ディープラーニングによる株価予測の入力情報の多重化による予測精度向上についての分析

池　田　欽　一

概 要

本研究では、深層学習技術（ディープラーニング）の株式予測において、入力情報を多重化にすることにより予測精度に改善が見られるか、シミュレーションにより検証する。具体的には、ディープラーニングの中でも画像や動画解析に応用が進む畳み込みニューラルネットワークを用い、株価のローソク足画像、および出来高のグラフによる「上がる」、「下がる」という株価変動予測の予測精度、およびそれら画像情報を組み合わせることによる予測精度の改善についての結果をまとめている。

1　はじめに

本研究で用いる畳み込みニューラルネットワーク（CNN, Convolutional Neural Network）は、視覚野での映像の処理をモデル化したネットワークモデルである。畳み込みニューラルネットワークは、特に画像からの特徴抽出能力に優れており、画像、動画認識、それを応用した自動運転や囲碁や将棋の盤面状況の良し悪しの推論などへの応用が進んでおり、人工知能技術の大幅な進展に寄与している。

以前筆者が示した、畳み込みニューラルネットワークの株価予測への応用 [1] では、ローソク足チャートと呼ばれる株価変動の画像を入力とし、数分後の株価の変動「上がる」「下がる」の予測への応用とシミュレーションによる予測精度の検証を示した。本論文では、まず株価への影響があると考えられる出来高の画像による予測精度をローソク足画像による予測と比較し、出来高による株価変動予測の可能性について分析し、さらにはローソク足チャートと出来高グラフを組み合わせた情報を入力とした場合の予測精度の変化をシミュレーションにより比較検証している。

畳み込みニューラルネットワークなど、ディープラーニングは、柔軟な入出力を学習できるが、学習するためのデータが少ないと、その柔軟性により学習データに特化、いわゆる過学習が発生することとなり、学習に用いていない検証用データの予測性能が低下する問題が発生することがある。そのため学習には、通常は大量のデータが必要となる。経済関連データでは、月次、四半期や年単位でしか入手できないデータが多いが、株価データは、各取引ごとのデータも入手でき、銘柄によっては日に数万件ののサンプルを得ることも可能である。この大量のデータにより、株価や取引数量の一見ランダムな細微な違いによる結果の違い、つまりバタフライ効果に類似のものをシステムに取り込むことが学習することができる可能性がある。

以下、2. では畳み込みニューラルネットワークについて述べ、3. では畳み込みニューラルネットワークを株価予測へ応用するための実装方法や用いる株価データ、パラメータ設定について述べる。4. では出来高の画像を入力とした場合の結果をシミュレーションにより検証し、5. で、ローソク図チャートと出来高グラフを組み合わせた入力、つまり多重入力情報による予測精度の改善についてのシミュレーション結果とその考察を示し、6. でまとめとする。

2 ディープラーニングとは

ディープラーニング（深層学習）は、脳神経細胞とそのネットワークを数理モデル化したニューラルネットワークを多層化した物ということができる。単に多層化するだけではシステムの出力側からの学習情報が入力側へ伝わる段階で急激に小さくなるいわゆる向配消失問題を事前学習や出力変換のための関数を工夫することにより解決されている。画像や動画認識、音声認識、多次元の入出力関係の学習、予測が可能であり、車の自動運転や囲碁、翻訳など幅広い分野など近年様々な分野への応用が進んでおり、従来の手法よりも良好な結果を残している [1, 4]。

ディープラーニングと一概に言っても、いくつかの方法に分けることができるが、本研究では特に画像や動画解析に有効である畳み込みニューラルネットワークを用いる。畳み込みニューラルネットワークへの入力は画像が用いられ、画像データと複数用意したフィルタへ畳み込み（convolution）を適用し、類似度を数値化する。フィルタや入力画像よりもサイズが小さいものが用いたれるが、フィルタ位置を少しずつずらしながら畳み込みを実施し、画像全体にフィルタに対応したパターンがどのように分布しているかという情報を得ることができる。この工程を実施する部分を畳み込み層と呼ぶ。さらに、畳み込みの後に、周辺の数値の平均や最大値を求める Pooling 層を準備することにより、類似パターンの分布情報にあいまいさを持たせることができるようになっている。フィルタは学習と呼ばれる最適化段階で、画像に含まれる特徴的パターンを抽出するように更新が繰り返されていく。一般的には、畳み込み層と Pooling 層を複数回実施し、入力画像に分布するフィルタに類似パターン、さらにその類似パターンの分布の状況を数値化することができる。

これらフィルタパターン分布の数値は、神経細胞をモデル化した多数のニューロンが層にまとめれたものがネットワーク状に接続されたフィードフォワードネットワークへの入力となり、数値予測やカテゴリの予測を出力層と呼ばれる最終の層で実施することができる。フィードフォワードネットワーク [5, 6, 7] は、データの入力側の 1 つ隣の層の各ユニット（神経細胞を数理モデル化したもの）からの出力を加重集計し、バイアス値を足し、活性化関数といわれる非線形関数により変換して出力側に順次送られていくシステムである。

これら、畳み込み層、Pooling 層、フィードフォワードネットワークにより、画像を入力として、何らかの数値やカテゴリを予測する畳み込みニューラルネットワーク全体が構成されることとなる。以下では、文献 [1] を引用しつつ、畳み込みニューラルネットワークについて説明する（模式図による説明など、詳細は文献 [1] を参照されたい）。

2.1 畳み込み層

本研究では、後述する株取引数量を表す出来高のグラフや株価の動き表すローソク図チャートを入力とする。ローソク図や出来高グラフの画像の画素サイズを $W \times W$ とし、画像データを $x(i,j), i = 0, 1, \cdots, W-1, j = 0, 1, \cdots, W-1$ で表すこととする。この画像の部分的な特徴を抽出するために、入力画像よりも画素数の小さい画像をフィルタとして用いるが、このフィルタのサイズを $H \times H$ とし、$h(p,q), p = 0, 1, \cdots, H-1, q = 0, 1, \cdots, H-1$ で表す。このフィルタ $h(p,q)$ を用いた画像 $x(i,j)$ の畳み込み $u(i,j)$ は以下の式のように表される [1, 3, 4]。

$$u(i,j) = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} h(p,q) x(i+p, j+q). \tag{1}$$

この畳み込みは前述のように、画像データ $x(i,j)$ をフィルタと同サイズに切り取った部分とフィルタ $h(p,q)$ の類似度を数値的に表すものとなる。類似度が高い場合には数値が大きくなり、類似度が

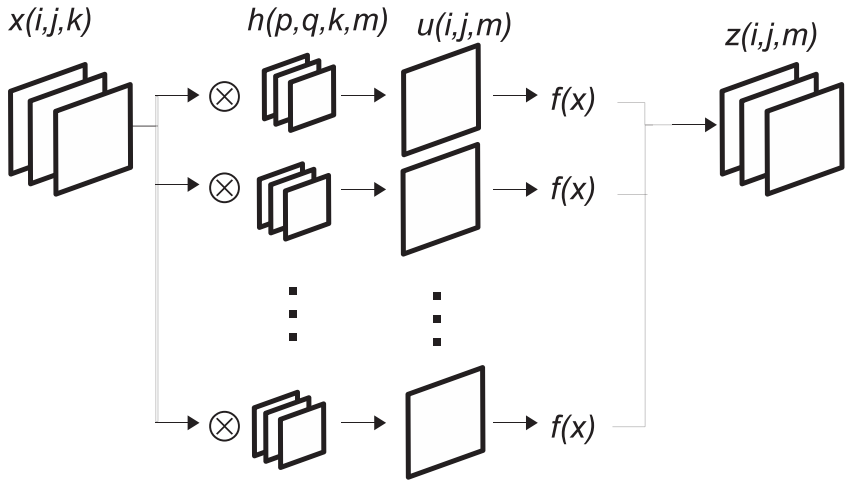

図 1: 畳み込み層（文献 [1] より引用）

低い場合には 0 に近くなり、写真のネガのように色を反転させた画像をフィルタの類似度が高い場合には負の数値となる。

フィルタをストライド s ずつ左右方向、上下方向それぞれずらしながら、この畳み込みを画像全体に適用することにより、フィルタに類似のパターンが画像全体にどのように分布しているかの 2 次元配列を得ることができる。この 2 次元配列もフィルタに類似のパターンがどのように配置しているかの画像と解釈することもできる。通常、フィルタは複数枚用意され、すべてのフィルタに同様の処理が実施され、出力された 2 次元配列へ後述する活性化関数を適用する一連の操作を実施する部分を、畳み込み層と呼ぶ。本研究ではストライド $s = 1$ としておく。畳み込み計算の際に画像の端は周辺の画素情報が不足し、周辺画素の平均値などで補うパディングを実施しないと、元の画像サイズ $x(i, j)$ に比べ、畳み込み $u(i, j)$ はフィルタサイズが奇数の場合 $2\lfloor H/2 \rfloor$、偶数の場合 $2\lfloor H/2 \rfloor$-1 だけ小さいサイズとなる。$\lfloor\ \rfloor$ は床関数である。本研究ではパディングは用いないこととする。

フィルタはランダムな数値で初期化されるが、ネットワークの出力と望ましい出力との誤差関数の勾配を各パラメータで偏微分することにより更新数量を求め、フィルタの数値、およびネットワークの接続ウェイトを修正（学習）していくことにより、入力画像とそれに対応した出力を学習していくこととなる [3]。畳み込み層のフィルタパターンは学習が進むにつれ、望ましい出力を得るために必要なある種の画像の特徴を表すようになっていく。

カラー画像の場合には、R,G,B の 3 原色に分解し、3 枚の画像を 3 チャネルとして入力し、R,G,B 画像それぞれにフィルタを適用し、結果の畳み込みの同じ位置の値の総和を取ることにより処理することができる。画素サイズを $W \times W$ でチャネル数が K である入力画像を $x(i, j, k), i = 0, 1, \cdots, W-1, j = 0, 1, \cdots, W-1, k = 1, 2, \cdots, K$ とし、入力画像の K のチャネルごとに、それぞれ M 枚のサイズ $H \times H$ のフィルタを $h(p, q, k, m), p = 0, 1, \cdots, H-1, q = 0, 1, \cdots, H-1, k = 1, 2, \cdots, K, m = 1, 2, \cdots, M$、各フィルタごとのバイアスを $b(i, j, k, m)$ で表すとき、畳み込み層の内部状態は次の式により表される。

$$u(i, j, m) = \sum_{k=1}^{K} \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} h(p, q, k, m) x(i + k, j + l, k) + b(i, j, k, m). \tag{2}$$

ただし、バイアスはフィルタの適用位置に依存せず $b(i, j, k, m) = b(k, m)$ とされるケースが多い。この

内部状態 $u(i,j,m)$ に以下のように活性化関数 $f(\,)$ を適用することにより、畳み込み層の出力 $z(i,j,m)$ が求められる。

$$z(i,j,m) = f(u(i,j,m)). \tag{3}$$

本研究では、グレースケールで表される出来高のグラフ、および株価から作成されるローソク図チャートをそれぞれ単独での入力画像とするモデル、出来高グラフとローソク図チャートを 2 チャネルとして入力するモデル、さらには株価グラフと出来高グラフをそれぞれ 1 チャネルで入力した畳み込み層、Pooling 層で処理し、その後、フィードフォワード層の入力部分で結合するモデルを提案し、シミュレーションにより予測精度を検証していく。

活性化関数には、以下のようなシグモイド関数、双曲線正接関数、 Relu（Rectified Linear Unit, Rectifier）関数（正規化線形関数）などが用いられる。

シグモイド関数：

$$f(x) = \frac{1}{1+\mathrm{e}^{-x}}. \tag{4}$$

双曲線正接関数:

$$f(x) = (\tanh(x)). \tag{5}$$

Relu 関数：

$$f(x) = \max(x, 0). \tag{6}$$

2.2 Pooling 層

Pooling 層は、通常畳み込み層（複数の畳み込み層を連続して用いることもある）の次に設置され、畳み込み層のフィルタにより表されるパターンの分布位置にあいまいさを持たせることができるものである。このフィルタパターンは学習により画像の特徴を示すようになってくるが、Pooling 操作はこの特徴の位置の変化にある程度柔軟性を持たせることができる。画像（畳み込み層で求められたフィルタと類似パターンの分布）への Pooling の適用は、画像の正方領域 $N \times N$ を考え、その中の画素値の最大値、あるいは平均値などを求める演算である。Pooling 演算の領域は、通常重複のないように適用されるので、出力サイズは入力画像の $1/N \times 1/N$ となる。$N = 2$ の場合、縦横のサイズは 1/2 画素数は 1/4 となり、この層以降の畳み込み層や後述するフィードフォワード層の計算コストの低減も可能となる。入力が多チャネルの場合にはチャネルごとに適用されるので、チャネル数を K とすると、(W, W, K) のサイズの入力からは、$(W/N, W/N, K)$ の出力が得られる [1]。Pooling 層の後に出力を標準化する正規化層を用いることもある [5]。

2.3 全結合層

畳み込み層、Pooling 層を経て（複数回適用の場合もある）、入力データは全結合層とも呼ばれるフィードフォワードネットワークへの入力となる。最終的にこのフィードフォワードネットワークの出力層において、入力データの所属カテゴリ（本研究では、株価が「上がる」「下がる」）や対応する数値の出力される。全結合層とは、以下のように神経細胞を模式化したニューロンを層状に並べたネットワークである。これは従来のフィードフォワードニューラルネットワークと同様なものである。

2.3.1 ニューロン

ニューロンとは神経細胞を次のようにモデル化したものである。

$$u_j = \sum_{i=1}^{I} w_{ji} x_i + b_j, \tag{7}$$

$$z_j = f(u_j). \tag{8}$$

ここで、u_j は第 j ニューロンの内部状態を表し、$x_i (i = 1, 2, \ldots, I)$ であらわされるニューロンへの入力（1 つ前の層のニューロンの出力値）、w_{ji} は 1 つ前の層の第 i ニューロンから第 j ニューロンへの接続の強さを表すウェイト、b_j で表されるバイアスから計算される。z_j はニューロンの出力値で、内部状態に出力関数（活性化関数）$f(\,)$ を適用した値で求められる。ただし、全結合層の最初の層への入力データは、通常 Pooling 層からのすべての出力画像（2 次元配列）の画素値をすべてベクトルとして 1 列に並べたものである。例えば、Pooling 層の出力サイズが $T \times T$、K チャネルである場合、全結合層への入力データは T^2K 次元のベクトルとなり、最初の層の必要ユニット数も同数となる。活性化関数 $f(\,)$ には、CNN の畳み込み層と同様に、シグモイド関数や双曲線正接関数、Relu 関数などが用いられる [1]。

2.3.2 フィードフォワードニューラルネットワーク

上記のようなニューロンをまとめた層をネットワークの入力側から出力へ、入力層、任意の数の中間層、出力層と並べ、各層は 1 つ入力側の層の全ニューロンからのみ入力を受け取る。このようなネットワークをフィードフォワードニューラルネットワークと呼ぶ。第 $l (l = 1, 2, \cdots, L)$ 層の内部状態 $\boldsymbol{u}^l = (u_1^l, u_2^l, \ldots, u_j^l, \ldots, u_{J^l}^l)'$ は次のように表される。

$$\boldsymbol{u}^l = W^l \boldsymbol{z}^{l-1} + \boldsymbol{b}^l. \tag{9}$$

ここで、$\boldsymbol{z}^{l-1} = (z_1^{l-1}, z_2^{l-1}, \ldots, z_i^{l-1}, \ldots, z_{I^l}^{l-1})'$ は 1 つ入力層側の第 $l-1$ 層のニューロンの出力からなる第 l 層への入力ベクトルで、$\boldsymbol{b}^l$ は第 l 層の各ニューロンのバイアス値を列ベクトルで表したものである。W は次のような第 $l-1$ 層と l 層のニューロン間のウェイト行列である。

$$W^l = \begin{pmatrix} w_{11}^l & w_{12}^l & \cdots & w_{1I^l}^l \\ w_{21}^l & w_{22}^l & \cdots & w_{2I^l}^l \\ \vdots & \vdots & \ddots & \vdots \\ w_{J^l1}^l & w_{J^l2}^l & \cdots & w_{J^lI^l}^l \end{pmatrix}. \tag{10}$$

第 l 層の出力 $\boldsymbol{z}^l = (z_1^l, z_2^l, \ldots, z_j^l, \ldots, z_{J^l}^l)'$ は次のように表される。

$$\boldsymbol{z}^l = \boldsymbol{f}(\boldsymbol{u}^l). \tag{11}$$

ここで、$\boldsymbol{f}(\boldsymbol{u})$ は活性化関数ベクトルで次のようなものである。

$$\boldsymbol{f}(\boldsymbol{u}^l) = (f(u_1^l), f(u_2^l), \ldots, f(u_{J^l}^l))'. \tag{12}$$

ただし、予測対象がカテゴリである場合、出力層のユニット数はカテゴリと同数とし、活性化関数ではなく、以下の softmax 関数により、各ユニットの出力値が 0 ～ 1 の値、総計が 1 となるように変形される。これは各カテゴリに所属する確率を表すと考えることができる。

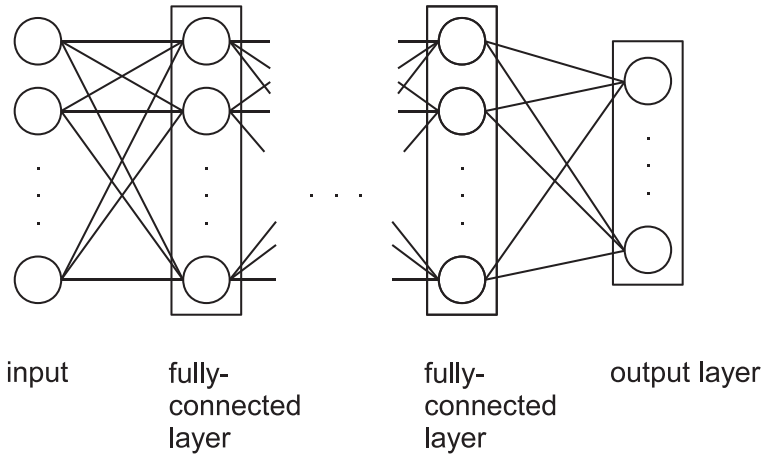

図 2: 全結合層（文献 [1] より引用）

$$z_j^L = \frac{\exp(u_j^L)}{\sum_{k=1}^{J^L} \exp(u_k^L)}. \tag{13}$$

全結合層の最も入力側にデータが与えられ、出力層へ入力データが順次渡され、出力層で CNN からのシステム全体の出力が得られる。教師データが連続値である場合は出力層のユニットは 1 つとして、学習後のこの 1 ユニットの出力を予測値として扱うことが一般的である。カテゴリデータのクラス分類の場合、出力層のユニット数をグループ（クラス）の数と同一とし、各ユニットを各グループに割り当て、学習後に入力データを与え、最も出力の大きいユニットに割り当てられたグループに所属すると予測することが一般的である [1, 4]。

2.4 畳み込みニューラルネットワーク（CNN）

入力画像は、これまでに述べたような畳み込み層（複数の場合もあり）、Pooling 層（正規化層でデータの標準化をする場合もある）を通常は数度経て、その後前述のように画素値の 1 つを 1 つの入力値として全結合層への入力値とし、全結合層を数層経過したのち出力層にて出力を得る。カテゴリ分類の場合、最終の出力層にはカテゴリの数と同数のニューロンユニットが配置され、出力の最も大きいユニットに対応するカテゴリへ分類されると予測をする。出力層のユニットの出力は式 (13) の softmax 関数により求められ、これらの出力値と実際のカテゴリ（教師データ）との間のクロスエントロピー誤差を計算し、この誤差を減ずるように誤差を入力側へ伝搬しながら、全結合層のウェイト、バイアス、畳み込み層のフィルタ値、バイアスを更新して学習を進める [3]。クロスエントロピーは、学習サンプル $n, n = 1, 2, \ldots, N$ の実際の所属カテゴリが C_n である場合、以下の式により求められる [4]。

$$E(\theta) = -\sum_{n=1}^{N} \log z_{C_n}. \tag{14}$$

ここで、z_{C_n} は学習サンプルの実際の所属カテゴリに対応する出力層のユニットの softmax 関数適用後の出力値であり、θ はネットワークのパラメータ、つまり、全結合層のウェイト、バイアス、畳み

込み層のフィルタ値、バイアスである。カテゴリ判別ではなく、数値を学習、予測する場合には、誤差関数として、望ましい出力（教師）とネットワークの出力の二乗誤差が用いられる。

本論文では、畳み込み層のフィルタ、バイアス、および、全結合層のウェイト、バイアスの学習には、Adam(Adaptive moment estimation) を用いる。Adam とは、少量のメモリで一次勾配のみにより適用可能な効率的な確率的最適化手法で、パラメータごとに勾配の平均、分散の推定値を利用する手法で、多くのケースにおいて良好な学習が得られることが示されている [8]。各パラメータの勾配の平均、分散を推定したものを利用し、勾配が疎になる場合や非定常に強い性質を持っている [9]。

3 畳み込みニューラルネットワークによる出来高を用いた株価変動予測

筆者が以前示した株価予測への畳み込みニューラルネットワークの応用では、1 分毎の株価四本値（始値、安値、高値、終値）より作成したローソク足チャートを入力画像、1 ～ 5 分後に株価が「上がる」か、「下がる」（変わらないを含む）かを予測する方法を示した [1]。本論文では株価のみではなく、株式の出来高情報も入力に用いた場合の予測精度向上の可否について分析するが、まずは出来高のみによる株価の変動（「上がる」、「下がる」）予測の方法と、および予測精度について、ローソク足チャートによる予測と比較しながら示すこととする。

出来高とは、株式売買が成立した株の数量であり、その増減が株価と関係性があるという考えもある。当然ながら、出来高は取引が活発になると増加し、取引が緩慢になると減少する。また、株価は人気のバロメータと考えることもでき、例えば業績が向上し、株を取得したいと考える人が増えると需要が供給を上回り株価が上昇する。業績の上方修正など株価への好材料がでると銘柄の人気が出て取引が活発になり出来高が増加する。逆に業績悪化など悪材料が出た場合にも一時的に出来高が増加することはあるが、人気減少で出来高は急激に減少する。その後株価への判断材料が出ない場合には変動が少ない動きを見せ、そろそろ上がるのではないかと考える人が増えてくると出来高が先行して上昇するケースがある。あるいは、株価上昇後に少しずつ株価が下がり始め、そろそろ下がり始めるのではないかと考える人が多くなると出来高が増え、さらに株価低下に拍車がかかるということも発生する。

これらの例のように株価と出来高は関係していると考えられるが、出来高が徐々に上昇しその後株価が上昇、逆に出来高が徐々に減少し、その後株価が下降というように、出来高が株価に先行して増減することがあると考えられており、株価そのものによる株価予測よりも、出来高を用いた株価予測の方がより先の予測に適している可能性もある。以下では、出来高画像、システム構築方法、パラメータの設定についてまとめていく。

3.1 入力に用いる出来高画像

ディープラーニングの学習では、システムが特定のデータに特化することを防ぐため、つまり汎化性を担保するためには、大量のデータによる学習が必要である。株価や出来高データを用いた学習を実施する場合、よりサンプル数の多いティックデータ（取引発生ごとのデータ）を用いる方が 1 日、あるいは 1 か月ごとなどのデータより適している。ただしティックデータは取引が発生する間隔が決まっておらず、1 分先、2 分先など確定的は期間先の株価の予測に用いる事は困難である。そこで、本研究では 1 分ごとに株価や数量が観測される 1 分足データを用いることとする。また、個別銘柄への適用も可能であるが、1 分間に 1 度も取引がないとデータが観測されず欠損値となり、また個別企

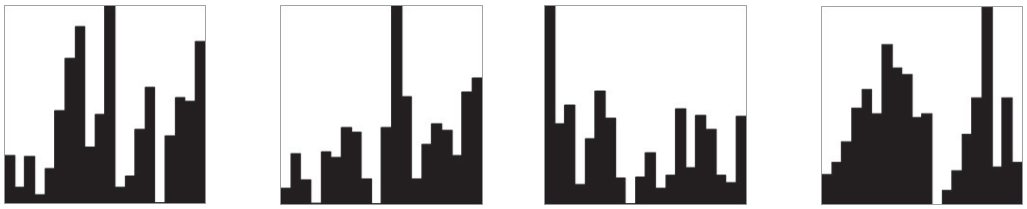

図 3: 入力に用いる出来高画像

業の業績などにより過去の株価や出来高と関連のない変動が発生することがあるので、本研究では、安定的にデータが観測され、個別企業の影響がない日経平均株価を予測の対象としてシミュレーションすることとする。

図 3 には、20 分間の出来高の変化を表す出来高画像の例を示している。x 軸が時間（1 分間隔）、y 軸が出来高である。20 分間としたのは、著者が示した文献 [1] の結果と比較検討するためである。この画像を用いると出来高の増減や傾向などをシステムへ入力することができ、株価と出来高の関連性がある場合には、予測が可能であると考えられる。株価は自然現象、物理現象と異なる点として、例えば先に示した出来高が株価に先行して増減するということを信じるトレーダー（人間であっても機械であっても）が存在すると、出来高が増えてきたので株価が上昇するだろうと考え、株を購入すると株式に対する需要が増え、さらに株価上昇に貢献する。つまり、一定数株価と出来高の関係性があると考えるトレーダがいると出来高による株価予測がさらに有効になると考えられる。

東京証券取引所では、朝 9 時から 11 時 30 分の前場と 12 時 30 分から 15 時までの後場があるが、前場、後場それぞれで、150 分分のデータが観測できる。本論文では、文献 [1] と同様に、1 つの画像に 20 分のデータを用い、最大 5 分先までの予測を実施するので、前場、後場それぞれ、126 の入力画像と教師データ（1 から 5 分後に上がるか下がるか）が作成できる（本研究では、前場後場、および本日と翌日の株価には連続性はないものとして取り扱う）。2017 年 8 月 1 日から 2018 年 1 月 26 日の間の市場営業日の 120 日分を用い、30,240 個の画像を作成した。この画像の 90%を畳み込みニューラルネットワークのパラメータ最適化に用いる学習データとし、残り 10%を学習には用いず、学習していないデータをどの程度正確に予測できるかの汎用性検証（テスト）データとして用いることとする。10%のテストデータは、実験ごとに全サンプルの中から毎回ランダムに選択している。

教師データはタイムステップ T_{pred} 先に、1 分足終値基準で「上がる」(1 で表す) か、または「下がる」(0 で表し、終値が変化しない場合も含む) のカテゴリデータを用いることとする。

3.2 実装方法

本研究の入力データである出来高グラフは、日経平均の 1 分足の出来高から Microsoft Excel の縦棒グラフを用い VBA により全期間分作成し、図 3 にあるような 52 ピクセル× 52 ピクセルの画像として出力した。

出来高による予測との比較や多入力情報の際に用いたローソク足チャートは、日経平均の 1 分足の出来高、始値、終値、高値、安値から、Microsoft Excel の株価チャートグラフ形式で 1 枚当たり 52 ピクセル× 52 ピクセル画像を作成し、これを出来高グラフと同様に入力データに加工した。ローソク足チャートの入力画像の例は図 4 に示しているように、陽線を白、背景色をグレー、陰線と上下のヒゲを黒で作成し、畳み込みニューラルネットワークへの入力の際、白を-1、背景を 0、黒が 1 となるように標準化した [1]。

図 4: ローソク足チャート（文献 [1] より引用）

CNN システムの構築には、文献 [1] と同様に Chainer フレームワーク（ライブラリ）を用いた [8]。Chainer とは、ネットワークを柔軟に表現できるフレームワークで、ディープラーニングの計算では、計算結果のみでなく計算過程や傾きを保存しておき、ウェイトの修正量計算に利用していることが特徴である。さらに Chainer では GPU を用いた並列計算が容易に実施可能で、本研究のシミュレーションでは、CPU のみによる計算のおよそ 10 倍の計算速度となり、並列化は大量のデータ（ビッグデータ）を使い大量の計算を実施するディープラーニングシミュレーションでは必須の条件と考えられる。また、Chainer 以外にもディープラーニングのフレームワークは多数あるが、Chainer は単純なネットワークであっても、複雑なネットワークであっても柔軟に設計ができることや、ネットワーク設定を別途準備するなどの必要もなくプログラミング言語 Python のみで記述が可能で、開発のしやすさも特徴である [8]。

なお、本研究では Python、Chainer、および GPU 並列計算用のライブラリである NVIDIA 社の CUDA ライブラリを Microsoft Windows10 へインストールしたシステム上でシミュレーションを実施した。

3.3 畳み込みニューラルネットワークの設定

畳み込みニューラルネットワークでは、入力されるデータは 2 次元画像で、本研究では、3.1、3.2 で示したような出来高グラフ画像、およびローソク足チャート画像である。まずは出来高グラフのみを入力に用い、出力層は 2 ユニットとし、$1, 2, 3, 4, 5$ 分後に株価「下がる」（変わらないを含む）かが「上がる」かを 2 ユニットのそれぞれに割り当て、どちらのユニットの出力が大きいかによりカテゴリ（「下がる」、か（「上がる」か）の予測を行う。つまり、1 つ目のユニット（0 番ユニット）の出力の方が大きければ株価予測は「下がる（変わらない含む）」、2 つ目のユニット（1 番ユニット）の出力の方が大きければ「上がる」という予測となる。システムの学習時には正しい出力が教師データとして与えられているので、予測との誤差を計算し学習を進めていくこととなる。

シミュレーションによる検証では、出来高画像やローソク図チャートに用いた最後の時刻の 1 分足終値に比較し、1 分後、2 分後、...5 分後の 1 分足終値が「上がる」か、「下がる」かを予測することとする（何分後の変動を予測するかを $T_{step} = 1, 2, 3, 4, 5$ で表すこととする）。本シミュレーションでは日経平均株価を用い、変わらないケースは稀にしか発生しないので、変わらない場合は特に別の区分を設けず、下がる区分へ含めることとした。

入力画像は、1 つ目の畳み込み層で 5×5 のサイズのフィルタ 20 枚により類似パターンの検出を行う。パディングは用いていないため、52×52 のサイズの画像が 48×48 の類似度の分布画像（2 次元配列）20 枚に変換され、出力される。これら出力は 2×2 の最大値プーリングにより 24×24 のサイズの類似パターンの分布位置にあいまいさを追加した数値出力 20 枚をなる。この数値出力も 2 次元であるため画像としてみることができ、この出力を 2 つ目の畳み込み層で 1 つ目の層と同様に 5×5

のサイズのフィルタ 50 枚により畳み込み計算を行い、20×20 のサイズの 2 次元数値出力 50 枚となる。さらに最大値プーリングにより、10×10 サイズの 50 枚の出力を得る。この 10×10 サイズの 50 枚の出力は、1 つ 1 つの画素値を一列に並べると一枚当たり 100 次元の数値ベクトルとなり、これを 50 枚分並べると 5000 次元のベクトルとなる。これらはフィルタパターンの分布の状況を表しているが、これを全結合層の入力とする。全結合層は第 1 層 5,000 ユニット、第 2 層 2,000、出力層は「上がる」「下がる（変わらない含む）」の 2 カテゴリに対応した 2 ユニットとした。畳み込み層、および、全結合層の活性化関数には式 (4) のシグモイド関数を用いた。

また、ディープラーニングでは、フィルタ枚数の増加や、全結合層の層の数や各層のユニット数を増やすと、畳み込み層のフィルタの値や全結合層の接続ウェイトの最適化に用いる学習データ（training data）にシステムが特化しすぎてしまい、学習に用いていない評価用データ（test data）の予測精度が低下する現象、いわゆる過学習（overfitting）が発生することが知らている。この過学習を防ぐためには、学習データを増やすことや、フィルタやウェイト更新の際、出力値をある割合強制的に 0 にするドロップアウト（dropout）を用いること、学習データや評価用データとは別に検証データを準備し、この検証データについての予測精度が低下する場合に学習を打ち切る early stopping などが知られている。本研究では、ドロップアウト [4, 11] を用い、全結合層で使用するユニットの比率、および畳み込みでの出力値を用いる比率を $p, 0 \leq p < 1$ と表すこととする。ドロップアウトを用いると、畳み込みの類似度出力値や全結合層のユニット出力値の合計が平均 $1/p$ 倍となるので、出力値を p 倍して、出力値の合計の平均値が同程度となるような調整が必要となる。学習後の評価、つまりテストデータを用いた予測では、すべてのユニットを用いて出力を計算する。

4 出来高画像による株価予測シミュレーション結果

ここでは出来高画像の予測精度について、ドロップアウト率の変化と予測のタイムステップ T_{pred} の違いによる影響をシミュレーションにより検証することとする。また、文献 [1] で筆者が行った、株価の変動パターンを表すローソク図チャートを入力に用いた場合と比較も行い、過去の出来高、株価それぞれの予測能力の考察も行うこととする。

ここでは予測の正確性を検証するため、文献 [1] と同様以下の予測精度（Accuracy、acc と標記）を用いる。

$$\mathrm{acc} = (\text{上がると予想して正解した数} + \text{下がると予測して正解した数})/\text{全データ数}. \qquad (15)$$

予測精度は、学習データについて求めると、システムがどの程度学習できているかの目安となり、評価用データについて求めると、未学習データへの適用可能性、つまり汎化性能を確認することができる。全データ数とは学習データ、あるいはテストデータの全サンプル画像の枚数である。

以下では、ドロップアウトの率、および、画像に用いたデータから何分後を予測するかの違いによる予測精度の差について検証していく。

4.1 ドロップアウト率 p の影響

まずは、ドロップアウトの際のユニット、フィルタ出力使用率 p を変更することによる、テストデータの予測精度（Accuracy）の変化について検証を実施する。ここでは、何分後の予測をするかのタイムステップは $T_{pred} = 3$ で固定し、ユニット使用比率は $p = 0.1, 0.3, 0.5, 0.7, 0.9, 1.0$ の 6 ケースについてシミュレーションを実施し、結果について考察を行う。$p = 1.0$ では学習時にすべてのフィルタ出力、ユニットを使用し、ドロップアウト未使用となる。

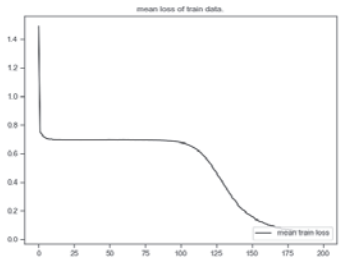

図 5: 学習データの loss

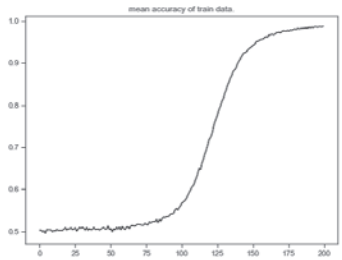

図 6: 学習データ acc

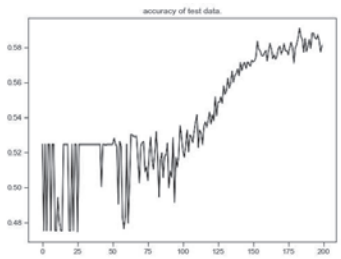

図 7: テストデータ acc

p	0.1	0.3	0.5	0.7	0.9	1.0
train acc	0.94	1.00	1.00	1.00	1.00	1.00
test acc	0.56	0.58	0.57	0.57	0.57	0.56
profit	3,426	4,796	4,304	3,700	4,013	3,064
Rate of profit	112.1%	157.0%	140.9%	121.1%	131.3%	100.3%

表 1: 出来高画像のユニット使用比率による結果への影響

図 5 には、ユニット使用率 $p = 0.3$ の場合の、学習データについての式 (14) で示されるクロスエントロピー誤差（loss と標記）の学習回数（epoch）による変化を示している。この図から学習回数が増えるとクロスエントロピー誤差は減少しており、学習データについて、入力画像とその 3 分後の株価の「上り」「下がり」の関係を、最初ランダムに作成されたフィルタとネットワーク接続強度の更新を通して学習できていることを示している。他の p の値を用いた際もほぼ同様の学習曲線となったが、acc に大きな変化が見られなくなる学習回数はユニット使用率が低いほど多くなった。

また、図 6 には、学習データについての予測精度（acc。以下、判別制度と標記）の学習回数（epoch）に対する変化を示している。学習開始時には判別精度は約 50%で、「上がる」、「下がる（変わらないを含む）」をランダムに選択した確率と同じであるが、学習回数が増えるにしたがって、100%に近づいていき、学習データの入出力関係をシステムが学習できていることを示している。また、図 7 には、テストデータについての予測精度を示している。学習回数が 60 回ほどまでは、およそ 47.5%から 52.5%の間で変動しているが、これはテストデータの「上がる」「下がる」のデータの比率に近い値となっており、システムはすべて「上がる」、あるいはすべて「下がる」という予測のどちらかを繰り返していると考えられる。その後学習回数が増えるに従って予測精度は上昇し、およそ 58%前後で微小な変動をしており、それまでと比べ大きな変化が見られず、学習がほぼ収束していると考えられる。

表 1 には、ユニット使用率（p）を変化させたときの、学習 200epoch 後の、学習データの判別精度（train acc）、テストデータの予測精度（acc）、テストデータの予測が「上がる」の時は株を 1 株購入し 3 分後に売却、「下がる」予測の時には株を空売り（借りて売却）し、3 分後に清算（買戻し返済）という取引をした際の損益（profit）のテストデータすべてについての合計額をまとめている。train acc と test acc は少数点第三位にて四捨五入、profit については小数点第一位にて四捨五入している。

学習データをどの程度判別できているかを表す判別精度（loss acc）については、ユニット使用率が $p = 0.1$ の場合には 0.94 となり、ほぼ学習はできているが、判別できていない学習データも存在していることが分かる。ユニット使用率 p が $0.3, 0.5, 0.7, 0.9, 1.0$ の場合には 1.00 となっている。表示の都合で数値は四捨五入をしているので 1.00 となっているが、$p = 0.9$ で 1 サンプル、$p = 0.5$ で 5 サンプル

p	0.1	0.3	0.5	0.7	0.9	1.0
train acc	0.97	1.00	1.00	1.00	1.00	1.00
test acc	0.58	0.61	0.59	0.61	0.61	0.60
profit	4,478	5,825	4,948	5,909	6,102	5,728
Rate of profit	146.5%	190.6%	161.9%	193.4%	199.7%	187.5%

表 2: ローソク足チャートのユニット使用比率による結果への影響（文献 [1] より引用）

について学習ができていないケースはあった。ただしデータ総数 27,216 サンプルと比較するとわずかな数であるので、ほぼ学習データの入出力関係を学習できていることを示している。$p = 0.3, 0.7, 1.0$ ではすべてのサンプルを正しく判別できていた。テストデータについての予測精度は、$0.56 \sim 0.58$ となっており、大きな差はないが、$p = 0.3$ のケースで 0.58 と最も良い予測精度となっている。また、テストデータについて、予測に従って取引をした場合の損益（profit）を見ると、$p = 0.3$ のケースで、4,796 となっている。この額は、約 58%の取引で利益が出て、約 42%の取引で損失が出るが、それらの額を集計した値である。テストデータのサンプル数3,025 で割るとおよそ 1.59 となり、1 回の取引により平均しておよそ 1.59 円の利益となった。この損益額は、もちろん予測精度にも影響を受けるが、大きく変動するケースをいかに正しく予測できるかということも影響している。

本シミュレーションで用いた 2017 年 8 月 1 日から 2018 年 1 月 26 日の 1 分足の終値の平均値は 21,557 円で、3 分間保有を継続するので、平均額からおよその必要と投資額を計算すると 64,671 円（$= 21557 \times 3$）となる。1 分ごとに予測に従い株の購入、空売りし、3 分後に売却、清算を 1 年間続けると、年間の利益率（Rate of profit）は 157.0%となり、投資資金は 1 年でおよそ 2.57 倍となる。最も利益率が低いケースでも $p = 1.0$ の 100.3%となり、投資資金はおよそ 2 倍となる結果になった。

ただし、profit や Rate of Profit の額は、株式市場は直接取引のできない日経平均株価という指標を対象としており、さらには、出来高のグラフに用いている最も遅い時間において予測、取引ができると仮定しているが、実際にはタイムラグが発生することとなり、さらには株式購入や空売りの手数料を考慮していないので、あくまで仮想的なシミュレーションでの値となる。

これらの出来高グラフによる株価の「上り」「下がり」の予測結果にから、出来高のグラフにより、3 分後の株価変動の予測はある程度可能であり、出来高と株価変動の関係性があるということが読み取れる。

次に、これらの結果を文献 [1] のローソク図チャートを用いた結果と比較してみる。表 2 には文献 [1] より引用したローソク図チャートによる予測の学習データ、テストデータについての予測精度（acc）、テストデータについての総利益（profit）、1 年間の投資の利益率（Rate of profit）を示している。ただし、Rate of profit については新たに計算し追加した。

出来高による予測とローソク図チャートによる予測を比較すると、学習データの判別制度についてはどちらの場合も $p = 0.3, 0.5, 0.7, 0.9, 1.0$ で 1.0、$p = 0.1$ で 0.94、0.97 と同様な傾向を示している。テストデータについての予測精度は全体的にローソク図チャートで高くなっており、テストデータに対する利益や 1 年間の予想利益率もローソク図チャートを用いた予測の方が高くなっている。これより、出来高よりも株価情報の方が 3 分後の株価変動について説明力があることが分かる。これは、株価の変化については直前の株価との関連性が高いことや、ローソク図チャートは 1 分ごとの始値、終値、高値、安値の四本値情報により作成されるので、出来高グラフよりも、より多くの情報を持っていることが理由であると考えられる。ローソク図チャートによる予測のテストデータについての予測精度は、$p = 0.3, 0.7, 0.9$ で他の使用率よりも高くなっているが、出来高グラフによる予測と同様に大きな差は生じていないことが分かる。

T_{pred}	1	2	3	4	5
train acc	1.00	1.00	1.00	1.00	1.00
test acc	0.49	0.52	0.58	0.61	0.62
profit	224	1,721	4,796	7,771	7,910
Rate of profit	22.0%	84.5%	157.0%	190.7%	155.3%

表 3: 出来高画像の T_{pred} の違いによる結果への影響

T_{pred}	1	2	3	4	5
train acc	0.99	1.00	1.00	1.00	1.00
test acc	0.49	0.57	0.61	0.63	0.63
profit	-60	2,795	5,825	7,726	8,801
Rate of profit	-5.9%	91.5%	190.6%	189.6%	172.8%

表 4: ローソク足チャートの T_{pred} の違いによる結果への影響（文献 [1] より引用）

4.2 予測のタイムステップ T_{pred} の違いによる影響

次に、予測のタイムステップ T_{pred}、つまり、出来高グラフやローソク図チャート作成に用いた最後の時刻から何分後の終値の「上り」「下がり」を予測するかによる、学習データの学習率、テストデータの予測精度、テストデータについて予測に従って投資行動を行った場合の損益額の違いの比較を行う。$T_{pred} = 1, 2, 3, 4, 5$ の 5 つのケースを仮定し、学習時のドロップアウトにおけるユニット使用率は、$p = 0.3$ で固定しておく。まず、出来高による予測についてまとめ、ローソク図チャートによる予測との比較を行う。

表 3 には、予測タイムステップ T_{pred} が 1 ～ 5 の場合の学習データについての判別制度（train acc）、テストデータの予測精度（test acc）、ドロップアウトの検証時と同様にテストデータの予測結果に従って投資行動を行った場合の損益額（profit）、1 年間の推定利益（Rate of profit）を示している。

どのタイムステップについても、判別制度は 1.0 で学習は収束していることが分かる。テストデータの予測精度については、タイムステップが長くなるほど精度が上がっていることが分かり、出来高の情報は特に 4 分後や 5 分後の株価の変動に関連がより強いことを示唆している。1 分後、2 分後の予測は、予測精度が 0.49、0.52 とランダムな投資をした場合の 50%に近い値となっており、出来高グラフによる予測が適切に行われていないことが分かる。これらのことより、出来高は直近の株価変動ではなく、一般的に言われているように、株価に対して出来高の先行性があることが読み取れる。テストデータの損益額は予測精度の向上に従って増加していくが、1 年間の推定利益率は予測タイムステップが長くなることにより必要投資額が増えて聞くので、$T_{pred} = 4$ が最も高い値となった。これにより、出来高グラフにより予測を行う場合には、4 分間ホールドすることが最適であることが分かる。

次に、表 4 に示した文献 [1] のローソク図チャートを用いた T_{pred} の違いによる結果との比較を行う。学習データの判別精度はほぼ 1 となり、出来高と同様、学習が収束していることが分かる。テストデータの予測精度については、出来高グラフの場合と同様に 4 分後や 5 分後の株価変動の予測精度が高いことが分かるが、精度は 63%と出来高グラフの 61%、62%よりも高い数値を示している。ローソク図チャートによる予測では、2 分後も比較的高い精度を示しており、過去の株価変動は出来高よりも株価変動への影響が早く出る可能性を示唆している。1 分後の予測は出来高グラフと同様、予測

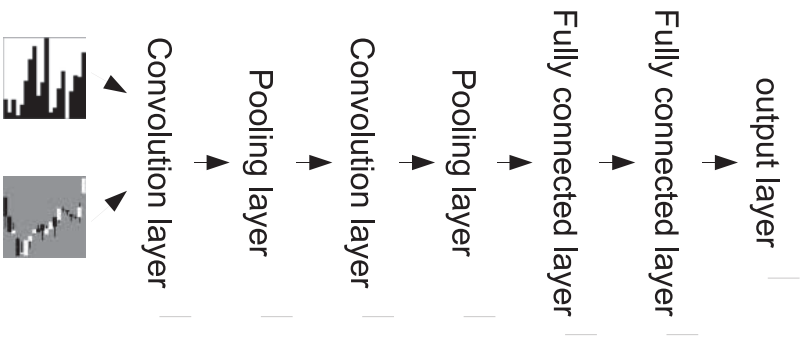

図 8: 画像を 2 チャネルとして入力するシステム

は約 50%となっており、予測が困難であることを示している。これは、株価変動を見て投資行動を決めているトレーダー、あるいはプログラムの行動が株価に影響を与えるまでに 1 分以上はかかるということが理由と考えられる。損益額（profit）、1 年間の推定利益（Rate of profit）は、ドロップアウト率 p の影響と同様に推計した値であるが、ローソク図チャートによる予測の精度が高いことが影響し、出来高グラフよりも高い数値を示している。

出来高グラフとローソク図チャートによる予測の比較より、全体的にローソク図チャートによる予測精度が高く、この予測に従って投資行動を行った場合のテストデータにつていの損益も、どちらの場合も予測が困難な 1 分後を除いて、ローソク図チャートによる予測の方が高いことが分かる。

5 入力情報の多重化による予測シミュレーション

ここまでは、畳み込みニューラルネットワークへの入力は、出来高グラフかローソク図チャートのいずれか 1 つであったが、ここではこれら情報を組合せ、入力画像の多重化による予測精度への影響をシミュレーションにより分析する。画像をシステムへ取り込む方法は 2 通りとし、1 つ目は式の畳み込み層の計算において、入力データを 2 チャネル化、つまり $K = 2$ とする方法で、2 つ目は、畳み込み層、および Pooling 層は、出来高グラフとローソク図チャートについてそれぞれ 1 チャネルで分けて計算し、全結合層で情報を結合する方法を設定した。前者では早い段階で入力画像が多重化され、後者では画像の畳み込み層による画像の特徴抽出は個別に行うことにより、各画像の影響を有効に利用できる可能性が考えられる。

5.1 2 チャネル入力による予測

図 8 には、出来高グラフとローソク図チャートを 2 チャネルの入力データとして用いるシステムの模式図を示している。最初の畳み込み層においては各画像ごとにフィルタを準備し、特徴検出を行うことができるが、それ以降の Pooling や 2 回目の畳み込み層、Pooling 層、全結合層では多重化されたデータが出力層まで伝えられていくことになる。つまり、この方法では、早い段階で各画像の情報が統合されることとなる。

表 5 には、出来高グラフとローソク図チャートを 2 チャネル入力として用いた場合の予測ステップ T_{pred} を 1 ～ 5 と変えた時の学習データの判別率、テストデータの予測精度、予測に従ったテストデータの損益額、1 年間の推定利益率を示している。学習データの判別率（test acc）は、すべての T_{pred}

T_{pred}	1	2	3	4	5
train acc	1.00	1.00	1.00	1.00	1.00
test acc	0.50	0.57	0.62	0.63	0.66
profit	779	3,588	7,170	8,696	10,930
Rate of profit	76.5%	176.1%	234.6%	213.4%	214.6%

表 5: 画像を 2 チャネルとして入力するシステムの T_{pred} の違いによる結果への影響

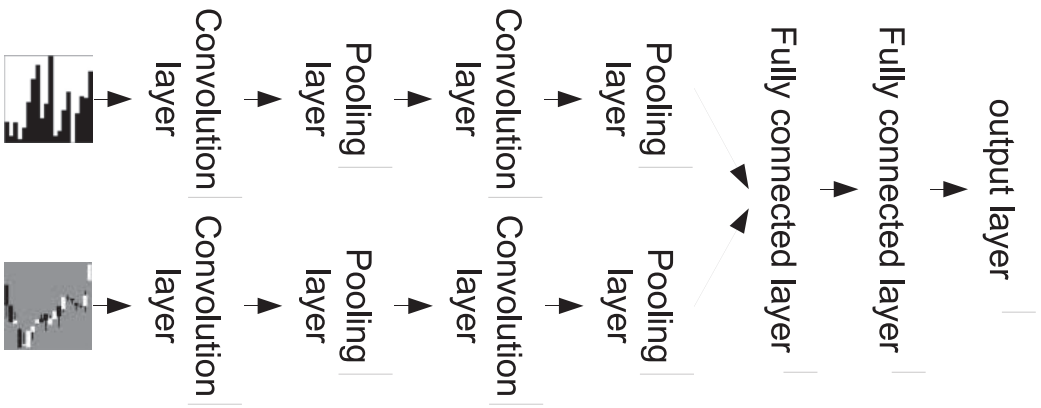

図 9: 畳み込み層と Pooling 層を画像ごとに用いるシステム

で 1.0 となり、十分な学習ができていることがわかる。テストデータの予測正解率（test acc）を、出来高グラフを単独で入力に用いた表 3 と比較すると、すべての T_{pred} で上回り予測精度が向上していることがわかる。ただし、$T_{pred} = 1$ のケースでは予測精度は 50%であるので、ランダムに予測した正解率と同じで、入力を複数画像にしても予測が困難であることに変わりはないと考えられる。ローソク図チャートを単独で入力に用いた表 4 の結果と比較すると、同等以上の予測精度を示している。特に $T_{pred} = 5$ のケースにおいて予測精度の向上が見られる。テストデータについて予測結果に従って投資した場合の損益（profit）、および年間の推定利益（Rate of profit）を見ると、出来高グラフ、ローソク図チャートをそれぞれ単独で用いた場合よりも、すべてのケースで増加している。これらの結果より、予測精度が向上したケースは個々の投資で利益を得る機会が増えたということもあるが、予測精度がそれほど向上していないケースでも利益が増加していることから、出来高とローソク図の情報を合わせることにより、株価が大きく上がる、あるいは大きく下がる場合の予測の正確性が上昇しているということが考えられる。1 年間の推定利益は $T_{pred} = 3$ のケースが最も高く、予測結果に従って投資を行うと、1 年間で投資資金はおよそ 3.34 倍になる。しかしこれらの投資結果は、先にも述べたように、瞬時の投資予測と日経平均への投資という仮定での話で、毎分投資を行う高頻度取引を行う必要もあり、シミュレーション上での結果となる。

5.2 全結合層での入力データの結合

図 9 には、出来高グラフとローソク図チャートの情報を全結合層で統合するシステムの模式図を示している。各画像ごと畳み込み層、Pooling 層を 2 回準備し、全結合層への入力時にデータは多重化

T_{pred}	1	2	3	4	5
train acc	0.99	1.00	1.00	1.00	1.00
test acc	0.49	0.58	0.63	0.67	0.66
profit	262	3,917	7,738	11,210	11,454
Rate of profit	25.7%	192.3%	253.2%	275.1%	225.0%

表 6: 畳み込み層と Pooling 層を画像ごとに用いるシステムの T_{pred} の違いによる結果への影響

され出力層まで伝えられる。つまり、この方法では、2 チャネルとして用いた場合に比べ、各画像からの特徴抽出をそれぞれ実施した後データが統合されることとなる。

表 6 には、全結合層で情報を多重化する方法において、T_{pred} を変化させた場合のシミュレーション結果を示している。学習データの判別精度はすべてのケースでほぼ 1 となり、学習は収束していることが分かる。テストデータの予測精度を最初の畳み込み層で情報の多重化を行う表 5 の結果と比較すると、$T_{pred} = 1$ では、下回っているものの、それ以外は同等以上で、特に $T_{pred} = 4$ のケースで予測精度の向上が見られる。$T_{pred} = 1$ では、これまでのすべてのシミュレーションで予測が困難で、誤差の範囲と考えられる。テストデータの予測に従った投資の損益は、$T_{pred} = 1$ 以外のケースすべてで入力層で多重化するモデルよりも増加しており、1 年間の推定利益率も $T_{pred} = 1$ 以外のケースで増加している。特に $T_{pred} = 4$ のケースでテストデータの損益額、1 年間の推定利益がどちらも他のケースと比べて大きく向上していることがわかる。様々な仮定の下での推定であるが、このケースでは 1 年間で投資額がおよそ 3.75 倍となる結果となった。これらの結果より、全結合層まで出来高グラフ、ローソク図チャートを別々に処理するシステムにより、それぞれの画像の特徴量が適切に抽出され、入力時に多重化するよりも良好な結果が得られることが分かった。この結果は、今回の入力画像が出来高グラフと株価のローソク図チャートという元のデータ、性質が異なる画像であったことが大きな要因であると考えられる。

6 むすび

本研究では、畳み込みニューラルネットワーク用いた株価変動予測について、出来高グラフによる予測、および入力画像の多重化による予測精度の向上について提案し、シミュレーションによる検証を実施した。出来高グラフによる予測では、以前著者が示したローソク図チャートによる予測よりも予測精度はやや劣るが、出来高のみにより株価の「上がり」「下がり」の変動予測が可能であることを示し、入力画像の多重化では、単独の画像を入力とするよりも予測精度、株価収益が向上することを示した。特に畳み込み層とと Pooling 層を画像ごとに準備するシステムの方が、各画像の特徴を適切に抽出でき、予測精度は向上することが分かった。今後は、さらなる予測精度の向上について検討を進めたい。

謝辞

本研究は JSPS 科研費 JP17K01267 の助成を受けたものであり、ここに感謝の意を表します。

参考文献

[1] 池田欽一, 株価ローソク足チャート画像を用いた畳み込みニューラルネットワークによる株価変動予測, 北九州市立大学商経論集,54(1,2,3,4), 1–18, 2019, https://kitakyu.repo.nii.ac.jp/?action=repository_uri&item_id=678&file_id=22&file_no=1.

[2] 池田欽一, 林田実, ディープラーニングの株価予測への応用, 北九州市立大学商経論集,52(1,2,3,4), 13–26, 2017, https://kitakyu.repo.nii.ac.jp/?action=repository_uri&item_id=552&file_id=22&file_no=1.

[3] Yann Lecun and Léon Bottou and Yoshua Bengio and Patrick Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, 2278–2324,1998.

[4] 岡谷貴之, 深層学習, 講談社,2015.

[5] Rosenblatt F.,“The Perceptron: A probabilistic model for information storage and organization in the brain”,Psychological Review, 65, 386–408, 1958.

[6] Minsky M. and Papert S.,Perceptrons: An Introduction to Computational Geometry. MIT Press, MA,1969.

[7] Rumelhart D. E. , Hinton G. E. and Williams R. J. ,“Learning internal representations by error propagation. Parallel Distributed Processing”, 1, MIT Press, MA, 318-362, 1986.

[8] Diederik P. K. , Jimmy L. B., “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”, Conference paper at ICLR 2015,1-15,2015.

[9] “A Powerful, Flexible, and Intuitive Framework of Neural Networks”,http://chainer.org/.

[10] 神嶌敏弘（編）, 麻生英樹, 安田宗樹　他（著）, 深層学習, 近代科学社, 2015.

[11] Srivastava N. , Hinton G. E. , Krizhevsky A. , Stutskever I. , and Salakhutdinov R, “Dropout: A simple way to prevent neural network from overfitting”, Journal of Machine Learning Research, 15(Jun), 1929–1958, 2014.