**令和３年度　　卒業論文**

卒業論文題目

|  |
| --- |
| 画像検索のための画像特徴ベクトルの次元数に着目した認識精度と計算コストの関係性の調査 |

　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　　　　　１８２１００５

　　　　　学生氏名　　　　　 吉岡　拓郎

　提出日　令和年　月日　指導教員　　印

　受理日　令和年　月日　情報工学科長　　印

# **論文要旨**

TwitterやInstagramなどのソーシャルネットワーキングサービスにおいて写真や画像の投稿が盛んになっており，大量の写真や画像が蓄積されている．

ユーザが目的の写真や画像にアクセスする手段として，画像検索機能の重要性が増している．

# 目次

[**1.** **まえがき** 1](#_Toc536149930)

[1.1 はじめに 1](#_Toc536149931)

[1.2 本論文の構成 2](#_Toc536149932)

[**2.** **関連研究** 3](#_Toc536149933)

[2.1 3](#_Toc536149934)

[**3.** **提案方式** 5](#_Toc536149938)

[3.1 提案方式の概要 5](#_Toc536149939)

[3.2 提案方式の特徴 5](#_Toc536149940)

[**4.** **実装** 9](#_Toc536149942)

[4.1 画像検索評価システムの実装 9](#_Toc536149943)

[4.1.1 実装環境概要 9](#_Toc536149944)

[4.1.2 実装環境の構築方法 9](#_Toc536149945)

[4.2 11](#_Toc536149948)

[4.2.1 実装環境概要 11](#_Toc536149949)

[4.2.2 12](#_Toc536149950)

[4.2.3 12](#_Toc536149951)

[4.3 提案システムの実装 17](#_Toc536149955)

[4.3.1 実装環境概要 17](#_Toc536149956)

[4.3.2 実装環境の構築方法 18](#_Toc536149957)

[4.3.3 実装システムのフォルダ構成 18](#_Toc536149958)

[**5.** **実験** 23](#_Toc536149962)

[5.1 実験1：写真撮影頻度算出方式に関する実験 23](#_Toc536149963)

[5.1.1 実験目的 23](#_Toc536149964)

[5.1.2 実験環境 23](#_Toc536149965)

[5.1.3 実験方法 24](#_Toc536149966)

[5.1.4 実験結果 24](#_Toc536149967)

[5.1.5 実験考察 26](#_Toc536149968)

[5.2 実験2：編集工程を加味した実験 28](#_Toc536149969)

[5.2.1 実験目的 28](#_Toc536149970)

[5.2.2 実験環境 28](#_Toc536149971)

[5.2.3 実験方法 30](#_Toc536149972)

[5.2.4 実験結果 30](#_Toc536149973)

[5.2.5 実験考察 31](#_Toc536149974)

[5.3 実験考察 32](#_Toc536149975)

[**6.** **むすび** 33](#_Toc536149976)

[**謝辞**](#_Toc536149977)

[**参考文献**](#_Toc536149978)

# **まえがき**

## はじめに

TwitterやInstagram等のソーシャルネットワーキングの普及に伴い，写真や画像の投稿が盛んに行われるようになり，大量の画像や写真が蓄積されている．そこで，ユーザが目的の写真や画像にアクセスする手段として，画像検索機能の重要性が増している．又，画像認識の性能向上のため，多くの研究が推進されている．

画像検索機能の画像認識において，画像の特徴量が高次元になるほど，認識性能が高くなる．しかし，特徴量が高次元になると特徴ベクトルの算出コスト，新式処理の計算コストが大きく増加してしまい，解析が困難になってしまう．この現象を次元の呪いという．次元の呪いとは，データの次元数が大きくなり過ぎると，そのデータで表現できる組み合わせが飛躍的に多くなってしまい，その結果，手元にある有限なサンプルデータでは十分な学習結果が得られなくなる.計算コストが莫大となるだけでなく，十分な学習結果が得られず、未知のデータに適切に対応出来なくなる等の不具合が発生することである．また，低次元だと，意味情報が損失し，認識性能が低くなるが，処理時間が減少するといったことが起こる．

上記のことを念頭に入れて，画像認識の画像処理が有効に行える範囲を調査する．ここで記載する意味情報とは，画像を認識する際に，その判断材料となる情報のことを指す．

画像認識の性能を向上させるために，画像認識に有効な特徴ベクトルとどれくらいの計算コストで処理できるのか計測する．それらの結果を踏まえて評価・考察をする．計算コストを抑えた画像認識手法を調査することで，画像検索システムを成り立たせるうえで，認識性能が正常に扱える有効な範囲はどこなのか明確にする．

## 本論文の構成

本論文は全６章で構成されている。本章では研究の背景や動機，目的について述べる．第２章では，モデルの作成手法，それらにまつわる関連研究について述べる．第３章では，本研究の提案方式とモデルの内容，学習データ，実験環境について述べる。第４章では，提案方式を構築した本システムの設計及び実験システムの実装について述べる．第５章では，Alexnetをもとに作成したモデルを使用して，性能について比較実験をする．実験の概要と，比較実験後の結果を述べる．第６章では，結論と今後の展望について述べ

# **関連研究**

## 

## 

## 本研究の位置づけ

# **提案方式**

本研究では，画像検索システムにおいて，画像処理が有効に扱える範囲を調査するための画像検索評価システムを提案する．学習モデルを自作し，次元数を操作して画像検索システムとして機能するかどうか実験・評価をする．

## 提案方式の概要

CNN　(Convolutional Neural Network)　を用いて抽出した画像の特徴ベクトルは，画像の意味情報が保持されてると仮定する．ここでの意味情報とは，画像を認識する際に，その判断材料となる情報のことを指す．

特徴ベクトルの生成には，CNNを利用する．CNNを用いた特徴ベクトルの生成方法としては，識別層の一つ二つ手前の全結合層を用いる．また，次元数を変化させる層は，特徴ベクトルを生成する層と同じく，識別層から一つ二つ手前の全結合層の値を変化させる．

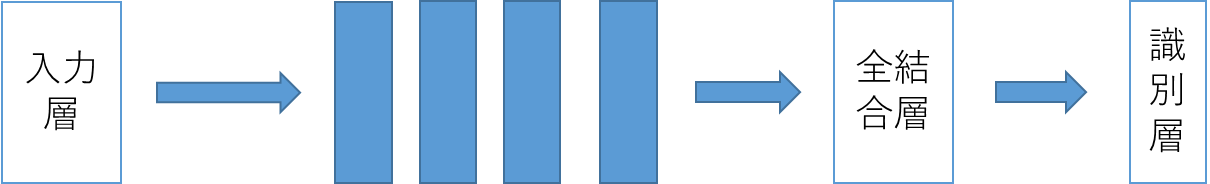


図 3.1　提案方式の概要図

## 提案方式の特徴

この提案方式では，画像認識の認識性能を高く保ち，計算コストを抑える最適な画像特徴ベクトルの次元数を調査する．画像集合を用意し，CNNを用いて特徴ベクトルを生成する．本研究では，cifar10データセット内のテストデータ一万件を画像集合として用意する．特徴ベクトルの次元数を変化させて，画像間の類似度を計算する．そのときの計算コストと精度について評価する．最適な次元数を見つけるための評価関数を以下に定義する．

評価関数

# **実装**

## 画像検索評価システムの実装

実装システムでは，画像検索において，画像処理が有効に扱える範囲を調査するために，GoogleCorabolatoryの仮想環境を使用して，pythonを用いて実装をした．検索の対象は，cifar10データセット内のテストデータ1万件を使用して，特徴ベクトルの抽出を行う．実装システムの流れを以下に示す．

1. Create\_model.jpynbのプログラムより，自作したモデル5つを使用する．
2. Cifar10\_vector\_pickle.jpynbに1で作成したモデルをロードして，cifar10データセット内のテストデータ1万件の特徴ベクトルの抽出を行う．
3. 作成した特徴ベクトルは，pickleファイルに保存する．
4. Search\_image.jpynbに3で作成した特徴ベクトルが保存されているpickleファイルをロードする．
5. Search\_image.jpynbにて，実験・評価を行う．

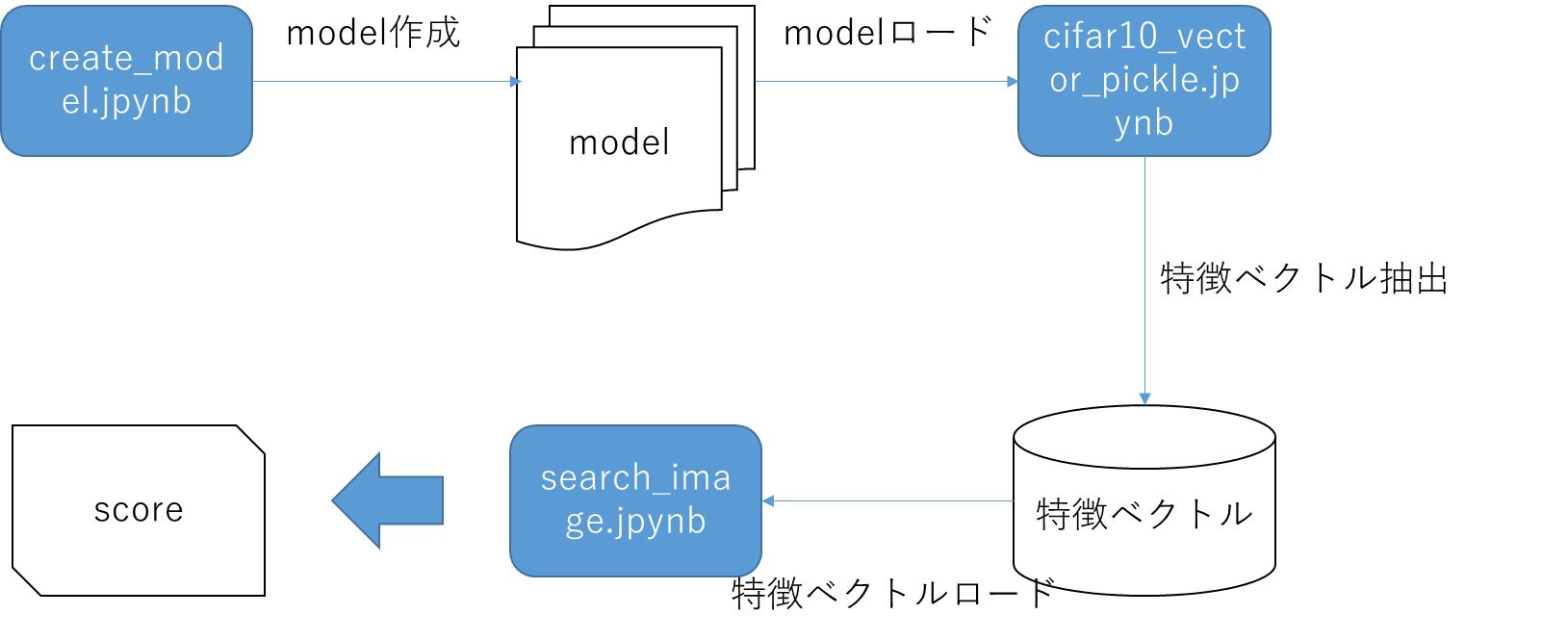


図 4-1　実装システム図

### 実装環境概要

本研究での画像検索評価システムの実装環境を表 4.1に示す．

表 4.1　画像検索評価システム構築環境

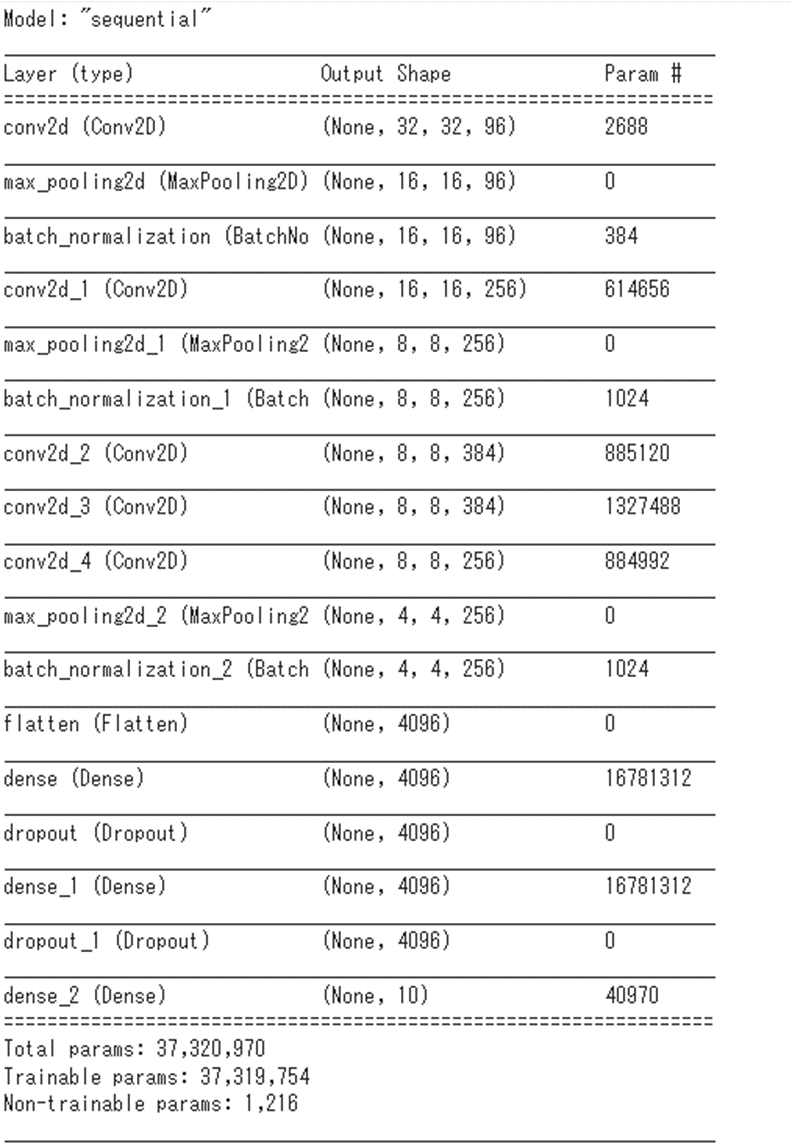
|  |  |  |
| --- | --- | --- |
| 利用用途 | 名称 | バージョン |
| OS | Windows10 | Windows10 |
| WebAPI |  |  |
| ソフトウェア・開発言語 | python |  |

### モデルの作成方法

モデルの作成では，GoogleCorabolatryを利用し，pythonでシステムの開発を行う．ここでは5との次元数の異なるモデルを作成する．それぞれ13層と15層のdence層の部の次元数を変化させてモデルを作成していく．以下に作成したモデルについて表に示す．



　作成するモデルの内容について，以下の図に示す．



### 特徴ベクトルの取得方法

特徴ベクトルを抽出するのは，cifar10\_vector\_pickle.jpynbによって行う．このプログラムでは，作成済みの学習済みモデルをロードする必要がある．

以下のコードでモデルをロードすることができる．

|  |
| --- |
| from keras.models import load\_model  model = load\_model('alexnet\_cifar10\_2000.h5')  model.summary() |

cifar10データセット内のテストデータ一万件を特徴ベクトル抽出に用いるため，cifar10データセットをロードします．以下にコードを示す．

|  |
| --- |
| from keras.datasets import cifar10  (x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data() |

　ここまでで，特徴ベクトルを取得できる準備は整っている．以下に特徴ベクトルを取得するコードを示す．

|  |
| --- |
| dense =models.Model(inputs=model.input, outputs=model.layers[-3].output)  dense\_out = dense.predict(x\_test) |

　一番上の行で，特徴ベクトルを抽出したい層を指定している．本研究では，上から15番目の層から特徴ベクトルの抽出を行う．次の行の所で，特徴ベクトルの抽出を行っている．これで，一万件の特徴ベクトルの抽出をすることができる．抽出した特徴ベクトルの保存については，pickleを用いる．理由としては，numpy配列として扱うことができるため，画像検索評価システムに落とし込みやすいからである．又，保存方法としてcsvも考えていたが，文字列として保存されてしまうので，扱える形に合うるのが少々手間である観点から使用していない．しかし，一万件のデータから一つ一つ取り出すといった作業は，pickleではできないため，必要に応じて使い分ける必要がある．以下に示すコードは，pickleでの保存，ロードの方法である．

|  |
| --- |
| with open('cifar10\_vecs2000.pickle', 'wb') as f:    pickle.dump(dense\_out, f) |
| with open('cifar10\_vecs2000.pickle', 'rb') as f:    b = pickle.load(f) |

　上記のように，簡単なコードで一万件の特徴ベクトルを保存することができる．例として，2000次元で抽出した特徴ベクトルを以下に示す．

|  |
| --- |
| array([[0.73600566, 0.4391328 , 0.685487 , ..., 0.9568737 , 0.68717253,  0.9354995 ],  [0.5806577 , 0.14781107, 0.7485752 , ..., 0.9273212 , 0.584302 ,  0.9039427 ],  [0.8010622 , 0.7585443 , 0.9483226 , ..., 0.83003205, 0.74461484,  0.67052794],  ...,  [0.7414021 , 0.61791515, 0.7498165 , ..., 0.9560347 , 0.284013 ,  0.8989733 ],  [0.68412614, 0.5127594 , 0.83316875, ..., 0.9662825 , 0.61242753,  0.94350785],  [0.7965311 , 0.4753833 , 0.9745048 , ..., 0.91995025, 0.7687617 ,  0.878242 ]], dtype=float32) |

### 

### 

### 

## 提案システムの実装

### 

### 

### 

### 

# **実験**

## 実験1：写真撮影頻度算出方式に関する実験

### 実験目的

### 実験環境

### 実験方法

### 実験結果

### 実験考察

## 実験2：

### 実験目的

### 実験環境

### 実験方法

### 実験結果

### 実験考察

## 実験考察

．

# **むすび**

# **謝辞**

# **参考文献**