深層学習における
二次元畳み込みフィルタの生成に関する研究

2024 年度

福　崎　翔　太

学位論文　博士（工学）

深層学習における
二次元畳み込みフィルタの生成に関する研究

2024 年度

慶應義塾大学大学院理工学研究科

福崎翔太

目次

図目次

表目次

第 1 章　序論

本章では，本研究の背景ならびに目的について述べる．本研究は画像処理の深層学習アプローチにおける畳み込みの訓練方法の改善に関するものである．深層学習を用いた画像処理における畳み込み自体の位置づけや訓練方法の改善方法に関する研究としての位置づけ，畳み込みに関する研究として見た場合の位置づけが本研究の背景となる．これらを述べたのち，本研究が果たすべき役割，すなわち目的を明示する．

1.1　研究背景

1.1.1　デジタル画像処理

本研究はデジタル画像処理（以降は単に画像処理と呼ぶ）に関するものである．ただし，ここでいうデジタル画像（以降は単に画像と呼ぶ）は主にスマートフォンやデジタルカメラ等で撮影されたような写真を指すものとする．コンピュータ上では数値の羅列として表現される画像をコンピュータを用いた計算処理によって分析・加工することを本論文では画像処理と呼ぶ．

画像処理は産業的にも文化的にも応用される．たとえば自動運転の実現方法の 1 つとして車載カメラから道路の状態を得るということが考えられる [3]．このとき，カメラで撮影された映像を自動車の挙動に反映させるためにはこれを自動的に分析できなければならない．ここで得られる映像は画像の列と見なすことができ，画像処理は映像処理の基本的な制約付き問題であると言える．ほかにも防犯カメラ等の設置されたカメラによる視覚的な異常の検知 [4] や医療用の診断画像を分析して病気の見落としを減らす応用 [5] などがある．また，写真や動画はインターネット上のコミュニケーションを促進するメディアとして盛んに利用されており，また，Instagram[1)] や TikTok[2)] 等の SNS（Social Networking Service；ユーザ同士の交流を目的とした Web サービス）においては撮影されたそのままの状態ではなく，見栄えを変える，一部をモザイク等で隠すと言った加工がなされる．画像処理はそういった場面にも応用される身近な技術である．

1.1.2　Token Mixer の一種として見た畳み込み

近年の画像処理では深層学習を用いたアプローチが主流である [4, 6–14]．2010 年代後半の深層学習モデルは畳み込み層を主とした畳み込みニューラルネットワーク [15]（Convolutional Neural Network；CNN）が主流であったが，Vision Transformer [16]（ViT）の登場を機に，画像内の複数の位置から特徴を抽出する方法として畳み込み以外の計算が検討されるようになり，もはや CNN

1） https://www.instagram.com/
2） https://www.tiktok.com/

とは呼べないものとなった．画像における複数の位置の情報を集約する計算を MetaFormer [17] の文脈においては Token Mixer と呼び，本論文においても畳み込みを Token Mixer の 1 種として取り扱う．

MetaFormer [17] は ViT を抽象化した画像処理モデルの雛型であり，入力する特徴マップ（あるいはトークン列）x を位置横断的に処理する残差ブロックと，入力する特徴マップ（あるいはトークン列）x を位置独立的に処理する残差ブロックを交互に繰り返す特徴抽出方法を指す． [17] においては，3×3 の平均値プーリングから入力特徴マップの値を差し引くだけのシンプルな Token Mixer による MetaFormer（PoolFormer）が高い精度を発揮することを示し，ViT が高精度な画像分類を実現する理由として，ViT の Token Mixer である Multi-Head Self-Attention [18]（MHSA）よりも MetaFormer の構造による寄与が大きいと主張した．

畳み込みは Token Mixer として唯一の選択肢ではなくなり，CNN 全盛期ほどは用いられなくなった．しかしながら，他の Token Mixer にはない特徴を持っており，これからの画像処理モデル設計は畳み込みの特徴を活かしたものとなることが予想される．つまり，畳み込みは畳み込みが得意なことのみを担当すればよく，畳み込みを用いてうまくいかない部分は他の Token Mixer で置き換えればよい．以下では，畳み込みが持つ Token Mixer としての特徴を相対的に記述するため，代表的な Token Mixer である MHSA，Multi-Layer Perceptron [19]（MLP），空間的シフト [20] と比較する．

MHSA と比較した場合，畳み込みは静的な重みをもつと言える．つまり，畳み込みの重みは MHSA における重みと違って入力サンプルや位置に依存しないものである．畳み込みの計算はあらゆる画像・位置において等しく同じ重み付け和を計算するという，MHSA よりも強い制約を持った計算であるということができる．畳み込みにおけるパラメータの最適化はこの強い制約のもと行われるため，MHSA よりも必要なサンプル数が少なくなる．

例えば，約 130 万枚の画像からなるクラス分類データセットである ILSVRC2012 データセット [21]（ImageNet Large Scale Visual Recognition Challenge 2012）について，乱数で初期化した ViT を訓練した場合の精度（正解率）は最高でも 77.9%（ViT-B/16；Vision Transformer の Base モデルで，画像を 16×16 個のパッチに分けトークン化したモデル）である．これは 2016 年の CNN である ResNet [22] の最軽量モデル ResNet-34（78.2%）を下回る水準であり，ViT は約 2,100 万枚の画像からなる ImageNet-21k [23] や約 3 億枚の画像からなる JFT-300M[3] [24] を事前学習に用いることで初めて当時の最高水準を達成したことを示している．つまり，こういった大規模なデータセットを用いることができない計算環境において精度の良いモデルを得るには畳み込みのほうが向いていると言える．

MHSA を 7×7 の小領域で行う Swin Transformer [25] では ILSVRC2012 のみを用いた訓練結果が改善され，当時の最高水準であった CNN である EfficientNet [26] と遜色ない結果を実現しており，こういった制約を設けることで画像処理モデルの学習に必要なサンプル数を削減している．また，DeiT [27] においては RegNetY [28] という CNN の中間特徴マップと互換性のある中間特徴トーク

3） JFT が何を意味するかは不明である．

ンを用いるよう ViT の学習を矯正する等の工夫により ILSVRC2012 のみを用いた学習による結果を改善した（77.9%→ 84.5%）．こうした，近傍の値を用いる特徴抽出を行うという制約も少数データによる学習に貢献すると言うことができる．相対位置によって重みが定まる畳み込みはその典型であり，大規模なデータによる学習が不可能な場合において精度の良いモデルの獲得に貢献する．

MLP と比較した場合，畳み込みはすべての位置において同じ相対的重みを用いる制約を持った特殊な線形計算であると見なすことができる．

MLP の学習も MHSA と同様，畳み込みと同程度以上の精度のモデルを獲得するには大規模なデータセットによる事前学習を必要とする．例えば，MLP のみを Token Mixer として用いた MLP-Mixer を ILSVRC2012 データセットのみを用いて訓練した場合，精度が 76.4%（MLP-B/16；Vision Transformer の Base モデルで，画像を 16×16 個のパッチに分けトークン化したモデル）程度にとどまり，ImageNet-21k や JFT-300M データセットを用いることで 83.9%（MLP-L/16）や 86.8%（MLP-L/16）といった ViT をやや下回る程度の精度を達成している．

一方，畳み込みフィルタのサイズを縦横それぞれ最大で画像サイズの 2 倍程度まで拡大した RepLKNet[4] [29] は ILSVRC2012 データセットのみで 84.8%の精度（RepLKNet-31B）を達成している．最大の受容野については MLP と同じ大きさを持つことから，MLP を少ないデータで学習した場合に精度が悪化する原因として受容野の大きさよりも，重み付けの方法が各位置で共有されていないことが挙げられる．つまり，各位置で同じフィルタ重みを利用して処理していることが少ないデータで訓練した画像処理モデルの精度に貢献していると言える．

学習とは有限の事例から求められる入出力の関係の一般化を行うことを指し，推論時には獲得した一般法則を未知の入力に対して適用することで所望の出力を得る．この一般化に際しては計算上の制約からバイアスがかかり，そのバイアスのもとで最適化がなされる．このバイアスを帰納バイアス（inductive bias）と呼ぶ．帰納バイアスが大きいほど計算の選択肢が減るため，十分な自由度を持たない場合は十分多い訓練データのもと精度低下の原因となりうるが，少ないサンプルによる一般化を容易にする．

自由度の制限が過度な例としては空間的シフトが挙げられ，これは 1 箇所だけ値が 1 であり，その他が 0 であるような平行移動フィルタを畳み込むことと等価である．実際，上下左右 1 要素分の平行移動しか行わない S^2-MLP[5] [30] においては，ILSVRC2012 のみを用いて訓練した場合の精度が 80.7%程度（S^2-MLP-deep；36 ブロック）にとどまり，2017 年の PyramidNet-200 [31] の精度 80.8%（$\alpha = 450$）と同程度の水準である．また，同じく MetaFormer の TokenMixer として畳み込みを適用した ConvFormer [32] は同じ入力画像サイズ（224×224）とブロック数（36）のもと，最低でも 84.1%（ConvFormer-S36）を達成しており，空間的シフトのみを TokenMixer として用いるのは少ないデータで学習する場合においても畳み込みより精度面で不利である．

以上の比較から，畳み込みは少ないデータによる訓練を想定した際に高精度なモデルを学習するための TokenMixer として適していると言える．ここまで少ないデータとしてきた ILSVRC2012

4） Rep は reparameterization（パラメータの再配置），LK は large kernel（フィルタの受容野が大きいこと）を意味する．

5） S^2 は spatial-shift を意味する．

データセットも約 130 万枚の画像を含んでおり，名前に「Large Scale」を含んでいるように，これより多くの訓練用データを準備できる状況は限られる．つまり，いずれかの TokenMixer をひとつ選択して用いなければならない場合，GAFAM[6)] に比肩するほどの大規模な計算環境や訓練データセットに恵まれていないならば，依然として畳み込みが有力な選択肢となる．

また，TokenMixer の組み合わせとして，入力に近い部分の処理では畳み込み，出力に近い部分の処理では MHSA を用いると，少ないデータでもいずれか一方のみを用いたモデルより高精度になることが知られている [32, 33]．このように，今後の画像処理は TokenMixer の特性に応じて適切なものを部分ごとに選択・適用することが求められ，畳み込みは特に入力画像，つまり，原信号に近い特徴マップに適した TokenMixer として重要な役割を果たすと考えられる．

1.1.3 最適化とモデルの潜在能力

本研究は畳み込みフィルタの最適化，つまり，より高精度な処理を実現する畳み込みフィルタを獲得するための方法に関する研究である．裏を返せば，基本的に深層学習モデルを真の意味で最適化することはできず，深層学習モデルを訓練した結果得られたパラメータよりも推論精度を高くすることができるパラメータ組み合わせが潜在的に存在する [34]．

たとえば，先の例においても ILSVRC2012 データセットのみで学習した場合と比べ，事前学習に JFT-300M や ImageNet-21k を用いた場合のほうが精度が向上しており，ILSVRC2012 のみでは深層学習モデルの最適なパラメータを見つけることができていないことを意味する．本研究では，訓練データが少数であるような状況や，大規模なデータ処理が難しいハードウェア環境など，比較的小規模な訓練環境に焦点を当て，画像処理モデル（特徴マップに作用する計算内容）の変更を最小限にとどめながら訓練結果の改善を試みる．

訓練方法の変更に伴う精度の改善には汎化（過学習の抑制）能力だけではなく，訓練の高速化（収束時間の短縮）も評価の対象となる．非現実的な時間をかけて最終的に精度が最高となるような方法であっても，確保できる時間的なコストの範囲では訓練後のモデルの精度が求める水準に達しないということがありえるためである．ただし，これらを区別するためには十分な時間的コストをかけた訓練結果を比較する，あるいは，数学的に証明する必要があり，訓練方法の改善に関する各提案論文における報告内容からは判断できない場合も少なくない．したがって，汎化能力の乏しい方法であっても訓練の高速化は実用上の価値を持つ．

深層学習では同じモデルを訓練する場合でも，初期値，パラメータ更新方法，訓練データ，損失関数等を変更することによって訓練結果が変化する．以下に代表的な研究例を挙げる．

パラメータの初期値を決める初期化の方法には，乱数や定数による初期化と事前学習したパラメータの転用の大きく 2 種類が存在する．なお，事前学習も元をたどれば乱数で初期化されている場合が多い．

事前学習したパラメータを初期値として用いる方法には転移学習と fine-tuning があり，後者は訓練したいタスク（downstream task）に合わせてパラメータを微調整する．また，事前学習に用いる

6) 世界的な IT 企業の代表格である Google，Amazon，Facebook（現・Meta），Apple，Microsoft の頭文字をとった総称．

タスクに関して，Variational AutoEncoder（VAE）や画像分類タスクが広く用いられていたが，近年では新たに Contrastive Language-Image Pre-training [35]（CLIP）や Masked AutoEncoder [36,37] が知られるようになった．

また，「知識の蒸留」 [38] と呼ばれる方法では，教師モデルと生徒モデルを用意し，学習済みの教師モデルを補助的に用いて生徒モデルの学習をおこなう．ここで，補助的に用いるとは，教師モデルの中間出力特徴マップを生徒モデルが模倣することによっても損失が小さくなるようにすることをいう．教師モデルを縮小（チャンネル数や繰り返し処理の回数を減らすなど）することによって生徒モデルを構成した場合，生徒モデルの訓練は教師モデルの補助がある場合のほうが高精度となる．

深層学習モデルはほとんどの場合，誤差逆伝播法 [39]（error back-propagation method）をベースとした勾配降下法で訓練する．パラメータ更新のための漸化式を変更することで，訓練時間や精度が改善する．パラメータ空間上の移動に対して慣性項を導入した Momentum 系の手法 [40, 41] や振動を抑制するために移動量を少なくしていく AdaGrad 系の手法 [42, 43]，そしてそれらを組み合わせた Adam 系の手法 [44–46] が知られる．また近年では，「パラメータ空間の極小点では周囲の勾配が小さいほどサンプルごとの精度のばらつきが小さい，汎化された状態である」という仮説の下，勾配上昇ステップと勾配降下ステップを交互に繰り返した Sharpness-Aware Minimization（SAM） [47] も組み合わせて使用される．

深層学習モデルの訓練には大量の画像からなるデータセットを使用するが，深層学習モデルはパラメータ数が膨大なため柔軟性が高く，データセット内の画像にのみ最適化された過学習を引き起こしやすい．そこで，乱数基づく方法で訓練用の画像を加工してから入力することで過学習を抑制する手法（Data Augmentation）が存在する．加工方法はタスクに応じて適切と考えられるものが提案・適用され，クラス分類タスクにおいては 2 つの画像をランダムな比率で重ね合わせる MixUp [48] や，画像の一部を切り抜いて別の画像に埋め込む CutMix [49]，いくつかの加工方法をランダムに選出・適用する RandAugment [50] などがある．

深層学習モデルは出力と期待される正解との距離を損失関数とするのが基本であるが，入力次第では比較対象を正解とするだけの必然性に欠ける場合がある．この場合，複数存在する正解候補を平均化した出力をすることで損失関数の最小化を図るが，このようにして学習されたモデルの出力結果はどっちつかずの中途半端なものとなる．そこで期待する正解に近いこと以外の性質を満たすほど値が小さくなる項を損失関数に加える手法が存在する．たとえば，画像を出力するタスクにおいては Generative Adversarial Network（GAN） [51] の判別器を使用した損失や VGG-16 の中間出力から計算する Perceptual Loss [52] などが加えられる．その他，クラス分類タスクにおいては正解ラベルを一部平滑化した Label Smoothing [53] が使用される．

これらの研究はそれぞれ異なる観点で訓練の改善を試みるものであり，その多くは組み合わせて利用することが可能である．本研究の提案手法は畳み込みフィルタの生成過程に焦点当てたものであるため，上記の手法と併用することが可能であり，画像処理モデルの訓練結果を改善することを目的とした研究だが，比較対象とはならない．

1.1.4 畳み込みに関する研究

1.1.4.1 深層学習における畳み込み

ネオコグニトロン [54] は CNN の原型とされており，画像中の特定のパターン対して大きい値を出力するようなフィルタを畳み込むことで画像認識における位置のずれやゆがみに対する頑健性を実現した．LeNet [55] ではネオコグニトロンの訓練方法に誤差逆伝播法 [39] が採用された．誤差逆伝播法によってフィルタ重みを獲得するという今日の標準的な畳み込みフィルタ訓練法はこの時点ですでに確立された．

深層学習が画像処理分野で本格的に用いられるようになり始めたのは ILSVRC2012 で AlexNet [56] が優勝したことがきっかけとされている．以降 CNN の深層学習は画像処理分野の標準的なアプローチとなり，2020 年ごろには高度な画像処理において必須のものとなった．一方，研究の中心となったのは Residual Block [22] や Channel Attention [57]，U-Net [58] など深層学習モデルの構造に関するものであり，depthwise 畳み込みや Grouped Convolution [59] の登場などはあったものの，畳み込みは深層学習モデルにおいてすでに完成した構成要素のひとつとして扱われた．

1.1.4.2 深層学習における畳み込みフィルタの設計

原則，畳み込みフィルタはパラメータの配列として表現される．例外として，パラメータを櫛型フィルタの非ゼロ部分の値として使用する Dilated Convolution [60] が広く用いられる．他には，パラメータを対称的に配置するものやフィルタ成分系の線形結合係数として使用する手法がある．

パラメータを対称的に配置するものとしては Symmetric Kernels [61]（SymKer），N-way Parameterization [62]，SymNet [63] がある．それぞれが異なった対称性をもったフィルタを提案している．1 つのパラメータが複数箇所で使用されることにより，パラメータ数を減らすことができ，たとえば水平方向に対称にパラメータを作用させる SymKer ではパラメータ数が約半分になる．ただし，畳み込みフィルタの要素数が増えるほど比例してパラメータ数が増加する．

いくつかの決まったパターンのフィルタ成分をあらかじめ定めておき，それらの線形結合係数としてパラメータを利用・学習する方法が Structured Receptive Field（SRF） [64] である．SRF の成分は関数をベースに設計されるため，フィルタサイズに依存せずフィルタ成分を定義することができる．このため，フィルタ成分系を構成する関数の系を定めれば，畳み込みフィルタの要素数が増えてもパラメータ数を一定に保つことができる．

SRF はフィルタサイズの大きい畳み込みにおけるパラメータ数の削減に有利な特性を持つが，成分系としては [64] で提案された N-Jet [1] 成分系以外知られていない．第 3 章の実験で示す通り，N-Jet 成分系は画像処理モデルによっては勾配爆発を引き起こし，うまく訓練することができない．このように，SRF はフィルタサイズに依存しない優れたパラメータ運用の手法であるが，成分系の開拓・検討は不十分であると考えられるため，本論文では畳み込みフィルタの主成分を近似モデル化した新たな成分系を提案する．

1.1.4.3　フィルタサイズの大きい畳み込みフィルタの利用

2010 年代の深層学習モデル（畳み込みニューラルネットワーク）では 3×3 畳み込みが中心に使用された．VGG-16[7)]が提案された論文 [2] では 3×3 畳み込みを繰り返すことにより，広い範囲の情報を集約することができるという主張がなされている．また，Dilated Convolution [60] では櫛型のフィルタを使用することで，まばらにではあるが，離れた位置の特徴量の参照が可能となった．以降，7×1 と 1×7 のフィルタを組み合わせて用いる Inception-v4 [65] や 15×1 と 1×15 のフィルタを用いる Gloval Convolutional Network [66]）が提案されたが，3×3 のフィルタが継続的に使用された．

Vision Transformer [16] や MLP-Mixer [19] といった新しい Token Mixer 登場を機に，モデル全体ではなく，モジュール単位で広範囲から特徴抽出をする方法が検討され始めた．Swin Transformer [25] の影響を受け，7×7 の畳み込みが採用した ConvNeXt [67] や，フィルタサイズを最大 31×31 まで拡大した RepLKNet [29] が登場し，ベースラインと比較して推論精度が向上すること，ならびに，実効受容野が実際に大きくなることを示した．その他，51×5，5×51，5×5 のフィルタを並列に使用するスパースな畳み込みが SLaK [68] のモジュールとして提案された．

フィルタサイズの大きい畳み込みは Transformer ベースのモデルでも検討されている．CrossFormer [69] では，入力画像に対する最初の処理であるパッチ化に際して最大 32×32 の畳み込みを採用した．また，各トークンの位置情報を表す Positional Encoding を特徴マップに対する畳み込みによって計算する Conditional Position Encoding [70]（CPE）では 27×27 の畳み込みフィルタを用いることで推論精度が向上することが報告されている．

以上のように，フィルタサイズの大きい畳み込みの利用が検討されている．しかしながら，フィルタサイズが大きくなるほど，パラメータ数および計算コストが増大するという問題がある．本論文はフィルタサイズの大きい畳み込みを使用する際においても，パラメータ数や訓練時の計算コストを削減するための方法を提案する．

1.1.4.4　本研究の対象に含まれないもの

最後に，畳み込みと銘打たれているものの，本研究では取り扱わない研究例を挙げる．取り扱わない理由はいずれも興味の対象としている畳み込みと異なる自由度を有しているためである．

1×1 畳み込みはフィルタサイズが 1×1 の畳み込みであり，Pointwise Convolution とも呼ばれる．この計算は特徴マップ上の他の位置の特徴を参照しないため，Token Mixer ではない．本研究は Token Mixer としての畳み込みを対象としたものであるため，画像処理モデル中に 1×1 畳み込みが含まれることはあるが，畳み込みに関する本論文の提案事項は適用されない．

特徴マップの値を用いてフィルタ値を生成する方法もある [71]．この方法は特徴マップに応じて適当なフィルタを臨機応変に生成するものだと捉えることもできるが，ある特徴マップに対して別のフィルタを畳み込んだのではチャンネルと情報の意味との対応をとることが難しいため，「行き当たりばったり」な処理による出力結果を期待することになる．したがって，本研究では「特徴

7）　これは Visual Geometry Group というチーム名の略称であるがモデル名として知られるようになった．

マップは各チャンネルに何らかの情報を持っており，畳み込みはその特徴を決まった重みによって処理することで別の情報を持った特徴マップへと変換する処理」であるべきだという立場をとる．この立場から，本研究では特徴マップの値に基づくフィルタの動的生成は取り扱わないが，本研究は畳み込みフィルタを生成するにあたり成分系ならびにサイズ変換に関する知見を提供するものであるため，特徴マップからフィルタを動的生成する際にも本研究の提案手法が応用できる可能性がある．

Sparse Convolution [72–74] では特徴マップの位置ごとに active/non-active を定めるものであり，畳み込みベースの Masked AutoEncoder の訓練に用いられる [37]．active の位置では周辺の active の位置のみフィルタ重みに基づく重み付き和をとり，non-active の位置は不変とする．これは，active の位置においては non-active の特徴を 0 としたうえで共通のフィルタ重み（学習可能）を用いるが，non-active の位置では Identity フィルタ（学習不可能）を畳み込むことを意味する．つまり，Sparse Convolution は位置ごとに異なる重みを用いる計算であり，厳密には畳み込みではない．したがって，本研究では取り扱わないが，計算過程の一部に non-active な位置の特徴を 0 とした特徴マップへの畳み込みが含まれるため，本研究で提案する手法を適用することができる．

Deformable Convolution [75, 76] は特徴マップ上の各位置で参照するべき相対位置を計算する．つまり，特定の重みに対してふさわしい相対位置を探し出すため，重みと相対位置が紐づいた畳み込みで表現することはできない．このため，本研究では Deformable Convolution を畳み込みとして扱わない．なお，Deformable Convolution は Token Mixer としてほぼ唯一畳み込みが用いられる時代に提案されたものであるため，重みにふさわしい相対位置の探索には畳み込みが実装として用いられている．このため，探索に用いる Token Mixer には検討の余地があるが，本研究による畳み込みの訓練方法の変更は Deformable Convolution にも適用できる．

1.1.5 研究背景に関するまとめ

本研究は深層学習モデルの代表的な Token Mixer である畳み込みの訓練方法に関するものである．畳み込みは強い制約を持った Token Mixer であり，訓練データが少ない場合に適している．深層学習モデルは同じ構造であっても初期値，パラメータ更新方法，訓練データ，損失関数等により得られるモデルの精度や訓練速度が変化する．本論文では畳み込みフィルタの生成過程に関する提案をすることでモデルの訓練速度を改善する．本研究の提案とその他の訓練手法は変更対象が異なるため，組み合わせて適用することが可能である．

深層学習における畳み込みは 3×3 の畳み込みを繰り返すことで広範囲の情報を段階的に集約するという考え方で使用されてきた．しかしながら，近年では広範囲の情報を集約するためにはフィルタサイズの大きい畳み込みが有効であることがわかってきた．フィルタサイズの大きい畳み込みはパラメータ数や計算コスト（FLOPs）が増大するため，畳み込みが強みを発揮する小規模な環境向きではない．このような背景の下，本研究ではパラメータ数がフィルタサイズに依存しない SRF の新しいフィルタ成分系やフィルタサイズを縮小した訓練を提案し，パラメータ数や訓練時間を削減する．

1.2 研究目的

近年の画像処理モデルにおいては畳み込み以外にも MHSA や MLP といった Token Mixer が広く利用されつつある．しかしながら，これらは畳み込みに比べて計算上の制約が少ないため，推論精度の高いモデルを獲得するには多くの訓練サンプルを必要とする．したがって，小規模な訓練環境においては依然として畳み込みを主としたモデルが有効である．

畳み込みは他の Token Mixer の影響を受け，7×7 や 31×31 といったフィルタサイズの大きい畳み込みが注目されている．これは推論精度を高める一方で，パラメータ数や計算コストを増加させ，小規模な計算環境での訓練には不向きである．

そこで，本論文では畳み込みフィルタのサイズが大きくなった場合においてもパラメータ数や計算コストの増加を抑えるための方法を提案する．これによって，小規模な訓練環境に適した畳み込みに対し，小規模な訓練環境に適した訓練方法の実現を目指す．

具体的には畳み込みフィルタを単純なパラメータの配列として表現するのではなく，パラメータから畳み込みフィルタを算出するアプローチをとる．以下，本論文ではこれを畳み込みフィルタの生成と呼ぶ．生成過程が微分可能であれば，正確には任意の入力に対する勾配が定義されていれば，誤差逆伝播法によりパラメータを調整することが可能であり，提案手法ではこの原理によって畳み込みフィルタを間接的に訓練する．畳み込みフィルタの生成および学習原理については第 2 章にて述べる．

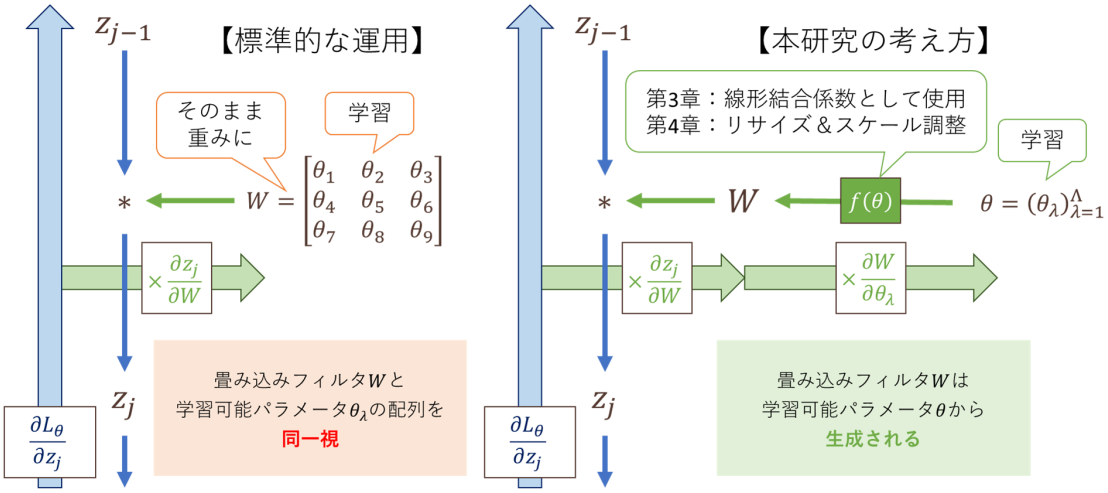


図 1.1: 本研究で扱う，畳み込みフィルタの生成を含む訓練フレームワーク（第 2 章）

本研究における畳み込みフィルタの生成および訓練フレームワークを図示すると図 1.1 のようになる．標準的な運用では畳み込みフィルタがパラメータの配列として表現されるが，本研究ではパ

ラメータから畳み込みフィルタを生成する．生成したフィルタを特徴マップに畳み込み，モデルの最終的な出力から逆伝播される損失関数の勾配を生成関数についても逆伝播することでパラメータを最適化する．パラメータが最適化されることにより，生成される畳み込みフィルタも最適化されることとなる．具体的な生成方法は第 3 章と第 4 章で異なっており，それぞれパラメータ数削減，訓練時間短縮を目的としている．

畳み込みフィルタをパラメータの配列として定義・使用すると，フィルタの要素数と同数のパラメータが必要になる．そこで，第 3 章ではパラメータ数がフィルタサイズに依存しない Structured Receptive Field（SRF）の成分系を検討する．SRF では定めた成分の個数分だけパラメータを定義し，パラメータで重み付けした線形結合の形式でフィルタを表現する．畳み込みフィルタの主成分を近似モデル化した新たな成分系を提案し，既存の N-Jet 成分系よりも多くのモデルに適用可能であること，より少ない成分で畳み込みフィルタを訓練可能であることを示す．

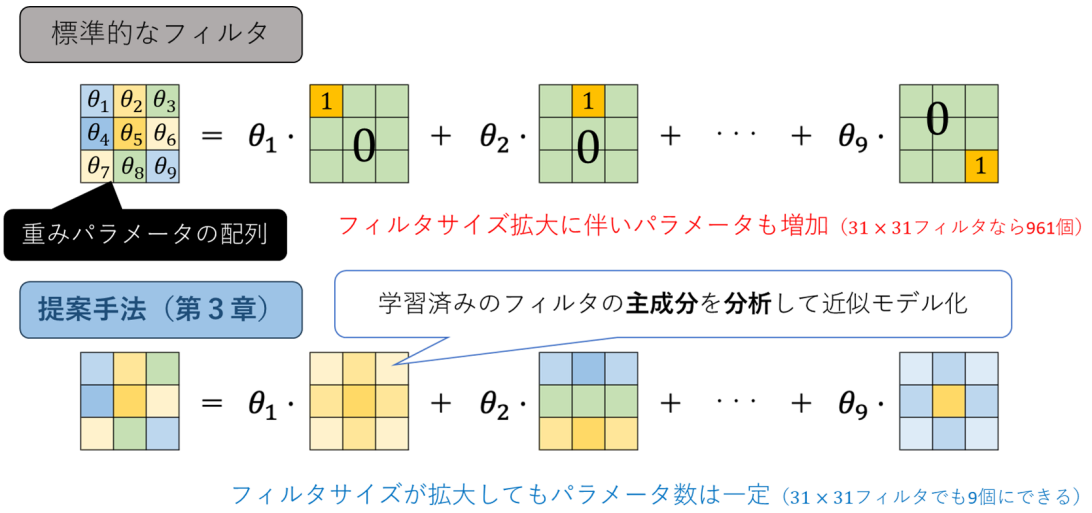


図 1.2: フィルタ生成例①：パラメータを係数としたフィルタ成分の線形結合（第 3 章）

第 3 章におけるフィルタ生成を図示したものが図 1.2 である．パラメータを係数としたフィルタ成分の線形結合で畳み込みフィルタが生成されると考えた場合，標準的なフィルタは暗に標準基底を仮定していることとなる．1 要素だけが 1，それ以外は 0 という局所的な成分で構成されており，フィルタの要素数と同数の係数（パラメータ）が必要である．一方，SRF では全体的に重みを持ったフィルタ成分を仮定するため，必要なフィルタを表現できる成分がそろっていればパラメータの数はフィルタの要素数と同じでなくても良い．提案手法では SRF の新しい成分系として畳み込みフィルタの主成分を近似モデル化し，従来の N-Jet 成分系よりも少数の成分でフィルタを表現することを試みる．

フィルタサイズが大きくなると畳み込みの計算量が増加する一方，畳み込みフィルタには縮小する余地が生まれる．そこで，フィルタサイズの大きい畳み込みを訓練する際にフィルタおよび訓練画像を縮小した状態で使用する方法を提案する．畳み込みフィルタと訓練画像の両方の要素数を減

らすことで訓練時の計算量が削減される．第 4 章では，縮小された状態での訓練が元のサイズにおける訓練の代替になるための条件，訓練後のモデルの推論精度を高めるための訓練方法を議論し，大きい畳み込みを持った深層学習モデルの訓練時間が短縮されることを示す．

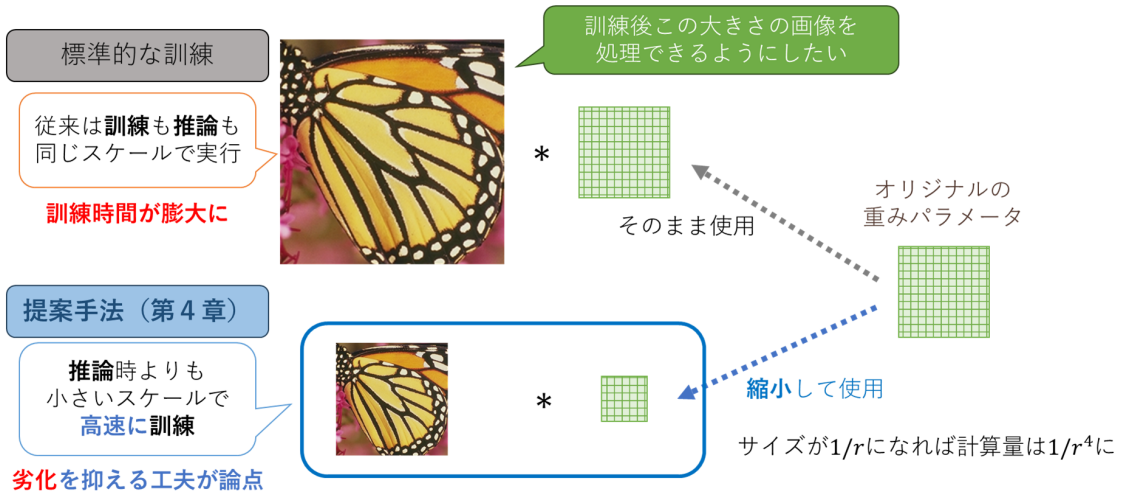


図 1.3: フィルタ生成例②：パラメータ配列のリサイズ，スケール調整（第 4 章）

第 4 章におけるフィルタ生成を図示したものが図 1.3 である．標準的な訓練方法では訓練後に処理したい大きさの画像に対し，パラメータの配列をそのまま畳み込む．この方法ではフィルタサイズが大きくなるほど訓練時の計算コストが大きくなる．一方，提案手法では訓練画像を小さくし，それに合わせてフィルタのサイズと値のスケールを調整する．元のスケールの訓練を小さいスケールの訓練で代替することにより，訓練時の計算コストを削減し，訓練時間を短縮する．

本研究は以上のようなアプローチにより，大きなフィルタサイズの畳み込みにおいてもパラメータ数や訓練時間を削減するための方法を提供する．つまり，畳み込みフィルタの生成を含む訓練方法の研究により，小規模な訓練環境における画像処理モデルの獲得に資することを目的とする．

1.3　本論文の骨子となった論文について

本論文は著者が慶應義塾大学大学院理工学研究科の後期博士課程で研究および発表した [77–79] をもとに執筆した．[77] は第 3 章，[78, 79] は第 4 章の内容に関係した研究論文である．

第 2 章　畳み込みフィルタの生成と学習原理

2.1　コンピュータ上における数値と本論文における数の取り扱いの注意

本研究は後述するデジタル画像をコンピュータ上で計算処理するためのものであるから，以下に述べられる議論における数値の大半は最終的にコンピュータ上で表現されることとなる．コンピュータは有限の計算資源によって構成され，特にデータは通例有限個のビット列（0 と 1 のいずれかの数に対応する状態の列）で表される．したがって，コンピュータ上で可算無限個存在する自然数や整数，不可算無限個存在する実数全体を区別することはできない．つまり，なんらかの方法を用いて有限個の元を選抜する必要があり，自然数や整数は絶対値の範囲を設定する，実数は有効数字のような表現（有効数字の桁数と指数の範囲が決まっている）をするといった方法がとられる．これらの数の集合は正確には有限集合であり，自然数全体の集合 $\mathbb{N} = \{0, 1, \cdots\}$（本論文では 0 を含む定義を採用する）や整数全体の集合 $\mathbb{Z} = \{\pm n \mid n \in \mathbb{N}\}$，実数全体の集合 $\mathbb{R}$ とは異なるものであるが，こういった理想的な数学的集合上で成り立つ法則がおよそ成り立つものとして扱う．本論文の以下の部分では $\mathbb{N}$，$\mathbb{Z}$，$\mathbb{R}$ の任意の元をコンピュータ上における数値で表現できるものとみなす．

2.2　画像処理における深層学習モデルと畳み込みフィルタ

画像処理における深層学習モデルでは画像の縦方向の要素数 D_1，画像の横方向の要素数 D_2，チャンネル数 C の 3 次元配列を処理する．stride が 2 以上の畳み込みを本論文では畳み込みとダウンサンプリングの 2 段階に分かれたものと見なすため，適当なパディングのもと，畳み込み層の処理は $\mathbb{R}^{D_1 \times D_2 \times C_{\mathrm{in}}} \to \mathbb{R}^{D_1 \times D_2 \times C_{\mathrm{out}}}$ の計算である．

いま，特徴マップ $\mathcal{X} \in \mathbb{R}^{D \times D \times C_{\mathrm{in}}}$，$\mathcal{Y} \in \mathbb{R}^{D \times D \times C_{\mathrm{out}}}$ がそれぞれ $\mathcal{W} \in \mathbb{R}^{k \times k \times C_{\mathrm{in}} \times C_{\mathrm{out}}}$ をもちいた畳み込み層による処理の入力，出力であるとする．この畳み込み層による計算は，$\mathcal{Y}$ における空間的な座標 (p, q)，ならびにチャンネルを表すインデックス c_{out} によって

$$\mathcal{Y}_{p,q,c_{\mathrm{out}}} = \sum_{c_{\mathrm{in}}=1}^{C_{\mathrm{in}}} \sum_{i=1}^{k} \sum_{j=1}^{k} \mathcal{W}_{i,j,c_{\mathrm{in}},c_{\mathrm{out}}} \cdot \mathcal{X}_{p+\Delta i, q+\Delta j, c_{\mathrm{in}}} \tag{2.1}$$

と書ける．ここで，Δi，Δj はそれぞれ $\mathcal{W}$ の空間的インデックスを表す i，j を特徴マップにおける相対位置座標に変換したものであり，小数点以下を切り上げて整数にする天井関数 $\lceil \cdot \rceil$ を用いて

$$\Delta i = i - \left\lceil \frac{k}{2} \right\rceil, \qquad \Delta j = j - \left\lceil \frac{k}{2} \right\rceil \tag{2.2}$$

と書ける．

ただし，本論文においてはチャンネル間の関係 $\sum_{c_{\mathrm{in}}=1}^{C_{\mathrm{in}}}$ は考慮せず，任意のチャンネル c_{in} に注目した際の畳み込み

$$\sum_{i=1}^{k}\sum_{j=1}^{k}\mathcal{W}_{i,j,c_{\mathrm{in}},c_{\mathrm{out}}}\cdot\mathcal{X}_{p+\Delta i,q+\Delta j,c_{\mathrm{in}}} \tag{2.3}$$

を考える．つまり，畳み込み層による計算を，各チャンネルにおける畳み込みの結果を合計したものとみなす．したがって，本論文で考える畳み込みは畳み込み層による処理よりもプリミティブな計算単位である．

さらに，

$$W_{i,j}=\mathcal{W}_{i,j,c_{\mathrm{in}},c_{\mathrm{out}}},\qquad x_{p,q}=\mathcal{X}_{p,q,c_{\mathrm{in}}} \tag{2.4}$$

で定義される $W\in\mathbb{R}^{k\times k}$ と $x\in\mathbb{R}^{D\times D}$ を用いて，

$$(W*x)_{p,q}=\sum_{i=1}^{k}\sum_{j=1}^{k}W_{i,j}\cdot x_{p+\Delta i,q+\Delta j} \tag{2.5}$$

と表現したシンプルな形式で議論を行うこととする．本論文において「畳み込みフィルタ」はもっぱら式 (2.5) の表式における W を指し，これを畳み込む際の「特徴マップ」は x のように個別のチャンネルに注目したものを意図している．

2.3 畳み込みフィルタの生成

3×3 畳み込みフィルタ

$$W=\begin{bmatrix}w_{1,1}&w_{1,2}&w_{1,3}\\w_{2,1}&w_{2,2}&w_{2,3}\\w_{3,1}&w_{3,2}&w_{3,3}\end{bmatrix} \tag{2.6}$$

を学習する典型的な方法は

$$\Theta=\begin{bmatrix}\theta_{1,1}&\theta_{1,2}&\theta_{1,3}\\\theta_{2,1}&\theta_{2,2}&\theta_{2,3}\\\theta_{3,1}&\theta_{3,2}&\theta_{3,3}\end{bmatrix} \tag{2.7}$$

のように学習可能なパラメータの配列を考え，それぞれのパラメータが最適な重みになるよう訓練することである．つまり，$W=\Theta$ のように畳み込みフィルタと学習可能なパラメータの配列を同一視する．

これに対し，本論文では必ずしも $W=\Theta$ を仮定せず，Λ 個の学習可能パラメータ $\theta=(\theta_\lambda)_{\lambda=1}^{\Lambda}$ によって

$$W_{i,j}=f(\theta,i,j) \tag{2.8}$$

のように一般化した形式を考える．この形式の下では，$W = \Theta$ は

$$f(\theta, i, j) = \theta_{ki+j} \tag{2.9}$$

とした特殊な例である．

式 (2.8) における f（すなわちフィルタの生成方法）を工夫することにより，畳み込みフィルタ W の学習に改善が期待できる．本研究のテーマは大きい畳み込みフィルタを訓練する状況を想定し，訓練後のモデルの推論精度向上や訓練時間の短縮を図ることである．

なお，式 (2.8) は入力特徴マップである x を変数に含まない．したがって，本研究で考える畳み込みフィルタの生成方法は静的なものに限られる．

2.4 生成を含む畳み込みフィルタの学習原理

本研究では Λ 個の学習可能パラメータ $\theta = (\theta_\lambda)_{\lambda=1}^{\Lambda}$ を最適化することにより，間接的に畳み込みフィルタ W を訓練・獲得する．この場合においても，深層学習モデルにおける他のパラメータと同様の方法で最適化を試みる．各パラメータ θ_λ に対する損失関数 L の勾配 $\dfrac{\partial L}{\partial \theta_\lambda}$ を求め，勾配降下法ベースの漸化式によってパラメータの更新を繰り返すことで損失関数 L の値を小さくする．

この際には，誤差逆伝播法 [39, 80]（error back-propagation method）にしたがい，

$$\frac{\partial L}{\partial \theta_\lambda} = \sum_{i=1}^{k}\sum_{j=1}^{k} \frac{\partial L}{\partial W_{i,j}} \cdot \frac{\partial W_{i,j}}{\partial \theta_\lambda} = \sum_{i=1}^{k}\sum_{j=1}^{k} \frac{\partial L}{\partial W_{i,j}} \cdot \frac{\partial f(\theta, i, j)}{\partial \theta_\lambda} \tag{2.10}$$

によって $\dfrac{\partial L}{\partial \theta_\lambda}$ を求める．ここで，$\dfrac{\partial L}{\partial W_{i,j}}$ は $W = \Theta$ の典型的な畳み込みの訓練に使用される勾配であるから，$\dfrac{\partial f(\theta, i, j)}{\partial \theta_\lambda}$ さえ求めることができれば訓練可能となる．したがって，f はそれぞれの θ_λ により微分可能であれば任意に定義することができ，f の定義を定めれば θ の最適化方法は自然に定まる．

2.5 畳み込みフィルタの生成コスト

本研究のように畳み込みフィルタ W を生成する場合，生成および訓練には追加の計算が必要である．具体的には，W を生成するための計算（式 (2.8)）と誤差逆伝播のための勾配計算（式 (2.10)）のコストが発生する．しかしながら，このコストは画像処理モデルの典型的な深層学習フレームワークの中では問題にならない．

深層学習では複数の入力画像をまとめたミニバッチ単位で推論および訓練を行う．たとえば，本研究の実験では 32 枚の画像を並行して処理する．この場合，画像処理モデル中の畳み込み処理は異なる 32 サンプルに対してそれぞれ行われる．これは 1 つの畳み込みフィルタ W が 32 回使用されることを意味する．一方，W の生成や誤差逆伝播の計算はミニバッチのサンプル数に依存せず，1 回のみ実行される．つまり，ミニバッチ中の画像 1 枚に対する畳み込みフィルタの生成コストは

ミニバッチのサンプル数に反比例する形で小さくなる．したがって，畳み込みフィルタの生成コストは確実に生じるが，十分なミニバッチサイズが確保された訓練では問題とならない．本研究の提案手法を導入すると訓練ステップ数や 1 ステップあたりの計算コストを小さくすることができるため，畳み込みの生成コスト込みでもトータルの訓練コストは改善される．

また，訓練終了後に画像処理モデルを使用する際には 1 度生成した畳み込みフィルタを再利用することができる．訓練時にはフィルタの生成に必要な θ の値が毎回変化するため，その都度 W を計算し直す必要が生じるが，訓練が終了した後は θ の値が変化しないため，何度生成しても W の値は変わらない．したがって，訓練済みモデルを使用する際においても畳み込みフィルタの生成コストは十分に小さいものとして扱える．

第 3 章　近似モデル化された主成分の線形結合によるフィルタ生成

畳み込みフィルタを学習可能なパラメータの配列で表す標準的な方法ではそれぞれのパラメータの役割が局所的である．任意の畳み込みフィルタが表現可能な一方で，畳み込みフィルタの縦横の要素数それぞれに比例した個数のパラメータを必要とする．

そこで，畳み込みフィルタを連続関数をベースとしたフィルタ成分の線形結合とみなす Structured Receptive Field（SRF）の考え方を導入する．SRF ではそれぞれの役割を持ったフィルタ成分の係数を学習パラメータとするため，パラメータの個数はフィルタ成分の個数と一致し，畳み込みフィルタのサイズに依存しない．SRF の成分系としては唯一 N-Jet のみ知られているのが現状である．

本章では畳み込みフィルタの主成分を近似モデル化した新しい成分系を提案する．より寄与が大きいと考えられる成分を優先的に採用することで，N-Jet よりも少ない成分によるフィルタ表現を可能とする．また，提案した成分系は N-Jet よりも勾配爆発を引き起こしにくく，多くのモデルで訓練可能である．

3.1　線形結合による畳み込みフィルタの生成

3×3 の畳み込みフィルタ W と重みを表す学習可能なパラメータの配列 Θ の関係 $W = \Theta$ は

$$W = \theta_{1,1} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \theta_{1,2} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \cdots + \theta_{3,2} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + \theta_{3,3} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

のように線形結合の形で書き表すことができる．式 (3.1) において（係数ではなく）ベクトルの役割を果たしている，インデックス (i, j) における値だけが 1 でほかが 0 であるような成分の集合を（この例では $\mathbb{R}^{3\times3}$ の）標準基底（standard basis）いう．その元を $S_{i,j}$ で書くとすれば，式 (3.1) は

$$W = \sum_{i=1}^{3} \sum_{j=1}^{3} \theta_{i,j} S_{i,j} \tag{3.2}$$

の形で表され，より一般に，Λ 個の学習可能パラメータ $(\theta_\lambda)_{\lambda=1}^{\Lambda}$ ならびにフィルタ成分 $(F_\lambda)_{\lambda=1}^{\Lambda}$ を用いて

$$W = \sum_{\lambda=1}^{\Lambda} \theta_\lambda F_\lambda \tag{3.3}$$

のような畳み込みフィルタの表式を得ることができる．本章におけるフィルタの生成は式 (3.3) の $(F_\lambda)_{\lambda=1}^{\Lambda}$ として適当なものを探す試みである．これにより，学習で獲得される W がより推論精度を向上させること，ある水準を満たす W の学習にかかる時間が短くなることを期待する．

式 (3.1) にあるような標準基底の線形結合による表現は任意の畳み込みフィルタを表現可能な一方，それぞれの学習可能パラメータ $\theta_{i,j}$ の持つ意味は限定的なものとなる可能性がある．各 $\theta_{i,j}$ はそれぞれが受け持つ相対位置（3×3 フィルタなら $(i-2, j-2)$）の情報の重みを表す．これは 3×3 のフィルタにおいてはいずれも重要であると考えられるが，たとえば以下のような 9×9 の成分

$$S_{7,3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad S_{8,3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

を考えた際に，この 2 成分の重みにどの程度の差があるのかは疑問である．このようにフィルタサイズが大きくなるほど，特に中心から離れた位置においてその重みを表す各パラメータの意味は限定的なものとなり，学習可能なパラメータを冗長に用いた表現となる．現状は 3×3 の畳み込みフィルタが広く用いられるため，それぞれの $\theta_{i,j}$ はある程度以上価値のあるものと考えられるが，近年では 7×7 や 31×31 といった畳み込みフィルタの価値が知られるようになってきており，大きいサイズの畳み込みフィルタが広く用いられるようになれば，上に挙げた懸念は意味を持つことになる．

そこで，本章における研究ではフィルタ成分 F_λ の各位置 (i, j) における値を i，j を用いた計算によって得られるものとし，また，フィルタ成分全体にわたって 0 以外の値をもつように設計する．これはたとえば Gaussian フィルタのようなものが考えられ，9×9 のフィルタなら

$$F_\lambda(i,j) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-5)^2+(j-5)^2}{2\sigma^2}\right) \tag{3.4}$$

の式から（$\sigma = 5$ のとき）

$$F_\lambda = \begin{bmatrix} 0.00336 & 0.00386 & 0.00427 & 0.00453 & 0.00462 & 0.00453 & 0.00427 & 0.00386 & 0.00336 \\ 0.00386 & 0.00444 & 0.00491 & 0.00521 & 0.00532 & 0.00521 & 0.00491 & 0.00444 & 0.00386 \\ 0.00427 & 0.00491 & 0.00542 & 0.00576 & 0.00588 & 0.00576 & 0.00542 & 0.00491 & 0.00427 \\ 0.00453 & 0.00521 & 0.00576 & 0.00612 & 0.00624 & 0.00612 & 0.00576 & 0.00521 & 0.00453 \\ 0.00462 & 0.00532 & 0.00588 & 0.00624 & 0.00637 & 0.00624 & 0.00588 & 0.00532 & 0.00462 \\ 0.00453 & 0.00521 & 0.00576 & 0.00612 & 0.00624 & 0.00612 & 0.00576 & 0.00521 & 0.00453 \\ 0.00427 & 0.00491 & 0.00542 & 0.00576 & 0.00588 & 0.00576 & 0.00542 & 0.00491 & 0.00427 \\ 0.00386 & 0.00444 & 0.00491 & 0.00521 & 0.00532 & 0.00521 & 0.00491 & 0.00444 & 0.00386 \\ 0.00336 & 0.00386 & 0.00427 & 0.00453 & 0.00462 & 0.00453 & 0.00427 & 0.00386 & 0.00336 \end{bmatrix}$$

が得られる．このようなフィルタ成分 F_λ の係数となる学習可能パラメータ θ_λ はこのフィルタ成分の重要性（あるいは使い方）を規定したものとなっており，また，フィルタサイズがどれだけ大きくなってもこの成分に対するパラメータは 1 つだけでよい．一方，標準基底によってこれを得るにはパラメータが 81 個必要であり，このフィルタを用いるのが最適だった場合であっても，これを各位置で独立してそれぞれの相対位置における $\theta_{i,j}$ を調整しなければならない．したがって，i

と j からフィルタ成分を算出するアプローチではパラメータ 1 つを調整すればよいため（そのフィルタ成分が実際に役に立つことを条件に）効率的に最適化することができる．

また，この方法ではフィルタ成分の設計意図により，その成分に対する係数が大きくなるよう学習された際にフィルタの意味を理解しやすくなることが期待できる．たとえば，前述のガウシアンフィルタ成分の係数が大きければ，そのフィルタは特徴マップに対するぼかし処理を行ったと考えられる．

このような，i と j からフィルタ成分を算出する方法を Structured Receptive Field [64]（SRF）といい，N-Jet と呼ばれる成分系の実装例がある．本章の研究においては畳み込みフィルタ W の主成分を近似モデル化し，モデル化された主成分の線形結合として生成したフィルタの学習を提案する．

3.2 関連研究

畳み込みフィルタの設計に関する先行研究についてまとめる．ここでは，式 (3.3) のようなフィルタ成分による線形結合の形で表した際の成分 F_λ に注目する．

畳み込みフィルタの設計に関する研究を成分によって分類した場合，「n-hot 型」 [61–63] と「連続関数型」 [64,81–83] にわけられる．ここで「n-hot」とは各成分が，n 個の点でのみ値 1 あるいは -1 を持ち，その他は 0 であるようなものをいう．なお，n は同じ成分系においても変化しうる．標準基底における成分はすべて one-hot であると言える．本章において提案する成分系 OtX は後者にあたる．

3.2.1 n-hot 型成分系

まずは n-hot 系統の手法について述べる．ただし，いずれも「わずかな劣化でパラメータ削減した」ことを強みとしており，実験では比較を行っていない．

Symmetric Kernels [61]（SymKer）は水平方向の対称性を担保することを目的としたものである．具体的には

$$W_1 = \begin{bmatrix} \theta_1 & \theta_2 & \theta_1 \\ \theta_3 & \theta_4 & \theta_3 \\ \theta_5 & \theta_6 & \theta_5 \end{bmatrix} \tag{3.5}$$

のようなフィルタとなる．これを線形結合として表すと

$$W_1 = \theta_1 \cdot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \theta_2 \cdot \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \cdots + \theta_5 \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} + \theta_6 \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.6}$$

のように，1 から 2 個の要素だけが 1 であるような成分であることが見てとれる．他の 2 手法と比較すると，この左右対称な 1 種類の成分系のみで学習を行っているのが特徴的である．

N-way Parameterization [62] は，3 種類のパターンで対称軸を考え，その軸で折り返して重ね合わさるようにパラメータを配置することを提案している．具体的には

$$W_2 = \begin{bmatrix} \theta_1 & \theta_2 & \theta_2 & \theta_1 \\ \theta_3 & \theta_4 & \theta_4 & \theta_3 \\ \theta_3 & \theta_4 & \theta_4 & \theta_3 \\ \theta_1 & \theta_2 & \theta_2 & \theta_1 \end{bmatrix}, \quad W_3 = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \\ \theta_2 & \theta_5 & \theta_6 & \theta_3 \\ \theta_3 & \theta_6 & \theta_5 & \theta_2 \\ \theta_4 & \theta_3 & \theta_2 & \theta_1 \end{bmatrix}, \quad W_4 = \begin{bmatrix} \theta_1 & \theta_2 & \theta_2 & \theta_1 \\ \theta_2 & \theta_3 & \theta_3 & \theta_2 \\ \theta_2 & \theta_3 & \theta_3 & \theta_2 \\ \theta_1 & \theta_2 & \theta_2 & \theta_1 \end{bmatrix} \tag{3.7}$$

の 3 種類である．水平方向と鉛直方向に対称な W_2 は

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

を，両対角線方向に対称な W_3 は

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

を，水平方向と鉛直方向と両対角線方向に対称な W_4 は

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

をそれぞれ成分に持つ．なお，どの成分系を使うかは割合をそれぞれ設定する．

SymNet [63] においては点対称な 3 種類のフィルタ

$$W_5 = \begin{bmatrix} \theta_1 & \theta_2 & \theta_1 \\ \theta_2 & \theta_3 & \theta_2 \\ \theta_1 & \theta_2 & \theta_1 \end{bmatrix}, \quad W_6 = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_4 \\ \theta_3 & \theta_2 & \theta_1 \end{bmatrix}, \quad W_7 = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & 0 & -\theta_4 \\ -\theta_3 & -\theta_2 & -\theta_1 \end{bmatrix} \tag{3.8}$$

が設定されている．等方的点対称な W_5 は

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

を，W_5 の等方性をなくした，単純に点対称な W_6 は

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

を，反点対称な W_7 は

$$\begin{bmatrix}1&0&0\\0&0&0\\0&0&-1\end{bmatrix}, \begin{bmatrix}0&1&0\\0&0&0\\0&-1&0\end{bmatrix}, \begin{bmatrix}0&0&1\\0&0&0\\-1&0&0\end{bmatrix}, \begin{bmatrix}0&0&0\\1&0&-1\\0&0&0\end{bmatrix}$$

を，それぞれ成分に持つ．負の成分を持った成分が特徴的であり，また，N-way Parameterization と同様，3 種類のフィルタの使用割合はハイパーパラメータとして設定される．なお，W_4 と W_5 は似通っているが，W_5 は等方的点対称であるため，11×11 に拡張した際，$(\Delta i, \Delta j) = (0, 5)$ と同じ値を持つ部分として，$(0, -5)$，$(5, 0)$，$(-5, 0)$ のほかに $(3, 4)$ なども含まれることとなる．したがって，W_4 と W_5 は厳密には異なる成分系であると言える．

これらの手法はフィルタに対する対称性を求めるための制約を設けているが，SymKer のように制約が厳しすぎたり，あるいは他の 2 種類の手法のように，フィルタ系の選択に関するハイパーパラメータ調整を要求される．また，フィルタサイズが大きくなるほど，その要素数にほぼ比例する形で必要な成分と学習可能パラメータが増えることとなる．

3.2.2 連続関数型成分系

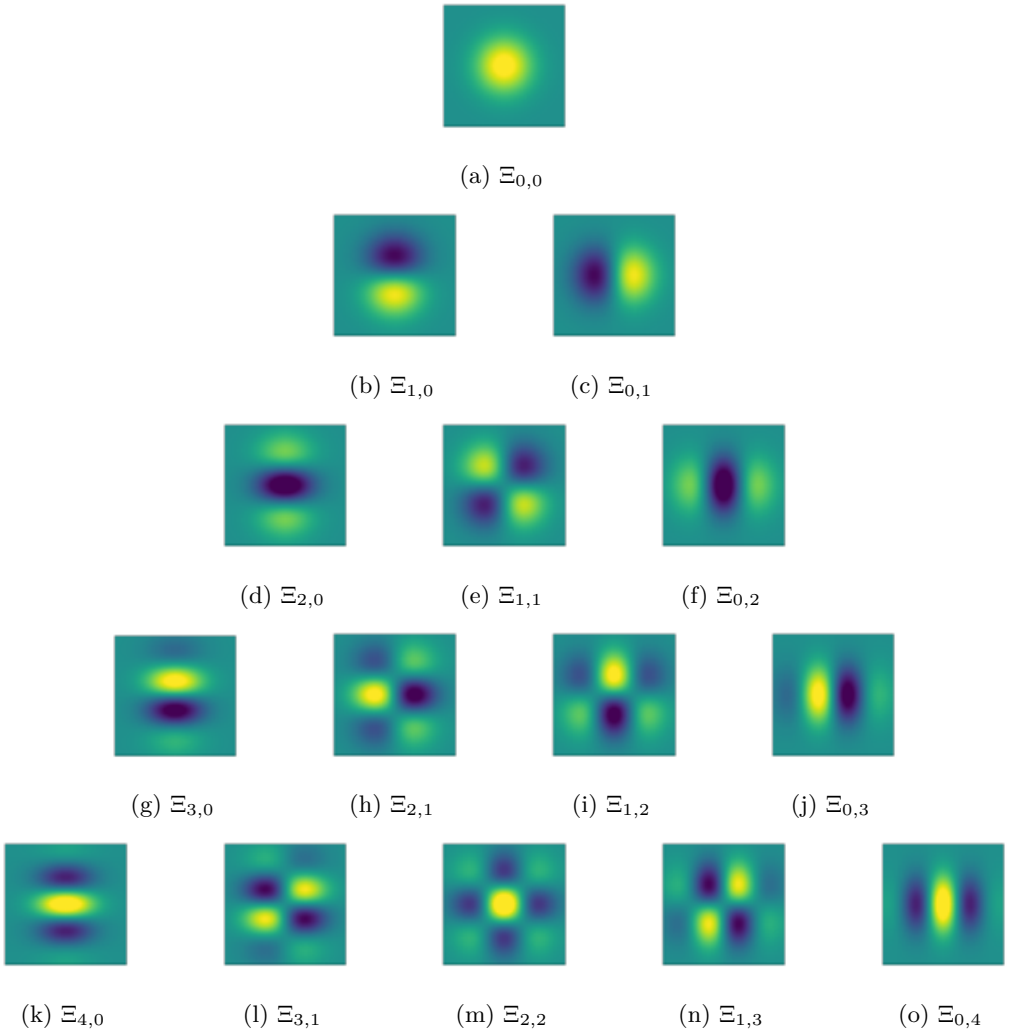
一方，連続関数型の成分系は n-hot 型の成分系（の組）が持つ上記のような点を克服している．適用する成分の数を決めるという選択肢は存在するが，適当な成分の数はおよそ決まっており，成分の数を調整するにも数回で十分だと考えられる．また，相対座標からフィルタの重みを計算するため，フィルタサイズによらず成分の数を固定することもできる．

ただし，現状この連続関数型アプローチは N-Jet [1] という 1 種類の成分系とその派生形からなる．N-Jet の成分 $\Xi_{m,n}$ は Gauss 関数 $\exp\left(-\dfrac{x^2}{2\sigma^2}\right)$ の m 階導関数と n 階導関数をそれぞれ鉛直方向と垂直方向の関数としてかけあわせたものである．Gauss 関数は C^∞-級であるため，N-Jet の成分系は可算無限個の成分からなるが，そのうち深層学習で用いられるものを 15 種類抜粋して図 3.1 に示す．これらの成分の意味やこれを生成するための数式は提案手法を述べてから説明したほうが見通しが良くなるため，3.4.5 節にて述べる．本章における研究はこれまで唯一用いられてきた N-Jet とは異なる連続関数型成分系を提案し，新たな選択肢を提示するものである．

N-Jet 成分系は深層学習において構造化受容野（SRF）を導入した [64] にて用いられた．図 3.1 における上位 4 段の 10 成分を用いたものを本論文では N-Jet-10 と呼ぶ．Network in Network [84]（NiN）を用い，Gauss 関数の σ のバリエーションを複数用意した[1)]N-Jet-10 により，ILSVRC2012 データセット [21] の 100 クラス抜粋したデータセットにおける実験を行った結果，標準基底による畳み込みフィルタよりも推論精度が高くなる（正答率 67.30%→ 69.65%）ことを示した．ただし，1000 クラスすべてを用いた場合には逆に標準基底よりも推論精度が低くなった（正答率 56.78%→ 50.08%）ことを報告している．

N-Jet の σ を学習対象とし，必要に応じてフィルタサイズを変更することを提案した N-Jet-Net [82] では，先の [64] と比較してはいないが， [83] によればこれを改善した（たとえば

1） 最初の層では $\sigma = 1, 2, 4, 8$ の 4 つ，そこから 1 つずつ大きい σ を減らしていき，最終の第 4 層で $\sigma = 1$ のみ



(a) $\Xi_{0,0}$

(b) $\Xi_{1,0}$ (c) $\Xi_{0,1}$

(d) $\Xi_{2,0}$ (e) $\Xi_{1,1}$ (f) $\Xi_{0,2}$

(g) $\Xi_{3,0}$ (h) $\Xi_{2,1}$ (i) $\Xi_{1,2}$ (j) $\Xi_{0,3}$

(k) $\Xi_{4,0}$ (l) $\Xi_{3,1}$ (m) $\Xi_{2,2}$ (n) $\Xi_{1,3}$ (o) $\Xi_{0,4}$

図 3.1: N-Jet [1] 成分系

CIFAR-100 データセットの正答率 61.50%→ 68.30%）ことが見てとれる．N-Jet-Net における提案自体はこういった空間的スケーリングに関するものであるから，本章における成分系の提案とは対立的な立場にない．本章において提案する成分系においても，空間的な広がりを制御するための σ を学習可能としている．

Frac-SRF [83] においては N-Jet における Gauss 関数の微分階数を整数から有理数に拡張し，これを学習可能とした．ただし，これは整数階の通常の N-Jet 成分の内分をとることによって近似したものである．層内のフィルタで共有される，先の σ に加え，水平方向・鉛直方向の階数を表すパラメータ p，q と成分の係数 α を 1 フィルタあたりに学習可能なパラメータとして設定している．$m \leq p < m+1$，$n \leq q < n+1$ を満たすような自然数 m と n を考えたとき，図 3.1 における $\Xi_{m,n}$，$\Xi_{m+1,n}$，$\Xi_{m,n+1}$，$\Xi_{m+1,n+1}$ が成分として用いられる．フィルタ 1 つに対しおよそ 3 つのパラメータのみを用いるというパラメータの少なさを強みとしており，成分系としては N-Jet を仮定していることに変わりない．こちらはパラメータを少なくすることには成功したが，推論精度については N-Jet-Net，つまり通常の N-Jet 成分とほぼ同等，やや下回る結果（CIFAR-100 データセットの正答率 68.30%→ 67.80%）となっている．

また，これは成分の線形結合によるフィルタ生成ではないが，決まった角度（$\frac{\pi}{6}$ や $\frac{\pi}{12}$ 刻み）の Gabor フィルタを画像に近い層で畳み込んだ研究例 [81] もある．この研究においては 2 層の畳み込みののち全結合計算を行う非常に軽量なモデルで検証を行っており，1 層目には Gabor フィルタのみ，2 層目には半分だけ Gabor フィルタを用いるのがよいとしている．

これらの背景を踏まえ，連続関数型で線形結合を行うための成分系として有力なものは N-Jet のみであると結論づけた．したがって，本章における実験では標準基底によるものと N-Jet 成分系によるものを比較対象としている．

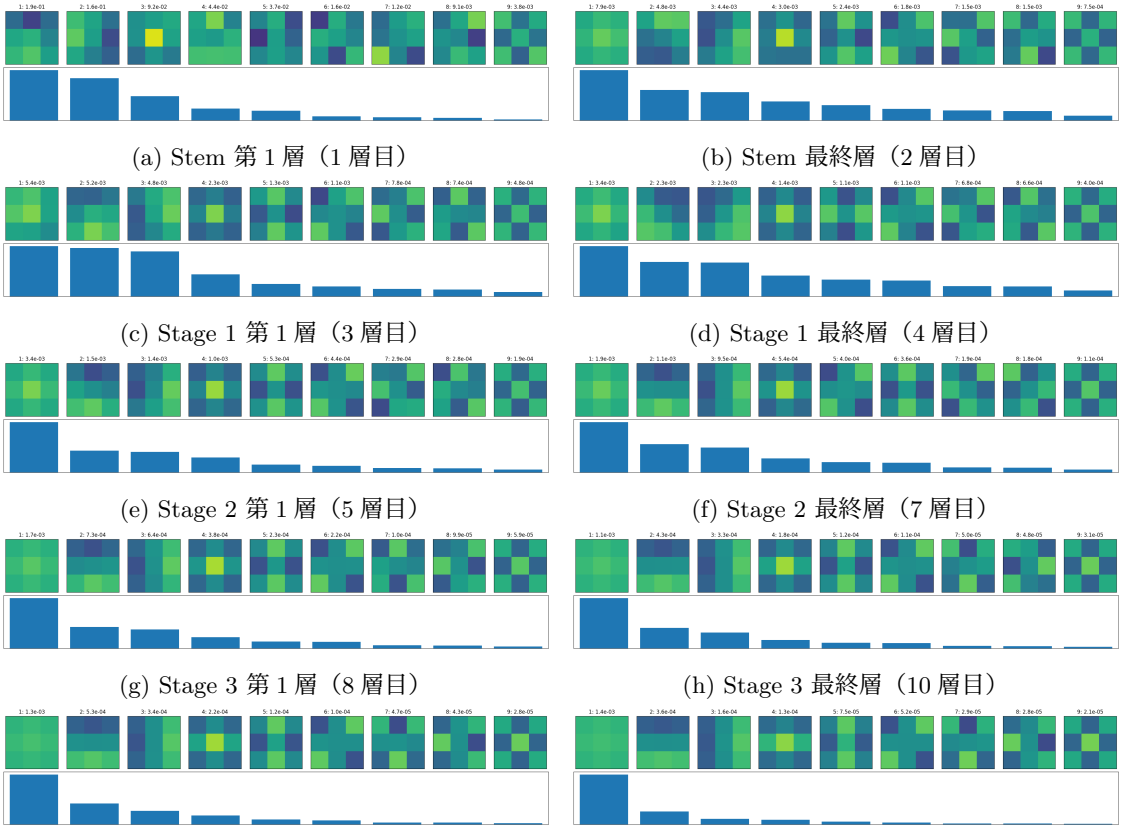
3.3 学習済み畳み込みフィルタの主成分分析

主成分分析（principal component analysis；PCA） [85, 86] はデータを少数の成分の線形結合で表現するための基本的な方法のひとつである．なるべく少数の成分で元データを近似できるような成分を抽出する．成分には優先順位が存在し，その優先順位が高いものから順に成分系に加えられる．ただし，元データは原点付近に団子状の分布を持ったものが望ましく，たとえばドーナツのような分布を持ったデータ群に対しては直接利用しても効果が薄い．この際には適当な写像をもって要件を満たすような線形空間に写すことが求められる．データの分布については注意深く議論するべきであるが，本章では暗に学習済み畳み込みフィルタが正規分布様の分布をもったデータであると仮定する．

もしも学習済みの畳み込みフィルタに共通の主成分が存在するとすれば，フィルタをそれらの線形結合で表すことでパラメータ効率の高い表現が得られる．線形結合の各係数を学習可能パラメータとすることで学習を行うが，優先順位の低い成分の係数を 0 と仮定すれば学習可能パラメータを少なくすることができる．優先順位の低い成分がなくとも学習済みフィルタは十分近似することができるため省略してもかまわないとするのである．逆にある程度以上優先順位の低い成分がノイズ

であるとすれば，より汎化されたフィルタが得られることも期待できる．

本研究では代表的な畳み込みニューラルネットワークモデルのひとつである VGG-16 [2] の畳み込みフィルタの主成分を分析した．VGG-16 には 13 個の畳み込み層が存在するが，そのうち 10 個を対象に分析を実施した．学習済みのパラメータとしては PyTorch Image Models [87]（TIMM）にて公開されている，ILSVRC2012 データセット [21] 訓練済みのものを利用した．なお，フィルタの各成分をそのまま数ベクトルの成分と見なした．つまり，フィルタ内の隣接位置のつながりなどはまったく考慮せず，各相対位置で独立に処理を行っている．



(a) Stem 第 1 層（1 層目）　(b) Stem 最終層（2 層目）

(c) Stage 1 第 1 層（3 層目）　(d) Stage 1 最終層（4 層目）

(e) Stage 2 第 1 層（5 層目）　(f) Stage 2 最終層（7 層目）

(g) Stage 3 第 1 層（8 層目）　(h) Stage 3 最終層（10 層目）

(i) Stage 4 第 1 層（11 層目）　(j) Stage 4 最終層（13 層目）

図 3.2: 学習済み VGG-16 [2] の畳み込みフィルタに対する主成分分析の結果

主成分分析の結果を可視化したものを図 3.2 に示す．(a) から (j) はそれぞれ異なる畳み込み層における分析結果であり，入力に近いものから順に並べてある．それぞれの事例においては上にフィルタ成分を可視化したものを，下にその標準偏差を棒グラフで示している．この棒グラフが相対的に高いほどフィルタの中でその成分が支配的を示しており，重要な成分だということができる．なお，黄色が正，青が負で，緑に近いほど絶対値が 0 に近いが，これらの成分は線形結合して使われることを想定しているため，そのスカラー倍，特に -1 倍を意味する正負（黄色と青色）を反転したものについては同一と見なすことができる．

各層における結果を見比べてみると，ほとんど同じような成分がほとんど同じような順番で並んでいることが見てとれる．このことから，少なくとも VGG-16 において畳み込みフィルタの主成分足りうる成分はおよそ決まっていることが予想される．つまり，ここから何らかの法則を導き出すことによりこれを生成することができれば，畳み込みフィルタの汎用的な成分系を構成することが可能であることを示唆している．したがって，以下ではこの結果を支配する法則を導くために詳細な観察を行う．

まず，ひとつひとつの成分について見ていくと，それぞれがなんらかの対称性をもっていることが見てとれる．たとえば (g) においては第 1 主成分は水平方向にも鉛直方向にも軸対称的であり，さらに点対称性も持っている．第 2 主成分は水平方向については偶対称的，鉛直方向には奇対称な対称性が見られる．このように対称性を見ていくと，第 1，第 4，第 9 主成分は $\frac{\pi}{2}$ 回転させたときにそれぞれ自身が重なることがわかる．これらの 3 成分は特に角度を指定せずとも，自身と重なってもおかしくないであろう回転対称性を持っている．

また，それ以外の成分は隣にペアとなるような回転異性体を持っていることが見てとれる．たとえば (g) においては，第 2，第 3 主成分，第 7，第 8 主成分が $\frac{\pi}{2}$ 回転させたときに重なるように見受けられる．以上の 2 組はわかりやすいが，さらに第 5，第 6 主成分についても $\frac{\pi}{4}$ 回転させれば重なることが予想される．

なお，前者の 2 組について π 回転させた際には正負反転した自身と重なり，第 5，第 6 主成分を $\frac{\pi}{2}$ 回転させた際にも正負反転した自身と重なる．先に述べたようにこれらの正負は無視することができるから，これらの 3 組についてはお互いが重なるような回転量について，回転異性体を網羅していると言える．つまり，それぞれの組について他の回転異性体になりえる成分を考える必要はない．

これによって，主成分は

1. 回転対称性を持つもの（第 1，第 4，第 9 主成分）
2. 隣に $\frac{\pi}{2}$ の回転異性体ペアを持つもの（第 2，第 3 主成分；第 7，第 8 主成分）
3. 隣に $\frac{\pi}{4}$ の回転異性体ペアを持つもの（第 5，第 6 主成分）

の 3 種類に分類できることがわかる．なお，2. と 3. について，標準偏差を表す下の棒グラフを見てみても，これらのペアではほとんど同じ高さとなっていることが見てとれるから，ほとんど同じ重要性を持ったペアであるということができる．

なお，この時点で学習済み畳み込みフィルタの主成分が図 3.1 に示した N-Jet 成分系ではないことが見て取れる．「余り」の成分である第 9 主成分を除いても，少なくとも第 1，第 4 主成分のように回転対称性を持つ成分が 2 つ以上存在しなければならないと考えられるが，N-Jet 成分系において回転対称性を持っているのは $\Xi_{0,0}$ の 1 つのみである．したがって，学習済みの畳み込みフィルタにおける主成分を構成するには N-Jet とは異なる成分系を導かなければならないと言える．

再び図 3.2 に注目すると，(a) を除く各層の第 1 主成分を比較した際，出力に近くなるほどのっぺりとした（空間的に広がりを持った）フィルタ成分になっていることがわかる．つまり，汎用的

な主成分を設計するうえでは，N-Jet-Net [82] の例に見られるように，空間的な広がりを意味するパラメータを学習可能とすることが望ましいと考えられる．その他，第 1 主成分は Gaussian フィルタに近い見た目をしていることや，各層において左ほど起伏の少ない成分となっていることが見てとれる．

最後に，図から読み取ることは難しいが，これが主成分分析であることが重要な点として挙げられる．主成分分析によって得られる成分同士はそれぞれ直交するから，学習済みフィルタの主成分としては直交なものを考えるのが妥当である．

3.4 畳み込みフィルタの主成分の近似モデル

3.3 節にて可視化した学習済み畳み込みフィルタの主成分を数式で表現する方法を提案する．この近似モデル化が本研究において得られた成果の 1 つである．結論として Hermite 多項式をベースとした可算無限個の成分が得られるが，その成分についての重要度を算出することもでき，実際の学習においては有限個の成分をこの重要度に基づいて選び出すことで成分系を構成する．

3.4.1 Hermite 多項式と Hermite-Gauss 関数

Hermite 多項式には「確率論学者の Hermite 多項式」と「物理学者の Hermite 多項式」が存在するが，本論文においてはもっぱら後者の「物理学者の Hermite 多項式」を指すこととする．これは Hermite 多項式が持つ重み付き直交性（後述）を記述する上で必要となる重み関数の違いにより生ずるものであり，畳み込みフィルタの主成分をモデル化する上で都合の良いほうを採用している．

まずは，天下り的ではあるが Hermite 多項式の定義を述べる．

定義 1（Hermite 多項式） $n \in \mathbb{N}$ とする．このとき n 次の Hermite 多項式 $H_n(x)$ を

$$H_n(x) = (-1)^n \exp(x^2) \frac{\mathrm{d}^n}{\mathrm{d}x^n} \exp(-x^2) \tag{3.9}$$

と定義する．

具体的には

$$\begin{aligned} H_0(x) &= 1, \\ H_1(x) &= 2x, \\ H_2(x) &= 4x^2 - 2, \\ H_3(x) &= 8x^3 - 12x, \\ H_4(x) &= 16x^4 - 48x^2 + 12, \\ &\vdots \end{aligned}$$

のような多項式となる．

Hermite 多項式を具体的に求める上では定義 1 から直接計算する以外にも以下の定理 1 を用いることもできる．

定理 1（Hermite 多項式の漸化式）　任意の自然数 $n \geq 1$ に対し，n 次 Hermite 多項式は漸化式

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x) \tag{3.10}$$

を満たす．

定理 1 によれば $H_0(x) = 1$，$H_1(x) = 2x$ から $n \geq 2$ の場合の $H_n(x)$ を求めることができる．実験プログラムの実装においてもこの方法を採用した．

また，定理 1 より以下の系 1 が導かれる．

系 1（Hermite 多項式の偶奇性）　Hermite 多項式は偶数次ならば偶関数，奇数次ならば奇関数である．すなわち任意の $m \in \mathbb{N}$ に対して

$$H_{2m}(-x) = H_{2m}(x), \tag{3.11}$$

$$H_{2m+1}(-x) = -H_{2m+1}(x) \tag{3.12}$$

が成り立つ．

Hermite 多項式の持つ最も特筆すべき性質として**重み付き直交性**が挙げられる．定理 2 に示すこの性質こそがそれぞれが直交するフィルタ成分を設計する上で重要な役割を果たす．

定理 2（Hermite 多項式の重み付き直交性）　Hermite 多項式は $\exp(-x^2)$ を重み関数とした重み付き直交性を持つ．つまり，m，$n \in \mathbb{N}$ としたとき，m 次の Hermite 多項式 H_m と n 次の Hermite 多項式 H_n に対し

$$\int_{\mathbb{R}} H_m(x)H_n(x)\exp(-x^2)\,\mathrm{d}x = \sqrt{\pi}2^n n!\delta_{m,n} \tag{3.13}$$

が成り立つ．ただし，ここで $\delta_{m,n}$ は $m = n$ のとき 1，$m \neq n$ のとき 0 となる．

式 (3.13) の左辺において被積分関数は本来 2 つの Hermite 多項式に Gauss 関数 $\exp(-x^2)$ による重み付けを行ったものとみなされるが，それぞれの Hermite 多項式に対して $\exp\left(-\dfrac{x^2}{2}\right)$ を乗じた関数同士の積とみなすこともできる．このとき 2 つの関数は直交する．つまり，この事実を定義と系の形でまとめると以下のようになる．

定義 2（Hermite-Gauss 関数）　$n \in \mathbb{N}$ とする．このとき n 次の Hermite-Gauss 関数 $\psi_n : \mathbb{R} \to \mathbb{R}$ を

$$\psi_n(x) = H_n(x)\exp\left(-\frac{x^2}{2}\right) \tag{3.14}$$

と定義する．

系 2（Hermite-Gauss 関数の直交性）　Hermite-Gauss 関数は直交性を持つ．つまり，m，$n \in \mathbb{N}$ としたとき，m 次の Hermite-Gauss 関数 ψ_m と n 次の Hermite-Gauss 関数 ψ_n に対し

$$\int_{\mathbb{R}} \psi_m(x)\psi_n(x)\,\mathrm{d}x = \sqrt{\pi}2^n n!\delta_{m,n} \tag{3.15}$$

が成り立つ．ただし，ここで $\delta_{m,n}$ は $m = n$ のとき 1，$m \neq n$ のとき 0 となる．

この Hermite-Gauss 関数 $\psi_n(x)$ を適当に 2 次元に拡張することによりフィルタ成分を構成するのが 3.4.2 節の試みである．ここでフィルタの成分系の直交性を保証する足がかりとなるのが系 2 である．

また，$\exp\left(-\dfrac{x^2}{2}\right)$ は偶関数であるから系 1 より以下の系 3 が成り立つ．この系により 3.4.2 節で提案するフィルタ成分の対称性が保証される．

系 3（Hermite-Gauss 関数の偶奇性）　Hermite-Gauss 関数は偶数次ならば偶関数，奇数次ならば奇関数である．すなわち任意の $m \in \mathbb{N}$ に対して

$$\psi_{2m}(-x) = \psi_{2m}(x), \tag{3.16}$$

$$\psi_{2m+1}(-x) = -\psi_{2m+1}(x) \tag{3.17}$$

が成り立つ．

3.4.2　提案するフィルタの成分系（OtX 成分系）

ここからは実際に畳み込みフィルタの主成分を近似モデル化し，「OtX 成分系」を定義する．OtX はそれぞれのアルファベットが球対称成分，直交 2 軸対称成分（水平・鉛直），直交 2 軸対称成分（$\dfrac{\pi}{4}$ 回転）のシンボルになっている．

3.4.2.1　座標の定義

最終的な畳み込みフィルタは重み成分の配列となるが，まずはフィルタ成分を $\mathbb{R}^2$ 上の関数として定義し，それをどういった配列に変換するのかという問題を分けて考えることにする．本論文では 3.4.4 節に配列化の実装例を示す．

$\mathbb{R}^2$ 平面上の座標は 2 次元の直交座標とする．画像の水平方向に相当する座標を x，鉛直方向に相当する座標を y とし，原点は畳み込みにおける注目座標を仮定する．なお，x と y の正負の方向は以下の議論に影響しないが，実装上は x の正方向は右，y の正方向は**下**となった．

3.4.2.2 球対称成分

畳み込みフィルタには可分なものと不可分なものがある．ここでいう可分フィルタは f, $g : \mathbb{R} \to \mathbb{R}$ を用いて $(x, y) \mapsto f(x)g(y)$ の形で書けるようなものを指し，$f(x)$ を用いて横方向に畳み込んだあと $g(y)$ を用いて縦方向に畳み込むことで縦横走査して畳み込んだ場合と同じ結果が得られる特徴がある．これは計算量を少なくできる一方，先に述べたようにこのようなフィルタは限られている．

たとえば，主成分分析で得られる 4 番目の成分は以下の定理 3 より不可分であることがわかる．

定理 3（不可分なフィルタの例） $h : \mathbb{R}^2 \to \mathbb{R}$ とする．ある実数 x_+，$y_+ > 0$ の組が存在して

$$h(0,0),\ h(x_+,0),\ h(0,y_+) > 0, \qquad h(x_+,y_+) < 0$$

を満たすならば $h(x, y) = f(x)g(y)$ を満たすような f，$g : \mathbb{R} \to \mathbb{R}$ は存在しない．

証明 $h(x, y) = f(x)g(y)$ を満たす f，g が存在するならば，$h(0, 0) = f(0)g(0) > 0$ より $f(0)$ と $g(0)$ の正負は一致し，同様に $h(x_+, 0)$，$h(0, y_+) > 0$ から結局 $f(0)$，$g(0)$，$f(x_+)$，$g(y_+)$ の正負が一致していることが言える．したがって $h(x_+, y_+) < 0 < f(x_+)g(y_+)$ となり，仮定 $h(x, y) = f(x)g(y)$ に反する矛盾が導かれる． □

また，9 番目の成分については他の 8 主成分の余りと見なすこともできるため，単独の成分として扱うべきかは議論の余地があるが，その余りの成分の主成分もまた不可分であることが予測できる．以上のことから設計する成分系は不可分な成分を含むため，$f(x)g(y)$ の形で書けないような 2 変数関数を含める必要があると言える．

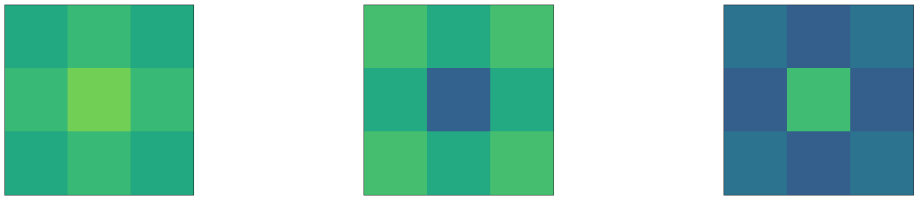
これらの不可分な成分を観察してみると $\frac{\pi}{2}$ 回転した際に自身とほぼ一致することがわかる．それゆえか，ペアとなるフィルタ成分が存在しない．そこで原点からの距離のみに依存する関数を定義することでこれを表現する．

定義 3（球対称成分 Φ_n） $n \in \mathbb{N}$ は偶数であるとする．また，$\sigma > 0$ とする．このとき n 次の球対称フィルタ成分 $\Phi_n : \mathbb{R}^2 \to \mathbb{R}$ を

$$\Phi_n(x, y) = \psi_n\left(\frac{r}{\sigma}\right) \tag{3.18}$$

と定義する．ただし，ここで $r = \sqrt{x^2 + y^2}$ である．

なお，定義 3 において σ は空間的な広がりを表すパラメータで [82] に倣い学習可能とする．このように定義された球対称成分によって，第 1，第 4，第 9 の主成分をそれぞれ Φ_0，Φ_2，Φ_4 によって近似することができる．ただし，これらの成分同士は直交性を持たない（命題 1）．実際に 3.4.4 節の方法で 3×3 のフィルタとして可視化したものを図 3.3 に示す．



(a) Φ_0　　(b) Φ_2　　(c) Φ_4

図 3.3: 3×3 の OtX 球対称成分 Φ_n

3.4.2.3　直交 2 軸対称成分

残る第 2，3 成分，第 7，8 成分，第 5，6 成分は直交 2 軸の対称成分として近似モデル化する．前 2 組は $\frac{\pi}{2}$ の回転異性体の関係にあり，残る 1 組は $\frac{\pi}{4}$ の回転異性体の関係にある．したがって，この回転による区別もパラメータ φ として含めた

$$\Psi^{(\varphi)}_{m,n}(x,y) = \psi_m\left(\frac{x_\varphi}{\sigma}\right)\psi_n\left(\frac{y_\varphi}{\sigma}\right) \tag{3.19}$$

のような形で定義する．ここで，x_φ，y_φ は座標 (x,y) を原点中心に φ の角度だけ回転させたものを指し，具体的には回転行列をかける

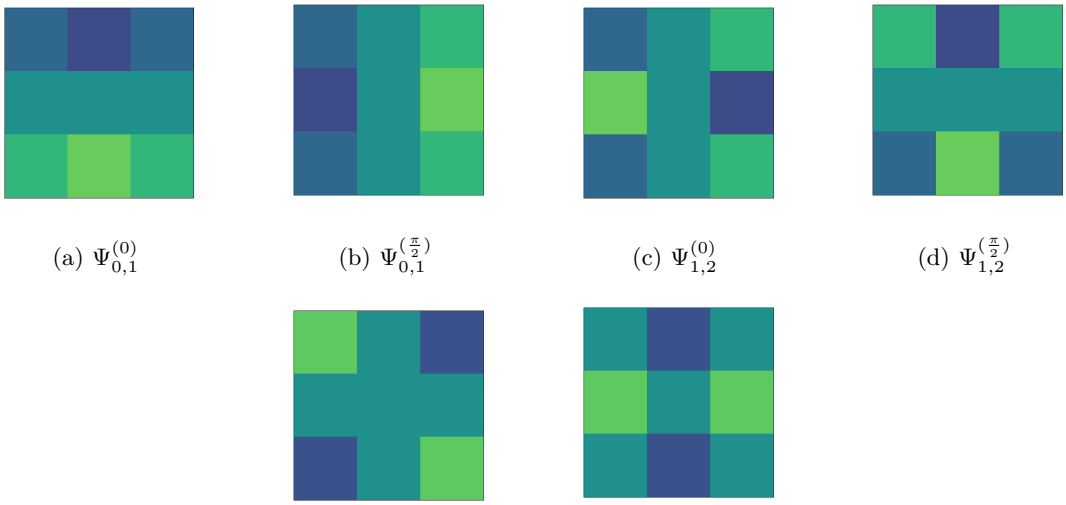
$$\begin{bmatrix} x_\varphi \\ y_\varphi \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.20}$$

の計算により求められる．空間的な広がりを表す学習可能パラメータ σ は球対称成分 Φ_n と共通のものを利用する．

この直交 2 軸対称成分のうち，図 3.2 の主成分として見られたものを図 3.4 に示す．ここでも 3.4.4 節の方法で 3×3 としている．(a)，(b) が第 2，3 主成分，(c)，(d) が第 7，8 主成分，(e)，(f) が第 5，6 主成分と似た形をしている．(a) から (d) は $\frac{\pi}{2}$ の回転異性体を持つ成分であったが，これらは $n = m+1$ となっている．また，$\frac{\pi}{4}$ の回転異性体を持つ (e)，(f) は 1 例しかないが，$n = m$ であり，かつ奇数次である．このことから，逆に，m，n と φ の関係は上記のものに限って直交 2 軸対称成分とする．つまり，$n = m+1$ のときは $\varphi \in \left\{0, \frac{\pi}{2}\right\}$，$n = m$ が奇数のときは $\varphi \in \left\{0, \frac{\pi}{4}\right\}$ とし，それ以外の m，n については $\Psi^{(\varphi)}_{m,n}$ は未定義とする．

3.4.3　有限個の OtX 成分の選び方

3.4.2 節で提案した成分系は可算無限個の球対称関数の集合 $\{\Phi_n\}$ と，同じく可算無限個の 2 軸線対称関数の集合 $\left\{\Psi^{(\varphi)}_{m,n}\right\}$ の元からなる可算無限個の成分を持っている．しかしながら，これらの成分は線形結合によってフィルタをなし，その係数は学習対象パラメータとなるため，すべての成分を使ってフィルタを構成するには可算無限個の学習対象パラメータを必要とする．つまり，有限のメモリという制約をもった実世界のコンピュータ上で提案手法を適用にするには有限個の成分を選抜しなければならない．また，選抜される有限個の成分によって必要なフィルタをおよそ近似できることが求められる．平たく言えば学習を行う上で有用な成分系を構築する必要がある．



(a) $\Psi_{0,1}^{(0)}$ (b) $\Psi_{0,1}^{(\frac{\pi}{2})}$ (c) $\Psi_{1,2}^{(0)}$ (d) $\Psi_{1,2}^{(\frac{\pi}{2})}$

(e) $\Psi_{1,1}^{(0)}$ (f) $\Psi_{1,1}^{(\frac{\pi}{4})}$

図 3.4: 3×3 の OtX 直交 2 軸対称成分 $\Psi_{m,n}^{(\varphi)}$

一般にこの問題は最も良い成分の組み合わせを求めなければならない．つまり，k 個の制約のもと最善である成分系を $\mathcal{C}_k$，$k+1$ 個の制約のもと最善である成分系 $\mathcal{C}_{k+1}$ としたとき $c \in \mathcal{C}_k \implies c \in \mathcal{C}_{k+1}$ は必ずしも成立しない．しかしながら，提案した OtX 成分系は畳み込みフィルタの主成分を近似モデル化したものであるから，畳み込みフィルタをより少ない成分で表現するための絶対的な主成分が存在すると仮定しても不自然ではない．この仮定のもとでは，可算無限個存在する成分の中から有用と考えられる上位 k 個の成分を貪欲に取り出す操作によって最適な成分系が得られることになる．したがって，以下では各成分の有用性を評価する方法について述べる．

提案する成分の評価方法は m，n の値を用いて成分のスコアを計算し，そのスコアが小さい順に成分を採用する方法である．球対称成分 Φ_n のスコアは単純に n とし，2 軸線対称成分 $\Psi_{m,n}^{(\varphi)}$ のスコアは $m+n+0.5$ とする．ここで後者に 0.5 を加えるのは球対称成分を選好するためのものである．たとえば，Φ_2 と $\Psi_{1,1}^{(\varphi)}$ とを比較した際は前者がより良い成分として評価される．したがって，加える値は開区間 $(0,1)$ の任意の元でよく，必ずしも 0.5 でなくともよい．また，2 軸線対称成分の回転異性体に対しては優劣をつけないこととし，原則的には 2 成分まとめて成分系に含めるか含めないかを判断し成分の数を決定する．つまり，6 成分の成分系に $\Psi_{1,2}^{(0)}$ と $\Psi_{1,2}^{(\frac{\pi}{4})}$ を加えて 8 成分にすることはあっても，いずれか一方のみを採用して 7 成分にする状況は考えないこととするのである．

この方法によって有用だと評価される OtX 成分系の上位 9 成分を並べたものが図 3.5 である．これを，実際の学習済み畳み込みフィルタの主成分分析結果である図 3.2 と見比べると，以上の近似モデル化，ならびに順序付けにより，きわめて類似した結果が得られたと言える．つまり，ここまで見てきた OtX 成分系は畳み込みフィルタの主成分をうまく近似モデル化したものであると考えられる．

(a) Φ_0 (b) $\Psi^{(0)}_{0,1}$ (c) $\Psi^{(\frac{\pi}{2})}_{0,1}$ (d) Φ_2 (e) $\Psi^{(0)}_{1,1}$ (f) $\Psi^{(\frac{\pi}{4})}_{1,1}$ (g) $\Psi^{(0)}_{1,2}$ (h) $\Psi^{(\frac{\pi}{2})}_{1,2}$ (i) Φ_4

図 3.5: 提案した OtX 成分系の上位 9 成分

3.4.4 連続フィルタの離散化

3.4.2 節にて設計したのは連続的な相対座標に対する関数フィルタ成分である．畳み込みを行う際に連続フィルタによる作用を理論的にシミュレートする方法も考えられるが，ここでは従来の畳み込みフィルタと同様に離散的な相対座標に対する重要度の $k \times k$ 配列を形成することとする．なお，フィルタ値の量子化については計算フレームワークの実装任せとして論じないこととする．

フィルタの受容野を x-y 直交座標上，$[-1,1] \times [-1,1]$ の領域と定める．これを $k \times k$ に等間隔にサンプリングしたものを $k \times k$ の離散フィルタにおける相対的な位置の情報として用いる．つまり，$k \times k$ 配列におけるインデックス (i,j) における相対座標 (y_i, x_j) を

$$y_i = \frac{2(i-1)}{k-1} - 1, \qquad x_j = \frac{2(j-1)}{k-1} - 1 \tag{3.21}$$

と計算し，これを各成分の x，y として用いることで離散的なフィルタ成分を算出する．また，各成分においては L_2 ノルムが 1 になるよう，値のスケーリングを行う．

3.4.5 N-Jet 成分系との違い

N-Jet 成分系は 3.4.2 節で提案した OtX 成分系と同様 Hermite 多項式と Gauss 関数をベースに設計された成分系である．しかしながら，設計理念や数学的な性質に違いがあるためその点について述べる．なお，原論文 [64] では成分系（component system）ではなく基底（bases）とされているが，連続関数で表現されるフィルタを想定しなければ基底であること（任意のフィルタをその成分だけで表現できること）は非自明であるためここでは成分系と表記する．

N-Jet 成分は Gauss 関数の n 回微分

$$\xi_n(x) = \frac{\mathrm{d}^n}{\mathrm{d}x^n} \exp(-x^2) = (-1)^n H_n(x) \exp(-x^2) \tag{3.22}$$

を用い

$$\Xi_{m,n}(x,y) = \xi_m(x)\xi_n(y) \tag{3.23}$$

と表されるものである．$m+n$ の値が小さいものほど重要な成分として選出され，原典では $m+n \leq 3$ を満たす 10 成分の成分系で実験が行われている．

以上の N-Jet の説明にあたり，定義 2 で導入した Hermite-Gauss 関数を用いずに式 (3.22) を新たに定義したのは恣意的である．両者を改めて併記すると

$$\psi_n(x) = H_n(x) \exp\left(-\frac{x^2}{2}\right), \qquad \xi_n(x) = (-1)^n H_n(x) \exp(-x^2)$$

のようになり，線形結合で利用されるフィルタ成分の定数倍（$(-1)^n$ の部分）が無視できることを鑑みれば両者の違いは exp 関数内の $-x^2$ を 2 で除するか否かのみである．しかしながら，この差が両者に大きく 2 点の違いを生み，それぞれは一般に比例の関係にはない．具体的には ψ_n は一般に Gauss 関数を微分して実数倍しても得ることができず，$\{\xi_n\}$ は直交性を持たない．これらの性質の違いは成分系の設計理念を反映して使い分けられる．たとえば直交性を持たない $\{\xi_n\}$ は主成分分析の結果の再現を試みる本研究の OtX 成分系の構成には不向きである．

N-Jet の設計理念は特徴マップの局所的 Taylor 展開に基づくものである．いま，簡単化のため 1 変数の関数 $f : \mathbb{R} \to \mathbb{R}$ を特徴マップとして考える．f が C^∞-級なら

$$f(x) \simeq \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{3.24}$$

のように Taylor 展開で近似することができる．これに Gauss 関数を畳み込むと

$$\begin{aligned}
f(x) * \exp(-x^2) &\simeq \left(\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \right) * \exp(-x^2) \\
&= \sum_{n=0}^{\infty} \frac{(\exp(-x^2) * f^{(n)}(x))(a)}{n!}(x-a)^n \\
&= \sum_{n=0}^{\infty} \frac{\left(\frac{\mathrm{d}^n}{\mathrm{d}x^n} \exp(-x^2) * f(x)\right)(a)}{n!}(x-a)^n \\
&= \sum_{n=0}^{\infty} \frac{(\xi_n * f)(a)}{n!}(x-a)^n
\end{aligned}$$

を得る．これはすべての $n \in \mathbb{N}$ に対する ξ_n を f に畳み込むことによって画像 f の Gauss 窓による局所 Taylor 展開の係数に比例する値が得られることを意味する．このようにして得られた係数（の定数倍）の線形結合をとることで有用な特徴抽出を試みる．ただし，n が大きくなるほど偶数の場合，奇数の場合でそれぞれ似通った（内積の絶対値が大きい）ξ_n となる [88] ため，実用的には $n = 4$ までで十分とされる．

また，これは原論文 [64] における説明とは異なるが ξ_i の線形結合からなるフィルタ

$$\xi(x) = \sum_{i=0}^{n} a_i \xi_i(x) = \sum_{i=0}^{n} a_i \frac{\mathrm{d}^i}{\mathrm{d}x^i} \exp(x^2) \tag{3.25}$$

を畳み込む状況を考えると

$$\begin{aligned}
f * \xi &= f(x) * \left(\sum_{i=0}^{n} a_i \frac{\mathrm{d}^i}{\mathrm{d}x^i} \exp(x^2) \right) \\
&= \sum_{i=0}^{n} a_i \left(f(x) * \frac{\mathrm{d}^i}{\mathrm{d}x^i} \exp(-x^2) \right) \\
&= \sum_{i=0}^{n} a_i \left(\exp(-x^2) * \frac{\mathrm{d}^i}{\mathrm{d}x^i} f(x) \right) \\
&= \exp(-x^2) * \left(\sum_{i=0}^{n} a_i f^{(i)}(x) \right)
\end{aligned}$$

より，画像の局所的な高階導関数に対する線形結合と見なすこともできる．つまり，各学習対象パラメータ（係数）はぼかしフィルタや微分フィルタをどの程度重視するかという尺度となる．

N-Jet には成分の任意の回転異性体を N-Jet 成分の線形結合で表すことができるという特徴もある．OtX 成分の直交 2 軸対称成分と同様 φ の回転を考えたものを $\Xi^{(\varphi)}_{m,n}$ などと書き表せば，たとえば

$$\Xi^{(\varphi)}_{2,0} = \cos^2(\varphi)\Xi_{2,0} - 2\cos(\varphi)\sin(\varphi)\Xi_{1,1} + \sin^2(\varphi)\Xi_{0,2}$$

$$\Xi^{(\varphi)}_{3,0} = \cos^3(\varphi)\Xi_{3,0} - 3\cos^2(\varphi)\sin(\varphi)\Xi_{2,1} + 3\cos(\varphi)\sin^2(\varphi)\Xi_{1,2} - \sin^3(\varphi)\Xi_{0,3}$$

などが成り立つ．これにより，任意の方向の微分フィルタなどが表現可能となる．

最後の特徴として N-Jet 成分同士は相関が大きい．成分同士の相関は標準的なパラメータ配列における標準基底においては直交するため 0 であり，本研究における OtX 成分系においてもほとんどの成分同士の相関は 0 である．本研究では主成分分析の結果との整合性を強化する目的で可能な限り直交性のある成分系の設計を試みたが，この性質は最適化を行う際にも有利にはたらくと考えられる．たとえば，$n \in \mathbb{N}_+$，$A \in \mathbb{R}^{n\times n}$，$b$，$x \in \mathbb{R}^n$ とする．線形方程式

$$Ax = b \tag{3.26}$$

の解は両辺の左側から A の逆行列 A^{-1} をかけることにより

$$x = A^{-1}b \tag{3.27}$$

と求められる．ここで，A が正規直交行列であるならば $A^{-1} = A^\top$ であるから単に

$$x = A^\top b \tag{3.28}$$

でよい．つまり，ターゲットとなるフィルタ b が確定している状態では線形結合の係数 x を求めるのに対応する成分と b の内積を計算すればよいことになる．このことから，本研究で提案する OtX 成分系は N-Jet 成分系よりも最適化が容易であると推測できる．

3.5 成分スケーリング

式 (3.3) を拡張し，

$$W = \sum_{\lambda=1}^{\Lambda} \alpha_\lambda(\theta_\lambda)F_\lambda \tag{3.29}$$

のような線形結合を考える．式 (3.3) においては $\alpha_\lambda(\theta_\lambda) = \theta_\lambda$ であったが，

$$\alpha_{\lambda,c_{\mathrm{in}},c_{\mathrm{out}}}(\theta_{\lambda,c_{\mathrm{in}},c_{\mathrm{out}}}) = \beta_\lambda\theta_{\lambda,c_{\mathrm{in}},c_{\mathrm{out}}} \tag{3.30}$$

のようにして，フィルタ成分の線形結合係数を求めることとする．ここで，入出力となる 3 次元の特徴マップにおけるチャンネルインデックス c_{in}，c_{out} を含めたのは，β_λ が畳み込み層内における

それぞれのフィルタ W 間で共有されることを強調するためである．この β_λ は成分の種類（Φ_0，Φ_2，$\Psi_{0,1}^{(0)}$ を区別する単位）のみに依存して決まる学習可能パラメータであり，

$$\beta_\lambda = \exp\left(-\frac{s_\lambda^2}{2}\right) \tag{3.31}$$

によって初期化を行う．ここで，s_λ は第 λ 成分の重要度を表すスコアであり，3.4.3 節において，必要な OtX 成分を選択する際に用いたものである．

この変更について，画像を処理するという観点においては等価であると言えるが，学習の容易化を期待することができる．OtX 成分系は学習済み畳み込みフィルタの主成分を近似モデル化したものであるから，それぞれの成分で標準偏差，つまり，持ちうる値のスケールが大きく異なることとなる．この値のスケールの差を成分の種類ごとに，ある程度よい形で初期化された β_λ のスケールの差に集約することで，フィルタごとに個別に用意された $\theta_{\lambda,c_{\mathrm{in}},c_{\mathrm{out}}}$ の学習負担を軽減する目的がある．

3.6 Weight Standardization の改変

本章の実験で扱う NFNet [89] においては Weight Standardization [90]（重みの標準化）に，学習可能なパラメータによる値のスケーリングを追加した Scaled Weight Standardization [91] が用いられる．Weight Standardization は，畳み込み層におけるフィルタの fan-in（入力マップのチャンネルまで考慮した受容野）におけるフィルタ値の平均を 0，標準偏差を 1 とするべく，

$$w'_{i,j,c_{\mathrm{in}},c_{\mathrm{out}}} = \frac{w_{i,j,c_{\mathrm{in}},c_{\mathrm{out}}} - \mu_{c_{\mathrm{out}}}}{\sigma_{c_{\mathrm{out}}}} \tag{3.32}$$

のような計算を行うものである．また，これに学習可能パラメータ $\gamma_{c_{\mathrm{out}}}$（学習可能パラメータ）を乗じた $\gamma_{c_{\mathrm{out}}} w'_{i,j,c_{\mathrm{in}},c_{\mathrm{out}}}$ を用いるのが Scaled Weight Standardization である．

しかしながら，本章の提案における OtX 成分系や既存手法の N-Jet の成分系はそれぞれ見てきたように $-\mu_{c_{\mathrm{out}}}$ のようなバイアスを加えないことを前提に設計されたものであるから，この（スケール化された）標準化処理を行うことは不適当であると考えられる．そこで，これらの 2 成分系に対しては平均値を 0 にすることはやめ，標準偏差の調整のみを行うこととした．

3.7 実験

3.7.1 実験条件

実験は CIFAR-100 データセット [92] における画像のクラス分類タスクによって行う．ある程度共通の条件下で訓練を行い，正解率に相当する Top-1 Accuracy によってそれぞれの方法を評価する．畳み込みフィルタの生成方法（成分系）のみを変え，深層学習モデルを訓練した結果を比較するため，推論結果が高いモデルほどよりよい畳み込みフィルタを獲得できたことになる．フィル

タ成分としては，学習パラメータの配列をそのまま用いる「標準」的な手法（潜在的に標準基底が仮定される）と本章で提案した「OtX」成分系のほかに，ガウス関数の高階導関数の積からなる「N-Jet」成分系 [1, 64] も比較する．なお，後者の 2 つについては相対座標に相当する x，y を学習可能パラメータ σ によって $\frac{x}{\sigma}$，$\frac{y}{\sigma}$ の形で空間的にスケーリングしたものを採用した．これは N-Jet-Net [82] の知見に沿った変更であるが，フィルタサイズ自体は標準的なものと同じまま不変とした．

比較は VGG-16 [2]，DenseNet-121 [93]，EfficientNetV2-S [94]，NFNet-F0 [89] の 4 つの分類モデルで行った．これらのモデルはオープンソースのモデル実装集 PyTorch Image Models [87]（TIMM）のものを用い，標準基底以外を用いる際には畳み込み層（1×1 を除く）のみ交換する形で実装した．VGG-16 は特徴マップ同士の加算や結合の演算を含まない，直鎖型のモデルの例として採用した．DenseNet-121 は特徴マップ同士の結合による分岐パスからなるモデルの例として採用した．EfficientNetV2-S ならびに NFNet-F0 は特徴マップ同士の加算による分岐パスをもつ ResNet [22] 系統のモデルのうち，新しいものとして採用した．EfficientNet [26] の後継モデルである EfficientNetV2-S は Batch Normalization [95] を含み，Weight Standardization [90] を含まない，近年の標準的かつ高水準な畳み込みニューラルネットワークということができ，NFNet-F0 は Scaled Weight Standardization [91] との相性を見るためにちょうどよいモデルであると言える．また，学習率（ピーク値）はモデルごとに変えており，標準基底で訓練した際に最も推論精度が高くなるものを設定した．

訓練用の画像に対しては Distorted Random Crop [2] と等確率のランダムな左右反転のみをデータ拡張として用い，そののち，bicubic 補間によって 192×192 に拡大した．また，0 から 255 の整数値からなる画素値は 255 で割って浮動小数点数にしたあと，データセットにおける R，G，B 各チャンネルの平均と標準偏差（表 4.5）を用いて標準化した．推論用の画像は bicubic 補間により 256×256 へ拡大したあと，224×224 となるようセンタークロップしたもの用い，224×224 を本実験における 1/1 スケールと定める．

最適化に関して，ミニバッチサイズを 32 に設定し，90 epoch の訓練を実施した．パラメータ調整に用いる Optimizer は標準的な Nesterov の加速勾配法 [40] つきの確率的勾配降下（SGD）Optimizer を採用し，momentum [41] は 0.9 とした．学習率は 5 epoch にわたる訓練ステップ単位の線形な warmup によって，0 から表 3.1 の学習率まで大きくしたあと，訓練終了時点で 0 になるよう cosine annealing により減衰させる．また，推論精度向上のため Sharpness-Aware Minimization（SAM） [47] のステップを 5 訓練ステップに 1 度行っており，勾配上昇には学習率 -0.05 の SGD を用いた．損失関数は Cross Entropy のみを用い，平滑化率 0.1 の Label Smoothing [53] によってラベルの平滑化を行った．

訓練のプログラムは機械学習用の Python フレームワークのひとつである PyTorch [96] を用いて実装した．また，すべての実験は NVIDIA GeForce RTX3090 GPU を搭載したグラフィックボード上で実行した．

実験は異なる乱数シードのもと各 3 回ずつ実施し，平均値をとることとした．なお，勾配爆発により訓練に失敗した場合は平均値の算出からは除外しており，3 回とも失敗した場合には「勾配爆

発により計測不可」と記載する．

3.7.2 ベースラインや N-Jet との比較

まずはメインの比較として標準基底，N-Jet 成分系，OtX 成分系をそれぞれ線形結合の成分に持つフィルタを用いた，各モデルに対する結果を示す．なお，N-Jet 成分系のベースラインとしては $m+n \leq 3$ なる 10 成分を用いた N-Jet-10 を，OtX 成分系のベースラインとしては×3 の主成分と考えられる上位 9 成分を用いた OtX-9 を用いた．これらは実際次節の比較においても推論精度のよい成分の数であることが示されている．

表 3.1: 畳み込みフィルタの成分系の違いによる訓練結果の違い

モデル	成分系	重み標準化	学習率	パラメータ数	Top-1 Accuracy（%）
VGG-16	標準基底	×	0.0125	134 M	72.7
VGG-16	N-Jet-10	×	0.0125	54.5 M	勾配爆発により計測不可
VGG-16	OtX-9	×	0.0125	50.8 M	73.4
DenseNet-121	標準基底	×	0.25	7.06 M	76.6
DenseNet-121	N-Jet-10	×	0.25	7.29 M	勾配爆発により計測不可
DenseNet-121	OtX-9	×	0.25	7.05 M	77.3
EfficientNetV2-S	標準基底	×	0.0625	20.3 M	79.3
EfficientNetV2-S	N-Jet-10	×	0.0625	20.4 M	勾配爆発により計測不可
EfficientNetV2-S	OtX-9	×	0.0625	20.3 M	勾配爆発により計測不可
NFNet-F0	標準基底	原典 [89] 通り	0.1875	68.7 M	81.7
NFNet-F0	N-Jet-10	3.6 節による	0.1875	70.7 M	82.3
NFNet-F0	OtX-9	3.6 節による	0.1875	68.7 M	82.2

結果を表 3.1 に示す．「重み標準化」は Weight Standardization を指し，NFNet-F0 における Scaled Weight Standardization のうち，値のスケーリング前の部分の違いについて表したものである．表の結果はそれぞれの手法において推論精度の高かった方を採用した．また，パラメータ数における M は 10^6 を示す Mega の略である．

正解率を表す Top-1 Accuracy に注目すると，「勾配爆発により計測不可」の事例が目立つ．学習率は標準基底にあわせて決定したため，当然ではあるが標準基底についてはすべて計測できている．N-Jet-10 においてはモデルが NFNet-F0 以外の場合においてすべて訓練に失敗している．OtX-9 においてはモデルが EfficientNetV2-S 以外の場合において計測できている．この性質は学習率のようなハイパーパラメータを複数回の実験で調整すれば改善されると考えられるが，OtX-9 は標準基底と同じハイパーパラメータにおいても N-Jet-10 と比較してより安定した訓練が可能であると言える．

次に，EfficientNetV2-S 以外における，標準基底と OtX-9 の比較を行う．OtX-9 は訓練が成功したすべての事例において標準基底を上回る推論精度を達成している．つまり，訓練は不安定にな

る可能性があるが，一切のハイパーパラメータ調整をせずとも，標準基底から OtX-9 に成分系を変更するだけでよりよい畳み込みフィルタを獲得可能であることを示唆している．ただし，ILSVRC2012 データセット [21] のような大規模なデータセット（本実験で用いた CIFAR-100 と比較してクラス数は 10 倍の 1000 クラス，訓練データ数は約 25 倍の 1,281,167 枚）においては，N-Jet を深層学習に用いることを提案した [64] における結果と同様，逆転する可能性がある．本研究においても ILSVRC2012 データセットにおける比較を試みたが，時間的な制約により断念した．

N-Jet-10 の訓練が成功した NFNet-F0 においては，N-Jet-10 と OtX-9 の結果がともに標準基底を上回っている．つまり，N-Jet-10 についても勾配が爆発しなければ同条件のもと標準基底を上回る推論精度を達成できる．なお，N-Jet-10（82.3%）がわずかに OtX-9（82.2%）を上回る推論精度となっているが，ほとんど同等といってよい．したがって，OtX-9 は N-Jet-10 よりも訓練の安定性において優位であり，また，ともに訓練が成功した事例においても N-Jet-10 とほぼ同等の推論精度を達成することができると言える．

また，学習可能なパラメータ数に注目すると，VGG-16 における標準基底のパラメータ数がほかの 2 成分系と比較して 2 倍以上になっていることが見受けられる．これは VGG-16 が 3.3 節で見た畳み込み層による処理の直後に得られる 7×7 の特徴マップを 1 次元化する処理において用いられる全結合な計算を，本実験で用いた TIMM による実装では 7×7 の畳み込みによって行っているためである．実際，標準基底においては $7 \times 7 \times 512 \times 4096$ の配列を用いることで 102.8 M のパラメータを要するが，N-Jet-10 においては $10 \times 512 \times 4096$ により 21.0 M，OtX-9 においては $9 \times 512 \times 4096$ により 18.9 M のみでよく，その他の層で 1 フィルタあたりのパラメータ数がともに 9 である標準基底と OtX-9（厳密には各層において空間スケール用の 1 パラメータ σ と，成分スケーリング用の 9 パラメータ $(\beta_\lambda)_{\lambda=1}^{9}$ を含むため異なるが，無視できる程度に小さい）を比較することにより $50.8\mathrm{M} + (102.8\mathrm{M} - 18.9\mathrm{M}) = 134.7\mathrm{M}$ となるから，この差が，この全結合計算の実装の差によって生じたものであると言える．クラス分類器におけるこういった 1 次元化の処理は最近のものでは全結合のものを用いないため，この点における優位性はいえないが，フィルタサイズの大きい畳み込みにおいてパラメータ数に大きな差が生じることを定量的に示すひとつの例である．N-Jet-10 ならびに OtX-9 はフィルタサイズによらず，1 フィルタあたりのパラメータ数をそれぞれ 10 と 9 の一定値にすることができる．これらの成分がその畳み込みにおいて果たすべき役割を行うためのフィルタを十分に表現あるいは近似可能であれば，より頑健なフィルタを得られる可能性がある．

その他のモデルにおいても，3×3 フィルタに 9 パラメータを用いる標準基底と比較し，1 フィルタに必要とするパラメータ数が約 9 であるような OtX-9 はほとんど同じパラメータ数であり，約 10 であるような N-Jet-10 ではやや上回る形となっている．この 2 成分系については次節で述べるが，標準基底と比較して，パラメータの多さにより高精度を実現したわけではないことが言える．つまり，これらの 2 成分系は標準基底と比較して訓練に不安定性をともなうが，訓練に成功した場合においてはよりよいフィルタを見つけることができると言える．また，N-Jet-10 と OtX-9 の両方において用いた空間的スケールのためのパラメータ σ や，OtX 成分系のために本章で提案した成分スケーリングのためのパラメータ $(\beta_\lambda)_{\lambda=1}^{\Lambda}$ が，各フィルタ独立ではなく，層内で共有されてい

るため無視できるほどに小さいことも同時に示されている．

3.7.3 成分の効率

N-Jet 成分系 [1, 64] や，本章で提案した OtX 成分系においては，1 フィルタあたりに用いる成分の数をハイパーパラメータとして決めることができる．そこで，本節ではその成分の数を変化させた際の比較を行う．これにより，成分数を増やした際におけるフィルタの表現力の向上と，それに伴って増えるパラメータ調整の複雑化についてのトレードオフを見ることができる．なお，実験では先の比較において N-Jet-10 と OtX-9 がともに訓練に成功した NFNet-F0 を用いている．

表 3.2: 用いる成分数を変化させた際の比較

成分系	パラメータ数	Top-1 Accuracy（%）
N-Jet-6	62.9 M	79.1
N-Jet-10	70.7 M	82.3
N-Jet-15	80.4 M	80.1
OtX-6	62.9 M	82.2
OtX-8	66.8 M	82.2
OtX-9	68.7 M	82.2

結果を表 3.2 に示す．それぞれの成分系の名前の末尾にある数値は成分系を構成する成分数を表している．

まず，N-Jet について見ていくと，N-Jet-10 が最も高精度となっている．成分数が増えるほど表現可能なフィルタは多くなるため，N-Jet-6 についてはフィルタの表現力が不足している可能性が考えられる．しかしながら，N-Jet-10 よりも表現力が高いはずの N-Jet-15 においては N-Jet-10 よりも低い推論精度となっており，うまくパラメータ調整ができていないことがうかがえる．ただし，NFNet-F0 における畳み込みフィルタは 3×3 が主であるため，成分数を増やしても表現力が上がらない可能性があり，無駄なパラメータが増えたために最適化が困難になっただけであるとの見方もできる．

一方の OtX 成分系においては，OtX-6 が 6 成分ながら OtX-9 と同じ精度を達成していることが示されている．これは OtX が畳み込みフィルタの主成分を近似モデル化したものであるため，推論時に必要なフィルタを十分に表現できているものと考えられる．同じ成分数の N-Jet-6 と比較しても高い精度を実現しており，OtX 成分系の設計意図の通り，少ない成分で畳み込みフィルタを効率よく近似できていることがわかる．また，OtX-6 の 6 成分で十分であるにもかかわらず，さらに成分を増やしてパラメータ数が増加した OtX-8 や OtX-9 の精度が N-Jet-10 から N-Jet-15 の変化のように低下していないのも特徴的である．

なお，OtX-6 を用いるのが最良であるようにも見えるが，OtX-9 は OtX-6 に対し 2 つの利点がある．第一の利点は上振れた際の精度である．それぞれについて 3 回の試行において最高の精度を比較すると，OtX-9 は 82.7%，OtX-6 は 82.4%であった．これは，成分数が多い場合でも，よりよい

フィルタのための係数の探索が成功することがあることを示している．第二の利点は，訓練の安定性である．OtX-6 は 3 回の実験のうち 1 回勾配爆発が発生したが，OtX-9 は 3 回とも安定して訓練できた．したがって，精度の上振れを狙った複数回の試行によってよりよいフィルタを得たい場合や，訓練の失敗を避けたい場合には，OtX-9 の方が適していると言える．

3.7.4 N-Jet と OtX の性質・条件を入れ替えた際の比較

N-Jet と OtX のどのような違いが性能に寄与しているかを明らかにするために，N-Jet と OtX の性質・条件を入れ替えた際の比較実験を行った．N-Jet に対する OtX の主な特徴は 3 つあり，

1. フィルタ成分の新しい対称性（球対称成分 Φ_n や対角線を軸とした線対称成分 $\Psi_{m,n}^{(\frac{\pi}{4})}$）
2. 成分同士の相関性（相関の大きい $\xi_n \propto H_n(x)\exp(-x^2)$ か，相関の小さい $\psi_n = H_n(x)\exp\left(-\dfrac{x^2}{2}\right)$ か）
3. 成分スケーリング（3.5 節）の適用

が挙げられる．各特性の有効性を検証するため，これら 3 つを適用するかしないかについてのすべての組み合わせについて実験を行った．前節と同様，NFNet-F0 を比較に用いている．なお，成分スケーリングを N-Jet-10 に適用するにあたって，各成分 $\Xi_{m,n}$ のスケール β_λ を初期化するためのパラメータ s_λ としては $m+n$ を採用した．

表 3.3: N-Jet と OtX の性質・条件を入れ替えた際の比較

事例	成分系	相関性	成分スケーリング	Top-1 Accuracy（%）
1	N-Jet-10	低	✓	81.7
2	N-Jet-10	低		82.2
3	N-Jet-10	高	✓	勾配爆発により計測不可
4	**N-Jet-10**	高		82.3
5	**OtX-9**	低	✓	82.2
6	OtX-9	低		勾配爆発により計測不可
7	OtX-9	高	✓	82.1
8	OtX-9	高		81.2

結果を表 3.3 に示す．表中において，成分系が太字で書かれている行（事例 4，事例 5）は N-Jet-10，OtX-9 における標準的な設定条件であり，これまで見てきた N-Jet-10，OtX-9 はこれら条件のもと実験・比較されてきたものである．

まず，「勾配爆発により計測不可」と記載のある事例 3，事例 6 に注目する．これらは太字の標準的な条件と比較した場合，N-Jet-10 で成分スケーリングを実施し，OtX-9 において成分スケーリングを実施しなかった例である．OtX 成分系において成分スケーリングを導入したのは，畳み込みフィルタにおける主成分の標準偏差の差から生じると考えられた，各フィルタ成分の係数パラメータの調整にかかる困難さの回避のためであった．これらの結果はこの導入意図をうまく反映し

たものとみることができる．つまり，フィルタの主成分を模したものではない N-Jet-10 の成分にとっては余計なスケーリングであり，フィルタの主成分を近似モデル化した OtX-9 には不可欠なスケーリングだったことが推察される．

次に，太字の標準的な条件と比較して直交性，つまり ξ_n と ψ_n を入れ替えて実験を行った事例 2，事例 7 に注目する．少なくとも 3×3 のフィルタにおいて，相関性の大小はそれぞれに影響を与えないものと思われる．これは特徴マップの局所的 Taylor 展開や主成分の近似モデル化といったそれぞれの設計意図とは異なった結果と言える．

最後に，成分系のみを入れ替えた事例 1，事例 8 を見ると，それぞれ事例 5，事例 4 と比較した際に劣化が見られる．しかしながら，事例 3，事例 6 で発生した勾配爆発は起こっていない．したがって，それぞれの成分系の相関性を入れ替えることにより，成分スケーリングとの相性の悪さが軽減されたと考えられる．ただし，相性が悪いことには変わりないため，事例 2 や事例 7 と比較しても劣化していることが見てとれる．

以上のことから，OtX 成分系が目指した直交性（相関が 0）や，OtX 成分系の訓練のために導入した成分スケーリングといった性質は，成分系のもつ成分の対称性との相性が顕著に表れる．つまり，連続関数を用いた成分系において，これらの性質・訓練方法は必須ではなく，理論的な考察に基づいて適用すべきかどうかを判断すべきであると言える．

3.8 第 3 章のまとめ

本章では畳み込みフィルタを線形結合で効率的に表現するための成分系として，フィルタの主成分を用いることを提案した．ILSVRC2012 データセット [21] によって学習済みの VGG-16 [2] の畳み込みフィルタの主成分を分析した結果，層によらず，なんらかの法則に従った成分が抽出されたため，これを近似モデル化するための準備として主成分分析の結果を観察ならびに考察した．これによって得られた知見から，重み付き直交性を持つ Hermite 多項式に注目し，3 種類のフィルタからなる OtX 成分系を定義した．球対称成分同士が直交性を持たないことを除き，観察結果に忠実な近似モデル化を実現したと言える．また，実際の深層学習における畳み込みフィルタの成分として利用するために，用いる成分の選定方法や離散化の方法について述べた．提案する成分の選定方法により，観察した実際の主成分と同じ並びのものが得られることを確認した．その後，同じく Hermite 多項式を利用した N-Jet 成分系の設計意図や提案した OtX 成分系の違いについて述べた．式の形は似ているが，前者が特徴マップの局所的 Taylor 展開を意図しており，成分同士が直交性を持たないことなどを強調した．

さらに，OtX 成分系を訓練するにあたり，成分同士の重要度の差によってパラメータ調整が困難になると考えられたことから，各成分の重要性を肩代わりするパラメータを導入する成分スケーリングを提案した．また，N-Jet や OtX といった設計背景を持つ成分系に対して Weight Standardization [90] をそのまま適用することの不適切さを指摘し，代替案として，平均値を差し引かないことを提案した．

実験においては，同一の畳み込みニューラルネットワークモデルにおけるフィルタの成分系のみ

を変更して訓練を行い，推論精度を比較した．これにより，その成分系がよりよいフィルタを見つけるのに役に立つのかどうかを調べることを意図している．標準基底を用いたモデルは訓練結果が安定しており，ハイパーパラメータを調整しなければ，N-Jet や OtX は勾配爆発を起こしうることを確認した．つまり，N-Jet 成分系や OtX 成分系による訓練が可能かはモデル次第である．ただし，OtX 成分系のほうがより多くのモデルに適用可能である．N-Jet 成分系も OtX 成分系も訓練が完了した際には標準基底を上回る推論精度を実現しており，特に OtX 成分系は比較的安定して推論精度を向上させることができた．また，N-Jet 成分系は 6 成分のみでは 10 成分の場合と比較して劣化を引き起こしたが，OtX 成分系では 6 種類の成分を用いるだけでも十分に必要なフィルタを表現できることを示した．つまり，少ない成分で効率よくフィルタを表現できることを期待した，主成分分析の近似モデル化の意図に沿った成分系となっていると言える．さらに，直交性を持った成分を用いることや，成分スケーリングを行うことについて，N-Jet 成分系，OtX 成分系のそれぞれとの相性があることが判明した．特に，成分間に重要度の差があると仮定した成分スケーリングについては，主成分を模している OtX 成分系にとっては必須ながらも，そうではない N-Jet 成分系においては訓練を阻害する要因とすることがわかった．

以上を総括すると，OtX 成分系は標準基底と異なり，適用できるモデルは限定されるが，従来の N-Jet 成分系よりも多くのモデルに適用可能である．また，適用可能なモデルにおいては畳み込みフィルタの主成分を近似モデル化した意図の通り，フィルタサイズに依存しない少数のパラメータで標準基底を上回る精度のモデルを獲得することができる．

第 4 章　画像サイズに合わせたフィルタ生成と縮小画像による訓練

畳み込みは縦横の要素数に比例した計算コストが発生する．近年注目されているフィルタサイズの大きい畳み込みは広範囲の情報を集約することができる一方，計算コストが膨大になる欠点がある．

通常，深層学習モデルではフィルタサイズを不変のものとするため，フィルタと画像サイズの比は入力する画像をリサイズすることで一定に保つが，本章では逆に画像に合わせてフィルタサイズを拡大縮小する方法を議論する．フィルタサイズが大きい畳み込みについて入力画像とフィルタを縮小して訓練することで，訓練時の計算コストを小さくすることができる．

なお，フィルタの生成と直接的に関連するものではないが，小さい画像と小さいフィルタを用いた低コストな訓練をよりよいものとするため，訓練にあたって有効な 4 つの工夫を提案する．これを導入することにより，標準的な訓練方法や先行研究の Progressive Learning よりも短時間の訓練で遜色ないモデルを獲得することが可能となる．

4.1　関連研究

4.1.1　小さい画像を用いた訓練に関する先行研究

小さい画像を用いて深層学習モデルの低コストな訓練を行うという発想はいたって単純なものであるが，意外なことにほとんど前例が見られない．こうした背景には，訓練の高速化が研究・開発をする立場の者に恩恵をもたらすものの，訓練済みの深層学習モデルを利用する立場の者にとっては無関係のことがらであることや本格的に取り組むためには本研究のようにフィルタの生成に着手する必要がある難しい問題であることが影響していると考えられる．

EfficientNetV2 [94] の提案論文内で報告された Progressive Learning は，訓練に用いる画像のサイズを徐々に大きくするというものである．画像サイズの変更に伴うモデルの変更はなく，当然のことながらフィルタサイズは訓練を通して不変である．極めて単純な方法ではあるが，学習の高速化のみならず，最初から大きい画像で訓練を行うよりも精度が改善される．

具体的に見ていくと，EfficientNetV2 [94] の中規模モデル EfficientNetV2-M はテスト時に 480×480 の画像を用いており [87]，その訓練のために 128×128 から 380×380 の画像を用いる．ここで，$380 < 480$ であることが「小さい画像を用いた訓練」ではない．これはテスト時に画像をやや拡大するという慣例によるもので，訓練時に画像をランダムクロップしてから拡大するため，深層学習モデルが訓練データよりもやや拡大された画像の処理に最適化されるためであると思われる．ILSVRC2012 データセットのクラス分類タスクの 350epoch にわたる訓練を 4 分割した各区間において，画像サイズを線形に変えながら訓練を行う．すなわち，1 から 87epoch 目までは

128 × 128，88 から 174epoch 目までは 212 × 212，175 から 261epoch 目までは 296 × 296，そして 262 から最終 350epoch 目までは 380 × 380 の画像を用いて訓練する．このとき，訓練画像の画素数の平均は約 270^2 であり，画像に対する処理が画素数に比例するとするならば，2.0 倍の高速化が期待される．また，訓練する画像のうち最後の 1/4 は最大サイズのものを用いるため，訓練の高速化の上限として 4 倍という水準が考えられる．

実際の訓練速度について，論文においては，EfficentNetV2-M と同等程度の精度を誇る Efficentnet-B7 [26] と比較して訓練時間が 11 倍高速になったことが強調されているが，この Progressive Learning 単体で見た場合の高速化の度合いは言及されていない．そこで，推論時間がほとんど同じ EfficientNet-B5 と比較すると EfficientNet-B5 の訓練が 43 時間を要したのに対し，EfficientNetV2-M の訓練時間は 13 時間であることから，約 3.3 倍程度の高速化がなされていると推測される．また，ResNet50，ResNet152 [22]，EfficientNet-B4，EfficientNet-B5 に対して Progressive Learning を適用した場合においてはその高速化の程度が示されており，29%から 65%の時間短縮（すなわち 1.4 から 2.9 倍の高速化）が報告されている．

以上より，Progressive Learning は多く見積もっても 4 倍，実際は 1.4 から 2.9 倍程度の高速化が期待されるが，小さい画像と大きい画像とで同じ結果が出力されるような深層学習モデルが存在すればさらなる高速化を期待することができる．本章の実験においては Progressive Learning で用いられた 1/3 よりもさらに小さい，1/4 程度のサイズの画像を中心に訓練し，また，推論時の画像と同程度のサイズの画像は 1/64 しか用いない．さらに，画像に比例してフィルタサイズも小さくなるため，小さい画像を用いることによる高速化の程度は Progressive Learning のようにモデルを画像のサイズに合わせて変更しない場合よりも大きくなる．このようにして，さらなる訓練の高速化を図るというのが小さい画像を用いた訓練に関する研究としての本研究の差別化点となる．

4.1.2 スケール等価性に関する研究

画像処理用の深層学習モデルが異なるサイズの画像を処理する際に，まったく同じ結果を得たい場合がある．たとえば，画像のクラス分類問題はある程度小さいサイズの画像までなら同じクラスであると認識されるべきである．こういった状況で深層学習モデルに求められるのがスケール不変性（scale-invariance）である．また，まったく同一でなくても，統一的な変換によって同じ結果が得られればよい場合にはこれを緩和してスケール等価性（scale-equivariance）という性質を求める．この「統一的な変換」は「何もしない変換」でもよいため，スケール不変性はスケール等価性に含まれる．

深層学習モデルのスケール等価性を追求する研究 [97–100] では基本的にモデル単位で設計を行う．これは本研究が畳み込みというプリミティブなモジュールに対してなされているのとは対照的である．なお，スケール等価なモデルを設計する際には「大きさが異なる同じフィルタ」を表現するために，第 3 章において比較対象とした N-Jet [64] 系統の線形結合系生成フィルタが用いられる．これは N-Jet がこれまで相対座標をフィルタ重みに変換するためのほとんど唯一の成分系であったことに由来するが，本章の研究により「大きさが異なる同じフィルタ」の一般系についての提案を行う．したがって，本章の研究はスケール等価性に関する研究に応用可能であると考えられる．

4.1.3 大きいサイズの畳み込みフィルタの利用に関する研究

少なくとも本論文の範囲では，一般に広く用いられる 3×3 の畳み込みフィルタを小さくすることができない．また，訓練に用いる画像のサイズを小さくし，それに合わせて畳み込みフィルタを小さくすることで訓練の低コスト化を図るという応用を前提としているため，結果として 3×3 の畳み込みフィルタは拡大するなどしてモデルから排する必要がある．つまり，本章の提案事項が実用されるには「大きいサイズの畳み込みフィルタ」が役に立つことが望ましい．

畳み込みニューラルネットワークの全盛を生み出すきっかけとなった AlexNet [56] においては 224×224 の画像入力に対し，最初の層で 11×11 のフィルタを畳み込んでいた．しかしながら，VGG[1)] の提案論文 [2] において，「3×3 の畳み込みを 2 回用いれば受容野は 5×5 に，3 回用いれば 7×7 へと広がっていく」という主張がなされたためか，以降の畳み込みとしては 3×3 のものが主流となった．実際，3×3 の畳み込みを行うコストを 9 とすると，7×7 の畳み込みを行うコストは 49 であり，3×3 の畳み込みを 3 回行ってもコストは 27 であるから，（勾配逆伝播のためのメモリの消費量を無視して）計算量の観点で見れば効率的である．そのうえ，非線形の処理も間に 2 回入るため，より複雑な処理を実現することができる．画像内における遠い位置を畳み込みで参照する方法として，櫛状のフィルタでまばらに特徴抽出を行う Dilated Convolution [60] が知られるようになり，大きいサイズの畳み込みフィルタを用いるモチベーションはさらになくなったと言える．それ以降，水平方向と鉛直方向の 2 軸の畳み込みを並列に行うことで，大きな受容野を持つ畳み込みの実現を試みる計算モジュール（7×1 と 1×7 のフィルタを用いる Inception-v4 [65] や 15×1 と 1×15 のフィルタを用いる Gloval Convolutional Network [66]）が提案されたが，主流になることはなかった．

この流れが大きく変わったのは，Vision Transformer [16] のような Transformer [18] ベースのモデルが画像処理に使われるようになってからである．部分的に Self-Attention [18] のような計算を用いる Non-local Neural Network [101] などは存在したが，大きい受容野の処理が 3×3 の畳み込みの繰り返しにより実現されるという仮説に疑問が持たれ始める．多層パーセプトロン（Multi-Layer Perceptron；MLP）によって画像全体を受容野とする線形変換ベースの計算モジュール [19] も提案され，モデル全体ではなく，モジュール単位で広範囲から特徴抽出をする方法に注目が集まった．

なお，3×3 による畳み込みの繰り返しによる方法では理論値よりもいくぶん小さい受容野となっていることが Effective Receptive Field [102]（実効受容野；ERF）という手法によって示された．これは，クラス分類器の処理の終盤，1 次元化をする直前において，特徴マップの中心の値の絶対値の合計を損失関数とした勾配を逆伝播し，入力画像における勾配マップから，これを近似する Gauss 分布の標準偏差を調べるという方法である．なお，同論文内においては受容野の大きさと推論精度の向上にはあまり関係がないことも主張されている．

Swin Transformer [25] では 7×7 の窓を用いて局所的な Self-Attention を行うことで高精度な画像処理を実現しており，その影響を受けた ConvNeXt [67] もその影響を受けて 7×7 の畳み込みを

1） これは Visual Geometry Group というチーム名の略称であるがモデル名として知られるようになった．

採用している．この流れを受けて，RepLKNet [29] では，フィルタサイズを最大 31×31 まで拡大した畳み込みを用い，ベースラインと比較して推論精度が向上すること，ならびに，実効受容野が実際に大きくなることを示した．なお，本章の実験におけるベースラインモデルはこの RepLKNet から訓練補助のための小受容野のフィルタを含む並列ブランチを削除したものである．31×31 などのサイズの大きい畳み込みは Tensorflow [103] や PyTorch [96] といったメジャーな機械学習フレームワークにおいて高速化のためのアルゴリズムが実装されておらず，RepLKNet の原論文においては MegEngine [104] というフレームワークを用いて高速化を図っている．本章で提案する小さいサイズの画像を訓練に用いるアプローチでは画像の縮小に合わせてフィルタサイズも小さくするため，訓練過程におけるこの欠点を軽減，あるいは克服するものである．

その他，畳み込みベースのさらなる受容野拡大を図る研究として SLaK [68] では 51×5，5×51，5×5 のフィルタを並列に用いたスパースな畳み込みを提案しており，Transformer 系のモデルにおいても大きいサイズの畳み込みの利用が検討されている．CrossFormer [69] では，入力画像に対する最初の処理であるパッチ化に際して最大 32×32 の畳み込みを用いており，これが推論精度の向上に寄与したことを報告している．また，Transformer では各トークンの位置情報を表すために，画像内における座標から算出した値を加算する Positional Encoding が行われるが，この値を特徴マップから畳み込みによって得ようとする Conditional Position Encoding [70]（CPE）がある．CPE では 27×27 といった大きいサイズの畳み込みフィルタを用いることで推論精度が向上することが報告されており，その理由として，特徴マップのサイズを保存するために行われるゼロパディングにより，端に近いほど 0 に近い値を持つようになるからではないかと考察されている．

本章で提案する訓練フレームワークはフィルタサイズの大きい畳み込みを用いなければならないが，受容野の大きい畳み込みは近年の研究からその有効性を示されており，推論精度を希求するうえでもよい方向にはたらく可能性がある．また逆に，こうしたフィルタサイズの大きい畳み込みが持つ，計算コストが高いという欠点を本章の提案により克服することができるため，近年の研究の発展を促すものであるとも言える．なお，この計算コストを低くするという点については，訓練中はもちろんのこと，推論精度をある程度犠牲にする必要はあるが，推論時においてもいうことができる．このことも実験によって示す．

4.2 画像サイズの変更に合わせた畳み込みフィルタの変換方法の提案

ここからは，小さいサイズの画像を用いることで訓練の低コスト化を図るという応用を念頭に，画像サイズに合わせて畳み込みフィルタを生成する具体的な方法について考察，提案を行う．まずは，訓練方法についての提案概要を述べたのち，画像に合わせた畳み込みフィルタに求められる条件を明確化する．そののちに，固定サイズの畳み込みフィルタではこれを実現できないことを示し，また，要件を満たすための変換方法について理論的な考察を行う．その過程で得られた知見をもとに，本章の実験で用いた具体的な生成の処理の流れを説明する．

4.2.1　小さいサイズの画像とフィルタの生成を用いた訓練フレームワーク

深層学習モデルを用いた画像処理では基本的に特徴マップのすべての空間的な位置に対して平等に同じ処理を行う．これにともなって，モデルにおける各計算は特徴マップのサイズ，つまり，空間方向の要素数に比例した数だけ行われるものが多い．本研究の題材となっている畳み込みはその代表例であり，ほかには活性化層による処理や，特徴マップ同士の要素ごとの足し算やかけ算などが挙げられる．

いま，$D \times D$ の画像を縦横それぞれ 2 倍にした $2D \times 2D$ の画像を考えれば，空間方向の要素数は 4 倍となるため，計算コスト（処理に必要な演算数）もおよそ 4 倍になる．逆に，縦横それぞれ 1/2 にした場合，計算コストが 1/4 程度になることが期待できる．これは同じ時間で 4 倍の数の画像を処理できることを意味し，膨大な数の画像を処理しなければならない場面において有効である．

深層学習においては，訓練に際してターゲットとなる画像サイズが（潜在的な場合も含め）定められる．これを処理するために訓練は同程度の画像サイズで行われるのが標準的である．しかしながら，先に述べたように，処理する画像のサイズを小さくすることで計算コストを下げることができるため，より小さい画像で訓練をすることができれば，そのぶん訓練時間を短縮することができる．上記の「ターゲットとなる画像サイズ」をターゲットスケール，あるいは，1/1 スケールと呼ぶこととすると，縦横がそれぞれ半分になった 1/2 スケールの画像や，さらにその半分の 1/4 スケールの画像を用いて訓練することで訓練の高速化を図るというのが本章において提案する訓練フレームワークの意図するところである．本章ではこのシンプルなアイデアを実現するための考察や具体的な提案，ならびに，実験を行う．

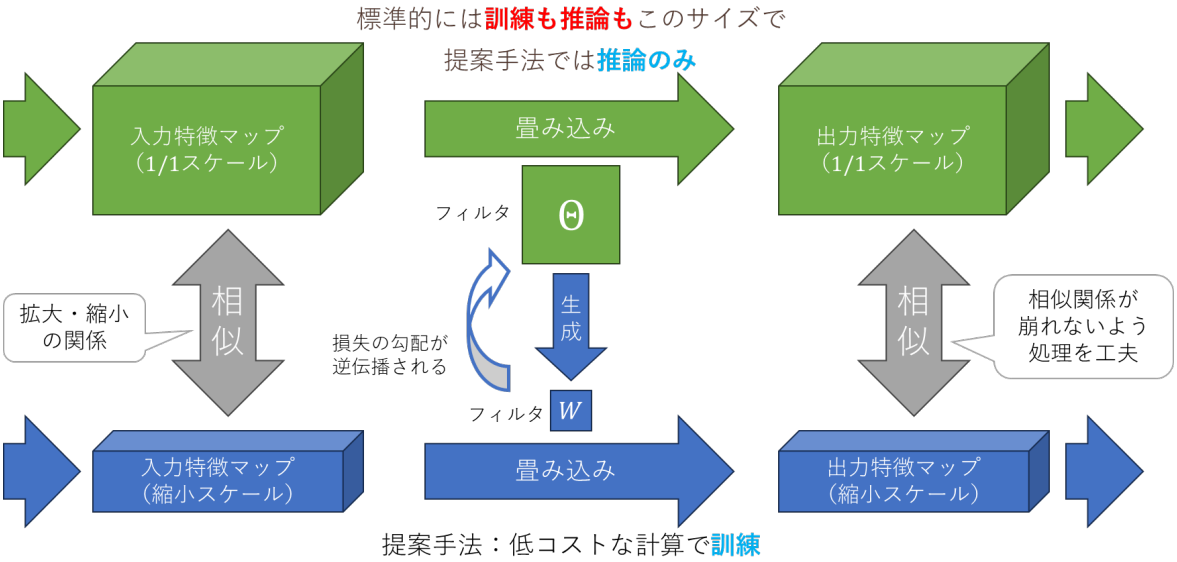


図 4.1: 本章の提案する訓練フレームワークの概要

提案の概要を図 4.1 に示す．ここでは，推論を行う際の画像サイズである，ターゲットスケール

の画像を処理する際に関連するものを緑色で示している．標準的なフレームワークにおいては訓練時にも推論時にもこの緑色のパスにおける処理が行われる．一方，下に示した青いパスはターゲットスケールよりも小さい，たとえば 1/4 スケールや 1/2 スケールの画像を同じ深層学習モデルで処理を表している．先に述べたように，特徴マップの処理はマップサイズに比例した計算量のものが多いため，青いパスにおける計算コストは緑のパスよりも小さく，程度としては 1/16 や 1/4 など，大幅な低コスト化になることが期待される．そこで，この青いパスの処理によって深層学習モデルを訓練しようというのが，本章の提案する訓練フレームワークのアイデアである．これにより，訓練時にかかる計算コストを下げ，訓練の高速化を図る．訓練時のメリットは学習済みモデルの利用者に直接影響を与えるものではないが，訓練を高速化することができれば，実験結果が出るまでの時間が短くなり，同じ時間で多くのことを試せるようになるため，研究・開発が促進されると考えられる．

もう少し具体的に見ていくと，青いパスによって訓練を行うということは，緑のパスにおける訓練を青いパスの処理によって代替するということである．したがって，青いパスにおいては緑のパスにおける処理を十分にエミュレートできていることが望ましい．つまり，図中の直方体の形で示された特徴マップについて，各処理における入出力がともに「相似」な関係であれば，この目的を達することができると考えられる．この「相似」については 4.2.2 節にて説明するが，およそ何らかの方法でリサイズした際に同じような見た目をしている状態を指す．よって，本章の研究においてはターゲットスケールより小さい訓練スケールにおいて，「入力が相似であれば出力も相似となるような処理」を繰り返すことでターゲットスケールと相似なモデル出力の獲得を目指す．その相似な出力に対して計算される損失を用いたパラメータ調整によってターゲットスケールにおける訓練の代替を図る．なお，ターゲットスケールに用いられるであろう学習可能なパラメータはそのまま，あるいは適宜微分可能な計算を通じて訓練スケールにおける処理に用いられる．その結果，訓練スケールの出力をもとに計算された損失関数の勾配が逆伝播され，訓練に利用される．

ここからは深層学習モデルにおける各計算モジュール単位で入力が相似な時に出力の相似性が担保されるかを見ていくことになるが，活性化層の計算や，特徴マップ同士の演算，スカラー倍や定数を加えるといった演算は特徴マップの各要素に対する独立な演算であるから出力の相似性をある程度精度よく担保するものと考えられる．また，1×1 の畳み込みについては他の位置の値を参照しないものであるから，これも同様に相似性を担保するものとして扱うこととする．

問題となるのは，1×1 以外の畳み込みである．これはたとえば，56×56 の特徴マップにおける 3×3 の領域とそれを 1/4 した 14×14 の特徴マップにおける 3×3 の領域では意味合いが大きく異なることが直感的に予想される．入力する特徴マップが相似であっても，畳み込みの際に参照する領域がスケール間で異なれば，それぞれで異なる値を用いて計算することになるから，一般に結果は相似にならない．少なくとも最低限，計算で参照する領域（受容野）はそれぞれの特徴マップの「同じ」部分でなければならない．以下の 2 節では上の抽象的な説明に関連する定式化とそれを用いた解釈を試みる．

4.2.2　特徴マップの相似

小さいサイズの画像を用いた訓練を論じるための準備として特徴マップの相似の概念を導入する．これはリサイズした画像がデータの上では配列の要素数の点で大きく異なりながらも見た目としてはほとんど同じ画像として感じられることをいくつかの用語を導入しながら説明するだけであるが，以降の説明の便宜上ここで導入する語を用いる．ただし，本節で導入する特徴マップの相似は平行移動や回転，反転を認めないため，幾何学における相似よりもかなり狭義なものである．

画像や特徴マップ上の空間的位置を指し示すパラメータとしては配列のインデックスを用いる場合が多い．しかしながら，これは異なる大きさの画像同士を比較することには向かないため，「正規化された空間座標」を導入する．これは画像の左上を $(0,0)$，右上を $(0,1)$，左下を $(1,0)$，そして右下を $(1,1)$ などと定義し，線形に座標をとったものを指す．これにより，画像の大きさが異なっている際に「感覚的に同じ位置」をひとつの正規化された空間座標によって表すことができる．本節では y を第 1 の正規化座標，x を第 2 の正規化座標とする．

いま，チャンネル数 C の特徴マップ U，V のあるチャンネル c に注目した u，v を考え，なんらかの補間方法 Ip により $w \in \{u, v\}$ に対する $\mathrm{Ip}(w) : [0,1]^2 \to \mathbb{R}$ が定まるものとする．ここで正規化された空間座標領域上における $\mathrm{Ip}(u)$ と $\mathrm{Ip}(v)$ の差の L_1-ノルム

$$d(u,v) = \int_0^1 \int_0^1 |\mathrm{Ip}(u)(y,x) - \mathrm{Ip}(v)(y,x)| \ \mathrm{d}y \, \mathrm{d}x \tag{4.1}$$

を u と v の距離 $d(u,v)$ と定義する．このとき，$\varepsilon > 0$ により $d(u,v) < \varepsilon$ が言えるならば，u と v は補間方法 Ip のもと ε-相似であるということとする．

ここで重要なのは相似を定義するための具体的な Ip や許容可能な ε の上限を議論することではなく $d(u,v)$ が小さいほど厳しい相似条件が満たされることである．直感的に肯定されるように特徴マップ同士が相似であるとはそれぞれの位置でだいたい同じ値を持っていることを指し，近ければ近いほど相似であると言える．この「位置」を指すものとしては正規化された空間座標が妥当であり，「だいたい同じ値を持っている」ことは $d(u,v)$ によって定量化されうるというのがあいまいな相似概念に対してひとつの具体的な記述方法を与える．

4.2.3　正規化された空間座標と畳み込みフィルタのサイズ

4.2.2 節で導入した正規化された空間座標の概念を用いて $k \times k$ の畳み込みフィルタ W を畳み込んだときの受容野について考える．この議論を通して固定長 $k \times k$ のフィルタを大きさは異なるが相似な 2 つの特徴マップにそれぞれ畳み込んだ際に相似な出力が期待できないことを示す．また，相似な出力を期待するための必要条件として畳み込みフィルタのサイズを画像サイズに比例して変化させるべきであることを示す．

いま，特徴マップ U のサイズが有理数 $p > 1$ を用いて $pk \times pk$ と書ける場合を考える．このとき，特徴マップの正規化された空間座標を考えるには配列のインデックスの値を下端や右端の座標とみなし，pk で割ったものを線形に補間して考えればよい．畳み込みフィルタ W の中心が正規化された空間座標上 (y,x) の位置にある場合，W が覆う範囲（受容

野）は $\left[y-\frac{1}{2p},\, y+\frac{1}{2p}\right]\times\left[x-\frac{1}{2p},\, x+\frac{1}{2p}\right]$ となる．同様に $q\neq p$ なる有理数 $q>1$ を用いて U と相似な $qk\times qk$ の特徴マップ U' 上で W の中心が (y,x) にある場合の受容野を考えると $\left[y-\frac{1}{2q},\, y+\frac{1}{2q}\right]\times\left[x-\frac{1}{2q},\, x+\frac{1}{2q}\right]$ となる．

正規化された空間座標は画像上の「同じ位置」を指し示すために導入された概念であるから，当然この座標が異なれば「違う位置」を指し示していることになる．いま $p\neq q$ を仮定しているため，U と U' では W の覆う範囲が異なる．つまり，U に W を畳み込んだ場合と U' に W を畳み込んだ場合とではそれぞれの画像上の「異なる範囲」を参照して計算が行われることになり，一般に相似な出力を期待することはできない．

実際に同一の畳み込みフィルタをさまざまなサイズの相似画像に畳み込んだ例は表 4.1 に見られる．表 4.1 の右 5 列はそれぞれ異なるサイズの畳み込みフィルタを計 4 種のサイズの相似画像に畳み込んだ結果の画像を表示しており，これを縦に見比べることで同一の畳み込みフィルタを異なるサイズの相似画像に畳み込んだ結果を比較することができる．例として $k=17$ の場合に注目して比べてみると，256×256 の画像に対する結果では蝶であることを認識可能な程度のぼやけかたであるが，画像のサイズが小さくなるほどぼやけの程度が大きくなっていき，32×32 の画像に対する結果では被写体が何かを判断することが困難なほどにぼやけている．正規化された空間座標において 17×17 のフィルタの受容野が占める面積は 256×256 の画像に対しては 0.4%程度であるが，32×32 の画像に対しては 28.2%もの面積を占める．このフィルタを畳み込むことを 256×256 の画像に対しては局所的なぼかし処理であると言えるが，32×32 の画像に対してはもはや局所的とはいえず，大域的な平滑化により画像の詳細が失われている．このように，相似な画像に対して同じフィルタを畳み込んだ場合においても画像のサイズが異なれば相似な結果は得られないことが見てとれる．

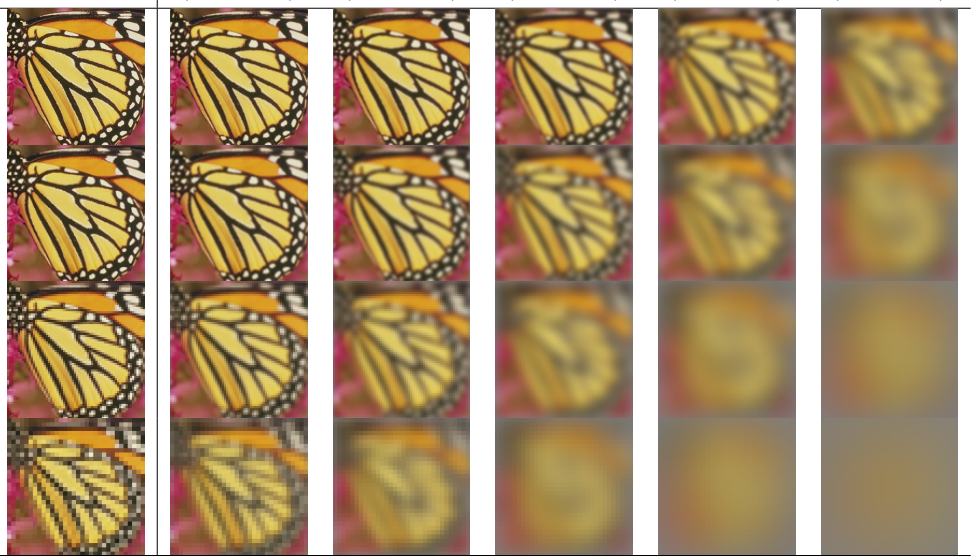
次に，U に W 畳み込んだ場合と「同じ範囲」を受容野とするために U' に畳み込むべきフィルタ W' のサイズ $k'\times k'$ を求める．つまり，U' に W' を畳み込む際，W' の中心が (y,x) にあるならば $\left[y-\frac{1}{2p},\, y+\frac{1}{2p}\right]\times\left[x-\frac{1}{2p},\, x+\frac{1}{2p}\right]$ の範囲で覆う状況を考える．ここで $qk=q'k'$ であるような有理数 q' を導入すれば，位置 (y,x) で W' が覆う範囲は $\left[y-\frac{1}{2q'},\, y+\frac{1}{2q'}\right]\times\left[x-\frac{1}{2q'},\, x+\frac{1}{2q'}\right]$ であり，$q'=p$ ならば，かつそのときに限り要件を満たすことがわかる．したがって，$k'=\frac{q}{p}k$ であるから，W' のフィルタサイズは $\frac{q}{p}k\times\frac{q}{p}k$ でなければならない．

いま，特徴マップ U と U' のサイズの比は $pk:qk=p:q$ であり，W と W' のサイズの比も同様に $k:\frac{q}{p}k=p:q$ である．つまり，相似な特徴マップにおいて「同じ範囲」を参照した畳み込みをするには特徴マップのサイズの比と同じ比率で畳み込みフィルタのサイズを変える必要がある．したがって，畳み込みフィルタのサイズを特徴マップのサイズに比例して変化させることが任意の相似な特徴マップに対する畳み込みで相似な出力を得るための必要条件のひとつとなる．

再び表 4.1 に注目する．先ほどは縦に見比べることで同じフィルタを異なるサイズの相似画像に畳み込んだ際の比較を行ったが，今度は右斜め上や左斜め下の関係（以下，斜め）にある画像同士を見比べる．この斜めの関係にある画像は十分に相似な関係であるといって差し支えない．つま

り，異なるサイズの相似画像から相似な出力を得たい場合に畳み込むべきフィルタのサイズの関係の例をこの表に見ることができる．表 4.1 において各行の画像サイズの比は 2 倍，1/2 倍の関係にあり，各列の畳み込みフィルタのサイズの比は約 2 倍，1/2 倍の関係にある．表中のある画像に注目した時，右上の結果は 2 倍のサイズの相似画像と約 2 倍のサイズの畳み込みフィルタから得られたものであり，左下の結果は 1/2 倍のサイズの相似画像と約 1/2 倍のサイズの畳み込みフィルタから得られたものである．このように，表 4.1 で用いた Gaussian フィルタのように適切な条件を満たせば，画像サイズにおよそ比例して畳み込みフィルタのサイズを変更することで畳み込みの出力が相似なものになりうる．

表 4.1: さまざまなサイズの相似な画像に対し，さまざまなサイズのフィルタを畳み込んだ例．

画像サイズ	入力画像	畳み込む $k \times k$ Gaussian フィルタ $(\sigma = k)$				
		$k = 3$ $(= 2^1 + 1)$	$k = 5$ $(= 2^2 + 1)$	$k = 9$ $(= 2^3 + 1)$	$k = 17$ $(= 2^4 + 1)$	$k = 33$ $(= 2^5 + 1)$
256×256						
128×128						
64×64						
32×32						

4.2.4 正規化された座標上の畳み込み

先の例では畳み込みフィルタのサイズのみに注目したが，次は実際の畳み込みについて述べる．

いま，相似な特徴マップ U と U' のあるチャンネル u，u' に注目し，それぞれに畳み込みフィルタ W と W' を畳み込んで相似な結果を得るための条件を考える．ここではある補間方法 Ip によって特徴マップを正規化された座標上のものに変換した $\mathrm{Ip}(u)$，$\mathrm{Ip}(u')$ のみならず，畳み込みフィルタを正規化された座標上のものに変換した $\mathrm{Ip}(W)$ や $\mathrm{Ip}(W')$ も導入する．また，議論を簡単にするため，いま畳み込み受容野を共通に $[0, \kappa] \times [0, \kappa]$ とする．つまり，今考えたいのは $\mathrm{Ip}(u * W)$ や

$\mathrm{Ip}(u' * W')$ の $\left(\frac{\kappa}{2},\ \frac{\kappa}{2}\right)$ における値である．具体的には

$$\mathrm{Ip}(u * W)\left(\frac{\kappa}{2},\ \frac{\kappa}{2}\right) = \int_0^\kappa \int_0^\kappa \mathrm{Ip}(u)(y,x) \cdot \mathrm{Ip}(W)\left(\frac{y}{\kappa},\ \frac{x}{\kappa}\right)\ \mathrm{d}y\ \mathrm{d}x \tag{4.2}$$

$$\mathrm{Ip}(u' * W')\left(\frac{\kappa}{2},\ \frac{\kappa}{2}\right) = \int_0^\kappa \int_0^\kappa \mathrm{Ip}(u')(y,x) \cdot \mathrm{Ip}(W')\left(\frac{y}{\kappa},\ \frac{x}{\kappa}\right)\ \mathrm{d}y\ \mathrm{d}x \tag{4.3}$$

となっており，$\mathrm{Ip}(u) = \mathrm{Ip}(u')$ であるような状況を考えると，$\mathrm{Ip}(W) = \mathrm{Ip}(W')$ ならば計算結果が一致する．もちろん，この条件を満たさずとも計算結果は一致しうるが，おそらくは特定の u や u' ならびに中心座標に依存したものであり，任意の u，u' に対し，すべての正規化された座標上で計算結果を一致させるには $\mathrm{Ip}(W) = \mathrm{Ip}(W')$，つまり，$W$ と W' が相似な状況を考えるのが無難である．したがって，以下では u に W を畳み込む場合の表式のみを示す．

式 (4.2) は Reimann 積分によって

$$\mathrm{Ip}(u * W)\left(\frac{\kappa}{2},\ \frac{\kappa}{2}\right) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i=1}^{N}\sum_{j=1}^{N} \mathrm{Ip}(u)\left(\frac{i}{N}, \frac{j}{N}\right) \cdot \mathrm{Ip}(W)\left(\frac{i}{\kappa N},\ \frac{j}{\kappa N}\right) \tag{4.4}$$

と変形できる．ここで，W のサイズを $k \times k$ とし，$N = nk$ なる n を導入し

$$\begin{aligned}
&\mathrm{Ip}(u * W)\left(\frac{\kappa}{2},\ \frac{\kappa}{2}\right) \\
=& \lim_{n\to\infty} \frac{1}{(nk)^2} \sum_{i=1}^{nk}\sum_{j=1}^{nk} \mathrm{Ip}(u)\left(\frac{i}{nk}, \frac{j}{nk}\right) \cdot \mathrm{Ip}(W)\left(\frac{i}{\kappa nk},\ \frac{j}{\kappa nk}\right) \\
=& \sum_{p=1}^{k}\sum_{q=1}^{k} \lim_{n\to\infty} \sum_{i=n(p-1)+1}^{np} \sum_{j=n(q-1)+1}^{nq} \mathrm{Ip}(u)\left(\frac{i}{nk}, \frac{j}{nk}\right) \cdot \frac{1}{(nk)^2} \mathrm{Ip}(W)\left(\frac{i}{\kappa nk},\ \frac{j}{\kappa nk}\right)
\end{aligned}$$

とすることもできる．したがって，任意の u，p，q に対して W が

$$u_{p,q} W_{p,q} = \lim_{n\to\infty} \sum_{i=n(p-1)}^{np} \sum_{j=n(q-1)}^{nq} \mathrm{Ip}(u)\left(\frac{i}{nk}, \frac{j}{nk}\right) \cdot \frac{1}{(nk)^2} \mathrm{Ip}(W)\left(\frac{i}{\kappa nk},\ \frac{j}{\kappa nk}\right) \tag{4.5}$$

を満たすことが，真に相似な結果を得るための条件である．さらなる議論の簡単化のため $p = q = 1$ の場合を考え，任意の n で

$$u_{1,1} W_{1,1} \simeq \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Ip}(u)\left(\frac{i}{nk}, \frac{j}{nk}\right) \cdot \frac{1}{(nk)^2} \mathrm{Ip}(W)\left(\frac{i}{\kappa nk},\ \frac{j}{\kappa nk}\right) \tag{4.6}$$

と近似する．さらに，k が十分大きければ $\mathrm{Ip}(u)\left(\frac{i}{nk}, \frac{j}{nk}\right)$ を $u_{1,1}$ で近似して

$$u_{1,1} W_{1,1} \simeq u_{1,1} \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{1}{(nk)^2} \mathrm{Ip}(W)\left(\frac{i}{\kappa nk},\ \frac{j}{\kappa nk}\right) \tag{4.7}$$

であるから，

$$W_{1,1} \simeq \frac{1}{(nk)^2} \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{Ip}(W)\left(\frac{i}{\kappa nk},\ \frac{j}{\kappa nk}\right) \tag{4.8}$$

を満たすとよい．

ここで，n は W の受容野が持つ $k \times k$ の領域それぞれをさらに $n \times n$ の小領域に分割することを意味するパラメータであり，n が大きくなればなるほど正確に $\mathrm{Ip}(u * W)$ の値を近似できる．また，式 (4.8) は W の各要素が持つべき値が $(nk)^2$，つまり考える小領域の分割個数に反比例することを示している．

いま，$nk \times nk$ の学習可能パラメータ配列 Θ として

$$\Theta_{i,j} \simeq \mathrm{Ip}(W)\left(\frac{i}{\kappa nk},\ \frac{j}{\kappa nk}\right) \tag{4.9}$$

となるようなものを想定する．このとき，畳み込むべき $k \times k$ フィルタ W は

$$W_{p,q} \simeq \frac{1}{k^2} \cdot \frac{1}{n^2} \sum_{i=n(p-1)+1}^{np} \sum_{j=n(q-1)+1}^{nq} \Theta_{i,j} \tag{4.10}$$

のように $k \times k$ の領域それぞれで $\Theta_{i,j}$ の平均を計算したあと，k^2 で割ってスケーリングすることが望ましいと考えられる．したがって，$k \times k$ フィルタ W と相似な $k' \times k'$ フィルタ W' の値のスケールの比として $k'^2 : k^2$ を想定し，k が k' の整数倍ではない場合の W から W' への変換時にはサイズが $k' \times k'$ になるよう補間をしたのち各要素の値を $\dfrac{k^2}{k'^2}$ 倍する方針をとる．

なお，ここで $W_{p,q}$ の総和をとれば，

$$\sum_{p=1}^{k}\sum_{q=1}^{k} W_{p,q} \simeq \frac{1}{(nk)^2} \sum_{p=1}^{k}\sum_{q=1}^{k} \sum_{i=n(p-1)+1}^{np} \sum_{j=n(q-1)+1}^{nq} \Theta_{i,j} = \frac{1}{(nk)^2} \sum_{i=1}^{nk}\sum_{j=1}^{nk} \Theta_{i,j} \tag{4.11}$$

となる．したがって，n_1，k_1，n_2，$k_2 \in \mathbb{N}_+$ が $n_1 k_1 = n_2 k_2$ の関係を満たすならば，共通の Θ から生成される $k_1 \times k_1$，$k_2 \times k_2$ の 2 種類のサイズのフィルタについていずれも総和が保存されることが言える．よって，特に $n_1 k_1 \times n_1 k_1$ フィルタから $k_1 \times k_1$ フィルタへの変換方法として，$n_1 \times n_1$ の各小領域における総和をとるという方法が考えられる．

4.2.5 画像のサイズの変化に伴うフィルタの生成の流れ

以上の議論から，具体的にフィルタを生成する方法について述べる．ただし，ここでは 1/4 スケールや 1/2 スケールの画像を用いた訓練を想定し，主に 2^{-n}（$n \in \mathbb{N}$）の形のリサイズのみを考える．

おおまかな流れとしては以下の 4 つのステップを条件に応じて適宜適用する．

a) 4.3.1 節の提案に基づいて Batch Normalization をモデルから除去する場合:
   1) Scaled Weighted Standardization [91] を適用する．
b) ターゲットスケールよりも小さいスケールに縮小する場合 ：
   2) 訓練スケールに合わせてフィルタをリサイズする．
   3) フィルタの値のスケール調整を行う．
c) 必要なフィルタサイズよりもサイズが大きい場合:
   4) フィルタの中央を切り抜く（センタークロップ）．

5) フィルタの値のスケール調整を行う．

条件 a) と処理 1) に関しては 4.3.1 節にて述べるが，本章の実験では比較のための事例を除いて基本的に 1) を適用する．条件 b) と c) については処理する特徴マップとフィルタのサイズの関係によって適用すべきかどうかが変化する．

この実装の詳細を説明するために，まずはフィルタの数値パラメータを定義する．フィルタは $K \times K$ 個の学習可能なパラメータの配列 Θ から生成されるものとする．ここで K は $M \in \mathbb{N}$ によって $K = 2^M + 1$ の形で表されるような状況を考える．この M は原フィルタとも呼べる K のスケールを表したものである．なお，Θ はスケール M で処理することを前提とした特徴マップに対して $K \times K$ の畳み込みフィルタとして直接利用することができる．つまり，スケールの変換を行わず，$K \times K$ のフィルタとして用いる場合においては標準的な「パラメータ＝フィルタ」の運用がなされる．また，このことから Θ はすでに第 3 章における線形結合によるフィルタの生成や上記の 1) の Scaled Weight Standardization といった前処理がすでになされたものを想定してもよい．ここから $k \times k$ の畳み込みフィルタを生成する．簡単化のため，k は正の奇数であるとし，完成するまではフィルタではなく 2 次元配列と呼び表す．

フィルタサイズは $K \times K$ から $k \times k$ へと直接変換させない場合もあり，一般には $nk_m \times nk_m$ のサイズにやや拡大してから，これを $k_m \times k_m$ に縮小したのち，センタークロップすることで $k \times k$ のフィルタを得る．ここで，k_m は自然数 $m \leq M$ によって $k_m = 2^m + 1$ の形で書けるようなものを指し，また，$\frac{k_m}{2} < k \leq k_m$ を満たす．すなわち，m は生成される $k \times k$ フィルタのスケールを表す．したがって，$K \times K$ から $k \times k$ への変換はスケール M からスケール m への変換であり，$k \times k$ のフィルタを畳み込むべき特徴マップは推論時に $K \times K$ で処理するターゲットスケールと比較して $1/2^{M-m}$ スケールの特徴マップであると言い表す．ターゲットスケールは $m = M$ の場合，つまり 1/1 スケールとも呼べるものである．実際には，m はターゲットスケールの特徴マップと処理したい特徴マップのサイズ比が，$1/2^{M-m}$ で近似できるようなものを選び，最終的な畳み込みフィルタのサイズ $k \times k$ も，m と後述するセンタークロップを行う条件によって自然に定まるものである．

前述の「やや拡大」させる際に用いる $n \in \mathbb{N}$ は $(n-1)k_m < K \leq nk_m$ を満たすように定める．適当な方法（本章の実験実装では bicubic 補間）によって $K \times K$ から $nk_m \times nk_m$ に拡大したのち，この 2 次元配列内における $n \times n$ の各小領域（窓）内において総和をとる（窓のサイズとずらし幅が $n \times n$ であるような総和プーリング）．これによって $k_m \times k_m$ の 2 次元配列を得るというのが，上記の 2) で示したリサイズの具体的な方法となる．

$nk_m \times nk_m$ のフィルタから $k_m \times k_m$ のフィルタの変換として総和プーリングをとる処理は 4.2.4 節の最後の部分により妥当であるとみなされる．したがって，さらに $K \times K$ から $nk_m \times nk_m$ への拡大処理がフィルタの変換として妥当であれば $K \times K$ から $k_m \times k_m$ へ妥当に変換されたものと言える．

$K \times K$ から $nk_m \times nk_m$ への補間による拡大処理は 4.2.4 節の考察により，変換後の値を $\frac{K^2}{(nk_m)^2}$ 倍することでフィルタの変換として妥当なものとなる．したがって，処理する 2 次元配列に対しこの値のスケーリングを行うが，$nk_m \times nk_m$ から $k_m \times k_m$ 総和プーリングをとったあとのほうが計

算回数を少なくできるため，値のスケール調整は総和プーリングの直後に実施する実装を推奨する．これが上記の 3) で示した処理に相当する．

ここまでで，$K \times K$ のフィルタを $k_m \times k_m$ のフィルタに変換する処理を行い，Θ から $k_m \times k_m$ のフィルタの生成する方法を提案した．しかしながら，$k_m \times k_m$ のフィルタが大きすぎるが，$\frac{k_m}{2} \times \frac{k_m}{2}$ では小さすぎる場合も存在する．

たとえば，本論文における実験のようにモデルの改変をなるべく小さくして比較したい場合は，元のモデルに合わせて 1/1 スケールのフィルタサイズを決めなければならない．こうした状況では元のモデルにおけるフィルタサイズを $K_o \times K_o$ としたとき，$2^{M-1}+1 < K_o \leq 2^M+1$ を満たすよう M を定め，$K \times K$ のフィルタの中心 $K_o \times K_o$ の領域のみを切り出してフィルタとして利用する．

あるいは，フィルタサイズが特徴マップのサイズの 2 倍以上である場合には，フィルタサイズが特徴マップのサイズ $D \times D$ の 2 倍未満になるよう $k = 2D-1$ と定める．$k = 2D-1$ のフィルタはすでに特徴マップ全体を参照するフィルタであるため，これ以上の拡大は無意味であると考えられるためである．

なお，このセンタークロップに際するフィルタ値のスケール調整には議論の余地があるが，本章における実験では先のスケール変換と同様にセンタークロップ前後のフィルタサイズを参照して $k \times k$ のフィルタの値をそれぞれ $\frac{k_m^2}{k^2}$ 倍した．

4.3 小さい画像でよりよい訓練をするための提案事項

4.2 節では，ターゲットスケールよりも小さい画像で訓練を行うフレームワークの概要と，それに際する畳み込みフィルタの変換処理について見てきたが，これを単純に適用しただけではもはや 1/1 スケールにおける訓練の代替とは言えないほど推論精度の低下を引き起こす．そこで本節では「Batch Normalization の除去」,「最低解像度の設定」,「混合スケール学習」，そして「訓練の延長」によって訓練後のモデルの推論精度を向上させることを提案する．前者の 3 つはターゲットスケールと学習スケール間のギャップを取り除くためのものであり，残る「訓練の延長」はスケール間のギャップを軽減するものではないが，提案手法の訓練フレームワークが標準的な方法と比較して何倍も高速であるために，状況に応じて導入することを提案するものである．

4.3.1 Batch Normalization の除去

Batch Normalization（BN） [95] は，特徴マップの各チャンネルにおける平均と分散を学習可能な定数に置き換える計算モジュールである．導入することにより訓練の安定化や過学習抑制による精度の向上が見られるため，広く用いられる．本章の実験で用いる RepLKNet [29] も Batch Normalization を含む深層学習モデルである．本節では，小さい画像で学習を行う提案フレームワークが Batch Normalization と併用することが難しいことを示す．また，精度の低下を抑えながら Batch Normalization を取り除く方法である，Normalizer-Free なモデル NFResNet [91] や NFNet [89] と，実験における RepLKNet への適用方法について述べる．

いま，特徴マップ $x \in \mathbb{R}^{D_1 \times D_2 \times N}$ を Batch Normalization $f_{\mathrm{bn}} : \mathbb{R}^{D_1 \times D_2 \times N} \to \mathbb{R}^{D_1 \times D_2 \times N}$ で処理

することを考える．このとき，

$$f_{\mathrm{bn}}(x) = \gamma \cdot \frac{x - \hat{\mu}}{\hat{\sigma}} + \beta \tag{4.12}$$

である．ここで，γ，$\beta \in \mathbb{R}^N$ は学習対象パラメータであり，$\hat{\mu}$，$\hat{\sigma} \in \mathbb{R}^N$ は各チャンネルに対する平均や標準偏差の推定値である．また，それぞれの加減乗除は x におけるすべての位置に対して行われるものとする．この処理により，平均値を $\hat{\mu}$ から β に，標準偏差を $\hat{\sigma}$ から γ に置き換える．

なお，$\hat{\mu}$，$\hat{\sigma}$ が表す「平均」や「標準偏差」は処理対象となるすべての画像を処理した際に入力となりうる特徴マップ全体の統計量であることを意図しており，訓練時と推論時で異なるものを用いる．訓練時に用いる統計量は処理している特徴マップから計算される．この際，平均 μ や標準偏差 σ はミニバッチ全体の各チャンネルで計算され，それぞれが式 (4.12) における $\hat{\mu}$ や $\hat{\sigma}$ として用いられる．推論時においてはこの μ，σ の指数移動平均 $\bar{\mu}$，$\bar{\sigma}$ を用いる．指数移動平均は各訓練ステップ t ごとに定数 $\alpha \in (0,\, 1)$ を用いた

$$\bar{\mu}^{(t+1)} = \alpha \cdot \bar{\mu}^{(t)} + (1 - \alpha) \cdot \mu^{(t)}, \qquad \bar{\sigma}^{(t+1)} = \alpha \cdot \bar{\sigma}^{(t)} + (1 - \alpha) \cdot \sigma^{(t)} \tag{4.13}$$

の漸化式で更新される．なお，α は慣例的に 0.9 程度が用いられており，本章の実験においてもこれに従った．このようにして，推論時に式 (4.12) の $\hat{\mu}$，$\hat{\sigma}$ として用いられる $\bar{\mu}$，$\bar{\sigma}$ は訓練データから得られたものを蓄積して獲得する．

本章の提案する訓練フレームワークにおいては実際に処理したい 1/1 スケールよりも小さいスケールの画像を用いて訓練を行う．つまり，1/1 スケールの画像を入力としたときの Batch Normalization においては $\hat{\mu}$，$\hat{\sigma}$ として 1/1 スケールの画像を処理したときの統計量が期待されるが，実際に用いられるのはそれよりも小さいスケール，たとえば，1/4 スケールの画像を訓練中処理した際にミニバッチから計算され，蓄積された統計量となる．したがって，期待される $\hat{\mu}$，$\hat{\sigma}$ を実際に用いられる $\bar{\mu}$，$\bar{\sigma}$ によって十分近似できていれば問題ないが，近似が不十分な場合，期待される計算との誤差を生じる原因となる．Batch Normalization は深層学習モデルの中で何度も用いられるため，発生した誤差は蓄積される可能性があり，結果として推論精度の劣化を招きうる．

また，Batch Normalization を用いたモデルは大きいミニバッチサイズ，つまり，同時に多くのサンプルを処理したときに推論精度が高くなる傾向にある．ミニバッチサイズを大きくするためには単純に同時に処理するサンプルを増やす方法のほかに，ミニバッチをさらに細かい単位に分割したマイクロバッチを考え，各マイクロバッチごとに計算した勾配を平均することにより，ミニバッチで処理した際に得られる勾配を得る方法がある．前者の単純に同時処理するサンプルを増やす方法は勾配を計算するために中間特徴マップを保存する必要があり，保存領域の容量の面でハードウェア的制約を受ける．後者のマイクロバッチに分割する方法は，多くの場合有効であるが，Batch Normalization と相性が悪い．Batch Normalization は訓練時，ミニバッチ内で平均や標準偏差を計算するが，マイクロバッチに分割すると統計量はマイクロバッチ内のものとなり，また，マイクロバッチごとに異なるものとなる．このことが要因となって，うまく訓練するには工夫が必要となる [90, 105]．ただし，ここではその詳細に触れず，第 3 の選択肢として Batch Normalization を除去することを考える．

Batch Normalization は画像処理用の深層学習モデルにおいて非常に強力な学習正則化効果を持つため，その有無によって大きく推論精度が変わる．近年では Transformer 系のモデルの発展により，Layer Normalization [106] が用いられる例も多いが，依然として Batch Normalization は有力な学習正則化手段のひとつである．しかしながら，上述したようなハードウェア的制約が存在するため，精度をなるべく落とすことなく Batch Normalization を取り除く試みがなされてきた．たとえば，異なる母集団によって似たような処理を行う Normalization 系統の方法がいくつかあり [106, 107]，なかでも Group Normalization [107] と畳み込みフィルタの重み正規化（Weight Standardization）を組み合わせた手法により Batch Normalization を上回る学習が可能であることが報告された [90]．小さい画像で訓練する提案フレームワークにおいてこうした手法で Batch Normalization を置き換えることも考えられるが，本章では別のアプローチを検討する．

本章において採用するのは Normalizer-Free な ResNet（NF-ResNet [91]）の戦略である．この戦略では Batch Normalization を用いたモデルにおける 2 点の特徴として

1. 各残差ブロックにおいて，特徴マップの各チャンネルにおける値の分散の平均値は一定値ずつ大きくなる．
2. 上記の値は，特徴マップがダウンサンプリングされる際（より正確には非残差ブロックにおける Batch Normalization の標準化処理が行われる際）に 1 にリセットされる．

ことを挙げており，特徴マップの分散コントロールしてこれを模倣することを試みる．具体的には以下のようにして分散をコントロールする．

1. 各残差ブロックにおいて増加させる分散の量の平方根 α を定数として定める．$\alpha = 0.2$ が標準的に用いられる．
2. 特徴マップに対する分散の増加量のコントロールは，残差ブロックの加算処理の直前において，ブロック内で処理した結果得られることとする，分散が 1 の特徴マップに α を乗じてからスキップした特徴マップと足し合わせることで実現する．
3. 各ブロックにおける分散の増加量が α^2 であると仮定すれば，各残差ブロックの入力特徴マップにおける期待分散を求めることができるため，残差ブロックではスキップしない特徴マップの値を期待分散の平方根で割り，ブロック内で処理する特徴マップの分散を 1 とする．
4. 各残差ブロックにおけるスキップしないほうの処理では計算モジュールごとに分散が 1 のまま保存されるよう補正をかける．
5. 各 Stage 間に存在する非残差の Transition の部分ではそれまでに蓄積されているであろう期待分散の平方根で特徴マップの値を割る．

このうち，4. の分散を 1 のまま保存するための補正について具体例を挙げると

1. 畳み込み層においてはフィルタの各 fan-in（入力チャンネルについても考慮した特徴マップの受容野）に関して重みの標準化（Weight Standardization [90]）を行い，また，学習可能な定数によってそれらをスケーリングする（Scaled Weight Standardization）．

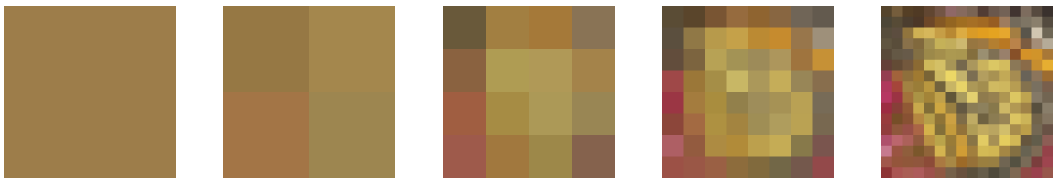
2. 活性化層においては，各活性化関数に対して実験的に求めた結果を用いてスケーリングする．実験は分散が 1 の正規乱数をその活性化関数の入力として利用し，その出力の分散を求めるものである．
3. Channel Attention [57] の直後には特徴マップの値を 2 倍する．

のようになる．なお，本章の実験においてはベースラインのモデルに対する変更を最小限にとどめるため，Channel Attention は使用していない．

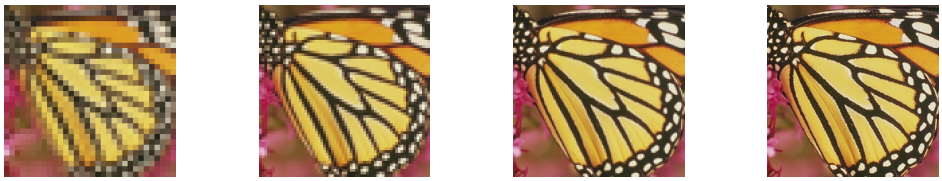
注意点として，1. の Scaled Weight Standardization を適用するのは本章の提案するフィルタの生成において最初でなければならない．本章の提案では画像サイズに合わせてフィルタを生成するが，これに伴ってフィルタサイズに反比例したスケーリングを実施する．もしも Scaled Weight Standardization が処理したいスケール用のフィルタ生成後に行われた場合，このスケーリングは標準化によって失われ，かつ，フィルタサイズに反比例しない学習可能な定数によって値が再スケーリングされる．これではこの畳み込みによる処理が異なるスケール間で相似とならないため，Scaled Weight Standardization はフィルタの生成前に，オリジナルの学習可能パラメータに対して実施するべきである．

4.3.2 最低解像度の設定

階層的な深層学習モデルにおいては特徴マップを段階的にダウンサンプリングしながら入力画像全体の特徴を抽出しようと試みる．これに伴って，特徴マップのサイズは小さくなる．推論時よりも小さい画像を入力とする本章の提案フレームワークでは極小な特徴マップが得られることがある．たとえば，本章の実験では 1/1 スケール（224×224）の画像を入力した際（表 4.2 の Case 1），Stage 4 において 7×7 の特徴マップが処理される．ここで，1/4 スケールの画像を入力した場合（表 4.2 の Case 3）を考えると，Stage 4 においては 2×2 の特徴マップを処理することとなる．こういった小さすぎる特徴マップが生じないよう，最小のマップサイズ（最低解像度）を定めるというのが本節の提案事項である．



(a) 1×1　(b) 2×2　(c) 4×4　(d) 8×8　(e) 16×16



(f) 32×32　(g) 64×64　(h) 128×128　(i) 256×256（原画像）

図 4.2: 画像の解像度による見た目の比較

たとえば，図 4.2 を見た場合，ある程度小さい解像度までは原画像とおよそ相似なものとみなすことができ，被写体が蝶であることを認識できるであろう．しかしながら，極端に解像度が小さくなると，原画像との相似性は失われ，もはや何が写っているのかわからないようになる．特徴マップにおいても同じことが言うことができ，ターゲットスケールの特徴マップと訓練で用いる小さいスケールの特徴マップの相似性を担保できる解像度には限界があるのではないかというのが最低解像度を定めるモチベーションである．

これに伴う実際の変更として，ダウンサンプリングした結果が最低解像度を下回る場合はダウンサンプリングを行わない．たとえば，stride が 2 の畳み込みでダウンサンプリングを行うことが想定される状況では，stride を 1 に変更する．フィルタサイズは特徴マップのサイズに比例して決定されるべきであるため，以降の部分では最低解像度の設定前よりも大きいサイズのフィルタを畳み込むことになる．

本章の実験で扱うモデルの例を用いて具体的に説明する．本章の実験においては 1/1 スケールにおける特徴マップの最小サイズ（表 4.2 の Case 1 における Stage 4）と同じになるよう最低解像度を 7×7 に設定した．1/4 スケール（56×56）の画像を入力とした場合，最低解像度を定めなければ表 4.2 の Case 3 のようになる．これは Case 2 の状態と 4 : 1 の関係になるようにしたシンプルな実装例であり，Stage 3 以降は 4×4 以下のサイズの特徴マップを処理していることがわかる．これに対し，最低解像度を 7×7 に設定したものが Case 4 であり，Case 3 との差分は太字によって示されている．Case 3 では Transition 2 において stride が 2 の畳み込みを行い，特徴マップのサイズを 7×7 から 4×4 にダウンサンプリングする．一方，Case 4 では Transition 2 における stride が 2 の畳み込みが設定された最低解像度を下回るサイズの特徴マップを出力するため，stride を 1 として畳み込みを行い，入力と同じ 7×7 の特徴マップを得る．その結果，Stage 3 において Case 4 で処理するのは 7×7 の特徴マップとなる．このとき，1/1 スケールの画像を処理した場合（Case 2）の Stage 3 における特徴マップは 14×14 であるから，Case 4 の Stage 3 ではその 1/2 のサイズの特徴マップを処理していることとなる．したがって，Case 4 の Stage 3 では Case 2 の Stage 3 のフィルタサイズ 27×27 を 1/2 スケールに変換したものとして，Case 3 のような 7×7 ではなく，13×13 のフィルタで畳み込みを行う．続く Transition 3 においても同様に stride は 1 としてマップサイズ 7×7 を保存し，かつ，畳み込みフィルタのサイズは Case 2 の 1/2 として 5×5 のものを用いる．最終の Stage 4 においては Case 2 と Case 4 で同じ処理を行う．以上が，最低解像度を定めた場合の具体的な処理の流れとなる．

特徴マップとフィルタの両方が大きくなるため計算コストは増加するが，計算コストの大きい，高解像度の処理は依然として小さいスケールで学習されたままであり，影響を受けるのは最低解像度を下回るような部分に限られる．したがって，最低解像度を定めたとしても十分に訓練高速化の恩恵を受けることができる．にもかかわらず，低解像度部分におけるスケール間ギャップを軽減し，小さい画像を用いた 1/1 スケール向けの相似なシミュレーションとそれに基づくパラメータ調整がしやすくなることから，訓練したモデルの精度向上につながると期待される．なお，最低解像度としてうまく機能する値はタスクによって異なると考えられるが，おそらく，ベースラインとなる深層学習モデルにおいて推論時に処理する最小の特徴マップサイズが適当であると思われる．逆

に，それ以上の解像度に設定することは 1/1 スケールにおける特徴マップよりも大きいサイズのものを処理しなければならなくなるため，最低解像度の上限は，ベースラインモデルの処理する最小特徴マップサイズとなる．

表 4.2: 実験で用いるクラス分類器の各ステージにおける特徴マップと畳み込みフィルタのサイズ

プロパティ	**Case 1**		**Case 2**		**Case 3**		**Case 4**	
モデル	BN_3，NF_3		BN_9，NF_9		BN_9，NF_9		BN_9，NF_9	
空間スケール	1/1		1/1		1/4		1/4	
最低解像度	-		-		-		7×7	
ブロック	**入力マップ**	**フィルタ**	**入力マップ**	**フィルタ**	**入力マップ**	**フィルタ**	**入力マップ**	**フィルタ**
Stem	224×224	3×3	224×224	$\mathbf{9 \times 9}$	56×56	3×3	56×56	3×3
Stage 1	56×56	31×31	56×56	31×31	14×14	9×9	14×14	9×9
Transition 1	56×56	3×3	56×56	$\mathbf{9 \times 9}$	14×14	3×3	14×14	3×3
Stage 2	28×28	29×29	28×28	29×29	7×7	9×9	7×7	9×9
Transition 2	28×28	3×3	28×28	$\mathbf{9 \times 9}$	7×7	3×3	7×7	3×3
Stage 3	14×14	27×27	14×14	27×27	4×4	7×7	$\mathbf{7 \times 7}$	$\mathbf{13 \times 13}$
Transition 3	14×14	3×3	14×14	$\mathbf{9 \times 9}$	4×4	3×3	$\mathbf{7 \times 7}$	$\mathbf{5 \times 5}$
Stage 4	7×7	13×13	7×7	13×13	2×2	3×3	$\mathbf{7 \times 7}$	$\mathbf{13 \times 13}$

4.3.3 混合スケール学習

ここまでは推論時よりも小さいスケールの画像のみを用いて訓練することを前提としてきた．これにより，大まかな部分で 1/1 スケールにおける処理を相似な形でシミュレーションし，それに適したパラメータを得ることができる．しかしながら，この方法には限界があり，訓練に用いた小さい画像サイズとのスケール間のギャップが生じる結果として，1/1 スケールにおける精度の低下を招く．

前節の最低解像度は 1/1 スケールにおける最小のマップサイズ以下に設定しなければならない．したがって，それ以上 1/1 スケールに近づけるためには深層学習モデル全体でひとつのものではなく，モジュール単位で最低解像度を定める必要が生じる．このような戦略をとった場合，調整の選択肢が多く生じ，実験を高速化するための実験に時間をとられることになる．そこで，実際に 1/1 スケールなど，メインで訓練するものよりも大きいサイズの画像を用いた訓練を行うこととする．ただし，メインで訓練する小さいスケールの画像によってもおおまかにパラメータを訓練することは可能であるため，大きいスケールの画像は補助的に用いればよい．そこで，低頻度でより大きなスケールの画像を訓練に用いる混合スケール学習を提案する．

具体的な例として，本章の実験においては 1/4 スケール（56×56）の画像を中心に学習するが，補助的に 1/2 スケール（112×112）や 1/1 スケール（224×224）の画像を用いて訓練する．実験設定では 1/4，1/2，1/1 の画像を 56 : 7 : 1 で使用することとした．これにより，基本的には 1/4 スケールで訓練することとなるが，64 訓練ステップにつき 1 ステップは 1/1 スケールの画像を，約 8 訓練ステップにつき 1 ステップは 1/2 スケールの画像を訓練の入力として用いる．ここで「約」と

なっているのは 1/2 スケールの画像を用いる条件として「ステップベースの時刻が”64 の倍数ではない 8 の倍数” から 1 引いたものとなるとき」に設定したためである．つまり，8 訓練ステップごとに 1/2 スケール以上の画像を用いる機会が訪れ，そのうち 8 回に 1 回はさらに大きい 1/1 スケールの画像を用いる実装である．

特筆すべきは 1/1 スケールの画像が用いられるのは 1/64 と極めて低頻度であることである．Progressive Learning [94] において 1/1 スケールの画像が 1/4 程度用いられることと比較しても，訓練時における 1/1 スケールの画像の使用をかなり抑えていると言える．さらに，1/2 スケールの画像も 7/64 程度用いるが，Progressive Learning においては 1/2 スケールより大きく，1/1 スケールよりも小さい画像が 1/2 用いられる．つまり，Progressive Learning と比較して 1/1 スケールの画像を用いる頻度は 1/16 に抑えられ（93.8%減），1/2 スケール以上 1/1 スケール未満の画像を用いる頻度は 7/32 に抑えられている（78.2%減）と言える．ただし，本章の提案フレームワークでは 3×3 の畳み込みを 1/1 スケールにおいて 9×9 に，1/2 スケールにおいて 5×5 としているため，1/1 スケールや 1/2 スケールの処理はやや高コストになっている可能性がある．したがって，あくまでも参考程度の概算であることには注意されたい．

本節の結びとして，なぜ混合スケールで学習する必要があるのかを考察する．たとえば，1/1 スケールにおいて 9×9 なるフィルタ（学習可能パラメータの配列 Θ）を考える．これを 1/4 スケールに変換するには単純にフィルタを 3×3 の小領域内で総和をとるだけでよい．したがって，生成された 1/4 スケールのフィルタを W とし，そのインデックスを p，q で表すと，

$$W_{p,q} = \sum_{i=1}^{3} \sum_{j=1}^{3} \Theta_{3(p-1)+i,3(q-1)+j} \tag{4.14}$$

となる．したがって，損失 L から 1/4 スケールにおけるフィルタ値 $W_{1,1}$ に逆伝播された勾配を用い，パラメータ $\Theta_{i,j}$（i，$j \in \{1,2,3\}$）に対する損失 L の勾配を

$$\frac{\partial L}{\partial \Theta_{i,j}} = \frac{\partial L}{\partial W_{1,1}} \cdot \frac{\partial W_{1,1}}{\partial \Theta_{i,j}} = \frac{\partial L}{\partial W_{1,1}} \tag{4.15}$$

と求められる．つまり，$\Theta_{i,j}$ の調整に必要な勾配は実際のところ総和される小領域内で一定となり，小領域内の各 $\Theta_{i,j}$ は各訓練ステップで全く同じ値を加算される．いま，$\Theta_{i,j}$ の初期値を $\Theta_{i,j}^{(0)}$，訓練開始時より $\Theta_{i,j}$ に足された値の合計を a とすると，任意の i_1，j_1，i_2，$j_2 \in \{1,2,3\}$ に対し，

$$\Theta_{i_1,j_1} - \Theta_{i_2,j_2} = \left(\Theta_{i_1,j_1}^{(0)} + a\right) - \left(\Theta_{i_2,j_2}^{(0)} + a\right) = \Theta_{i_1,j_1}^{(0)} - \Theta_{i_2,j_2}^{(0)} \tag{4.16}$$

が成り立つ．このことから，$W_{1,1}$ における内部のパラメータ同士の差は訓練開始時より変化することなく保存されることがわかる．したがって，$\Theta_{i,j}$ の総和は最適化されるが，1/1 スケールのフィルタとして用いる際に個々別々に重みとして用いられる各 $\Theta_{i,j}$ 単体の最適化は行われない．そこで個々の $\Theta_{i,j}$ に対する最適化を促すのが混合スケール学習であると解釈することができる．

4.3.4 訓練の延長

パラメータの微調整を繰り返して最適化する深層学習において，訓練ステップ数は重要な要素である．極論を言えば，どれほど優れた深層学習モデルであっても 1 訓練ステップで最適化されるこ

とはない．つまり，十分に精度が収束する，あるいは，過学習を引き起こして精度が下降し始める程度に訓練することが望ましいが，実際にはそれだけの訓練ステップを確保できない場面が多く，時間的な制約と精度を勘案して適当な訓練ステップ数を設定する．

本章の提案する訓練フレームワークは 4.3.2 節の最小解像度の設定や 4.3.3 節の混合スケール学習による速度の低下を加味したとしてもなお 1/1 スケールのみで行う訓練より高速になる．例として，本章の実験においては 9.4 倍高速な訓練を実現した．そこで，短縮されたぶんの時間を訓練ステップ数の増加，すなわち，訓練の延長に用いることを提案する．たとえば，上記の例において 2 倍の訓練ステップを確保したとしても，4.7 倍高速に学習済みモデルを得ることができる．ステップ数が不十分な条件設定の場合，この訓練の延長によってより精度の高い結果が期待される．

しかしながら，訓練ステップ数を増やすことに関する公平性の議論は避けられない．状況によってはこれを適用することが不適切だとみなされうる．たとえば，なんらかのタスクで最高峰の結果を得ることが目的の状況では，訓練を延長しても問題ないと判断されるであろう．あるいは，なんらかの提案事項 P に対する効果を示したい場合，片方を P なし，もう片方を P 含めて訓練することになるが，後者のみに対し訓練の延長を行って，良い結果が得られたとしても P が有効であることをいうことはできない．

ただし，「P ＋訓練の延長」をひとつの提案と見れば，この提案はある種の有効性を持っている可能性がある．P なしで通常の訓練ステップ訓練するよりも，P と延長された訓練ステップぶん訓練したほうが短時間で済む場合，前者が不利とは言い切れないためである．たとえば，P なしの条件で 30 時間，P ありの条件で 5 時間訓練に要するとする．ここで P ありの条件に追加で訓練ステップの延長を行い訓練ステップ数を 2 倍にした結果，P なしの場合と同程度以上の精度の学習モデルが得られたすれば，これはもとの 1/3 の時間で同程度以上の精度が達成されたことになる．もし P なしのモデルに対しても 2 倍の訓練延長を行った結果が P ありに訓練延長を行った場合より精度がよかったとしても，「P あり＋訓練延長」という提案自体は「同じ時間で得られる結果がよりよいものとなる」価値を持ちうる．何をもって公平であるとみなすかは状況次第であるが，本論文においては「P なしの場合は 3 倍の時間をかけても P ありの場合の比較においてあらゆる観点から公平である」という態度はとらない．つまり，P なしの場合に P ありの場合と比較して 3 倍も時間をかけて構わないとするのは，ある種の不公平を生むと考える．したがって，本章で提案する訓練フレームワークにおける提案事項として，比較対象とする手法の訓練よりも短時間で終了することを条件に訓練の延長を精度向上のための提案として含めることとする．

4.4 実験

4.4.1 実験の設定

実験はすべて画像のクラス分類タスクによって行う．ある程度共通の条件下で訓練を行い，正解率に相当する Top-1 Accuracy や訓練速度によってそれぞれの方法を評価する．ほとんど同一の深層学習モデルを訓練した結果を比較することにより，それぞれの訓練方法が引き出せるモデルのポ

テンシャルの高さ（パラメータ調整のうまさ）と高速化の程度を調べる．訓練方法としては，終始 1/1 スケールの画像で訓練する「標準」的な手法と本章の「提案手法」のほかに，小さい画像から訓練をはじめ，徐々に画像を大きくしていく「Progressive」な方法 [94] も比較する．

本章の提案する訓練フレームワークにおいては小さい画像を用いて訓練すること，ならびに，それと比例して畳み込みフィルタの縮小を伴う生成を行うことを前提としている．また，多くの深層学習モデルで採用されている 3×3 畳み込みの縮小については本研究の範囲を逸脱する複雑なものであるため，なるべく 3×3 の畳み込みを含まない RepLKNet [29] のベースラインモデルである RepLKNet-31B をベースに実験用のモデルを設計した．RepLKNet における「Rep」は Re-parameterization（パラメータの再配置） [108] を指しており，これは訓練後のモデルに対する推論のための最適化工程である．31×31 などの受容野の大きい畳み込みだけではなく，並列して 5×5 の小受容野の畳み込みを行い，結果を足し合わせることで，大きい受容野の畳み込みの訓練を補助しながらも，この Re-parameterization を用いることにより，推論時の計算コストを小さくすることができるというのが RepLKNet の原典 [29] における主張点のひとつとなっている．しかしながら，今興味があるのは「画像サイズに合わせてフィルタを生成することを前提に小さい画像を中心に用いて訓練すれば，十分に 1/1 スケールの画像を用いた標準的な訓練の代替足りえるのか」といったもう少し単純なことであるから，本実験においては補助的に用いられる小受容野ブランチを削除することとした．

表 4.3 に本実験で用いる派生モデルのプロパティを示す．前述の理由のために，RepLKNet-31B から単純に小受容野ブランチを除いたモデルが本研究のベースラインとなる BN_3 である．また，本研究の提案事項には「Batch Normalization を取り除き，NF-ResNet の戦略に基づいて分散コントロールを行うこと」「小さすぎるフィルタは大きくすること」が含まれているため，それぞれのモデル改変を適用した比較用のモデルも定義した．本実験においてはこの 2 点の改変の有無を表した記号によってモデルを区別することとする．

表 4.3: 第 4 章の実験で用いる RepLKNet-31B の派生モデル

モデル	ベースモデル	プロパティ			計算量（MFLOPs）	
		小受容野ブランチ	BN	$2\downarrow$ 畳み込み	訓練時	推論時
不使用	RepLKNet-31B [29]	✓	✓	3×3	31,101	31,101
BN_3	RepLKNet-31B [29]		✓	3×3	30,917	30,917
NF_3	RepLKNet-31B [29]			3×3	30,684	30,667
BN_9	RepLKNet-31B [29]		✓	9×9	31,273	31,273
NF_9	RepLKNet-31B [29]			9×9	31,040	31,023

データセットは CIFAR-10 [92]，CIFAR-100 [92]，STL-10 [109] を用いる．表 4.4 にデータセットのプロパティを示す．CIFAR-10 は 1 クラスあたりのサンプル数が多く得られた場合を想定しており，CIFAR-100 はクラス数が多い状況を想定している．STL-10 は他 2 つに比べて高解像度なこと

や訓練データが少ないことが特徴的である．また，学習率（ピーク値）はデータセットごとに変えており，BN_3 を標準的な方法で訓練した際に最も推論精度が高くなるものを設定した．

表 4.4: 第 4 章の実験で用いるクラス分類データセット

データセット	原画像サイズ	クラス数	サンプル数		学習率
			訓練用（1 クラスあたり）	検証用	
CIFAR-10 [92]	32×32	10	50000 (5000)	10000	1.0×10^{-3}
CIFAR-100 [92]	32×32	100	50000 (500)	10000	1.0×10^{-3}
STL-10 [109]	96×96	10	5000 (500)	8000	5.0×10^{-3}

推論用の画像は bicubic 補間により 256×256 へ拡大したあと，224×224 となるようセンタークロップしたもの用い，224×224 を本実験における 1/1 スケールと定める．したがって，1/2 スケールは 112×112 の画像を，1/4 スケールは 56×56 の画像をそれぞれ入力とした際のことをいう．なお，表 4.4 にあるように，本実験で用いる画像データはいずれのデータセットにおいても 224×224 よりも小さいが，224×224 の画像入力に対して訓練することが推論精度に貢献することも 4.4.4 節で示す．訓練用の画像に対しては Distorted Random Crop [2] と等確率のランダムな左右反転のみをデータ拡張として用い，そののち，bicubic 補間によって入力として用いる画像サイズに拡大される．また，0 から 255 の整数値からなる画素値は 255 で割って浮動小数点数にしたあと，データセットにおける R，G，B 各チャンネルの平均と標準偏差（表 4.5）を用いて標準化した．

表 4.5: 各データセットにおける画像の色成分の平均と標準偏差

データセット	R	G	B
CIFAR-10 [92]	0.491 ± 0.247	0.482 ± 0.243	0.447 ± 0.262
CIFAR-100 [92]	0.507 ± 0.268	0.487 ± 0.257	0.441 ± 0.276
STL-10 [109]	0.441 ± 0.268	0.428 ± 0.261	0.387 ± 0.269

最適化に関して，ミニバッチサイズを 32 に設定し，90 epoch の訓練を基本とする．パラメータ調整に用いる Optimizer は AdamW [45] を使用する．学習率は 20 epoch にわたる訓練ステップ単位の線形な warmup によって，0 から表 4.4 の学習率まで大きくしたあと，訓練終了時点で 0 になるよう cosine annealing により減衰させる．AdamW の weight decay の値は 0.05 としたが，例外として

1. 畳み込み層や Batch Normalization におけるバイアス
2. Scaled Weight Standardization [91] における畳み込みフィルタのゲイン
3. 非残差ブロック（Stem と Transition）における Batch Normalization の γ

のパラメータに対しては重み減衰を行っていない．なお，3. については [110] の知見に基づいて除外している．また，推論精度向上のため Sharpness-Aware Minimization（SAM） [47] のステップを 5 訓練ステップに 1 度行っており，勾配上昇には学習率 -0.05 の SGD を用いた．NFNet [89] にお

いては NF-ResNet [91] の方法によって改変したモデルに対し，Adaptive Gradient Clipping（AGC）を適用することで推論精度が改善すると報告されているが，本実験の条件では逆に NF_3 の推論精度が低下したため，NF_9 の訓練時も含め，一切使用していない．

損失関数は Cross Entropy のみを用い，平滑化率 0.1 の Label Smoothing [53] によってラベルの平滑化を行った．

プログラムコードは機械学習用の Python フレームワークのひとつである JAX [111] を用いて実装した．訓練や推論のステップに対しては XLA [112] による実行時コンパイルを行っており，初回実行の前に計算グラフの最適化を行った．これによってコンパイルのオーバーヘッドが発生したが，訓練時間としてはこれを含めないこととした．なお，数値のデータ型としては広く用いられる float32 ではなく，bfloat16 [113] を採用し，高速化を図った．訓練時においても推論時においても bfloat16 による計算を行っている．また，すべての実験は NVIDIA GeForce RTX3090 GPU を搭載したグラフィックボード上で実行した．

4.4.2 従来手法との比較

本節では以下の 3 つの手法について比較を行う．

1. 標準的な方法：
   - フィルタは生成しない．
   - モデル改変も必要としない．
   - 常に同じ画像サイズ（224×224）で訓練する．
2. Progressive Learning [94]：
   - フィルタは生成しない．
   - モデル改変も必要としない．
   - 訓練時における画像サイズは 1 から 22 epoch 目にかけて 98×98，23 から 44 epoch 目にかけて 140×140，45 から 66 epoch 目にかけて 182×182，67 から 90 epoch 目にかけて 224×224 とする．
3. 提案手法：
   - 画像サイズに合わせてフィルタを生成する．
   - 小さい画像に対してフィルタを縮小するために元のモデルの畳み込みフィルタを最低限必要なサイズに拡大する．
   - また，Batch Normalization [95] が含まれる場合には NF-ResNet [91] の方法によってこれを取り除く．（Batch Normalization の除去；4.3.1 節）
   - 特徴マップが 7×7 を下回るようなダウンサンプリングは行わない．（最低解像度の設定；4.3.2 節）
   - 訓練では 56×56，112×112，224×224 の画像を訓練ステップごとに同じ頻度で切り替えながら 56 : 7 : 1 の割合で用いる．（混合スケール学習；4.3.3 節）
   - 標準的な方法や Progressive Learning よりも早く訓練が終わることを条件に訓練ステップ

数を 2 倍の 180 epoch にする．（訓練の延長；4.3.4 節）

推論精度に関する結果を表 4.6 に示す．上の 4 事例は標準的な訓練方法で表 4.3 を訓練したものであるが，これは提案手法がモデル改変を必要とするため，同じモデルの標準的な訓練結果を比較するためのものである．また，同じ epoch 数による比較も行うため，提案手法については訓練の延長の有無のみ違う実験の結果を両方示した．

CIFAR-10 データセットにおいてはほとんど同じ推論精度となっている．訓練延長を行わない場合の提案手法では精度がやや劣るが，訓練延長の結果，他の 2 手法と同等の水準の推論精度を達成している．標準的な方法で訓練した NF_9 も同程度の推論精度のため，これはモデルによる優位性ではないと考えられる．

CIFAR-100 データセットにおいては Progressive Learning（BN_3）が標準的な方法や提案手法を上回る推論精度を達成している．提案手法については先ほどと同様，訓練延長を行わない場合に標準的な方法に劣るが，訓練の延長によりほとんど同程度の水準で訓練できている．一方，同じモデル NF_9 を訓練した結果としては他の 2 手法よりよいものとなっている．

STL-10 データセットにおいては逆に，提案手法が最も高精度となっている．ただし，これは標準的な方法で訓練した NF_9 が BN_3 よりも高精度であるため，モデルとデータセットの相性が原因であると考えられる．とはいえ，NF_9 を標準的な方法で訓練した場合と比較して，同等の推論精度を達成しており，Progressive Learning を用いたものより高精度である．一方，Progressive Learning で訓練した BN_3 の精度が顕著に低い値となっている．理由については 4.4.3 節にて考察するが，条件次第ではこういうことも起こりうることが示された．

以上のことから，提案手法は訓練の延長を行った場合，標準的な訓練方法と同程度の推論精度が達成できることが示された．また，Progressive Learning と比較して，訓練後のモデルの精度が劣ることもあるが，STL-10 データセットにおける事例のように大きな失敗がないというのが本手法の強みであると考える．

表 4.6: モデルや訓練方法の違いによるクラス分類精度の比較

モデル	訓練方法	epoch 数	Top-1 Accuracy (%)		
			CIFAR-10	CIFAR-100	STL-10
BN_3	標準	90	95.5	79.5	80.6
NF_3	標準	90	94.8	75.5	72.6
BN_9	標準	90	95.5	78.8	84.0
NF_9	標準	90	95.5	79.0	81.7
BN_3	Progressive	90	95.7	80.3	64.8
NF_9	Progressive	90	95.8	78.8	78.1
NF_9	提案手法	90	95.1	78.8	78.6
NF_9	提案手法	180	95.7	79.4	81.7

次に，訓練速度について見る．結果を表 4.7 に示す．表中における 1epoch あたりの秒数（s/epoch）は実行時コンパイルによる影響を考慮し，最終 epoch の値を参照した．ただし，Progressive Learning においては画像サイズが切り替わる直前の epoch の秒数も参照し，平均値をとった．また，比較を容易にするため，BN_3 を標準的な方法で訓練した際の訓練時間（s/epoch×epoch 数により算出）を 1 とする相対的な訓練時間も併記した．この相対値は訓練の延長も考慮されているため，訓練速度に関しては単純にこの値を比較すればよい．

提案手法による訓練はいずれもベースラインとした BN_3 の標準的な方法による訓練と比較し，4.62 倍以上高速に行われた．また，Progressive Learning も BN_3 の訓練において標準的な訓練方法と比較した場合，1.76 倍以上の高速化に成功しているが，提案手法はさらにその 2.61 倍以上高速に訓練している．標準的な訓練方法による NF_9 の例では BN_3 の約半分程度の訓練速度となっており，この高速化がモデルによるものではなく，提案した訓練方法によって実現されたと言える．なお，標準的な訓練方法において NF_9 の訓練が表 4.3 に示した計算量（MFLOPs）の増加の程度（1.03 倍）に見合わないほど遅くなっているが，これは実行時コンパイルにおいて，高速に計算可能な畳み込みアルゴリズムを見つけられなかったことに起因する．訓練ステップの約 1/4 をターゲットスケールで訓練する Progressive Learning はこの影響を強く受けており，標準的な方法による BN_3 と比較した場合の高速化は 1.1 倍程度にとどまっている．これに対して，提案手法はこのような不利な条件にありながらも安定して高速な訓練を標準的な方法と同等程度の精度で実現している．これにより，たとえば，提案手法による訓練を行うことで実験回数を増やしたり，より大規模な訓練に挑戦できるようになることが期待される．

表 4.7: モデルや訓練方法の違いによる訓練速度の比較

条件			訓練速度					
モデル	訓練方法	epoch 数	CIFAR-10		CIFAR-100		STL-10	
			s/epoch	相対	s/epoch	相対	s/epoch	相対
BN_3	標準	90	1169	1	1165	1	118	1
NF_3	標準	90	1147	1.02	1142	1.02	116	1.02
BN_9	標準	90	2269	0.52	2275	0.51	231	0.51
NF_9	標準	90	2246	0.52	2251	0.52	228	0.52
BN_3	Progressive	90	658	1.78	657	1.77	67	1.76
NF_9	Progressive	90	1070	1.09	1070	1.09	109	1.09
NF_9	提案手法	90	124	9.43	124	9.40	12	9.83
NF_9	提案手法	180	123	4.75	126	4.62	12	4.91

4.4.3 Progressive Learning の弱点

Progressive Learning が STL-10 データセットにおける BN_3 の訓練において著しく低い精度となっているのは STL-10 の訓練データの数が CIFAR-10 や CIFAR-100 の 1/10 であり，それに伴っ

て各画像サイズごとの訓練ステップ数を十分に確保できなかった（3432 ステップ）ためであると考えられた．そこで，STL-10 における Progressive Learning の訓練延長を行い，各画像サイズごとの訓練ステップ数を増加させた実験を行った．結果を表 4.8 に示す．Progressive Learning の訓練ステップ数を 2 倍（180 epoch），4 倍（360 epoch）にすると，推論精度に改善が見られた．しかしながら，8 倍（720 epoch）で訓練した場合，過学習のためか精度の低下がみられ，このような粗い設定では標準的な訓練方法の精度（80.6%）を上回ることはなかった．さらに言えば，Progressive Learning においては訓練ステップ数を 2 倍にした時点で標準的な訓練方法よりも時間がかかっており，180 から 720 epoch のいずれかに最適な訓練ステップ数が存在し，かつ，標準的な方法より高精度にモデルを訓練できたとしても，ほかのハイパーパラメータを調整しない限り標準的な方法に対する高速化は実現しない．一方で，提案手法を用いた訓練では，訓練の延長を含めるとベースラインの BN_3 や用いたモデル NF_9 を標準的な方法で訓練した場合と比較して同等程度以上の推論精度を実現しており，また，依然として 4.6 倍以上高速な訓練を実現している．したがって，提案手法による訓練は Progressive Learning と比較して Progressive Learning に適した条件下（CIFAR-100）では推論精度の面で劣るが，高速かつ安定しており，推論精度を引き出すための調整を必要としないと言える．

表 4.8: STL-10 データセットにおける Progressive Learning の訓練延長結果

モデル	訓練方法	epoch 数	Top-1（%）	相対訓練速度（倍）
BN_3	標準	90	80.6	1
NF_9	標準	90	81.7	0.52
BN_3	Progressive	90	64.8	1.76
BN_3	Progressive	180	68.1	0.87
BN_3	Progressive	360	76.1	0.44
BN_3	Progressive	720	70.1	0.22
NF_9	提案手法	90	78.6	9.83
NF_9	提案手法	180	81.7	4.91

4.4.4 提案事項の一部を除いた場合の比較

ここまで，本章における提案事項すべてを適用した場合の結果のみを見てきたが，ここでは各提案事項が推論精度や訓練速度に及ぼす影響について述べる．CIFAR-100 データセットを用いて実験を行った結果を表 4.9 に示す．表中では，本章における提案事項とは条件が異なる部分に下線を引いており，また，各事例で最も得意とする（推論精度が高い）スケールにおける推論精度を太字で表示している．いちばん上の事例が．通例通り畳み込みフィルタの生成を行わず，すべて 1/1 スケールでモデルを訓練する標準的な方法，つまり，これまで「訓練方法：標準，モデル：BN_3，epoch 数：90」とした事例である．また，もっとも下の 2 例がこれまで「提案手法」としてとりあげてきたものであり，訓練の延長を行うかどうかだけが異なる．

まず，フィルタ生成を行わない上の 3 例に注目する．これはつまり，任意の画像サイズに対して同一の畳み込みフィルタを用いることを意味する．3 例で異なるのは訓練に用いた画像のサイズであり，いずれも訓練したスケールにおける推論精度がほかのスケールに比べて顕著に高い．したがって，フィルタ生成を行わない，標準的な畳み込みフィルタの運用方法では訓練したスケール以外の画像を推論する際の精度に期待することができない．また，CIFAR-100 データセットの原画像サイズは 32×32 であるにもかかわらず，訓練する画像サイズが大きいほど推論結果が高いのは特筆すべきである．つまり，推論精度の高さを競う上ではある程度大きい画像サイズに対する推論を想定したほうが有利であると考えられる．

次に，フィルタ生成を取り入れ，1/4 スケールのみで訓練した 4 つ目の事例に注目する．最も得意とするスケールが訓練した 1/4 であることには変わりないが，スケール間の推論精度差が小さくなっていることがわかる．これにより，フィルタ生成がスケール間のギャップを大きく改善し，小さいサイズの画像を用いた訓練に貢献していると言える．また，特筆すべき点として，1/4 スケールにおける訓練速度にも改善がみられる．これは，フィルタ生成を行う際には画像に合わせて小さいフィルタを用いるため，特徴マップの各位置における重み付き和の計算コストを下げることができるためであると考えられる．しかしながら，いちばん上の標準的な事例と比較した場合，最も得意とするスケールにおける精度（79.5%→ 75.8%）が十分とはいえず，これによって標準的な訓練の代替とするのは難しい．

また，提案手法のうち一つだけ提案事項を取り除いた，5 から 8 例目に注目する．5 例目は「Batch Normalization の除去（4.3.1 節）」，6 例目は「最低解像度の設定（4.3.2 節）」，7 例目は「混合スケール学習（4.3.3 節）」，8 例目は「訓練の延長（4.3.4 節）」をそれぞれ取り除いたものである．このうち，スケール間ギャップを軽減するために導入した提案事項が抜けている 5 から 7 例目にかけては，重点的に訓練した小さいサイズの画像の推論を得意とする．顕著なのは 5 例目と 7 例目であり，小さい画像とフィルタを用いた訓練において「Batch Normalization の除去」と「混合スケール学習」は必須であると言える．一方，「最低解像度の設定」を行わない 6 例目は比較的 1/1 スケールにおける推論精度も高くなっているが，8 例目のように最低解像度を設定し，訓練の延長を行わないほうが 1.70 倍高速かつ高精度（77.5%→ 78.8%）な訓練を実現できる．続く 8 例目で取り除いた「訓練の延長」は 9 例目と比較することにより各スケールにおける推論精度を改善している．ただし，訓練速度は epoch 数に反比例して小さくなるため，推論精度と訓練速度のバランスをとって設定することが望ましい．

最後に，提案事項をすべて導入した 9 例目に注目する．9 例目は 1/4，1/2，1/1 スケールのいずれにおいても上位 2 番目以内の精度を誇っており，1 位との差は最大でも 0.4%（77.7%→ 77.3%；1/4 スケール）である．つまり，スケール間ギャップを小さくし，さまざまなスケールの画像を織り交ぜて訓練することにより，各スケールにおいて推論精度の高いモデルの獲得に成功している．これはたとえば推論時においても計算コストと精度のトレードオフを考慮して入力に用いる画像サイズを変えるという応用を可能にする．各スケールに特化したモデルを用いても同様のことは実現できるが，上の 3 例のように通常のモデルをそのまま各スケールで訓練した場合と比較しても，同等以上の結果が得られている．1/4 スケールや 1/2 スケールにおける提案手法の優位性は，2 例目

3 例目のモデルとして 1/1 スケールのものをそのまま利用したためである可能性があるが，こうしたスケールごとの調整をすることなく，1/1 スケール向けのモデルの高速な訓練の副産物として，より小さいスケール向けのモデルが得られるというのも本手法の強みであると言える．ただし，スケール間ギャップは改善されたものの，依然として入力画像のサイズが大きい場合のほうが推論精度が高くなっており，さらなる改善の余地がある．

表 4.9: 提案事項の一部を除いた場合の比較（CIFAR-100）

	条件					結果				
事例	ﬁlタ生成	モデル	最低解像度	スケール比 1/4 : 1/2 : 1/1	epoch 数	Top-1（%） 1/4	1/2	1/1	訓練速度 s/epoch	相対
1	×	BN_3	1×1	0 : 0 : 64	90	3.0	28.7	**79.5**	1165	1
2	×	BN_3	1×1	0 : 64 : 0	90	29.9	**78.2**	46.6	238	4.89
3	×	BN_3	1×1	64 : 0 : 0	90	**75.1**	12.0	0.8	90	12.94
4	✓	BN_9	1×1	64 : 0 : 0	90	**75.8**	70.0	64.4	66	17.65
5	✓	BN_9	7×7	56 : 7 : 1	180	**76.0**	74.8	72.5	133	4.37
6	✓	NF_9	1×1	56 : 7 : 1	180	74.7	**77.5**	77.3	101	5.77
7	✓	NF_9	7×7	64 : 0 : 0	180	**77.7**	72.3	70.3	84	6.93
8	✓	NF_9	7×7	56 : 7 : 1	90	75.9	77.4	**78.8**	124	9.40
9	✓	NF_9	7×7	56 : 7 : 1	180	77.3	78.5	**79.4**	126	4.62

4.4.5 3×3 畳み込みの拡大を行わない場合の比較

ここまでの実験では BN_3 や NF_3 における $2\downarrow$ の 3×3 畳み込みフィルタを 9×9 に拡大した BN_9 や NF_9 を用いて提案手法のフレームワークの結果を示してきた．これは 1/4 スケールで訓練するために，これ以上サイズを小さくできない 3×3 の代わりとして 9×9 のものを用い，1/4 スケールにおいて 3×3 フィルタとなるよう意図したものである．しかしながら，一方で 1×1 のフィルタについては手を加えておらず，任意の画像サイズに対して 1×1 の固定長としても問題ないものと思われる．そこで，本節では 3×3 の畳み込みを置き換えるモデル改変が必要かどうかについて述べる．

表 4.10 に実験結果を示す．この実験ではダウンサンプリング時に用いられる 3×3 の畳み込みを 9×9 としないモデル（NF_3）を用意し，また，この 3×3 畳み込みに関してはフィルタの生成を行わないこととした．つまり，入力として用いる画像サイズによらず常に 3×3 の畳み込みフィルタとして利用される．これはモデル全体に存在する 1×1 の畳み込みフィルタと同じ挙動である．その他の訓練条件は提案事項すべてを適用しており，表中の下線部が本章の提案事項と異なる部分を強調している．また，各事例で最も得意とするスケールにおける推論精度を太字で強調した．

いずれのデータセットにおいても，3×3 の静的フィルタを含む方法では，1×1 を除くすべてのフィルタを生成した場合よりも推論精度が落ちる．その傾向は特に 1/1 スケールにおいて見られる

表 4.10: ダウンサンプリング用の 3×3 畳み込みを生成せず，そのまま利用する場合の比較

条件				結果				
モデル	フィルタ生成		データセット	Top-1（%）			訓練速度	
	Stage	その他		1/4	1/2	1/1	s/epoch	相対
$\underline{\mathrm{NF}_3}$	✓	$\underline{\times}$	CIFAR-10	94.5	**95.4**	92.4	106	5.51
NF_9	✓	✓	CIFAR-10	94.7	95.5	**95.7**	123	4.75
$\underline{\mathrm{NF}_3}$	✓	$\underline{\times}$	CIFAR-100	76.7	**78.5**	73.2	107	5.44
NF_9	✓	✓	CIFAR-100	77.3	78.5	**79.4**	126	4.62
$\underline{\mathrm{NF}_3}$	✓	$\underline{\times}$	STL-10	78.7	**79.5**	67.4	10	5.90
NF_9	✓	✓	STL-10	78.5	80.9	**81.7**	12	4.91

が，これはおそらく 3×3 のフィルタが高頻度で学習している 1/4 スケールや 1/2 スケール向けに最適化された結果であると考えられる．このため，1/1 スケール用のモデルの学習としては提案手法よりも劣っており，また，得意なスケールにおける精度もすべて提案手法を下回っている．したがって，本章の提案フレームワークを利用する際には 1×1 を除くすべてのフィルタを特徴マップの大きさに合わせて生成する必要があり，また，縮小不可能なほどに小さいフィルタは適宜拡大することが望ましい．

4.4.6 訓練時における計算量の比較

本章の提案する訓練フレームワークにおいて高速化が行われるのは，主に訓練時における深層学習モデルの計算量が削減されるためである．浮動小数点数の演算総数を表す FLOPs（the number of floating point operations）は論文ごとに算出方法が異なるなど統一感に欠け，また，演算の並列性や実行時コンパイルにおける計算グラフの最適化を考慮しない指標であるが，広く用いられているため試算結果をここに報告する．なお，本論文においては加算と乗算の総数を表したものを指し，その結果，RepLKNet の原典 [29] の記載と比較した場合，約 2 倍の数値となる．また，活性化層等における非線形関数などは考慮しないこととする．

表 4.11 に，4.4.2 節で比較した事例の期待計算量（MFLOPs）を示す．ここで，M は 10^6 を示す Mega の略である．深層学習モデルの改変に関する研究では基本的に

- 訓練時と推論時で（共通に用いるモデルの）計算量は変わらない．
- ミニバッチの計算量はミニバッチサイズ（bs）に比例する．

といった状況が多いため，（モデル，入出力のサイズ）の組と計算量が 1 対 1 に対応した示し方をする．しかしながら，本研究においては上記 2 つは成立しない．そこで，以下では 1 サンプルあたりの期待計算量の算出の際の考え方を述べながら比較を行う．なお，表中の値は CIFAR-100 に対して算出したものであり，厳密には CIFAR-10 や STL-10 と異なるが，その違いは 0.2 MFLOPs 程度であるため割愛する．

表 4.11: モデルや訓練方法の違いによる期待計算量（MFLOPs）の比較

条件		1 サンプルあたりの期待計算量			
モデル	訓練方法	訓練時（bs=32）		推論時（224×224）	
		MFLOPs	相対	MFLOPs	相対
BN_3	標準	30,917	1	30,917	1
NF_3	標準	30,684	0.99	30,667	0.99
BN_9	標準	31,273	1.01	31,273	1.01
NF_9	標準	31,040	1.00	31,023	1.00
BN_3	Progressive	18,685	0.60	30,917	1
NF_9	Progressive	18,758	0.61	31,023	1.00
NF_9	提案手法	7,779	0.25	31,023	1.00

本章の提案する訓練方法において，訓練時と推論時では主に以下の 2 点の違いがある．

1. 訓練時には推論時よりも小さい画像を用いる．
2. 入力する画像のサイズが決まっているとき，推論時に用いるフィルタはあらかじめ計算できる．

1. について，本手法は推論時よりも小さい画像を中心に用いた低コストな訓練を提案したものであるため，推論時とは計算量が大きく変わる．実際，表中の推論時の計算量は相対値 1.00 となっており，標準的な方法で訓練したものと変わらない．一方で，訓練時の相対値は 0.25 となっており，推論時の約 1/4 程度で訓練できていることがわかる．同様に，Progressive Learning も訓練に用いる画像を徐々に大きくしていき，最終的に推論時と同じにするものであるため，最初から推論時と同じ大きさの画像を用いるよりも計算コストが低い．このことが表中にも表れており，推論時における計算量は標準的な訓練方法のものと変わらないが，訓練時における相対値は約 0.6 倍となっている．ただし，提案手法と比較した場合，Progressive Learning の訓練時における期待計算量は 2.4 倍あり，提案手法が訓練時における計算量を大幅に削減していることがわかる．このように，Progressive Learning も提案手法も複数のサイズの画像を利用するため，訓練時における計算量はその使用割合に応じて求めた期待値とするのが妥当である．

また，2. について，本手法では画像のサイズに合わせてフィルタの生成を行う．訓練時には学習可能パラメータに損失関数から生じる勾配を逆伝播させる都合上，微分可能な処理によってフィルタを訓練ステップごとに算出する必要があるが，推論時においては 1 度計算したフィルタを再利用することができる．このことを反映し，表 4.11 の推論時における計算量にはフィルタの生成にかかるコストを算入していない．

訓練時におけるフィルタの生成にかかるコストは，サンプルごとではなく，ミニバッチ（より正確には 1 訓練ステップ）単位で生じる．したがって，1 回のフィルタ生成に対し，これを用いるサンプル数が多ければ多いほど，それに反比例して 1 サンプルあたりのフィルタ算出コストは低くな

る．すなわち，訓練時における計算量は「サンプルごとにかかるコスト」と「ミニバッチ内で共有できるコスト」にわけられ，前者にサンプル数を乗じ，後者を加えたものが 1 訓練ステップに必要なものとなる．これをミニバッチサイズで割ることにより，1 サンプルあたりの計算量を求めることができる．表 4.11 においては，実験で用いた値としてミニバッチサイズ（bs）を 32 とした場合の計算量をもとに算出している．

サンプルごとにかかるコストとミニバッチで共有されるコストの詳細を表 4.12 に示す．ミニバッチで共有されるコストを生じさせるものとしては，提案手法におけるフィルタのリサイズ処理のほかに，NF-ResNet [91] の方法で Batch Normalization を取り除く場合に生じる Scaled Weight Standardization がある．したがって，表中では NF とついたモデルに「ﾊﾞｯﾁ共有」のコストが発生している．ただし，いずれも推論時には事前計算を行うことが可能であるため，推論時の計算量としては「ｻﾝﾌﾟﾙごと」のもののみを参照するのが適当である．なお，訓練時における「ﾊﾞｯﾁ共有」のコストはミニバッチサイズ（bs）が大きくなるほど小さくなることも表中に見られる．実験で用いた（b=32）の場合においては，この「ﾊﾞｯﾁ共有」のコストが 4.4.2 節における事例のすべてで 0.5%にも満たないことが示されている．つまり，フィルタの生成にかかるコストは訓練時のミニバッチサイズがある程度大きければ無視できる程度に小さくなる．

表 4.12: 本実験で用いた実験事例ごとの計算量（MFLOPs）の詳細

条件				計算量（MFLOPs）		バッチ共有オーバーヘッドが占める割合（%）					
モデル	入力	ﾌｨﾙﾀ生成	最低解像度	ｻﾝﾌﾟﾙごと	ﾊﾞｯﾁ共有	bs=1	bs=2	bs=4	bs=8	bs=16	bs=32
BN_3	224×224		1×1	30,917	0	0	0	0	0	0	0
NF_3	224×224		1×1	30,667	548	1.76	0.89	0.44	0.22	0.11	0.06
BN_9	224×224		1×1	31,273	0	0	0	0	0	0	0
NF_9	224×224		1×1	31,023	549	1.74	0.88	0.44	0.22	0.11	0.06
BN_3	98×98		1×1	7,562	0	0	0	0	0	0	0
BN_3	140×140		1×1	12,882	0	0	0	0	0	0	0
BN_3	182×182		1×1	22,268	0	0	0	0	0	0	0
NF_9	98×98		1×1	7,577	549	6.76	3.50	1.78	0.90	0.45	0.23
NF_9	140×140		1×1	12,921	549	4.08	2.08	1.05	0.53	0.26	0.13
NF_9	182×182		1×1	22,329	549	2.40	1.21	0.61	0.31	0.15	0.08
BN_9	56×56	✓	1×1	2,058	431	17.32	9.48	4.98	2.55	1.29	0.65
BN_9	112×112	✓	1×1	7,026	385	5.19	2.67	1.35	0.68	0.34	0.17
BN_9	224×224	✓	1×1	31,273	368	1.16	0.58	0.29	0.15	0.07	0.04
NF_9	56×56	✓	1×1	2,040	1,006	33.03	19.78	10.98	5.81	2.99	1.52
NF_9	112×112	✓	1×1	6,963	960	12.12	6.45	3.33	1.69	0.85	0.43
NF_9	224×224	✓	1×1	31,023	942	2.95	1.50	0.75	0.38	0.19	0.09
BN_9	56×56	✓	7×7	7,299	386	5.02	2.58	1.30	0.66	0.33	0.16
BN_9	112×112	✓	7×7	8,446	383	4.34	2.22	1.12	0.56	0.28	0.14
BN_9	224×224	✓	7×7	31,273	368	1.16	0.58	0.29	0.15	0.07	0.04
NF_9	56×56	✓	7×7	7,255	961	11.70	6.21	3.21	1.63	0.82	0.41
NF_9	112×112	✓	7×7	8,379	958	10.26	5.41	2.78	1.41	0.71	0.36
NF_9	224×224	✓	7×7	31,023	942	2.95	1.50	0.75	0.38	0.19	0.09

4.4.7 特徴マップの統計量のスケール間ギャップの比較

本章の提案する訓練フレームワークにおいて重要なのは「異なるスケール間で相似な特徴マップを処理すること」である．そこで，どの程度相似な特徴マップを処理できているのかを評価するために特徴マップの平均値と標準偏差を検証用サンプルごとにプロットし，スケールごとに色分けを行ったものを観察する．各散布図において各点が検証用サンプル 1 例を表しており，1,000 サンプル程度を抜粋している．

表 4.13 はクラス分類器における各 Stage 直後の特徴量について順次プロットを行ったものである．通常の訓練や Progressive Learning によって得られたモデルでは，処理が進むにしたがってスケール間のギャップが肥大化し，最終的にはまったく異なる分布となっている．これは 4.2.4 節で考察したように，サイズの異なる画像に対して同じ畳み込みフィルタをリサイズせずに畳み込んだ出力は相似にならないということを示している．一方で，提案手法のフレームワークで訓練した場合，分布はスケール間でおよそ重なったものとなる．つまり，提案手法は意図した通り相似な特徴マップを処理していくことによって小さい画像による訓練を実現していることがわかる．

また，表 4.14 には提案手法にあげた工夫のうちいずれかひとつを取り除いたものを示す．全体として表 4.13 に示した通常の訓練方法よりも分布の重なりは大きいが，Batch Normalization の除去や最低解像度の設定，混合スケール学習を行わなかった場合には分布の重なりがやや小さくなる．したがって，これらの提案はスケール間で相似な学習を行うことに貢献していると言える．Batch Normalization を用いた場合に生じる差は当然のことながら Batch Normalization によるものであり，Batch Normalization も処理する画像のサイズによって推論精度が左右されることが見てとれる．最低解像度を設定しない場合，特に緑で示された 1/4 スケールの特徴マップは処理が進むにつれ 4×4 や 2×2 など局所性を損ないかねないほど小さいサイズとなる．実際，分布図を確認しても Stage 2 の直後まではスケール間ギャップが認められず，4×4 の特徴マップを中心に訓練した Stage 3 の直後からギャップが生じ始める．このことから，このモデル・タスクにおいて最低解像度を 7×7 に設定したことは妥当であると考えられる．混合スケール学習を行わない場合の訓練対象は散布図上緑で示された 1/4 スケール（56×56）の画像のみである．Stage 3 直後の画像に見られるように，途中までは緑の分布のみが離れた状態にあることが特徴的である．スケジュール延長はスケール間ギャップを減らすための提案ではないため，その有無にかかわらず同じように分布が重なる．

以上のように本章の提案事項は意図した通り，学習を行う小さい空間スケールと，実際に処理したいターゲットスケールの特徴マップを相似な状態で処理することでスケール間ギャップを軽減していることが定性的にも示されたと言える．特に，今後畳み込みを中心としない ViT や MLP 系統の深層学習モデルの訓練時に小さい画像の使用を試みる場合においても，最小解像度の設定や混合スケール学習を行うことは検討に値する．

表 4.13: 特徴マップの平均値・標準偏差を散布図にしたもの（訓練方法間の比較）

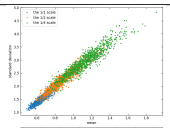
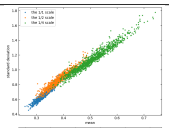
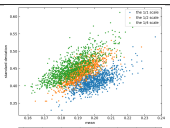
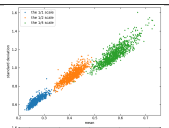
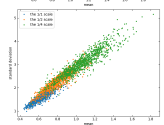
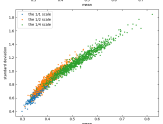
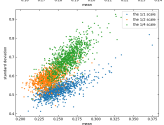
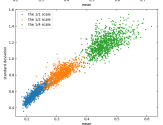
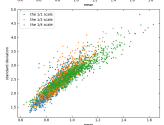
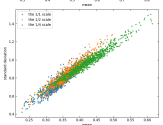
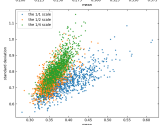
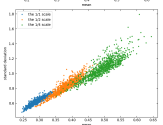
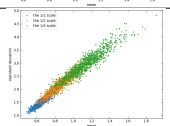
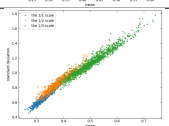
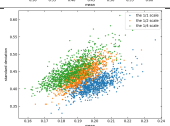
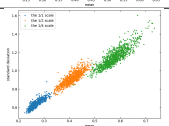
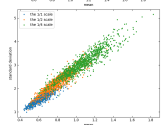
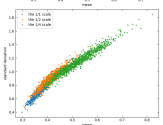
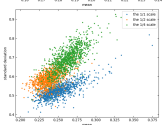
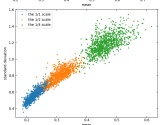
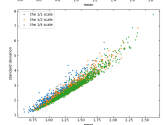
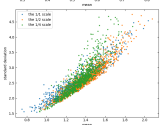
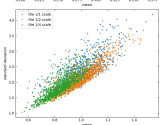
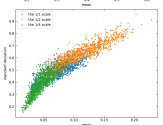
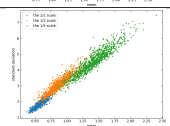
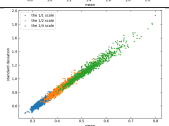
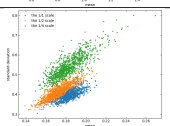
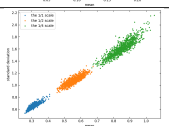
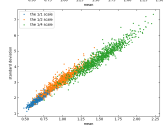
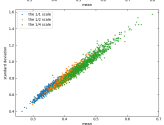
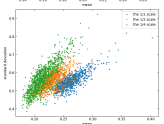
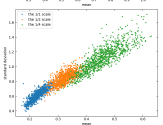
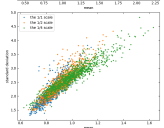
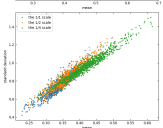
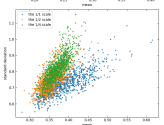
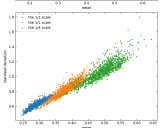
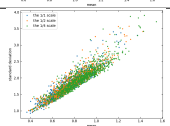
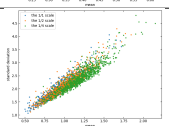
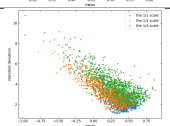
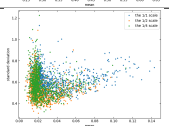
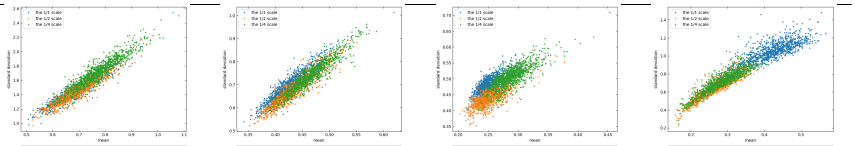
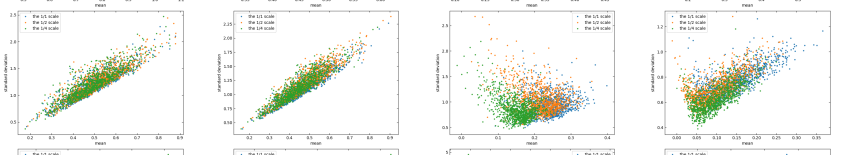
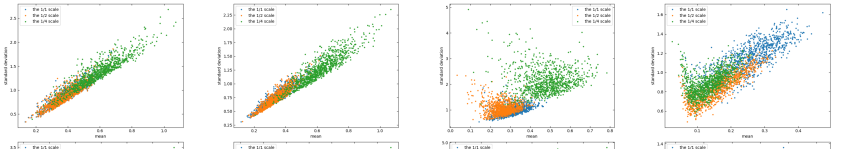
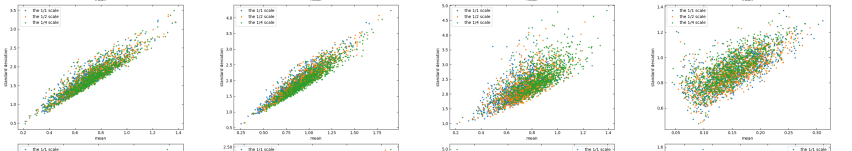
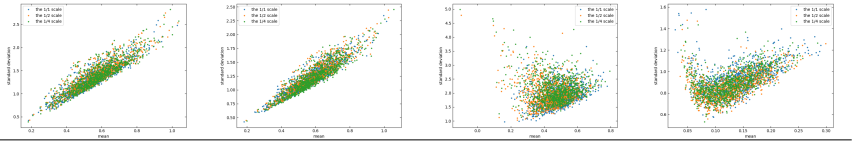
訓練方法（モデル） @ データセット	特徴マップ検証点 Stage 1 直後	Stage 2 直後	Stage 3 直後	Stage 4 直後
標準（BN_3） @ CIFAR-10				
標準（BN_3） @ CIFAR-100				
標準（BN_3） @ STL-10				
標準（NF_9） @ CIFAR-10				
標準（NF_9） @ CIFAR-100				
標準（NF_9） @ STL-10				
Progressive（BN_3） @ CIFAR-10				
Progressive（BN_3） @ CIFAR-100				
Progressive（BN_3） @ STL-10				
提案手法（NF_9） @ CIFAR-10				
提案手法（NF_9） @ CIFAR-100				
提案手法（NF_9） @ STL-10				

表 4.14: 特徴マップの平均値・標準偏差を散布図にしたもの（提案事項を一部除いた場合の比較）

訓練方法	**特徴マップ検証点**			
w/o 除いた提案事項	**Stage 1 直後**	**Stage 2 直後**	**Stage 3 直後**	**Stage 4 直後**
提案手法 w/o BatchNorm 除去				
提案手法 w/o 最低解像度				
提案手法 w/o 混合スケール				
提案手法 w/o 訓練延長				
提案手法 すべて適用				

4.5 第 4 章のまとめ

第 4 章では訓練の高速化に焦点を当て，異なるスケールの相似な入力に対する処理結果が相似であるような畳み込みを実現するための方法を研究した．はじめに，画像に対する相似の概念を導入し，サイズの異なる相似な画像に対する畳み込みについての考察を行った．結果として，サイズの異なる相似な画像に対し，同一のフィルタを畳み込んだ際，出力結果が相似とならないことを理論的にも実験的にも示した．また，相似な結果を得るための条件としてフィルタのサイズを画像のサイズに比例して変化させること，フィルタの要素数に反比例するスケーリングを行うことなどを提案した．しかしながら，小さい画像を用いた訓練は入出力の相似性を維持するために畳み込み以外の点においても工夫する必要があり，考察の結果，Batch Normalization を除去すること，特徴マップの最低解像度を定めること，低頻度で大きなサイズの画像を訓練に用いる混合スケール学習を提案した．これにより，高速かつ精度の劣化の小さい訓練フレームワークを実現した．また，高速化により多くのステップ数を短時間で訓練できるようになったぶん，訓練ステップ数を増やすことも提案した．訓練ステップを 2 倍にしても標準的な方法より 4 倍程度高速に訓練することができ，また，これによって比較対象とした（小受容野ブランチを取り除いた）RepLKNet-31B と同等の精度をもつ訓練結果が得られた．

先行研究の Progressive Learning と比較した場合，本章の提案フレームワークではさらに低コストな訓練を実現できた．Progressive Learning において用いられるターゲットスケールの画像は 1/4 程度を占めるが，提案手法においてはさらにその 1/16 の 1/64 しか用いない．また，画像を小さくした際に，フィルタサイズも小さくしているため，小さい画像の処理においても畳み込みの計算コストを低減している．さらに，Progressive Learning では訓練する画像サイズの変更に伴ってフィルタをその画像サイズに適応させる処理が必要となるが，提案手法はフィルタを画像サイズに合わせて適応させるため，常にターゲットスケールと訓練スケールの間のギャップを小さく保っているため，訓練ステップ数が小さい場合においても十分な効果を発揮する．

しかしながら，広く用いられる 3×3 の畳み込みフィルタに対しては本章の提案する訓練方法を適用をすることができないという欠点がある．これは 3×3 のフィルタのサイズを画像サイズに合わせてさらに小さくすることが難しいためである．この性質により，実験したい深層学習モデルにおける 3×3 の畳み込みは 5×5 や 9×9 のものに置き換えられることが求められ，推論時の計算コストが増大する．ただし，フィルタサイズの大きい畳み込みに注目が集まる昨今の研究動向を踏まえると，これが欠点になりうるかどうかの結論は今後の画像処理分野の研究を待たねばならない．少なくとも，フィルタサイズの大きい畳み込みを用いた深層学習モデルを検討したい場合には，本章の提案したフレームワークを適用することにより効率的な実験が可能になることが期待できる．

第 5 章　結論

本論文では画像処理深層学習モデルの代表的な Token Mixer のひとつである畳み込みに注目して研究を行った．畳み込みは他の Token Mixer よりも強い帰納バイアスを持っており，少ないデータ数，ステップ数で訓練することができる．このように，畳み込みは小規模な訓練環境に適した特徴を持っているが，近年ではフィルタサイズの大きい畳み込みが注目されており，パラメータ数や計算コストが増加するなど，訓練に必要な訓練環境は大規模化が進んでいる．そこで，本研究では畳み込みフィルタの生成方法を研究し，フィルタサイズの大きい畳み込みを含むモデルのパラメータ数や訓練時間を削減した．

畳み込みフィルタは重みを表すパラメータの配列として表現され，同一視されるのが一般的である．第 2 章ではこれを一般化し，パラメータから関数によってフィルタの重みを計算することをフィルタの生成と定義した．フィルタの生成関数が微分可能であれば，誤差逆伝播法のフレームワークで最適化できる．フィルタの生成には追加の計算コストが必要であるが，ミニバッチのサンプル数が大きくなるほど，その影響は小さくなる．こうした基礎の下，第 3 章と第 4 章ではそれぞれパラメータ削減，訓練時間短縮のための具体的な生成方法を検討した．

第 3 章では，学習済み畳み込みフィルタの主成分分析の結果を定式化し，そのフィルタ成分をパラメータで重み付けて線形結合することで畳み込みフィルタを生成した．主成分を近似した連続フィルタ成分の線形結合としてフィルタを表現するため，必要なパラメータ数をフィルタの要素数よりも少なく，一定にすることができる．実験では最小 6 個のパラメータで標準的な 3×3 フィルタと同等以上の精度のモデルを獲得することができた．モデルによっては勾配爆発により訓練ができないこともあるが，既存の N-Jet 成分系よりも訓練可能なモデルが多い．また，N-Jet 成分系による訓練が成功した場合においても，遜色ない結果が得られることを示した．少数のフィルタ成分で十分に高精度なフィルタが得られることや成分の重要性にもとづく成分スケーリングが有効なことから，設計時の趣旨に沿った性質を持つフィルタ成分系が得られたと言える．

第 4 章では，パラメータの 2 次元配列を縮小し，同じ比率で縮小した特徴マップに対する出力が元のサイズと相似になるようなフィルタの生成方法を提案した．訓練時に使用する画像とフィルタサイズを 1/4 程度に縮小することで訓練時間を短縮することができる．しかしながら，これだけでは小さい画像を用いた訓練において精度の劣化が生じる．そこで，画像スケール間のギャップに注目して考察し，3 つの改善案を提示した．3 つの改善案を適用し，訓練ステップ数を増加させることにより，標準的な方法よりも 4.6 倍以上高速に同等以上の結果が得られることを示した．また，小さい画像から訓練を始めるが，フィルタサイズの変更は行わない Progressive Learning と比較した場合も，提案手法によってさらなる高速化が実現されていることや，Progressive Learning がうまくいかない状況においても，提案手法が標準的な訓練方法と同等以上の推論精度を達成することを示した．これらのことから，小さい画像に合わせたフィルタの生成と，スケール間ギャップを軽

減する改善案等により，推論時に処理したい画像サイズよりも縮小した画像を中心に使用してモデルを訓練することができたと言える．

以上のように，従来パラメータの配列と同一視されてきた畳み込みフィルタの生成方法を変更することによって，パラメータ数の削減や訓練時間の短縮といったメリットを享受することが可能である．特にフィルタサイズを大きくした際には，そのメリットが顕著になる．つまり，近年注目が集まるフィルタサイズの大きい畳み込みが含まれるモデルを小規模な訓練環境で訓練する場合，本研究で提案したように畳み込みフィルタを生成して訓練する方法が有効である．

今後の展望として，第 3 章で提案したような SRF の成分系を第 4 章の方法でリサイズしながら訓練する応用が挙げられる．必要なパラメータ数と 1 ステップ当たりの計算量の両方が削減され，訓練時間のさらなる短縮が見込まれる．特に，サイズの大きい畳み込みフィルタの訓練に適した成分系が開拓された場合の恩恵は大きい．サイズの小さい畳み込みフィルタを使用した場合と比較して，精度向上が期待できるだけではなく，入力画像を縮小して訓練することが可能なため訓練時間の面でも有利になる．この場合，フィルタサイズの大きい畳み込みを採用することが精度の高いモデルを短時間で獲得することに繋がる．以上，畳み込みフィルタの生成に関する研究が今後さらなる発展と応用を遂げることを期待し，本論文の結びとする．

参考文献

[1] Luc Florack, Bart ter Haar Romeny, Max Viergever, and Jan Koenderink. “The Gaussian scale-space paradigm and the multiscale local jet”. *International Journal of Computer Vision*, Vol. 18, No. 1, pp. 61–75, 1996.

[2] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Gabriel J. Brostow, Julien Fauqueur, and Roberto Cipolla. “Semantic object classes in video: A high-definition ground truth database”. *Pattern Recognition Letters*, Vol. 30, No. 2, pp. 88–97, 2009. Video-based Object and Event Analysis.

[4] Kilian Batzner, Lars Heckler, and Rebecca König. “EfficientAD: Accurate Visual Anomaly Detection at Millisecond-Level Latencies”. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 128–138, 2024.

[5] Yezi Ali Kadhim, Muhammad Umer Khan, and Alok Mishra. “Deep Learning-Based Computer-Aided Diagnosis (CAD): Applications for Medical Image Datasets”. *Sensors*, Vol. 22, No. 22, 2022.

[6] Siddharth Srivastava and Gaurav Sharma. “OmniVec: Learning robust representations with cross modal sharing”. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1236–1248, 2024.

[7] Zhuofan Zong, Guanglu Song, and Yu Liu. “DETRs with Collaborative Hybrid Assignments Training”. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6748–6758, 2023.

[8] Peng Wang, Shijie Wang, Junyang Lin, Shuai Bai, Xiaohuan Zhou, Jingren Zhou, Xinggang Wang, and Chang Zhou. “ONE-PEACE: Exploring One General Representation Model Toward Unlimited Modalities”. *arXiv preprint arXiv:2305.11172*, 2023.

[9] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. “Restormer: Efficient Transformer for High-Resolution Image Restoration”. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5728–5739, 2022.

[10] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. “ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation”. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, pp. 38571–38584, 2022.

[11] Weichen Fan, Jinghuan Chen, Jiabin Ma, Jun Hou, and Shuai Yi. “StyleFlow For Content-Fixed Image to Image Translation”. *arXiv preprint arXiv:2207.01909*, 2022.

[12] Yuanhao Cai, Hao Bian, Jing Lin, Haoqian Wang, Radu Timofte, and Yulun Zhang. “Retinexformer: One-stage Retinex-based Transformer for Low-light Image Enhancement”. In *Pro-*

ceedings of the IEEE/CVF International Conference on Computer Vision, pp. 12504–12513, October 2023.

[13] Haitian Zheng, Zhe Lin, Jingwan Lu, Scott Cohen, Eli Shechtman, Connelly Barnes, Jianming Zhang, Ning Xu, Sohrab Amirghodsi, and Jiebo Luo. "Image Inpainting with Cascaded Modulation GAN and Object-Aware Training". In *Proceedings of the European Conference on Computer Vision*, pp. 277–296. Springer, 2022.

[14] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. "Pre-Trained Image Processing Transformer". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12299–12310, 2021.

[15] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". *Neural Computation*, Vol. 1, No. 4, pp. 541–551, 1989.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". *arXiv preprint arXiv:2010.11929*, 2020.

[17] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. "MetaFormer Is Actually What You Need for Vision". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10819–10829, 2022.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[19] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. "MLP-Mixer: An all-MLP Architecture for Vision". In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 34, pp. 24261–24272, 2021.

[20] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. "Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9127–9135, 2018.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "ImageNet Large Scale Visual Recognition Challenge". *International Journal of Computer Vision*, Vol. 115, pp. 211–252, 2015.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, pp. 770–778, 2016.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "ImageNet: A large-scale hierarchical image database". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

[24] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 843–852, 2017.

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

[26] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In *Proceedings of the International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

[27] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. "Training data-efficient image transformers & distillation through attention". In *Proceedings of the International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

[28] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. "Designing Network Design Spaces". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10428–10436, 2020.

[29] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. "Scaling Up Your Kernels to 31x31: Revisiting Large Kernel Design in CNNs". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11963–11975, 2022.

[30] Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. "S^2-MLPv2: Improved Spatial-Shift MLP Architecture for Vision". *arXiv preprint arXiv:2108.01072*, 2021.

[31] Dongyoon Han, Jiwhan Kim, and Junmo Kim. "Deep Pyramidal Residual Networks". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5927–5935, 2017.

[32] Weihao Yu, Chenyang Si, Pan Zhou, Mi Luo, Yichen Zhou, Jiashi Feng, Shuicheng Yan, and Xinchao Wang. "MetaFormer Baselines for Vision". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[33] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. "CoAtNet: Marrying Convolution and Attention for All Data Sizes". In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 34, pp. 3965–3977, 2021.

[34] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". *arXiv preprint arXiv:1803.03635*, 2018.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. "Learning Transferable Visual Models From Natural Language Supervision". In *Proceedings of the International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

[36] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. "Masked Autoencoders Are Scalable Vision Learners". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

[37] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. "ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16133–16142, 2023.

[38] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the Knowledge in a Neural Network". *arXiv preprint arXiv:1503.02531*, 2015.

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". *Nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

[40] Yurii Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence O(1/k^2)". In *Doklady an ussr*, Vol. 269, pp. 543–547, 1983.

[41] Ning Qian. "On the momentum term in gradient descent learning algorithms". *Neural Networks*, Vol. 12, No. 1, pp. 145–151, 1999.

[42] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". *Journal of Machine Learning Research*, Vol. 12, No. 7, 2011.

[43] Matthew D Zeiler. "ADADELTA: An Adaptive Learning Rate Method". *arXiv preprint arXiv:1212.5701*, 2012.

[44] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". *arXiv preprint arXiv:1412.6980*, 2014.

[45] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In *Proceedings of the International Conference on Learning Representations*, 2019.

[46] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. "Adaptive Gradient Methods with Dynamic Bound of Learning Rate". In *Proceedings of the International Conference on Learning Representations*, New Orleans, Louisiana, May 2019.

[47] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. "Sharpness-aware Minimization for Efficiently Improving Generalization". In *Proceedings of the International Conference on Learning Representations*, 2020.

[48] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In *Proceedings of the International Conference on Learning Representations*, 2018.

[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features". In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.

[50] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. "RandAugment: Practical Automated Data Augmentation with a Reduced Search Space". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 702–703, 2020.

[51] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[52] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In *Proceedings of the European Conference on Computer Vision*, pp. 694–711. Springer, 2016.

[53] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[54] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". *Biological cybernetics*, Vol. 36, No. 4, pp. 193–202, 1980.

[55] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 25, 2012.

[57] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018.

[58] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241. Springer, 2015.

[59] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017.

[60] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". *arXiv preprint arXiv:1511.07122*, 2015.

[61] Viacheslav Dudar and Vladimir Semenov. "Use of symmetric kernels for convolutional neural networks". In *Proceedings of the International Conference on Data Science and Intelligent*

Analysis of Information, pp. 3–10, 2018.

[62] Xu Shell Hu, Sergey Zagoruyko, and Nikos Komodakis. "Exploring Weight Symmetry in Deep Neural Networks". *Computer Vision and Image Understanding*, Vol. 187, , August 2019.

[63] Gregory Dzhezyan and Hubert Cecotti. "SymNet: Symmetrical Filters in Convolutional Neural Networks". *International Journal of Machine Learning and Cybernetics*, Vol. 12, , 07 2021.

[64] Jorn-Henrik Jacobsen, Jan van Gemert, Zhongyu Lou, and Arnold W. M. Smeulders. "Structured Receptive Fields in CNNs". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2016.

[65] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31, 2017.

[66] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. "Large Kernel Matters – Improve Semantic Segmentation by Global Convolutional Network". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4353–4361, 2017.

[67] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. "A ConvNet for the 2020s". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

[68] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkainen, Mykola Pechenizkiy, Decebal C Mocanu, and Zhangyang Wang. "More ConvNets in the 2020s: Scaling up Kernels Beyond 51x51 using Sparsity". In *Proceedings of the International Conference on Learning Representations*, 2023.

[69] Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. "CrossFormer: A Versatile Vision Transformer Hinging on Cross-scale Attention". In *Proceedings of the International Conference on Learning Representations*, 2022.

[70] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, and Chunhua Shen. "Conditional Positional Encodings for Vision Transformers". In *Proceedings of the International Conference on Learning Representations*, 2023.

[71] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. "WeightNet: Revisiting the Design Space of Weight Networks". In *Proceedings of the European Conference on Computer Vision*, pp. 776–792. Springer, 2020.

[72] Benjamin Graham and Laurens Van der Maaten. "Submanifold Sparse Convolutional Networks". *arXiv preprint arXiv:1706.01307*, 2017.

[73] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. "3D Semantic Segmentation with Submanifold Sparse Convolutional Networks". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9224–9232, 2018.

[74] Christopher Choy, JunYoung Gwak, and Silvio Savarese. "4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks". In *Proceedings of the IEEE/CVF Conference on*

Computer Vision and Pattern Recognition, pp. 3075–3084, 2019.

[75] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. "Deformable Convolutional Networks". In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 764–773, 2017.

[76] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. "Deformable ConvNets v2: More Deformable, Better Results". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9308–9316, 2019.

[77] Shota Fukuzaki and Masaaki Ikehara. "Principal Components of Neural Convolution Filters". *IEEE Access*, Vol. 10, pp. 104328–104336, 2022.

[78] Shota Fukuzaki and Masaaki Ikehara. "Training Large Kernel Convolutions with Resized Filters and Smaller Images". In *Proceedings of the IEEE Global Conference on Consumer Electronics*, pp. 32–33, 2023.

[79] Shota Fukuzaki and Masaaki Ikehara. "Faster Training of Large Kernel Convolutions on Smaller Spatial Scales". *IEEE Access*, Vol. 12, pp. 161312–161328, 2024.

[80] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic Differentiation in Machine Learning: a Survey". *Journal of Marchine Learning Research*, Vol. 18, pp. 1–43, 2018.

[81] Syed Shakib Sarwar, Priyadarshini Panda, and Kaushik Roy. "Gabor Filter Assisted Energy Efficient Fast Learning Convolutional Neural Networks". In *Proceedings of IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 1–6. IEEE, 2017.

[82] Silvia L Pintea, Nergis Tomen, Stanley F Goes, Marco Loog, and Jan C van Gemert. "Resolution learning in deep convolutional networks using scale-space theory". *IEEE Transactions on Image Processing*, 2021.

[83] Nikhil Saldanha, Silvia L Pintea, Jan C van Gemert, and Nergis Tomen. "Frequency learning for structured CNN filters with Gaussian fractional derivatives". In *Proceedings of the British Machine Vision Conference*, November 2021.

[84] M. Lin, Q. Chen, and S. Yan. "Network in Network". In *Proceedings of International Conference on Learning Representations*, pp. 14–16, April 2014.

[85] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Vol. 2, No. 11, pp. 559–572, 1901.

[86] Christopher M. Bishop. *"Pattern Recognition and Machine Learning"*. Springer, 2006.

[87] Ross Wightman. "PyTorch Image Models". `https://github.com/rwightman/pytorch-image-models`, 2019.

[88] Jan J Koenderink and Andrea J van Doorn. "Representation of local geometry in the visual system". *Biological Cybernetics*, Vol. 55, No. 6, pp. 367–375, 1987.

[89] Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. "High-Performance Large-

Scale Image Recognition Without Normalization". In *Proceedings of the International Conference on Machine Learning*, pp. 1059–1071. PMLR, 2021.

[90] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. "Micro-Batch Training with Batch-Channel Normalization and Weight Standardization". *arXiv preprint arXiv:1903.10520*, 2019.

[91] Andrew Brock, Soham De, and Samuel L. Smith. "Characterizing signal propagation to close the performance gap in unnormalized ResNets". In *Proceedings of the International Conference on Learning Representations*, 2021.

[92] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning Multiple Layers of Features from Tiny Images", 2009.

[93] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely Connected Convolutional Networks". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

[94] Mingxing Tan and Quoc Le. "EfficientNetV2: Smaller Models and Faster Training". In *Proceedings of the International Conference on Machine Learning*, pp. 10096–10106. PMLR, 2021.

[95] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In *Proceedings of the International Conference on Machine Learning*, pp. 448–456. pmlr, 2015.

[96] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 32, 2019.

[97] Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. "Scale-Invariant Convolutional Neural Networks". *arXiv preprint arXiv:1411.6369*, 2014.

[98] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. "Locally Scale-Invariant Convolutional Neural Networks". *arXiv preprint arXiv:1412.5104*, 2014.

[99] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. "Scale-Equivariant Steerable Networks". In *Proceedings of the International Conference on Learning Representations*, 2019.

[100] Rohan Ghosh and Anupam K Gupta. "Scale Steerable Filters for Locally Scale-Invariant Convolutional Neural Networks". *arXiv preprint arXiv:1906.03861*, 2019.

[101] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. "Non-local Neural Networks". In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

[102] Bum Jun Kim, Hyeyeon Choi, Hyeonah Jang, Dong Gu Lee, Wonseok Jeong, and Sang Woo Kim. "Dead Pixel Test Using Effective Receptive Field". *Pattern Recognition Letters*, Vol. 167, pp. 149–156, 2023.

[103] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean,

Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. “TensorFlow: A system for large-scale machine learning”. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, Vol. 16, pp. 265–283, 2016.

[104] “MegEngine:A fast, scalable and easy-to-use deep learning framework”. `https://github.com/MegEngine/MegEngine`, 2020.

[105] Sergey Ioffe. “Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models”. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 30, 2017.

[106] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. “Layer Normalization”. *arXiv preprint arXiv:1607.06450*, 2016.

[107] Yuxin Wu and Kaiming He. “Group Normalization”. In *Proceedings of the European Conference on Computer Vision*, pp. 3–19, 2018.

[108] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. “RepVGG: Making VGG-style ConvNets Great Again”. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13733–13742, 2021.

[109] Adam Coates, Andrew Ng, and Honglak Lee. “An Analysis of Single Layer Networks in Unsupervised Feature Learning”. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011. `https://cs.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf`.

[110] Bum Jun Kim, Hyeyeon Choi, Hyeonah Jang, Dong Gu Lee, Wonseok Jeong, and Sang Woo Kim. “Guidelines for the Regularization of Gammas in Batch Normalization for Deep Residual Networks”. *arXiv preprint arXiv:2205.07260*, 2022.

[111] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. “JAX: composable transformations of Python+NumPy programs”, 2018.

[112] Tensorflow. “XLA”. `https://www.tensorflow.org/xla`, 11 月 2023 年. （アクセス日: 2023 年 12 月 19 日；日付は最終更新日によるもの）.

[113] Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, et al. “A Study of BFLOAT16 for Deep Learning Training”. *arXiv preprint arXiv:1905.12322*, 2019.

[114] 若松寛. “4.4.1　Hermite 微分方程式に関する計算”. `https://www.chem.ous.ac.jp/~waka/compchem/harmonic_oscillator/ho-4-1.html`, 7 月 2007 年. （アクセス日: 2023 年 11 月 23 日）.

謝辞

本研究は，著者が慶應義塾大学大学院理工学研究科の後期博士課程在学中および所定単位取得退学後の 1 年間に行ったものである．

本論文の作成にあたり，幾多のご意見とご指導を賜りました指導教授および本論文の主査である慶應義塾大学理工学部の池原雅章教授に心より感謝申し上げます．著者自身以上に著者に期待を寄せてくださり，実験環境等に惜しげもなくご投資をいただいたおかげで様々なアイデアを試すことができました．また，著者が所定単位取得退学したあとも，在学中と変わらず学位取得に向けて辛抱強くご指導いただきありがとうございます．7 年間たいへんお世話になりました．

また，ご多忙な中，本論文の副査を快く引き受けてくださった慶應義塾大学理工学部の萩原将文教授，青木義満教授，久保亮吾教授に深く御礼申し上げます．全体の構成から細部の表現に至るまで詳細にご検討いただいたおかげで本論文を現在の形にまとめることができたと考えております．

最後に，学業に専念できる環境を整え，応援してくれた両親に最大限の感謝を伝えたい．

付録 A　OtX 成分系の直交性に関連した定理等の数学的証明

A.1　Hermite 多項式に関する定理

以下の記述の骨子は [114] による.

A.1.1　微分方程式を用いた Hermite 多項式の導入

微分方程式

$$\frac{\mathrm{d}y}{\mathrm{d}x} + 2xy = 0 \tag{A.1}$$

の解は

$$y = c\exp(-x^2)$$

である．実際 $\dfrac{\mathrm{d}y}{\mathrm{d}x} = -2cx\exp(-x^2)$，$2xy = 2cx\exp(-x^2)$ より左辺の和は 0 である.

ここで $n \in \mathbb{N}$ を導入し，式 (A.1) の両辺をさらに $(n+1)$ 回 x で微分すると

$$\frac{\mathrm{d}^2 z}{\mathrm{d}x^2} + 2x\frac{\mathrm{d}z}{\mathrm{d}x} + 2(n+1)z = 0, \quad \text{where. } z = \frac{\mathrm{d}^n y}{\mathrm{d}x^n} = c\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2) \tag{A.2}$$

が得られ，z が適当な x の n 次多項式 $u_n(x)$ によって

$$z = u_n(x)\exp(-x^2) \tag{A.3}$$

と書けることに注意すれば,

$$\begin{aligned}
\frac{\mathrm{d}z}{\mathrm{d}x} &= \exp(-x^2)\left(\frac{\mathrm{d}u_n(x)}{\mathrm{d}x} - 2xu_n(x)\right), \\
\frac{\mathrm{d}^2 z}{\mathrm{d}x^2} &= c\exp(-x^2)\left(\frac{\mathrm{d}^2 u_n(x)}{\mathrm{d}x^2} - 4x\frac{\mathrm{d}u_n(x)}{\mathrm{d}x} + 4x^2 u_n(x) - 2u_n(x)\right)
\end{aligned}$$

より式 (A.2) の左辺は

$$\begin{aligned}
\frac{\mathrm{d}^2 z}{\mathrm{d}x^2} + 2x\frac{\mathrm{d}z}{\mathrm{d}x} + 2(n+1)z = {} & \exp(-x^2)\left(\frac{\mathrm{d}^2 u_n(x)}{\mathrm{d}x^2} - 4x\frac{\mathrm{d}u_n(x)}{\mathrm{d}x} + 4x^2 u_n(x) - 2u_n(x)\right) \\
& + \exp(-x^2)\left(2x\frac{\mathrm{d}u_n(x)}{\mathrm{d}x} - 4x^2 u_n(x)\right) \\
& + \exp(-x^2)\left(2nu_n(x) + 2u_n(x)\right) \\
= {} & \exp(-x^2)\left(\frac{\mathrm{d}^2 u_n(x)}{\mathrm{d}x^2} - 2x\frac{\mathrm{d}u_n(x)}{\mathrm{d}x} + 2nu_n(x)\right)
\end{aligned}$$

となる．したがって式 (A.2) の両辺を $\exp(-x^2) \neq 0$ で割れば Hermite の微分方程式

$$\frac{\mathrm{d}^2 u_n(x)}{\mathrm{d}x^2} - 2x\frac{\mathrm{d}u_n(x)}{\mathrm{d}x} + 2nu_n(x) = 0 \tag{A.4}$$

を得ることができ，その解は式 (A.2)，(A.3) から求まるように

$$u_n(x) = z\exp(x^2) = c\exp(x^2)\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2) \tag{A.5}$$

なる多項式である．ここで $c = (-1)^n$ とした特殊解が n 次の Hermite 多項式

$$H_n(x) = (-1)^n\exp(x^2)\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2) \tag{A.6}$$

であり，定義 1 と一致することが確認できる．

A.1.2 Hermite 多項式の漸化式

式 (A.6) の両辺を x について 1 階微分，2 階微分するとそれぞれ

$$\begin{aligned}
\frac{\mathrm{d}H_n(x)}{\mathrm{d}x} &= (-1)^n\left(2x\exp(x^2)\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2) + \exp(x^2)\frac{\mathrm{d}^{n+1}}{\mathrm{d}x^{n+1}}\exp(-x^2)\right) \\
&= 2xH_n(x) - H_{n+1}(x), \\
\frac{\mathrm{d}^2 H_n(x)}{\mathrm{d}x^2} &= 2H_n(x) + 2x\frac{\mathrm{d}H_n(x)}{\mathrm{d}x} - \frac{\mathrm{d}H_{n+1}(x)}{\mathrm{d}x} \\
&= 2H_n(x) + 4x^2H_n(x) - 2xH_{n+1}(x) - 2xH_{n+1}(x) + H_{n+2}(x) \\
&= (4x^2+2)H_n(x) - 4xH_{n+1}(x) + H_{n+2}(x)
\end{aligned}$$

を得る．Hermite 多項式は Hermite の微分方程式 (A.4) の解であるから

$$\begin{aligned}
2nH_n(x) &= -\left((4x^2+2)H_n(x) - 4xH_{n+1}(x) + H_{n+2}(x)\right) + 2x\left(2xH_n(x) - H_{n+1}(x)\right) \\
&= -2H_n(x) + 2xH_{n+1} - H_{n+2}(x)
\end{aligned} \tag{A.7}$$

が任意の $n \in \mathbb{N}$ について成り立つ．ここで $m = n+1$ とすれば式 (A.7) を解いて

$$\begin{aligned}
H_{n+2}(x) &= 2xH_{n+1}(x) - 2(n+1)H_n(x) \\
H_{m+1}(x) &= 2xH_m(x) - 2mH_{m-1}(x)
\end{aligned} \tag{A.8}$$

なる漸化式を任意の $m \in \mathbb{N}_+$ について得られる．これが式 (3.10) の証明となる．

A.1.3 Hermite 多項式の偶奇性

式 (A.6) に $n = 0$，$n = 1$ を代入することにより

$$H_0(x) = (-1)^0\exp(x^2)\exp(-x^2) = 1, \qquad H_1(x) = (-1)^1\exp(x^2)\cdot(-2x)\exp(-x^2) = 2x \tag{A.9}$$

は容易に求まる．したがって，$H_0(x)$ が偶数次の項のみからなることと $H_1(x)$ が奇数次の項のみからなることが言える．

以下に自明な演繹的推論を 2 つ挙げる．これを交互に繰り返すことにより系 1 が示される．

演繹的推論 1. いま，$k \in \mathbb{N}_+$ が奇数であるとし，また，$H_k(x)$ が奇数次の項のみからなること，$H_{k-1}(x)$ が偶数次の項のみからなることが既知であるとする．このとき漸化式 3.10 から $H_{k+1}(x)$ は偶数次の項のみからなることがわかる．

演繹的推論 2. いま，$n_e \in \mathbb{N}_+$ が偶数であるとし，また，$H_k(x)$ が偶数次の項のみからなること，$H_{k-1}(x)$ が奇数次の項のみからなることが既知であるとする．このとき漸化式 3.10 から $H_{k+1}(x)$ は奇数次の項のみからなることがわかる．

$k = 1$ について，$H_0(x)$ は偶数次の項のみからなり，$H_1(x)$ は奇数次の項のみからなる．したがって，演繹的推論 1 から $H_2(x)$ が偶数次の項のみからなることが言える．次に $k = 2$ の場合について考えると直前の推論結果から $H_2(x)$ が偶数次の項のみからなることがいえており，$H_1(x)$ が奇数次の項のみからなることもすでに示されている．したがって，演繹的推論 2 から $H_3(x)$ が奇数次の項のみからなることが言える．$k = 3$ 以降の推論では直前 2 つの推論結果より仮定の条件を満たすことができるため，数学的帰納法により任意の $n \in \mathbb{N}$ について Hermite 多項式の次数が奇数ならば奇数次の項のみ，偶数ならば偶数次の項のみからなることが言える．任意の $d \in \mathbb{N}$ について x^d は d が奇数ならば奇関数，偶数ならば偶関数であり，それぞれの分類は線形結合について閉じているから系 1 の主張が示される．

A.1.4 Hermite 多項式の重み付き直交性

任意の d，$n \in \mathbb{N}$ と x の d 次多項式 $a(x)$ について

$$\lim_{x\to\infty} a(x)\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2) = 0, \qquad \lim_{x\to-\infty} a(x)\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2) = 0 \tag{A.10}$$

が成り立つ．また，

$$\frac{\mathrm{d}H_n(x)}{\mathrm{d}x} = 2xH_n(x) - (2xH_n(x) - 2nH_{n-1}(x)) = 2nH_{n-1}(x) \tag{A.11}$$

である．

m，$n \in \mathbb{N}$ とする．ただし，仮定しても一般性を損なわない関係式 $m \le n$ を満たすものとする．

$$\begin{aligned}
&\int_{\mathbb{R}} H_m(x)H_n(x)\exp(-x^2)\,\mathrm{d}x \\
=&(-1)^n \int_{\mathbb{R}} H_m(x)\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2)\,\mathrm{d}x \qquad (A.12)\\
=&(-1)^n \left(\left[H_m(x)\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}\exp(-x^2)\right]_{-\infty}^{\infty} - \int_{\mathbb{R}} \frac{\mathrm{d}H_m(x)}{\mathrm{d}x}\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}\exp(-x^2)\,\mathrm{d}x\right) \\
=&(-1)^{n+1} \int_{\mathbb{R}} \frac{\mathrm{d}H_m(x)}{\mathrm{d}x}\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}\exp(-x^2)\,\mathrm{d}x \\
=&(-1)^{n+1}\,2m \int_{\mathbb{R}} H_{m-1}(x)\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}\exp(-x^2)\,\mathrm{d}x \qquad (A.13)
\end{aligned}$$

ここで，(A.12)，(A.13) に注目すると，積分記号の前が $-2m$ 倍されており，かつ，被積分関数の m と n がそれぞれ 1 小さくなった式変形と見なすことができる．$m = 0$ となるまでこの式変形を繰り返し行えば，

$$\begin{aligned}
\int_{\mathbb{R}} H_m(x)H_n(x)\exp(-x^2)\,\mathrm{d}x =&(-1)^n \int_{\mathbb{R}} H_m(x)\frac{\mathrm{d}^n}{\mathrm{d}x^n}\exp(-x^2)\,\mathrm{d}x \\
=&(-1)^{n+1}\,2m\int_{\mathbb{R}} H_{m-1}(x)\frac{\mathrm{d}^{n-1}}{\mathrm{d}x^{n-1}}\exp(-x^2)\,\mathrm{d}x \\
&\vdots \\
=&(-1)^{n+m}\,2^m m!\int_{\mathbb{R}} H_0(x)\frac{\mathrm{d}^{n-m}}{\mathrm{d}x^{n-m}}\exp(-x^2)\,\mathrm{d}x \\
=&(-1)^{n+m}\,2^m m!\int_{\mathbb{R}} \frac{\mathrm{d}^{n-m}}{\mathrm{d}x^{n-m}}\exp(-x^2)\,\mathrm{d}x \qquad \text{(A.14)} \\
=&(-1)^{n+m}\,2^m m!\left[\frac{\mathrm{d}^{n-m-1}}{\mathrm{d}x^{n-m-1}}\exp(-x^2)\right]_{-\infty}^{\infty} \qquad \text{(A.15)}
\end{aligned}$$

を得る．ここで $n \geq m+1$ ならば $n-m-1 \geq 0$ であるから (A.15) と式 (A.10) より 0 となる．一方 $n = m$ ならば (A.14) の積分は Gauss 積分（$\sqrt{\pi}$）であるから，$2^m m!\sqrt{\pi}$ となる．定理 2 が示された．

A.2 OtX 成分系の直交性

3.4.2 節にて導入した OtX 成分系は，学習済み畳み込みフィルタの主成分分析の結果を再現することを目標に近似モデル化した．主成分分析の結果得られる主成分同士は直交する性質を持っているため，OtX 成分系は可能な限り成分同士が直交するように設計した．結果として，Φ_{2n} 同士を除き，成分同士が直交する成分系となった．以下では，成分同士の直交性についての証明を行う．

上記の主張を表 A.1 に示した．同じ種類の成分同士の内積については $\perp$ と書いてあり，これは $m = n$ のときのみ非 0 の値を持つことを意味する．Φ_{2m} と Φ_{2n} については直交しないことを表すため $\times$ とした．そして，異なる種類の成分同士の内積はすべて 0 となる．つまり，OtX 成分系において直交性を持たないのは球対称成分同士のみである．

表 A.1: OtX 成分同士の内積の関係表

	Φ_{2n}	$\Psi^{(0)}_{n,n+1}$	$\Psi^{(\frac{\pi}{2})}_{n,n+1}$	$\Psi^{(0)}_{2n+1,2n+1}$	$\Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}$
Φ_{2m}	$\times$	0	0	0	0
$\Psi^{(0)}_{m,m+1}$	0	$\perp$	0	0	0
$\Psi^{(\frac{\pi}{2})}_{m,m+1}$	0	0	$\perp$	0	0
$\Psi^{(0)}_{2m+1,2m+1}$	0	0	0	$\perp$	0
$\Psi^{(\frac{\pi}{4})}_{2m+1,2m+1}$	0	0	0	0	$\perp$

表 A.1 における結果は表 A.2 のなかの対応する位置に書かれた命題を根拠に証明される．なお，実数の積に関する対称性から m，n を入れ替えた位置を参照することで示される部分については

「-」と表記した.

表 A.2: OtX 成分系における直交性の証明表

	Φ_{2n}	$\Psi^{(0)}_{n,n+1}$	$\Psi^{(\frac{\pi}{2})}_{n,n+1}$	$\Psi^{(0)}_{2n+1,2n+1}$	$\Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}$
Φ_{2m}	直交性なし	系 4	系 4	系 4	系 4
$\Psi^{(0)}_{m,m+1}$	-	系 5	(補題 1, 系 5)	系 5	系 6
$\Psi^{(\frac{\pi}{2})}_{m,m+1}$	-	-	系 5	(補題 1, 系 5)	(補題 1, 系 6)
$\Psi^{(0)}_{2m+1,2m+1}$	-	-	-	系 5	系 6
$\Psi^{(\frac{\pi}{4})}_{2m+1,2m+1}$	-	-	-	-	系 5

以下の命題 1 は偽である．もしもこれが真ならば OtX 成分系は完全な直交性を持っていたこととなる．

命題 1（Φ_{2n} 同士の直交性（成立しない）） Φ_{2n} は直交する．つまり，

$$\int_{\mathbb{R}^2} \Phi_{2m}(x,y)\Phi_{2n}(x,y)\ \mathrm{d}(x,y) \tag{A.16}$$

は $m=n$ ならば非 0，$m\neq n$ ならば 0 である．

実際に内積を計算すると

$$\begin{aligned}\int_{\mathbb{R}^2} \Phi_{2m}(x,y)\Phi_{2n}(x,y)\ \mathrm{d}(x,y) &= \int_0^{2\pi}\left(\int_0^{\infty} \psi_{2m}(r)\psi_{2n}(r)\cdot r\ \mathrm{d}r\right)\mathrm{d}\theta \\ &= \left(\int_0^{2\pi}\mathrm{d}\theta\right)\left(\int_0^{\infty}\psi_{2m}(r)\psi_{2n}(r)\cdot r\ \mathrm{d}r\right) \\ &= 2\pi\int_0^{\infty}\psi_{2m}(r)\psi_{2n}(r)\cdot r\ \mathrm{d}r\end{aligned}$$

となる．ここで $r\ \mathrm{d}r$ の形ではなく $\mathrm{d}r$ であれば直交性が成立した．なお，$\dfrac{1}{\sqrt{r}}\psi_{2n}(r)$ の形で定義することでこの r を打ち消せるように思われるが，$r\to 0$ で無限大に発散する（フィルタの中心の値の絶対値が無限大に発散する）ため望ましくない．

ここからが，表 A.2 にあった命題である．ただし，以下に登場する積分はすべて絶対可積分であることを証明なしに仮定している．

系 4（Φ_{2n} と $\Psi^{(\varphi)}_{i,j}$ の内積） Φ_{2n} と $\Psi^{(\varphi)}_{i,j}$ の内積は 0 である．

$$\int_{\mathbb{R}^2} \Phi_{2m}(x,y)\Psi^{(\varphi)}_{i,j}(x,y)\ \mathrm{d}(x,y) = 0 \tag{A.17}$$

証明　$\Psi_{i,j}^{(\varphi)}$ の定義より，i，j の少なくとも一方は奇数である．実際，奇数 i，$j=i$ について定義された $\dfrac{\pi}{4}$ 回転対称成分はこれを満たすし，i が偶数であっても $j=i+1$ は奇数である．したがって，$\psi_i(x_\varphi)$，$\psi_j(y_\varphi)$ のいずれか一方は少なくとも奇関数であると言える．

いま，i が奇数，つまり $\psi_i(x_\varphi)$ が奇関数であるとすると，

$$\begin{aligned}\int_{\mathbb{R}^2}\Phi_{2n}(x,y)\Psi_{i,j}^{(\varphi)}(x,y)\,\mathrm{d}(x,y)&=\int_{\mathbb{R}^2}\Phi_{2n}(x,y)\psi_i(x_\varphi)\psi_j(y_\varphi)\,\mathrm{d}(x,y)\\&=\int_{\mathbb{R}}\left(\int_{\mathbb{R}}\Phi_{2n}(x_\varphi,y_\varphi)\psi_i(x_\varphi)\,\mathrm{d}x_\varphi\right)\psi_j(y_\varphi)\,\mathrm{d}y_\varphi\\&=\int_{\mathbb{R}}0\cdot\psi_j(y_\varphi)\,\mathrm{d}y_\varphi\\&=0\end{aligned}$$

ここで，

$$\Phi_{2n}(-x_\varphi,y_\varphi)=\psi_{2n}\left(\sqrt{(-x_\varphi)^2+y_\varphi^2}\right)=\psi_{2n}\left(\sqrt{x_\varphi^2+y_\varphi^2}\right)=\Phi_{2n}(x_\varphi,y_\varphi)$$

より $\Phi_{2n}(x_\varphi,y_\varphi)$ が x_φ について偶関数であることを利用した．また，j が奇数の場合も同様であるから主張が成り立つ．□

系 5（$\Psi_{i,j}^{(\varphi)}$ の内積（φ が固定の場合））　m，$n\in\mathbb{N}$ について $\Psi_{m,n}^{(\varphi)}$ は直交し，

$$\int_{\mathbb{R}^2}\Psi_{i,j}^{(\varphi)}(x,y)\Psi_{m,n}^{(\varphi)}(x,y)\,\mathrm{d}(x,y)=2^{m+n}m!n!\pi\delta_{i,m}\delta_{j,n}\tag{A.18}$$

が成立する．

証明

$$\begin{aligned}\int_{\mathbb{R}^2}\Psi_{i,j}^{(\varphi)}(x,y)\Psi_{m,n}^{(\varphi)}(x,y)\,\mathrm{d}(x,y)&=\int_{\mathbb{R}^2}\psi_i(x_\varphi)\psi_j(y_\varphi)\psi_m(x_\varphi)\psi_n(y_\varphi)\,\mathrm{d}(x,y)\\&=\left(\int_{\mathbb{R}}\psi_i(x_\varphi)\psi_m(x_\varphi)\,\mathrm{d}x_\varphi\right)\left(\int_{\mathbb{R}}\psi_j(y_\varphi)\psi_n(y_\varphi)\,\mathrm{d}y_\varphi\right)\\&=2^m m!\sqrt{\pi}\delta_{i,m}\cdot 2^n n!\sqrt{\pi}\delta_{j,n}\\&=2^{m+n}m!n!\pi\delta_{i,m}\delta_{j,n}\ \square\end{aligned}$$

以下の補題 1 は $\Psi_{i,j}^{(\frac{\pi}{2})}$ に他の系を適用する前に $\Psi_{j,i}^{(0)}$ の形式に変換するためのものである．

補題 1（$\Psi_{i,j}^{(\frac{\pi}{2})}$）　$\Psi_{i,j}^{(\frac{\pi}{2})}$ は $\Psi_{j,i}^{(0)}$ に比例する．

証明

$$\begin{bmatrix}x_{\frac{\pi}{2}}\\y_{\frac{\pi}{2}}\end{bmatrix}=\begin{bmatrix}\cos\dfrac{\pi}{2}&-\sin\dfrac{\pi}{2}\\\sin\dfrac{\pi}{2}&\cos\dfrac{\pi}{2}\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0&-1\\1&0\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}-y\\x\end{bmatrix}\tag{A.19}$$

したがって，

$$\Psi^{(\frac{\pi}{2})}_{m,n}(x,y) = \psi_m(-y)\psi_n(x) = (-1)^m\psi_n(x)\psi_m(y) = (-1)^m\Psi^{(0)}_{n,m}(x,y)\ \square \tag{A.20}$$

系 6（$\Psi^{(\varphi)}_{i,j}$ の内積（φ が固定の場合））　Φ_{2n} と $\Psi^{(\varphi)}_{i,j}$ の内積は 0 である．

$$\int_{\mathbb{R}^2} \Phi_{2m}(x,y)\Psi_{2n}(x,y)\,\mathrm{d}(x,y) = 0 \tag{A.21}$$

証明

$$\begin{bmatrix} x_{\frac{\pi}{4}} \\ y_{\frac{\pi}{4}} \end{bmatrix} = \begin{bmatrix} \cos\dfrac{\pi}{4} & -\sin\dfrac{\pi}{4} \\ \sin\dfrac{\pi}{4} & \cos\dfrac{\pi}{4} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} x-y \\ x+y \end{bmatrix} \tag{A.22}$$

したがって，

$$\begin{aligned} \Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(-x,y) &= \psi_{2n+1}\left(\frac{-x-y}{\sqrt{2}}\right)\psi_{2n+1}\left(\frac{-x+y}{\sqrt{2}}\right) \\ &= \psi_{2n+1}\left(\frac{x-y}{\sqrt{2}}\right)\psi_{2n+1}\left(\frac{x+y}{\sqrt{2}}\right) \\ &= \Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(x,y) \end{aligned}$$

と

$$\begin{aligned} \Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(x,-y) &= \psi_{2n+1}\left(\frac{x-(-y)}{\sqrt{2}}\right)\psi_{2n+1}\left(\frac{x+(-y)}{\sqrt{2}}\right) \\ &= \psi_{2n+1}\left(\frac{x-y}{\sqrt{2}}\right)\psi_{2n+1}\left(\frac{x+y}{\sqrt{2}}\right) \\ &= \Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(x,y) \end{aligned}$$

から，$\Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(x,y)$ が x，y 方向それぞれについて偶関数であることが言える．

いま，$\Psi^{(0)}_{i,j}$ において i，j の少なくとも一方は奇数であり，i が奇数であるとすれば，

$$\begin{aligned} \int_{\mathbb{R}^2} \Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(x,y)\Psi^{(0)}_{i,j}(x,y)\,\mathrm{d}(x,y) &= \int_{\mathbb{R}^2} \Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(x,y)\psi_i(x_\varphi)\psi_j(y_\varphi)\,\mathrm{d}(x,y) \\ &= \int_{\mathbb{R}} \left(\int_{\mathbb{R}} \Psi^{(\frac{\pi}{4})}_{2n+1,2n+1}(x,y)\psi_i(x_\varphi)\,\mathrm{d}x_\varphi\right)\psi_j(y_\varphi)\,\mathrm{d}y_\varphi \\ &= \int_{\mathbb{R}} 0\cdot\psi_j(y_\varphi)\,\mathrm{d}y_\varphi \\ &= 0 \end{aligned}$$

となり，j が奇数の場合も同様だから主張が成り立つ．$\square$