural computer vision architectures. We presented a novel architecture based on this idea, named Xception, which has a similar parameter count as Inception V3. Compared to Inception V3, Xception shows small gains in classification performance on the ImageNet dataset and large gains on the JFT dataset. We expect depthwise separable convolutions to become a cornerstone of convolutional neural network architecture design in the future, since they offer similar properties as Inception modules, yet are as easy to use as regular convolution layers.

我々は、通常の畳み込みとDepthwise separable convolutionがどのように概念上の対極に存在し、Inceptionモジュールがどのようにその対極の間に存在するかを示した。この結果は画像処理のニューラルネットワークの構造を従来のInceptionモジュールからdepthwise separable convolutionへ置き換えることを勧めるものである。我々はこの見解に基づき、Inception V3と同程度のパラメータ数を有する、Xceptionと名付けた新たなネットワーク構造を提案した。Inception V3との比較において、XceptionはImageNetデータセットに対する画像分類性能で僅かな改善を、JFTデータセットに対する画像分類性能で大きな改善を達成した。Depthwise separable convolutionは通常の畳み込みのように利用できるにも関わらずInceptionモジュールと似た特性を有するため、我々はこの手法が将来的にニューラルネットワークの構造設計の礎となると考える。

論文の目的と結論の整合性の確認(整理)

目的

Inceptionモジュールはどのように機能し、どういった点で通常の畳み込みと異なるのかを明らかにし、Inceptionをさらに発展させる手法を提案する

結論

Regular convolutionとDepthwise separable convolutionが対極に存在し、Inceptionモジュールがどのようにその中間に位置するかを説明する概念を提示した

提案した観点に基づいて、Inception V3と同程度のパラメータ数を持つXceptionと名付けた新たなネットワーク構造を提案した

Inception V3との比較において、提案手法であるXceptionがImageNetデータセットを用いた画像分類性能評価で僅かな改善を、IJFデータセットを用いた画像分類性能評価で大きな改善を達成することを示した

1. Introduction

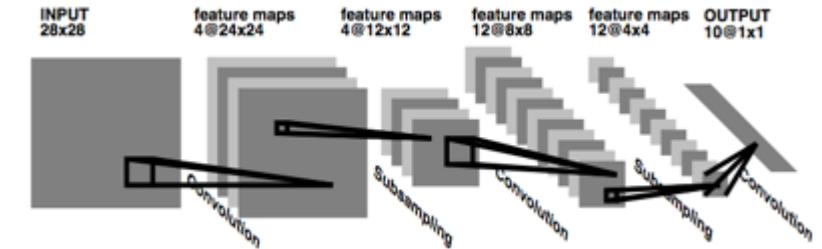
近年、畳み込みニューラルネットワークは画像処理アルゴリズムの主軸となった

畳み込みニューラルネットワークの始まりであるLeNetでは、ネットワーク構造は畳み込みで特徴抽出後にmax-poolingでサブサンプリングする構造の積み重ねというシンプルなものだった

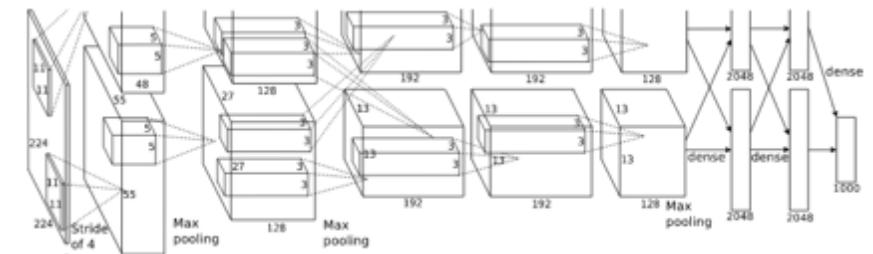
2012年のAlexnetではこの構造がさらに複雑化され、max-pooling間の畳み込みは複数となり、各空間規模において豊富な特徴量を学習できるようになった

その後このスタイルのネットワークをさらに深くしていく
(Zeiler and Fergus in 2013、VGG architecture in 2014)
方向へネットワーク設計戦略が進んだ

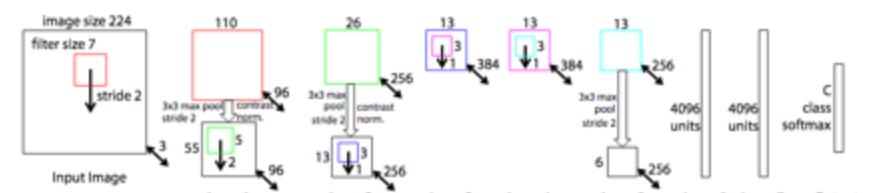
そのようなネットワーク設計の発展過程で、Inception構造が登場した



Y. LeCun, et al. Neural networks: the statistical mechanics perspective, 261:276, 1995.



A. Krizhevsky, et al. In Advances in neural information processing systems, pages 1097–1105, 2012



M. D. Zeiler and R. Fergus. In Computer Vision–ECCV 2014, pages 818–833. Springer, 2014

1. Introduction

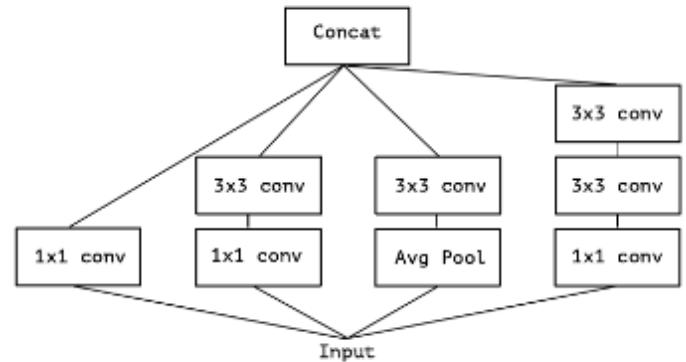
Inceptionネットワークモデルは右図のようなInceptionモジュールの積み重ねから成る

モジュールを多段に積み重ねるというアイデア自体は、単純な畳み込みを重ねるという初期のVGG-styleの設計戦略からきている

Inceptionモジュールは概念的には通常の畳み込みに似ているが、経験的にはInceptionモジュールは通常の畳み込みに比べてより複雑な特徴をより少ないパラメータで学習できる

ではそもそもInceptionモジュールはどのように機能し、どういった点で通常の畳み込みと異なるのか、そしてInceptionをさらに発展させる手法はどのようなものか

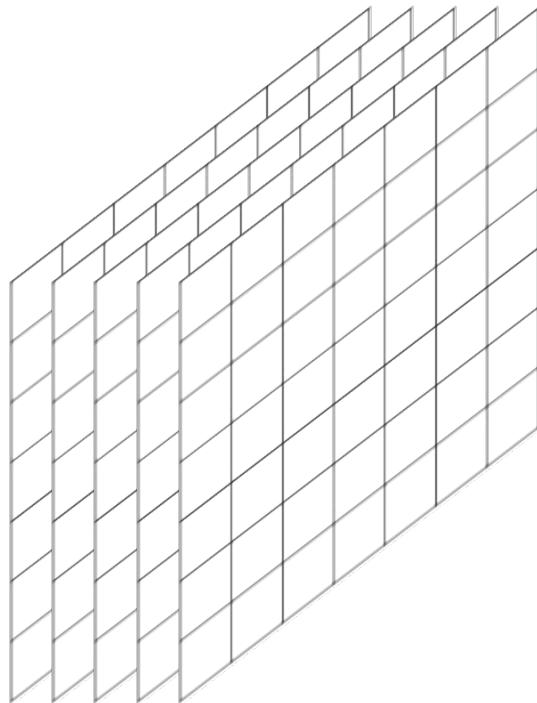
Figure 1. A canonical Inception module (Inception V3).



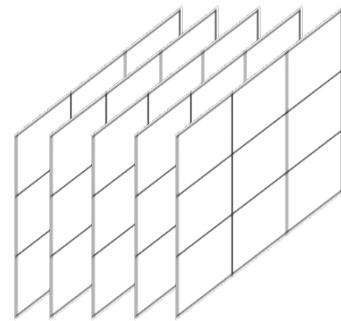
1.1. The Inception hypothesis

畳み込み層は3次元空間 (width, height, channel) に対して学習を行う
つまり、一つの畳み込みカーネルはチャネル方向と2次元画像方向の情報を同時にマッピングしていることになる

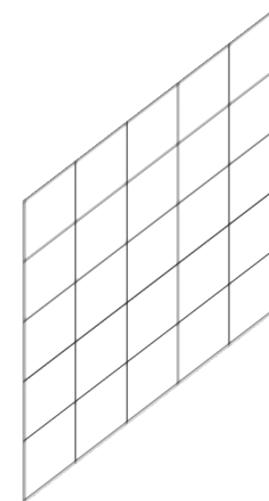
Regular convolution



$7 \times 7 \times 5$
入力



$3 \times 3 \times 5$
畳み込みフィルター
(畳み込みカーネル)

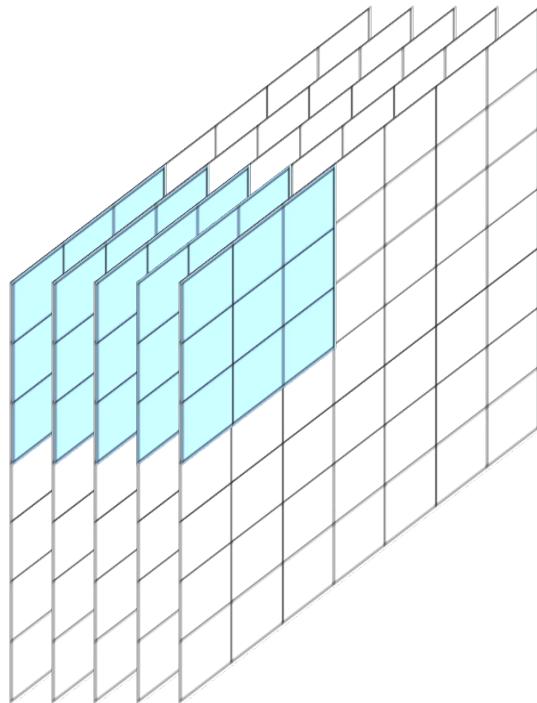


$5 \times 5 \times 1$
出力

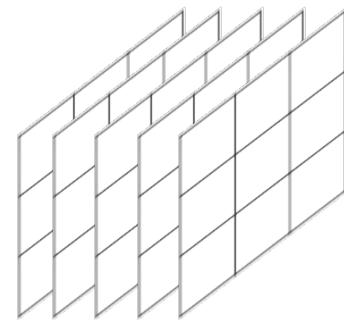
1.1. The Inception hypothesis

畳み込み層は3次元空間 (width, height, channel) に対して学習を行う
つまり、一つの畳み込みカーネルはチャネル方向と2次元画像方向の情報を同時にマッピングしていることになる

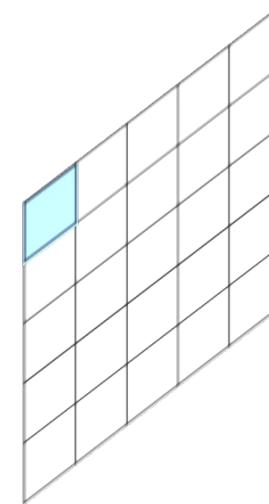
Regular convolution



$7 \times 7 \times 5$
入力



$3 \times 3 \times 5$
畳み込みフィルター
(畳み込みカーネル)

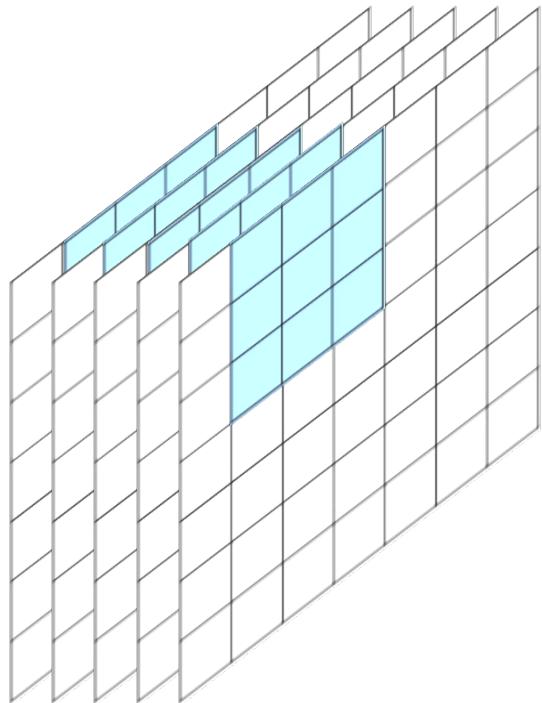


$5 \times 5 \times 1$
出力

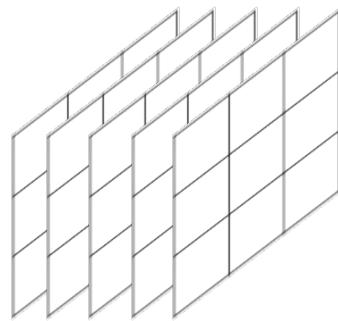
1.1. The Inception hypothesis

畳み込み層は3次元空間 (width, height, channel) に対して学習を行う
つまり、一つの畳み込みカーネルはチャネル方向と2次元画像方向の情報を同時にマッピングしていることになる

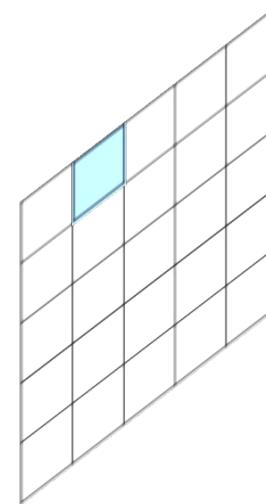
Regular convolution



$7 \times 7 \times 5$
入力



$3 \times 3 \times 5$
畳み込みフィルター
(畳み込みカーネル)

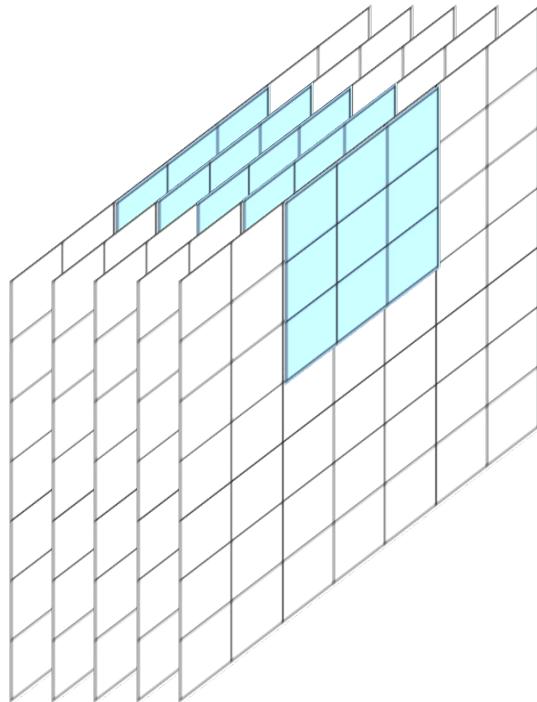


$5 \times 5 \times 1$
出力

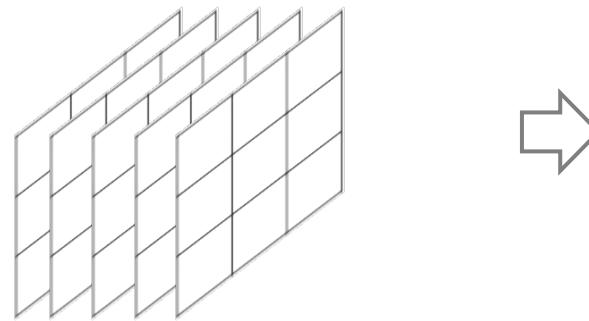
1.1. The Inception hypothesis

畳み込み層は3次元空間 (width, height, channel) に対して学習を行う
つまり、一つの畳み込みカーネルはチャネル方向と2次元画像方向の情報を同時にマッピングしていることになる

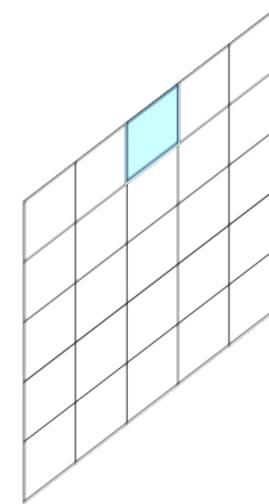
Regular convolution



$7 \times 7 \times 5$
入力

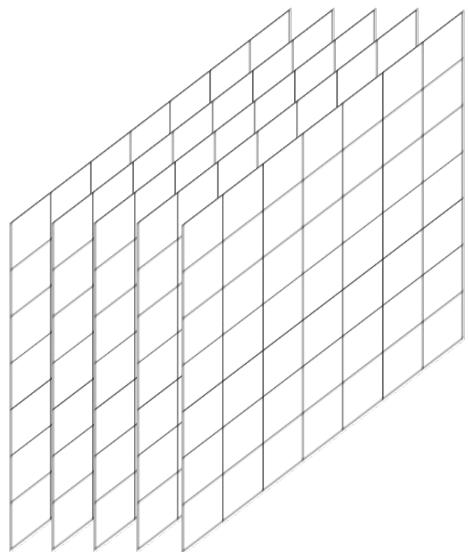


$3 \times 3 \times 5$
畳み込みフィルター
(畳み込みカーネル)

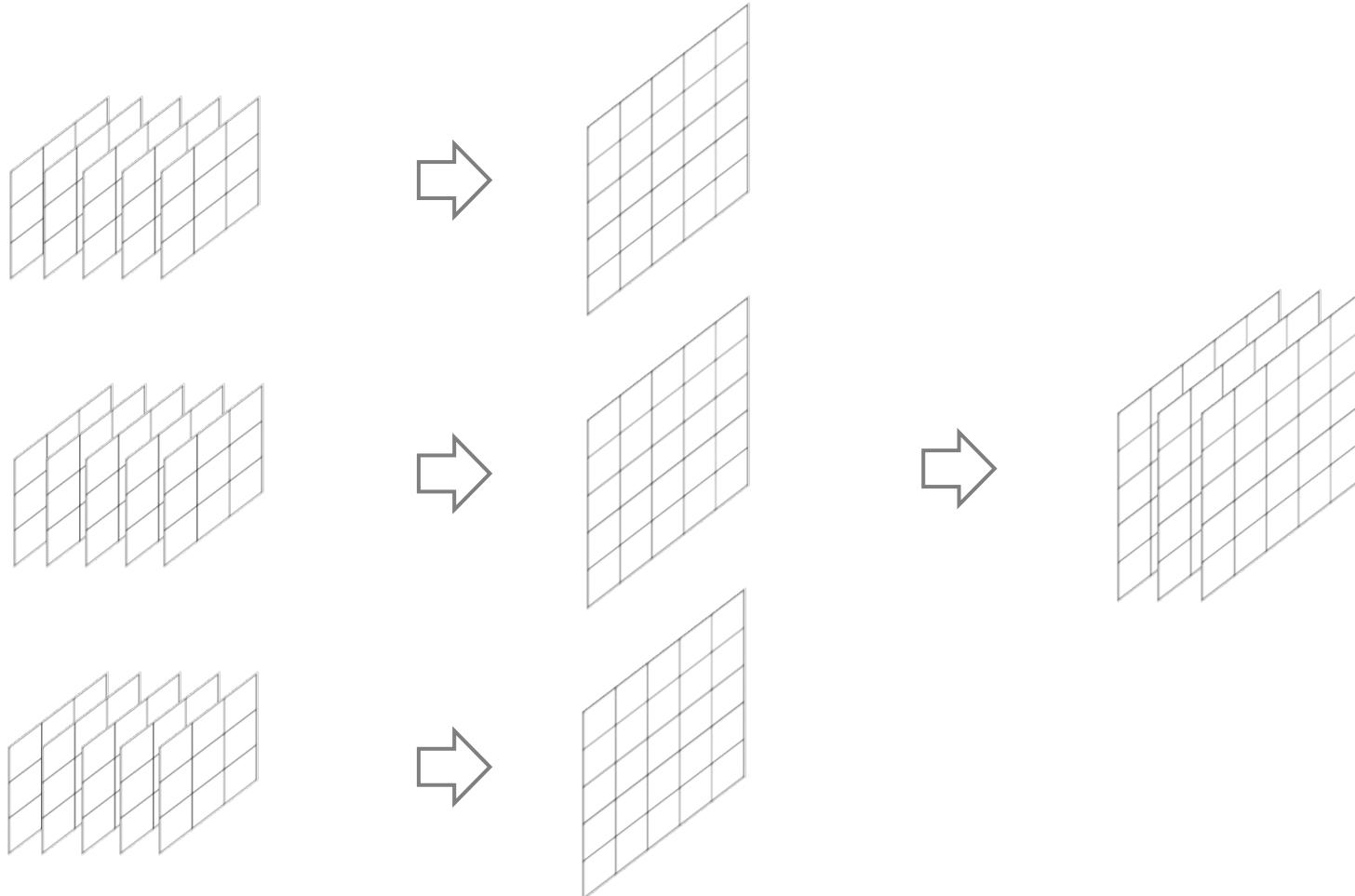


$5 \times 5 \times 1$
出力

Regular convolution



$7 \times 7 \times 5$
入力



$3 \times 3 \times 5$
畳み込みフィルター
(畳み込みカーネル)
3個

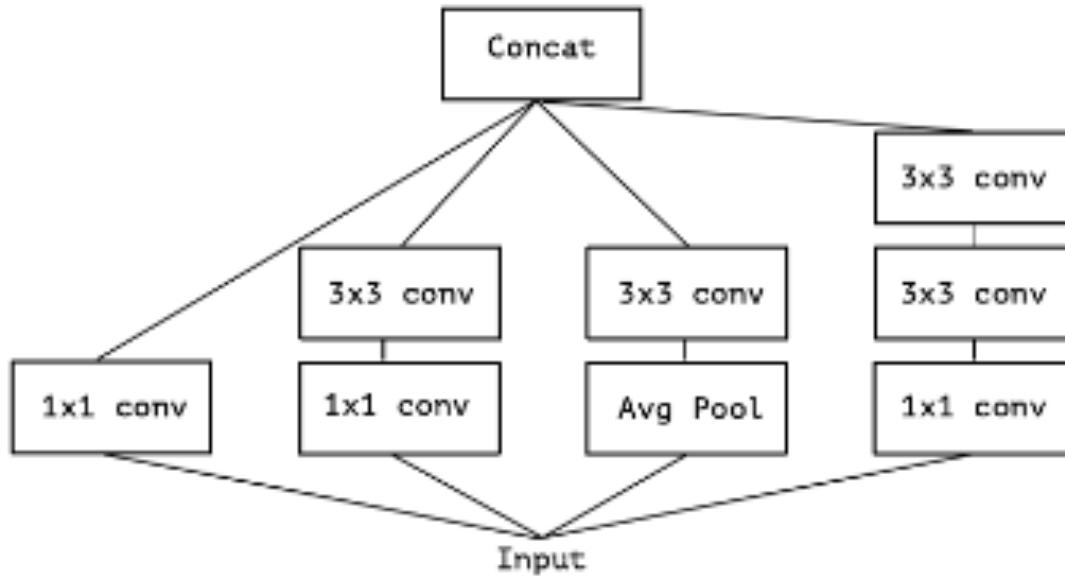
$5 \times 5 \times 1$
出力
3個

$5 \times 5 \times 3$
出力

1.1. The Inception hypothesis

典型的なInceptionモジュールを詳細に見ると、最初に 1×1 の畳み込みでチャネル方向の畳み込みを行い、その後に得られたチャネル方向の相互関係に対して2次元画像方向を含めた畳み込みを行なっている

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



ということは、「Inceptionモジュールはチャネル方向と2次元画像方向の相互情報がなるべく一緒にマッピングされないように分離している」と理解できるのではないか

1.1. The Inception hypothesis

より単純なInceptionモジュールを考える

Figure 2. A simplified Inception module.

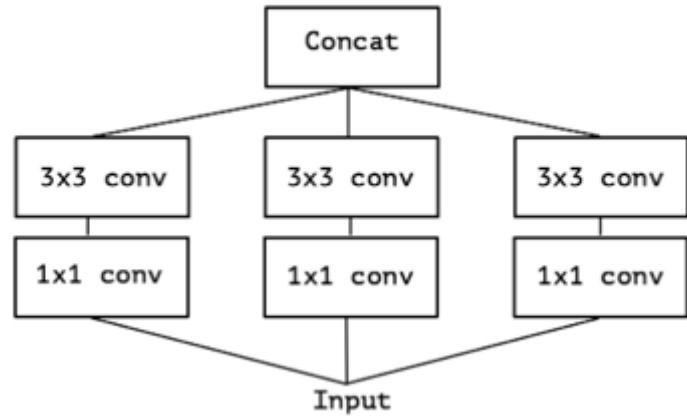
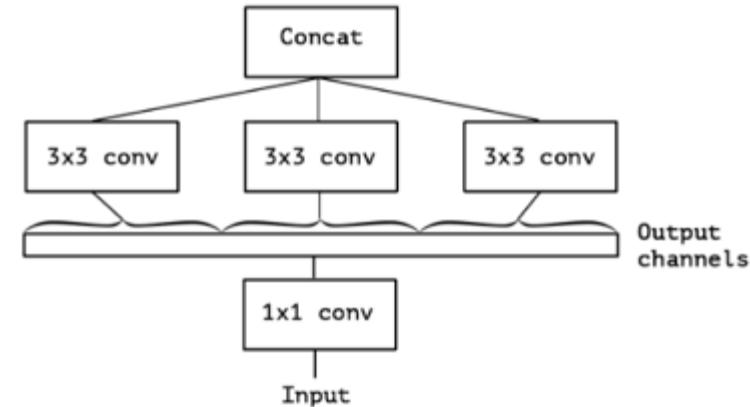


Figure 3. A strictly equivalent reformulation of the simplified Inception module.



1x1畳み込みの出力チャネルで3x3畳み込みに進む際にオーバーラップがないとすれば、Figure 2はFigure 3のように書き直せる

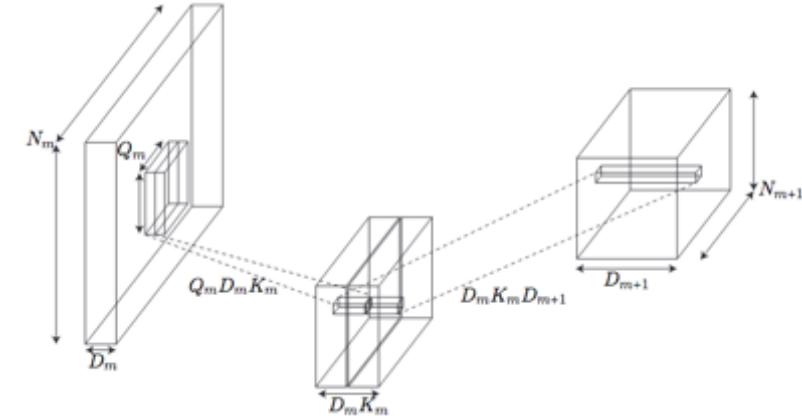
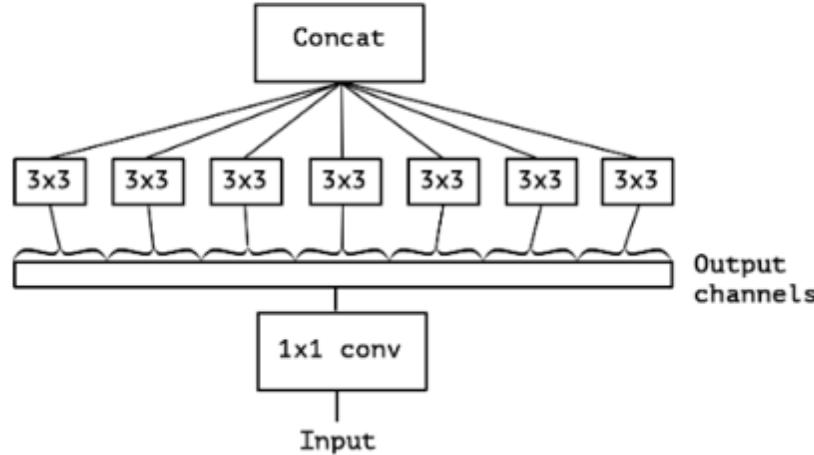
図ではチャネルを3つに分割して3x3の畳み込みに進むが、ではその最適な分割数はいくつになるのか

究極的にはチャネル方向の相互情報と2次元画像方向の相互情報は完全に分割してマッピングしてしまって良いのではないか（チャネルを複数に分割して2次元画像方向の畳み込みに進むのではなく、チャネル1つごとに2次元画像方向の畳み込みをすれば良いのではないか）

1.2. The continuum between convolutions and separable convolutions

つまり究極的なInceptionとは、下図のようにチャネル方向への 1×1 畳み込みで得られたチャネルに対して、1チャネルごとに2次元画像方向の畳み込みをかけるものであるといえる

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

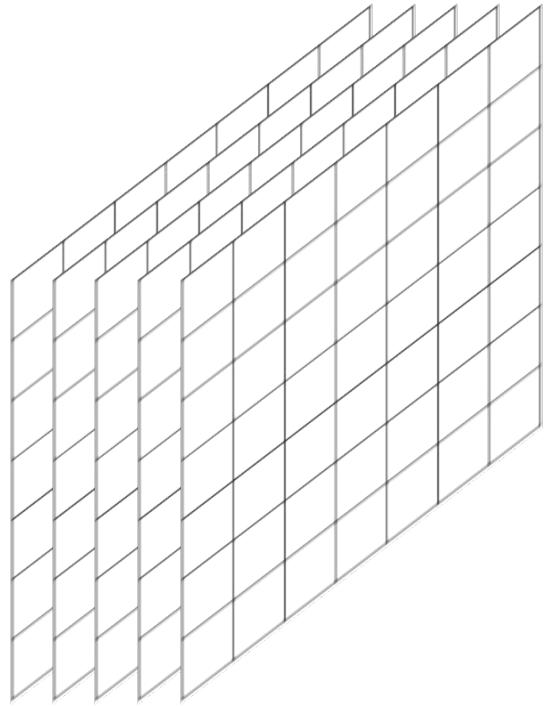


L. Sifre. Rigid-motion scattering for image classification, 2014. Ph.D. thesis.

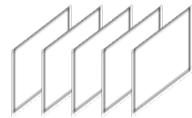
この究極的なInceptionモジュールは、2014年にすでに設計されていたDepthwise separable convolution (L. Sifre. Rigid-motion scattering for image classification, 2014. Ph.D. thesis.)とほぼ同一である

このDepthwise separable convolutionはTensorFlowやKerasでは“separable convolution”と呼ばれており、チャネルごとの2次元画像方向の畳み込み(spatial convolution)ののちに、チャネルをまたいだ 1×1 畳み込みが行われる

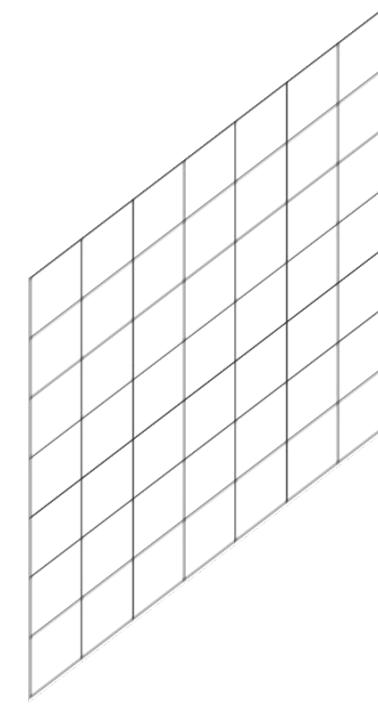
Pointwise convolution



$7 \times 7 \times 5$
入力

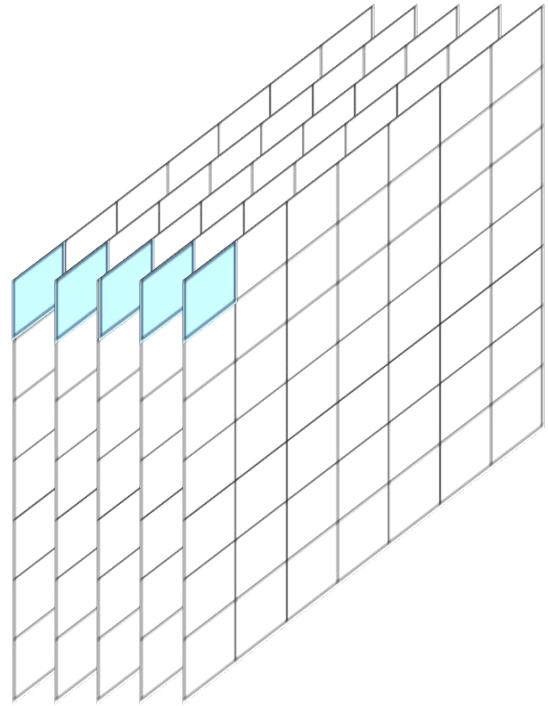


$1 \times 1 \times 5$
畳み込みフィルター
(畳み込みカーネル)

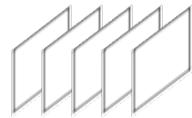


$7 \times 7 \times 1$
出力

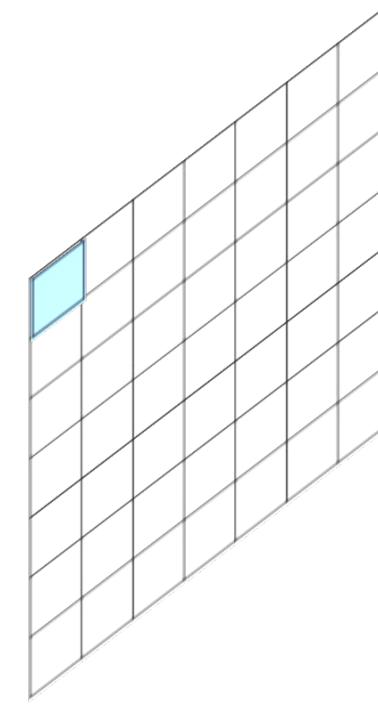
Pointwise convolution



$7 \times 7 \times 5$
入力

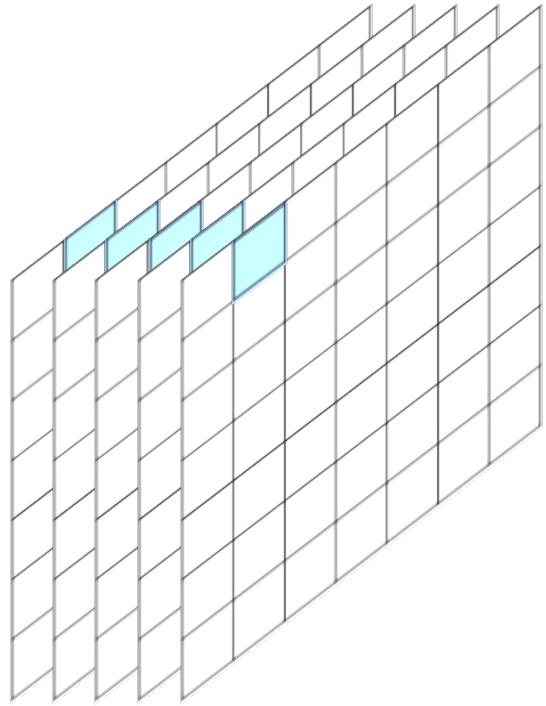


$1 \times 1 \times 5$
畳み込みフィルター
(畳み込みカーネル)

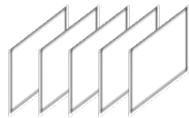


$7 \times 7 \times 1$
出力

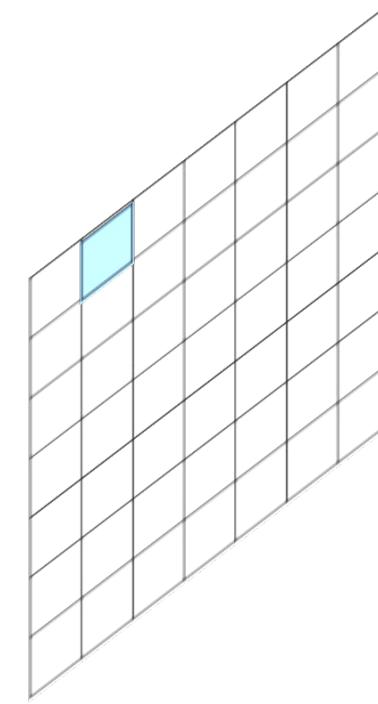
Pointwise convolution



$7 \times 7 \times 5$
入力

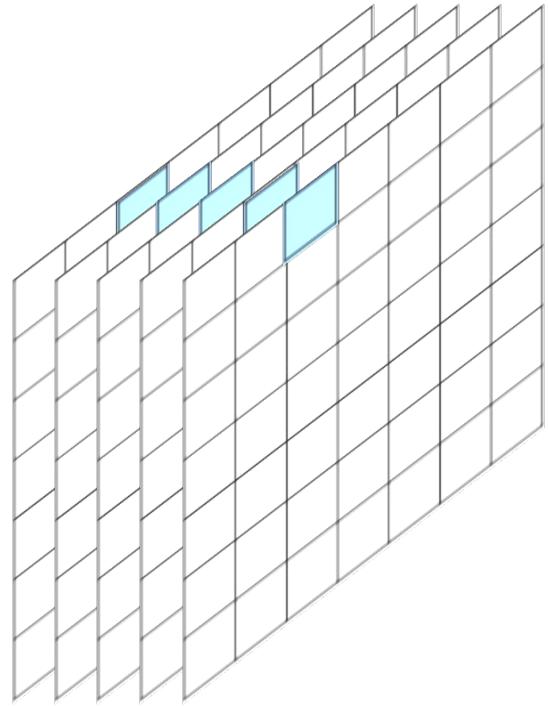


$1 \times 1 \times 5$
畳み込みフィルター
(畳み込みカーネル)

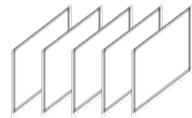


$7 \times 7 \times 1$
出力

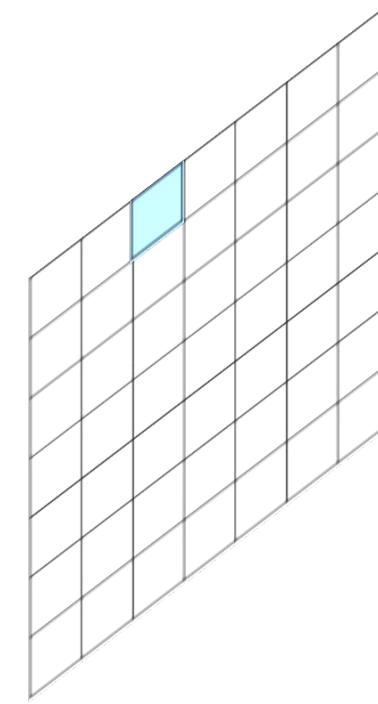
Pointwise convolution



$7 \times 7 \times 5$
入力

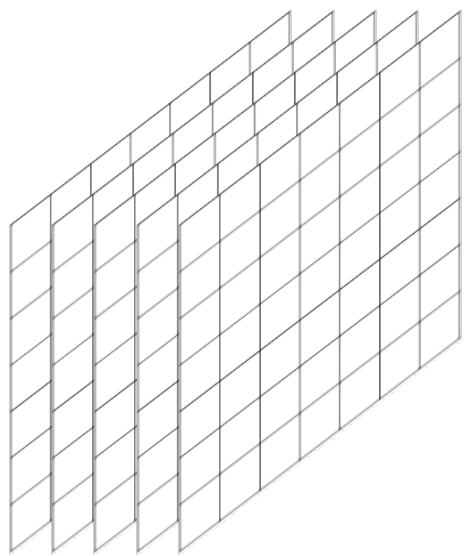


$1 \times 1 \times 5$
畳み込みフィルター
(畳み込みカーネル)



$7 \times 7 \times 1$
出力

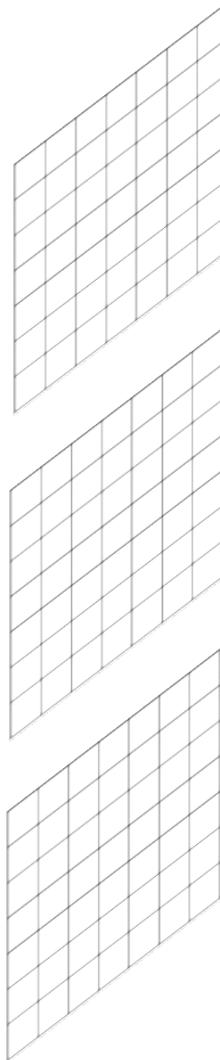
Pointwise convolution



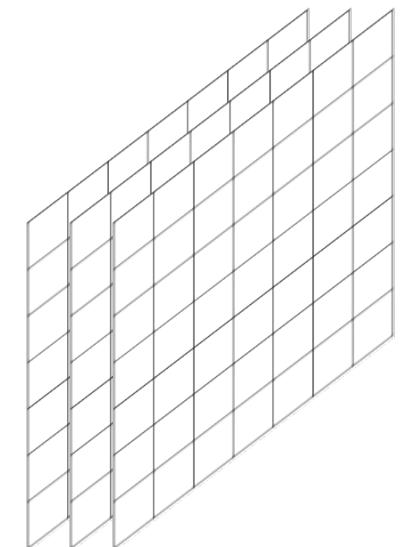
$7 \times 7 \times 5$
入力



$1 \times 1 \times 5$
畳み込みフィルター
(畳み込みカーネル)
3個

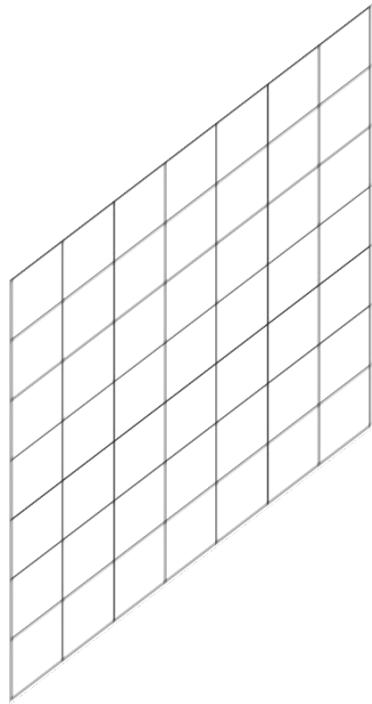


$7 \times 7 \times 1$
出力
3個

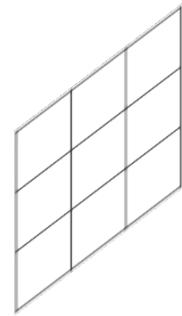


$7 \times 7 \times 3$
出力

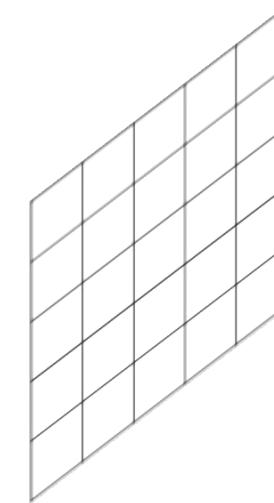
Depthwise convolution



$7 \times 7 \times 1$
入力

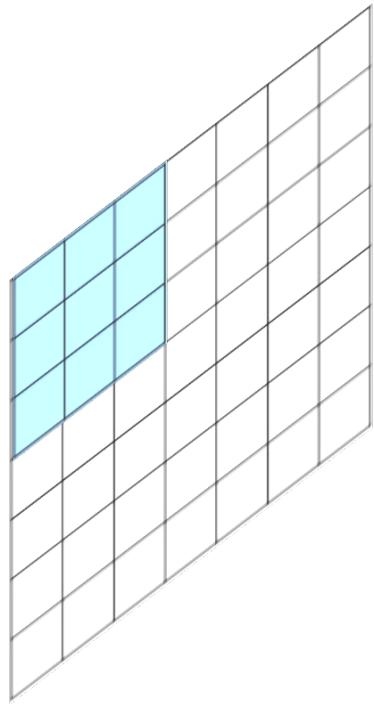


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

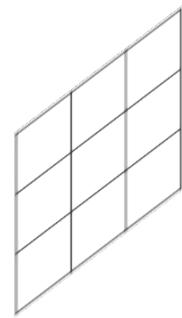


$5 \times 5 \times 1$
出力

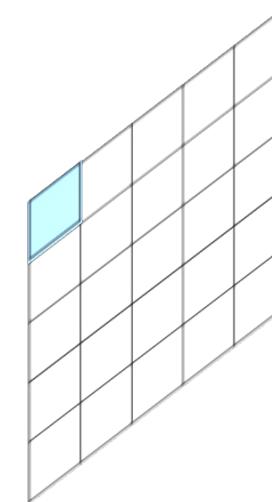
Depthwise convolution



$7 \times 7 \times 5$
入力

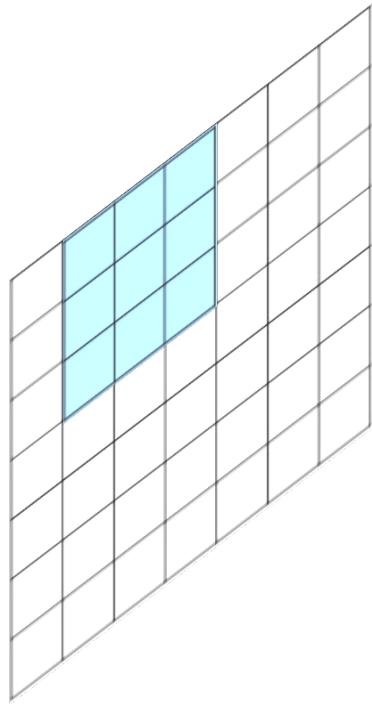


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

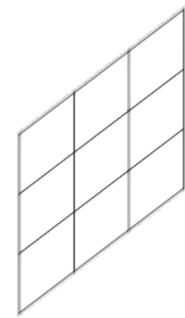


$5 \times 5 \times 1$
出力

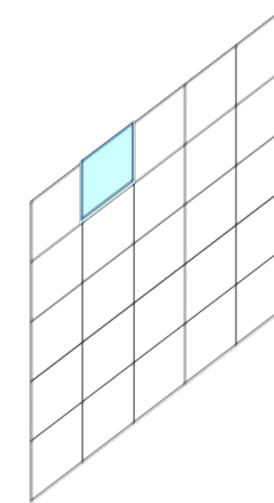
Depthwise convolution



$7 \times 7 \times 5$
入力

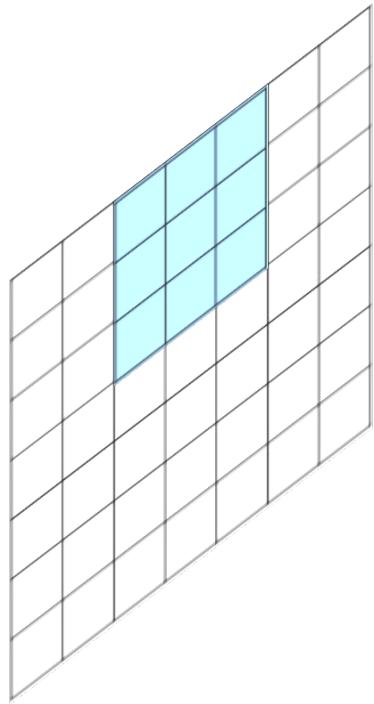


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

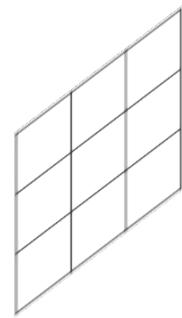


$5 \times 5 \times 1$
出力

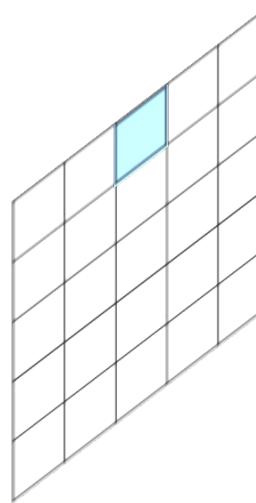
Depthwise convolution



$7 \times 7 \times 1$
入力

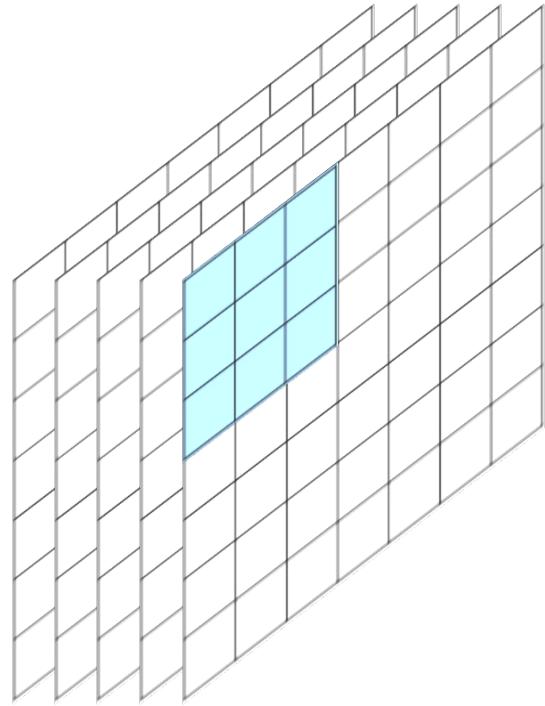


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

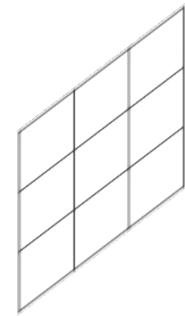


$5 \times 5 \times 1$
出力

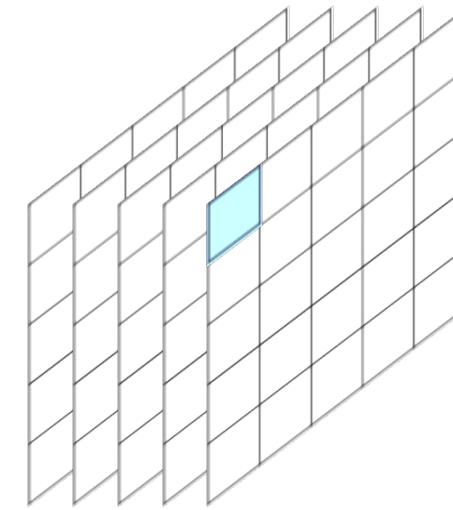
Depthwise convolution



$7 \times 7 \times 5$
入力

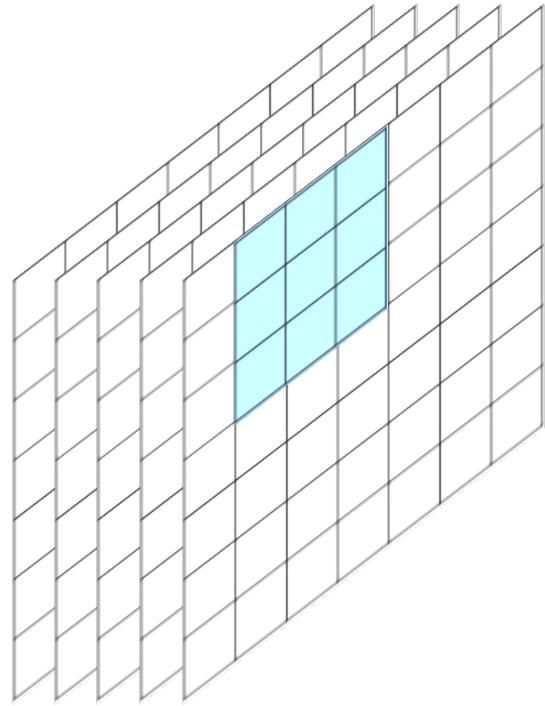


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

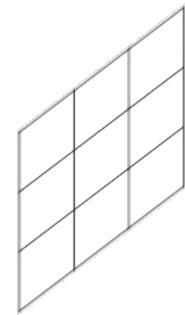


$5 \times 5 \times 5$
出力

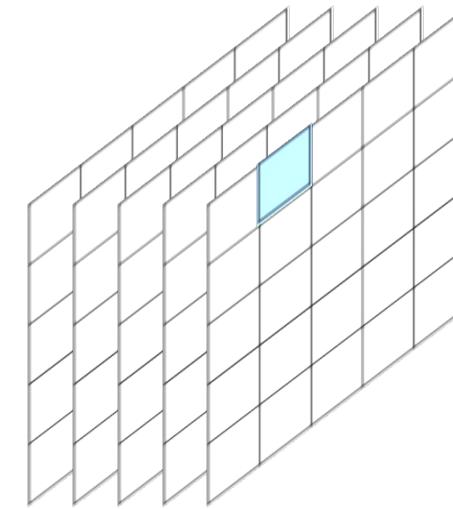
Depthwise convolution



$7 \times 7 \times 5$
入力

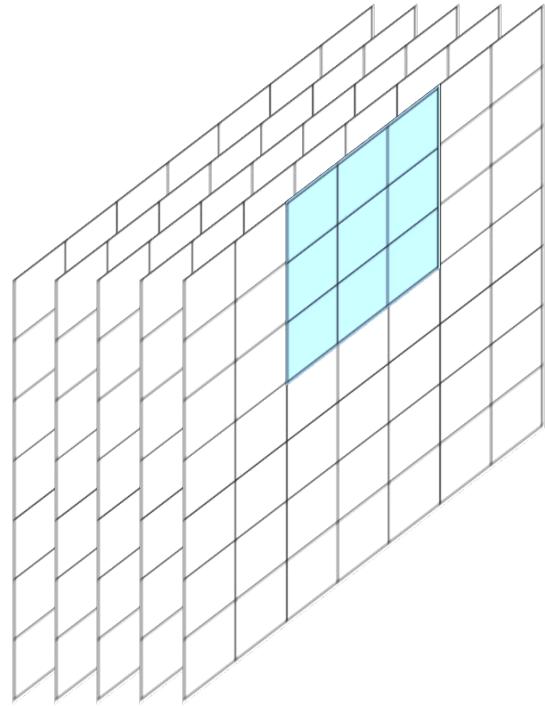


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

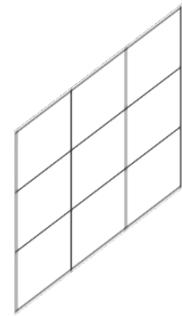


$5 \times 5 \times 5$
出力

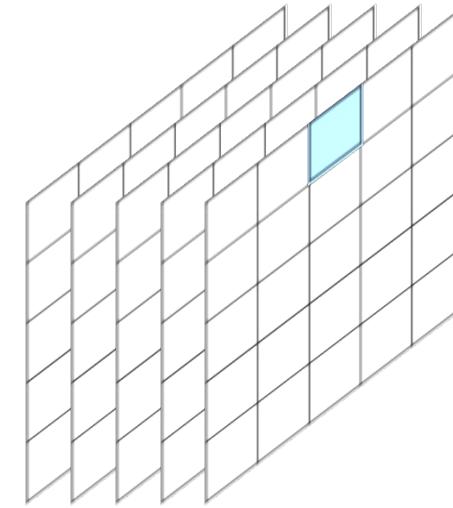
Depthwise convolution



$7 \times 7 \times 5$
入力

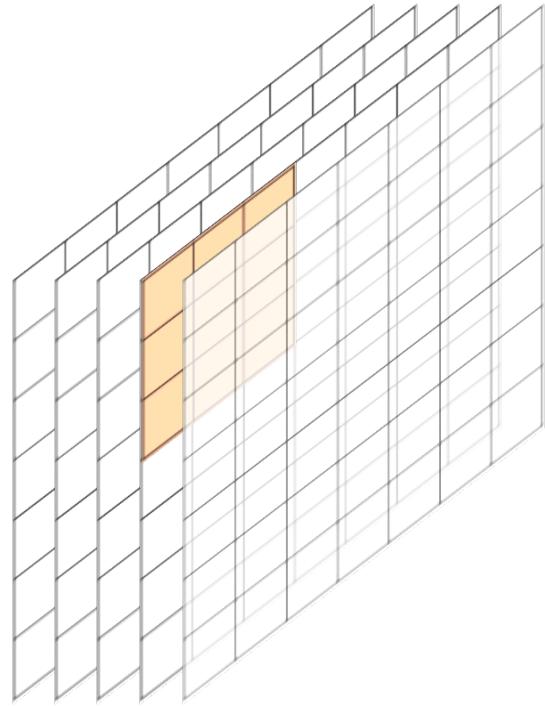


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

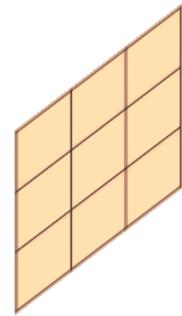


$5 \times 5 \times 5$
出力

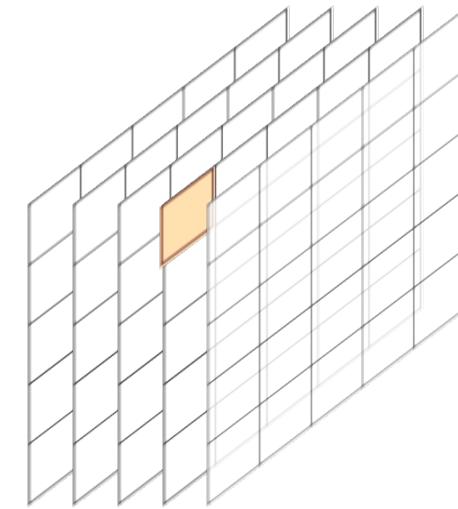
Depthwise convolution



$7 \times 7 \times 5$
入力

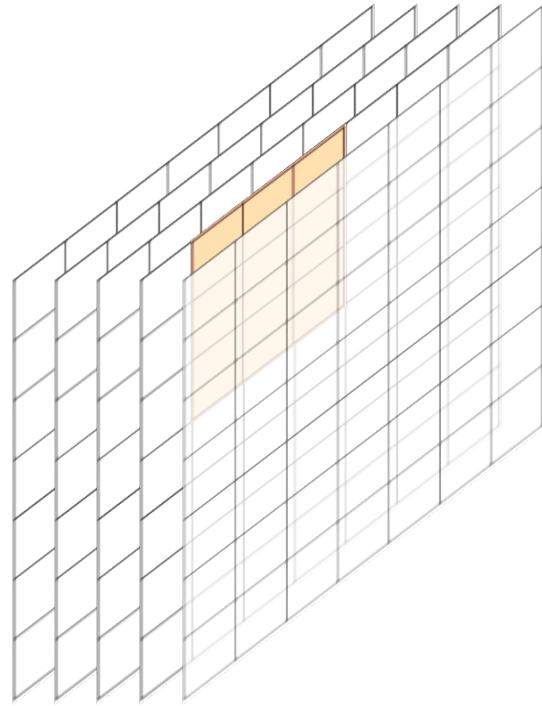


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

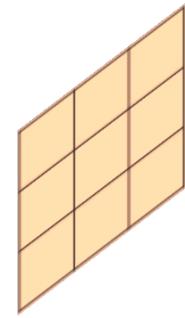


$5 \times 5 \times 5$
出力

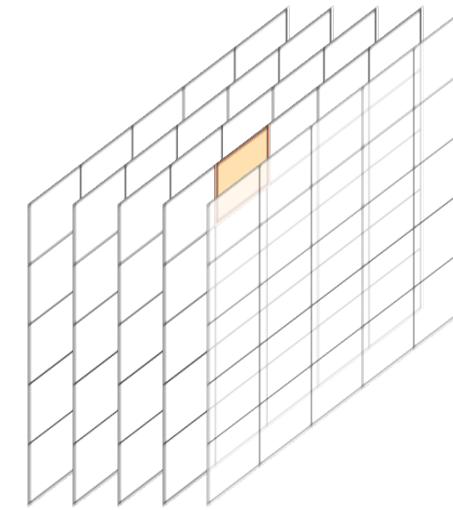
Depthwise convolution



$7 \times 7 \times 5$
入力

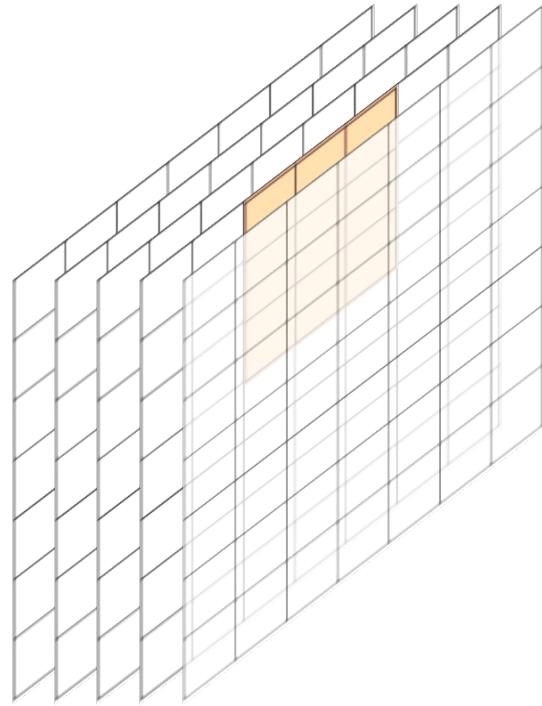


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

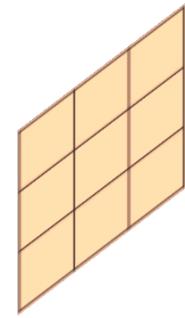


$5 \times 5 \times 5$
出力

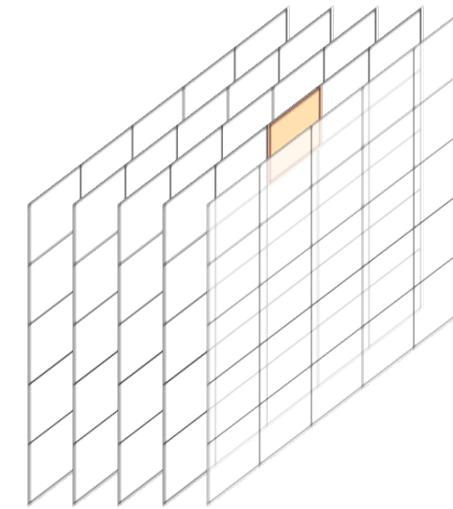
Depthwise convolution



$7 \times 7 \times 5$
入力

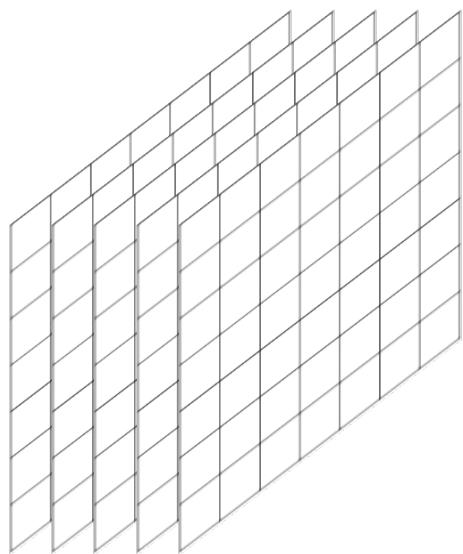


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)

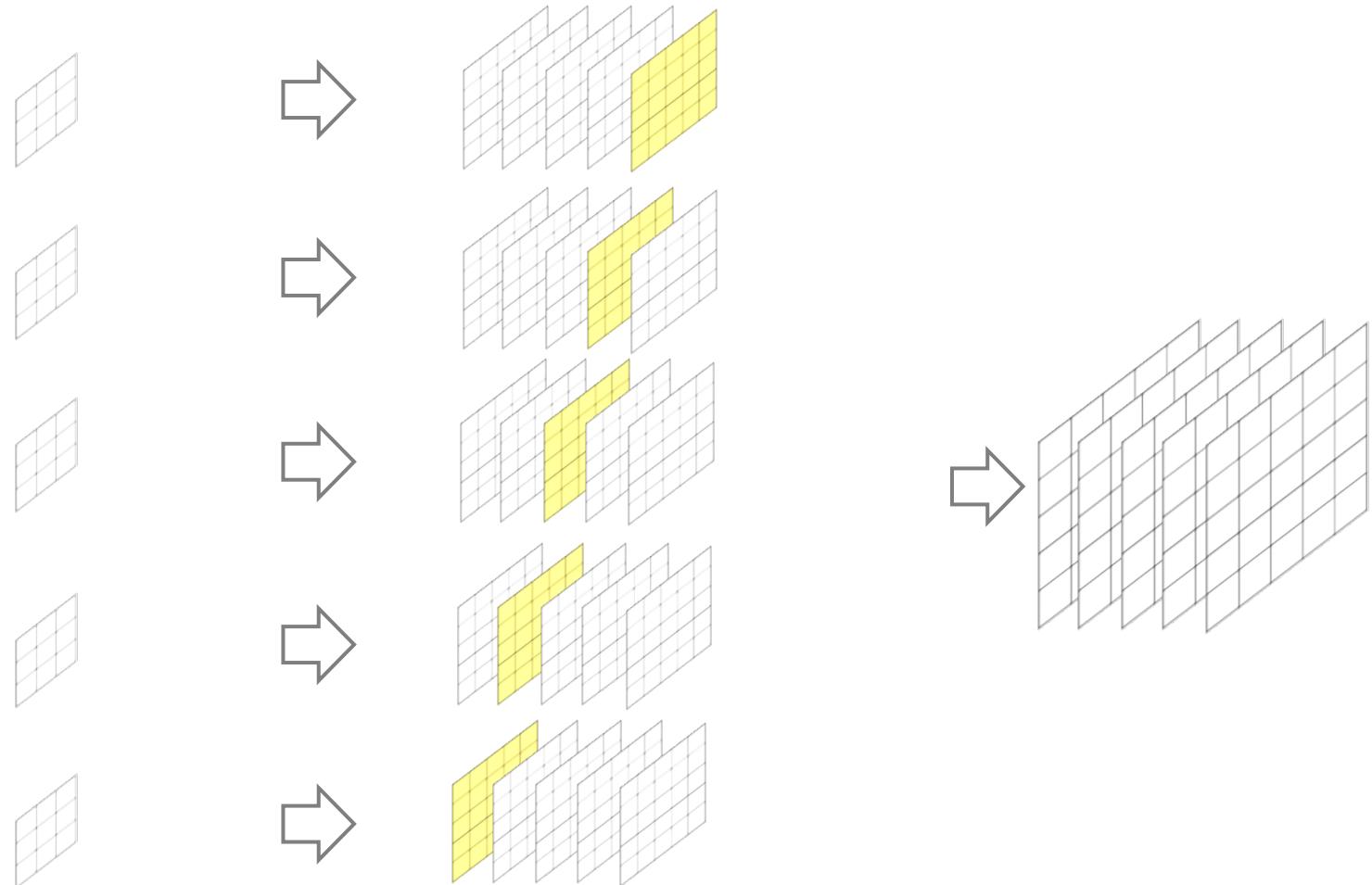


$5 \times 5 \times 5$
出力

Depthwise convolution



$7 \times 7 \times 5$
入力

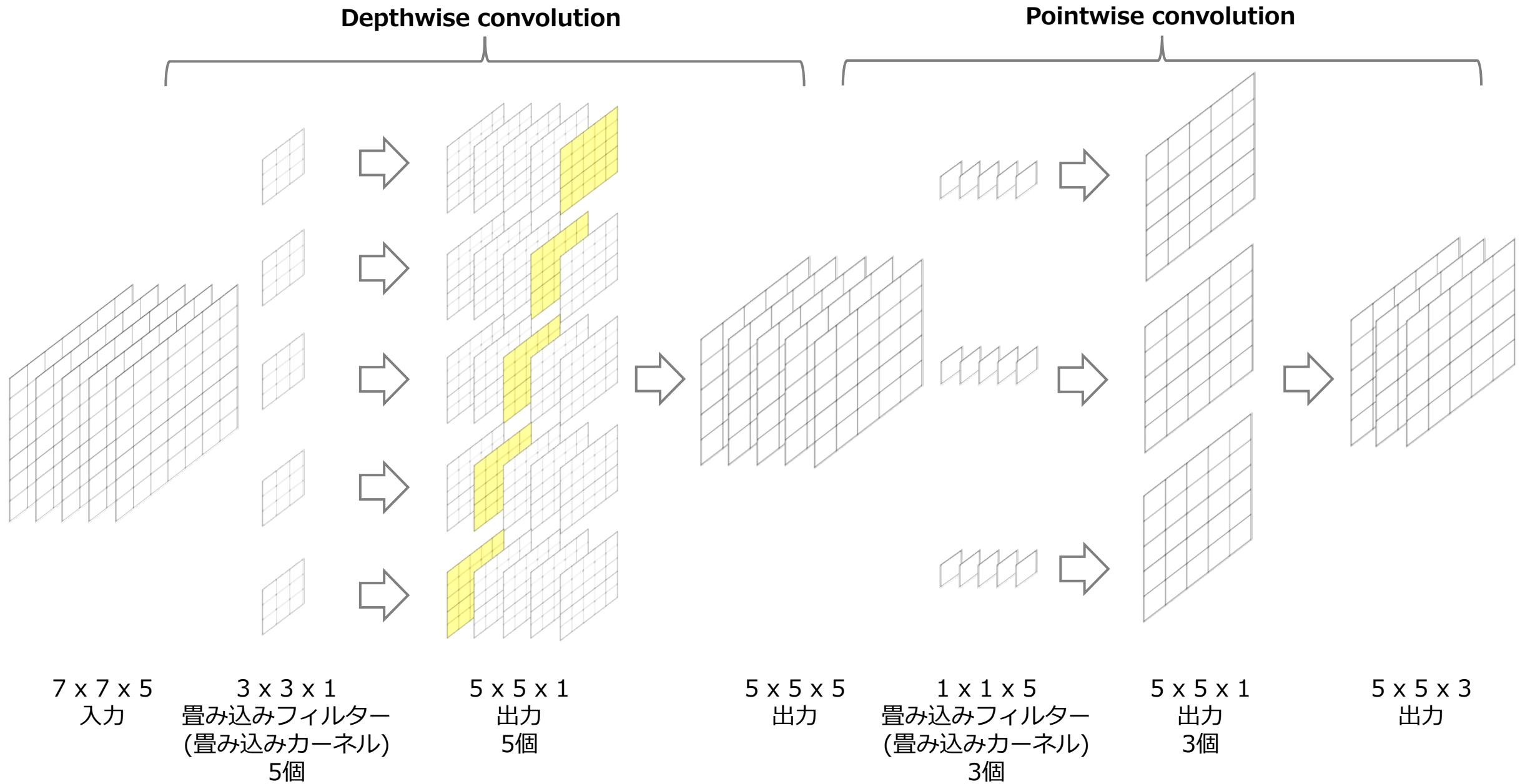


$3 \times 3 \times 1$
畳み込みフィルター
(畳み込みカーネル)
5個

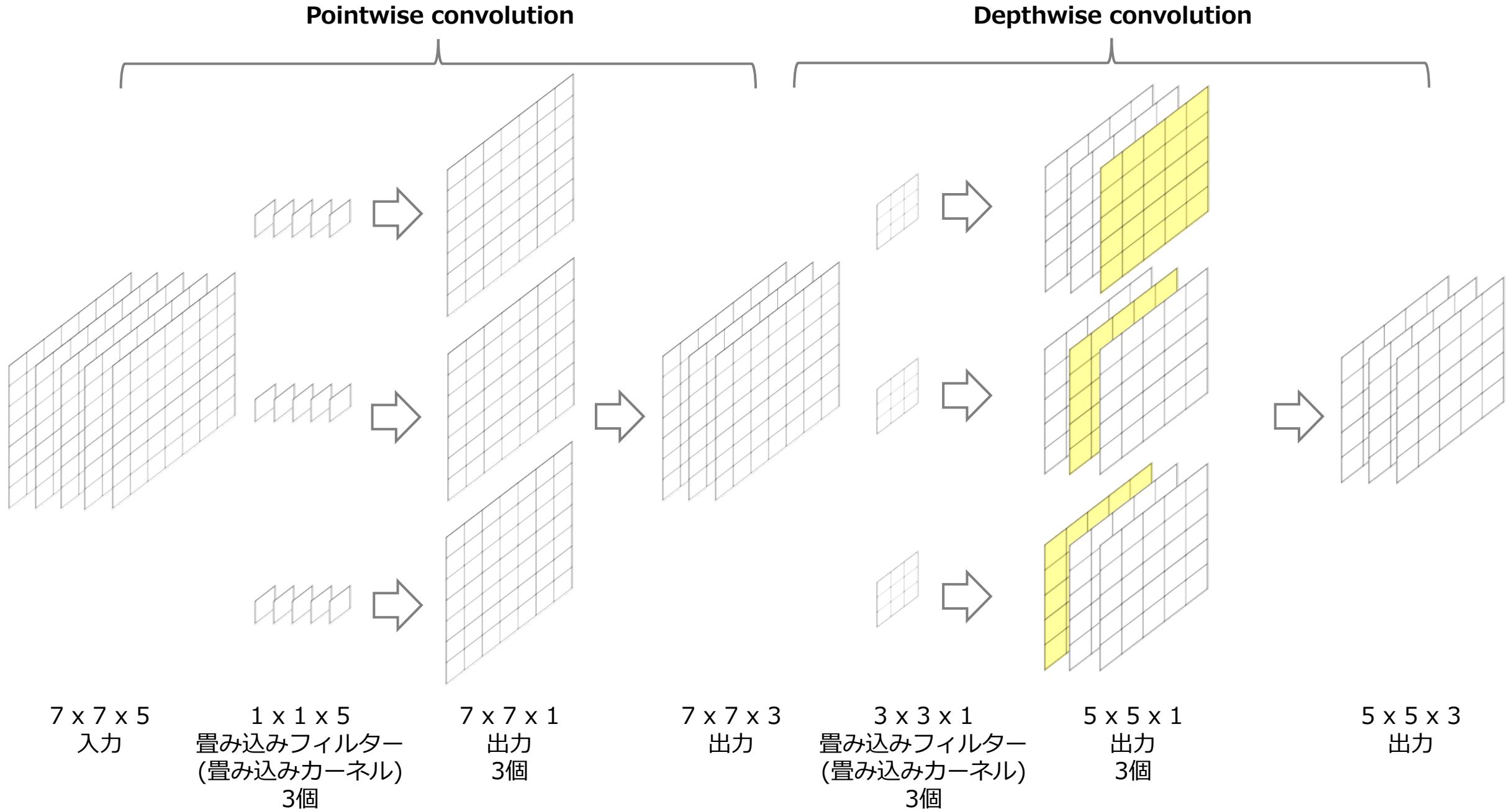
$5 \times 5 \times 1$
出力
5個

$5 \times 5 \times 5$
出力

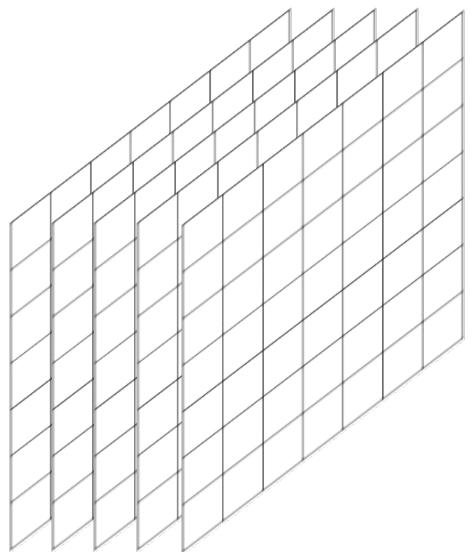
Depthwise separable convolution



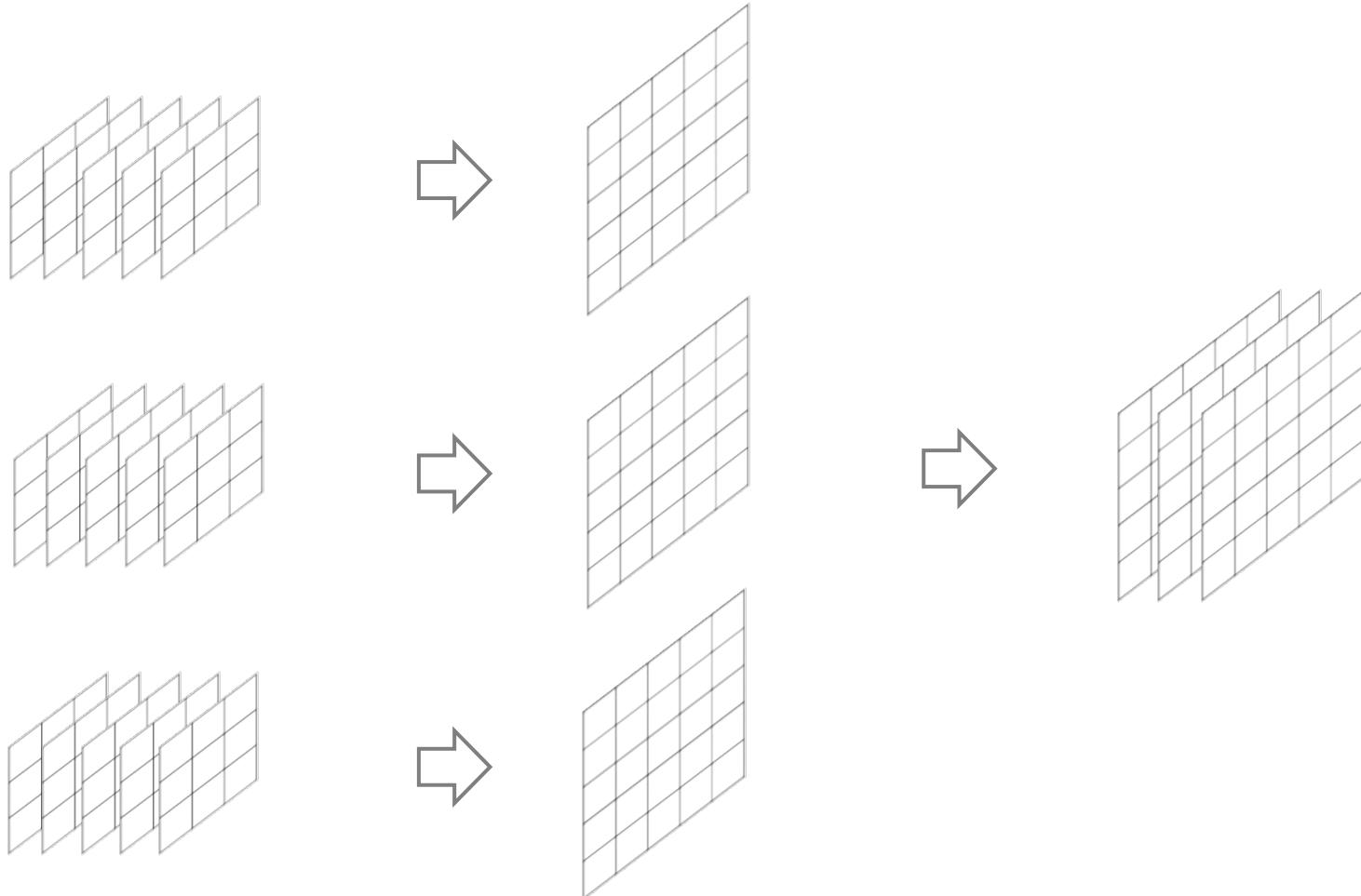
筆者らの仮説としての究極的なInceptionモジュールの形



Regular convolution



$7 \times 7 \times 5$
入力



$3 \times 3 \times 5$
畳み込みフィルター
(畳み込みカーネル)
3個

$5 \times 5 \times 1$
出力
3個

$5 \times 5 \times 3$
出力

1.2. The continuum between convolutions and separable convolutions

これまで説明してきた究極的なInceptionモジュールの形と、すでに過去に報告されているDepthwise separable convolutionsでは以下の2点が異なる

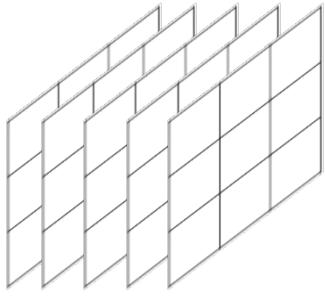
1. Depthwise separable convolutionsでは最初にチャネルごとの2次元画像方向の畳み込みが行われ、そのうちに 1×1 畳み込みでチャネルをまたいだ畳み込みが行われるが、Inceptionモジュールでは順番が逆である
2. Depthwise separable convolutionでは2次元画像方向、チャネル方向の畳み込み計算間に非線形な活性化関数が存在しないが、Inceptionモジュールでは両者とも非線形な活性化関数(ReLU)による変換を伴う

1点目の畳み込み方向の順番については、モジュールはその後積み重ねるものなのであまり重要ではないであろう

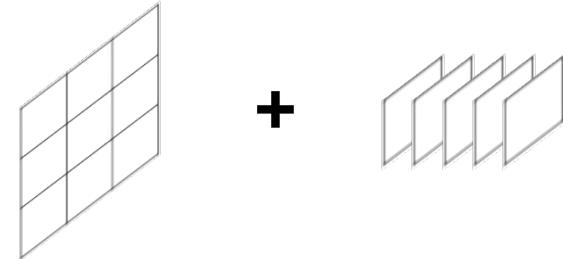
2点目の非線形関数の有無についてはthe experimental sectionでさらに言及する(Fig10)

1.2. The continuum between convolutions and separable convolutions

Regular convolution



Depthwise separable convolution



両者は概念的に両極端なもの

Inceptionモジュールはこの概念の中間に位置する手法であった

それならばInceptionネットワークモデルのInceptionモジュールをDepthwise separable convolutionに完全に置き換えるべきなのではないか

2. Prior work

畳み込みニューラルネットワーク

(VGG-16は特に設計戦略上、複数の点で提案手法に影響を与えている)

Inceptionネットワークモデルシリーズ

Depthwise separable convolutions

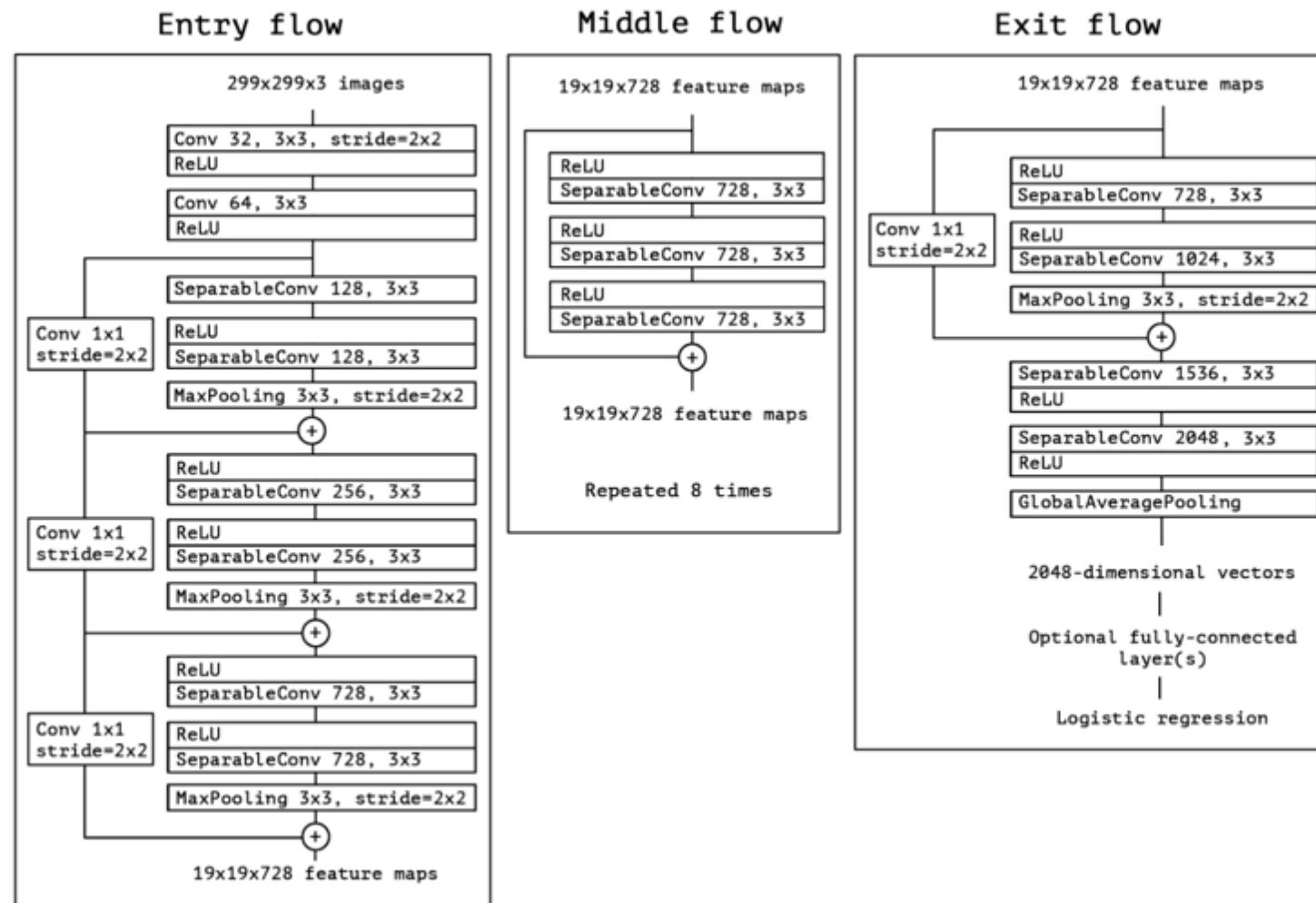
(この論文の提案手法はほぼこれにのっとっている)

Residual Learning

(提案手法内で使用している)

3. The Xception architecture

提案手法はチャネル方向と2次元画像方向の畳み込みを完全に分離したInceptionであり、“Extreme Inception”からとてXceptionと名付けた



Xceptionは線形なresidual connectionの接続を有する14個のモジュール(畳み込み層は全部で36層)から成る

SeparableConv部分は Depthwise→Pointwiseの順で 畳み込みをしており非線形関数は挟まず

KerasやTensorflow-slimを用いれば30-40行でコードが書けるシンプルな構造である

4. Experimental evaluation

ここまで仮説の検証のため、XceptionとInception V3の比較を行った

両者はパラメータ数がほぼ等しい

Table 3. Size and training speed comparison.

	Parameter count	Steps/second
Inception V3	23,626,728	31
Xception	22,855,952	28

ImageNetデータセットの1000分類タスク、およびIJFTデータセットの17000分類(マルチラベル)タスクで両ネットワークの性能を比較した

4.1. The JFT dataset

JFTデータセットはGoogleが有するマルチラベルの大規模画像データセット(3億5000万枚以上、17000クラス)

Validationには付属しているFastEval14kセット(14000枚、6000クラス(1画像あたり36.5ラベル))を用いた

FastEval14kセットによる検証にあたっては予測結果top100のMean Average Precision を用いた
(さらにラベルごとにsocial mediaでどれほどcommonな画像かに基づいてスコアを重み付けした)

4.2. Optimization configuration

データセットに応じて異なるハイパーパラメータを使用

On ImageNet

Optimizer: SGD

Momentum: 0.9

Initial learning rate: 0.045

Learning rate decay: decay of rate 0.94 every 2 epochs

On JFT

Optimizer: RMSprop

Momentum: 0.9

Initial learning rate: 0.001

Learning rate decay: decay of rate 0.9 every 3,000,000 samples

XceptionとInception V3でそれぞれ同じパラメータを用いた

この設定はInception V3でチューニングされたハイパーパラメータである

4.3. Regularization configuration

Weight decay (ImageNetとJFT両者に対して使用)

Inception V3: 4e-5

Xception: 1e-5

Dropout (Inception V3とXception両者で使用)

ImageNet: ロジスティック回帰のレイヤーを除いて0.5に設定

JFT: データが大きいので学習の範囲内で過学習を危惧する必要ないと判断しDropoutは適用せず

Auxiliary loss tower 使用せず

4.4. Training infrastructure

それぞれ60 NVIDIA K80 GPUを用いて学習

ImageNetの学習ではdata parallelism with synchronous gradient descent、JFTの学習では asynchronous gradient descentでパラメータを更新

ImageNetの学習はそれぞれ3日、JFTの学習ではそれぞれ1ヶ月以上を要した(JFTはデータをフルに学習できていない。全部終えるのに3ヶ月以上かかる。)

Data parallelism with synchronous gradient descent

Algorithm 3 SimuParallelSGD(Examples $\{c^1, \dots, c^m\}$, Learning Rate η , Machines k)

```
Define  $T = \lfloor m/k \rfloor$ 
Randomly partition the examples, giving  $T$  examples to each machine.
for all  $i \in \{1, \dots, k\}$  parallel do
    Randomly shuffle the data on machine  $i$ .
    Initialize  $w_{i,0} = 0$ .
    for all  $t \in \{1, \dots, T\}$ : do
        Get the  $t$ th example on the  $i$ th machine (this machine),  $c^{i,t}$ 
         $w_{i,t} \leftarrow w_{i,t-1} - \eta \partial_w c^i(w_{i,t-1})$ 
    end for
end for
Aggregate from all computers  $v = \frac{1}{k} \sum_{i=1}^k w_{i,t}$  and return  $v$ .
```

Zinkevich, Martin, et al. "Parallelized stochastic gradient descent." Advances in neural information processing systems. 2010.

データを複数マシンに分散し、各マシン上で計算されたパラメータの平均を新規のパラメータとする

Asynchronous gradient descent

Algorithm 1: Async-SGD worker k

```
Input: Dataset  $\mathcal{X}$ 
Input:  $B$  mini-batch size
1 while True do
2     Read  $\hat{\theta}_k = (\theta[0], \dots, \theta[M])$  from PSs.
3      $G_k^{(t)} := 0$ .
4     for  $i = 1, \dots, B$  do
5         Sample datapoint  $\tilde{x}_i$  from  $\mathcal{X}$ .
6          $G_k^{(t)} \leftarrow G_k^{(t)} + \frac{1}{B} \nabla F(\tilde{x}_i; \hat{\theta}_k)$ .
7     end
8     Send  $G_k^{(t)}$  to parameter servers.
9 end
```

Algorithm 2: Async-SGD Parameter Server j

```
Input:  $\gamma_0, \gamma_1, \dots$  learning rates.
Input:  $\alpha$  decay rate.
Input:  $\theta^{(0)}$  model initialization.
1 for  $t = 0, 1, \dots$  do
2     Wait for gradient  $G$  from any worker.
3      $\theta^{(t+1)}[j] \leftarrow \theta^{(t)}[j] - \gamma_t G[j]$ .
4      $\bar{\theta}^{(t)}[j] = \alpha \bar{\theta}^{(t-1)}[j] + (1 - \alpha) \theta^{(t)}[j]$ .
5 end
```

Chen, Jianmin, et al. "Revisiting distributed synchronous SGD." arXiv preprint arXiv:1604.00981(2016)

パラメータサーバにあるパラメータを各ワーカーマシンで読み取ってパラメータの更新量を計算し、パラメータサーバ側ではワーカーマシンで計算が終了した更新量にそって順次パラメータを更新する

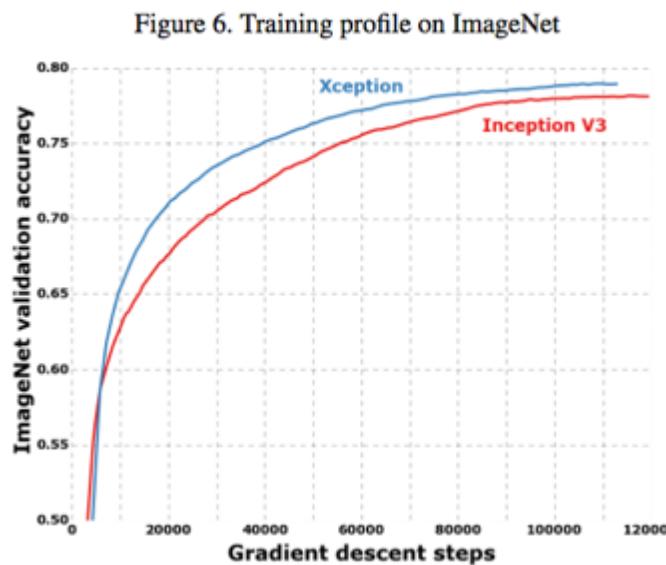
4.5. Comparison with Inception V3

4.5.1 Classification performance

ImageNet

	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

(VGG-16とResNet-152の結果は過去の報告より)



JFT

	FastEval14k MAP@100
Inception V3 - no FC layers	6.36
Xception - no FC layers	6.70
Inception V3 with FC layers	6.50
Xception with FC layers	6.78

(全結合層の有無の2条件でそれぞれを比較)

Figure 7. Training profile on JFT, without fully-connected layers

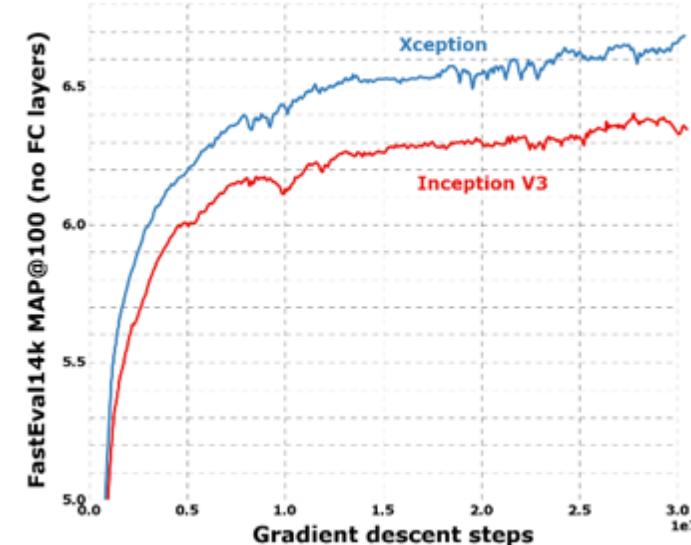
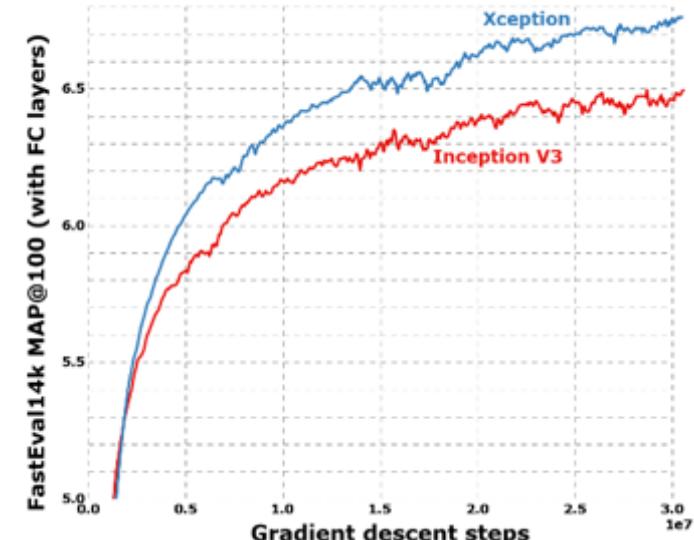


Figure 8. Training profile on JFT, with fully-connected layers



XceptionはInception V3との比較において、ImageNetデータセットで僅かなaccuracyの改善を、JFTデータセットでaccuracyの大きな改善を達成した

4.5.2 Size and speed

Table 3. Size and training speed comparison.

	Parameter count	Steps/second
Inception V3	23,626,728	31
Xception	22,855,952	28

パラメータ数はImageNet学習時、FC layer抜きのものStep数は60 K80 GPUsを用いて synchronous gradient descentでImageNetを学習した時のもの

Xceptionは処理速度がわずかにInception V3より遅い

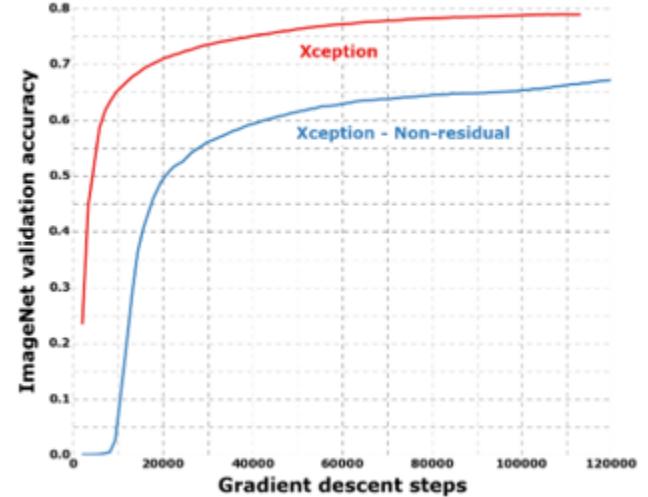
4.6. Effect of the residual connections

Residual connectionの恩恵を確認するためにResidual connection抜きの条件も比較した

Residual connection抜きでは明らかに収束が遅くaccuracyが悪い

ただしResidual connection抜きの条件でハイパーパラメータを調整していないのでResidual connectionが収束に本当に必須かはわからない

Figure 9. Training profile with and without residual connections.



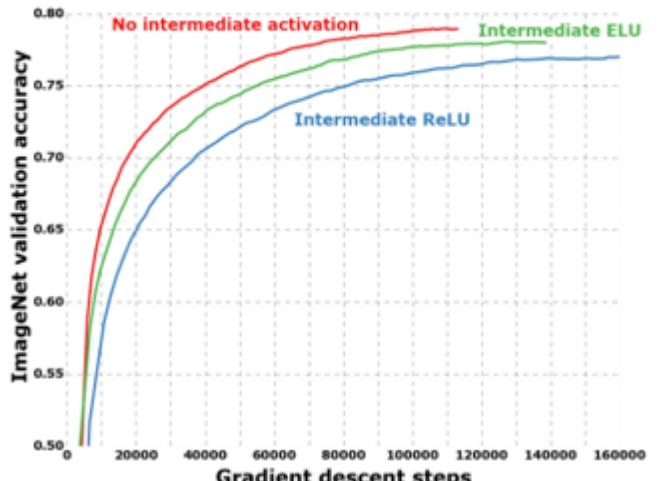
4.7. Effect of an intermediate activation after pointwise convolutions

XceptionにおけるDepthwiseとPointwiseの間の活性化関数(非線形計算)の影響を検証

非線形関数を挟まない方が収束も早く最終的な分類のaccuracyも良かった

二次元画像方向の畳み込みが適用される中間特徴空間のdepthが、非線形関数の必要・不必要に関わるのではないか

Figure 10. Training profile with different activations between the depthwise and pointwise operations of the separable convolution layers.



5. Future directions

InceptionモジュールはRegular convolutionsとDepthwise separable convolutionsが対極として形成する離散的スペクトラムの中の中間に位置する手法である

我々の検証では、Inceptionモジュールの究極的な形態としてDepthwise separable convolutionが有益な結果をもたらした

しかし我々の提唱するスペクトラムの中でDepthwise separable convolutionが最適であるという保証はない

我々の提唱するスペクトラムの中で中間に位置する他のネットワーク形態がさらなる有用性を潜在的に有している可能性があり、その点についてはさらなる検証が必要である

6. Conclusions

Regular convolutionとDepthwise separable convolutionが対極に存在し、Inceptionモジュールがどのようにその対極の中間に位置する概念を提示した

提案した概念に基づいて、Inception V3と同程度のパラメータ数を持つXceptionと名付けた新たなネットワーク構造を提案した

Inception V3との比較において、提案手法であるXceptionがImageNetデータセットを用いた画像分類性能の評価で僅かな改善を、IJFTデータセットを用いた画像分類性能の評価で大きな改善を達成することを示した

Major concern (全ての論文は批判的に読む)

Depthwise separable convolutionsとResidual connectionの多段の積み重ねによってInception V3を上回る性能が得られたという内容

Depthwise separable convolutionsそのものに関しては当該論文発表時にすでに新規の手法ではない

チャネル方向の畳み込みの分離に関して、確かにRegular convolutionとDepthwise separable convolutionは対極にあるが、Inceptionモジュールがその間に存在する手法であるかどうかについては非線形関数の有無に対する挙動の違いもあり、チャネル方向の畳み込みの分離だけでは説明がつかない部分が残る

十分にパラメータチューニングをしたInception V3とパラメータチューニングをほとんどしていないXceptionの比較、という比較の前提条件の正当性に対しては客観的判断が困難

また、筆者らが論文内に記載している通り、Residual connectionの必要性の評価については検証が不十分なままとなっている

XceptionとInception V3の著明な性能の差は主にGoogle内部のIJFTデータセットを用いて証明されており、外部による検証が困難