H-002

畳み込みニューラルネットワークを用いた視線認識
Gaze detection by Convolutional Neural Network

本田 為彬† 松本 直樹†
Tameaki Honda Naoki Matsumoto

1. まえがき

人間の視線方向や眼球運動の検出は古くから医学・生体工学などの様々な分野で行われている。現在では視線の動きを検出することで感動推定や動作予測を行うことが可能になっている為、これから先幅広い分野への応用が見込まれる。しかし、現在最も高い精度で検出可能となっている赤外線カメラを用いた視線認識の手法は使用する機器にコストがかかる為、幅広い分野での実用化が現実的ではない。また、長時間被験者の眼球に赤外線 LED を照らし続けることに対し、安全性が確立されていない。その為、赤外線 LED を用いないカメラでも従来と同じ精度で検出を行うことが望まれる。本研究では現在画像処理の分野で脚光を浴びている畳み込みニューラルネットワーク[CNN(Convolutional Neural Network)]を応用し、視線認識システムを構築することで、市販のウェブカメラにより従来の赤外線カメラによる手法と同等の精度のシステムを構築することを目的とする。

CNN は多層ニューラルネットワークに畳み込み層が追加されたネットワークであり、2012 年に開催された大規模画像認識のコンペティション(ILSVRC)のクラス分類タスクにおいてこの年に初めて登場した CNN による手法(AlexNet)が圧倒的な大差で優勝し、これまでの画像認識に対するアプローチを根本から覆した。それ以降 ILSVRC では毎年 CNN を用いた手法が優勝し続けており、より精度の高いモデルが利用されるようになり、AI の急速な発展を支える技術となった。しかし、現時点では用いるデータセットやケースによってどの手法やモジュールを用いることが最適なのかは判明されていない。本研究では歴代の ILSVRC での優勝モデルを参考にし、ネットワークの構造に注目することで、どのモジュールや層を用いることがネットワークの精度を向上させるのかを検討し、視線認識システムにより適したネットワークの構築を行い、評価を行う。また、CNN のネットワークの構築ではパラメータ数や学習に関わる時間が議論になるが、本研究ではネットワークの構造にのみ着目することにする。

2. 畳み込みニューラルネットワーク

これまでの ILSVRC で優勝した CNN で用いられた層について紹介する。3 章で述べるが本研究で利用するモジュールにも以下の層が様々な形で用いられる。

2.1 Convolution 層

CNN のメインとなる層であり、この convolution 層がニューラルネットワークに追加されたことにより最近の画像処理の精度は飛脚的に向上した。convolution 層では入力画像に対して畳み込みを行い、用いる階層によってストライドやフィルタのサイズを変更する。入力される画像は三次元配列で表され、W×W×N と表記すると、W は入力画像の縦横サイズ、N はチャネル数である。畳み込むフィルタは同様に P×P×N の形を持ち、P<W でありチャネル数 N は必ず同じ値をとる。

第 l 層にある畳み込み層が M 個のフィルタを持つとき、W×W×N の配列の入力をz_{ijn}^{l-1}、全フィルタの要素をh_{pqnm}(p,q = 0…P-1, n=0…N-1, m=0…M-1)、バイアスをb_mとするとこの l 層の出力z_{ijn}^{l}は次のように表される。

$$z_{ijm}^{l} = \sum_{n=0}^{N-1}\sum_{p=0}^{P-1}\sum_{p=0}^{P-1} z_{i+p,j+q,n}^{l-1} h_{pqnm} + b_m \qquad (1)$$

また後述する活性化関数を適用し、z_{ijm}^{l}は次の層の入力となる。

2.2 Pooling 層

convolution 層の後に対で使われ、特徴として重要な情報を残しながら元の画像を縮小する役割があり、ネットワーク全体で画像サイズを大きく縮小させる層である。pooling 層には、フィルタにより選択された小領域の最大値を出力する max pooling や、選択された小領域の平均値を出力する average pooling があるが、CNN では主に max pooling が用いられる。convolutioin 層とは異なり、ストライドは必ず 2 以上である。このストライドが 2 の場合は、出力は 1/2 になることを示す。Pooling 層のフィルタサイズとストライドの値は同じである。

2.3 ReLU 層

Rectified Linear Unit の略であり、入力した値が 0 以下のとき 0 を出力し、正の値をとるときにその値を出力させる活性化関数である。従来はシグモイド関数や tanh(x)が利用されていたが、ReLU が活性化関数として利用されるようになったことで、深いネットワークで発生する勾配消失問題の解消や、学習速度短縮に貢献している。

3. ILSVRC での歴代の優勝モデル

ILSVRC での CNN による歴代の優勝モデルについて述べる。ILSVRC の分類タスクでは数百万枚の画像を 1000 のカテゴリーに分類し精度を競うため、出力は全て 1000 である。本研究で用いるデータセットを本章で紹介する全てのネットワークで画像分類の推論を行うことで、各ネットワークやモジュールの特性を理解し、精度の高いモデルの構築に活かす。また、5 章では本章で紹介したモジュールや構造を組み合わせることで新たに提案するネットワークについても触れる。

3.1 AlexNet[1](2012 年優勝モデル)

AlexNet は convolution 層 5 層と全結合層 3 層の全 8 層からなるネットワークであり、1,2,5 層目の convolution 層の後にはサイズが 3×3 のストライドが 2 である maxpooliing が用いられる。1 層目の convolution 層はフィルタサイズが 11×11 のストライドが 4 であり、2 層目はフィルタサイズが 5×5 のストライドが 1 である。その後の convolution 層では全てサイズが 3×3 のストライドが 1 であり、8 層目の全結合層では Dropout と呼ばれる、学習時に隠れ層のニューロンをランダムに半分消される動作が行われる。この Dropout の働きにより、データによ

†明治大学大学院理工学研究科電気工学専攻
Electrical Engineering, Meiji University, Kanagawa, Japan.

りネットワークが過剰に学習されてしまう過学習を抑制することができる。

3.2 ZFNet[2](2013 年優勝モデル)

ZFNet は構造の面のみに着目すると AlexNet と非常に似ている。AlexNet での 1 層目で極端に高周波と低周波の情報を取得してしまう問題を指摘し、1 層目のフィルタサイズを 11×11 から 7×7 に縮小し、ストライドを 4 から 2 に縮小した。2 層目の convolution 層ではストライドを 1 から 2 に拡大するが、それ以外は AlexNet と全く同じ構造をとる。

3.3 GoogleNet[3](2014 年優勝モデル)

GoogleNet の一番の特徴は、複数の convolution 層や pooling 層から成る Inception モジュールと呼ばれる小さなネットワークを用いることであり、この Inception モジュールを組み合わせることで、大きなネットワークを作り上げている。GoogleNet で用いられる Inceptioin モジュールを図 1 に示す。Inception モジュールではネットワークを分岐させ、フィルタサイズの異なる畳み込みを行い、その出力をつなぎ合わせる構造をとる。この目的は、convolution 層の重みを sparse にし、パラメータ数のトレードオフを改善することである。分岐した後の convolution 層の前に 1×1 の convolution 層を置くことで、次元削減を行なっている。

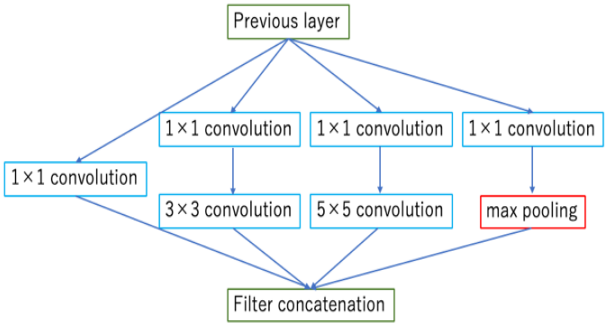

図 1 Inception モジュール

3.4 ResNet[5](2015 年優勝モデル)

2014 年に GoogleNet に次ぐ準優勝した VGGNet では、CNN において層の数が深いほど精度が上がることが証明された。しかし、むやみに層を増やすだけでは勾配消失問題や過学習が起こり、効率的な学習が困難になる。そこで ResNet では処理ブロックによる変換 F(x)を単純に次の層に渡していくのではなく、その処理ブロックの入力 x を加算し、F(x)+x を次の層に渡す手法を提案した。この図を図 2 に示し、このモジュールを Residual モジュールと呼ぶ。ResNet はこの Residual モジュールのショートカットが逆伝搬時に直接下の層に伝わっていくことで、深いネットワークでも勾配消失問題を起こさないモデルを作成することができた。[5]

Residual モジュールは 3×3 の convolution 層の後再び 3×3 の畳み込みを行い、ショートカットにより入力を足し合わせた後に ReLU 層を配置する。各 convolution 層の後に配置されている bn は batch normalization を示し、パラメータの更新によって次の層への分布が変化してしまう内部共変シフトを正規化させて学習を円滑に行う役割がある。

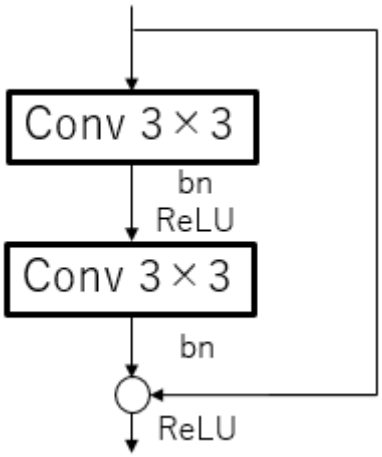

図 2 Residual モジュール

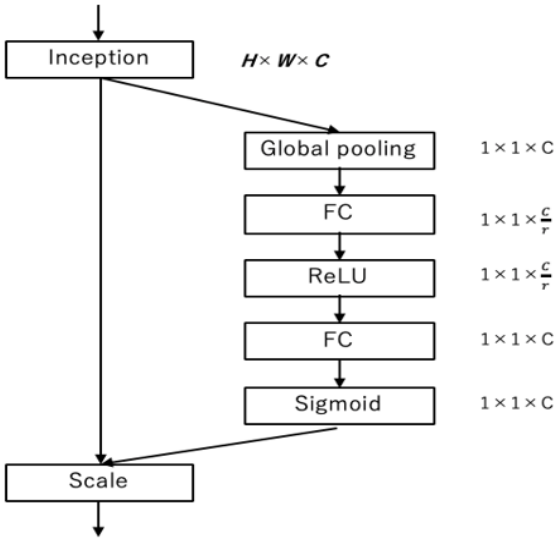

図 3 SE ブロック(Inception)

3.5 SENet[6](2017 年優勝モデル)

SENet は特に提案されたネットワークがあるわけではなく、従来の GoogleNet や ResNet に SE ブロック(Squeeze-and-Excitation)を追加することで、高い精度のモデルを作成することができたネットワークの総称である。

SE ブロックでは通常の convolution 層の出力をそのまま使うのではなく、各チャンネルを計算し、最後に出力を重み付けすることで、チャンネル間の関係性の学習を重点に置いた構造である。GoogleNet に SE ブロックを加えることを例にとり、Inception モジュールと SE ブロックの構造を図 3 に示す。初めに、Squeeze で画像全体の特徴を得る為に、Global average pooling を行う。そこからチャンネル/ratio に個数を減らし、非線形の形をとり、またチャンネル数を戻し、最後にシグモイド関数で計算を行

う。その後、計算された値で各チャンネルに重み付けを行う。FC は Fully Connection(全結合層)を表す。

SE ブロックを追加する利点は他のネットワークのモジュールや convolution 層の後に付け加えるだけで簡単に実装することが可能であり、また計算量も 1%程度しか増加しないため、学習時間の増加も抑えることが可能である。例えば ResNet-50 に SE ブロックを加えることで、ResNet-101 と同等の精度を出し、且つ半分の計算量でネットワークを構築できることが文献[6]で証明されている。

4. 実験方法

本章では本研究で用いるデータセットの取得方法からデータの水増し、3 章で紹介した CNN を用いて視線認識を行う方法について述べる。

4.1 データの取得方法

まず、被験者は標的を映し出すモニターに対し 1,000 mm の位置に着席し、被験者の前に、被験者とモニターの直線上に被らぬ位置に Web カメラを設置する。被験者はモニターに映し出された 9 つの標的を順番に目視し、その間に被験者がそれぞれの標的を目視する際の目元付近の画像のみを Web カメラから OpenCV により読み込む。本研究で用いるモニターのサイズは、26 インチであり、標的はモニターの中央、上下左右、各角に配置する。

4.2 データの水増し

CNN の分野において、画像分類に必要なデータセットの枚数は、最低でも数千から数万だと言われている。そこで、データ数を増やす為に、様々な手法によりデータの水増しを行う。AlexNet など 3 章で紹介したモデルの論文では、水平方向の反射や RGB 変換によってデータの水増しを行っているが、本研究で用いるデータは、色の変換やデータの方向ベクトルを変更すると実際の結果に影響が出てしまう為、従来のデータの水増しの手法を用いることができない。そこで本研究では、色による影響や、方向を変更しない手法を用いてデータの水増しを行う。

本研究では 5 つの手法を用いてデータを 8 倍に水増しする。1 つ目はコントラスト調整によりデータを 2 枚に増加させる。0～255 の値で表現される RGB 形式の画像を、30～225 に変換し、明るくした画像、暗くした画像を 2 枚生成する。2 つ目はガンマ変換により、画像を 2 枚生成する。ガンマγの値はそれぞれ、本研究では 0.75,1.5 で行う。3 つ目は平滑化フィルタであり、本研究では 7×7 のフィルタをかける。4 つ目はガウシアンノイズをかけ、5 つ目はインパルスノイズにより画像を生成する。

4.3 CNN による視線認識

本研究では 70 人の被験者により 9 方向を目視する時の画像を取得し、データを 8 倍に水増しを行う。よって、70×9×8＝5,040 枚の画像を用いて実験を行う。そのうち 4,320 枚を訓練画像(ネットワークの学習のみに用いる画像)として用意し、残りの 720 枚をテスト画像(学習には関与せず、ネットワークの精度の評価を行う画像)として用意する。本来理論上、ILSVRC で 2012 年から 2017 年まで優勝してきたモデルは最近になるにつれて精度が高いネットワークであることになるが、本研究で用いるデータセットは ILSVRC の分類タスクのように、多種多様の画像を分類するわけではなく、似たデータ(全て目元の画像)を用いて行うため、同じように結果が出ない可能性があることが考えられる。似ているデータの分類はどのような構造やモジュールが適しているか傾向を掴むために、まず 3.1 章～3.4 章で紹介した AlexNet,ZFNet,GoogleNet,ResNet を用いて精度の評価を行い、その後それぞれのモデルに SE ブロックを加え、計 8 つのネットワークで評価を行い、どのネットワークが本研究のネットワークとして適しているのかのシミュレーションを行う。その後、シミュレーション結果を元に、新たにネットワークを作成し、評価を行う。尚、ILSVRC の画像分類タスクでは 1000 種類に画像を分類する為、通常ネットワークの出力は 1000 であるが、本研究では出力は 9 としてネットワークを構築する。

本研究のシミュレーションでのミニバッチサイズは 50、epoch 数は 1000 とし、入力画像のサイズは全て 224×224 に統一する。(OpenCV により取得した目元付近の画像のサイズは 100×100 であるが、リサイズを行う。) 学習は勾配降下法を用い、パラメータの更新方法は Adam を用いる。学習率の初期値は10^{-4} とし、また過学習抑制の目的でランダムにニューロンを消しながら学習を行う、Dropout の割合は、本研究では 0.5 に統一する。(半分のニューロンを消しながら学習を行う。) また、今回シミュレーションに用いる GPU は GTX-1060 6GB である。

5. シミュレーション

5.1 各モデルでシミュレーション

前節で説明した、AlexNet,ZFNet,GoogleNet,ResNet の 4 つのネットワークとそれぞれに SE ブロックを追加した 4 つの計 8 つのネットワークで学習を行う。AlexNet,ZFNet の二つは convolution 層の後ろにある ReLU 層の後ろに SE ブロックを配置し、GoogleNet では Inception 層の後ろに、ResNet では、Residual モジュール内で最後に入力を足し合わせる add の前に SE ブロックを配置する。Squeeze の際に行うチャンネル数を割る ratio の値は、SE-ResNet 以外は全て 16 に統一し、SE-ResNet では 16,32,64,128 をそれぞれ代入する。それぞれに対しての訓練画像、テスト画像でのシミュレーション結果を表 1 に示す。

表 1　8 つのネットワークの訓練画像、テスト画像での正解率(%)

	訓練画像	テスト画像
AlexNet	100	87.4
SE-AlexNet	100	92.6
ZFNet	100	86.3
SE-ZFNet	100	90.1
GoogleNet	14.0	12.2
SE-GoogleNet	94.0	91.0
ResNet	58.0	51.0
SE-ResNet	86.0	83.3

全章で述べたが、理論上後に考案されたネットワークほど精度が高くなるはずだが、本研究ではネットワーク毎に大きくそれぞれの特徴が出ていることが見られる。まず SENet(SE ブロック)の精度に注目すると、SE ブロックを用いてないネットワークと SE ブロックを用いたネットワークでは SE ブロックを用いる方が圧倒的に高い精度を示し

ている。また、AlexNet や ZFNet など、CNN でのシンプルな構造をしたネットワークが最も高い精度を示したが、訓練画像での学習で 100%の精度が出ているにも関わらず、学習に用いないテスト画像での検証結果は 9 割程度の精度のため、学習に用いるデータのみを過剰に学習してしまい、汎用的なネットワークが作成できていない可能性がある。また GoogleNet や ResNet のように複雑で深いネットワークは精度が悪かったため、本研究のデータセットのように目元付近の画像同士のような似たデータを用いる場合は適していない。しかし、SE ブロックを追加させることで大きく精度が改善されたため、SE ブロックは本研究において非常に大きな役割を果たしている。また、訓練画像とテスト画像の正解率が近いため、汎用的なネットワークが作成できているので、ネットワーク構造の改良次第でこの先さらに精度の高いモデルを作成できることが見込まれる。

5.2 ネットワークの提案

5.1 の結果から、GoogleNet や ResNet など複雑で無駄に深い構造をとるネットワークは逆に精度が下がることが実証されたため、よりシンプルで且つ汎用的なネットワークの作成を行う。一つは AlexNet や ZFNet を改良したモデルであり、表 1 の結果が示すように SE-AlexNet が最も高い精度を出したことから、一層目の出力サイズを揃え、一層目の convplution 層のフィルタサイズを 11×11 から 7×7 変更し、フィルタの数を減少させシンプルな構造のネットワークを作成した。これを Suggession1 とし、ネットワーク構造を表 3 に示す。また参考にした SE-AlexNet の構造を表 2 に示す。

表 2 SE-AlexNet のネットワーク構造

name	Output size	Filter
input	227×227,3	
conv1	55×55,96	11×11 stride4 SE[16]
pooling1	27×27,96	3×3 stride2
conv2	25×25,256	5×5 stride1 SE[16]
pooling2	13×13,256	3×3 stride2
conv3	13×13,384	3×3 stride1 SE[16]
conv4	13×13,384	3×3 stride1 SE[16]
conv5	13×13,256	3×3 stride1 SE[16]
pooling3	7×7,256	3×3 stride2 SE[16]
fc	4096	dropout
fc	4096	
fc	1000	

もう一つは、GoogleNet や ResNet で用いた Inception モジュールや Residual モジュールを改良し、且つ複雑な構造を取らないネットワークを提案する。表 1 で高い精度が出た SE-GoogleNet と SE-ResNet を改良し、高い精度が出た Se-ResNet を Suggession2 とし、提案する。Suggestion2 で用いる Residual モジュールでは、図 2 で示した Residual モジュールとは異なり、batch normalization を一つ目の convolution 層の前に配置し、二つ目の convolution 層の後ろに SE ブロックを配置し、勾配をそのまま入力に近い層に伝えていく目的でショートカットの後の ReLU 層を取り除いた。改良した Residual モジュール全体の構造を図 4 に示し、ネットワーク構造を表 5 に示す。また、参考にした SE-ResNet-34 のネットワーク構造を表 4 に示す。表の Output size は(フィルタサイズ×フィルタサイズ,チャンネル数)で表現し、Filter に、用いるフィルタサイズ、ストライド数、SE ブロックの ratio を示す。尚、SE ブロックは SE[ratio]で示す。fc は fully connection(全結合層)を表す。

表 3 Suggestion1 ネットワーク構造

name	Output size	Filter
input	112×112,3	
conv1	56×56,64	7×7 stride2 SE[16]
pooling1	28×28,64	3×3 stride2
conv2	28×28,128	5×5 stride1 SE[16]
pooling2	14×14,128	3×3 stride2
conv3	14×14,256	3×3 stride1 SE[16]
conv4	14×14,256	3×3 stride1 SE[16]
pooling3	7×7,256	3×3 stride2 SE[16]
fc	1000	dropout
fc	9	

表 4 SE-ResNet-34 のネットワーク構造

name	Output size	Filter
input	224×224,3	
conv1	112×112,64	7×7 stride2 SE[16]
pooling1	56×56,64	3×3 stride2
residual×3	56×56,64	3×3 stride1 3×3 stride1 SE[16]
residual×4	28×28,128	3×3 stride2 3×3 stride1 SE[32]
residual×6	14×14,256	3×3 stride2 3×3 stride1 SE[64]
residual×3	7×7,512	3×3 stride2 3×3 stride1 SE[128]
pooling2	512	7×7 average pooling
fc	1000	

表 5 Suggestion2 ネットワーク構造

name	Output size	Filter
input	112×112,3	
conv1	56×56,64	7×7 stride2 SE[16]
pooling1	28×28,64	3×3 stride2
residual×3	28×28,64	3×3 stride1 3×3 stride1 SE[16]
residual×4	28×28,128	3×3 stride1 3×3 stride1 SE[32]
pooling2	14×14,128	3×3 stride2
residual×6	14×14,256	3×3 stride1 3×3 stride1 SE[64]
pooling3	7×7,256	3×3 stride2
residual×3	7×7,512	3×3 stride1 3×3 stride1 SE[128]
pooling4	512	7×7 average pooling
fc	9	

表 6 提案する 2 つのネットワークの訓練画像、テスト画像でのシミュレーション結果(%)

	訓練画像	テスト画像
Suggestion1	100	96.7
Suggestion2	100	97.6

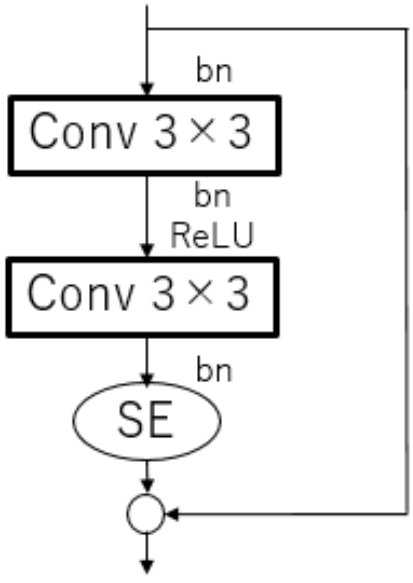

図.4 Suggestion2 での Residual モジュール

続いて提案する二つのネットワーク Suggestion1, Suggestion2 を 4 章で作成した本研究のデータセットを用いてシミュレーションを行った。それぞれのネットワークでの訓練画像、テスト画像でのシミュレーション結果を表 6 に示す。

Suggestion1 は参考にした 4 つのネットワークで最も精度の良かった SE-AlexNet よりテスト画像での評価が 4.1%高く、Suggestion2 は参考にした SE-ResNet34 よりテスト画像での評価が 14.3% 高い精度が出た。Suggestion1 は convolution 層と pooling 層を重ねただけの形である為、同様の構造ではこれ以上精度の向上は見込めない可能性がある。Suggestion2 では Residual モジュール内の構造の変更や、Residual モジュール内の SE ブロックの位置の変更、GoogleNet を足し合わせること等で更に精度が向上する可能性があるので、今後の課題である。

6. 考察及び今後の展望

CNN による視線認識システムは学習に関与しないテスト画像での認識精度が 100%にはならなかったが、高い精度で認識することが可能であることを示した。誤認がある理由としては、使用するデータセットが少ないことや、Inception モジュールや Residual モジュールが最適でないこと、容易にモジュールを組み合わせることが難しいことが考えられるが、CNN はブラックボックス化してしまう為、最適なシステムの提案が難しいことが欠点である。今後は Residual モジュール内の構造や SE ブロックの配置の位置に焦点を合わせ、どの層がネットワークの精度に影響を与えているかを明確にし、最適なネットワークの作成を行うことが課題である。

CNN を用いた視線認識はあらかじめ決められた方向を見ている時のデータを用いて学習を行うため、複数の角度への認識を行うことは難しい。しかし、特定の角度や位置を目視している時の認識を行うことは可能である。よって、自動車運転時の運転手のよそ見検知や、試験中のカンニング防止システムなどへ応用することが可能である。

参考文献

[1]A.Krizhevsky I.Sutskever G.E.Hinton ImageNet classification with Deep Convolutional neural network. In Proc. of NIPS,2012

[2]M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In Proc. of ECCV,2014

[3]C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proc. of CVPR,2015

[4]K.Simonyan and A.Zisserman. Very deep convolutional networks for large-scale image recognition. In Proc. of ICLR,2015

[5]K.He, X.Zang, S.Ren, and J.Sun. Deep residual learning for image recognition. In Proc of CVPR,2016

[6]J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. arXiv:1709.09844,2017