４．画像分類、学習済みモデルの活用、tkinter によるインタラクション

https://www.kkaneko.jp/cc/enshu2/index.html

金子邦彦

① 実用的なスキルの習得

② コストと時間の節約

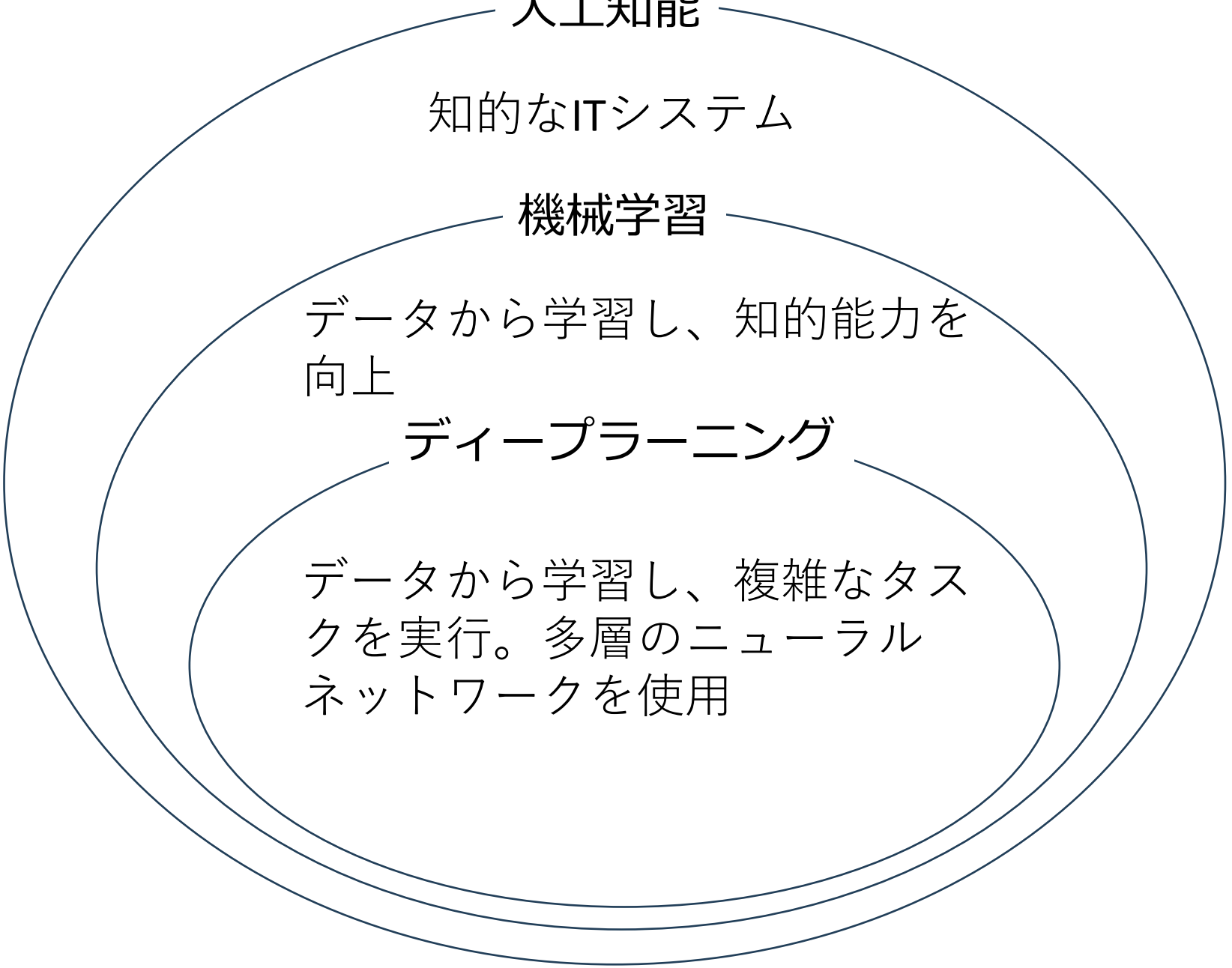
③ プログラミングのスキル向上

④ ユーザビリティの理解

アウトライン

1. イントロダクション
2. ディープラーニングの学習済みモデル、画像分類
3. tkinter の基本的な利用方法

5-1. イントロダクション


人工知能
知的なITシステム
機械学習
データから学習し、知的能力を向上
ディープラーニング
データから学習し、複雑なタスクを実行。多層のニューラルネットワークを使用

Python 言語が広く使用されている理由

文法のシンプルさ

- **Python** は，**直感的で読みやすい**文法
- 例えば、**print** で簡単に**出力**できる、**if** や **else** で**条件分岐**、**for** や **while** で**繰り返し（ループ）**

拡張性

- **多岐にわたる分野で利用**が可能
- 例えば、**関数やクラスを定義**するための **def** や **class**、**継承**や**オブジェクトの属性名と値**を操作するための **super** や **vars** などがある。

柔軟性

- シンプルなスクリプトも、高度なプログラムも作成可能
- **オブジェクト指向**の機能を持ち、**__init__** や **self** のようなキーワードを使用して**クラス**を利用できる。

5-2. ディープラーニングの学習済みモデル、画像分類

ディープラーニングの学習済みモデル

- 大規模データセットで**学習されたモデル**
- ニューラルネットワークの**パラメータ**（重み、バイアスなど）は**最適化済み**
- **特定のタスク**（所定の 1000種類の画像分類など）に**特化**して**最適化**されている
- **転移学習**や**ファインチューニング**を利用して、**異なるタスクにも適用可能**
- **転移学習**や**ファインチューニング**にはそのためのデータが必要。**少量のデータ**でも**効果的に動作する**場合がある

ディープラーニングの学習済みモデルの活用

- **コスト削減**

学習済みモデルは、**大規模データセット**で**学習済み**のため、最初から学習するよりも時間や手間や設備が省ける

- **高い性能**

大規模データセットでの学習により、**優れた性能**を持つ

- **転移学習、ファインチューニング**

学習済みモデルの層を再利用し、異なるタスクに適用することも可能

学習済みモデルの活用シーン

- **迅速な技術検証**

新しい技術を素早く試して性能を確認したい場合。

- **専用モデルの利用**

既存の学習済みモデルが自分のタスクに適している場合、例えば顔検出や姿勢推定など。

- **転移学習、ファインチューニングの導入**

既存の学習済みモデルをもとにして、新しいタスクに適応させるための転移学習、ファインチューニングを予定している場合。

ImageNet-1K で学習済みのモデル

ImageNet-1k

- 数百万枚の画像を含むデータセット。
- **1,000種類のクラス（カテゴリ）**を持つ。
- 日常の物体や生物をはじめとする多様なカテゴリを網羅。

ImageNet-1k で学習済みモデルを画像分類で活用

1. **迅速な技術検証**: 画像分類の最新技術をすばやく試すために、学習済みモデルを利用。
2. **専用モデルの適用**: ImageNet-1kの1,000種類のクラスが、自分の目的とする画像分類タスクと一致する場合、そのままのモデルを使用。
3. **転移学習・ファインチューニング**: ImageNet-1kのクラスが目的のタスクと一致しない場合でも、学習済みモデルをベースにして、転移学習やファインチューニングが可能。

演習 1

【トピックス】

- ImageNet-1K による学習済みモデル
- 最新の画像分類モデル
- 画像分類の実行

パソコンでの実行手順

① 次のページの説明に従って、必要な前準備（Python, timm のインストールなど）を行う

https://www.kkaneko.jp/ai/win/timm.html

② 画像分類プログラムの実行（ソースコードは次ページ）

【機能詳細】

- 使用モデル: **画像分類モデル EVA02**
- 学習データ: **ImageNet-1Kで学習済みのモデル**を利用

【プログラムの説明】

- load_image 関数：指定されたURLから画像をダウンロードし、モデルの入力形式に変換する
- classify_image 関数：画像をモデルで分類し、上位k個のクラスを返す
- メイン: 上の関数を使用してプログラムを構成

```
import timm
import torch
import requests
from PIL import Image
from io import BytesIO

# ImageNet 1kのラベル情報をダウンロード
IMAGENET_1k_URL = 'https://storage.googleapis.com/bit_models/ilsvrc2012_wordnet_lemmas.txt'
IMAGENET_1k_LABELS = requests.get(IMAGENET_1k_URL).text.strip().split('¥n')

def load_image(url, transform):
    # 指定されたURLから画像をダウンロードし、モデルの入力形式に変換する関数。
    image = Image.open(requests.get(url, stream=True).raw)
    image_tensor = transform(image)
    return image_tensor

def classify_image(model, image_tensor, topk=5):
    # 画像をモデルで分類し、トップkのクラスを返す関数。
    output = model(image_tensor.unsqueeze(0))
    probabilities = torch.nn.functional.softmax(output[0], dim=0)
    values, indices = torch.topk(probabilities, topk)
    return [{'label': IMAGENET_1k_LABELS[idx], 'index': idx, 'value': val.item()} for val, idx in zip(values, indices)]

if __name__ == "__main__":
    # モデルを読み込む
    model_name = 'eva02_large_patch14_448.mim_in22k_ft_in1k'
    model = timm.create_model(model_name, pretrained=True).eval()
    transform = timm.data.create_transform(**timm.data.resolve_data_config(model.pretrained_cfg))
    # 画像ファイルの指定
    url = 'https://datasets-server.huggingface.co/assets/imagenet-1k/--/default/test/12/image/image.jpg'
    image_tensor = load_image(url, transform)
    # 画像を分類し、結果を表示
    top_classes = classify_image(model, image_tensor)
    for i in top_classes:
        print(i)
    # 期待される結果
    expected_indices = torch.tensor([162, 166, 161, 164, 167])
    print("Expected Indices:", expected_indices)
```

使用モデル: **画像分類モデル EVA02**
学習データ: **ImageNet-1Kで学習済みのモデル**を利用

分類したい画像の設定

③画像分類結果の確認

- 下記のように「**上位5位**」までの分類結果が表示される
- 5つの結果が表示され、**上位のものほど**AIによる分類の**可能性が高い**と評価されている

```
{'label': 'beagle', 'index': tensor(162), 'value': 0.7920794486999512}
{'label': 'basset, basset_hound', 'index': tensor(161), 'value': 0.009329043328762054}
{'label': 'bluetick', 'index': tensor(164), 'value': 0.0041307127103209496}
{'label': 'Walker_hound, Walker_foxhound', 'index': tensor(166), 'value': 0.0013211222831159 83}
{'label': 'English_foxhound', 'index': tensor(167), 'value': 0.0010351481614634395}
Expected Indices: tensor([162, 166, 161, 164, 167])
```

元画像

5-3. tkinter の基本的な利用方法

Tkinter とは

Tkinter は、 **Pythonの標準ライブラリ**に含まれる**GUIツールキット**

- 主な機能: GUI（グラフィカルユーザインタフェース）の作成
- 利点: GUIを手軽に作成可能

Tkinter の概要

- **ウィンドウの作成**:
  メインウィンドウやダイアログボックスなどを簡単に作成可能
- **ウィジェットの種類**:
  メニュー、ボタン、ラベル、テキストボックス、リストボックス、スライダーなど、多彩なウィジェットを利用できる
- **イベントハンドリング**:
  ユーザーアクション（例: ボタンのクリックやキーボードの操作）に対する応答を実装できる

Trinket のプログラム例

① tkinterモジュールと、ファイルダイアログの**機能をインポート**

```
import tkinter as tk
from tkinter import filedialog
```

② Tk()メソッドを使用して**メインウィンドウ**を作成

```
root = tk.Tk()
```

③ **ファイルダイアログ**と**ラベル**（選択されたファイル名をラベルに表示）の**作成**

```
file_label = tk.Label(root, text="")
file_label.pack()
filepath = filedialog.askopenfilename()
if filepath:
    file_label.config(text=filepath)
```

④ イ**ベントループ**を開始し、ユーザーのアクションを待つ

```
root.mainloop()
```

演習 2

【トピックス】

- コマンドプロンプト
- python の実行
- Tkinter のファイルダイアログ
- Tkinter の画面

メインウィンドウが表示されると同時にファイル選択ダイアログが開きます。選択したファイルのパスがウィンドウ内のラベルに表示されるます。

パソコンでの実行手順

① コマンドプロンプトを開く

② コマンドプロンプトで、次のコマンドを実行

python

③ 次ページのプログラムを実行

④ ファイルダイアログが開くので、ファイルを選択

⑤ ファイル名が表示されるので確認。表示画面は、右上の「x」で閉じることができる。

```
import tkinter as tk
from tkinter import filedialog

root = tk.Tk()

file_label = tk.Label(root, text="")
file_label.pack()
filepath = filedialog.askopenfilename()
if filepath:
    file_label.config(text=filepath)

root.mainloop()
exit()
```

GUI のメリット

- コマンド入力やファイルの編集と比べて、GUIは視覚的で直感的に操作が可能
- ユーザーの使いやすさが向上。

演習 3

【トピックス】

- **ImageNet-1K による学習済みモデル**
- **最新の画像分類モデル**
- **画像ファイルの選択**
- **画像分類の実行**

```
import tkinter as tk
from tkinter import filedialog
import timm
import torch
import requests
from PIL import Image

# ImageNet 1kのラベル情報をダウンロード
IMAGENET_1k_URL = 'https://storage.googleapis.com/bit_models/ilsvrc2012_wordnet_lemmas.txt'
IMAGENET_1k_LABELS = requests.get(IMAGENET_1k_URL).text.strip().split('¥n')

def load_image(file_path, transform):
    # 指定されたファイルパスから画像をロードし、モデルの入力形式に変換する関数。
    image = Image.open(file_path)
    image_tensor = transform(image)
    return image_tensor

def classify_image(model, image_tensor, topk=5):
    # 画像をモデルで分類し、トップkのクラスを返す関数。
    output = model(image_tensor.unsqueeze(0))
    probabilities = torch.nn.functional.softmax(output[0], dim=0)
    values, indices = torch.topk(probabilities, topk)
    return [{'label': IMAGENET_1k_LABELS[idx], 'index': idx, 'value': val.item()} for val, idx in zip(values, indices)]

if __name__ == "__main__":
    # モデルを読み込む
    model_name = 'eva02_large_patch14_448.mim_in22k_ft_in1k'
    model = timm.create_model(model_name, pretrained=True).eval()
    transform = timm.data.create_transform(**timm.data.resolve_data_config(model.pretrained_cfg))
    # 画像ファイルの指定
    root = tk.Tk()
    root.withdraw() # GUIウィンドウを表示しないようにする
    file_path = filedialog.askopenfilename(title="画像ファイルを選択してください", filetypes=[("JPEG files", "*.jpg"), ("PNG files", "*.png"), ("All files", "*.*")])
    if not file_path:
        print("ファイルが選択されませんでした")
        exit()
    image_tensor = load_image(file_path, transform)
    # 画像を分類し、結果を表示
    top_classes = classify_image(model, image_tensor)
    for i in top_classes:
        print(i)
```

使用モデル: **画像分類モデル EVA02**
学習データ: **ImageNet-1Kで学習済みのモデル**を利用

分類したい画像の設定
（GUI による）

画像分類結果

- 下記のように「**上位5位**」までの分類結果が表示される
- 5つの結果が表示され、**上位のものほど**AIによる分類の**可能性が高い**と評価されている

```
{'label': 'lakeside, lakeshore', 'index': tensor(975), 'value': 0.5609492659568787}
{'label': 'seashore, coast, seacoast, sea-coast', 'index': tensor(978), 'value': 0.0835859403014183}
{'label': 'alp', 'index': tensor(970), 'value': 0.036509111523628235}
{'label': 'valley, vale', 'index': tensor(979), 'value': 0.026669658720493317}
{'label': 'sandbar, sand_bar', 'index': tensor(977), 'value': 0.02396279200911522}
```

元画像

全体まとめ ①

ディープラーニングの学習済みモデル

- 大規模データセットで学習済み
- ニューラルネットワークの最適化済みのパラメータ（重み、バイアスなど）
- 特定のタスクに特化して最適化
- 転移学習やファインチューニングで異なるタスクにも適用可能

ディープラーニングの学習済みモデルの活用メリット

- コスト削減: 大規模データセットでの学習により時間や手間が省ける
- 高い性能
- 転移学習やファインチューニングによる再利用と適用

ImageNet-1Kの学習済みモデル

- 数百万枚の画像と1,000種類のカテゴリを持つデータセット
- 日常の物体や生物をはじめとする多様なカテゴリを網羅
- 画像分類の最新技術の試験、専用モデルの適用、転移学習・ファインチューニングの利用が可能

全体まとめ ②

Tkinterの基本

- Pythonの標準ライブラリに含まれるGUIツールキット
- ウィンドウの作成、多彩なウィジェットの利用、ユーザーアクションへの応答などのイベントハンドリングが可能

Trinketによるファイル選択プログラムの例

- tkinterモジュールとファイルダイアログのインポート
- メインウィンドウの作成
- ファイルダイアログとラベルの作成
- イベントループの開始

GUIのメリット

- 視覚的で直感的な操作が可能
- ユーザーの使いやすさが向上

① 実用的なスキルの習得:

学習済みモデルの利点や活用方法を理解し、実際の問題に適用する能力を身につけることができます。これにより、新しい技術の評価、または特定のタスクの実行がスムースに行えるようになります。

② コストと時間の節約

学習済みモデルの利点は、新しいモデルを最初から訓練するよりもコストがかからず、高い性能を持つモデルを利用できることです。

③ プログラミングのスキル向上

Tkinterを使用したGUIの基本的な作成方法を学ぶことで、ユーザーフレンドリーなアプリケーションを開発する能力を向上させることができます。

④ ユーザビリティの理解

GUIの利点を学ぶことで、ユーザビリティの重要性を理解し、これを自身のスキルに取り入れることができます。