令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」

人工知能（ＡＩ）技術応用教材資料 1

本人工知能（ＡＩ）技術応用教材資料は、文部科学省の教育政策推進事業委託費による委託事業として、一般社団法人全国専門学校情報教育協会が実施した令和７年度「地方やデジタル分野における専修学校理系転換等推進事業」の成果物です。

情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

目　次

人工知能（ＡＩ）技術応用_導入編 3

人工知能（ＡＩ）技術応用_実習１ 9

Part1 9

Part2 20

Part3 28

Part4 35

人工知能（ＡＩ）技術応用_実習２ 43

Part1 43

Part2 53

Part3 62

Part4 70

人工知能（ＡＩ）技術応用_実習 3..... 79

Part1 79

Part2 87

Part3 96

Part4 105

人工知能（ＡＩ）技術応用_導入編



人工知能（AI）技術応用 実習ガイド

人工知能（AI）技術応用

この教材は、『人工知能（AI）技術基礎』で学んだ理論を、実際に手を動かして確認するための**実践教材**です。基礎教材では全7章で構成された内容で、人工知能の理論や仕組みを体系的に学習しました。『**人工知能（AI）技術応用**』では、それらの理論を３つの実習で、機械学習・画像認識・自然言語処理について、**「理解」を「実践」に変える**ことを目指します。

『**人工知能(AI)技術基礎**』（理論・知識の学習）
- 第1章 人工知能の基礎
- 第2章 機械学習の基礎
- 第3章 ディープラーニングの基礎
- 第4章 機械学習と統計学
- 第5章 相関関係と因果関係
- 第6章 データマイニングの基礎
- 第7章 AIとビッグデータ

『人工知能（AI）技術応用』
（実践・体験による理解の深化）
- 実習１ 機械学習で手書き数字を認識
- 実習２ 画像認識AIで犬と猫を分類
- 実習３ 自然言語処理で感情分析AIを作る

人工知能（AI）技術応用 実習１

学習のゴール

人工知能（AI）技術応用

- ✓ AIの「基礎知識」を「使えるスキル」に変える
- ✓ AI開発の基本的なプロセス（データ準備→学習→評価→応用）を体験する
- ✓ 3つの異なる分野（機械学習、画像、自然言語）のAI開発に挑戦する
- ▶ これらによって幅広い実践力を身につける



実習構成

人工知能（AI）技術応用

3つのステップで学ぶ AI実践

実習 1 機械学習

手書き数字認識 (MNIST)
決定木, KNN, ロジスティック回帰

実習 2 画像認識

犬と猫の分類
（CNN, 転移学習）

実習 3 自然言語処理

感情分析
（RNN/LSTM, BERT）

実習で使用する主なツール・環境

人工知能(AI)技術応用

Python:
AI開発の標準言語

Google Colaboratory (Colab): (推奨)
インストール不要のPython実行環境。無料でGPUも使えます

scikit-learn: (実習１で使用)
伝統的な機械学習のためのライブラリ

TensorFlow / Keras: (実習２,３で使用)
ディープラーニングのための主要ライブラリ

Hugging Face Transformers: (実習３で使用)
最新のNLPモデル（BERTなど）を簡単に使うためのライブラリ

ポイント
すべて無料で利用できるツール・環境を選定しています

実習の進め方

人工知能(AI)技術応用

実習	： Part１～４
導入ガイド	： 実習の概要、環境準備、データの入手方法等
オンライン講義（動画）	： 各Partの解説とデモ
ワークブック	： 実際のコードが記載、考察を記入する教材

実習ステップ

人工知能（AI）技術応用

準備	**導入ガイド**を読み、環境を準備する
視聴	**オンライン講義**（動画）を視聴する
実践	**ワークブック**のタスクに沿って、自分でコード実行する
記録	**ワークブック**に実行結果や気づきを記録する
考察	**考察問題**に挑戦し、自分の言葉でまとめる

人工知能（AI）技術応用

「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」

実習１　機械学習で手書き数字を認識
実習２　画像認識AIで犬と猫を分類
実習３　自然言語処理で感情分析AIを作る



NEXT

人工知能（AI）技術応用

「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」

- **実習１　機械学習で手書き数字を認識**
- 実習２　画像認識AIで犬と猫を分類
- 実習３　自然言語処理で感情分析AIを作る



人工知能（ＡＩ）技術応用_実習１

Part1


人工知能（AI）技術応用
「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」
実習１ 機械学習で手書き数字を認識
実習２ 画像認識AIで犬と猫を分類
実習３ 自然言語処理で感情分析AIを作る
Mirai no tobira


人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira

実習１ 機械学習で手書き数字を認識

人工知能（AI）技術応用

実習１ 機械学習で手書き数字を認識 → 実習２ 画像認識AIで犬と猫を分類 → 実習３ 自然言語処理で感情分析AIを作る

☞ **Part１：環境準備とデータ確認**

Part２：3つのアルゴリズム実装

Part３：結果の比較と評価

Part４：実データでの検証



Part １：環境構築とデータ確認 の達成目標

実習１ 機械学習で手書き数字を認識

- Google Colabの基本操作ができる
- MNISTデータセットを読み込み、構造を確認できる
- 訓練データとテストデータの分割の目的を理解できる
- 手書き数字の画像を可視化し、データの特徴を理解できる
- 各数字のデータ分布をグラフで確認できる

Google Colaboratory

実習１ 機械学習で手書き数字を認識 Part１

Google Colab とは？

- ブラウザで Python が実行できる
- インストール不要
- 無料で使える
- Googleアカウントがあれば OK

Google Colabノートブック準備

実習１ 機械学習で手書き数字を認識 Part１

Google Colabにアクセスして、新しいノートブックを作成

1. colab.research.google.com にアクセス
2. 「ノートブックを新規作成」をクリック
3. ノートブックに名前をつける

（例：AI実習1_Part1_自分の名前）

ポイント
Googleアカウントでログインする必要があります

Google Colab 画面構成

実習１ 機械学習で手書き数字を認識 Part 1

Google Colab の画面構成を理解する

- セル：コードを書く場所
- 実行ボタン：▶ マーク （Shift＋Enter でもOK）
- ファイル名の変更方法

Google Colab 動作確認

実習１ 機械学習で手書き数字を認識 Part 1

Pythonを動かしてみよう

```
print("Hello, Machine Learning!")
print("2 + 3 =", 2 + 3)
print("10 * 5 =", 10 * 5)
```

▶ 期待される出力：

```
Hello, Machine Learning!
2 + 3 = 5
10 * 5 = 50
```

ライブラリの読み込み

実習１ 機械学習で手書き数字を認識 Part 1

Python の便利な道具ライブラリをインポート

AI開発で頻繁に利用される標準的なライブラリを読み込みます

```
# 必要なライブラリをインポート
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.model_selection import
train_test_split
```


MNIST

実習１ 機械学習で手書き数字を認識 Part 1

MNISTデータセットとは？

- **MNIST ：Modified National Institute of Standards and Technology**
  - **手書き数字のデータベース**
  - 70,000枚の手書き数字画像
  - 28×28ピクセルの白黒画像
  - 0から9までの数字
  - 機械学習の「練習問題」として世界中で使用

MNISTデータ準備

実習１ 機械学習で手書き数字を認識 Part 1

データをロードし、その構造を確認します

７万枚の手書き数字データ（MNIST）を読み込みます

- データをロードし、その構造（shape）を確認します

```
# MNISTデータのロード（初回のみ時間がかかる場合があります）
X, y = fetch_openml('mnist_784', version=1, return_X_y=True,
as_frame=False, parser='auto')

# データの構造（形状：shape）を確認
print("特徴量データ（X）の形状:", X.shape)
print("正解ラベルデータ（y）の形状:", y.shape)
```


データ準備

実習１ 機械学習で手書き数字を認識 Part 1

訓練データとテストデータ

学習用の「訓練データ」6万枚と、確認用の「テストデータ」1万枚に分けます

種類	枚数	用途
訓練データ	60,000枚	AIに教える
テストデータ	10,000枚	実力を確認

```
# 訓練用6万枚、テスト用1万枚に分割
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=10000,
train_size=60000, random_state=42)

print("訓練データの数:", len(X_train))
print("テストデータの数:", len(X_test))
```

データ可視化

実習１ 機械学習で手書き数字を認識 Part 1

データを見てみよう

実際のデータを画像として16枚表示し、中身を確認します

4×4のグリッドで16枚の数字を表示

- ▶ 各画像の上にラベル（正解）を表示
- ▶ 観察ポイント：
  - ✓ 同じ数字でも書き方が違う
  - ✓ 読みにくい数字もある
  - ✓ 7と1、3と8など似ている数字

```
plt.figure(figsize=(10, 10))
for i in range(16):
    plt.subplot(4, 4, i + 1)
    # 784個の数値を28x28ピクセルの形に戻して表示
    plt.imshow(X_train[i].reshape(28,
28), cmap='gray')
    plt.title(f"Label: {y_train[i]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```


データ可視化

実習１ 機械学習で手書き数字を認識 Part 1

データを見てみよう

各数字が訓練データに何枚ずつ含まれているか棒グラフで確認します

- ▶ 各数字（0-9）の枚数を棒グラフで表示
- ➢ どの数字もほぼ同じ枚数
- ➢ バランスの良いデータセット

```
import collections
# 各ラベルの出現回数をカウント
counts = collections.Counter(y_train)
counts = dict(sorted(counts.items()))
# 棒グラフで表示
plt.bar(counts.keys(), counts.values())
plt.xlabel('Digit')
plt.ylabel('Count')
plt.title('Distribution of digits in training set')
plt.show()plt.show()
```

実習１ Part１のまとめ

実習１ 機械学習で手書き数字を認識 Part１

今回やったこと：

- ✅ Google Colabでプログラムを実行
- ✅ MNISTデータ（70,000枚）を読み込み
- ✅ データを訓練用とテスト用に分割
- ✅ 手書き数字を表示して観察
- ✅ データの統計情報を確認

NEXT

実習１ Part２：3つのアルゴリズム実装

実習１ 機械学習で手書き数字を認識

実習１ Part２では以下を行います

- 🤖 ３つのアルゴリズムを実装
- 📊 決定木
- 📊 k近傍法
- 📊 ロジスティック回帰
- ⚡ それぞれの特徴を比較


人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira


NEXT
人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira

完成イメージ

実習１ 機械学習で手書き数字を認識 Part１

何を作る？

手書き数字認識システム

1. 手書きで数字を書く
2. コンピュータが自動で判断
3. 「これは○です」と表示

人工知能（ＡＩ）技術応用_実習１

Part2


人工知能（AI）技術応用
「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」
実習１ 機械学習で手書き数字を認識
実習２ 画像認識AIで犬と猫を分類
実習３ 自然言語処理で感情分析AIを作る
Mirai no tobira


人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira

実習１ 機械学習で手書き数字を認識

人工知能(AI)技術応用

実習１ 機械学習で手書き数字を認識 → 実習２ 画像認識AIで犬と猫を分類 → 実習３ 自然言語処理で感情分析AIを作る

Part１：環境準備とデータ確認

☞ **Part２：3つのアルゴリズム実装**

Part３：結果の比較と評価

Part４：実データでの検証



Part２：3つのアルゴリズム実装 の達成目標

実習１ 機械学習で手書き数字を認識

- ✓ 3つの代表的な機械学習アルゴリズムの特徴を理解する
- ✓ 決定木 (Decision Tree) を実装できる
- ✓ k近傍法 (KNN) を実装できる
- ✓ ロジスティック回帰 を実装できる
- ✓ 各モデルの「学習」と「予測」のプロセスを体験する
- ✓ 各モデルの学習時間や予測結果の違いを観察する

3つのアルゴリズム概要

実習1 機械学習で手書き数字を認識 Part2

今回試す3つのアルゴリズム

決定木
考え方
「質問」を繰り返す
例
「20の質問ゲーム」
特徴
理由が分かりやすい

k近傍法 (KNN)
考え方
「多数決」で決める
例
「近くの友達に聞く」
特徴
シンプルで直感的

ロジスティック回帰
考え方
「確率」を計算する
例
「天気予報」
特徴
確率が分かる

タスク1：決定木 (Decision Tree)

実習1 機械学習で手書き数字を認識 Part2

「質問」を繰り返して分類するモデル

```
# 1. 決定木モデルのインポート
from sklearn.tree import DecisionTreeClassifier
import time
# 2. モデルの作成（質問は最大10回まで）
dt_model = DecisionTreeClassifier(max_depth=10, random_state=42)
# 3. 学習（訓練データで）
start_time = time.time()
dt_model.fit(X_train, y_train)
print(f"学習時間: {time.time() - start_time:.2f}秒")
# 4. 予測（テストデータで）
dt_predictions = dt_model.predict(X_test)
print("予測:", dt_predictions[:10])
print("正解:", y_test[:10].values)
```

タスク2：k近傍法 (KNN)

実習１ 機械学習で手書き数字を認識 Part2

「近くの仲間」の多数決で分類するモデル

```
# 1. KNNモデルのインポート
from sklearn.neighbors import KNeighborsClassifier
# 2. モデルの作成（近くの5人を参考にする）
knn_model = KNeighborsClassifier(n_neighbors=5)
# 3. 学習（データを覚えるだけなので速い）
start_time = time.time()
knn_model.fit(X_train, y_train)
print(f"学習時間: {time.time() - start_time:.2f}秒")
# 4. 予測
knn_predictions = knn_model.predict(X_test)
print("予測:", knn_predictions[:10])
print("正解:", y_test[:10].values)
```


タスク3：ロジスティック回帰

実習１ 機械学習で手書き数字を認識 Part2

「確率」を計算して分類するモデル

```
# 1. ロジスティック回帰モデルのインポート
from sklearn.linear_model import LogisticRegression
# 2. モデルの作成（計算を100回繰り返す）
lr_model = LogisticRegression(max_iter=100,
random_state=42)
# 3. 学習（最適な確率計算ルールを見つけるため時間がかかる）
start_time = time.time()
lr_model.fit(X_train, y_train)
print(f"学習時間: {time.time() - start_time:.2f}秒")
# 4. 予測
lr_predictions = lr_model.predict(X_test)
print("予測:", lr_predictions[:10])
print("正解:", y_test[:10].values)
```

タスク4：比較と考察

実習１ 機械学習で手書き数字を認識 Part2

3つのモデルを比較しよう

☞ ワークブックの比較表に、結果を記録しましょう

アルゴリズム	学習時間（例）	予測（10件中）
決定木	約 2-3 秒	(各自の結果)
k近傍法 (KNN)	約 1-2 秒	(各自の結果)
ロジスティック回帰	約 30-60 秒	(各自の結果)

考えてみよう
・なぜ学習時間がこんなに違う？ ・最初の10件の予測結果で、一番良かったのは？ ・ どのモデルが一番良さそう？

実習１ Part２のまとめ

実習１ 機械学習で手書き数字を認識 Part2

- ✅ ３つの機械学習アルゴリズム（決定木、KNN、ロジ回帰）を実装した
- ✅ `.fit()` で学習させ、`.predict()` で 予測する流れを学んだ
- ✅ アルゴリズムによって、学習時間や特徴が異なることを体験した

Part２：3つのアルゴリズム実装　人工知能（AI）技術応用 実習１

NEXT

実習１ Part３：結果の比較と評価

実習１ 機械学習で手書き数字を認識

実習１ Part３では以下を行います

- ✓ 10,000件のテストデータで、本当の「正解率」を比較
- ✓ AIが「どんな間違い」をしているか分析する**「混同行列」**
- ✓ **一番優秀なモデルを決定**する

人工知能（AI）技術応用



実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証


NEXT
人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira

人工知能（ＡＩ）技術応用_実習１

Part3


人工知能（AI）技術応用
「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」
実習１ 機械学習で手書き数字を認識
実習２ 画像認識AIで犬と猫を分類
実習３ 自然言語処理で感情分析AIを作る
Mirai no tobira


人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira

実習１ 機械学習で手書き数字を認識

人工知能（AI）技術応用

実習１ 機械学習で手書き数字を認識 → 実習２ 画像認識AIで犬と猫を分類 → 実習３ 自然言語処理で感情分析AIを作る

Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
☞ **Part３：結果の比較と評価**
Part４：実データでの検証



Part３：結果の比較と評価 の達成目標

実習１ 機械学習で手書き数字を認識

- ✓ テストデータ（10,000件）**全体**でモデルの性能を評価する
- ✓ **正解率** (Accuracy) を計算し、客観的に比較する
- ✓ 比較結果をグラフで分かりやすく**「可視化」**する
- ✓ **混同行列** (Confusion Matrix) の見方を理解する
- ✓ **AIが「どんな間違い」をしているか**パターンを分析する
- ✓ データにもとづき、**最適なモデル**を選定する理由を考察する

タスク1：正解率の計算

実習１ 機械学習で手書き数字を認識 Part３

「10,000件のテストデータのうち、何件正解できたか？」を計算

```
# 正解率の計算
from sklearn.metrics import accuracy_score
# 各モデルの正解率を計算
dt_accuracy = dt_model.score(X_test, y_test)
knn_accuracy = knn_model.score(X_test, y_test)
lr_accuracy = lr_model.score(X_test, y_test)
print(f"決定木: {dt_accuracy*100:.2f}%")
print(f"k近傍法: {knn_accuracy*100:.2f}%")
print(f"ロジスティック回帰: {lr_accuracy*100:.2f}%")
```

ポイント
.score() メソッドを使えば一瞬でできます
Part２の「最初の10件」の結果と比べてどうですか？
10,000件のデータで評価する方が、モデルの「本当の実力」が分かります

タスク2：結果の可視化

実習１ 機械学習で手書き数字を認識 Part３

比較グラフの作成／数字の比較は棒グラフにすると一目瞭然

```
import matplotlib.pyplot as plt

models = ['決定木', 'k近傍法', 'ロジスティック回帰']
accuracies = [dt_accuracy, knn_accuracy, lr_accuracy]

plt.bar(models, accuracies, color=['#2ecc71', '#3498db', '#e74c3c'])
plt.title('3つのアルゴリズムの性能比較')
plt.ylabel('正解率')
plt.ylim(0.7, 1.0) # 70%～100%の範囲で表示
plt.show()
```

Part３：結果の比較と評価　人工知能(AI)技術応用 実習１

タスク3：混同行列 (Confusion Matrix)

実習１ 機械学習で手書き数字を認識 Part３

AIがどんな間違い方をしたか を分析する表

混同行列の作成と可視化

実習１ 機械学習で手書き数字を認識 Part３

一番性能が良かった「k近傍法」で見る

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
# k近傍法モデルの予測結果で混同行列を作成
cm = confusion_matrix(y_test, knn_predictions)
# ヒートマップで可視化
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues', xticklabels=range(10),
yticklabels=range(10))
plt.xlabel('予測した数字')
plt.ylabel('実際の数字')
plt.title('混同行列 (k近傍法)')
plt.show()
```

考察ポイント
対角線以外のマスで、数字が大きい（色が濃い）場所を探そう

Part３：結果の比較と評価　人工知能(AI)技術応用 実習１

実習１ Part３のまとめ

実習１ 機械学習で手書き数字を認識 Part３

- ✅ `.score()` でモデルの「正解率」を計算し、客観的に比較した
- ✅ 棒グラフで性能差を「可視化」した
- ✅ `confusion_matrix` で「間違いのパターン」を分析した
- ✅ 今回のデータでは、k近傍法が最も性能が良いことが分かった

NEXT

実習１ Part４：実データでの検証

実習１ 機械学習で手書き数字を認識

実習１ Part４では以下を行います

- ✓ **「自分で書いた数字」でAIをテスト**します
- ✓ 画像のアップロードと**「前処理」の方法を学びます**
- ✓ 学習したAIは、**自筆の文字を正しく認識**できるか？
- ✓ **実習全体のまとめレポート**を作成


人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira


NEXT
人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira

人工知能（ＡＩ）技術応用_実習１

Part4


人工知能（AI）技術応用
「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」
実習１ 機械学習で手書き数字を認識
実習２ 画像認識AIで犬と猫を分類
実習３ 自然言語処理で感情分析AIを作る
Mirai no tobira


人工知能（AI）技術応用
実習１ 機械学習で手書き数字を認識
Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
Part４：実データでの検証
Mirai no tobira

実習１ 機械学習で手書き数字を認識

人工知能（AI）技術応用

実習１ 機械学習で手書き数字を認識 → 実習２ 画像認識AIで犬と猫を分類 → 実習３ 自然言語処理で感情分析AIを作る

Part１：環境準備とデータ確認
Part２：3つのアルゴリズム実装
Part３：結果の比較と評価
☞ **Part４：実データでの検証**



Part４：実データでの検証 の達成目標

実習１ 機械学習で手書き数字を認識

- ✓ 自分で用意した画像（手書き数字）を Google Colabにアップロードできる
- ✓ 画像をAIが認識できる形に「前処理」する方法を学ぶ
- ✓ 学習済みのモデル（Part２,３で作成）を使って、自分の画像を予測させる
- ✓ AIが「なぜ間違えるのか」を考察する
- ✓ 実習１全体の学びを「最終レポート」としてまとめる

タスク1：手書き数字の準備

実習１ 機械学習で手書き数字を認識 Part４

AIにテストさせる「問題」を自分で作る

手順

1. **書く：** 白い紙に、黒いペンで、0～9の数字をはっきりと書く
2. **撮る：** スマホなどで撮影する（影や傾きに注意）
3. **送る：** PCに送信する（メール、クラウドストレージなど）

タスク2：画像の前処理

実習１ 機械学習で手書き数字を認識 Part４

同じ形に揃える

撮影した画像は、AIが学習したMNISTデータと「形」が違うので、
同じ形に揃える必要がある

実行する処理

- ▶ **アップロード**　：Colabに画像をアップロード
- ▶ **読み込み**　：Pythonで画像を読み込む (PILライブラリ)
- ▶ **グレースケール化**：カラー画像を白黒に変換
- ▶ **リサイズ**　：画像を 28×28 ピクセルに縮小
- ▶ **反転**　：「白背景に黒文字」を「黒背景に白文字」（MNISTに合わせる）
- ▶ **正規化**　：0～255の色の濃さを 0～1 の範囲に変換
- ▶ **平坦化**　：28×28の画像を、784個の数字の列に変換

前処理

実習１ 機械学習で手書き数字を認識 Part４

各処理を一括で行う

```
from PIL import Image
import numpy as np
# 1. アップロード (Colabの左サイドバーから手動で)
image_path = "your_image.jpg" # 自分のファイル名に変更
# 2-5. 読込、グレースケール、リサイズ、反転
img = Image.open(image_path).convert('L') # 白黒読込
img_resized = img.resize((28, 28))
img_array = np.array(img_resized)
img_inverted = 255 - img_array # 白黒反転
# 6. 正規化
img_normalized = img_inverted / 255.0
# 7. 平坦化 (予測用)
img_for_prediction = img_normalized.flatten().reshape(1, -1)
plt.imshow(img_normalized, cmap='gray') # 確認用表示
```


タスク3：予測実行

実習１ 機械学習で手書き数字を認識 Part４

AIに予測させてみよう

Part３で一番性能が良かったモデル（k近傍法）などで予測させます

```
# 一番良かったモデル（例：knn_model）で予測
my_prediction = knn_model.predict(img_for_prediction)
print(f"AIの予測結果: {my_prediction[0]}")

# （参考）ロジスティック回帰で確率も見る
lr_prob = lr_model.predict_proba(img_for_prediction)
print(f"AIの自信度 ( {lr_model.predict(img_for_prediction)[0]}である確率 ):
{np.max(lr_prob)*100:.1f}%")
```

タスク4：最終レポートの作成

実習１ 機械学習で手書き数字を認識 Part４

実習１の総まとめとして、分析レポートを作成

レポートに含める内容：

1. **分析の目的：** 何をしようとしたか？（手書き数字認識）
2. **使用データと手法：** MNISTデータ、3つのアルゴリズム（決定木, KNN, ロジ回帰）
3. **分析結果（性能比較）：**
   - 各モデルの正解率（棒グラフ）
   - 混同行列からわかった「間違いやすい数字」
   - どのモデルが一番良かったか？
4. **実データ検証：** 自分の手書き数字での結果（成功したか、失敗したか）
5. **考察と学び：**
   - なぜ性能に差が出た？ なぜ自分の字は間違えた？
   - この実習で学んだこと、難しかったこと、面白かったこと

実習１ まとめ

実習１ 機械学習で手書き数字を認識

今実習で達成したこと

- ✅ **データ分析環境 (Colab) を使いこなした**
- ✅ **70,000件のデータを準備・分析した**
- ✅ **３種類のAIモデルを実装・学習・評価した**
- ✅ **自分のデータでAIをテストした**
- ✅ **機械学習の基本プロセスをすべて体験した**

実習１で身につけたスキル

実習１ 機械学習で手書き数字を認識

- ✓ **Google Colab** と **Python**、**Scikit-learn** を使った分析環境の構築・実行スキル
- ✓ AI開発における**「訓練データ」**と**「テストデータ」**の重要性の理解
- ✓ **決定木、KNN、ロジスティック回帰**という３つの機械学習モデルの実装スキル
- ✓ **.fit()（学習）**と**.predict()（予測）**という基本操作の習得
- ✓ **「正解率」**や**「混同行列」**を使った、AIの性能評価スキル
- ✓ AIの限界と、**学習データ（MNIST）**と、**実データ**（自分の文字）の違いの考察力

NEXT

実習２ 画像認識AIで犬と猫を分類

実習１ 機械学習で手書き数字を認識

実習２では以下を行います

- ☞ 手書き数字より難しい「犬と猫」のカラー写真に挑戦
- ☞ より強力なAI技術**「ディープラーニング(CNN)」**を使います


人工知能（AI）技術応用
「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」
実習１　機械学習で手書き数字を認識
実習２　画像認識AIで犬と猫を分類
実習３　自然言語処理で感情分析AIを作る


NEXT
人工知能（AI）技術応用
「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」
実習１　機械学習で手書き数字を認識
実習２　画像認識AIで犬と猫を分類
実習３　自然言語処理で感情分析AIを作る

人工知能（ＡＩ）技術応用_実習２

Part1


人工知能（AI）技術応用
「基礎から実践へ、3つの実習で学ぶ
機械学習・画像認識・自然言語処理」
実習１　機械学習で手書き数字を認識
実習２　画像認識AIで犬と猫を分類
実習３　自然言語処理で感情分析AIを作る
Mirai no tobira


人工知能（AI）技術応用
実習２　画像認識AIで犬と猫を分類
Part１：データ準備とCNN入門
Part２：CNNモデルの構築と学習
Part３：モデルの評価と改善
Part４：転移学習と実データテスト
Mirai no tobira

実習２ 画像認識AIで犬と猫を分類

人工知能（AI）技術応用

実習１ 機械学習で手書き数字を認識 → **実習２ 画像認識AIで犬と猫を分類** → 実習３ 自然言語処理で感情分析AIを作る

☞ **Part１：データ準備とCNN入門**

Part２：CNNモデルの構築と学習

Part３：モデルの評価と改善

Part４：転移学習と実データテスト



Part １：データ準備とCNN入門 の達成目標

画像認識AIのためのデータ準備

今回の目標

- ✓ 実習データセットを準備する
- ✓ 画像データの前処理方法を理解する
- ✓ 画像認識の心臓部「CNN」の必要性と基本を知る
- ✓ AI学習のための「前処理（正規化・データ拡張）」を行う

画像データとは？

実習２ 画像認識AIで犬と猫を分類 Part１

RGBと3層構造

AIにとって、画像は巨大な「数値の配列（行列）」

- ▶実習１（数字）：白黒（1層）
- ▶**実習２（写真）：カラー（３層：赤・緑・青）**
- ▶各ピクセルは 0～255 の諧調（数値）で表される

ディープラーニング（CNN）

実習２ 画像認識AIで犬と猫を分類 Part１

なぜ CNN なのか？

従来の機械学習（実習1）では、複雑な写真の分類は困難

- ➢ **理由：**写真はずれる、傾く、明るさが変わる
  ピクセル単位の比較では対応できない
- ☞ **解決策：CNN（畳み込みニューラルネットワーク）**
- ✓ **CNNの特徴：**画像から「エッジ」「形」「模様」といった**特徴を自動で抽出**可能

環境準備とデータセット

実習２ 画像認識AIで犬と猫を分類 Part１

環境設定とデータの分割

▶**環境設定**：計算量が多いため、Colab の「**GPU**」を必ず有効に

▶**データセット**：犬と猫の画像（合計 約25,000枚）

▶**データの分割**：

- **訓練データ** (Train)　　：AIが勉強するため用
- **検証データ** (Validation)：勉強の成果のチェック用

Google Colab で GPU設定

実習２ 画像認識AIで犬と猫を分類 Part１

GPU を有効化して学習を高速化します

手順

1. Google Colab で新しいノートブックを作成
2. メニューから「ランタイム」→「ランタイムのタイプを変更」
3. 「ハードウェアアクセラレータ」で「T4 GPU」を選択
4. 「保存」をクリック

```
# GPUが使えるか確認
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
print("GPU available:",
tf.config.list_physical_devices('GPU'))
```

データセットのダウンロード

実習２ 画像認識AIで犬と猫を分類 Part１

Dogs vs Cats データセットをダウンロード

```
# データセットのダウンロード
!wget https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip
!unzip -q cats_and_dogs_filtered.zip
print("✅ ダウンロード完了")
```

▶ データセット：cats_and_dogs_filtered

- train （訓練用）
  - cats （猫の画像 1,000枚）
  - dogs （犬の画像 1,000枚）
- validation （検証用）
  - cats （猫の画像 500枚）
  - dogs （犬の画像 500枚）

画像の確認

実習２ 画像認識AIで犬と猫を分類 Part１

ダウンロードした画像を表示します

```
import matplotlib.pyplot as plt import os from tensorflow.keras.preprocessing
import image
# 画像パスの設定
train_cats_dir = 'cats_and_dogs_filtered/train/cats'
train_dogs_dir = 'cats_and_dogs_filtered/train/dogs'
# 猫と犬の画像を4枚ずつ表示
fig, axes = plt.subplots(2, 4, figsize=(12, 6))
# 猫の画像
for i in range(4): img_path = os.path.join(train_cats_dir, os.listdir(train_cats_dir)[i])
img = image.load_img(img_path) axes[0, i].imshow(img) axes[0, i].set_title('Cat') axes[0, i].axis('off')
# 犬の画像
for i in range(4):
img_path = os.path.join(train_dogs_dir, os.listdir(train_dogs_dir)[i])
img = image.load_img(img_path) axes[1, i].imshow(img) axes[1, i].set_title('Dog') axes[1, i].axis('off')
plt.tight_
layout()
plt.show()
```

Part 1：データ準備とCNN入門　人工知能（AI）技術応用 実習２

前処理①：正規化 (Normalization)

実習２ 画像認識AIで犬と猫を分類 Part１

データの数値を整える

AIが学習しやすくするためにデータの数値を整える

- ▶ **元のデータ：** 0 ～ 255　（範囲が広い）
- ▶ **変換後　　：** **0.0 ～ 1.0**（範囲が狭い）
- ▶ **方法　　：** 全ての値を 255 で割る

メリット
計算が安定し、学習のスピードと精度が向上します

前処理②：データ拡張 (Data Augmentation)

実習２ 画像認識AIで犬と猫を分類 Part１

AIを「応用力のある賢い子」にするための工夫

- ▶ **課題：「過学習（Overfitting）」**
  決まった画像しか認識できなくなる
- ☞ **対策：画像を加工して、擬似的にデータを増やす**
  - 回転させる、ズームする、左右反転させるなど

実装：ImageDataGenerator

実習２ 画像認識AIで犬と猫を分類 Part１

前処理の自動化

Keras の便利な機能を使って、前処理を自動化する

▶画像をフォルダから読み込みながら、リアルタイムで加工してAIに渡す

▶メモリを節約しながら大量のデータを学習できる

実装：ImageDataGenerator

実習２ 画像認識AIで犬と猫を分類 Part１

前処理の自動化

```
from tensorflow.keras.preprocessing.image
import ImageDataGenerator
# 画像の前処理設定
train_datagen = ImageDataGenerator(rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255)
# データジェネレータの作成
train_generator =
train_datagen.flow_from_directory( 'cats_and_dogs_filtered/train', target_size=(150, 150),
batch_size=20, class_mode='binary' )
validation_generator =
validation_datagen.flow_from_directory( 'cats_and_dogs_filtered/validation',
target_size=(150, 150), batch_size=20, class_mode='binary' )
print(f"訓練データ: {train_generator.samples}枚")
print(f"検証データ: {validation_generator.samples}枚")
```

実習１ Part１のまとめ

実習２ 画像認識AIで犬と猫を分類 Part１

今回やったこと：

- ✅ 画像データ（カラー）の仕組みを確認した
- ✅ GPU環境を整えた
- ✅ 実習用の画像データを準備した
- ✅ 「正規化」と「データ拡張」で、学習の準備を整えた

NEXT

実習２ Part２：CNNモデルの構築と学習

実習２ 画像認識AIで犬と猫を分類

実習２ Part２では以下を行います

- ✓ **CNNモデル**をゼロから設計
- ✓ AIを実際に学習させ**「学習曲線」**からその成長を見守る
- ✓ 強力なAIの力の裏にある**「過学習」の課題**に直面する


人工知能（AI）技術応用
実習２ 画像認識AIで犬と猫を分類
Part１：データ準備とCNN入門
Part２：CNNモデルの構築と学習
Part３：モデルの評価と改善
Part４：転移学習と実データテスト
Mirai no tobira


NEXT
人工知能（AI）技術応用
実習２ 画像認識AIで犬と猫を分類
Part１：データ準備とCNN入門
Part２：CNNモデルの構築と学習
Part３：モデルの評価と改善
Part４：転移学習と実データテスト
Mirai no tobira

令和7年度「地方やデジタル分野における専修学校理系転換等推進事業」
情報成長分野の教育プログラム整備と教員育成による学科転換・新設推進事業

人工知能（ＡＩ）技術応用教材資料 1

令和８年２月
一般社団法人全国専門学校情報教育協会
〒164-0003 東京都中野区東中野 1-57-8 辻沢ビル３F
電話：03-5332-5081 FAX.03-5332-5083

●本書の内容を無断で転記、掲載することは禁じます。