卒 業 研 究 報 告 題 目

CNNを利用した柔道競技における ポイントの判定

Judgement of Points on Judo using CNN

指 導 教 員　　出口 利憲 教授

岐 阜 工 業 高 等 専 門 学 校　電気情報工学科

2015E23 日下部完

令 和 2 年（ 2 0 2 0 年 ）　　2 月 1 4 日 提 出

Abstract

In the sports, error judgement is the ploblem we must solve. So, many sports are adapting systems to sovle it such as video judging, chalenge system etc.

These days, the system that can solve complicated problems like a human is growing in high speed called AI. AI is used on wide fields. For example automatic translation system and automatic driving system.

I think of developing the application that calculates the score of Judo automatically and that reduces error judges in Judo. In this study, I studied judge assistance in Judo using image recognization and tracing players in a screen using object detection to achieve the goal.

In image recognization, I used model of CNN known as achieving high accuracy. And I used model of SSD that is recently revealed in object detection.

目 次

Abstract

第 1 章 Introduction **1**

第 2 章 Machine learning **2**

2.1 人間と機械 2

2.2 従来のプログラムの限界 2

2.3 機械学習 2

第 3 章 Neural network **4**

3.1 ニューロンのモデル化 4

3.1.1 ニューロン 4

3.1.2 モデル化 5

3.2 ニューロンのネットワーク化 5

3.3 回帰と分類 7

3.3.1 回帰 7

3.3.2 分類 8

第 4 章 Train **9**

4.1 学習則 9

4.1.1 ヘッブ則 9

4.1.2 デルタ則 10

4.2 損失関数 10

4.2.1 二乗和誤差 10

4.2.2 交差エントロピー誤差 11

4.3 勾配降下法 12

4.4 バックプロパゲーション 14

第 5 章 Convolution neural network **16**

5.1 畳み込み層 17

5.2 プーリング層 17

5.3 全結合層 18

5.4 パディング 18

5.5 ストライド 19
5.6 ReLU 19
第 6 章 Object detection **21**
6.1 デフォルトボックス 21
6.2 評価指標 21
6.3 jaccard 係数 21
6.4 ハードネガティブマイニング 22
第 7 章 SSD **23**
7.1 モデル 23
7.1.1 出力サイズ 23
7.2 デフォルトボックスのスケールとアスペクト比選択 23
7.3 損失関数 23
7.4 速度 25
第 8 章 CNN を用いた柔道競技中の状態認識の実験 **26**
8.1 実験目的 26
8.2 実験方法 26
8.3 実験結果・考察 26
第 9 章 SSD による選手の追跡 **29**
9.1 実験目的 29
9.2 実験内容 29
9.3 実験結果・考察 29
9.4 追加実験内容 32
9.5 追加実験結果・考察 32
第 10 章結論 **34**
謝辞 **35**
参考文献 **36**

第1章　Introduction

スポーツ競技において、審判員による誤審がよく問題となる。選手にとって選手生命をかけた大会に、誤審で敗れてしまうことは何としても避けるべき問題である。この、誤審を防ぐため、各競技は様々な対応をとっている。例としてテニスを挙げる。テニスでは、チャレンジシステムという、選手自身が審判員に申告することにより計算機で予想された球の弾道に基づきスコアを再度設定するシステムである。私は今回、スポーツの中でも柔道競技に焦点を置いた。

柔道競技では、試合場を自由に歩き回れる主審が1人と、試合場対角線に着席している副審2名の計3人の人間によって判定されていた。これが、1994年にジュリーと呼ばれる試合場外から試合を停止、意見する権利を持った審判委員が追加され計4人によって試合が判定されるようになる。それでも誤審は後を絶たず、2000年にはCAREシステムと呼ばれる試合場を撮影した3台のカメラを導入した。スコアの判定が人間の目によっては瞬時に判断できない場合、CAREシステムで撮影された映像をジュリーが複数回確認することによって最終的なスコアを判定する。

現在では最後に記した方法で、柔道競技が進行されている。これにより限りなく誤審が抑制される傾向にあるが最終的には人間の判断であることは変わらないため、人為的な誤差を取り除くことはできない。さらに、CAREシステムの導入には大幅なコストがかかることなどから、国際的な試合などの大規模な大会でしか反映されない。

そこで私は、現在急激な技術の発展を見せている人工知能(AI)を利用し、選手を認識し、瞬時にスコアを判定するアプリケーションを作成することにより、以上の問題を解決することができないかと考えた。この論文の目的は柔道競技において試合進行を補助するアプリケーションプログラムの作成である。

以上の目標を達成するため、作業を以下のように階層化した。

1. 試合中の画像を認識し、選手の状態を判断する。
2. 試合場全体を撮影した動画から、競技者を追跡する。
3. 2.で追跡した動画を1.で状態を判断し、技の判定を行う。

第2章　Machine learning[1),2)]

2.1　人間と機械

人間の体の中で、最も素晴らしい能力を備えているのが脳という器官である。脳は視覚や聴覚、嗅覚、味覚、触覚など感覚器官を全て司っている。脳の働きによって我々人間は、単純な反射運動のみを行う原子生物と区別される。

人間の赤ちゃんは、生後わずか数カ月で両親を認識し、物体と背景を区別でき、声を聴き分けることが可能である。生後 1 年では、自然界の物理を直感的に理解でき、対象の一部または全部が物陰に隠れていても動きを追跡でき、音声とその意味を関連付けられるようになる。さらに幼少期の早い段階で数千の語彙や正しい文法を身につけることが出来る。

過去数 10 年にわたり、人間と同じような脳を持った知的な機械をつくることが目標とされてきた。

2.2　従来のプログラムの限界

従来のプログラムは、明示された順に沿った計算はできるが、抽象的な問題を解決することは苦手である。手書き数字 (0〜1) の認識を例として挙げる。手書き数字は、全てが違う形をしている。従来のプログラムでは、それぞれの数字にルールを設定し、区別するという手法で判断されていた。そのため、人間からしてみれば、容易に判断できるような数字であっても設定されたルールでは区別できないような数字に対してはプログラムは判定することができない。そのため、従来のプログラムで手書き数字を正確に区別するには多数のルールを設定する必要がある。慎重な観察と長期の試行錯誤を経て、多数のルールを追加していくことは可能であるが、明らかに面倒である。

このような物体の識別、他にも音声認識や自動翻訳でも同じような問題に直面してしまう。

2.3　機械学習

幅広い人工知能 (AI) に関する研究の一分野に機械学習というものがある。近年巷をにぎわしているディープラーニングは機械学習の一部である。(Figure 2.1)

機械学習では 2.2 節のような問題解決の為にルールを大量に与えるといったことは行

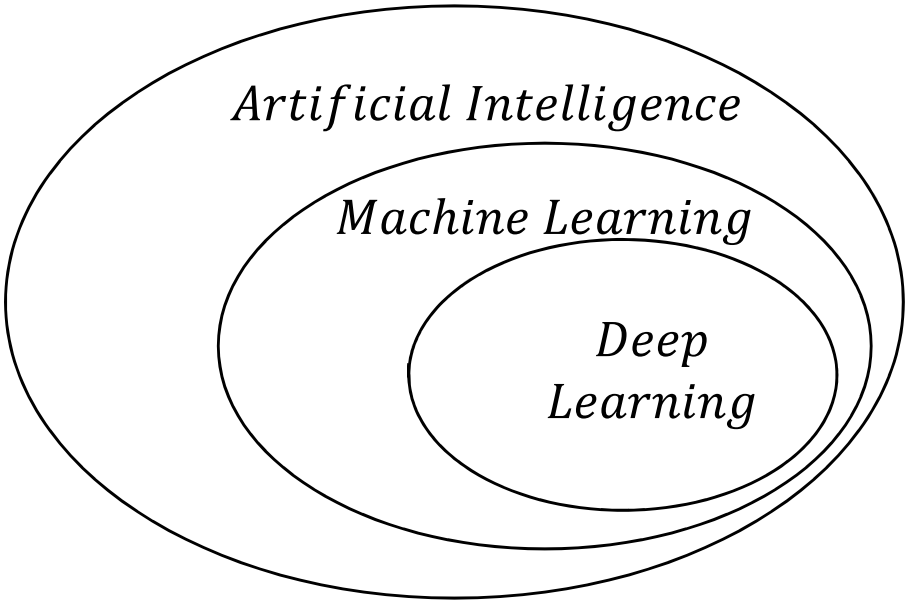


Figure 2.1　Categorization of AI, Machine learning, and Deep Learning.

わない。その代わりに入力された実例に対して判断を行うためのモデルと、判断が誤った場合にモデルを修正するための簡単な手順の 2 つが与えられる。このように徐々に現在のモデルを理想的なモデルに修正していくことは学習と呼ばれる。学習を重ねるに連れ、モデルの精度が向上し、きわめて正確に問題を解決できるようになることが見込まれる。学習回数は epoch と呼ばれる。

第3章　Neural network[1),2)]

3.1　ニューロンのモデル化

3.1.1　ニューロン

人間の脳を構成する基本的な要素は、ニューロン (neuron) と呼ばれる。脳から米粒ほどのごく小さな範囲を切り出しても、その中には 10,000 個以上のニューロンが含まれている。それぞれのニューロンは平均 6,000 もの他のニューロンと接続されている。この大規模な生物学的ネットワークを使い、生物は外界を知覚する。このニューロンをモデル化することによりプログラムが人間の脳と同等のやり方で問題を解決していくことが出来る。

ニューロンは、他のニューロンから情報を受け取り、その情報を独自の方法で処理し、結果を他の細胞に伝達する。ニューロンはこの手段を効率よく行えるように最適化されている。Figure 3.1 はニューロンの概略である。ニューロンは樹状突起 (Dendride) と呼ばれるアンテナのような構造を介して情報を受け取る。これらの接続の 1 つ 1 つは、その使用頻度に応じて動的に強弱が調整される。このような流れで生物は新しい概念を学習する。ニューロン間の接続の強度が、ニューロンからの出力に対するそれぞれの入力の貢献度 (重み) を表す。接続の強度によって重み付けが行われた入力の値は、細胞体 (Cell Body) の中で合計が計算される。この合計値は新しい出力信号へと変換され、細胞の軸索 (Axon) を通じて他のニューロンへと伝達される。

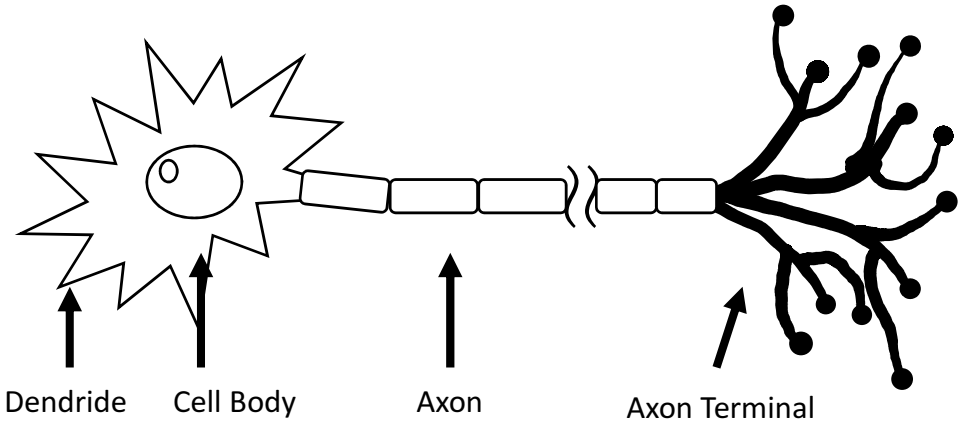


Figure 3.1　Neuron.

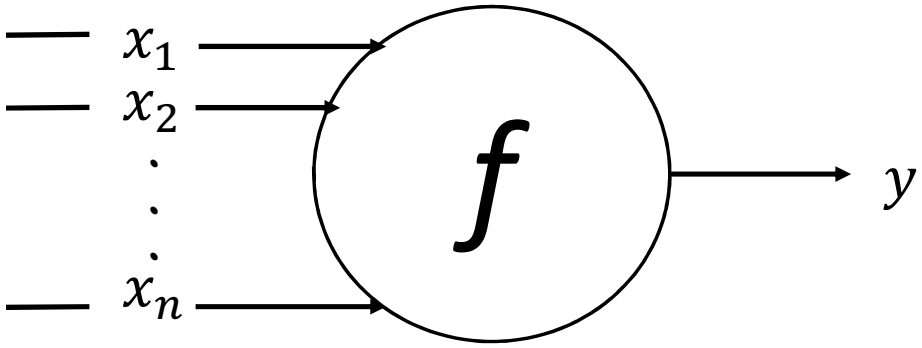


Figure 3.2　Artificial neuron.

3.1.2　モデル化

ニューロンの機能をモデル化し、コンピュータ上で表現することができる。生物学的なニューロンと同様に、プログラム上で表現された人工的ニューロンも複数個の入力 $x_1, x_2, ..., x_n$ を受け取る。それぞれの入力には重み $w_1, w_2, ..., w_n$ が設定されており、入力値との乗算が行われる。これらの重みづけされた値は、生物のニューロンの場合同様に合計される。合計値はロジットと呼ばれ以下のように表現される。

$$\sum_{i=0}^{n} w_i x_i \tag{3.1}$$

Figure 3.2 は、モデル化されたニューロンの概略図である。Figure 3.2 では省略しているが、ロジットにはバイアスという定数も加算される。そしてロジットの値が関数 f に渡され、出力値 $y = f(z)$ が算出される。そして最後に、この出力値が他のニューロンの送信される。ニューロンの機能、またニューラルネットワークの機能は行列として表現される。前者の場合、入力されるデータは、$\boldsymbol{x} = [x_1 x_2 ... x_n]$ というベクトルであるとみなし、ニューロンの重みは $\boldsymbol{w} = [w_1 w_2 ... w_n]$ というベクトルで表現される。これらによりニューロンの出力値を計算する場合は $y = f(\boldsymbol{x} \cdot \boldsymbol{w} + b)$ となる。ここで b はバイアスを表す。

3.2　ニューロンのネットワーク化

ニューロンが 1 つであっても従来の手法 (線形パーセプトロン等) よりも十分強力であるが、手書きの数字を認識することなどのコンピュータにとって複雑な問題を解決するほどの能力はない。こうした複雑な作業をするには、より高度な機械学習のモデルが必要となる。脳のニューロンは階層的に構成されている。人間の知性の大部分をつかさどる

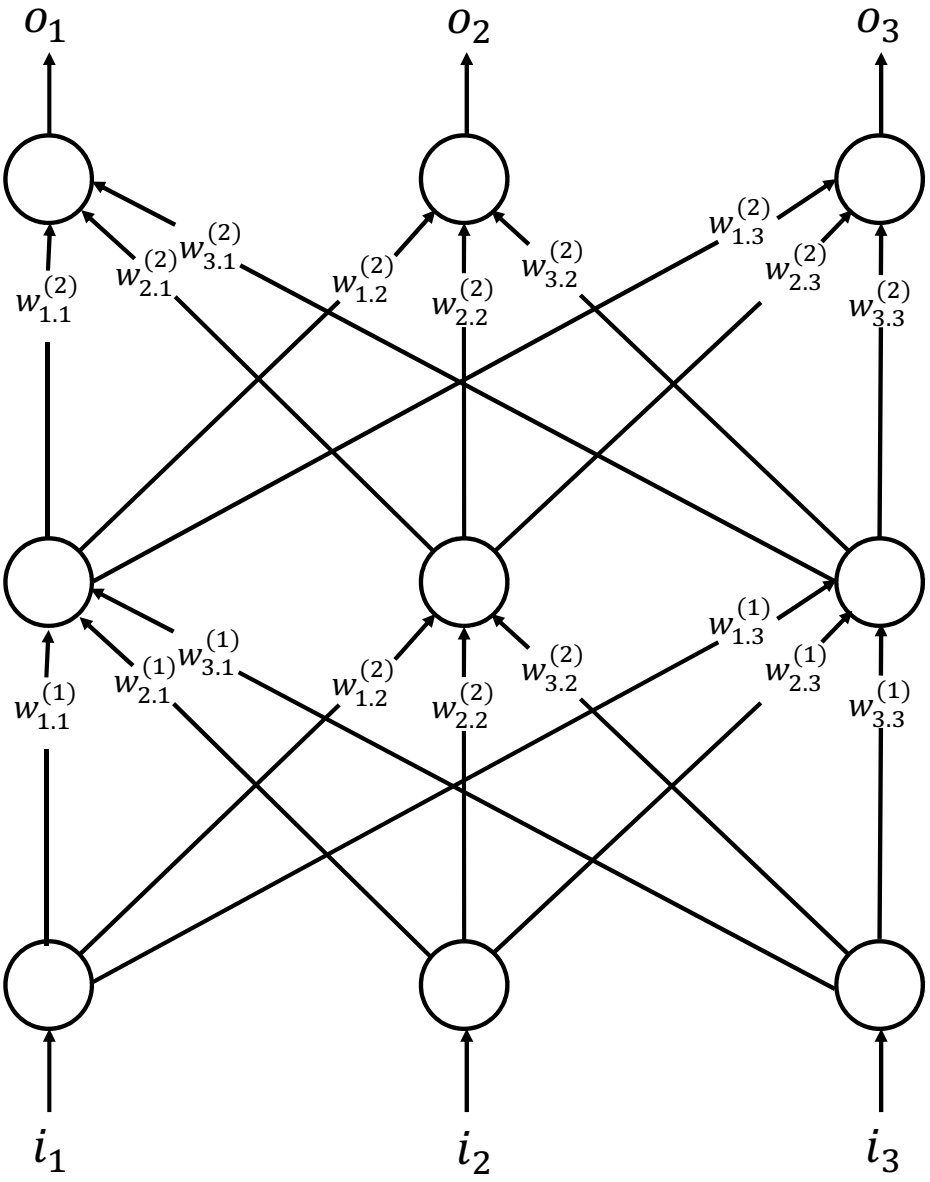


Figure 3.3　Model of artificial neural network.

大脳皮質は、6 層のニューロンからなることがわかっている。層から層へと情報が伝わっていくうちに、感覚器官からの入力が概念的な理解へと変換される。例えば脳の視覚やにあるニューロンの最下層は、目から視覚情報をそのまま受け取る。この情報がそれぞれの層で処理され、上層へと受け流されていく。最終的には第 6 層で、自分が見ているものが何かを判断する。 Figure 3.3 は層が 3 つの単純なネットワークである。このようにニューロンを複数の層状に組み合わせたものを人工ニューラルネットワーク (artificial neural network) という。入力が複数のニューロンを介して、最終的には出力ノードへとたどり着く。出力データは、学習の問題に対するネットワークからの回答に相当する。このニューラルネットワークという考え方は、1943 年に *McCulloch* と *Pitt* によって初めて

発表された。Figure 3.3 では、2 層目は計算過程であり、隠れていることから隠れ層と呼ばれる。隠れ層は複数存在しても構わず、隠れ層が 2 つ以上になると Deep learning と呼ばれる (Figure 2.1)。この隠れ層では問題解決のための重要な処理のほどんどを受け持っている。2.2 節の手書き文字認識の例にもあったように以前は意味のある特徴を発見するために多くの手間と時間を費やす必要があったが、隠れ層によってこの作業が自動化された。また、第 k 層の i 番目のニューロンから第 $k+1$ 層の j 番目のニューロンへの接続に対する重みは、$w_{i,j}^{(k)}$ として表現される。これらの重みの値は、パラメータベクトルを構成する。このパラメータベクトルの最適な値を見つけられるかがニューラルネットワークの問題解決能力を左右させる。

Figure 3.3 では、下層から上層へ一方校の接続のみで構成されており、このような構成はフィードフォワードニューラルネットワークと呼ばれる。各層のニューロンの数に制約はなく、必ずしも次の層のすべてのニューロンに接続をする必要はない。

以上を数学的に表現すると、i 番目の層への入力は、$\boldsymbol{x} = [x_1, x_2, ..., x_n]$ というベクトルとして表現できる。そして、ニューロンを通して入力データを伝搬させた結果も $\boldsymbol{y} = [y_1, y_2, ..., y_n]$ というベクトルであるとする。重みを表す大きさ $n \times m$ の行列 $\boldsymbol{W}$ と、バイアスを表す大きさ m のベクトルがあるとして、伝搬をシンプルな行列の乗算として以下のように表現できる。

$$\boldsymbol{y} = f(\boldsymbol{W}^T \boldsymbol{x} + \boldsymbol{b}) \tag{3.2}$$

3.3 回帰と分類

ニューラルネットワークで扱う問題では、回帰問題と分類問題に分けることができる。

3.3.1 回帰

回帰問題では、データの傾向から連続的な数値を予測する。複数のデータから傾向を見つけ出し、近似直線や曲線を用い、それらをもとに予測する方法が一般的に用いられている。

3.3.2　分類

分類問題では、データを決められた複数のラベルに分類する。分類結果は離散的な値となる。画像をニューラルネットワークの入力にする場合は、画像の各ピクセルを入力とする。

第4章　Train[1),2)]

ディープラーニングでは、訓練 (train) というプロセスを経て、ニューラルネットワークでのすべての接続に対する重み (パラメータベクトル) が決定されていく。訓練では、ニューラルネットワークに多数の例となるデータを与え、誤りが最小化されるように重みを徐々に調整する。この時使用するのがバックプロパゲーション (back propagation) と呼ばれるニューラルネットワークの重みの伝搬方法である。また、訓練に使用するデータを適切に用意することでニューラルネットワークの精度が大幅に上昇するという研究が多数発表されている。

4.1　学習則

ニューラルネットワークにおいて、訓練時にどのように結合強度が変更されるかを記述した法則を学習則という。代表的な学習則に、ヘッブ則とデルタ則がある。

4.1.1　ヘッブ則

ヘッブ則 (Hebbian rule) は、1949 年に *Donald Hebb* によって提唱された脳のシナプスの変化した状態が保たれる性質 (可塑性) についての法則である。神経同士の接合部をシナプス (synapse) と呼ぶ。ヘッブ則では、シナプス前の神経細胞が興奮し、シナプス後の神経細胞が興奮するとそのシナプスの伝達効率が増強される。また、興奮が長時間起こらないとそこのシナプスの伝達効率は減衰する。ヘッブ則で示されるシナプスの可塑性は、長期記憶に関係していると考えられている。

ニューラルネットワークにおける、ヘッブ則による伝達効率の増強は、結合強度 (重み) の変化量を Δw、シナプス前のニューロンの興奮度合い (出力) を y_i、シナプス後のニューロンの興奮の度合い y_j として以下の式で表すことができる。(γ は定数)

$$\Delta w = \gamma y_i y_j \tag{4.1}$$

y_i、y_j がともに大きければ、結合強度が大きく増強される。シナプス前後のニューロンの興奮が繰り返し起きることで、次第にシナプスでは情報が効率的に伝達されるようになる。

4.1.2 デルタ則

デルタ則 (Delta rule) は、1960 年 *Widrow* と *Hoff* らによって提唱されたニューラルネットワークの学習に関する規則である。デルタ則には、

・出力と正解の差が大きいほど、重みの修正量を大きくする。

・入力が大きいほど、重みの修正量を大きくする。という二つの規則で構成されている。この場合の正解とは、ニューロンの出力のあるべき値のことを指している。

デルタ則は、4.1.1 項同様の変数と、正解の値 t を用いて以下の式で示すことができる。(η は学習係数と呼ばれる定数)

$$\Delta w = \eta(y_j - t)y_i \tag{4.2}$$

デルタ則により、理想の状態とはなれているほど、理想の状態に戻るための重みの修正量が大きくなる。また、ニューロンの大きな入力があれば、シナプスに強い刺激があったとみなされ重みは変化しやすくなる。

4.2 損失関数

出力と正解の誤差を定義する関数は損失関数 (loss function) と呼ばれる。誤差はあるべき状態との隔離度合いである。誤差の値が大きければ、ニューラルネットワークが望ましい状態から離れているということである。

損失関数には様々な種類があるが、ディープラーニングでは一般的に、二乗和誤差と交差エントロピー誤差が用いられる。

4.2.1 二乗和誤差

出力値と正解値の差を二乗し、全ての出力層のニューロンで総和をとったものを二乗和誤差と呼ぶ。二乗和誤差は、E を誤差、y_k を出力層の各出力値、t_k をを正解値として、以下の式で表すことができる。

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2 \tag{4.3}$$

y_k と t_k の差を二乗し、全ての出力層のニューロンで総和をとる。$\frac{1}{2}$ をかけるのは微分値を扱いやすくするためである。

4.2.2　交差エントロピー誤差

2 つの分布間のズレを表すための損失関数を交差エントロピー誤差と呼ぶ。交差エントロピー誤差は、出力 y_k の自然対数と正解値の積の総和をマイナスしたもので、以下の式で表すことができる。

$$E = -\sum_k t_k log(y_k) \tag{4.4}$$

分類問題において正解値は、1 つが 1 で、残りが 0 のベクトル (one-hot 表現) で表すことができる。したがって、式 (4.4) 右辺のシグマ内で t_k が 0 以外の項のみ誤差に影響を与えることとなり、t_k が 0 の項の影響は無視される。この結果正解値が 1 のたった 1 つの項しか誤差に影響を与えない。また、分類数を N として、式 (4.4) を N で割ったものを categirical cross entropy と呼ぶ。 Figure 4.1 は、$y = -\log(x)$ の軌跡である。見てわか

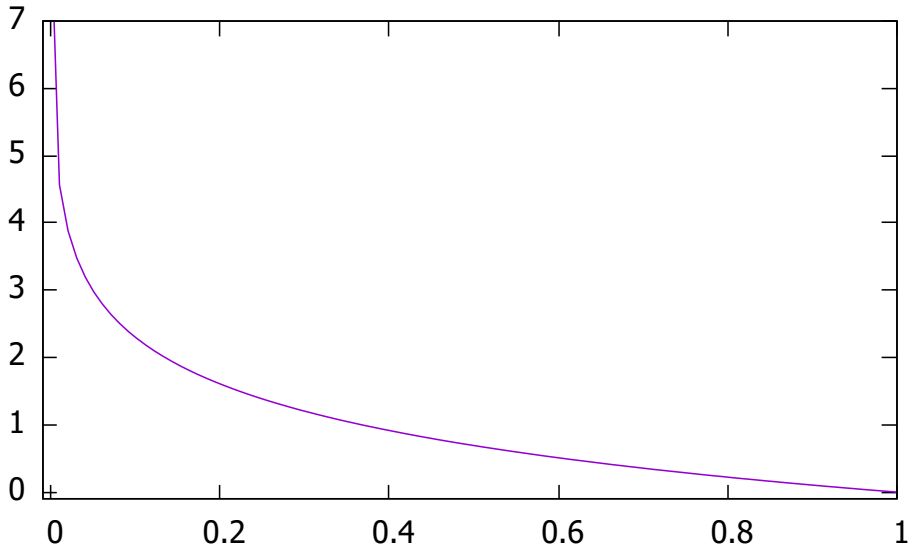


Figure 4.1　$y = -\log(x)$.

る通り、x の値が 1 に近づくほど y の値は 0 に近づき、x の値が 0 に近づくほど y の値は無限大に近づく。また、x の値が負の値をとると y は値をとらない。この性質を利用し、交差エントロピー誤差は、正解に近づくほど小さくなっていく。また、対数を使用することで、交差エントロピーは出力値と正解値の隔離が大きいとき学習速度が速いという利点がある。

交差エントロピー誤差では、対数関数の中身が 0 となると、自然対数が無限小に発散してしまい計算を続けることができなくなる。計算時は、y に微小な値を加えることでそれを防いでいる。

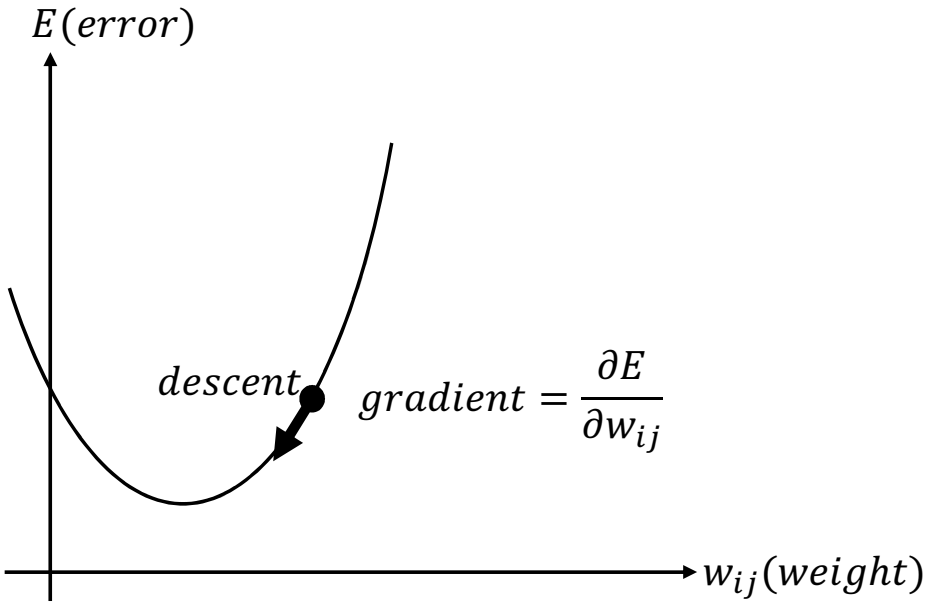


Figure 4.2　Gradient dscent.

4.3　勾配降下法

訓練時は、誤差を次々と前の層へ伝搬させ、重みとバイアスを徐々に更新して最適化する。そのためには、勾配降下法というアルゴリズムを使用する。あるパラメータ x_k の変化量に対する関数 $y(x_1, x_2, ..., x_k, ...)$ の変化量の割合、すなわち勾配 $\frac{\partial y}{\partial x_k}$ を求めて、この勾配に基づいてパラメータを調整し、y を最適化するアルゴリズムを勾配法と呼ぶ。

本章の勾配降下法 (gradient descent) は勾配法の一種で、結果が y の最小値に向かって降下するようにパラメータ y_k を変化させる。バックプロパゲーションにおいては損失関数により求めた誤差の値を起点に、ニューラルネットワークを遡って重みとバイアスの修正を行うが、この際に勾配降下法を用いて修正量を決定する。勾配降下法では、誤差が小さくなるように、ニューラルネットワークの重みとバイアスを調整する。

Figure 4.2 では、横軸の w_{ij} がある重み、縦軸の E が誤差である。重みの値に応じて誤差は変化するが、実際はこのような曲線の形状を知ることはできないため、足元の曲線の傾き (勾配) に応じて少しずつ重みを更新する。ネットワークのすべての重みをこの曲線を降下するように変化させると、誤差を次第に小さくしていくことができる。この際の各重みの変化量は、勾配で決定する。バイアスの場合も同様となる。従って、ニューラルネットワークのすべての重みとバイアスを更新する為にすべての重みとバイアスに対する誤差の勾配を求める必要がある。

なお、w_{ij} の変化に対する E の変化は、必ずしも Figure 4.2 のような軌跡となるとは

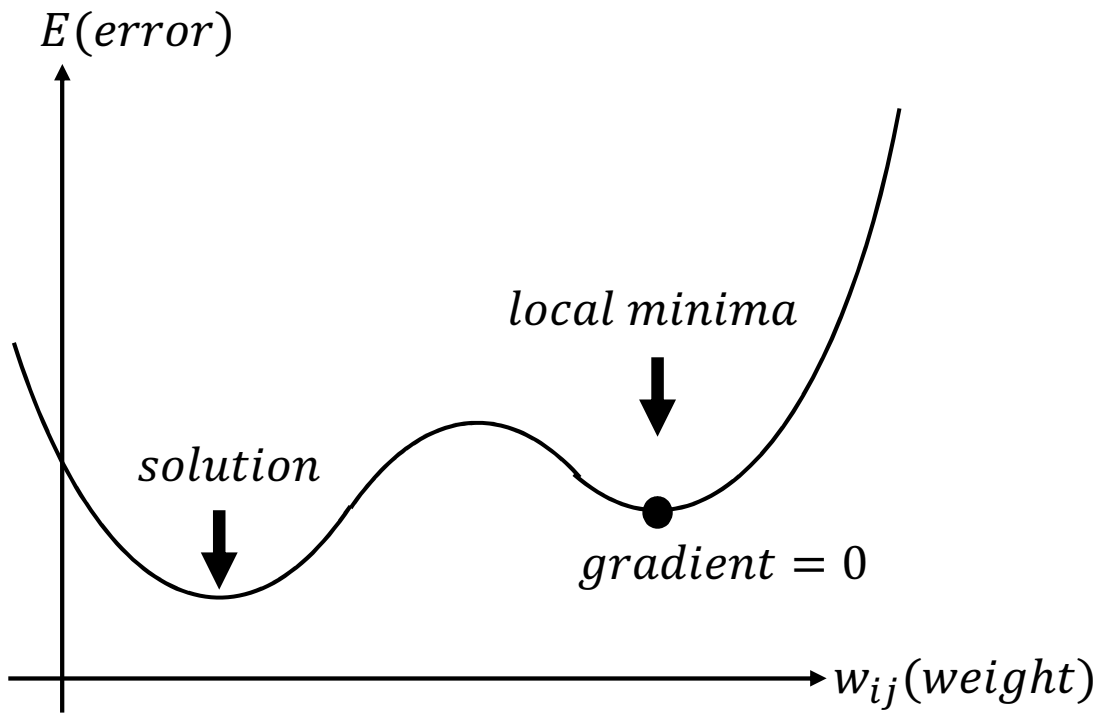


Figure 4.3　Local minima.

限らず Figure 4.3 のような複雑な軌跡となり、局所解と呼ばれる勾配はないが最適解ではない値に収束してしまう可能性がある。

勾配降下法による重みとバイアスの更新は、w を重み、b をバイアス、E を誤差として、偏微分を用いた次式で表すことができる。尚上式は実際に実装する際は、変形することがある。

$$w \leftarrow w - \eta \frac{\partial E}{\partial w} \tag{4.5}$$

$$b \leftarrow b - \eta \frac{\partial E}{\partial b} \tag{4.6}$$

上式で、矢印は重みの更新を示している。η は学習係数と呼ばれる定数で、$\frac{\partial E}{\partial w}$ と $\frac{\partial E}{\partial b}$ が勾配となる。

学習係数は、学習の速度を決定する定数である。0.1 や 0.01 などの小さな値が使われることが多いが、小さすぎると学習速度が低下し、局所解に収束してしまう。学習係数が大きすぎると誤差が収束しにくいという問題が発生する為、効率よく最適解へと導くには、学習係数の値を適切に設定する必要がある。

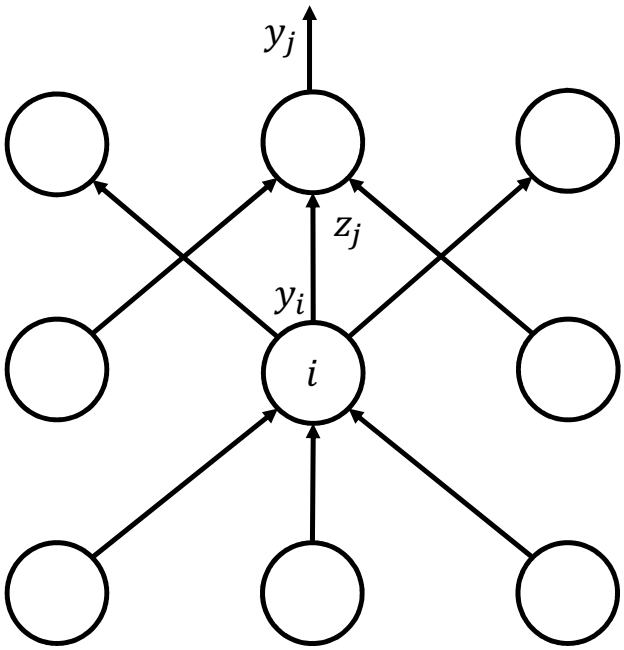


Figure 4.4　Model for indicationg back propagation.

4.4　バックプロパゲーション

ニューラルネットワークの重みの更新には、1986 年 *David E. Rumelhart, Geffrey E. Hinton, Ronald J. Williams* の 3 名によって提唱されたバックプロパゲーション (back propagation) によって決定する。バックプロパゲーションは、「隠れている処理の内容はわからないが、処理を変化させるとどの程度の誤差が変化するかは計算可能である」という考え方が背景にある。例として、訓練データが 1 つだけの場合の損失関数の導関数の計算を考える。

隠れている処理のそれぞれが、多くの出力先に影響を与えている。つまり、誤差に対するそれぞれの影響をわかりやすくまとめる必要がある。隠れ層での損失関数の導関数がわかればこの値を使って 1 つ下の層で行われる処理について損失関数の導関数を求められる。かくして、全ての隠れ層で損失関数の導関数がわかれば、隠れた処理への入力の重みに対する隠れ層全体として損失関数の導関数も容易に求められる。以降、説明を簡易化する為、Figure 4.4 のような記法を定める。 下付き添え字はニューロンの階層を示している。y はニューロンからの出力、z はニューロンへのロジットである。基礎的な部分から考えると、出力層での損失関数は以下の通りである。

$$E = \frac{1}{2}\sum_j (t_j - y_j)^2 \Longrightarrow \frac{\partial E}{\partial y_j} = -(t_j - y_j) \tag{4.7}$$

第 j 層での損失関数の導関数が既にわかっていると仮定し、第 j 層のそれぞれのニューロンでのロジットに及ぼす影響を以下の式で表す。

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j} \frac{\mathrm{d}z_j}{\mathrm{d}y_i} = \sum_j w_{ij} \frac{\partial E}{\partial z_j} \tag{4.8}$$

また、次式も成り立つ。

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\mathrm{d}y_j}{\mathrm{d}z_j} = y_j(1-y_j) \frac{\partial E}{\partial y_j} \tag{4.9}$$

これらを組み合わせることにより、第 j 層での損失関数の導関数を使用し、第 i 層での損失関数の導関数を表現できる。

$$\frac{\partial E}{\partial y_i} = \sum_j w_i j y_j (1-y_j) \frac{\partial E}{\partial y_j} \tag{4.10}$$

よって重みに対する誤差の変化量は、

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}} \frac{\partial E}{\partial z_j} = y_j y_i (1-y_j) \frac{\partial E}{\partial y_j} \tag{4.11}$$

と表すことができる。これがここ k の訓練データを与えた後に変化させるべき重みの量となる。

最後に、訓練データに対する偏微分の値を合計する。重みの変化は次式で表現される。(ε は係数)

$$\Delta w_{ij} = -\sum_{k|indataset} \varepsilon y_i^{(k)} y_j^{(k)} (1-y_j^{(k)}) \frac{\partial E^{(k)}}{\partial y_j^{(k)}} \tag{4.12}$$

第5章　Convolution neural network[1),2)]

画像認識に焦点を置く。第3章のような単純なニューラルネットワークは画像認識にはあまり向いていない。なぜなら、この方法では、画像のピクセルごとの情報を各ノードに渡し、計算する必要がある。しかしこれでは、意味のある情報とノイズの比 (SN 比) の値が低く、確実な学習ができない。また、認識画像の画素値が上がるほど扱う情報の量が増え、それに伴い大量のデータを扱う必要が出てくる。また、学習データのみに頑強なネットワークになる過学習という現象が起こり易くなってしまう。

画像認識の分野では、世界的に有名な *ImageNet* と呼ばれるコンテストがある。ここでは、450000 枚の訓練データを使用し、200 のクラス分けの精度を競っている。2011 年の勝者の誤答率は、27.5% であったが、2012 年 *Alex Krizhevsky* が適用した畳み込みニューラルネットワーク (convolution neural network) では、誤答率が 16% と他に大差で勝利を収めた。畳み込みニューラルネットワークは CNN と略されることが多い。

一番単純な畳み込みニューラルネットワークの概略図を Figure 5.1 に示す。四角で囲われた部分は各層を示し、畳み込み (convolution) 層では、出力が入力の一部しか影響を受けない局所性の強い処理が行われ、プーリング (pooling) 層では、認識する対象の位置に柔軟になる処理が行われる。最後に全結合 (fully conected) 層につながれ結果出力する。また、畳み込み層とプーリング層は複数回繰り返しても良く、精度の高い CNN はこれを行っている。以下で各層を詳しく説明する。数学的には $m \times n$ の行列で処理がされていく。

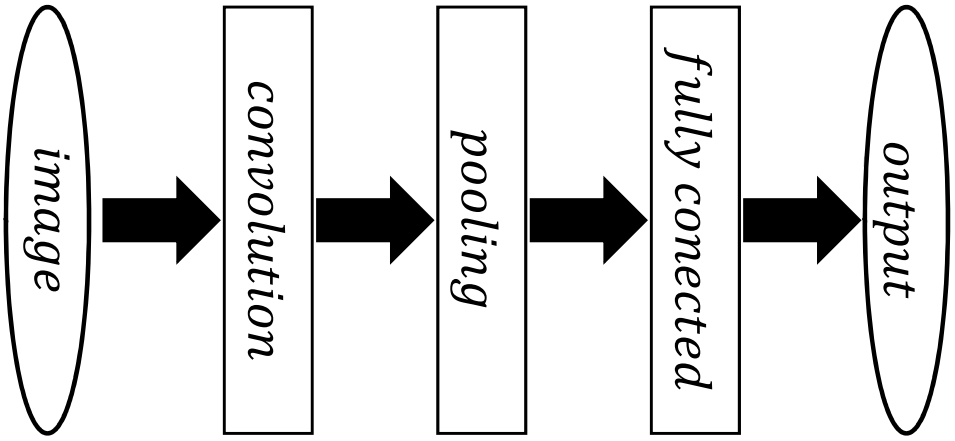


Figure 5.1　Mdoel of convolution neural network.

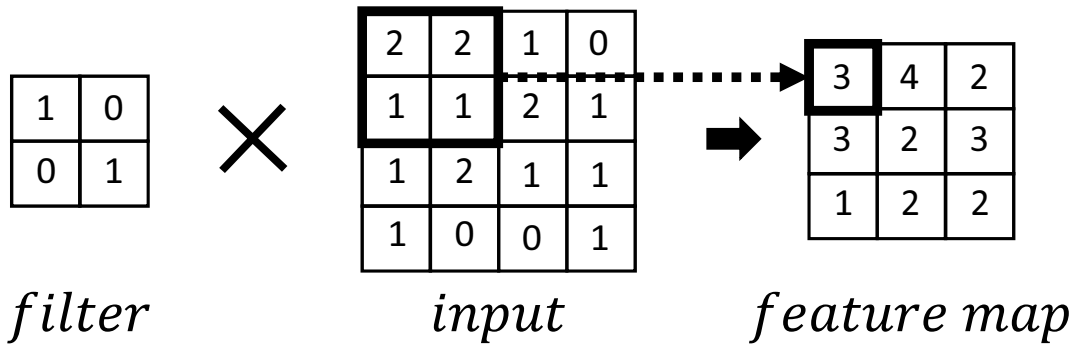


Figure 5.2　Convolution.

5.1　畳み込み層

入力画像では、各ピクセルが隣同士のピクセルの値と強い関係性のある性質 (局所性) を持っている。この局所性を使用し、特徴を抽出するのが畳み込み層である。特徴抽出には複数のフィルタという正方行列を使用し、入力が画像と各フィルタ (filter) を畳み込み演算することによって得られる。 Figure 5.2 では、入力画像サイズが 4×4、フィルタサイズが 2×2 で畳み込み (convolution) をした例である。入力行列の左上とフィルタの重なる各要素を乗算し、その総和を一つの要素とする。フィルタを右にずらしてながら同じ計算を行い、最も右まで行きついたら下の段へ移動し同じ処理を行う。やがてフィルタの移動ができなくなり、計算結果の行列を特徴マップ (feature map) という。フィルタの値によって特徴マップに現れる特徴が変わり、Figure 5.2 では右斜め下にかかる線の特徴を抽出している。通常、CNN では複数のフィルタを使用する。入力の行列の数 n、フィルタの数 m とすると次の層へは $n \times m$ の数の行列が渡される。

5.2　プーリング層

プーリング層は通常畳み込み層の直後に配置される。この層では、入力を各領域に区切りその領域の代表する値を抽出し並べたものである。この処理により、元画像の位置がぼかされる。さらに扱う情報量が削減され、計算も容易になる利点がある。 プーリングには、Figure 5.3 のように領域の最大値をとるマックスプーリング (max pooling) と領域の平均をとるアベレージプーリング (average pooling) がある。

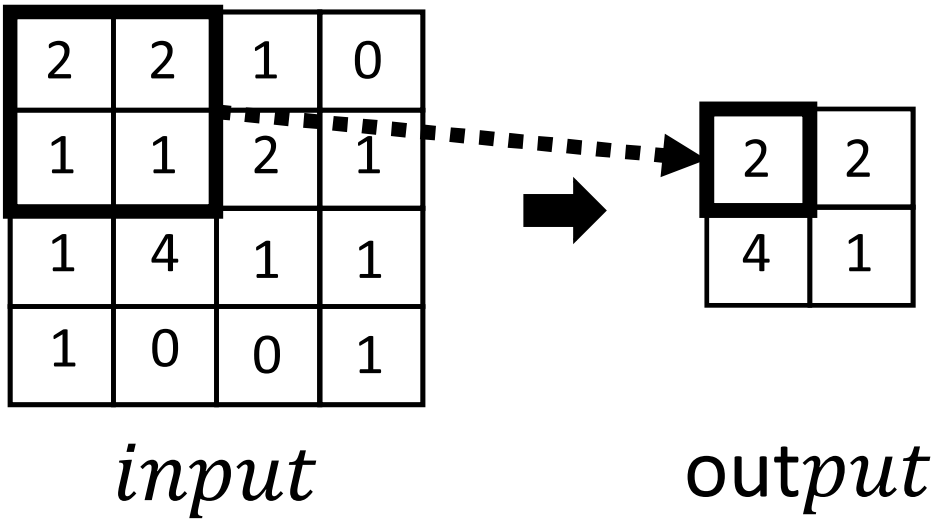


Figure 5.3　Max pooling.

5.3　全結合層

全結合層は、通常のニューラルネットワークで用いられる層のことである。全結合層は、通常畳み込み層とプーリング層を何度か繰り返した後に配置される。畳み込み層とプーリング層により抽出された特徴量に基づき、演算を行い、結果を出力する。

全結合層同士の接続では、通常のニューラルネットワークと同様にニューロンは隣り合う層のすべてのニューロンと接続される。畳み込み層やプーリング層の出力を全結合層に入力する場合は、画像を平坦なベクトルに変換する必要がある。仮に、出力の画像の高さがH、幅がW、チャンネル数がFである場合、全結合層の入力はサイズが$H \times W \times F$のベクトルになる。

5.4　パディング

畳み込み層やプーリング層において、入力行列を取り囲むように要素を配置することをパディング (padding) という。 Figure 5.4 では、入力行列の周囲に 0 でパディングしており、これをゼロパディング (zero padding) という。他にもパディングは存在するが、CNN では専らゼロパディングが使用されることが多い。

畳み込み層やプーリング層が繰り返されることで計算する行列サイズが小さくなってしまい、やがては1×1になってしまう。パディングには畳み込みを繰り返しても画像サイズが変わらないという利点がある。また、畳み込みの特性上、入力行列の端では、畳み込みの回数が少なくなってしまうが、パディングにより画像の端のデータに対する畳み込みの回数が増えるため、端の特徴も取り入れられるという利点もある。

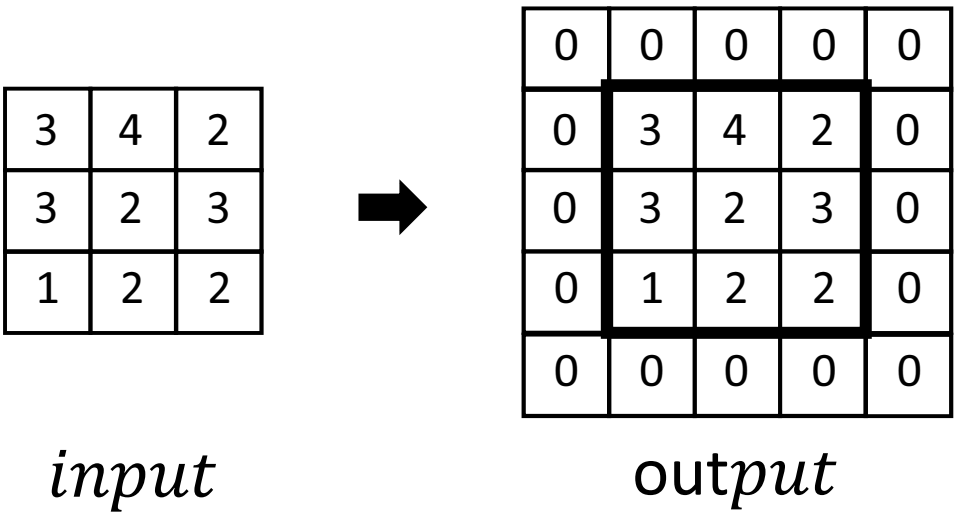


Figure 5.4　Zero padding.

5.5　ストライド

ストライド (Stride) は畳み込みする際にフィルタが移動する間隔のことである。Figure 5.2 では、フィルタの移動間隔は 1 で行われているためストライドは 1 となる。ストライドが大きいと、フィルタの移動距離が大きくなるため、生成される画像のサイズが小さくなる。大きすぎる画像を縮小する為にストライドの値を 1 より大きくすることがある。しかし、ストライドの値を大きくするほど特徴を見逃してしまう可能性がある為、通常、ストライドは 1 に設定する。入力行列サイズを $I_h \times I_w$、フィルタサイズを $F_h \times F_w$、パディングの幅を D、ストライドの値を S とすると、出力画像の高さ O_h、幅 O_w は次式となる。

$$O_h = \frac{I_h - F_h + 2D}{S} + 1 \tag{5.1}$$

$$O_w = \frac{I_w - F_w + 2D}{S} + 1 \tag{5.2}$$

5.6　ReLU

計算結果が線形的ではないニューロンとして、広く使われているものは大きく 3 種類ある。そのうちの一つが ReLU(Rectified Liner Unit) である。Figure 5.5 にその軌跡を示す。ReLU は、$f(z) = \max(0, z)$ という式で表すことができる。ReLU は、$x = 0$ において非連続で微分不可能であるという欠点があるが、微分が容易であることなどから近年機械学習の活性化関数で主流となっている。

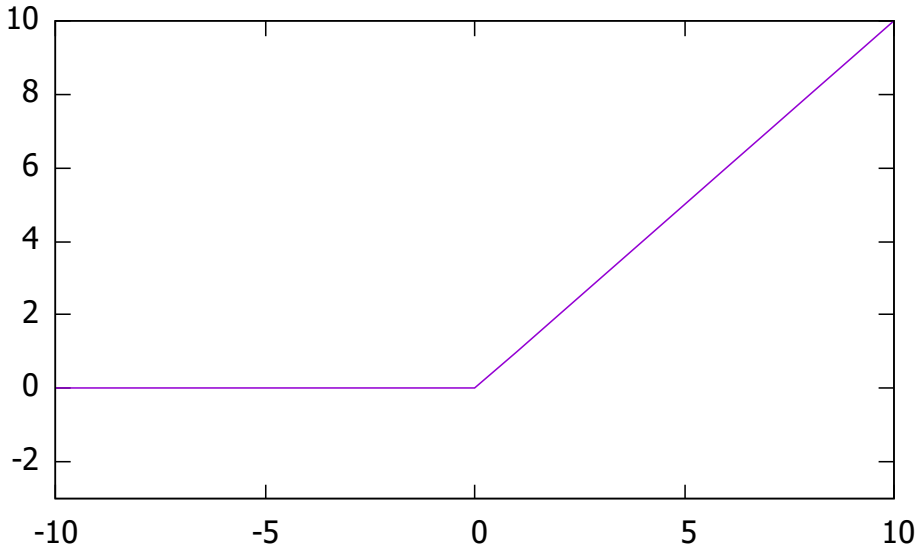


Figure 5.5 ReLU.

第6章　Object detection[1),3)]

6.1　デフォルトボックス

画像認識を使用し、物体の分類に加え、物体の位置を予測する手法を物体検出 (object detection) と呼ぶ。画像から物体のクラスを予測する代わりに、クラスとその物体を含むデフォルトボックス (defolt box) を予測する必要がある (Figure 6.1)。デフォルトボックスを識別するには、デフォルトボックスのクラス、座標 (x, y)、幅、高さの 4 項目が必要である。物体検出では、入力画像から異なるサイズ、種類のデフォルトボックスの位置を取得し、これらの領域を画像分類器によって分類するという処理の流れとなる。

物体は画像中に様々なサイズで存在するため、物体認識用データを複数サイズに変更した画像を用意し学習する。一般的には、特定の条件 (最小サイズに達するなど) が得られるまで、画像はダウンサンプリングされ、それぞれについて固定サイズのボックス検出が実行される。

6.2　評価指標

物体検出を評価する指標として物体検出精度と処理速度がある。

物体検出精度とは、画像中の検出したい物体を、作成したモデルがどの程度正しく検出できるかを評価する指標である。

処理速度はその名の通り、物体検知を処理する速度であり、一般的には fps(frames per second) を使用する。

6.3　jaccard 係数

ある集合の類似度を測る値として jaccard 係数 (jaccard index) がある。ある集合 A とある集合 B について、jaccard 係数は以下の式で定義される。

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{6.1}$$

物体検出では、jaccard 係数を物体の位置を決定する際に用いられることがある。また物体検出では、jaccard 係数を用い、最も高い予測値のバウンディングボックスに重なる他のデフォルトボックスにおいて、二つのボックスの jaccard 係数が一定値以上の場合、後者を候補から除外する操作を non-maximum suppression と呼ぶ。この処理を行うことにより、同じ物体を同じラベルで複数検出することを防ぐとこができる。

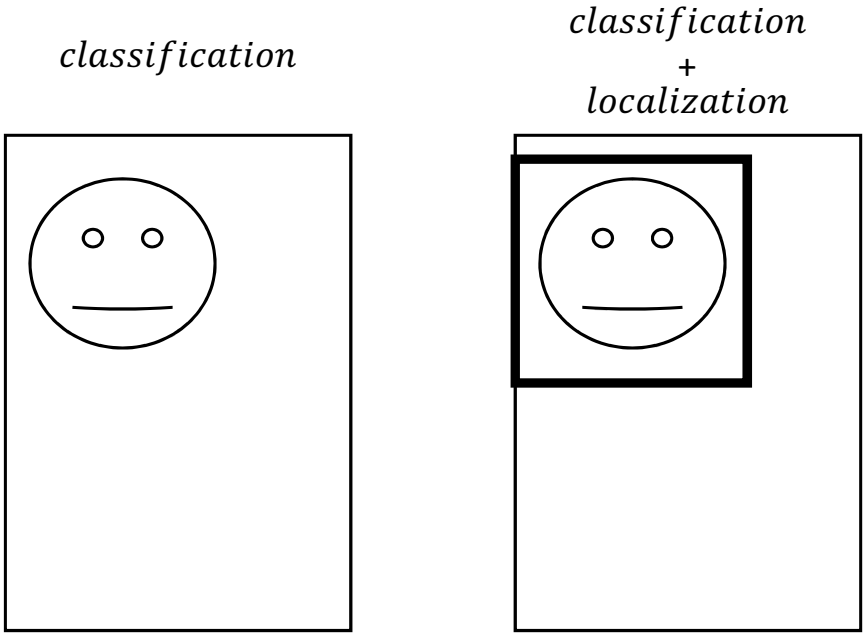


Figure 6.1　Object detection.

6.4　ハードネガティブマイニング

ほとんどの場合、画像中には認識した物体よりも背景部分の割合の方が多い。その為、正解ラベルとの対応付けによりほとんどのデフォルトボックスが負の値をとり、これを全て学習させると背景のような負例しか出力しないネットワークでも損失関数の値を下げていってしまう。そこで、確信度誤差が高い順にソートし、負例と正例が最大でも 3 : 1 となるように調整しそれらについて損失関数を計算する事をハードネガティブマイニング (hard negative mining) と呼ぶ。

第7章　SSD[4)]

7.1　モデル

物体検出をするモデルの中に2016年、*Wei Liu*らによって発表されたSSD(single shot detector)と呼ばれるものがある。このモデルでは、それまで発表されてきた物体検出のモデルと比較し、シンプルなネットワーク構成であり、高速で、精度が高いという利点を持っている。実際の内容をFigure 7.1に示す。見てわかる通り、CNNのVGG16というモデルを軸に、畳み込み途中で特徴マップを計算することで、特徴マップを徐々にスケールダウンしながら計算を行う。これにより、幅広い画素の物体に対して検出が可能となる。

7.1.1　出力サイズ

SSDのクラススコア数は、c次元のデフォルトボックスのオフセット値4つ(座標、幅、高さ)とセルにおけるデフォルトボックスの数k、特徴マップのサイズ$m \times n$によって、以下のように示すことが出来る。

$$(c+4)kmn \tag{7.1}$$

7.2　デフォルトボックスのスケールとアスペクト比選択

各特徴マップにおけるデフォルトボックスのスケールは、k番目の特徴マップのスケールs_kと特徴マップの数Nを使用して、以下のように示すことができる。

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{N-1}(k-1) \tag{7.2}$$

$$k \in [1, N] \tag{7.3}$$

ただし、デフォルトボックスの敷き詰め方は完全には決められていない。

7.3　損失関数

SSDの損失関数は、物体の位置ずれ(localization loss)の予測位置l、正解位置g、デフォルトボックスの位置dによって以下の式で表される。

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in cx,cy,w,h} x_{ij}^{p} smooth_{L1}(l_i^m - g_j'^m) \tag{7.4}$$

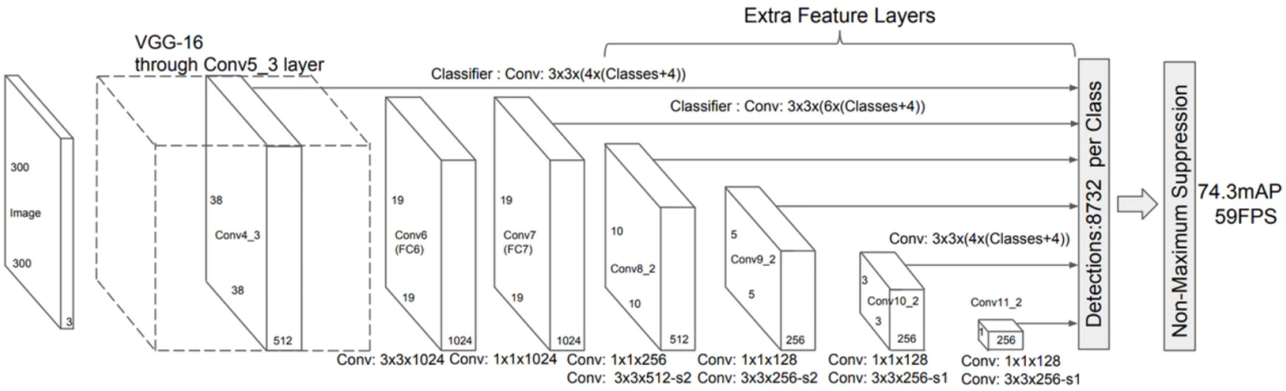


Figure 7.1　Model of SSD.

$$g_j^{\prime m} = \log\left(\frac{g_j^w}{d_i^w}\right) \tag{7.5}$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if(|x| < 1) \\ |x| - 0.5 & (otherwise) \end{cases} \tag{7.6}$$

x_{ij}^p は、0,1 で表される。これは、カテゴリ p での i 番目のデフォルトボックスと j 番目の正解ボックスのマッチングを示している。Pos は正値である。

物体のクラス損失 (confidence loss) は、上の変数に加え、クラス確信度 c(softmax 関数で求める) を用いて、以下の式で表すことができる。

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p \log\left(c_i^{\prime p}\right) - \sum_{i \in Neg} \log\left(c_i^{\prime 0}\right) \tag{7.7}$$

$$c_i^{\prime p} = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)} \tag{7.8}$$

Neg は負値である。

式 (7.4)、式 (7.7) を結合したものに、ハイパラメータ α とマッチしたデフォルトボックスの数 N を使用して、SSD の損失関数は以下の式で示される。

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g)) \tag{7.9}$$

7.4 速度

SSD は、Pascal VOC2007 と呼ばれる公開のデータセットを学習した際にはおよそ 59fps の速度で処理が行われており、処理の 8 割が VGG16 による計算時間に割かれている。速度は計算機の能力や上記のクラス数などに依存するため一概には言えない。

第8章　CNNを用いた柔道競技中の状態認識の実験

8.1　実験目的

現在の画像認識では、MNIST等の、インターネット上にデータセットとして提供されているものを利用すれば高精度な認識結果を得られる。しかし、柔道競技において応用した例は、見つかっていない。そこで独自でデータセットを作成し、画像認識を適用する。これにより、第1章に挙げた1を達成する。

8.2　実験方法

データセットとして、柔道の競技者の各状態をアスペクト比1 : 1のpng形式で準備した。それらを、CNNのVGG16と呼ばれるモデル (Figure 8.1) を利用し、画像認識を行い経過を観察した。データセットは柔道競技において、どちらか一方の選手が技の効果により体が宙に浮き、その後畳に体が接地した瞬間の画像を、「背中が完全に畳に接地している」「背中が半面もしくはそれ以下が畳に接地している」「背中が畳に接地していない」の3つに分類した。各ラベル20枚ずつの色付きデータを用意し、そのうち8割を訓練用、残りをテスト用として使用した。epochは10に設定し、accuracyとlossの変動を観察する。loss関数にはcategorical cross entropyを使用した (4.2.2節)。以上の条件以外の制約は設定しない。各ラベルの例をFigure 8.2に示す。また、同一ラベル内のデータの数枚を例としてFigure 8.2に示す。

8.3　実験結果・考察

実行結果のaccuracyとlossをそれぞれFigure 8.4、Figure 8.5に示す。

accuracyの値の変化は全く見られなかった (Figure 8.4)。また、lossの値は、test時は下がっていったものの、train時は上昇しいる (Figure 8.5)。これらから、学習の精度が期待したほど上昇しなかった。この原因として、データ数の量が不十分であったこと、分類をもっと細かくすべきであったという点が考えられる。VGG16自体はCNNの代表的なモデルとして高精度の画像認識を行うことが周知されているため、データの与え方に問題があったと考えるのが妥当である。期待した結果は得られなかったものの、test時のlossの値が減少していることより、柔道競技の画像でも精度の上昇は期待できると考える。

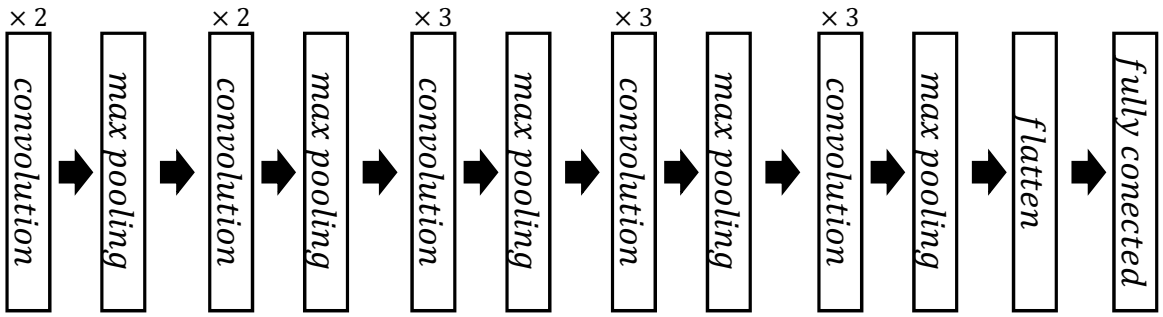


Figure 8.1 Model of VGG16.

Figure 8.2 Each label sample.

Figure 8.3 Images in one label.

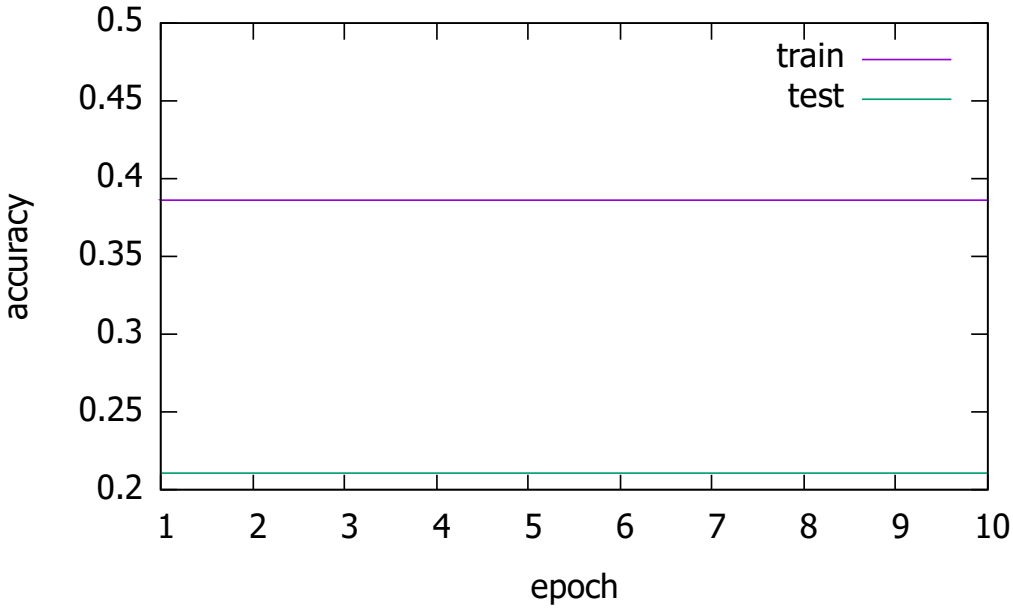


Figure 8.4 VGG16 accuracy.

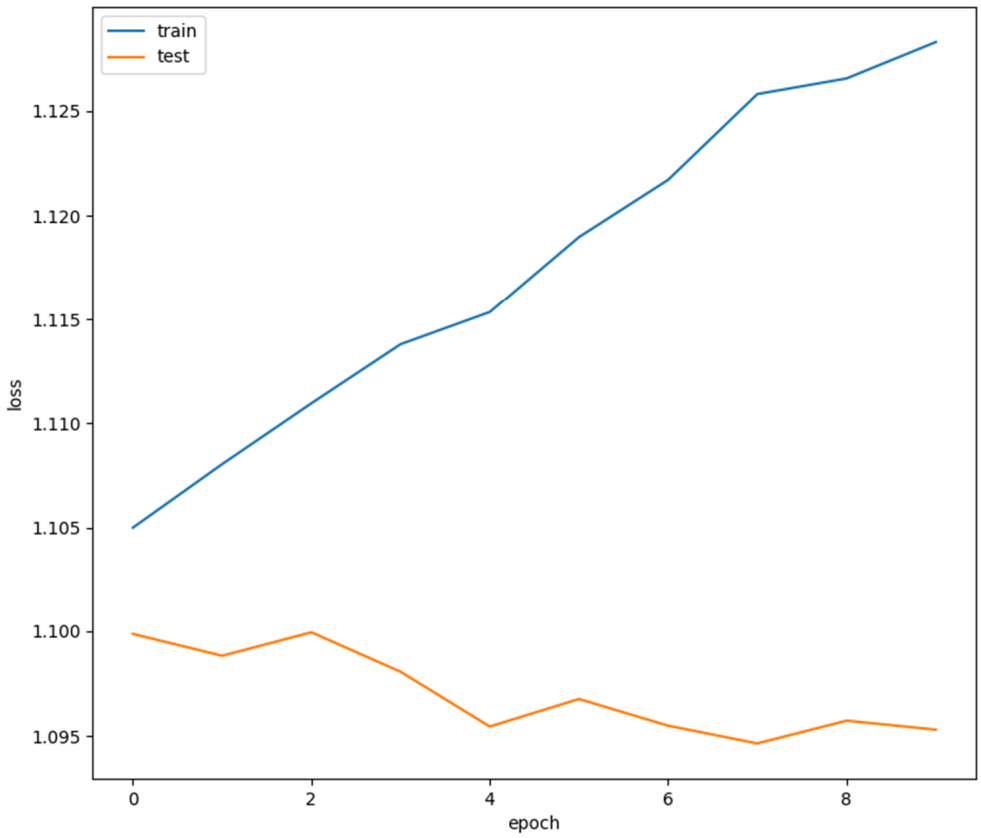


Figure 8.5 VGG16 loss.

第 9 章　SSD による選手の追跡

9.1　実験目的

実際に柔道の試合をカメラで撮影しながらスコアを判定するには、第 8 章の対象物体を画面中から認識する必要がある。本章ではそれを行い、動画への適用を目指す (1 章 2 番)。

9.2　実験内容

物体検知には SSD のモデルを使用した。学習させるデータは 2 つ用意した。1 つ目は、画面上に選手同士が重なっている、選手がそれぞれ画面上に重なっていない、その他の人間の 3 種類のラベル分けをしたデータ。2 つ目は、選手のみをラベル付けしたデータである。後者では、選手同士が画面上で重なっている場合はそれを 1 つの物体とした。

2 つのデータに使用した画像は同一大会で行われた、ほぼ同一の角度から撮影したカラー画像を 384 枚準備した。そのうち 307 枚が学習用、その他がテスト用データである。これらのデータでそれぞれ 150 回学習を繰り返し、accuracy と loss を観察する。

9.3　実験結果・考察

実行結果として、各データの accuracy と loss のグラフを 1 つのラベルで行ったものをそれぞれ Figure 9.2、Figure 9.3 に、3 つのラベルで行ったものをそれぞれ Figure 9.4、Figure 9.5 に示す。　まず、Fiugure 9.2、Fiugure 9.4 を観察する。それぞれ、初めは不安定だった数値が epoch を重ねるごとに安定していることがわかる。また、最終的には 1 つ

$(a)1\ label$　　$(b)3\ label$

Figure 9.1　Example of training data.

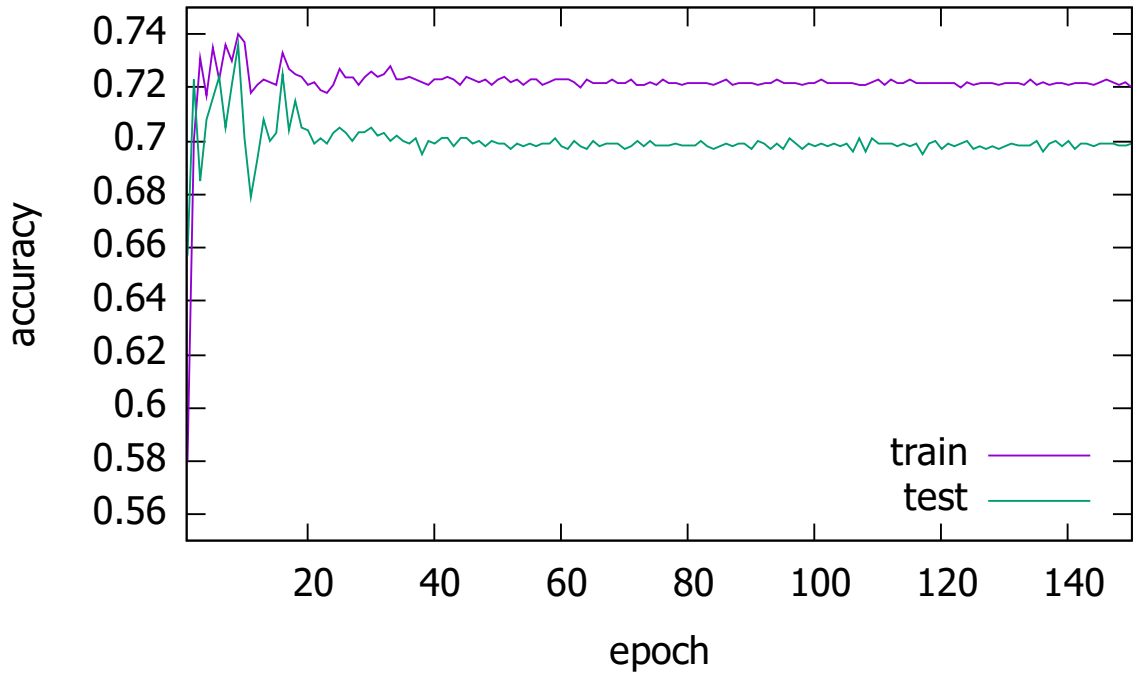


Figure 9.2 Accuracy (single label).

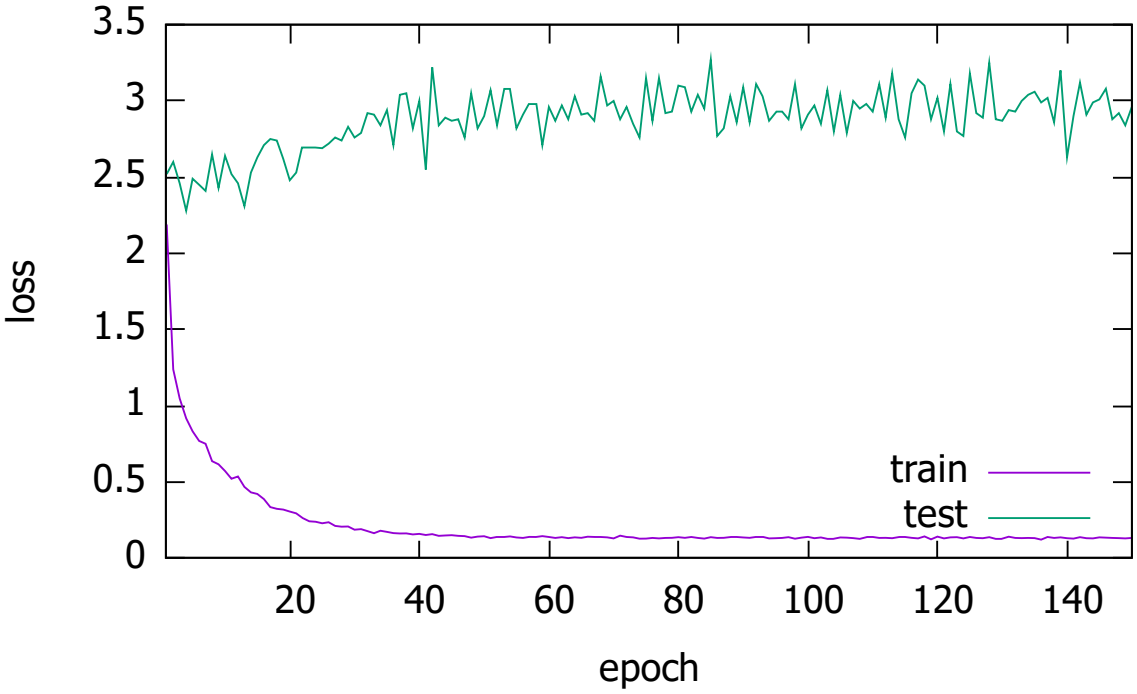


Figure 9.3 Loss (single label).

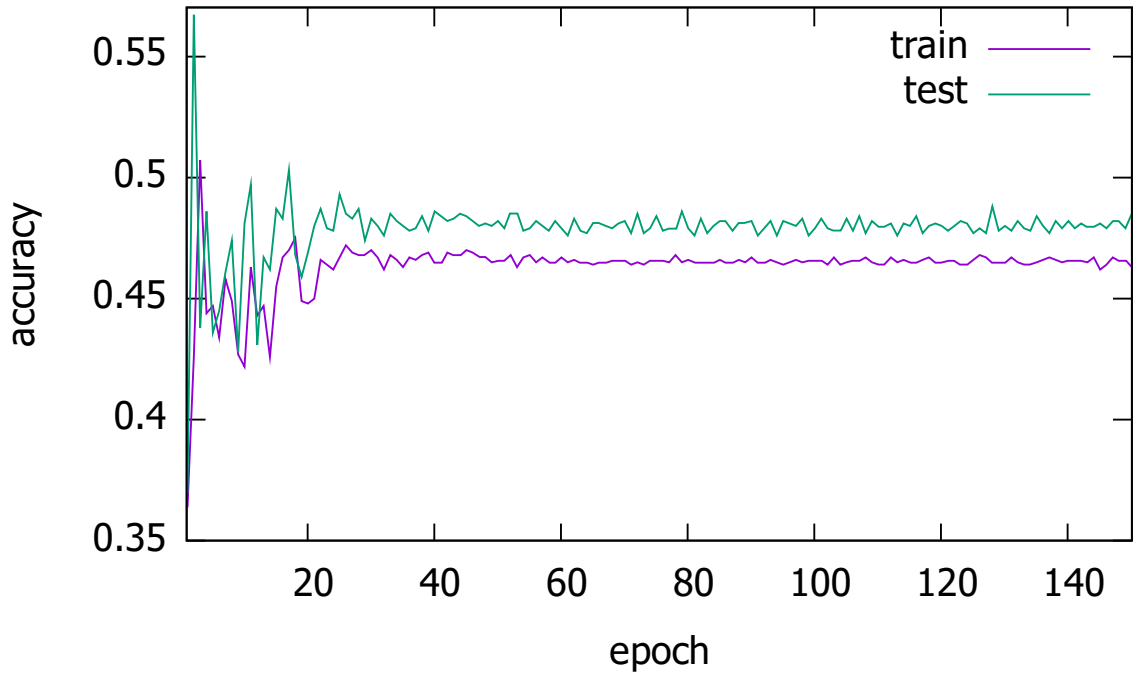


Figure 9.4 Accuracy (3 labels).

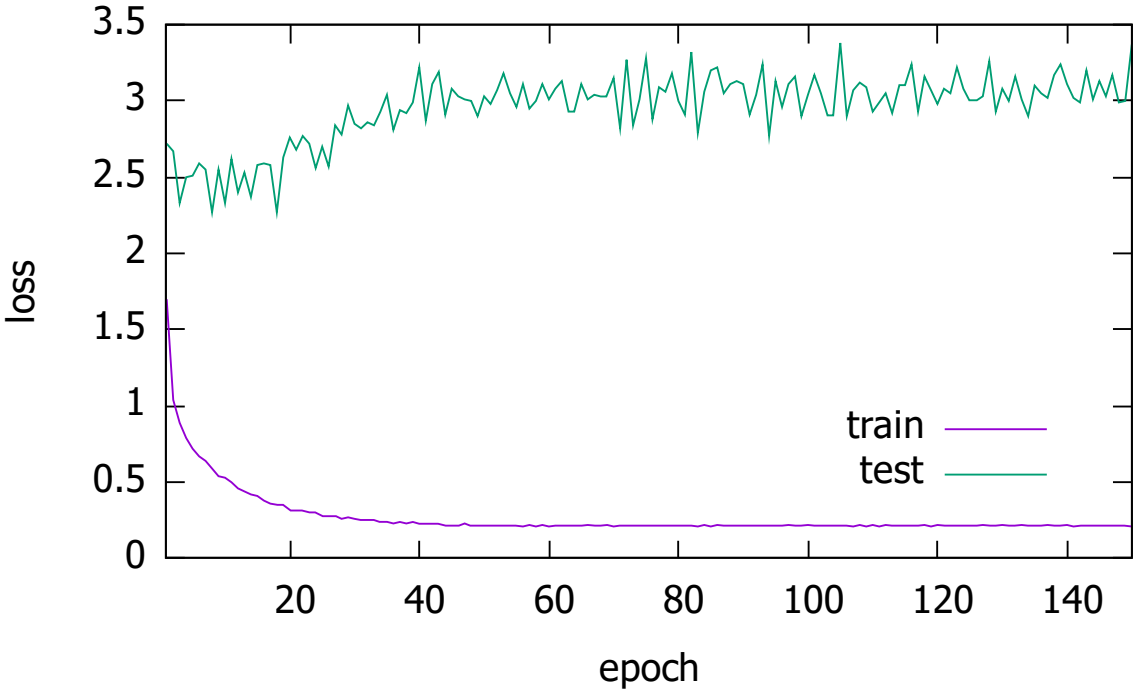
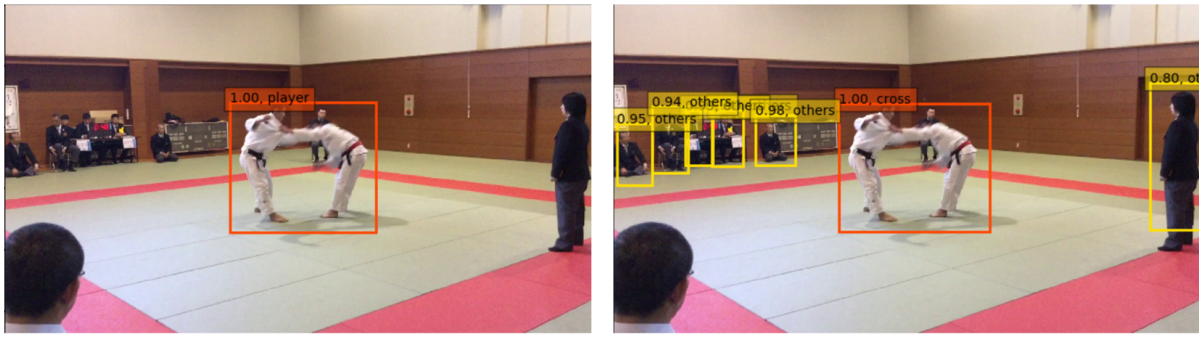


Figure 9.5 Loss (3 labels).



$(a)1\ label$ $(b)3\ label$

Figure 9.6　Ex. good detection.

ラベル付けしたデータの accuracy がおおよそ 0.7、3 つラベル付けしたデータの accuracy がおおよそ 0.47 であることから、前者の方が精度が高いことがわかる。ここで loss の変化も見てみる。Figure 9.3、Figure 9.5 を比較してみるが、ほとんど違いがみられず、訓練時には loss が順調に減少しているが、テスト時は loss が減少していない。それぞれのデータの場合で、loss が大きく減少しているときの epoch では accuracy も不安定な状態から安定な状態に推移していることがわかる。

accuracy の数値が 1 しかラベルを付けていない方が高かった理由として、選手以外をまとめて同一の物体として扱ってしまったため、その分類がうまくいかなかったことが推測される。また、両者の accuracy がそれほど高くなかった理由として物体の分類ではなく位置のズレが大きいのではないかと推測した。

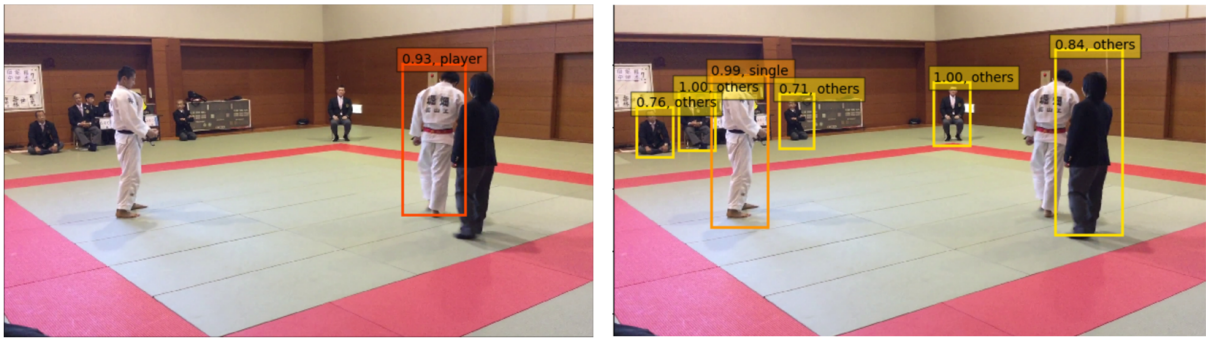
このままでは実際に選手を追跡できているか、さらには、実用的であるかどうかの評価がしにくいため以下の追加実験を遂行した。

9.4　追加実験内容

それぞれのデータを学習し終えた状態で、学習をさせていない試合の画像を複数枚検知させ表示させた。画像は 60fps の動画を 1 フレームごとに切り取り 3000 枚を用意した。

9.5　追加実験結果・考察

用意した画像を検知させた場合においては、どちらのデータを学習したモデルでも、選手間がつかみ合っている (試合が行われている) 状態は完全に検知することに成功していた。その中の例を Figure 9.6 に示す。



$(a)1\ label$ $(b)3\ label$

Figure 9.7　Ex. bad detection.

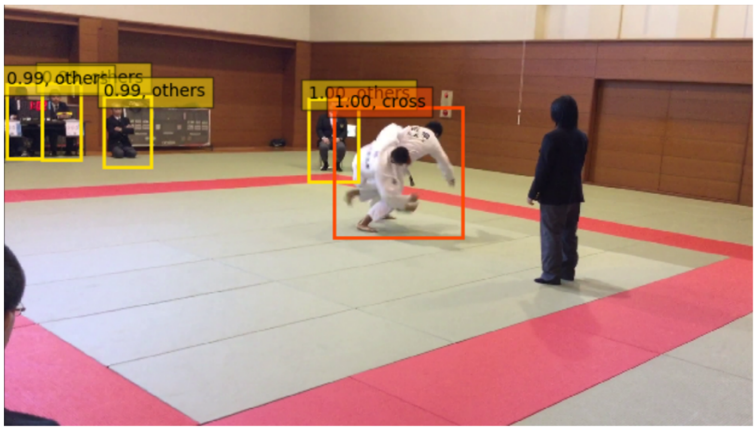


Figure 9.8　Miss detection (3 labels).

しかし、選手同士がそれぞれ離れている場合や、実際には選手同士が離れているが画像上では重なっている場合には検知できない場合があった。それぞれ検知しきれなかった画像を Figure 9.7 に示す。両者のモデルともに検知しきれていない選手が存在している。このように検知しきれていない場合は、試合の進行が止まっている場合ばかりであった。

最後に 9.3 節で推測した 3 つのラベル付けをしたデータの精度が 1 つの場合と比べて低かったことについての理由の推測の正当性を主張するための画像を Figure 9.8 に示す。Figure 9.8 では画面中央辺りにいる人間を本来は Figure 9.7(b) のように選手外として検知すべきところをできていない。また、Figure 9.6(b) の同人物のフレームがズレいてることからも、3 つのラベル付けをしたモデルでは、選手外のラベルの精度が低いためモデル全体の精度も下がってしまったことがわかる。

第10章　結論

本研究の目標としては、自動で柔道競技のスコアを計算するAIを作成する事であったが、実際にはCNNを用いた試合状況の認識とSSDを用いた選手の追跡のみを取り扱った。

試合状況の認識の実験では、VGG16と呼ばれるCNNのモデルの中でも広く一般的なものを使用したにもかかわらず期待された精度が出なかった。これを改善するため、データ数の大幅増加と分類方法を再検討する必要がある。

選手の追跡の実験では、accuracyの値が十分には上昇しなかったものの実際に、どの程度選手を追跡できているかを確認したところ、十分追跡できていた。しかし実際のカメラでの搭載を考えると処理速度の面を考慮せねばならないため、この面での研究が必要である。また、追跡とは関係ないがSSDはVGG16のモデルを基にして画像認識を行っているが、選手の認識はできていることからも、試合状況の認識の実験は述べた通りの改善点が挙げられるだろう。

また注意すべき点は、SSDを用いた物体検知を行う際は、処理速度が計算機やクラス数に依存するため、一般的なタブレット端末などの計算機で実装した場合、リアルタイムでの検出が難しい可能性がある。速度の面から見てもクラス分けは選手のみで十分である。

最終段階であるスコアの計算部に取り掛かれなかったため、今後はこの部分の研究を行い、用意された試合の動画をスコアリングできることをまず第一に目指したい。

謝辞

本研究を進めるにあたり、御多忙中にも関わらず多大なご指導を賜りました出口利憲先生に深く感謝すると共に、同研究室において共に勉学に励んだ浅野蒼太氏、三上麟太郎氏に厚く御礼を申し上げます。

参考文献

1) 我妻 幸長 (2009)『はじめてのディープラーニング』 SB クリエイティブ株式会社

2) *Nikhil Buduma* 訳:牧野 聡 (2018)『実装 DeepLearning -Python と TensorFlow で学ぶ次世代の機械学習アルゴリズム』 株式会社オーム社

3) 原田達也 (2017)『画像認識 』 講談社

4) *Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu and Alexander C. Berg*, SSD:Single Shot Multibox Detector , 2016 [https://arxiv.org/pdf/1512.02325.pdf] (2020 年 1 月検索)