**２０２３年度　卒　業　論　文**

卒業論文題目

|  |
| --- |
| 自然環境映像を対象とした画像検索における  画像特徴ベクトルのインデクシング手法 |

　指 導 教 員　　　　　　　鷹野　孝典　教授

神奈川工科大学　情報工学科

　　　学 籍 番 号　　　　　　　2021065

　　　学 生 氏 名　　　　　　　瀬尾　幸斗

　提 出 日　2023年　12月　26日　　　印

　受 理 日　2023年　12月　26日　情報工学科長　　印

# **論文要旨**

地球温暖化により海面の上昇や砂漠化，生態系の変化など，地球規模での環境問題が発生している．そのため，自然環境をモニタリングしたアーカイブは，環境問題の把握や対策のために必要であるが，世界各地で記録される自然環境のアーカイブはデータ量が膨大なため，アーカイブの検索には高い計算コストを必要とする．先行研究では，画像を入力としてニューラルネットワークにより抽出した低次元の画像特徴ベクトルを用いた画像シーン検索手法を提案してきたが，中間層によって得られる特徴ベクトルの性質にばらつきが生じるため，適切な中間層を選び，特徴ベクトルを抽出する処理に課題があった．そのため本研究では，オートエンコーダによる画像特徴抽出および距離学習ニューラルネットワークを適用した画像特徴のインデクシング手法を提案する．提案手法の特徴として，抽出した画像特徴の再利用性を保ちつつ，インデクシングされた距離の最適化された特徴ベクトルを検索に応用することで，画像特徴を検索可能にする点である．

実験では，国土地理院の航空写真を用いて，複数のニューラルネットワークモデルの再現率や精度を比較し，提案手法の実現可能性を検証する．

目次

[**論文要旨** 2](#_Toc154423075)

[**1.** **まえがき** 1](#_Toc154423076)

[1.1 はじめに 1](#_Toc154423077)

[1.2 本論文の構成 2](#_Toc154423078)

[**2.** **関連研究** 3](#_Toc154423079)

[2.1 オートエンコーダによる特徴抽出 3](#_Toc154423080)

[2.2 距離学習を組み込んだオートエンコーダによる特徴抽出 3](#_Toc154423081)

[**3.** **提案方式** 4](#_Toc154423082)

[3.1 提案手法の概要 4](#_Toc154423083)

[3.2 提案手法の手順 4](#_Toc154423084)

[3.3 オートエンコーダを用いる利点 6](#_Toc154423085)

[3.4 距離学習ニューラルネットワークを用いる利点 6](#_Toc154423086)

[3.5 代表特徴ベクトルのインデクシングによる利点 6](#_Toc154423087)

[**4.** **実験システム** 7](#_Toc154423088)

[4.1 実験システムの概要 7](#_Toc154423089)

[4.2 実装環境 7](#_Toc154423090)

[4.2.1 環境の概要 7](#_Toc154423091)

[4.2.2 プログラミング言語 9](#_Toc154423092)

[4.2.3 Pythonライブラリ 9](#_Toc154423093)

[4.3 使用モデルについて 9](#_Toc154423094)

[4.3.1 オートエンコーダ 9](#_Toc154423095)

[4.3.2 距離学習ニューラルネットワーク 11](#_Toc154423096)

[4.4 画像集合の読み込みと訓練用データ，検証用データへの分割 12](#_Toc154423097)

[4.5 オートエンコーダのネットワーク構築と画像集合による事前学習 13](#_Toc154423098)

[4.6 特徴ベクトルの抽出 14](#_Toc154423099)

[4.7 距離学習ニューラルネットワークによる距離学習 14](#_Toc154423100)

[4.8 クラスタリングによる類似画像分類 15](#_Toc154423101)

[4.9 コサイン類似度による特徴ベクトルの類似度計算 15](#_Toc154423102)

[4.10 Webサーバの構築 15](#_Toc154423103)

[4.11 Raspberry Piでのモニタリング機能の構築 17](#_Toc154423104)

[**5.** **実験** 18](#_Toc154423105)

[5.1 実験目的 18](#_Toc154423106)

[5.2 実験環境 18](#_Toc154423107)

[5.3 実験1 19](#_Toc154423108)

[5.3.1 実験目的 19](#_Toc154423109)

[5.3.2 実験方法 19](#_Toc154423110)

[5.3.3 実験結果 19](#_Toc154423111)

[5.3.4 考察 21](#_Toc154423112)

[5.4 実験2 22](#_Toc154423113)

[5.4.1 実験目的 22](#_Toc154423114)

[5.4.2 実験方法 22](#_Toc154423115)

[5.4.3 実験結果 22](#_Toc154423116)

[5.4.4 考察 24](#_Toc154423117)

[**6.** **むすび** 25](#_Toc154423118)

[謝辞 1](#_Toc154423119)

[**参考文献** 2](#_Toc154423120)

# **まえがき**

## はじめに

地球温暖化による気候の変化，海面の上昇，砂漠化，生態系の変化など，地球規模での自然環境問題の解決が緊急課題となっている．例として，国土交通省では革新的河川技術プロジェクトを実施しており，水位計や監視カメラ，ドローンといったIoT機器を用いた河川管理を一般の企業から公募している1)．そのため，自然環境を常時観測し，動画や画像などの視覚データとして記録しておくことで，自然現象の時系列的な把握や類似する自然現象の抽出などによるデータ解析が重要な役割を担う．しかし，世界各地で自然環境を対象とした観測動画データを記録することを想定した場合，データ量が非常に膨大となるため，データ分析に要する処理コストも非常に高くなることが想定される．我々はこれまで，このような課題を解決するために，画像を入力としてニューラルネットワークにより抽出した低次元の画像特徴ベクトルを用いた画像シーン検索手法を提案してきた2)．しかし，VGG畳み込みニューラルネットワークなどの中間層を多く持つニューラルネットワークを用いて特徴ベクトルを抽出する際に，中間層によって得られる特徴ベクトルの性質にばらつきが生じるため，適切な中間層を選び，特徴ベクトルを抽出する処理に課題があった．色や形状といった画像内の特徴を含む特徴ベクトルは分類に適しているとは限らず，また分類に適した特徴ベクトルは画像が本来持つ特徴を含んでいるとは限らないためである．

本研究では，先行研究の提案手法を拡張し，画像特徴を抽出するネットワークとしてオートエンコーダ，および距離学習ニューラルネットワークを適用した画像特徴の分類手法およびインデクシング手法を提案する．オートエンコーダを適用する理由は，ネットワーク構造が複雑ではなく実装しやすい点，エンコーダの出力として得られる画像特徴ベクトルの次元数を調節しやすい点などにある．また，画像特徴ベクトルをそのまま利用しても画像分類精度が高いとは限らないため，距離学習ニューラルネットワークを適用することで，正例，負例の教師データを用いた学習を実施する．これにより，画像分類精度を向上することが可能となる．提案手法では，オートエンコーダと距離学習ニューラルネットワークを連結した構成を持つモデルを構築することにより，オートエンコーダから抽出した画像が本来持つ特徴の再利用性を保ちつつ，オートエンコーダと距離学習ニューラルネットワークを組み合わせた柔軟なモデル構築を実現することができる．また，オートエンコーダで抽出した画像特徴を含むベクトルに対し，分類に適したベクトルをインデックスとして設定することで，抽出した画像特徴をベクトルの類似度で検索することができる．さらに，分類に適したベクトルにクラスタリングを適用し，重心ベクトルといった代表値をインデックスとして設定することで，クラスタとの類似度による検索が可能になる．

本研究の実験システムとして，複数地点で撮影した画像集合から提案手法を用いて画像特徴ベクトルを抽出し，クラスタリングアルゴリズムを用いてクラスタの代表値となるベクトル平均や重心ベクトルを算出することで，入力画像とクラスタとの類似度計算を実現するシステムを作成した．実験システムは，軽量なWebアプリケーションフレームワークであるFlaskを用いて実装しているため，インターネット上からアクセスすることで利用することができる．

実験では，国土地理院の航空写真データセット7)を用いて，複数のニューラルネットワークモデルの再現率といった精度を比較し，提案手法の実現可能性を検証する．実験で用いるニューラルネットワークモデルとして，オートエンコーダと距離学習ニューラルネットワークを組み合わせた6つのモデルを構築した．はじめに，訓練用の航空写真を用いて事前学習済みオートエンコーダを作成する．次に，事前学習済みオートエンコーダを用いて検証用の航空写真から画像特徴を抽出する．そして，画像特徴に距離ニューラルネットワークを適用し，ラベル間の距離が最適化された特徴ベクトルに変換する．以上で得られた画像特徴ベクトルや特徴ベクトルにクラスタリング手法の1つであるk-Meansを適用する．k-Meansでの分類結果から，分類やランキングに適したベクトルが算出できる．ニューラルネットワークモデルを，再現率や適合率といった精度から考察する．

## 本論文の構成

本論文は全6章で構成されている．本章では，研究背景や課題，先行研究の概要と本研究の概要について述べる．第2章では，本研究と関連する研究について述べる．第3章では，提案手法の仕組みについて述べる．第4章では，提案手法を用いて実装したシステムの詳細について述べる．第5章では，提案手法を評価するために実施した実験についての結論と考察を述べる．第6章では，結論と今後の展望について述べる．

# **関連研究**

本章では，オートエンコーダによる特徴抽出の研究や，オートエンコーダに距離学習を組み込んだモデルを使用した研究について述べる．

## オートエンコーダによる特徴抽出

オートエンコーダは，様々なデータの特徴抽出器として利用されている．文献3)は，花押検索を目的として，花押という筆者署名の機能を持つ記号から，畳み込みオートエンコーダを用いて字形特徴を抽出する手法を提案している．文献4)では，筆跡鑑定を目的として，字種ラベルという条件を付加した条件付きオートエンコーダを提案している．デコーダの画像生成部分のみに字種ラベルを付加して学習することで，エンコーダ部分は筆跡画像から字種ラベルに依存しない筆者固有の書き癖を特徴として抽出する． 文献5)では，不整脈のリアルタイム解析を目的として，IoT機器とオートエンコーダを用いた不整脈分類手法を提案しており，ECG波形から特徴を抽出している．

## 距離学習を組み込んだオートエンコーダによる特徴抽出

オートエンコーダに距離学習を組み合わせたモデルでの研究として，文献6)で変分オートエンコーダとトリプレット損失を組み合わせたニューラルネットワークを構築し，類似性の概念を抽出するための埋め込み表現を学習するオートエンコーダモデルの提案がされている．また，オートエンコーダとトリプレット損失を組み合わせたニューラルネットワークを構築し，ネットワークのトラフィックから攻撃や侵入を検知する手法7)の提案がされている．

# **提案方式**

本章では，本研究で提案する手法についての概要と手順を述べ，提案手法で用いる技術についての特徴を述べる．

## 提案手法の概要

提案手法では，オートエンコーダを用いて画像から圧縮された低次元の画像特徴を抽出し，距離学習ニューラルネットワークを適応することで，分類可能な特徴ベクトルに変換する．得られた特徴ベクトルに，分類アルゴリズムを適用し，得られたクラスタの平均や重心を画像特徴ベクトルに代表特徴ベクトルとしてインデクシングする．

## 提案手法の手順

手順としては以下のような流れになる．

1. 画像集合 を対象としてオートエンコーダを適用して学習させ，画像特徴が抽出可能な学習済みモデル()を作成する．
2. STEP-1で構築したのエンコーダの中間層の出力から，画像集合の画像特徴ベクトルを抽出する．ここで，画像特徴として，畳み込み層の出力である階のテンソルを用いることもできる．各は多次元ベクトルである．本研究では，この階のテンソルを画像特徴テンソルと呼ぶことにする．中間層からの画像特徴ベクトルの抽出を図3.1に，畳み込み層からの画像特徴テンソルの抽出を図 3.2に示す．



図3.1　中間層による画像特徴ベクトル抽出



図 3.2　畳み込み層による画像特徴テンソル抽出

1. STEP-2で抽出した画像特徴は画像分類精度が高いとは限らないため，距離学習ニューラルネットワーク()を適用し，正例，負例の教師データを用いた学習を実施する．を用いることで，画像特徴ベクトル または画像特徴テンソル をランキングや分類に適した特徴ベクトル に変換する． 図 3.3は距離学習ニューラルネットワークの構造を示しており，図中のEmbeddingは類似する特徴ベクトルの距離関係が最適化された特徴ベクトルである．の距離関係の損失計算は，正例ペアの内積をソフトマックス関数に適用し，予測値との差を計算している．この損失計算により，正例のペアは近く，他のペアは遠くするといった距離学習を実施している．図 3.4は，画像特徴の距離関係が最適化された例である．で入力層は，入力の画像特徴のデータ形式に合わせる．例えば，画像特徴ベクトル形式であれば全結合層，画像特徴テンソル形式であれば畳み込み層などを適用する．



図 3.3　距離学習ニューラルネットワークによる画像特徴の最適化



図 3.4　画像特徴の距離学習

1. STEP-3で得られた特徴ベクトルにクラスタリングを適用し，平均や重心といったクラスタの代表となる値を算出し，代表特徴ベクトルとしてクラスタ内の画像特徴ベクトルに対しインデックスとして設定する．代表特徴ベクトルのインデクシングの例について図 3.5に示す．



図 3.5　代表特徴ベクトルを用いたデータ集合のインデクシングの例

## オートエンコーダを用いる利点

画像特徴が抽出可能な学習済みオートエンコーダモデルを用いて，エンコーダの中間層または畳み込み層から，画像特徴ベクトルまたは画像特徴テンソルを抽出する．本研究でオートエンコーダを用いる理由として，エンコーダ出力で圧縮された低次元の画像特徴ベクトルを得られるという点や，抽出する画像特徴ベクトルの次元数を容易に調整することが可能であるという点，モデル構造が複雑ではなくモデルの拡張性があるという点から用いることにした．

## 距離学習ニューラルネットワークを用いる利点

距離学習ニューラルネットワークを用いて，画像特徴を類似する特徴ベクトルの距離関係が最適化された特徴ベクトルに変換する．距離学習における損失計算手法はいくつかあるが，本研究では，同じクラスのデータペアの類似度を計算したベクトルを用いた多クラス用の交差エントロピーによる確率分布ベースでの損失計算を実施している．クラスタリングアルゴリズムによる画像特徴の分類を行うために，画像特徴を分類やランキングに適したベクトルへと変換する必要があるため，距離学習ニューラルネットワークを用いることにした．

## 代表特徴ベクトルのインデクシングによる利点

距離学習ニューラルネットワークで変換した特徴ベクトルにクラスタリングアルゴリズムを適用し，分類された各クラスタのベクトル平均や重心ベクトルといった代表値を算出する．先行研究では，中間層から得られる特徴ベクトルにばらつきがという問題があったが，この代表値をオートエンコーダで抽出した画像特徴ベクトルに代表特徴ベクトルとしてインデクシングすることで，画像検索時にクラス毎による類似度比較を行うことが可能となる．

# **実験システム**

本章では，提案手法を用いた実験システムについて述べる．

## 実験システムの概要

実験システムでは，オートエンコーダによる画像特徴の抽出，距離学習ニューラルネットワークの適用による画像分類やランキングに適した特徴ベクトルへの変換，クラスタリング結果からの代表特徴ベクトルの算出，コサイン類似度を用いた代表特徴ベクトルとの類似度計算機能，FlaskによるWebサーバの作成をPythonで実装した．また，カメラによる映像記録を実施するプログラムの作成にC言語を用いて実装した．

## 実装環境

### 環境の概要

本研究では，UbuntuがインストールされたPC(Personal computer)と，WindowsがインストールされたPCの2台を用いている．また，映像記録デバイスにはRaspberry piを用いた．UbuntuがインストールされたPCについて，PC環境を表 4.1に，プログラミング言語などの開発，実験環境を表 4.2に，実装に使用したPython ライブラリを表 4.3に示す．また，WindowsがインストールされたPCについて，PC環境を表 4.4に，プログラミング言語などの開発，実験環境を表 4.5に，実装に使用したPythonライブラリを表 4.6に示す．さらに，Raspberry Piの環境を表 4.7に，プログラミング言語などの開発，実験環境を表 4.8に示す．Raspberry PiのC言語によるプログラム実装には，標準ライブラリを使用している．

表 4.1　Ubuntu環境

|  |  |
| --- | --- |
| 名称 | ハードウェア・OS |
| PC | CPU: Intel i9-12900KF  RAM: 32GB  GPU: GeForce RTX3090 |
| OS | Ubuntu22.04 |

表 4.2　ubuntu環境における開発・実験環境

|  |  |  |
| --- | --- | --- |
| 名称 | バージョン | 利用用途 |
| Python | 3.10.12 | プログラミング言語 |
| venv | 3.10.12 | 仮想環境構築 |
| Nvidia driver | 535.129.03 | GPUのドライバー |
| Cuda compilation tools | 11.5 | GPUによる処理 |

表 4.3　 ubuntu環境におけるPythonライブラリ

|  |  |  |
| --- | --- | --- |
| 名称 | バージョン | 利用用途 |
| Tensorflow | 2.15.0 | 機械学習 |
| Keras | 2.15.0 | 深層学習 |
| cuDNN | 8.9.5.29 | GPUアクセラレーション型プリミティブライブラリ |
| scikit-learn | 1.3.1 | 機械学習  データ分割 |
| matplotlib | 3.8.0 | グラフ描画 |
| numpy | 1.24. | 数値計算 |
| Flask | 3.0.0 | Webサーバの設置 |
| Opencv-python | 4.8.1.78 | 画像の読み込み，保存 |
| Pillow | 10.1.0 | 画像の読み込み |

表 4.4　windows環境

|  |  |
| --- | --- |
| 名称 | ハードウェア・OS |
| PC | CPU: Ryzen7 5800x  RAM: 32GB  GPU: GeForce RTX3070Ti |
| OS | Windows 11 |

表 4.5　 windows環境における開発・実験環境

|  |  |  |
| --- | --- | --- |
| 名称 | バージョン | 利用用途 |
| Python | 3.8.10 | プログラミング言語 |
| Anaconda | 23.3.1 | 仮想環境構築 |
| Visual Studio Code | 1.85.1 | 実験環境 |
| CUDA toolkit | 11.2.2 | GPUによる処理 |

表 4.6　 windows環境におけるPythonライブラリ

|  |  |  |
| --- | --- | --- |
| 名称 | バージョン | 利用用途 |
| tensorflow | 2.10.1 | 機械学習 |
| cudnn | 8.1.0.77 | GPUアクセラレーション型プリミティブライブラリ |
| keras | 2.10.0 | 深層学習 |
| matplotlib | 3.8.0 | グラフ描画 |
| numpy | 1.26.0 | 数値計算 |
| opencv-python | 4.8.1.78 | 画像の読み込み，保存 |
| Pillow | 2.1.4 | 画像の読み込み |
| scikit-learn | 1.3.1 | 画像の |
| seaborn | 0.13.0 | データ可視化 |

表 4.7　Raspberry Pi環境

|  |  |
| --- | --- |
| 名称 | OS |
| Raspberry Pi 4 | OS: GNU / linux  Version: 10 |

表 4.8　 Raspberry Pi環境における開発・実験環境

|  |  |  |
| --- | --- | --- |
| 名称 | バージョン | 利用用途 |
| C言語 | 8.3.0 | プログラミング言語 |

### プログラミング言語

本研究では，PythonとC言語を用いて実装を行った．Pythonはコーディングが他の言語と比較してシンプルであるという点や，機械学習のライブラリが豊富に提供されているといった利点から用いている．Pythonの仮想環境として，Windows11ではAnaconda3，Ubuntu22.04ではvenvを使用している．また，Raspberry Piで用いるC言語は実行速度が速く，汎用性が高いといった利点があり，本研究では映像記録プログラムの実装に使用している．

### Pythonライブラリ

本研究で使用しているPythonライブラリについて説明する．提案手法で用いるオートエンコーダ，距離学習ニューラルネットワークはKerasの深層学習ライブラリを用いて実装している．評価手法であるk-Meansの実装には，scikit-learnの機械学習ライブラリを用いており，t-SNEによる次元圧縮はseabornのライブラリを用いている．また，図やグラフによる可視化にはmatplotlibを用いて実装している．ベクトル同士の数値計算にはnumpyのライブラリを用いている．

## 使用モデルについて

### オートエンコーダ

本研究では，画像からの特徴抽出に畳み込み層と全結合層からなるオートエンコーダと，実験では使用しないがベクトル量子化という手法を適用したオートエンコーダであるVQ-VAEの2種類のモデルでの実装を行った．提案手法では特徴抽出にオートエンコーダのエンコーダのみを用いるため，表 4.9，表 4.10に，畳み込み層と全結合層からなるオートエンコーダのネットワーク構造をそれぞれエンコーダ部分とデコーダ部分に分割して示す．またVQ- VAEによる特徴抽出モデルでの実装もおこなったので，表 4.11にVQ-VAEのネットワーク構造を示す．実装したオートエンコーダが多層パーセプトロンではなく畳み込み層を含む構成になっている理由として，画像を全結合層に直接入力すると計算量が大きくなってしまうという問題がある．今回のモデルでは，畳み込み層を用いることで入力画像を3分の1まで圧縮して全結合層に入力している．このように，低次元の画像特徴抽出が可能な畳み込み層を使用し，モデルの学習時における計算量を削減することを目的として実装している．同様の目的でプーリング処理も実施されているが，畳み込み層に組み込まれているため表 4.9，表 4.10では明記せずにまとめてConv2Dと記載している．畳み込み層と全結合層からなるオートエンコーダは，エンコーダで抽出する画像特徴ベクトルを1000，500，100次元の3種類に変更したため，表 4.9，表 4.10中の全結合層では各次元のパラメータ数を記載している．

表 4.9　オートエンコーダのエンコーダ部分のネットワーク構成

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Parameters |
| InputLayer | (None, 256, 256, 3) | 0 |
| Conv2D | (None, 256, 256, 3) | 84 |
| LeakyReLU | (None, 256, 256, 3) | 0 |
| Conv2D | (None, 128, 128, 16) | 448 |
| LeakyReLU | (None, 128, 128, 16) | 0 |
| Conv2D | (None, 64, 64, 32) | 4640 |
| LeakyReLU | (None, 64, 64, 32) | 0 |
| Conv2D | (None, 32, 32, 64) | 18496 |
| LeakyReLU | (None, 32, 32, 64) | 0 |
| Flatten | (None, 65536) | 0 |
| Dense | (None, 1000)  または  (None, 500)  または  (None, 100) | 1000次元：65537000  500次元：32768500  100次元：6553700 |

表 4.10　オートエンコーダのデコーダ部分のネットワーク構成

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Parameters |
| InputLayer | (None, 1000)  または  (None, 500)  または  (None, 100) | 0 |
| Dense | (None, 65536) | 1000次元：65601536  500次元：32833536  100次元：6619136 |
| Reshape | (None, 32, 32, 64) | 0 |
| Conv2DTranspose | (None, 64, 64, 64) | 36928 |
| LeakyReLU | (None, 64, 64, 64) | 0 |
| Conv2DTranspose | (None, 128, 128, 32) | 18464 |
| LeakyReLU | (None, 128, 128, 32) | 0 |
| Conv2DTranspose | (None, 256, 256, 16) | 4624 |
| LeakyReLU | (None, 256, 256, 16) | 0 |
| Conv2DTranspose | (None, 256, 256, 3) | 435 |
| Activation (sigmoid) | (None, 256, 256, 3) | 0 |

表 4.11　VQ-VAEのネットワーク構成

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Parameters |
| InputLayer | (None, 256, 256, 3) | 0 |
| Conv2D + ReLU | (None, 128, 128, 32) | 896 |
| Conv2D + ReLU | (None, 64, 64, 64) | 18496 |
| Conv2D + ReLU | (None, 64, 64, 16) | 9232 |
| VectorQua | (None, 64, 64, 16) | 1024 |
| Conv2DTranspose + ReLU | (None, 128, 128, 64) | 9280 |
| Conv2DTranspose + ReLU | (None, 256, 256, 32) | 18464 |
| Conv2DTranspose + ReLU | (None, 256, 256, 3) | 867 |

### 距離学習ニューラルネットワーク

本研究では，オートエンコーダから抽出したベクトル形式またはテンソル形式の画像特徴を，分類やランキングに適した特徴ベクトルに変換する必要があるため，多層パーセプトロンで構成された全結合層のみのモデルと，畳み込み層と全結合層で構成されたモデルの2種類で距離学習ニューラルネットワークを用いている．全結合層のみで構成されたモデルは，畳み込み層をもつモデルよりも全結合層のユニット数と全結合層を増やしており，全結合層のみで構成されたモデルの表現力を向上させるという目的で実装した．全結合層のみで構成されたモデルを表 4.12に，畳み込み層と全結合層で構成されたモデルを表 4.13に示す．

表 4.12　全結合層のみの距離学習ニューラルネットワーク構造

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Parameters |
| InputLayer | (None, 1000)  または  (None, 500)  または  (None, 100) | 0 |
| Dense | (None, 1000) | 1000次元：1001000  500次元：501000  100次元：101000 |
| Dense | (None, 2000) | 2002000 |
| Dense | (None, 1000) | 2001000 |
| TFOpLambda | (None, 1000) | 0 |

表 4.13　畳み込み層と全結合層からなる距離学習ニューラルネットワーク構造

|  |  |  |
| --- | --- | --- |
| Layer | Output Shape | Parameters |
| InputLayer | (None, 32, 32, 64)  画像の場合：(None, 256, 256, 3) | 0 |
| Conv2D + ReLU | (None, 15, 15, 16)  画像の場合：(None, 127, 127, 16) | 9232 |
| Conv2D + ReLU | (None, 7, 7, 32)  画像の場合：(None, 63, 63, 32) | 4640 |
| GlobalAveragePooling2D | (None, 32) | 0 |
| dense | (None, 1000)  (None, 500)  (None, 100) | 1000次元：33000  500次元：16500  100次元：3300 |
| TFOpLambda | (None, 100) | 0 |

## 画像集合の読み込みと訓練用データ，検証用データへの分割

　本研究の実験システムでは，Raspberry Piを用いて独自に収集したデータを用いるため，クラス毎の画像集合の読み込みにおいて，フォルダの構造でクラス情報を取得している．そのため，事前にクラス毎でフォルダに保存し，クラス毎のフォルダを同じ階層のディレクトリに配置されていることを前提としている．はじめに，train\_data\_pathでクラス毎に保存されているフォルダまでのディレクトリの階層パスを指定する．次に，image\_sizeで画像を読み込むサイズを整数で指定する．実験システムにおいて，画像は1:1の正方形で読み込まれたのち，numpy配列に格納される．color\_settingでは，グレースケールであれば0，カラー画像であれば3を指定する．以上の設定事項を適切に指定することで，numpy配列に格納された画像集合のデータと数字で表記されたクラス情報を訓練用データと検証用データに分割された状態で取得することが可能である．以下の表 4.14に画像集合を読み込むプログラムのソースコードを示す．また，表 4.15に訓練用データと検証用データに分割するプログラムのソースコードを示す．

表 4.14　画像集合データを読み込むプログラム

|  |
| --- |
| train\_data\_path = 'DIR'  image\_size = IMAGE\_SIZE  color\_setting = 3  folder = [f for f in os.listdir(train\_data\_path) if os.path.isdir(os.path.join(train\_data\_path, f))]  class\_number = len(folder)  print('今回のデータで分類するクラス数は「', str(class\_number), '」です。')  X\_image = []  Y\_label = []  label\_count = np.zeros(class\_number)  for index, name in enumerate(folder):  read\_data = train\_data\_path + '/' + name  files = glob.glob(read\_data + '/\*.png')  print('--- 読み込んだデータセットは', read\_data, 'です。')  num=0  for i, file in enumerate(files):  if color\_setting == 1:  img = load\_img(file, color\_mode = 'grayscale' ,target\_size=(image\_size, image\_size))  elif color\_setting == 3:  img = load\_img(file, color\_mode = 'rgb' ,target\_size=(image\_size, image\_size))  array = img\_to\_array(img)  X\_image.append(array)  num +=1  Y\_label.append(index)  print('index: ',index,' num:',num)  label\_count[index] = num  X\_image = np.array(X\_image)  Y\_label = np.array(Y\_label) |

表 4.15　訓練用データと検証用データに分割するプログラム

|  |
| --- |
| train\_images,valid\_images,train\_labels,valid\_labels=train\_test\_split(X\_image,Y\_label,test\_size=0.20,shuffle = True)  x\_train = train\_images  y\_train = train\_labels  x\_test = valid\_images  y\_test = valid\_labels |

## オートエンコーダのネットワーク構築と画像集合による事前学習

特徴抽出を行うオートエンコーダの構築と，構築したモデルの事前学習を実施する．オートエンコーダの構築にはKerasのライブラリを用いて実装した．構築するオートエンコーダはエンコーダで1000，500,100次元ベクトルの画像特徴を抽出するために，Denseの部分のパラメータZ\_DIMに各次元数の値を指定している．以下の表 4.16にオートエンコーダを構築するプログラムのソースコードを示す． また，実験システムのオートエンコーダでは損失関数の処理を指定しているため，損失関数についてのソースコードを表 4.17に，指定した損失関数を用いたモデルの学習をするプログラムのソースコードを表 4.18に，また学習済みモデルを保存するプログラムのソースコードを表 4.19に示す．

表 4.16　オートエンコーダのモデル構築プログラム

|  |
| --- |
| encoder\_input = Input(shape=(IMAGE\_SIZE, IMAGE\_SIZE,3), name='encoder\_input')  x = encoder\_input  x = Conv2D(filters=3, kernel\_size=3, strides=1, padding='same', name='encoder\_conv\_0')(x)  x = LeakyReLU()(x)  x = Conv2D(filters=16, kernel\_size=3, strides=2, padding='same', name='encoder\_conv\_1')(x)  x = LeakyReLU()(x)  x = Conv2D(filters=32, kernel\_size=3, strides=2, padding='same', name='encoder\_conv\_2')(x)  x = LeakyReLU()(x)  x = Conv2D(filters=64, kernel\_size=3, strides=2, padding='same', name='encoder\_conv\_3')(x)  x = LeakyReLU()(x)  shape\_before\_flattening = K.int\_shape(x)[1:]  x = Flatten()(x)  encoder\_output = Dense(Z\_DIM, name='encoder\_output')(x)  encoder = Model(encoder\_input, encoder\_output)  decoder\_input = Input(shape=(Z\_DIM,), name='decoder\_input')  x = Dense(np.prod(shape\_before\_flattening))(decoder\_input)  x = Reshape(shape\_before\_flattening)(x)  x = Conv2DTranspose(filters=64, kernel\_size=3, strides=2, padding='same', name='decoder\_conv\_0')(x)  x = LeakyReLU()(x)  x = Conv2DTranspose(filters=32, kernel\_size=3, strides=2, padding='same', name='decoder\_conv\_1')(x)  x = LeakyReLU()(x)  x = Conv2DTranspose(filters=16, kernel\_size=3, strides=2, padding='same', name='decoder\_conv\_2')(x)  x = LeakyReLU()(x)  x = Conv2DTranspose(filters=3, kernel\_size=3, strides=1, padding='same', name='decoder\_conv\_3')(x)  x = Activation('sigmoid')(x)  decoder\_output = x  decoder = Model(decoder\_input, decoder\_output)  decoder.summary()  # エンコーダ/デコーダ連結  model\_input = encoder\_input  model\_output = decoder(encoder\_output)  model = Model(model\_input, model\_output) |

表 4.17　損失関数の定義をするプログラム

|  |
| --- |
| def r\_loss(y\_true, y\_pred):  return K.mean(K.square(K.log(y\_true + 1) - K.log(y\_pred + 1)), axis=-1) |

表 4.18　指定した損失関数を用いたモデルの学習をするプログラム

|  |
| --- |
| model.compile(optimizer=optimizer, loss=r\_loss, metrics=['accuracy'])  history=model.fit(  x\_train,  x\_train,  batch\_size=BATCH\_SIZE,  epochs=EPOCHS,  validation\_data=(x\_test, x\_test),  ) |

表 4.19　学習済みモデルを保存するプログラム

|  |
| --- |
| model.save("DIR") |

## 特徴ベクトルの抽出

事前学習させたオートエンコーダを用いて画像から，画像特徴ベクトルまたは画像特徴テンソルを中間層または畳み込み層から抽出する．オートエンコーダはKerasのライブラリを用いて，保存済みのモデルを呼び出すことで使用する．以下の表 4.20に保存したモデルをロードするソースコードを示す．また，学習済みオートエンコーダの特定の中間層までをモデルとして定義するプログラムのソースコードを表 4.21に示す．

表 4.20　保存したモデルをロードするプログラム

|  |
| --- |
| Encoder\_model = keras.models.load\_model("DIR", custom\_objects={"r\_loss": r\_loss }) |

表 4.21　エンコーダのみをモデルとして定義するプログラム

|  |
| --- |
| layer\_name = 'flatten'  intermediate\_layer\_model = Model(inputs=Encoder\_model.input,  outputs=Encoder\_model.get\_layer(layer\_name).output) |

## 距離学習ニューラルネットワークによる距離学習

事前学習させた距離学習ニューラルネットワークを用いて，画像特徴ベクトルまたは画像テンソルをラベル間の距離が最適化された特徴ベクトルに変換する．以下の表 4.22に距離学習ニューラルネットワーク構造を定義するプログラムのソースコードを示す．

表 4.22　距離学習ニューラルネットワーク構造を定義するプログラム

|  |
| --- |
| inputs2 = layers.Input(shape=(Z\_DIM))  x2 = layers.Dense(units=1000, activation='relu')(inputs2)  x2 = layers.Dense(units=2000, activation='relu')(x2)  embeddings = layers.Dense(units=1000, activation=None)(x2)  embeddings = tf.nn.l2\_normalize(embeddings, axis=-1)  model2 = EmbeddingModel(inputs2, embeddings) |

## クラスタリングによる類似画像分類

距離ニューラルネットワークにより変換した距離が最適化された特徴ベクトルに対し，クラスタリングアルゴリズムを適用することで，類似画像毎のクラスタを取得する．実験システムでは，機械学習に関連したライブラリであるscikit -learnを用いて，非階層クラスタリングアルゴリズムであるk-Meansによる分類プログラムを実装している．k-Meansを用いる理由としては，代表的な分類アルゴリズムであるという点と，分類するクラス数が事前に決定しているという点があるためである．k-Meansによるクラスタリングを行うプログラムのソースコードを以下の表 4.23に示す．

表 4.23　 k-Meansによるクラスタリングを行うプログラム

|  |
| --- |
| n\_clusters=4  kmeans\_model = KMeans(n\_clusters)  predict\_clus = kmeans\_model.fit\_predict(feature\_vector) |

## コサイン類似度による特徴ベクトルの類似度計算

実験システムでは，提案手法により抽出された入力画像の特徴ベクトルと代表特徴ベクトルの類似度計算にコサイン類似度を用いて算出している．以下の表 4.24にコサイン類似度を計算するプログラムのソースコードを示す．

表 4.24　コサイン類似度を計算するプログラム

|  |
| --- |
| def cos\_sim(X,Y):  return np.dot(X,Y)/(np.linalg.norm(X)\*np.linalg.norm(Y))  for j,f\_vector in enumerate(npz\_['f\_vector']):  f\_sim = cos\_sim(feature\_vector[0],f\_vector) |

## Webサーバの構築

実験システムではUbuntuがインストールされたPC環境上で，Raspberry Piで撮影した画像データを受け取る処理と，ユーザからの入力画像に対する代表特徴ベクトルとの類似度計算結果を返すWebフォームを，Flaskのライブラリを用いたWebサーバを構築することで実装する．以下の表 4.25にRaspberry Piからの画像データを受け取るプログラムを示す．また，ユーザからのアクセスに対し，画像入力フォームをユーザに返すプログラムのソースコードを表 4.26に，入力画像と代表特徴ベクトルの類似度計算結果を返すプログラムのソースコードを表 4.27に示す．さらに，画像入力フォームのスクリーンショットを図 4.7に，実験システムによる入力画像とクラスタ毎の類似度比較を表示したWebページのスクリーンショットを図 4.8に示す．

表 4.25　 Raspberry Piから画像データを受け取るプログラム

|  |
| --- |
| @app.route("/send", methods=["POST"])  def send():  \_bytes = np.frombuffer(request.data, np.uint8)  img = cv2.imdecode(\_bytes, flags=cv2.IMREAD\_COLOR)  dt\_now = str(datetime.datetime.now())  dt\_now = dt\_now.replace(' ','')  dt\_now = dt\_now.replace('.','')  print(dt\_now)  client = str(request.remote\_addr)  print(client + " から送信されました")    SAMPLE\_DIR = client  if not os.path.exists('./static/rasp\_img/'+SAMPLE\_DIR):  # ディレクトリが存在しない場合、ディレクトリを作成する  os.makedirs('./static/rasp\_img/'+SAMPLE\_DIR)  cv2.imwrite('./static/rasp\_img/'+SAMPLE\_DIR+'/'+dt\_now+'.jpg', img, [cv2.IMWRITE\_JPEG\_QUALITY, 95])  return jsonify(img.shape) |

表 4.26　画像入力フォームをユーザに返すプログラム

|  |
| --- |
| @app.route('/upload', methods=['GET', 'POST'])  def upload():  if request.method == 'GET':  return render\_template('upload.html')  elif request.method == 'POST':  file = request.files['example']  dt\_now = str(datetime.datetime.now())  dt\_now = dt\_now.replace(' ','')  dt\_now = dt\_now.replace('.','')  print(dt\_now)  file.filename = dt\_now + file.filename  file.save(os.path.join('./static/image', file.filename))  return redirect(url\_for('uploaded\_file', filename = file.filename)) |

表 4.27　入力画像と代表特徴ベクトルの類似度計算結果を返すプログラム

|  |
| --- |
| @app.route('/uploaded\_file/<string:filename>')  def uploaded\_file(filename):  img = Image.open('./static/image/'+filename)  d,h = Search.search('./static/image/'+filename)  return render\_template('uploaded\_file.html', filename="../static/image/"+filename, data=zip(d,h)) |

|  |
| --- |
|  |

図 4.1　画像入力フォームのスクリーンショット

|  |
| --- |
|  |

図 4.2　類似度比較結果を表示するWebページのスクリーンショット

## Raspberry Piでのモニタリング機能の構築

Raspberry Piとカメラを用いてモニタリング機能を実装する．以下の表 4.28に，定期的に画像を撮影して画像を送信するプログラムのソースコードを示す．

表 4.28　モニタリングをするプログラム

|  |
| --- |
| with picamera.PiCamera() as camera:  camera.resolution = (640, 640)  camera.start\_preview()  time.sleep(10)  input("please enter any key\n")  while i!=120:  time.sleep(5)  camera.capture('test.jpg')  # read image data  f = open("test.jpg", "rb")  reqbody = f.read()  f.close()  # create request with urllib  url = "http://10.16.34.12:8000/send"  req = urllib.request.Request(  url,  reqbody,  method="POST",  headers={"Content-Type": "application/octet-stream"},  )  i+=1  # send the request and print response  with urllib.request.urlopen(req) as res:  print(json.loads(res.read())) |

# **実験**

本章では，実験概要と実験目的，実験結果について述べる．

## 実験目的

　実験は2つ実施した．提案手法を用いた実験システムでの特徴抽出を行うことで，画像分類やランキングに適した，距離の最適化された特徴ベクトルを得られるかを確認する．抽出した特徴ベクトルのクラスタリング結果や特徴ベクトルの可視化した結果を確認することで，提案手法の実現可能性を評価する．

## 実験環境

画像データセットとして，国土地理院の航空写真データセット8)を利用する．航空写真は576枚からなり，「田畑」，「水源地帯」，「森林」，「建物」の4つクラスに144枚ずつ分類した．画像データの例を図 5.1に示す．

オートエンコーダAEは，畳み込み層を持つもの1種類を用意する．また，距離学習ニューラルネットワークは畳み込み層を持たない距離学習NN-1，畳み込み層を持つ距離学習NN-2の2種類を用いる．オートエンコーダAEの学習には，航空写真576枚を用いて訓練する．距離学習ニューラルネットワークの学習は，オートエンコーダAEから抽出された画像特徴（画像ベクトルまたは画像テンソル）または画像データを用いて教師付きで訓練する．

|  |  |
| --- | --- |
| クラス1  田畑 | タイルの壁  低い精度で自動的に生成された説明  建物, 屋外, 座る, ウォーキング が含まれている画像  自動的に生成された説明 |
| クラス2  水源地帯 | 波にのっている  低い精度で自動的に生成された説明 羊, 立つ, 民衆, グループ が含まれている画像  自動的に生成された説明 |
| クラス3  森林地帯 | 木, ウォーキング, 吊るす, グリーン が含まれている画像  自動的に生成された説明 緑の木々  低い精度で自動的に生成された説明 |
| クラス4  建物 | 建物, 草 が含まれている画像  自動的に生成された説明 |

図 5.1　航空写真データの例 (4分類)

## 実験1

### 実験目的

提案手法により抽出した特徴ベクトルを用いた画像分類精度から，提案手法の実現可能性を確認する．

### 実験方法

下記の6つのモデルについてk-meansを適用して分類した精度を算出し，結果を比較する．それぞれのニューラルネットワークにおいてオートエンコーダの中間層および距離学習ニューラルネットワークの出力層の次元を，1000次元，500次元，100次元で変化させる．

M1：AEのみで構成する．AEのエンコーダからの出力を特徴ベクトルとする．

M2：AEのみで構成する．AEのエンコーダ内にある畳み込み層の出力をベクトル化したものを特徴ベクトル（32\*32\*64次元）とする．

M3：M1のAEと距離学習NN-1で構成する．M1のAEから抽出した特徴ベクトルを，距離学習NN-1で変換したものを特徴ベクトルする．

M4：M2のAEと距離学習NN-1で構成する．M1のAEから抽出した特徴ベクトルを，距離学習NN-1で変換したものを特徴ベクトルする．

M5：M1のAEと距離学習NN-2で構成する．M1のAEかエンコーダ内にある畳み込み層から抽出される特徴テンソル（32次元, 32次元, 64次元）を，距離学習NN-2で変換したものを特徴ベクトルする．

M6：距離学習NN-2のみで構成する．画像を特徴テンソル（256次元, 256次元, 3次元）として入力とし，距離学習NN-2で変換したものを特徴ベクトルする．

再現率，適合率は下記のそれぞれ(1)，(2)に式で算出する．

(1)

(2)

### 実験結果

以下にk-Meansのクラスタリングを適用した結果から算出した各モデルの再現率の比較を図 5.2に，各モデルの適合率を図 5.3に示す．また，M5，M6における平均再現率と平均適合率の比較結果を表 5.1に，学習における損失値の比較を図 5.4に示す．

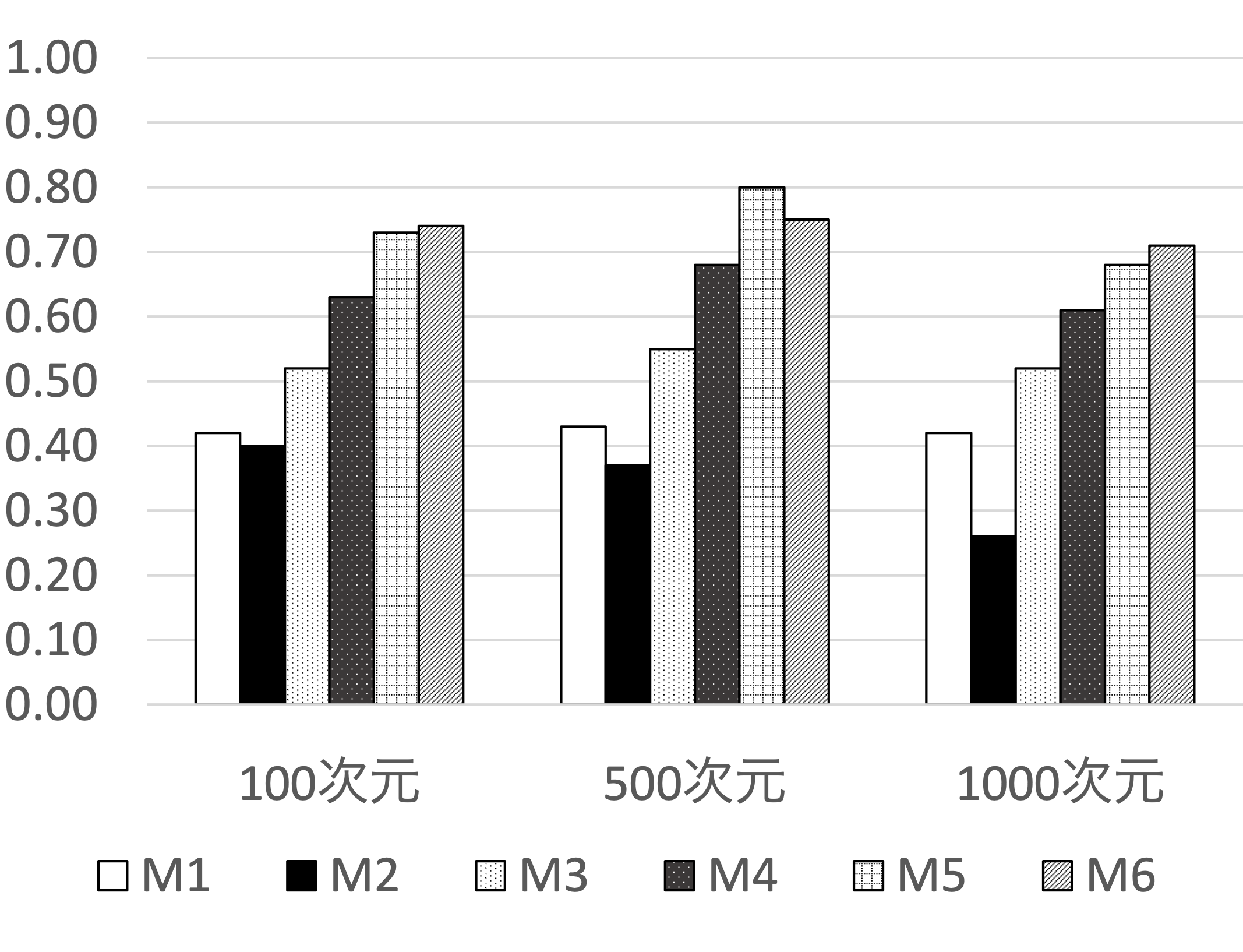


図 5.2　各モデルの再現率の比較

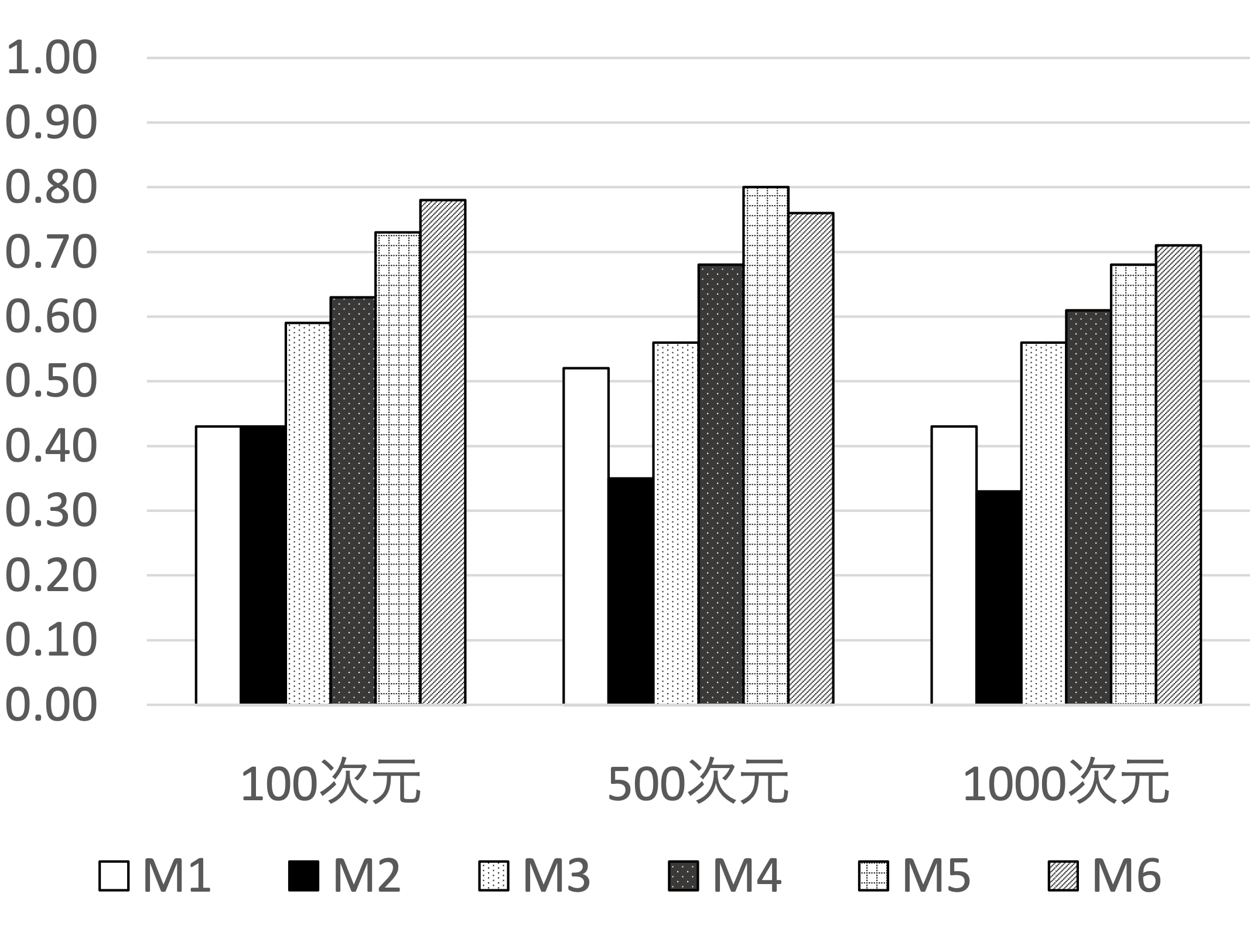


図 5.3　各モデルの適合率の比較

表 5.1　平均再現率と平均適合率の比較 (M5とM6)

|  |  |  |
| --- | --- | --- |
|  | M5 | M6 |
| エポック数 | 600 | 1000 |
| 平均再現率 | 0.736 | 0.733 |
| 平均適合率 | 0.736 | 0.750 |

|  |  |
| --- | --- |
| グラフ, ヒストグラム  自動的に生成された説明 | グラフ, ヒストグラム  自動的に生成された説明 |
| M5の損失値変化 | M6の損失値変化 |

図 5.4　学習における損失値の比較

### 考察

図 5.2，図 5.3は，各モデルの再現率，適合率の比較結果を示している．モデルM1とM2はオートエンコーダのみで構成され，モデルM3とM4はオートエンコーダと距離学習ニューラルネットワークから構成されるモデルである．モデルM3とM4が再現率，適合率ともに向上しているため，距離学習ニューラルネットワークの適用により，画像分類精度を向上することが可能であることが確認できる．

また，モデルM5は，M3，M4と同様にオートエンコーダと距離学習ニューラルネットワークから構成されるが，距離学習ニューラルネットワークに畳み込み層をもつモデルである．一方，モデルM3とM4は距離学習ニューラルネットワークに畳み込み層ではなく，全結合層で構成している．モデルM5の再現率，適合率が向上していることから，画像分類においては，距離学習ニューラルネットワークに畳み込み層を用いて構成した方が分類精度向上の効果が高いことがわかる．

モデルM6はオートエンコーダを使用せずに，距離学習ニューラルネットワークのみで構成され，入力には抽出した画像特徴ではなく，画像データそのものを用いる．オートエンコーダの中間層および距離学習ニューラルネットワークの出力層を100次元と1000次元とした場合に，M6の再現率，適合率が向上する結果となった．

ここで，表 5.1および図 5.4は，モデルM5とM6における平均再現率と平均適合率および学習時(100次元)の損失変化の比較結果を示している．表 5.1より，100次元，500次元，1000次元の平均再現率ではM5の方が高く，平均適合率はM6の方が高いが，再現率平均適合率の差分は0.014と小さいことがわかる．また，図 5.4の結果から，M5の方が，損失値の変動が小さく学習が安定していることが確認できる．また，学習時において損失値がともに1.2付近から開始しており，オートエンコーダから抽出した特徴テンソルを用いた場合および画像データを用いた場合ともに，学習開始時点では距離の関係が十分に適切でないことがわかる．

M5の学習エポック数が小さいのは，オートエンコーダで学習したことにより，距離学習に適切な画像特徴が抽出されているためと考えられる．

以上，モデルM5とM6の比較結果から，オートエンコーダと距離学習ネットワークを分離して構成することは，分類精度および学習の安定性の観点から一定の効果があると考えられる．

## 実験2

### 実験目的

提案手法により抽出した特徴ベクトル同士の距離を可視化することで，提案手法の実現可能性を確認する．

### 実験方法

実験1で使用した下記の6つのモデルについてt-SNE (t-Distributed Stochastic Neighbor Embedding)を適用して2次元まで次元圧縮し，特徴ベクトルの距離関係をグラフに描画することで，視覚的な評価に基づいた比較を実施する．t-SNEは可視化目的に用いられるアルゴリズムであるため，本実験では正確な数値での精度評価は行わない．それぞれのニューラルネットワークにおいてオートエンコーダの中間層および距離学習ニューラルネットワークの出力層の次元を，100次元，500次元，1000次元で変化させる．

M1：AEのみで構成する．AEのエンコーダからの出力を特徴ベクトルとする．

M2：AEのみで構成する．AEのエンコーダ内にある畳み込み層の出力をベクトル化したものを特徴ベクトル（32\*32\*64次元）とする．

M3：M1のAEと距離学習NN-1で構成する．M1のAEから抽出した特徴ベクトルを，距離学習NN-1で変換したものを特徴ベクトルする．

M4：M2のAEと距離学習NN-1で構成する．M1のAEから抽出した特徴ベクトルを，距離学習NN-1で変換したものを特徴ベクトルする．

M5：M1のAEと距離学習NN-2で構成する．M1のAEかエンコーダ内にある畳み込み層から抽出される特徴テンソル（32次元, 32次元, 64次元）を，距離学習NN-2で変換したものを特徴ベクトルする．

M6：距離学習NN-2のみで構成する．画像を特徴テンソル（256次元, 256次元, 3次元）として入力とし，距離学習NN-2で変換したものを特徴ベクトルする．

### 実験結果

実験方法で定義した6つのモデルを用いて，抽出した100，500，1000次元の画像特徴ベクトルを次元圧縮し，描画した結果を表 5.1に示す．

表 5.1　各モデルによる次元毎の抽出した画像特徴ベクトルの描画

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1000次元 | 500次元 | 100次元 |
| M1 |  |  |  |
| M2 |  |  |  |
| M3 |  |  |  |
| M4 |  |  |  |
| M5 |  |  |  |
| M6 |  |  |  |

### 考察

実験結果の表 5.1より，M1，M2と比較して，M3，M4はクラスを考慮した描画がされていることがわかる．また，M5，M6においても，クラスを考慮した描画がされている．このことから，距離学習ニューラルネットワークを適用することで，画像特徴をクラス毎の距離情報を抽出しやすい特徴ベクトルへと変換できることがわかる．

また，M3，M4は描画した特徴ベクトルが線状に集中してしまっているが，M5，M6はクラス毎に分かれつつ，クラス内においてはある程度分散していることがわかる．t-SNEはこのことから，M3，M4と比較して，M5，M6は距離の情報以外に何らかの情報を特徴として保持しているのではないかと考えられる．

以上，モデルM3，M4，M5，M6の結果から，オートエンコーダで抽出した画像特徴に距離学習ネットワークを適用することで，画像特徴を画像分類やランキングに適した特徴ベクトルに変換可能であることが確認できた．

# **むすび**

本研究では，画像特徴を抽出するネットワークとしてオートエンコーダ，および距離学習ニューラルネットワークを適用した画像分類手法を提案した．航空写真データを用いた実験において，提案手法を用いて画像から低次元で抽出した画像特徴に，距離学習ニューラルネットワークを適用することで，抽出した画像特徴の分類精度を向上させることができることを示した．また，t-SNEを用いた画像特徴ベクトルの可視化による実験において，オートエンコーダで抽出した特徴ベクトルを，距離学習ニューラルネットワークを適用し，変換した特徴ベクトルを2次元に次元圧縮して可視化することで，クラス毎の適切な距離関係を含むベクトルが距離学習の適用により得られることを示し，提案手法による画像分類手法についての実現可能性を確認した．

今後の予定として，インデクシングした特徴ベクトルを使用した類似度による検索精度を調査し，提案手法のインデクシング手法における実現可能性を確認する．また，提案手法ではオートエンコーダと距離学習ニューラルネットワークを分割して特徴ベクトルの抽出を実施しているが，他にもオートエンコーダに距離学習を組み合わせたモデルや，ベクトル量子化を組み込んだVQ-VAEといった他のオートエンコーダを用いた実験も検討している．また，提案手法を適用して，自然環境観測において異なる複数地点に設置された観測カメラから記録・蓄積される映像データを対象とした映像シーン分析システムの構築に応用していくことを検討している．

# 謝辞

# **参考文献**

1. 国土交通省: “https://www.mlit.go.jp/river/gijutsu/inovative\_project/index.html”
2. Hiroki Mimura, Masaya Tahara, Kosuke Takano, Nobuya Watanabe, Kin Fun Li: Video Indexing for Live Nature Camera on Digital Earth, International Conference on Advanced Information Networking and Applications, pp.660-667 (2023).
3. 鬼塚洋輔, 大山航, 山田太造, 井上聡, 内田誠一：花押類似検索のための畳み込みオートエンコーダによる画像特徴抽出，じんこんもん2018論文集，pp.257-262 (2018).
4. 細江麻梨子, 山田智輝, 加藤邦人, 山本智一: 条件付きAutoEncoderによる書き癖抽出手法の提案，情報処理学会第80回全国大会，2C-06, No. 2, pp. 37-38 (2018).
5. 高橋柊，落合桂，深澤祐介: Stacked Convolutional Denoising Autoencodersを用いた2誘導心電図からの特徴抽出および不整脈分類, 情報処理学会論文誌, Vol.59, No.12, pp.2213-2220 (2018).
6. Haque Ishfaq, Assaf Hoogi, Daniel Rubin: TVAE: Triplet-Based Variational Autoencoder using Metric Learning, arXiv:1802.04403 [stat.ML] (2018).
7. Giuseppina Andresini, Annalisa Appice, Donato Malerba: Autoencoder-based deep metric learning for network intrusion detection, INFORMATION SCIENCES, Volume 569, pp. 706-727 (2021).
8. 国土地理院：CNNによる水田抽出のための教師画像データ，国土地理院技術資料 H1-No.26 (2023).