# Xception : Deep Learning with Depthwise Separable Convolutions

Franc¸ois Chollet Google, Inc. fchollet@google.com

ワンハオチン

# Xception

- 我々は畳み込みニューラルネットワークにおけるInceptionモジュールが通常の畳み込みと"depthwise separable convolution"の中間段階であるとして解釈できることを提唱する。この点において、"depthwise separable convolution"はInceptionモジュールの中で最もtowerの数が大きなケースだと理解されうる。この洞察はInceptionに触発された新しいDeepLearningのネットワーク構造を我々に示唆してくれ、その新しいネットワーク構造ではInceptionの代わりに"depthwise separable convolutions"を用いている
- 我々はXceptionと名付けたこの構造がImageNetデータセットを用いてInceptionV3をわずかに上回り、また3.5億枚の1.7万クラスのより大きなデータセットにおいてははるかに高いパフォーマンスを出したことを示す。

# Inception

- **Convolutional Neural Network（C N N）**

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

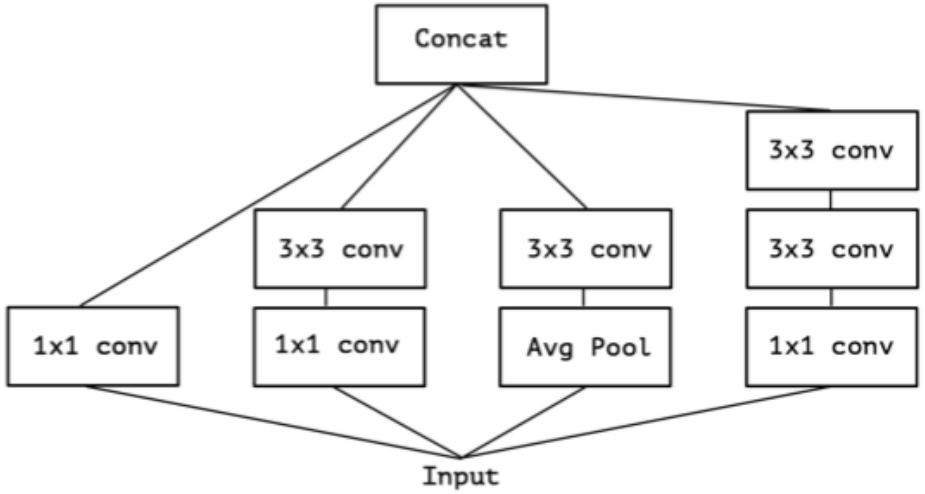
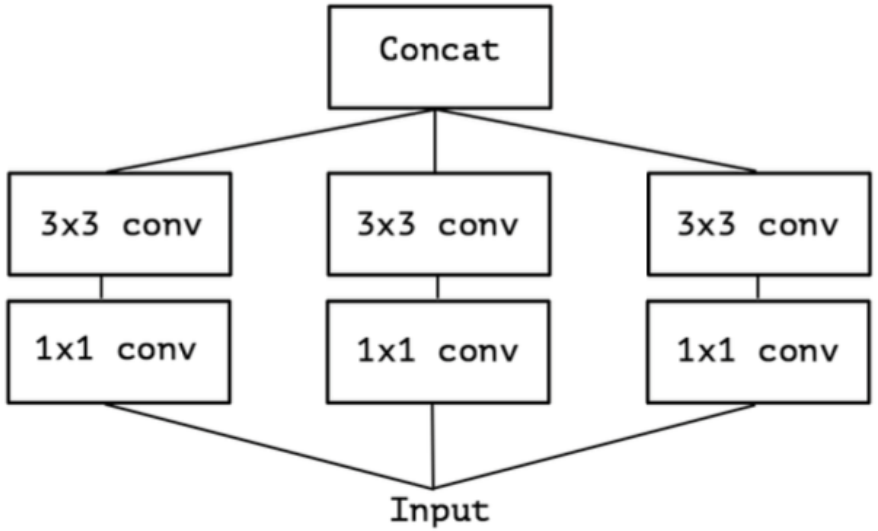
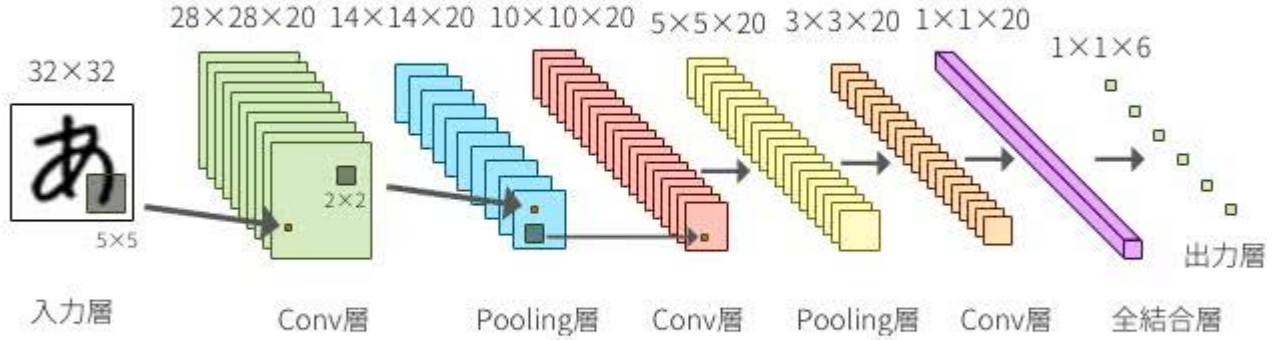



Figure 2. A simplified Inception module.



# Convolutional Neural Network（ＣＮＮ）

Convolutional Neural Networkは略してCNNと呼ばれる。CNNは一般的な順伝播型のニューラルネットワークとは違い、全結合層だけでなく**畳み込み層(Convolution Layer)**と**プーリング層(Pooling Layer)**から構成されるニューラルネットワークのことだ。
畳み込み層とプーリング層では下図のように入力のニューロンの一部の領域を絞って、局所的に次の層へと対応付けをしていく。各層は**フィルタ**と呼ばれる検出器をいくつも持っているイメージだ。

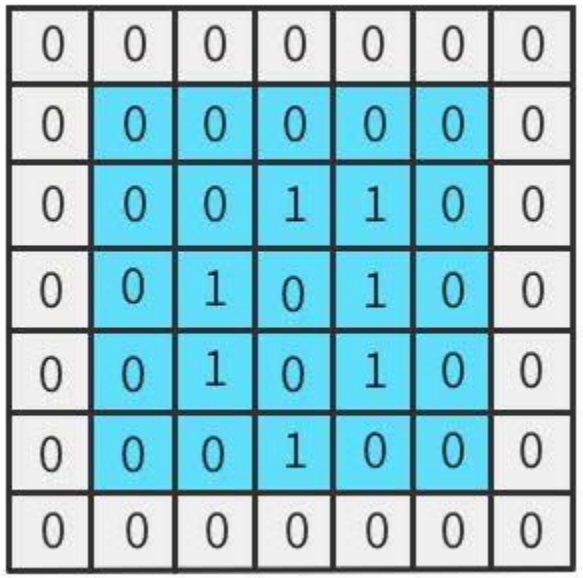



画像認識の例でいうと、最初の層でエッジを検出して、次の層でテクスチャを検出し、さらに次のそうではより抽象的な猫の耳などの特徴を検出したりする。CNNはこういった特徴を抽出するための検出器である**フィルタのパラメータを自動で学習していく**のだ。

# Convolutional Neural Networkの構成要素

- **ゼロパディング（zero padding）**
- **ストライド（ stride ）**
- **Fully Connected層**
- **Convolution層**
- **Pooling層**

# ゼロパディング（zero padding）




中間のピクセルポイントは複数回の計算に参加することがありますが、端のピクセルポイントは一回だけ参加することができます。だから私たちの結果は端の特徴を失うかもしれません。

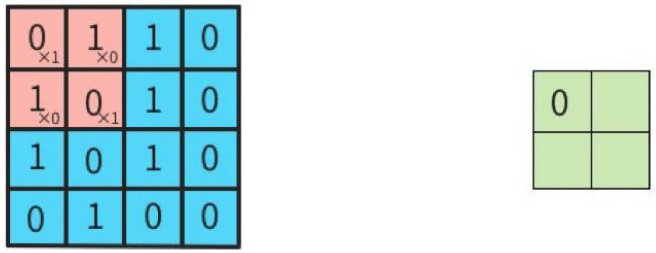
ゼロパディングは上図のように、入力の特徴マップの周辺を0で埋める。こうすることで以下のようなメリットがある。

- 端のデータに対する畳み込み回数が増えるので端の特徴も考慮されるようになる
- 畳み込み演算の回数が増えるのでパラメーターの更新が多く実行される
- カーネルのサイズや、層の数を調整できる

Convolution層とPooling層で出力サイズは次第に小さくなるので、ゼロパディングでサイズを増やしたりすると層の数を増やすことができる。

# ストライド（stride）

- ストライドは畳み込みの適用間隔のことで、図で表すとわかりやすい。入力データ4×4に対してストライドを1とすると以下のようになる。




- これがストライド2になると2つ間隔でフィルタが進むようになる。

# Fully Connected層

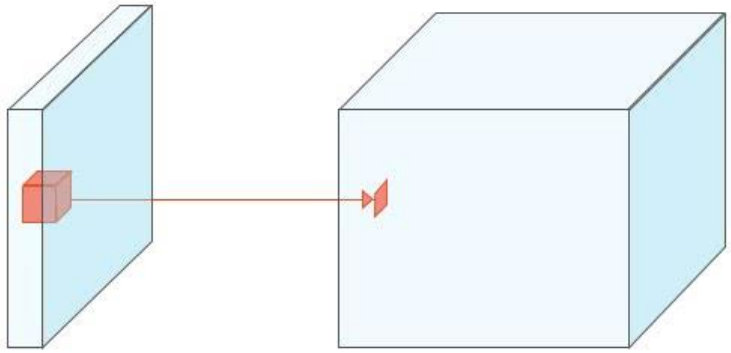
- 層の一つ目は、一般的なニューラルネットワークでも見られるFully Connected層（全結合層）である。各層のユニットは次の層のユニットと**全て繋がっている**。
- それぞれのユニットとの接続は**重み**を持っていて、図では濃さで表している。入力は1次元のベクトルで、出力も1次元のベクトルとなる。
- CNNでは出力層の手前で使われることが多く、この部分が検出された特徴の組み合わせから、予測結果に分類するための**識別部**となる。

- 実際には、全連結層は極端な場合の畳み込み層としてもよく、その畳み込み核サイズは入力ユニットサイズであるため、出力行列の高さと幅の大きさはいずれも1である。

| | Batch | | Height | | Width | | In Depth | | Out Depth |
|---|---|---|---|---|---|---|---|---|---|
| **Input** | 256 | × | 32 | × | 32 | × | 512 | | |
| **Kernel** | | | 32 | × | 32 | × | 512 | × | 4096 |
| **Output** | 256 | × | 1 | × | 1 | | | × | 4096 |

# Convolution層

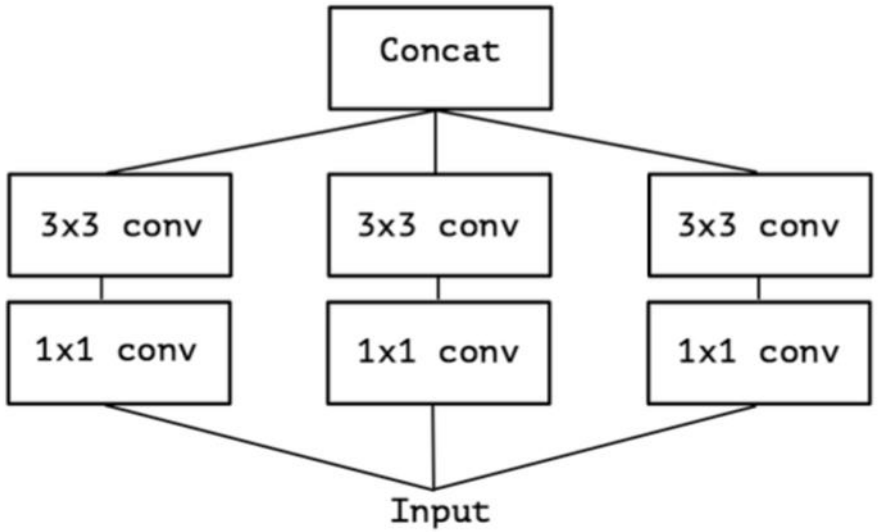
- Convolution層は空間的な情報を維持することができる。widthとheightとdepthの3次元を入力値として3次元を出力する例を考えてみよう。



| | Batch | | Height | | Width | | In Depth | | Out Depth |
|---|---|---|---|---|---|---|---|---|---|
| Input | 256 | × | 227 | × | 227 | × | 3 | | |
| Kernel | | | 11 | × | 11 | × | 3 | × | 96 |
| Output | 256 | × | 55 | × | 55 | | | × | 96 |

# 1 x 1 conv

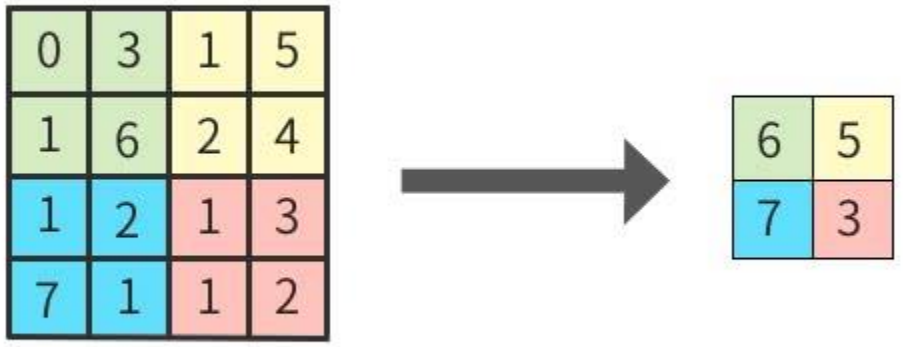
Figure 2. A simplified Inception module.




| | Batch | | Height | | Width | | In Depth | | Out Depth |
|---|---|---|---|---|---|---|---|---|---|
| **Input** | 256 | × | 227 | × | 227 | × | 512 | | |
| **Kernel** | | | 1 | × | 1 | × | 512 | × | 32 |
| **Output** | 256 | × | 227 | × | 227 | | | × | 32 |

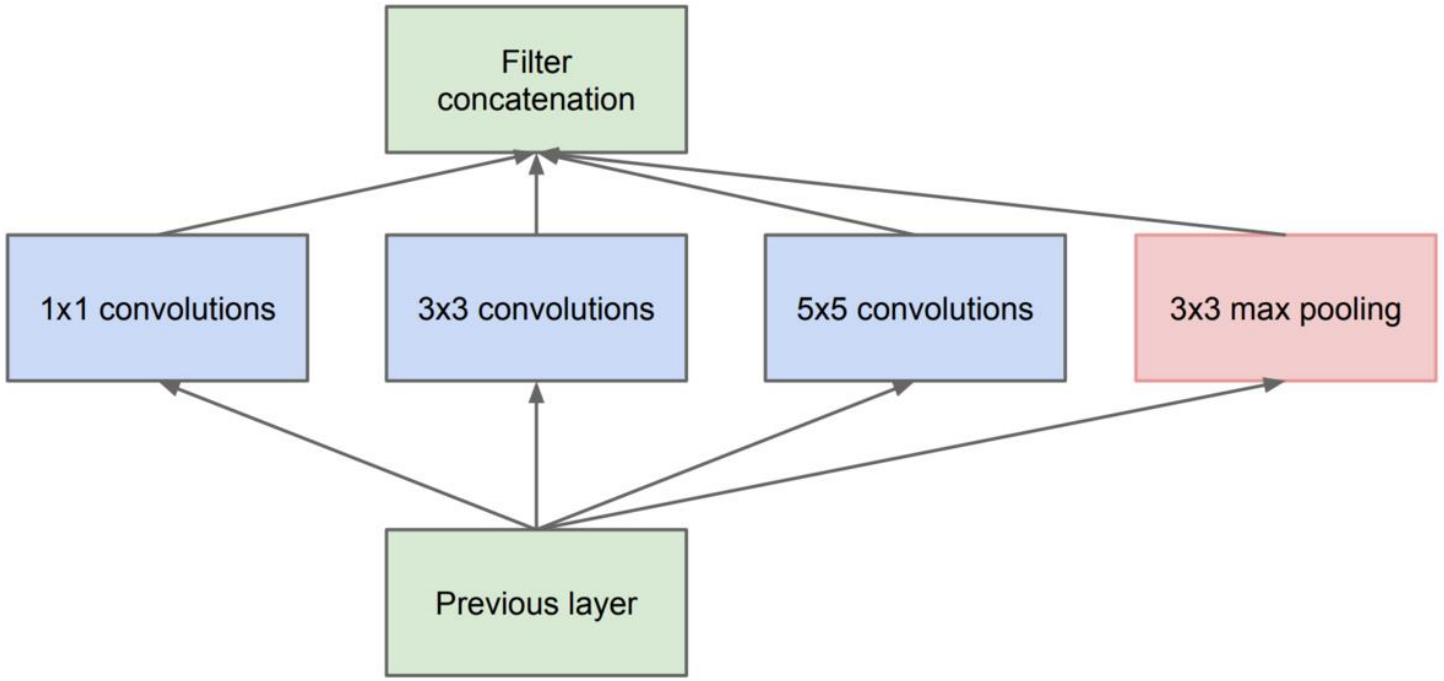
# Pooling層

- Pooling層はたいてい、Convolutoin層の後に適用される。入力データをより扱いやすい形に変形するために、情報を圧縮し、down samplingする。
- max poolingと呼ばれる手法では、操作は以下のように小領域に対して、最大のものを選択する操作となる。



# Inceptionの進化

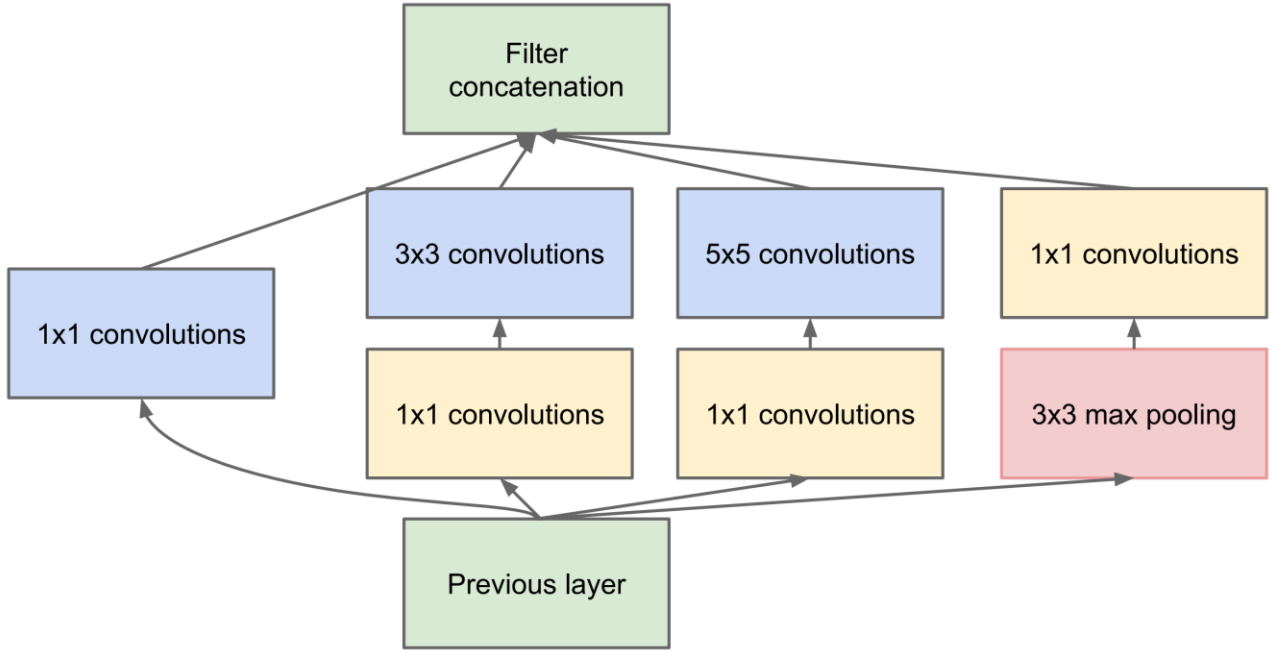
- Inceptionが最初に提案したバージョンは，入力データを多重サイズの畳み込み核で観察することを中心思想としている。




- そこで私達のネットワークは太ってきて、ネットの幅を増加して、同時に異なった尺度に対する適応度をも高めました。

# Pointwise Conv

- しかし、私たちのネットワークは太ってきた同時にパラメータ数も計算量も大きくなりました。だから、パラメータ数と計算量を減らすために、Inception v 1の最終バージョンに1 x 1カーネルの畳み込み加えました。



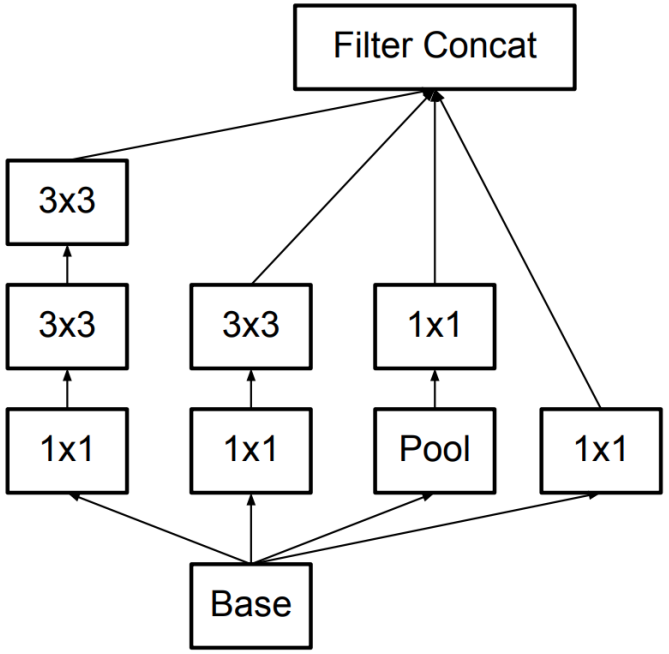
- 入力した特徴図を1 x 1カーネルを用いて減次元処理すると、パラメータ数と計算量が大幅に減少する。

| | Batch | | Height | | Width | | In Depth | | Out Depth |
|---|---|---|---|---|---|---|---|---|---|
| **Input** | 256 | × | 227 | × | 227 | × | 512 | | |
| **Kernel** | | | 1 | × | 1 | × | 512 | × | 32 |
| **Output** | 256 | × | 227 | × | 227 | | | × | 32 |

- これはPointwise Covolution（ＰＷ）です。通称は1 x 1畳み込みといいます。次元削減に使います。

# カーネルを置き換える

- たとえPWがあったとしても、５ｘ５と７ｘ７カーネルの直接畳み込み計算のパラメータ数は非常に大きいので、訓練時間はまだ長いです。方法の一つは大きなカーネルの代わりに、いくつかの小さなカーネルを使う方法が、Inception v 3です。



- 5x5カーネルの代わりに2つの3x3カーネルを使用しても、効果は同じですが、パラメーター数が大幅に削減され、最適化の目的が達成されます。パラメータ数が少ないだけでなく、層の数も増えることになる。
- 私たちは256次元を入力して512次元を出力すると仮定してパラメータ数を計算します。
- 5x5

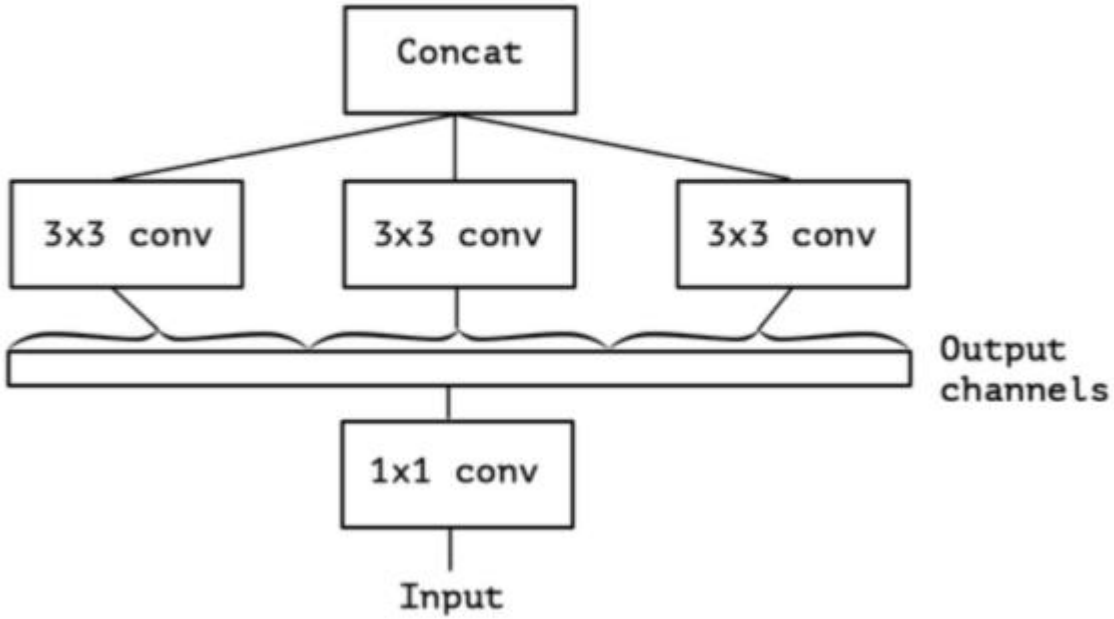
256*5*5*512=3276800

- 3x3
- 256*3*3*256+256*3*3*512=589824+1179648=1769472
- 対照　1769472/3276800＝0.54

# Depthwise Separable Conv

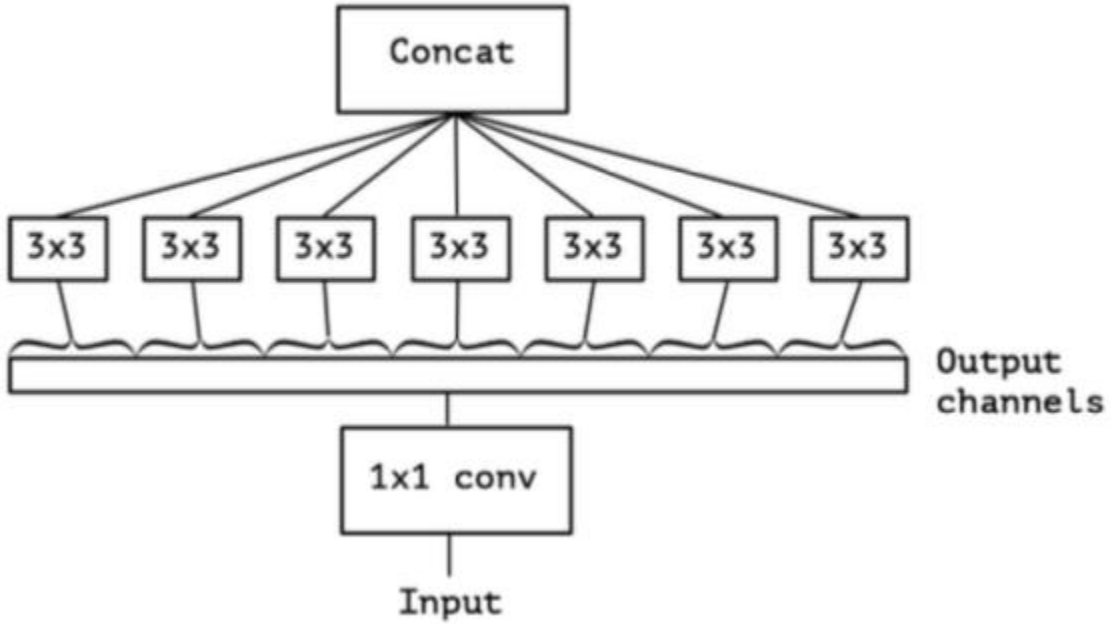
- これは、**1x1 convの出力チャンネルが3倍あって、それを3x3 convが分担して受け取っている**とも考えられる。

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



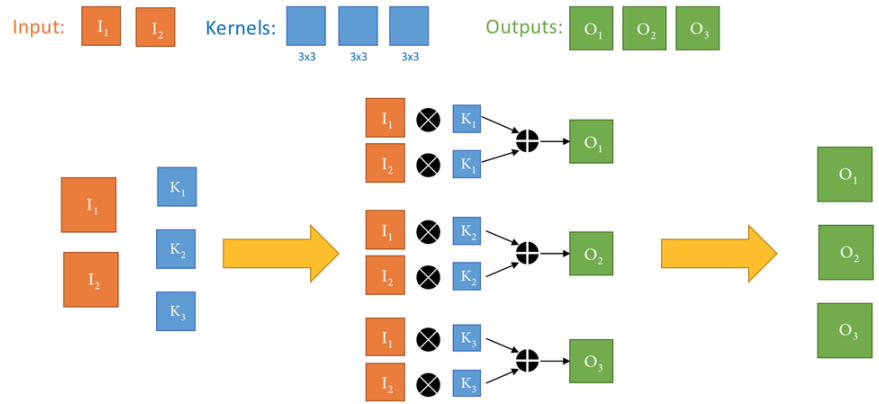
- 更に、これを極端(eXtreme)にするならば

Figure 4. An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.




こうなる。
3x3 convの数自体は増えるが、それぞれのconvに対するinput channelを少なくしている（というか1）ので計算量が激減する。
この時、もはや3x3 convは1チャンネルしか入力されていないので、チャンネル同士の関係性を見ることはない。
これをXception Moduleと呼ぶ。

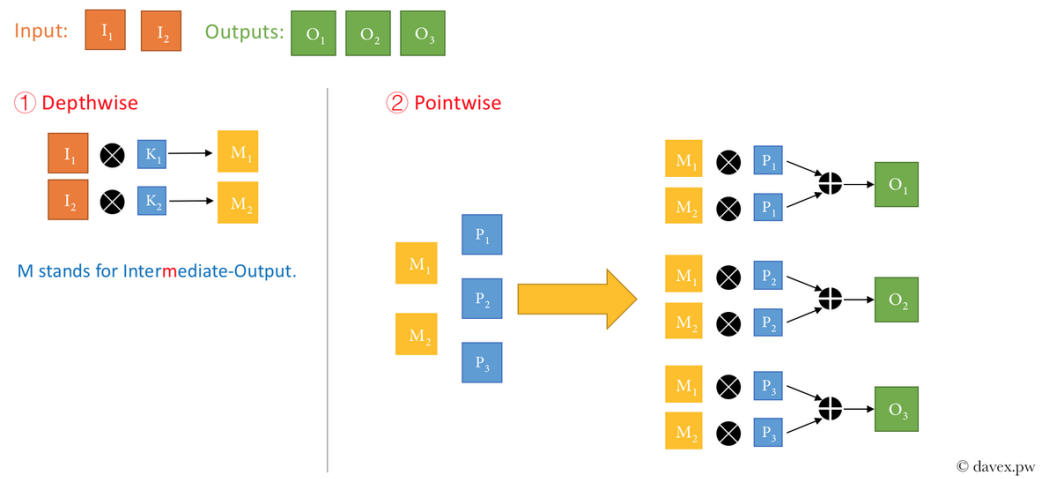
# Standard Convolution




Feature Maps: $2 * 3 = 6$

© davex.pw

# Depthwise Separable Convolution

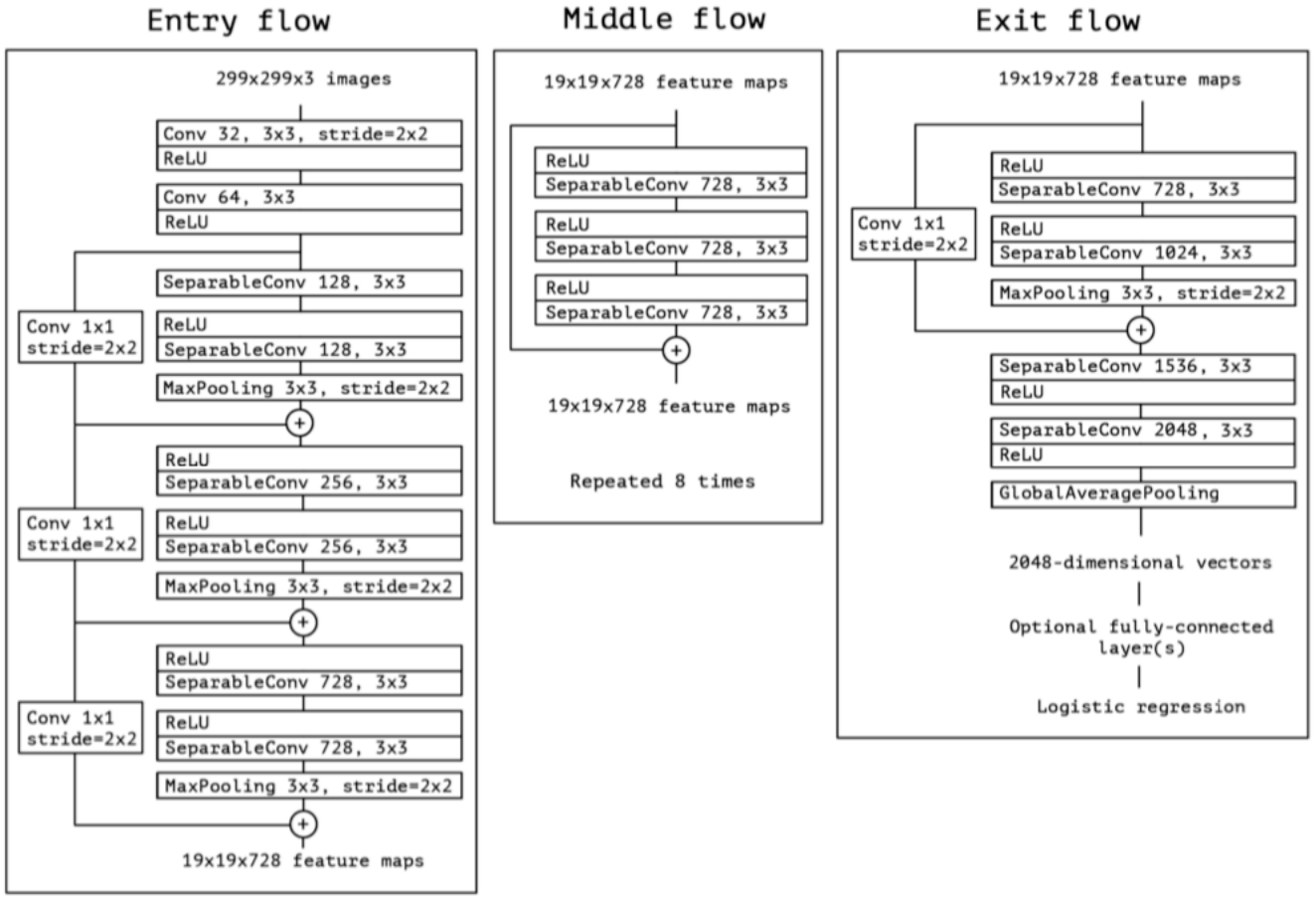



M stands for Intermediate-Output.

© davex.pw

# アーキテクチャ

Figure 5. The Xception architecture: the data first goes through the entry flow, then through the middle flow which is repeated eight times, and finally through the exit flow. Note that all Convolution and SeparableConvolution layers are followed by batch normalization [7] (not included in the diagram). All SeparableConvolution layers use a depth multiplier of 1 (no depth expansion).




入り口、中間、出口の３パートに別れており、中間部分は同じモジュールが８回繰り返されています。
入り口と出口部以外、residual(前の層の値)が追加されています。

# The JFT dataset

- The JFT dataset JFT is an internal Google dataset for large-scale image classification dataset, first introduced by Hinton et al. in which comprises over 350 million high-resolution images annotated with labels from a set of 17,000 classes. To evaluate the performance of a model trained on JFT, we use an auxiliary dataset,FastEval14k. FastEval14k is a dataset of 14,000 images with dense annotations from about 6,000 classes (36.5 labels per image on average). On this dataset we evaluate performance using Mean Average Precision for top 100 predictions (MAP@100), and we weight the contribution of each class to MAP@100 with a score estimating how common (and therefore important) the class is among social media images. This evaluation procedure is meant to capture performance on frequently occurring labels from social media, which is crucial for production models at Google.
- A different optimization configuration（最適化） was used for ImageNet and JFT

# 結果

## 精度

Image-NetとJFT dataset(JFTは大規模な画像分類用のデータセット)を使って実験しています。

Image-NetではInceptionV3に対してXceptionの方が少し精度が上回りJFTではXceptionが圧勝しています。

Table 1. Classification performance comparison on ImageNet (single crop, single model). VGG-16 and ResNet-152 numbers are only included as a reminder. The version of Inception V3 being benchmarked does not include the auxiliary tower.

| | 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** |

Table 2. Classification performance comparison on JFT (single crop, single model).

| | 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** |

# スピード

- スピードではInceptionに負けましたが、将来最適化が進むことでXceptionが上回る可能性に期待したいと論文は言っています。
- Inception V3とXceptionのパラメータ数はほぼ同じなので、モデルのパフォーマンス向上の要因は容量の増加というよりもモデルのパラメータの効率性によるものと考えられるといっています。

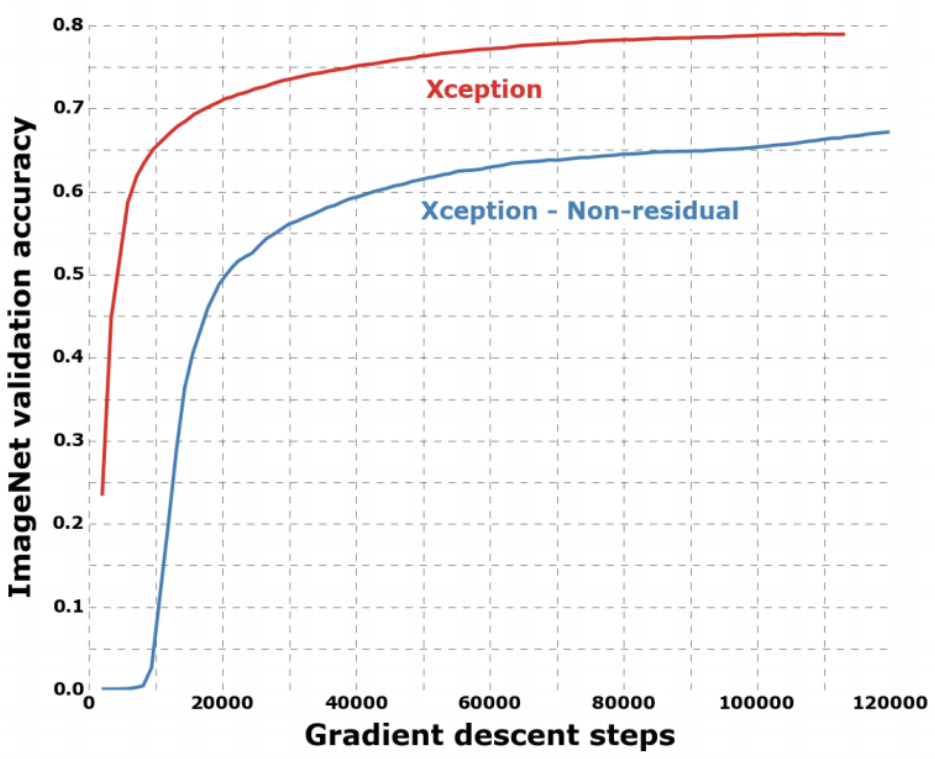
Table 3. Size and training speed comparison.

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

# residualを追加するか

- 下の図を見ればわかりますが、residualは追加した方が良さそうです。
  ただしNon-residualのほうはハイパーパラメーターの調整をしていないのがすこしハンデではある。

Figure 9. Training profile with and without residual connections.

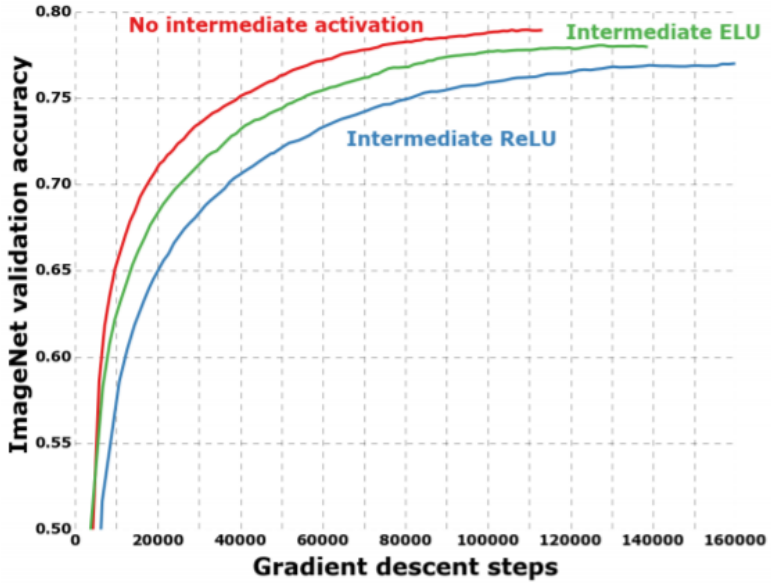


# 非線形関数を挟むか

- 1×1のカーネル（チャネル方向の畳み込み）と3×3のカーネル（空間的な畳み込み）の間に非線形関数は挟まないほうがよさそうです。
- 普通、深い特徴空間では非線形な変換は必要ですが、逆効果。

for deep feature spaces (e.g. those found in Inception modules) the non-linearity is helpful, but for shallow ones (e.g. the 1-channel deep feature spaces of depthwise separable convolutions) it becomes harmful, possibly due to a loss of information.

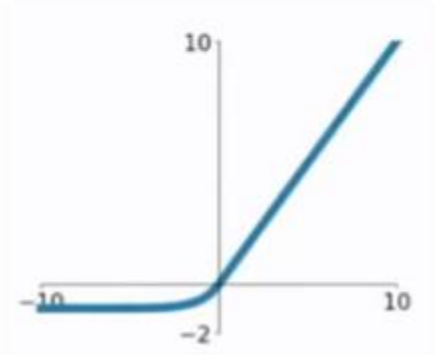
Figure 10. Training profile with different activations between the depthwise and pointwise operations of the separable convolution layers.



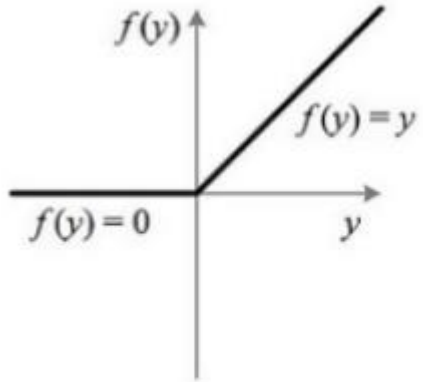
# 非線形関数

- ELU

$$f(x) = \begin{cases} x & , x > 0 \\ \alpha(e^x - 1) & , x \leq 0 \end{cases}$$




ReLU




$$ReLU(x) = \begin{cases} x & if x > 0 \\ 0 & if x \leq 0 \end{cases}$$

# 最後

- XceptionとInception V3ではほぼ同じパラメーター数で学習しているのに、精度がよくなったということは、より効率よくパラメーターを使えているということ
- Xceptionモデルは「チャネル方向の畳み込み」と「空間方向の畳み込み」を完全に分離したものととらえることができる