時系列的整合性を捉えた２Ｄ畳み込みオートエンコーダによる異常検知手法の提案

加藤涼斗[1]　小島俊輔[1]

概要：生産現場の異常検知手法として一般的に用いられる２Ｄ畳み込みオートエンコーダは，静止画像など空間情報に強い反面，時間情報に弱く，単体では時系列的整合性を捉えられないという問題がある．本研究では，プログレスバーを用いて画像に時間情報を埋め込むという前処理レベルの手法を提案する．さらに，前処理の種類によって時系列的整合性へどのような影響を与えるか検証する．プログレスバーの前処理と２Ｄ畳み込みオートエンコーダの組み合わせ手法は形状異常だけではなく，出現タイミングの異常に対しても有効に機能した．

キーワード：：動画像認識・理解，異常検知，機械学習，Autoencoder

An Anomaly Detection Method Using a 2D Convolutional Autoencoder with Temporal Consistency

RYOTO KATO[†1]　SHUNSUKE OSHIMA[†1]

Abstract: Two-dimensional convolutional autoencoders (2D-CAEs), which are commonly used for anomaly detection in manufacturing processes, are highly effective at capturing spatial information but have limited capability in modeling temporal information. As a result, a standalone 2D-CAE cannot adequately capture temporal consistency in time-series data. In this study, we propose a preprocessing-level approach that embeds temporal information into images by introducing a progress bar representation. Furthermore, we investigate how different types of preprocessing affect temporal consistency in anomaly detection. Experimental results show that the proposed combination of progress-bar-based preprocessing and a 2D convolutional autoencoder is effective not only in detecting shape-related anomalies but also in identifying anomalies in occurrence timing.

Keywords: Video Recognition and Understanding, Anomaly Detection, Machine Learning, Autoencoder

1. はじめに

現在の製造業界において，部品を製造する機械類は稀に不良品を製造する．現在は人の目視により監視し，異常が発生すれば人の手により異常を除去し，製造装置を停止・復旧する．このように，人への依存度が高く，人的コストや材料など生産コストを削減する目的から，画像処理技術や深層学習を用いた異常検知の研究が広く行われている[1].

産業現場における異常検知手法の一つとして，オートエンコーダ[2]が用いられる．オートエンコーダは構成がシンプル，かつ軽量であり，また入力データの再構成に基づく手法であるため，学習に異常データを必要としない教師なし学習である[1]．そのため，異常の発生が稀で異常データを十分に得られない産業現場に適しており，画像を用いた異常検知ではオートエンコーダの一種である２Ｄ畳み込みオートエンコーダ（Two-Dimensional Convolutional Auto Encode，以下 2D-CAE と表記）が採用される．

しかし，2D-CAE には静止画像など２次元に強い反面，時間軸に対する異常検知が弱いという欠点がある．たとえば，図 1 に示すように製造途中のパーツが機器から脱落し，あるいは材料がうまく送られなかった場合に，製造装置の初期状態（画像フレームＡ）と同じ画像フレームＢが一成形の中で出現することがある（以下，動画中の１枚または連続した画像のことを単に**フレーム**と表記）．このような状況であっても，1 枚のフレームのみを見た場合，2D-CAE では正常と判定する．他にも，装置異常のため引っ掛かりがある動作をしている製造装置において，機器が短時間や小刻みに停止した場合も同じフレームが連続するが，２D-CAE は正常と判定してしまう．

本研究では，あるフレームが，本来出現するべき t_1 の時刻にあるかどうかを示すことを**時系列的整合性**と呼ぶ．また，ある検出モデルが時系列的整合性を検出できるとは，本来出現すべき時刻 t_1 以外（図中の t_2）に，同じフレームが出現した場合に異常と判断できることをいう．つまり，あるフレームを単体で見て，正常か否かを判定する通常の 2D-CAE では，時系列的整合性を捉えることができず，先の異常をうまく検知できない．

1 熊本高等専門学校
National Institute of Technology, Kumamoto College

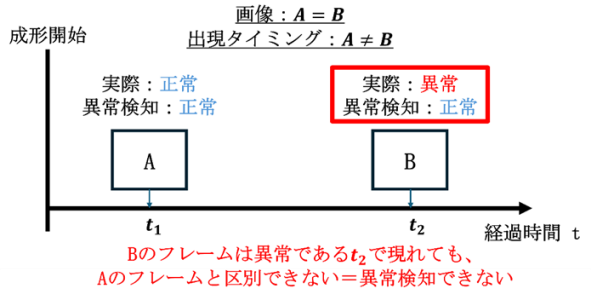


図 1　2D-CAE では捉えられない異常例

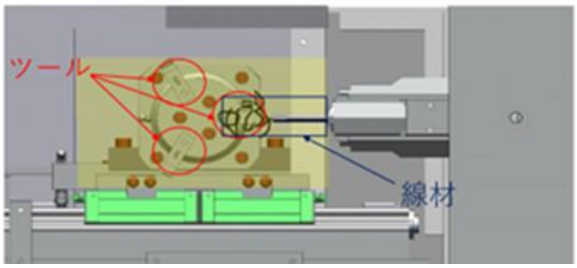


図 2　ばね成形機の状態異常例

表 1　ばね製造装置と画像サンプリングの主な機器

1．ばね成形機	旭精機工業（株）製 ばね機械 T シリーズ　T2
2．信号仲介機	（株）安川電機製 MP コントローラ
3．カメラ	オムロンセンテック（株）製 STC-MCS43POE

空間情報に時間情報を含めて異常検知する手法として，ConvLSTM（Convolutional Long Short-Term Memory）[3]や 3D 畳み込みオートエンコーダ（以下 3D-CAE と表記）[4]がある．しかし，これらの手法は時間軸方向まで含めたモデルの学習が必要であり，入力データの次元数が大きくなるデメリットがある．このような学習コストや使用メモリの大きいモデルを，現場の小型製造装置 1 つ 1 つに搭載するのは現実的ではない. 2D-CAE を用いて時系列データを扱った研究として文献[5]があるが，２次元のうち 1 次元を時間軸に使用しており，扱えるデータはセンサの数値など 1 次元データに限る．

そこで，本研究では，2D-CAE を基本として，前処理を施した画像を学習させることで，時系列的整合性を捉えることができる検出モデルを提案する．

本研究で対象とする製造装置は，ばね成形機である．ばねを成形する工程では，図 2 に示すように，線材やツール等の曲がり・欠けにより，線材が想定外の方向に曲がるなどし，線材同士が絡み合い製品形状が崩れるなどの異常が発生するが，本研究では，ほんのわずかな成形歪み（以下，状態異常と表記）を検出できる，精度の高い異常検知を目指すこととした．

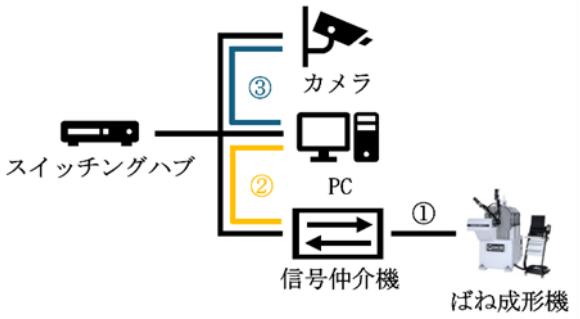


図 3　サンプリングのシステム構成

2.　ばね成形ごとのサンプリング装置の製作

本研究では，まず初めに，ばね成形機の正常動作の様子を撮影した「正常データ」を用意する必要がある．そこで，ばね成形機を撮影するカメラを近くに設置し撮影した．カメラは途切れのない連続した画像を撮影しており，これを，ばね 1 個の成形ごとに連続フレームとして切り出す必要がある．そこで，1 成形を 1 つの連続フレームとして自動サンプリングする装置を作成した．

サンプリング装置に使用する 3 つの主な機器を表 1 に示す．「2．信号仲介機」は，製造装置（ばね成形機）の状態を外部からモニターするための装置である．

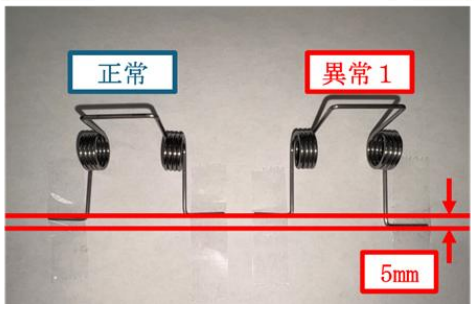
サンプリングのシステム構成図を図 3 に示す．機器間の通信は，①ばね成形機―信号仲介機，②PC―信号仲介機，③PC―カメラの 3 つである．①の通信は，製造装置に依存しており，装置ごとに仕様が異なるため，ここでは詳細は省略する．②の通信は，Ethernet を用いており，ソケットと拡張 MEMOBUS プロトコルに基づいている．この MEMOBUS プロトコルとは，株式会社安川電機などの産業用機器の通信方式として用いられている，デファクトスタンダードの MODBUS プロトコルをベースにした機能限定版である[6]．ばね成形機は，現在の装置の状態を信号仲介機の内部のレジスタ値として保存している．そこで，画像保存用の PC は，信号仲介機に保存されている「成形開始」を示すレジスタを連続して読みながら，「成形開始」の合図と同時に，1 成形の連続フレームを記録するようにした．③は連続撮影したカメラ画像を PC に送信する通信である．カメラの仕様を表 2 に示す[7]．カメラが GigE（Gigabit Ethernet）仕様のため，Ethernet で接続している．カメラへの接続および画像取得・保存は，カメラメーカー提供の専用ライブラリ StApi を用いて Python で実装する．カメラの解像度は，有効画素数の RGB 720×540 ピクセル，そして通信や PC の速度を考慮し，30 fps とした．

最後に，これらの機器を制御するプログラムを Python で記述し PC で実行した．これにより，ばね 1 成形ごとのフレームデータを連続してサンプリングすることに成功した．

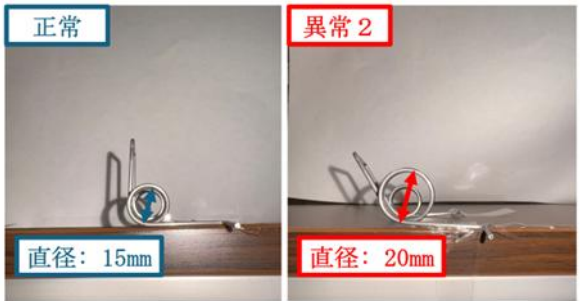
ばね成形機は各種のばねを製造することができる．本研究では，ばねの形状が複雑なダブルトーションばねを対象

表 2　カメラの仕様

型番	STC-MCS43POE
カメラタイプ	エリアカメラ
解像度	0.4MP
カラー/モノクロ	カラー
有効画素数	720(H) x 540(V)
フレームレート	282.8fps
インタフェース	PoE:IEEE802.3af CLASS2(1000BASE-T)
プロトコル	GigE Vision 2.1
電源	Power Over Ethernet (IEEE802.af 準拠) 外部電源:(+10.8～+26.4 Vdc)
消費電力	PoE: 4.7W 以下 外部電源: 3.7W 以下
動作温度	(周辺環境温度) 下限: 0℃ 上限: +39℃
保存温度	(周辺環境温度) 下限: -20℃, 上限: +70℃



(1)　形状異常 1 ：足の長さが 5 ㎜異なる



(2)　形状異常 2 ：コイル径が 5 ㎜異なる

図 4　2 種類の形状異常

として検証を行った．まず，正常なばね 85 個分の成形データを用意した．1 つの製品の成形は 13.8 秒であり，これは約 415 フレーム(85 個分で約 35,275 フレーム)に相当する．これらの正常データのうち，70 個分の成形を撮影したフレーム画像を検出モデルの学習用，残りの 15 個をテスト用とした．

また，材料の形状異常を 2D-CAE 単体で検知できるか否かを評価するため，製品自体の形状が異常な 2 種類の**形状異常データ**を用意した．

具体的な形状異常について説明する．図 4(1)は，正常なばねと比べて片方の足が 5 ㎜長いもの（以下，**形状異常 1** と表記），(2)は，ばねのコイル径の片方が 5 ㎜大きいもの（以下，**形状異常 2** と表記）とした．図 4 から分かるように，線材の太さが約 1 ㎜と細く，成形過程で線材の先端がわずかにずれたり，ばね自体が振動したりすることがあり，通常の画像差分を用いた方法では，異常をうまく検知することができない．そこで 2D-CAE による複数の正常画像の学習により，ばねの成形ごとの微妙な揺れを含めた「正常」を学習することにより，異常を判断できると考えた．なお，この形状異常 1，形状異常 2 は，線材が絡む図 2 の異常と比べて形状の変化がわずかである．つまり，形状異常 1 や形状異常 2 をうまく検知できれば，図 2 のような線材が絡むような大きな異常は当然検知できる．

次に，時系列的整合性の有無を確認するため，正常データの成形工程のタイミングをずらした，**時間異常データ**を用意した．具体的には，正常データに対して製造装置が動作する直前の 30 フレーム(1 秒)を追加したものを用いる．機械の何らかの不具合により，動作が 1 秒遅れて開始した想定である．製品 1 個分の成形データは正常データの 415 フレームに 30 フレームを追加した 445 フレームとなる．追加した 30 フレームおよび正常データ 415 フレームは 2D-CAE から見るとすべて正常なデータであり，この異常を検知出来れば時系列的整合性を有するかを確認できる．

今回は，上記の 2 種類の形状異常，および 1 種類の時間異常の計 3 種類について，それぞれ 15 個ずつ用意した．

3. 異常検知手法の検討

オートエンコーダは，異常検知の分野で広く使用されているモデルである．そこで，本研究では，異常検知手法として 2D-CAE を用いて，ばねのような細い線材について異常検知できるか検証する．次に，元画像に時間情報を埋め込む前処理の加工をし，2D-CAE の時系列的整合性の有無を検証する．

3.1 オートエンコーダ

オートエンコーダとはエンコーダ部とデコーダ部の 2 つの機構から構成される．まず，入力したデータをエンコーダと呼ばれる部分で特徴量抽出・次元削減を行うことで低次元の潜在ベクトルに圧縮し情報量を小さくする．その後，デコーダと呼ばれる部分で，圧縮した潜在ベクトルを，情報量を元の入力データと同じ大きさまで復元する．オートエンコーダは，この入力データと出力データの差，すなわち再構成誤差が小さくなるように学習する．

オートエンコーダの特徴の 1 つは，教師なし学習である

ことである．つまり，異常データを得ることが難しく，また未知の異常が発生するような生産現場の異常検知に適しているといえる．

オートエンコーダによる異常検知手法を説明する．まず，正常データのみを学習させたモデルに対して，正常データを入力したときは元データを正しく復元することができ，元データと復元データの差，つまり再構成誤差は小さくなる．一方，学習していないデータ，つまり装置の異常な状態を入力すると，入力画像を正しく復元することができず，再構成誤差が大きくなる．この再構成誤差の大小により，異常の有無を検知することができる．

再構成誤差の計算には，一般的に MSE（Mean Squared Error）と呼ばれる平均二乗誤差や SSIM（Structural Similarity Index）と呼ばれる構造的類似度指数の手法が用いられる．本研究では，リアルタイム検知の観点から，計算が単純で高速な MSE を採用した．

3.2 ２Ｄ畳み込みオートエンコーダ（2D-CAE）

2D-CAE はオートエンコーダの一種であり，２次元の画像データに特化したモデル構造となる．通常のオートエンコーダが全結合層（Fully Connected Layer）で構成されるのに対し，2D-CAE は畳み込みニューラルネットワーク CNN（Convolutional Neural Network）の畳み込み層（Convolutional Layer）とプーリング層（Pooling Layer）を利用する．そのため，入力データが画像データの場合において，オートエンコーダでは，画像を一次元形状のベクトルにフラット化して扱うのに対し，2D-CAE では画像をそのまま二次元形状で扱うため，あるピクセルに近接した上下左右の情報を失わない．また，3D-CAE や ConvLSTM と比較して，モデルの構造がシンプルのため，高速に動作するという大きなメリットがあるが，時系列情報は扱うことができない．

本研究では，ばね成型機などの製造装置に設置できる Jetson などのワンボードコンピュータでの運用を想定しており，性能が低いコンピュータでも高速動作が期待できる 2D-CAE に注目した．

3.3 3.3 学習とテストに使用するフレームのリサイズ

本研究では，１）時間情報を使用しなかった場合（**時間なし**と表記），２）時間情報を文字として埋め込んだ場合（**時間の文字埋め込み**と表記），３）時間情報をプログレスバーとして埋め込んだ場合（**時間のプログレスバー埋め込み**と表記），の３種類について評価・検討した．

１）～３）の各評価では，それぞれ専用の学習用データを用いてモデルを学習し，テストを行っている．

各フレームのサンプリング画像は 540×720 ピクセル RGB 256 階調である．そこで，オートエンコーダに適した形とするため画像サイズを落とす．まず，フレームデータをリサイズとグレースケール化により（108，144，1）の形状にする．グレースケール値は，0～255 を 0～1.0 に正規化した．

図 5　経過時間を文字として画像に描画したもの

3.3.1 時間なし

先のリサイズのみを行った 70 個×415 フレームのすべての画像について，1 つ 1 つを独立した静止画としてオートエンコーダで学習する．確率的勾配降下法により学習する画像はランダムに選択されるため，時間情報を一切含まないモデルが構築されることを想定している．この構築したモデルで，2D-CAE が繊細な線材加工の異常を検知できるかテストする．

3.3.2 時間の文字埋め込み

前処理後のフレームに対して，成形開始を 0 として，経過した時間情報を文字で埋め込む方法を提案する．図 5 に埋め込み後の画像を示す．今回は，リサイズ後の画像の下部に 8 ピクセル分を新たに追加し，その範囲に OpenCV ライブラリを用いて文字を埋め込む．具体的には，フォント：白，背景：黒，フォント種類：FONT_HERSHEY_SIMPLEX，フォントサイズ：0.25，太さ：1 として描画した．これによりデータ形状は（116，144，1）となる．表示する時間の単位は 0.1 ミリ秒であり，図 5 の 100284 は 10.0284 秒を意味する．

つまり，製造装置自体のフレーム画像はまったく同じであっても，時間は常に変化しており，ばねの成形画像と経過時間の両方をモデルが学習することができれば，モデルに時系列的整合性を持たせることができる．

3.3.3 時間のプログレスバー埋め込み

先の前処理後のフレームに対して，成形の開始から経過した時間をプログレスバー（工程の進捗）の形で埋め込む方法を提案する．

図 6 はプログレスバーを追加した後の画像である．文字情報の埋め込みと同様，前処理後の画像の下に 8 ピクセル分のエリアを新たに追加した．プログレスバーの追加処理を行った後のデータ形状は（116，144，1）である．具体的には，縦 8 ピクセルの領域に，グレースケール値 1.0，横方向 1 ピクセルの画像を，時間の進行とともに左から順に書き込むことで実現する．今回は，ばねの成形時間を最大で 20 秒と想定しており，20 [秒]×30[フレーム/秒] / 144[ピ

図 6　経過時間によるプログレスバーの変化

クセル]＝約 4.2[フレーム/ピクセル]であるため，4.2 フレームごとにプログレスバーを 1 ピクセル書き込むこととした．

プログレスバーの効果として，先の時間の文字埋め込みと同様，ばね成形のフレーム画像とプログレスバーを両方同時に 2D-CAE のモデルが学習することで，モデルに時系列的整合性を持たせることができると考えた．

また，文字埋め込みと比較して，時間軸に対するプログレスバーの変化量が常に一定であり，数フレームごとに 1 ピクセル増加するため，文字と比べてフレームのずれによる再構成誤差が少なくなることが予想でき，時間の揺れに対してロバストな学習モデルになると考えた．つまり，プログレスバーを 1 つ進めるために要するフレーム数は，時間のゆれを許容するパラメータとして機能する．

4.　実験方法と前準備

4.1　本研究で行う実験の種類と実験方法

本研究では，次の 2 つの実験を行う．

実験 1．時間なし，時間の文字埋め込み，時間のプログレスバー埋め込みの性能評価

時間なし，時間の文字埋め込み，時間のプログレスバー埋め込み，の 3 つの学習データを用いて，それぞれで学習モデルを構築する．次に，テストデータにおいて MSE の比較を行い，形状異常 1，形状異常 2 の各異常を検知できるか検証する．さらに，学習モデルごとに，時間異常の検知性能，すなわち時系列的整合性を捉えるか否かを検証する．

実験 2．時間のプログレスバー埋め込みにおける時系列的整合性の評価

時間異常において，プログレスバーが 1 つ進むのに要するフレーム数を 5 フレームから 30 フレームまでの範囲で変化させ，結果を比較する．実験 1 で，約 4.2 フレームでプログレスバーを 1 ピクセル進行したため，ここでは 5 フレーム刻みで 5～30 フレームの範囲で実験した．

4.2　本研究で用いる 2D-CAE のモデル構造と学習条件

本研究で使用する 2D-CAE のモデルと各層ごとのデータ形状を図 7 に示す．入力画像に対して，エンコーダ部で畳み込み層と最大プーリング層による特徴量抽出削減を 2 回

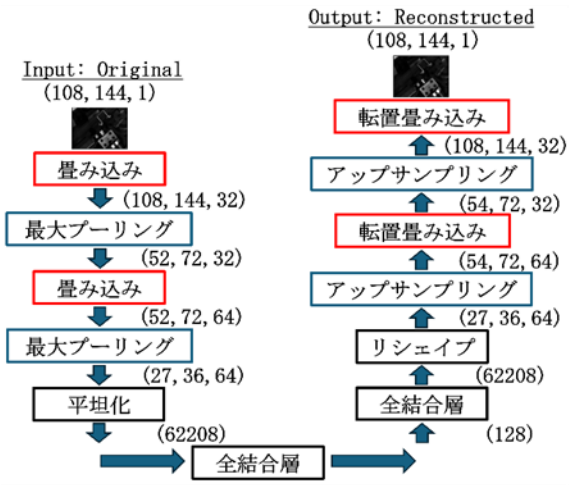


図 7　本研究で使用するオートエンコーダモデルと各層ごとのデータ形状

繰り返し，その後平坦化と全結合層により，128 次元の潜在ベクトルまで圧縮する．そして，デコーダ部で圧縮した潜在ベクトルから復元を行い，画像を再構成する．損失関数には MSE を用い，入力画像と再構成した画像の誤差が小さくなるよう学習を行う．

機械学習で使用した PC の仕様を表 3 に，使用した主なライブラリを表 4 に示す．今回，OS として Windows を搭載している PC 上に WSL（Windows Subsystem for Linux）と

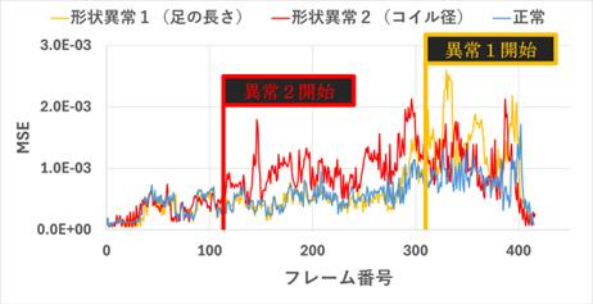
表 3　機械学習で使用した PC の仕様

ホスト OS	Windows 11 Home
WSL バージョン	WSL 2.5.9.0
WSL ディストリビューション	Ubuntu 24.04
Docker バージョン	29.1.3
コンテナベースイメージ	nvidia/cuda:12.5.1-devel-ubuntu24.04
CUDA	12.5.1
CuDNN	9.3.0
GPU	NVIDIA GeForce RTX4070
GPU ドライバ	581.80
CPU	Intel Core i7-13700
RAM	32GB
言語	Python3.12

表 4　使用した主なライブラリ

tensorflow	2.19.0
keras	3.10.0
numpy	2.1.3
pandas	2.3.3
opencv	4.12.0

(1)　時間なし



(2)　時間の文字埋め込み

(3)　時間のプログレスバー埋め込み

図 8　３つの学習モデルにおける
形状異常１，形状異常２の MSE 変化

Docker を用いてコンテナ環境を構築し，そのコンテナ内に NVIDIA が提供する CUDA および CuDNN を導入することで，GPU アクセラレーションを利用した機械学習用の環境を構築した．使用した主なライブラリは 5 つである．Tensorflow，Keras は機械学習，Numpy は数値計算・データ処理, Pandas はデータ分析, OpenCV は画像処理に使用した．

70 個の正常な成形データ約 29,900 フレームを用い，50 エポック，バッチサイズを 64 とし，Tensorflow, Keras を用いてモデルの学習を行った．

5.　結果・考察

5.1　実験１：時間なし，時間の文字埋め込み，時間のプログレスバー埋め込みの実験結果

時間なし，時間の文字埋め込み，時間のプログレスバー埋め込みの３つの学習データを使って学習した各モデルにおいて，足の長さが異なる形状異常１，コイル径が異なる形状異常２，時間がずれた時間異常の３つの異常を判定した．

図 8 は正常なフレーム（青），および形状異常１（黄），形状異常２（赤）について，異常値（MSE）を比較したグラフである．学習モデルが異なる場合，正常なフレームの MSE も異なるため，すべて掲載した．

まず，正常フレームにおいて，全体的に MSE が小さいことから，学習モデルは学習したすべてのフレームにおいて，再構成能力がある適切なモデルであることがわかる．わずか 128 次元のベクトルに圧縮したことを考えると，改めてオートエンコーダの再構成能力の高さが伺える．

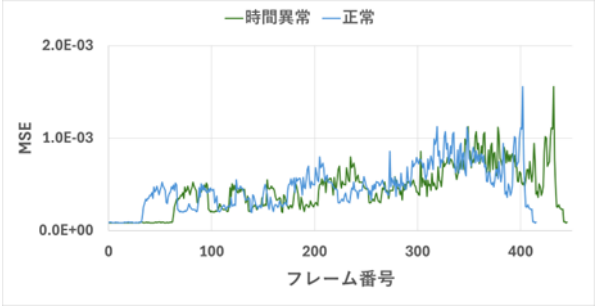
次に形状異常について確認した．まず，撮影された映像を確認したところ，形状異常１は 310 フレーム目以降で，また，形状異常２は 115 フレーム目以降で，正常とは外れたばねの成形を開始していた．図 8(1)(2)(3)の各グラフを見ると，いずれの学習モデルにおいても，成形の異常が開始されたフレームから MSE が増加していることが確認できる．意外だったのは，図 8(1)時間なしの学習モデルで，時系列データを学習していないにもかかわらず形状異常を判定できており，時系列的整合性が無くても 2D-CAE の再構成能力は高い．しかしこれは，今回学習に使用したばねの成形過程において，たまたま同じフレームが無かったことが影響していると考える．

図 9 は正常フレーム（青）と時間異常の MSE（緑）を比較したグラフである．

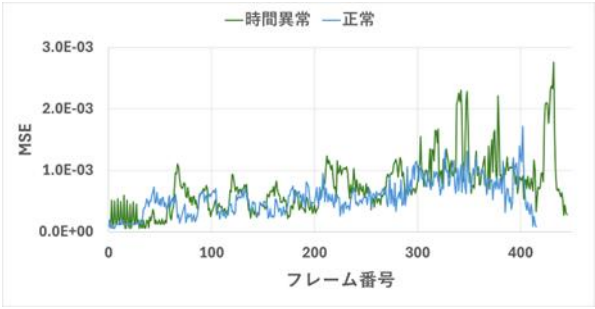
時間異常は，正常フレームの時間を約 1 秒（30 フレーム）ずらしたものである．したがって，図 9(1)の時間なし学習モデルでは，同一の MSE が 30 フレームずれて出力される．結果からわかるように，時間異常の MSE が正常フレームのそれと比べて同程度であり，つまり 2D-CAE 単体では時系列的整合性を捉えることはできない．また，図 9(2)の時間の文字埋め込みについても，正常フレームの MSE の高低が不安定な結果となった．これは，たとえば 33333 と 88888 は時刻は異なるが，画像としては似ている．時間情報が文字の数ドット分の変化しかなく，再構成誤差が小さくなったためと考える．文字フォントサイズを大きくすれば，また違った結果となる可能性はある．

最後の図 9(3)時間のプログレスバー埋め込みのグラフでは，時間異常の MSE が，正常データのそれと比べて高くなる結果が得られた．この結果から，プログレスバーを追加する前処理により，時系列的整合性を持った学習モデルを 2D-CAE で構築できることがわかった. 今回の結果はプログレスバーの縦幅が 8 ピクセル, 約 4.2 フレームごとに 1 ピクセル白色のバーが進行しており，少なくとも 30 フレームの成形タイミングのずれを時間異常として十分に検知できている．

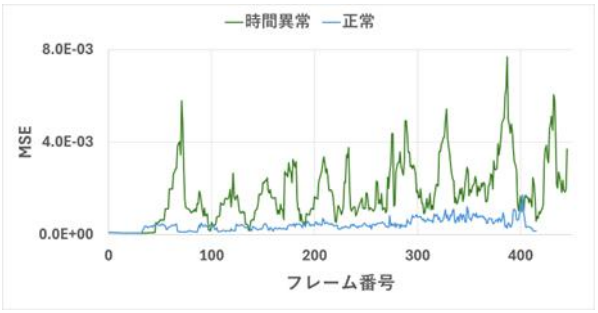
ところで，図 8，図 9 では，MSE の変化が大きく，異常の有無を判断する閾値を一意に設定することができない．そこで，フレームごとの MSE を積算することとした．結果を図 10 に示す．



(1)　時間なし



(2)　時間の文字埋め込み



(3)　時間のプログレスバー埋め込み

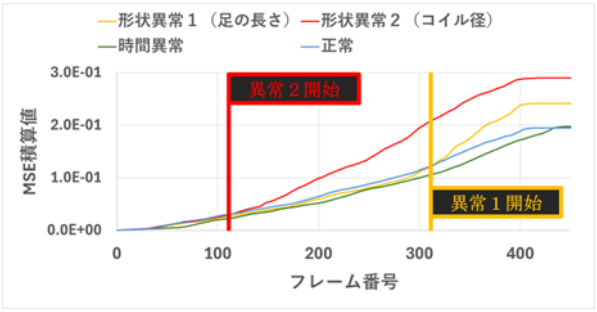
図 9　3つの学習モデルにおける時間異常の MSE 変化

図 10(1) 時間なし, 図 10(2) 時間の文字埋め込み, 図 10(3) 時間のプログレスバー埋め込み，の各モデルにおいて，形状異常１，形状異常２，時間異常のそれぞれの結果を見ると，異常が開始したフレームから，積算値が大きくなることが分かる． 特に，図 10(3)の時間のプログレスバー埋め込みでは，正常データのグラフとの違いは明確である．このことから，閾値としてたとえば正常と比べた MSE 積算値が 10%増加した場合は異常と判断することができる. 実験の結果, 2D-CAE 単体では時系列的整合性を捉えられないのに対し，プログレスバーを追加する前処理で時系列的整合性を十分に捉えることができた.

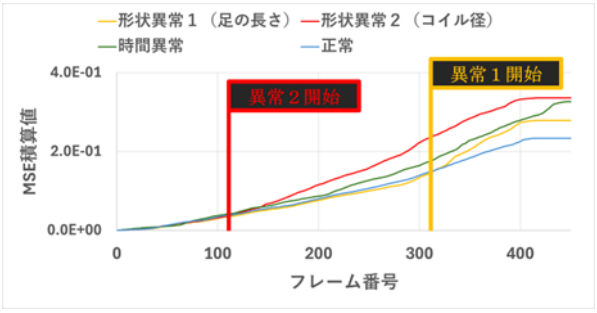
5.2　実験２：時間のプログレスバー埋め込みにおける時系列的整合性の性能評価

表 5 は時間異常において，異常な状態のフレーム数を 5～30 まで変化させた場合の MSE の変化率を測定したものである．1 成形あたりの平均値を算出した.

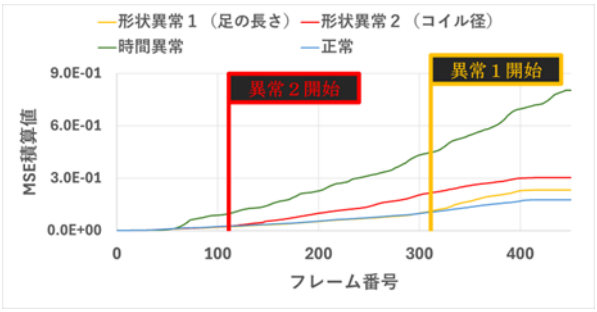
この結果より，フレーム数のずれが 10 以下のものは正常データと MSE の差があまりないことが分かる．一方で，15 フ



(1)　時間なし



(2)　時間の文字埋め込み



(3)　時間のプログレスバー埋め込み

図 10　成形開始からの全フレーム MSE 積算値

表 5　時間異常の検知におけるフレーム数と MSE 変化率（1 成形における平均値）

フレーム数	5	10	15	20	25	30
変化率（倍）	0.97	1.03	1.45	2.03	2.75	3.64

フレーム以上，実時間で 0.5 秒のずれは，正常データとの差が大きく，またそれ以降フレームのずれが大きいほど MSE も大きくなる．この結果から，閾値を 45%以上に設定すれば 0.5 秒のタイミングのずれを検知することができる.

6.　実環境への適用

Jetson は GPU を搭載したワンボードマイコンであり，産業現場の異常検知エッジデバイスとして広く利用されている．そこで，製造装置への Jetson の実装を想定し，学習総時間や 1 エポックあたりの平均学習時間，平均推論時間を計測した．本研究で用いた 4.2 節の機械学習用 PC と併せて

表 6　Jetson の詳細な仕様

機器	Jetson Orin NX 16GB
CPU	8-core Arm Cortex-A78AE v8.2 64-bit CPU 2MB L2 + 4MB L3
GPU	1024-core NVIDIA Ampere architecture GPU with 32 Tensor Cores
メモリ	16GB
SDK	Jetpack 6.2.1

表 7 機械学習用 PC と Jetson の性能比較

	機械学習用 PC	Jetson	Jetson／PC（倍）
総学習時間 [s]	584	4728	8.1 倍
推論 [ms]	29.56	64.33	2.2 倍

比較する．

まず, 本研究で用いた Jetson の仕様を表 6 に示す[8]. GPU は 1024 個の計算コアと 32 個の Tensor コアを持ち，AI 処理の性能を示す指標は 157 TOPS である．モデルの学習における各種条件は 4.2 節と同等とした．また，推論は 100 回行い平均時間を算出した．表 7 に，機械学習用 PC と Jetson における学習総時間と平均推論時間を示す．表より，Jetson の総学習時間は 4,728 秒, PC に対して 8.1 倍の時間がかかった．また，推論時間は PC と比較して 2.2 倍の時間を要し，64.33 ミリ秒であった．これらの結果より，製造装置において，新たな製品の異常検知をする場合, 約 1.3 時間で学習モデルを構築でき，また，推論が 64.33 ミリ秒と高速のため，少なくとも 0.1 秒間に 1 回異常検知ができる. これらのことから，提案手法は実環境へそのまま適用できる手法であるといえる．

7. まとめ

本研究では，時系列の画像データの異常検知について，モデルを複雑にするのではなく画像に時間情報を埋め込む手法により，時系列的整合性を捉える手法を提案した．具体的には，従来の 2D-CAE の学習に，プログレスバーを埋め込んだ画像を用いる．その結果，オートエンコーダ単体では出現タイミングのずれを捉えることができなかったが, プログレスバーを用いた手法では，時系列的整合性を捉えることができた．また，2D-CAE の軽量性の利点を活かし，リアルタイム性が必要な現場において, Jetson レベルの装置でリアルタイム異常検知できることを示した．

今後の展望として，他の手法，たとえば ConvLSTM や 3D-CAE と比較した場合の学習コストや推論速度, 検知精度の評価を検討している．また，実際の製造装置に組み込んだ場合の運用面での課題，特に環境光の影響について検討したい．

謝辞

本研究は，旭精機工業株式会社との共同研究として実施しました．遂行に際し，ばね機やロボットアームをご提供いただきました旭精機工業株式会社に厚く御礼申し上げます．また，本研究に対する質問や助言をいただきました技術・教育センターの岩本舞氏と中村佑介氏，および同じ研究室の石川君，武藤君，山下君に厚く御礼申し上げます．

参考文献

[1] Y. Cui，Z. Liu，and S. Lian，A Survey on Unsupervised Anomaly Detection Algorithms for Industrial Images，IEEE Access，vol. 4，2016.
[2] G. E. Hinton and R. R. Salakhutdinov，Reducing the Dimensionality of Data with Neural Networks，Science，vol. 313，pp. 504-507，2006.
[3] X. Shi，Z. Chen，H. Wang，D. -Y. Yeung，W. -K. Wong，and W. -C. Woo，Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, arXiv:1506.04214v2, 2015.
[4] Y. Zhao，B. Deng，C. Shen，Y. Liu，H. Lu，and X. -S. Hua，Spatio-Temporal AutoEncoder for Video Anomaly Detection，ACM, In Proceedings of the 25th ACM International Conference on Multimedia，pp. 1933-1941，2017.
[5] J. Kang，M. Kim，J. Park，and S. Park，Time-Series to Image-Transformed Adversarial Autoencoder for Anomaly Detection，IEEE Access，vol. 12，pp. 119671-119684，2024.
[6] MEMOBUS 通信と MODBUS 通信の違いを教えて下さい．| 株式会社安川電機（オンライン），入手先〈https://www.e-mechatronics.com/support/faq/detail.html?num=883&parent0=68&cate=68〉（参照 2026-01-16）.
[7] STC-MCS43POE | オムロンセンテック株式会社（オンライン），入手先〈https://sentech.co.jp/products/stc-mcs43poe〉（参照 2026-01-15）
[8] NVIDIA Jetson Orin（オンライン），入手先〈https://www.nvidia.com/en-au/autonomous-machines/embedded-systems/jetson-orin/〉（参照 2026-02-06）