

VIP チートシート: 畳み込みニューラルネットワーク

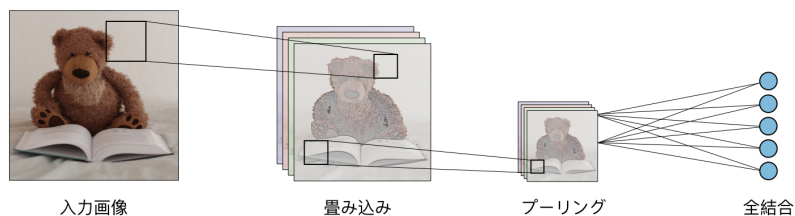
アフシンアミディ・シエルビンアミディ 著

October 7, 2019

チャントウアンアイン・中井喜之訳

概要

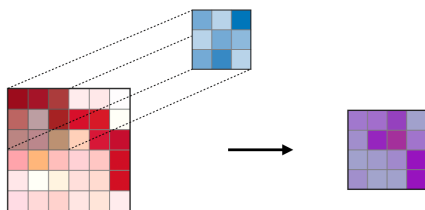
□ **伝統的な畳み込みニューラルネットワークのアーキテクチャ** – CNNとしても知られる畳み込みニューラルネットワークは一般的に次の層で構成される特定種類のニューラルネットワークです。



畳み込み層とプーリング層は次のセクションで説明されるハイパーパラメータに関してファインチューニングできます。

層の種類

□ **畳み込み層(CONV)** – 畳み込み層(CONV)は入力 I を各次元に関して走査する時に、畳み込み演算を行うフィルタを使用します。畳み込み層のハイパーパラメータにはフィルタサイズ F とストライド S が含まれます。結果出力 O は特徴マップまたは活性化マップと呼ばれます。

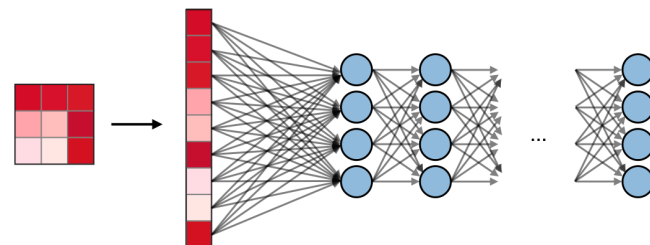


注: 畳み込みステップは1次元や3次元の場合にも一般化できます。

□ **プーリング(Pool)** – プーリング層(Pool)は位置不変性をもつ縮小操作で、通常は畳み込み層の後に適用されます。特に、最大及び平均プーリングはそれぞれ最大と平均値が取られる特別な種類のプーリングです。

| | 最大プーリング | 平均プーリング |
|------|---|--|
| 種類 | 各プーリング操作は現在のビューの中から最大値を選ぶ | 各プーリング操作は現在のビューに含まれる値を平均する |
| 図 | | |
| コメント | <ul style="list-style-type: none"> - 検出された特徴を保持する - 最も一般的に利用される | <ul style="list-style-type: none"> - 特徴マップをダウンサンプリングする - LeNetで利用される |

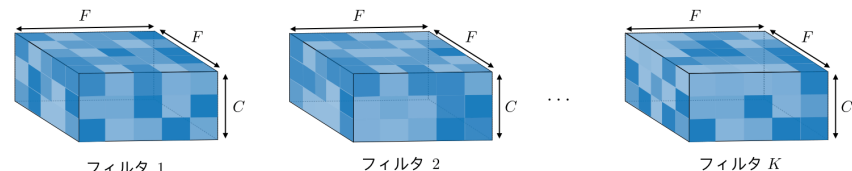
□ **全結合(FC)** – 全結合(FC)層は平坦化された入力に対して演算を行います。各入力はあるニューロンに接続されています。FC層が存在する場合、通常CNNアーキテクチャの末尾に向かって見られ、クラススコアなどの目的を最適化するため利用できます。



フィルタハイパーパラメータ

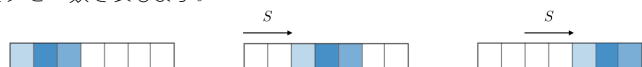
畳み込み層にはハイパーパラメータの背後にある意味を知ることが重要なフィルタが含まれています。

□ **フィルタの次元** – C 個のチャネルを含む入力に適用される $F \times F$ サイズのフィルタの体積は $F \times F \times C$ で、それは $I \times I \times C$ サイズの入力に対して畳み込みを実行して $O \times O \times 1$ サイズの特徴マップ(活性化マップとも呼ばれる)の出力を生成します。



注: $F \times F$ サイズの K 個のフィルタを適用すると、 $O \times O \times K$ サイズの特徴マップの出力を得られます。

□ **ストライド** – 畳み込みまたはプーリング操作において、ストライド S は各操作の後にウィンドウを移動させるピクセル数を表します。



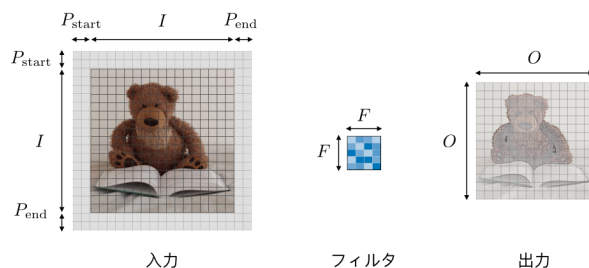
□ **ゼロパディング** – ゼロパディングとは入力各境界に対して P 個のゼロを追加するプロセスを意味します。この値は手動で指定することも、以下に詳述する3つのモードのいずれかを使用して自動的に設定することもできます。

| | Valid | Same | Full |
|----|--|--|---|
| 値 | $P = 0$ | $P_{\text{start}} = \left\lfloor \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rfloor$ $P_{\text{end}} = \left\lceil \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rceil$ | $P_{\text{start}} \in [0, F - 1]$ $P_{\text{end}} = F - 1$ |
| 図 | | | |
| 目的 | - パディングなし - もし次元が合わなかったら最後の畳み込みをやめる | - 特徴マップのサイズが $\lceil \frac{I}{S} \rceil$ になるようなパディング - 出力サイズは数学的に扱いやすい - 「ハーフ」パディングとも呼ばれる | - 入力の一番端まで畳み込みが適用されるような最大パディング - フィルタは入力を端から端まで「見る」 |

ハイパーパラメータの調整

□ **畳み込み層内のパラメータ互換性** – I を入力ボリュームサイズの長さ、 F をフィルタの長さ、 P をゼロパディングの量、 S をストライドとすると、その次元に沿った特徴マップの出力サイズ O は次式で与えられます：

$$O = \frac{I - F + P_{\text{start}} + P_{\text{end}}}{S} + 1$$



注: 多くの場合 $P_{\text{start}} = P_{\text{end}} \triangleq P$ であり、上記の式の $P_{\text{start}} + P_{\text{end}}$ を $2P$ に置き換える事ができます。

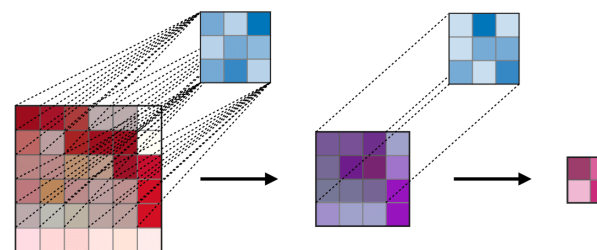
□ **モデルの複雑さを理解する** – モデルの複雑さを評価するために、モデルのアーキテクチャが持つパラメータの数を測定することがしばしば有用です。畳み込みニューラルネットワークの各層では、以下に行なわれます。

| | CONV | POOL | FC |
|---------|--|--|--|
| 図 | | | |
| 入力サイズ | $I \times I \times C$ | $I \times I \times C$ | N_{in} |
| 出力サイズ | $O \times O \times K$ | $O \times O \times C$ | N_{out} |
| パラメータの数 | $(F \times F \times C + 1) \cdot K$ | 0 | $(N_{\text{in}} + 1) \times N_{\text{out}}$ |
| 備考 | - フィルタごとに1つのバイアスパラメータ - ほとんどの場合、 $S < F$ - K の一般的な選択は $2C$ | - プール操作はチャンネルごとに行われる - ほとんどの場合、 $S = F$ | - 入力は平坦化される - ニューロンごとにひとつのバイアスパラメータ - FCのニューロンの数には構造的制約がない |

□ **受容野** – 層 k における受容野は、 k 番目の活性化マップの各ピクセルが「見る」ことができる入力の $R_k \times R_k$ の領域です。層 j のフィルタサイズを F_j 、層 i のストライド値を S_i とし、慣例に従って $S_0 = 1$ とすると、層 k での受容野は次の式で計算されます：

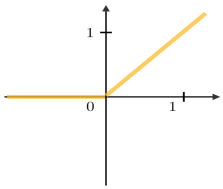
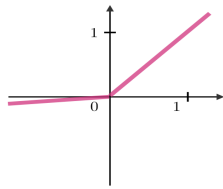
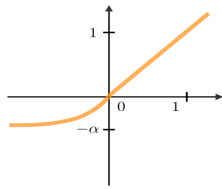
$$R_k = 1 + \sum_{j=1}^k (F_j - 1) \prod_{i=0}^{j-1} S_i$$

下記の例のように $F_1 = F_2 = 3$ 、 $S_1 = S_2 = 1$ とすると、 $R_2 = 1 + 2 \cdot 1 + 2 \cdot 1 = 5$ となります。



よく使われる活性化関数

□ **正規化線形ユニット** – 正規化線形ユニット層(ReLU)はボリュームの全ての要素に利用される活性化関数 g です。ReLUの目的は非線型性をネットワークに導入することです。変種は以下の表でまとめられています：




| ReLU | Leaky ReLU | ELU |
|---|---|---|
| $g(z) = \max(0, z)$ | $g(z) = \max(\epsilon z, z)$ ただし $\epsilon \ll 1$ | $g(z) = \max(\alpha(e^z - 1), z)$ ただし $\alpha \ll 1$ |
|  |  |  |
| 生物学的に解釈可能な非線形複雑性 | 負の値に対してReLUが死んでいる問題に対処する | どこでも微分可能 |

□ **ソフトマックス** – ソフトマックスのステップは入力としてスコア $x \in \mathbb{R}^n$ のベクトルを取り、アーキテクチャの最後にあるソフトマックス関数を通じて確率 $p \in \mathbb{R}^n$ のベクトルを出力する一般化されたロジスティック関数として見ることができます。次のように定義されます。

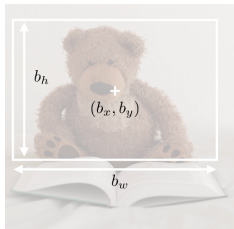
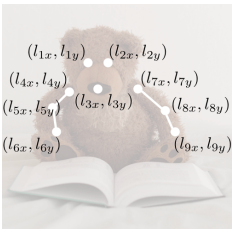
$$p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \quad \text{ここで} \quad p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

物体検出

□ **モデルの種類** – 物体認識アルゴリズムは主に3つの種類があり、予測されるものの性質は異なります。次の表で説明されています。

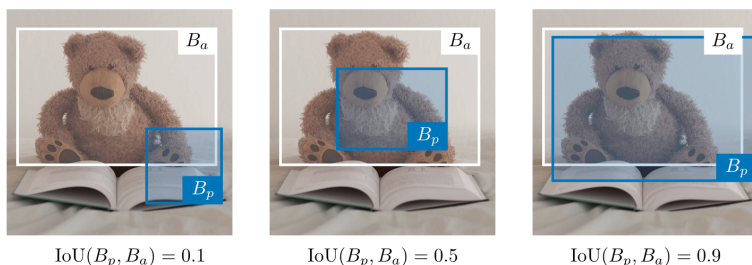
| 画像分類 | 位置特定を伴う分類 | 検出 |
|---|---|---|
|  |  |  |
| - 画像を分類する - 物体の確率を予測する | - 画像内の物体を検出する - 物体の確率とその位置を予測する | - 画像内の複数の物体を検出する - 複数の物体の確率と位置を予測する |
| 伝統的なCNN | 単純されたYOLO, R-CNN | YOLO, R-CNN |

□ **検出** – 物体検出の文脈では、画像内の物体の位置を特定したいだけなのかあるいは複雑な形状を検出したいのかによって、異なる方法が使用されます。二つの主なものは次の表でまとめられています：

| バウンディングボックス検出 | ランドマーク検出 |
|--|--|
| 物体が配置されている画像の部分を検出する | - 物体（たとえば目）の形状または特徴を検出する - よりきめ細かい |
|  |  |
| 中心 (b_x, b_y) , 高さ b_h 、幅 b_w のボックス | 参照点 $(l_{1x}, l_{1y}), \dots, (l_{nx}, l_{ny})$ |

□ **Intersection over Union** – Intersection over Union (IoUとしても知られる)は予測された境界ボックス B_p が実際の境界ボックス B_a に対してどれだけ正しく配置されているかを定量化する関数です。次のように定義されます：

$$\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

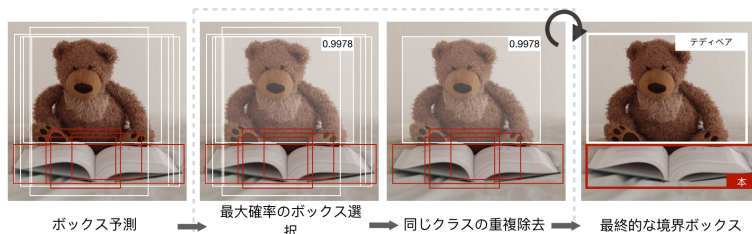


注：常に $\text{IoU} \in [0,1]$ となります。慣例では、 $\text{IoU}(B_p, B_a) \geq 0.5$ の場合、予測された境界ボックス B_p はそこそこ良いと見なされます。

□ **アンカーボックス** – アンカーボクシングは重なり合う境界ボックスを予測するために使用される手法です。実際には、ネットワークは同時に複数のボックスを予測することを許可されており、各ボックスの予測は特定の幾何学的属性の組み合わせを持つように制約されます。例えば、最初の予測は特定の形式の長方形のボックスになる可能性があり、2番目の予測は異なる幾何学的形式の別の長方形のボックスになります。

□ **非極大抑制** – 非極大抑制技術のねらいは、最も代表的なものを選択することによって、同じ物体の重複した重なり合う境界ボックスを除去することです。0.6未満の予測確率を持つボックスを全て除去した後、残りのボックスがある間、以下の手順が繰り返されます：

- ステップ1: 最大の予測確率を持つボックスを選ぶ。
- ステップ2: そのボックスに対して $\text{IoU} \geq 0.5$ となる全てのボックスを破棄する。



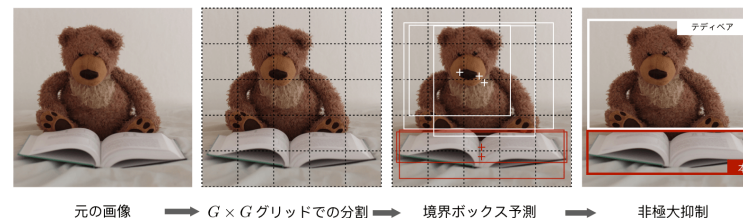
□ **YOLO** – You Only Look Once (YOLO)は次の手順を実行する物体検出アルゴリズムです。

- ステップ1: 入力画像を $G \times G$ グリッドに分割する。
- ステップ2: 各グリッドセルに対して次の形式の y を予測するCNNを実行する：

$$y = \underbrace{[p_c, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_p, \dots]}_{k \text{ 回繰り返す}}^T \in \mathbb{R}^{G \times G \times k \times (5+p)}$$

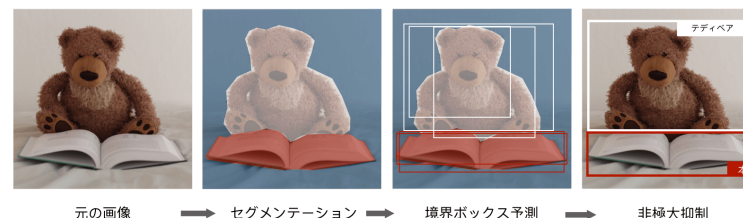
ここで、 p_c は物体を検出する確率、 b_x, b_y, b_h, b_w は検出された境界ボックスの属性、 c_1, \dots, c_p は p 個のクラスのうちどれが検出されたかのOne-hot表現、 k はアンカーボックスの数です。

- ステップ3: 重複する可能性のある重なり合う境界ボックスを全て除去するため、非極大抑制アルゴリズムを実行する。



注： $p_c = 0$ のとき、ネットワークは物体を検出しません。その場合には、対応する予測 b_x, \dots, c_p は無視する必要があります。







□ **R-CNN** – Region with Convolutional Neural Networks (R-CNN)は物体検出アルゴリズムで、最初に画像をセグメント化して潜在的に関連する境界ボックスを見つけ、次に検出アルゴリズムを実行してそれらの境界ボックス内で最も可能性の高い物体を見つけます。



注：元のアルゴリズムは計算コストが高く遅いですが、*Fast R-CNN* や *Faster R-CNN* などの、より新しいアーキテクチャではアルゴリズムをより速く実行できます。

顔認証及び認識

□ **モデルの種類** – 2種類の主要なモデルが次の表にまとめられています：

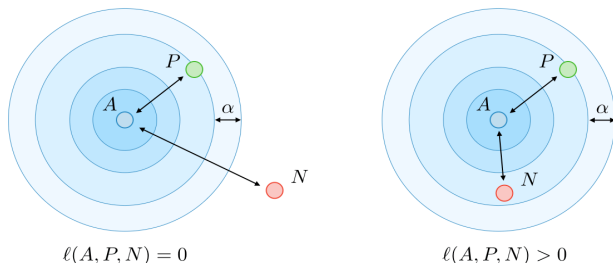
| 顔認証 | 顔認識 |
|--|--|
| - これは正しい人ですか？ - 1対1検索 | - これはデータベース内のK人のうちの1人ですか - 1対多検索 |
| クエリ   参照  | クエリ   データベース  |

□ **ワンショット学習** – ワンショット学習は限られた学習セットを利用して、2つの与えられた画像の違いを定量化する類似度関数を学習する顔認証アルゴリズムです。2つの画像に適用される類似度関数はしばしば $d(\text{画像1}, \text{画像2})$ と記されます。

□ **シムネットワーク** – シムネットワークは画像のエンコード方法を学習して2つの画像の違いを定量化することを目的としています。与えられた入力画像 $x^{(i)}$ に対してエンコードされた出力はしばしば $f(x^{(i)})$ と記されます。

□ **トリプレット損失** – トリプレット損失 ℓ は3組の画像 A (アンカー)、 P (ポジティブ)、 N (ネガティブ)の埋め込み表現で計算される損失関数です。アンカーとポジティブ例は同じクラスに属し、ネガティブ例は別のクラスに属します。マージンパラメータを $\alpha \in \mathbb{R}^+$ と呼ぶことによってこの損失は次のように定義されます:

$$\ell(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$



ニューラルスタイル変換

□ **モチベーション** – ニューラルスタイル変換の目的は与えられたコンテンツ C とスタイル S に基づく画像 G を生成することです。



□ **活性化** – 層 l における活性化は $a^{[l]}$ と表記され、次元は $n_H \times n_w \times n_c$ です。

□ **コンテンツコスト関数** – $J_{\text{content}}(C, G)$ というコンテンツコスト関数は生成された画像 G と元のコンテンツ画像 C との違いを測定するため利用されます。以下のように定義されます:

$$J_{\text{content}}(C, G) = \frac{1}{2} \|a^{[l](C)} - a^{[l](G)}\|^2$$

□ **スタイル行列** – 与えられた層 l のスタイル行列 $G^{[l]}$ はグラム行列で、各要素 $G_{kk'}^{[l]}$ がチャネル k と k' の相関関係を定量化します。活性化 $a^{[l]}$ に関して次のように定義されます。

$$G_{kk'}^{[l]} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_w} a_{ij,k}^{[l]} a_{ij,k'}^{[l]}$$

注: スタイル画像及び生成された画像に対するスタイル行列はそれぞれ $G^{[l](S)}$ 、 $G^{[l](G)}$ と表記されます。

□ **スタイルコスト関数** – スタイルコスト関数 $J_{\text{style}}(S, G)$ は生成された画像 G とスタイル S との違いを測定するため利用されます。以下のように定義されます:

$$J_{\text{style}}^{[l]}(S, G) = \frac{1}{(2n_H n_w n_c)^2} \|G^{[l](S)} - G^{[l](G)}\|_F^2 = \frac{1}{(2n_H n_w n_c)^2} \sum_{k,k'=1}^{n_c} \left(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)} \right)^2$$

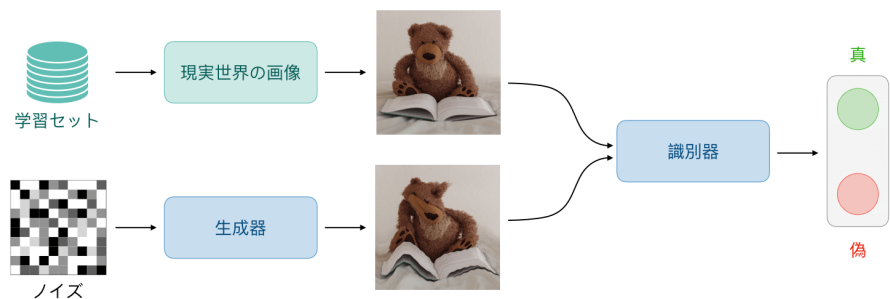
□ **全体のコスト関数** – 全体のコスト関数は以下のようにパラメータ α, β によって重み付けされたコンテンツ及びスタイルコスト関数の組み合わせとして定義されます:

$$J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$$

注: α の値を大きくするとモデルはコンテンツを重視し、 β の値を大きくするとスタイルを重視します。

計算トリックを使うアーキテクチャ

□ **敵対的生成ネットワーク** – 敵対的生成ネットワーク (GANsとも呼ばれる) は生成モデルと識別モデルで構成されます。生成モデルの目的は、生成された画像と本物の画像を区別することを目指すとする識別モデルに与えられる、最も本物らしい出力を生成することです。



注: GANsの変種を使用するユースケースにはテキストからの画像生成、音楽生成及び合成があります。

□ **ResNet** – Residual Networkアーキテクチャ (ResNetとも呼ばれる) は学習エラーを減らすため多数の層がある残差ブロックを使用します。残差ブロックは次の特性方程式を有します。

$$a^{[l+2]} = g(a^{[l]} + z^{[l+2]})$$

□ **インセプションネットワーク** – このアーキテクチャはインセプションモジュールを利用し、特徴量の多様化を通じてパフォーマンスを向上させるため、様々な畳み込みを試すことを目的としています。特に、計算負荷を限定するため 1×1 畳み込みトリックを使います。

★ ★ ★