1. **勾配消失問題**

**機械学習の多層化したニューラルネットワークにおいて、ある段階を越えると学習が進まなくなること。学習は予測値と実際の値の誤差を最小化する過程で進むが、活性化関数の勾配がゼロに近づくことによって、ネットワークの重み付けの修正ができなくなり、結果的に層数が増えるほど学習が困難となる。活性化関数としてシグモイド関数ではなく、ReLU関数を用いると勾配消失が起こりにくくなることが知られる（コトバンクより）。**

**勾配消失問題の解決策には下記が考えられる（Qiitaより）。**

1. **活性化関数の選択**

**ReLU関数を使う。シグモイド関数と違い、閾値を超えた値がそのまま出力されるので小さくなりづらい。**

1. **重みの初期値設定**
2. **Xavier(ザビエル)**

**重みの要素を、前の層のノード数nに対して、平均0、標準偏差がである正規分布から重みを設定する手法。**

**以下の活性化関数を用いるとき、Xavierによる初期値が有効**

**ReLU、シグモイド(ロジスティック)、双曲線正接**

1. **He(エイチイー)**

**ReLU関数に付随させ、よりもっと勾配消失を起こさないようにする重みの要素を、ノード数nに対して、平均0、標準偏差がである正規分布から重みを設定。**

1. **重みの初期値に0を設定すると、どのような問題が発生するか？**

**すべての値(次に伝播する値)が同じ値で伝わるため、パラメータチューニング行われ、学習の概念が阻害されてしまう。**

1. **バッチ正規化**

**バッチ正規化は、学習時のミニバッチ毎に各チャンネルを平均０分散１になるように正規化することで、学習の安定化や速度アップが出来る必須のテクニックである。**

**また、ミニバッチ単位へ入力値のデータの偏りを抑制する手法で、活性化関数に値を渡す前後にバッチ正規化の処理を孕んだ層を加えることで使用する。**

1. **バッチ正規化の効果**
2. **計算の高速化**
3. **勾配消失が起こりづらくなる**
4. **弱点**
5. **高解像度の画像を扱うことでメモリが不足しバッチサイズが小さくなると、平均・分散の推定が不安定になり上手く学習が進まない。**
6. **①の対策として複数のGPUで対応すると、高価なハードウエアが必要で実装や最適化が複雑になる**
7. **ビデオの隣接フレームといった相関がある画像の場合、平均・分散の推定が不安定になり上手く学習が進まない。**

* **確認テスト１**
* **連鎖律の原理を使い、を求めよ。**
* **確認テスト2**
* **シグモイド関数を微分した時、入力値が0の時に最大値をとる。その値として正しいものを選択肢から選べ。**

**(2)0.25**

* **確認テスト3**
* **重みの初期値に0を設定すると、どのような問題が発生するか。簡潔に説明せよ。**

**全ての値が同じ値で伝わるためパラメータのチューニングが行われなくなる。**

* **確認テスト4**
* **一般的に考えられるバッチ正規化の効果を2点挙げよ。**
* **計算が早くなる点。**
* **勾配消失が起こりにくくなる点。**

1. **学習率最適化手法**

**学習率の値が大きい場合は、最適値にいつまでもたどり着かず発散してしまう。また、学習率の値が小さい場合は、発散することは無いが小さすぎると収束するまでに時間がかかってしまう。もしくは大域局所最適値に収束しづらくなる。そのため、学習率最適化手法を利用して学習率を最適化する。**

1. **モメンタム**
2. **モメンタム**

**誤差をパラメータで微分したものと学習率の積を減算した後、現在の重みに前回の重みを減算した値と慣性の積を加算する。**

1. **勾配降下法**

**誤差をパラメータで微分したものと学習率の積を減算する。**

* **モメンタムのメリット**
* **局所的最適解にはならず、大域的最適解となる。**
* **谷間についてから最も低い位置（最適値）にいくまでの時間が早い。**

1. **AdaGrad**
2. **AdaGrad**

**誤差をパラメータで微分したものと再定義した学習率の積を減算する。**

1. **勾配降下法**

**誤差をパラメータで微分したものと学習率の積を減算する。**

* **AdaGradのメリット**
* **勾配の緩やかな斜面に対して、最適値に近づける。**
* **課題**
* **学習率が徐々に小さくなるので、鞍点問題を引き起こす事があった。**

1. **RMSProp**
2. **RMSProp**

**誤差をパラメータで微分したものと再定義した学習率の積を減算する。**

1. **勾配降下法**

**誤差をパラメータで微分したものと学習率の積を減算する。**

* **RMSpropのメリット**
* **局所的最適解にはならず、大域的最適解となる。**
* **ハイパーパラメータの調整が必要な場合が少ない。**

1. **Adam**

**モメンタムの、「①過去の勾配の指数関数的減衰平均」、RMSPropの、「②過去の勾配の2乗の指数関数的減衰平均」、①と②をそれぞれ孕んだ最適化アルゴリズムである。**

* **Adamのメリット**
* **モメンタムおよびRMSPropのメリットを孕んだアルゴリズムである。**
* **欠点**
* **計算量が多い。**
* **確認テスト5**
* **モメンタム・AdaGrad・RMSPropの特徴をそれぞれ簡潔に説明せよ。**
* **モメンタム：局所的最適解にはならずに大域的最適解となり谷についてから最適値までの時間が早い。**
* **AdaGrad：勾配の緩やかな斜面に対して最適値に近づく。**
* **RMSProp：誤差をパラメータで微分したものと再定義した学習率の積を減算する。**

1. **過学習**

**過学習とは、回帰分析や機械学習で、学習データに対してあまりに忠実に適合しすぎて本来データが示唆する傾向から大きく外れてしまう現象である（IT用語辞典 e-Wordsより）。テスト誤差と訓練誤差とで学習曲線が乖離すること。**

* **原因**
* **パラメータの数が多い**
* **パラメータの値が適切でない**
* **ノードが多い**

1. **Weight decay(荷重減衰)**
2. **過学習の原因**

**重みが大きい値をとることで、過学習が発生することがある。学習させていくと、重みにばらつきが発生する。重みが大きい値は、学習において重要な値であり、重みが大きいと過学習が起こる。**

1. **過学習の解決策**

**誤差に対して、正則化項を加算することで、重みを抑制する。過学習がおこりそうな重みの大きさ以下で重みをコントロールし、かつ重みの大きさにばらつきを出す必要がある。**

1. **正則化**

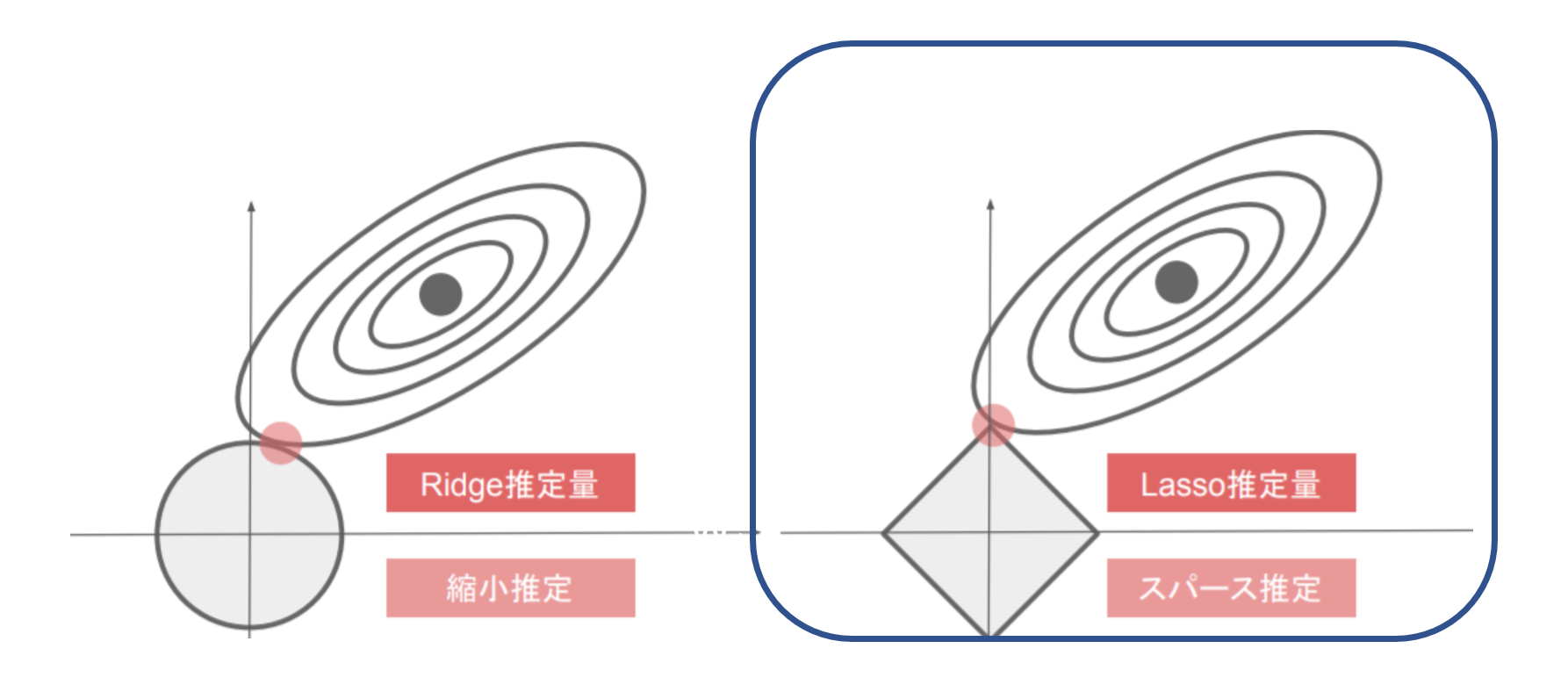
**正則化とは、ネットワークの自由度を制約すること。正則化の手法について下記があげられる。**

1. **L1正則化、L2正則化**
2. **ドロップアウト**

**ドロップアウトとは、ニューラルネットワークの学習時に、一定割合のノードを不活性化させながら学習を行うことで過学習を防ぎ（緩和し）、精度をあげるために手法（Qiitaより）。ランダムにノードを削除して学習させること。**

* **ドロップアウトのメリット**
* **データ量を変化させずに、異なるモデルを学習させていると解釈できる。**
* **確認テスト6**

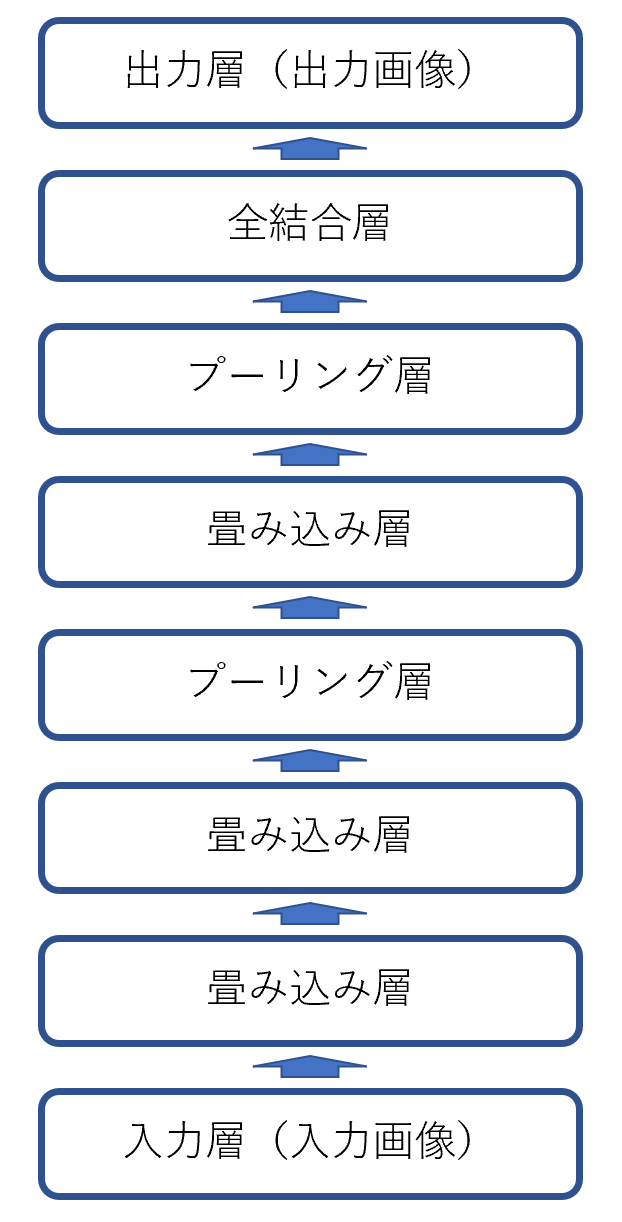
**下図について、L1正則化を表しているグラフはどちらか答えよ。**

****

**右図がL1正則化を表している。**

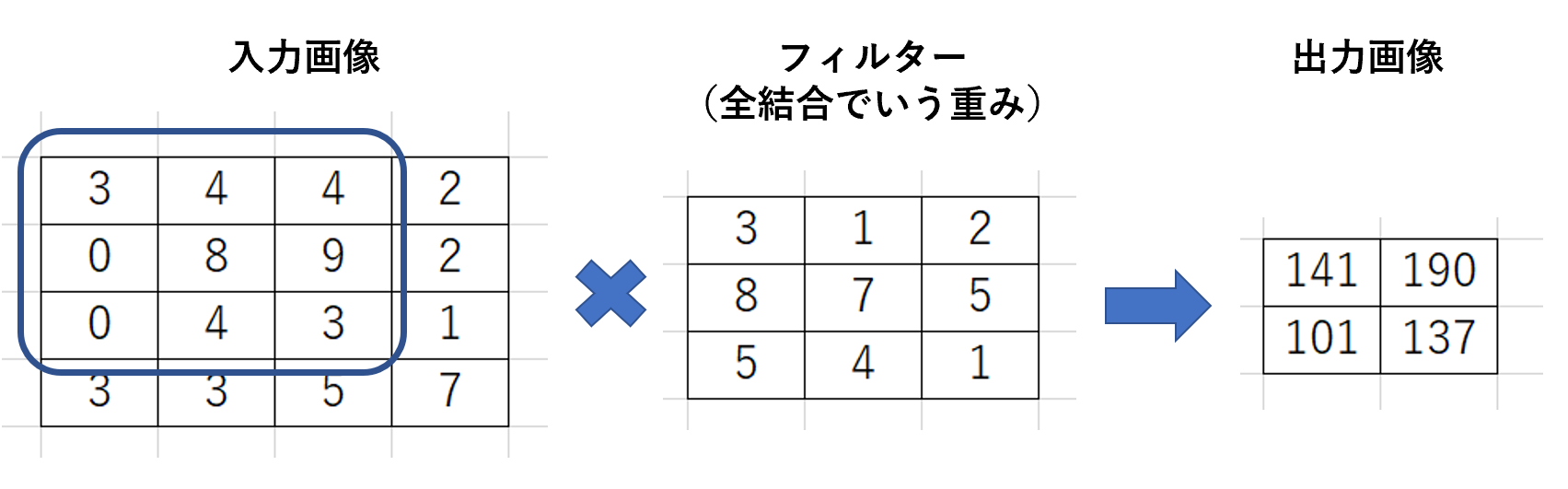
1. **畳み込みニューラルネットワークの概念**

**畳み込みニューラルネットワーク（Convolutional Neural Network）は、CNNと略されて呼ばれ、主に画像データで利用される。全体像（CNNの構造図）は下記のようになっている。**

****

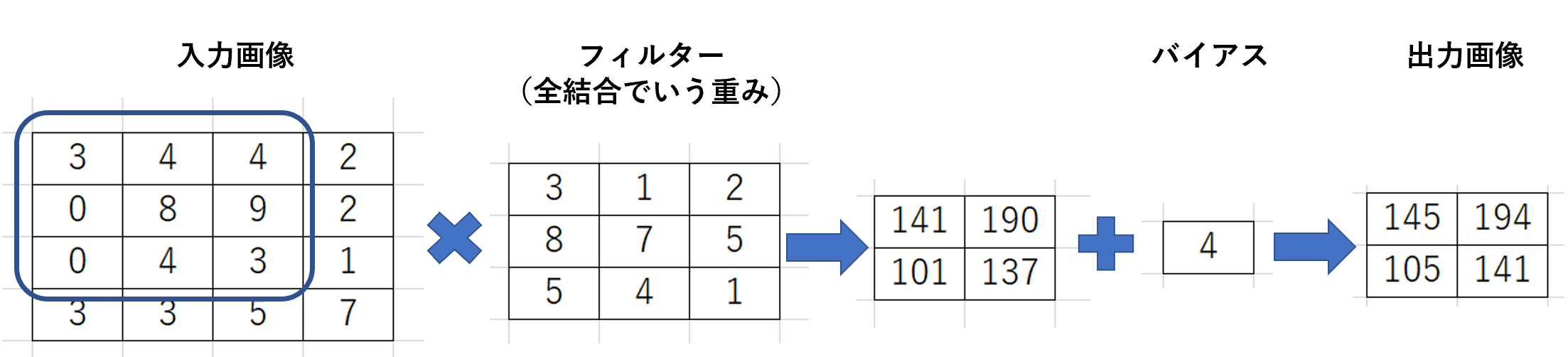
1. **畳み込み層**

**畳み込み層では、画像の場合、縦、横、チャンネルの3次元のデータをそのまま学習し、次に伝えることができる。結論として、3次元の空間情報も学習できるような層が畳み込み層である。畳み込みの演算概念は下記のようになっている。**

****

1. **バイアス**

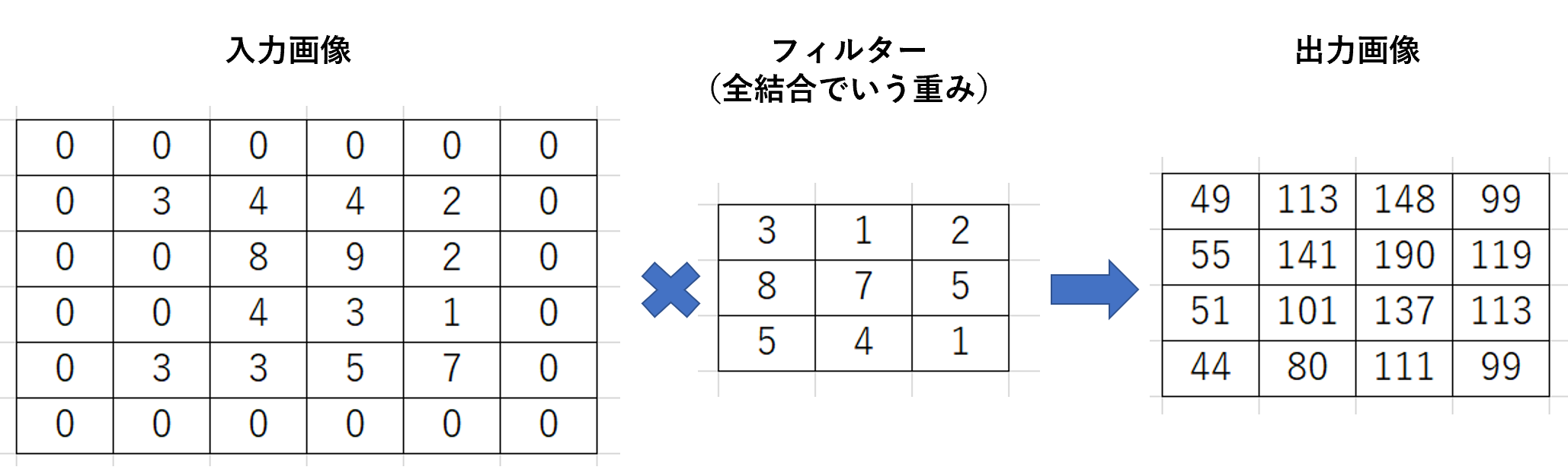
**機械学習（や統計学）のモデルによる予測においてバイアス（偏り：Bias）とは、予測値と真の値（＝正解値）とのズレ（つまり「偏り誤差：Bias error」）を指す。この予測誤差は、モデルの仮定に誤りがあることから生じる（※なお、ニューラルネットワークのニューロンおける重みとバイアスの「バイアス」とは別物なので注意）（@ITより）。**

****

1. **パディング**

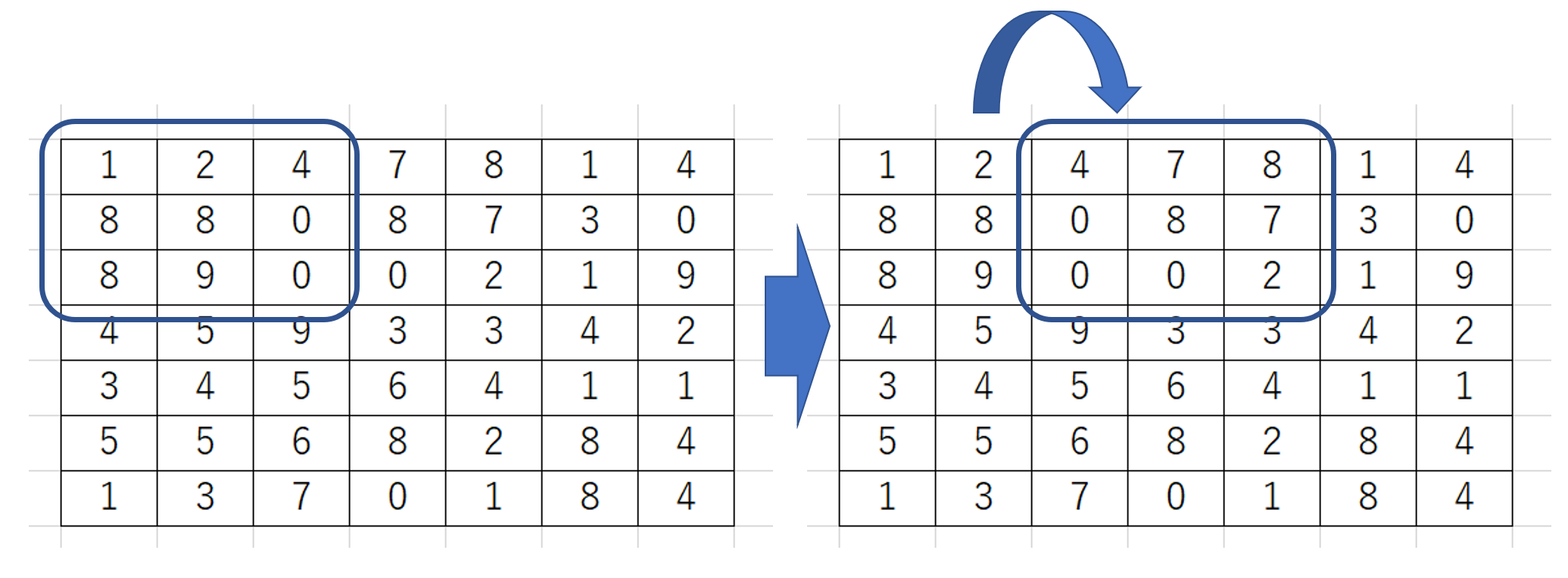
**一般的にはゼロパティングが使われ、入力画像の周りをゼロで埋める。パディングの目的は、出力サイズを調整するためである（Qiitaより）。パディングによるメリットは下記である（DeepAge 人工知能の今と一歩先を発信するメディアより）。**

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

****

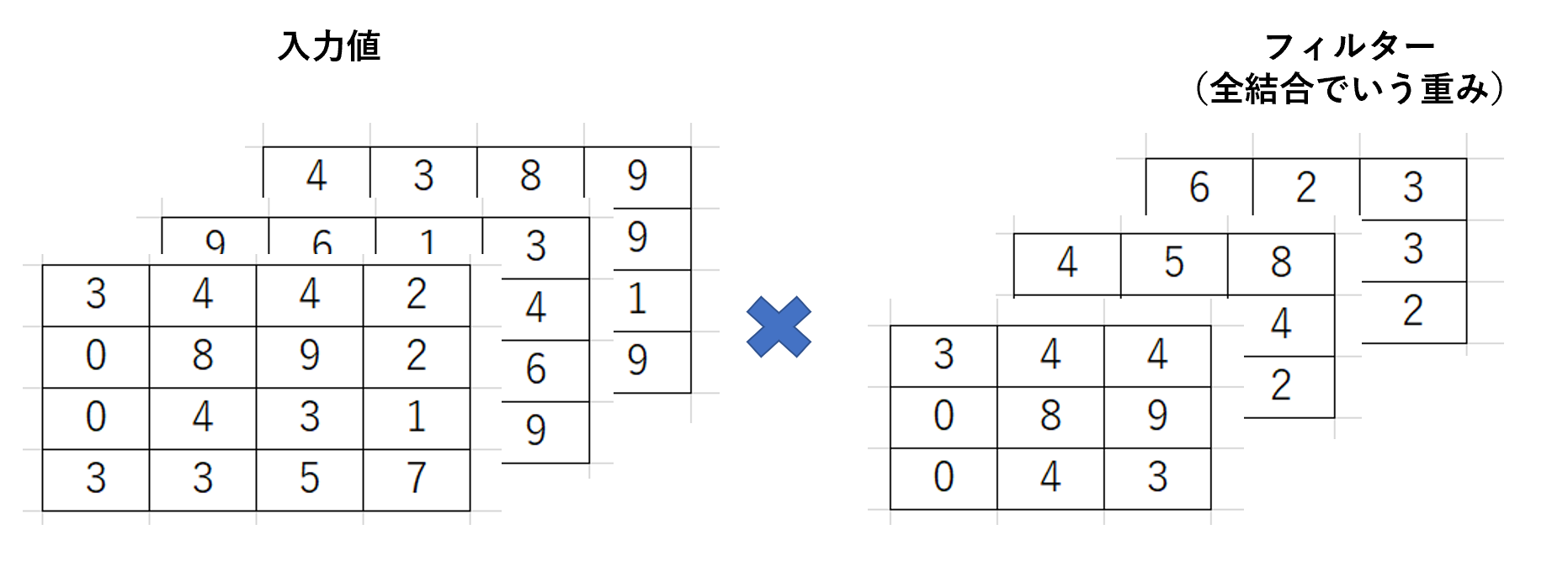
1. **ストライド**

**フィルターをずらす数値で、ストライドを大きくすると、出力サイズが小さくなる。**

****

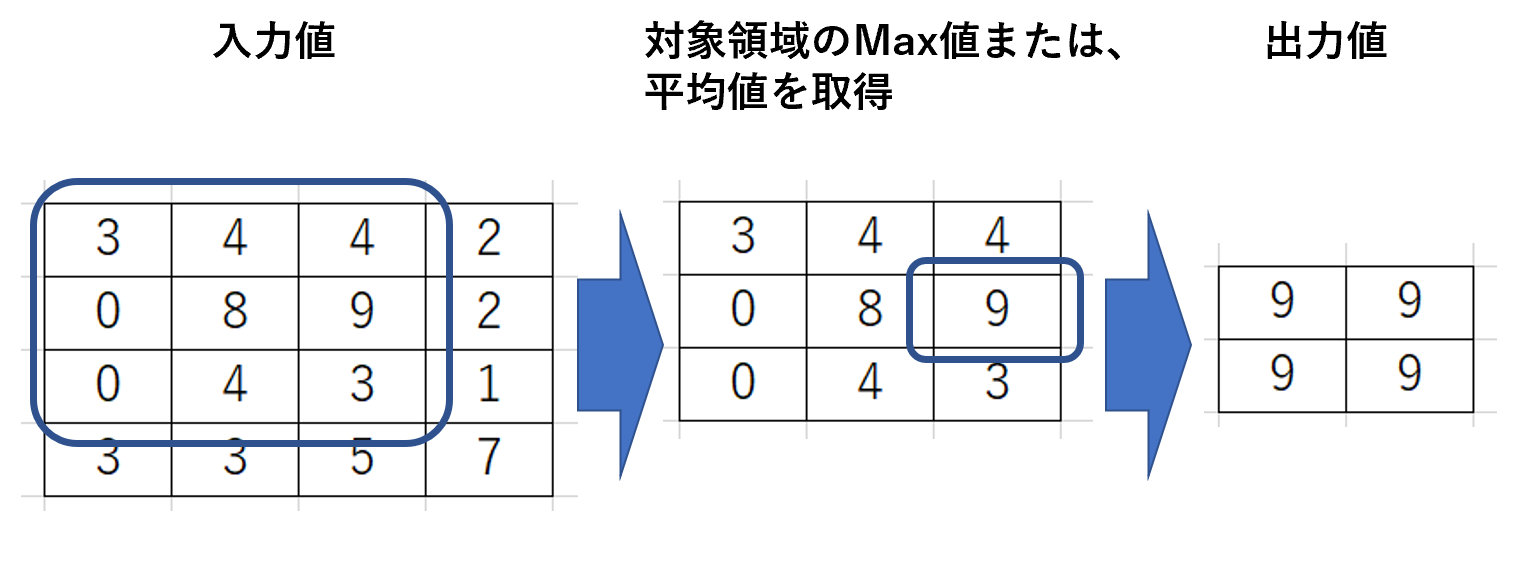
1. **チャンネル（チャネル）**

**畳み込み層を設計する際には、入力層と畳み込み層のニューロンセットを複数個、構成する。この各フィルターに対応する畳み込み層内のニューロンセットを、「チャネル」と呼ぶ（iMagazineより）。**

****

1. **プーリング層**

**プーリング層とは、入力画像の中から重要だと思われる情報のみを抜き出してデータサイズを小さくするレイヤのことで、この場合の重要な情報というのは、大抵の場合は最大値だったり平均値だったりする（Qiitaより）。**

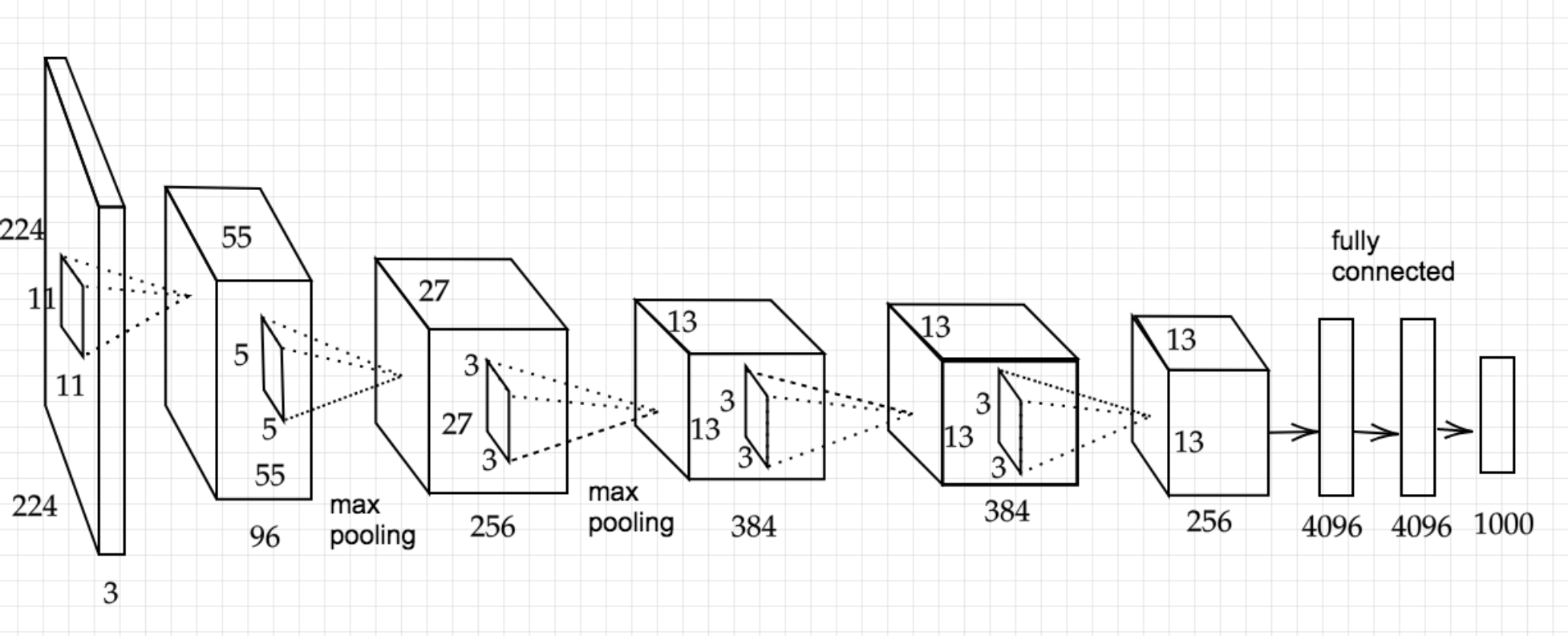
****

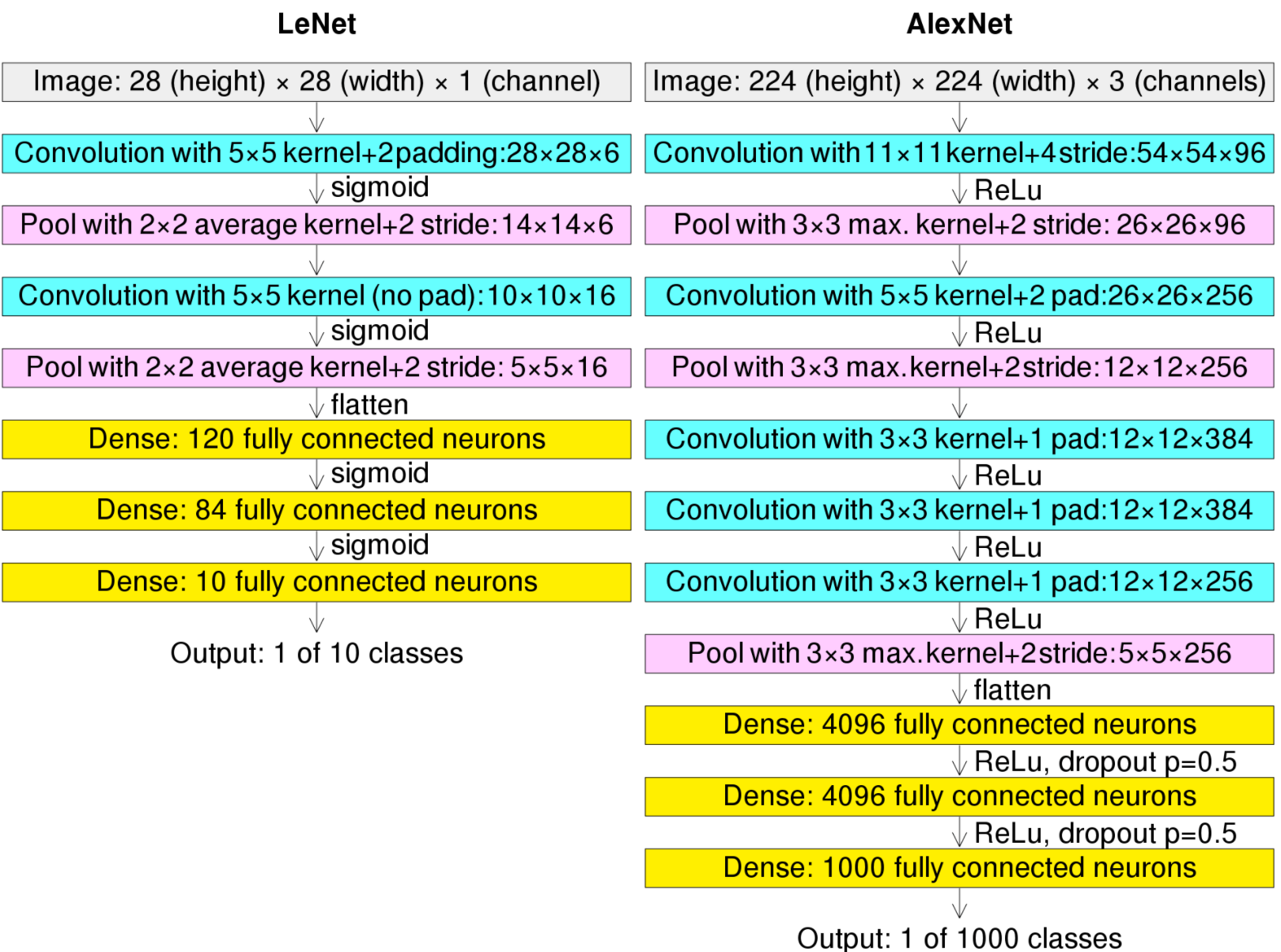
* **確認テスト7**
* **サイズ6×6の入力画像を、サイズ2×2のフィルタで畳み込んだ時の出力画像のサイズを答えよ。なおストライドとパディングは1とする。**

**7×7**

1. **最新のCNN（AlexNet）**

**AlexNet は畳み込みニューラル ネットワーク（CNN）のアーキテクチャの名前であり、Alex Krizhevsky が博士課程の指導教官である Ilya Sutskever および ジェフェリー・ヒントン と共同で設計した。AlexNet は、2012 年 9 月 30 日に開催された ILSVRC 2012[3] に参加した。AlexNet はエラー率 15.3% で優勝し、次点よりも 10.8% 以上低かった。この論文の主な内容は、モデルの深さが高性能には不可欠であるというもので、計算コストは高くなるものの、GPU を用いて学習することで実現した（wikiペディアより）。**

****

****