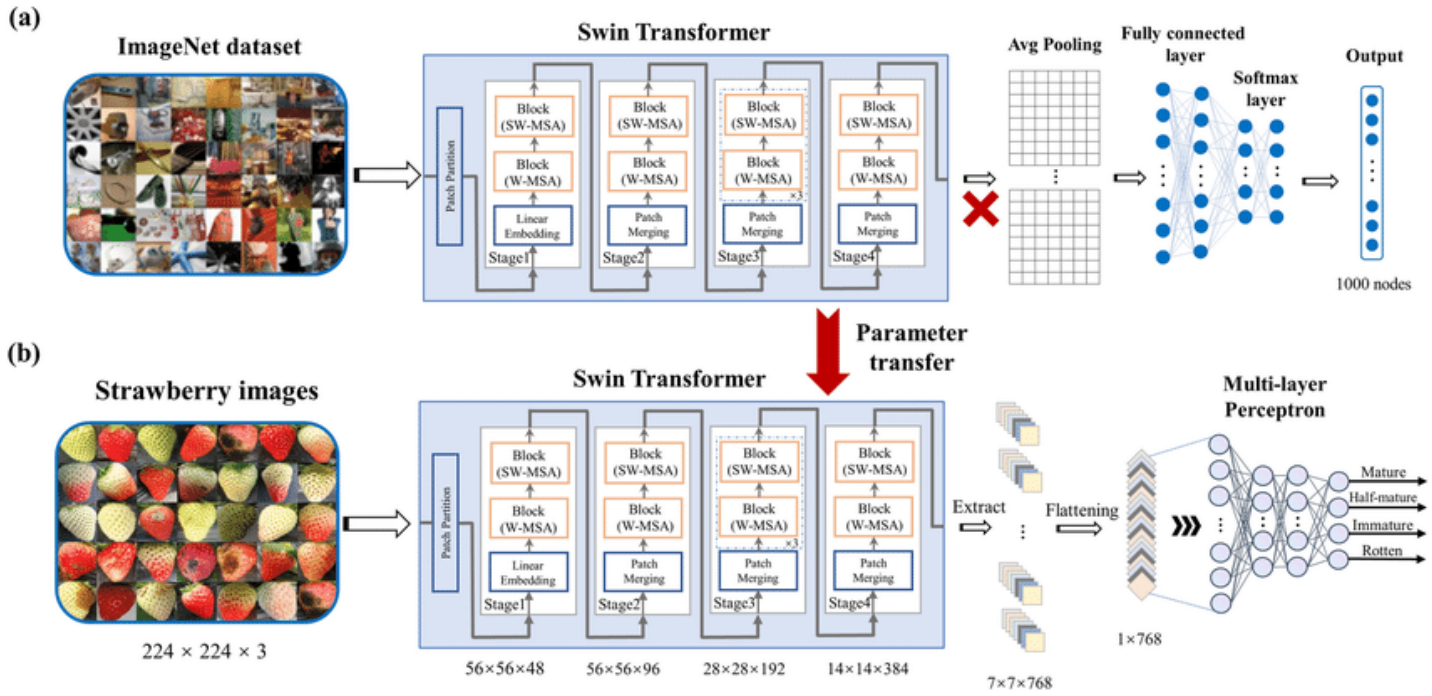
PyTorch

Poziom 1

Klasyfikator w terminalu



1. Utwórz nowy folder w VS Code.

2. Utwórz izolowane środowisko (`Venv`)
   z Python-em 3.12.X (ważne dla kompatybilności z PyTorch!)

3. Utwórz nowy plik, np. `main.py`

4. Wklej do niego kod z kolejnych dwóch slajdów.
   lub pobierz kod z wklejki https://pastebin.pl/view/1461543e

```python
# Importowanie bibliotek
# Torch - głębokie sieci neuronowe (ang. Deep Neural Networks)
import torch
from torchvision import transforms
from PIL import Image
import timm

# Wczytaj model Swin Transformer z biblioteki timm
# ta wersja jest wyuczona na zbiorze ImageNet-21k
nazwa_modelu = "swin_large_patch4_window7_224"
klasyfikator = timm.create_model(nazwa_modelu, pretrained=True)
# klasyfikator.eval()

# Pobierz zbiór ImageNet (1000 klas)
import json
imagenet_labels_url = "https://raw.githubusercontent.com/anishathalye/imagenet-simple-labels/master/imagenet-simple-labels.json"
import requests
labels = json.loads(requests.get(imagenet_labels_url).text)

# Potok przetwarzania obrazu na potrzeby klasyfikatora
preprocess = transforms.Compose([
    transforms.Resize((224, 224)),  # model Swin Transformer bierze na wejściu obrazy 224x224
    transforms.ToTensor(),
    transforms.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),  # Normalizacja
])
```

```python
def recognize_image(img_path):
    """Wczytuje obraz z pliku, przetwarza zgodnie z potokiem preprocess a na koniec klasyfikuje"""
    img = Image.open(img_path).convert("RGB")
    # Dodatkowy wymiar na wejściu
    img_tensor = preprocess(img).unsqueeze(0)

    # Predykcja, czyli przypisanie prawdopodobieństw klas do obrazu wejściowego
    with torch.no_grad():
        logits = klasyfikator(img_tensor)
        probabilities = torch.nn.functional.softmax(logits, dim=-1)
        top5_prob, top5_catid = torch.topk(probabilities, 5)

    # Przeliczenie prawdopodobieństw na etykiety klas
    results = [(labels[catid], prob.item()) for catid, prob in zip(top5_catid[0], top5_prob[0])]
    return results


if __name__ == "__main__":
    img_path = "obrazek.jpg"
    wyjscie_klasyfikatora = recognize_image(img_path)
    for i, (label, confidence) in enumerate(wyjscie_klasyfikatora):
        print(f"{i + 1}: {label} ({confidence:.2f})")
```

5. Próbuj uruchamiać program, doinstalowując wymagane biblioteki na żądanie. Zapewne będzie trzeba zainstalować:
   - `torch`
   - `torchvision`
   - `timm`

   Pozostałe biblioteki prawdopodobnie doinstalują się jako powiązane, ale zachowaj czujność i zwracaj uwagę na komunikaty w terminalu
6. Ostatnim błędem zgłoszonym w terminalu będzie brak obrazka.

   Należy go umieścić w bieżącym folderze i wprowadzić nazwę w odpowiednim miejscu programu. Może być to zdjęcie zrobione telefonem (niekoniecznie kot).

7. Zwróć uwagę na rezultat (numery klas – prawdopodobieństwa).
8. Poeksperymentuj z trzema różnymi obrazkami i przejdź na kolejny poziom.

Flask

web development,
one drop at a time

Poziom 2

Aplikacja webowa

1. Utwórz nowy folder w VS Code.

2. Utwórz izolowane środowisko (`Venv`)
   z Python-em 3.12.X (ważne dla kompatybilności z PyTorch!)

3. Utwórz nowy plik, np. `app.py`

4. Wklej do niego kod z kolejnych trzech slajdów.
   lub pobierz kod z wklejki https://pastebin.pl/view/1df4b651

```python
from flask import Flask, request, render_template, redirect, url_for
import torch
from torchvision import transforms
from PIL import Image
import timm
import json
import requests
import os

app = Flask(__name__)
UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# Folder na załadowane obrazki
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# Wczytaj model Swin Transformer z biblioteki timm
# ta wersja jest wyuczona na zbiorze ImageNet-21k
nazwa_modelu = "swin_large_patch4_window7_224"
klasyfikator = timm.create_model(nazwa_modelu, pretrained=True)
#klasyfikator.eval()

# Pobierz zbiór ImageNet (1000 klas)
imagenet_labels_url = "https://raw.githubusercontent.com/anishathalye/imagenet-simple-labels/master/imagenet-simple-labels.json"
labels = json.loads(requests.get(imagenet_labels_url).text)

# Potok przetwarzania obrazu na potrzeby klasyfikatora
preprocess = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
])
```

```python
def recognize_image(img_path):
    """
    Wczytuje obraz z pliku, przetwarza zgodnie z potokiem preprocess a na koniec klasyfikuje
    """
    img = Image.open(img_path).convert("RGB")
    # Dodatkowy wymiar na wejściu
    img_tensor = preprocess(img).unsqueeze(0)

    # Predykcja, czyli przypisanie prawdopodobieństw klas do obrazu wejściowego
    with torch.no_grad():
        logits = klasyfikator(img_tensor)
        probabilities = torch.nn.functional.softmax(logits, dim=-1)
        top5_prob, top5_catid = torch.topk(probabilities, 5)

    # Przeliczenie prawdopodobieństw na etykiety klas
    results = [(labels[catid], prob.item()) for catid, prob in zip(top5_catid[0], top5_prob[0])]
    return results
```

Następny slajd to część, za którą odpowiada biblioteka `Flask`

```python
@app.route('/')
def index():
    return render_template("index.html")

@app.route('/classify', methods=['POST'])
def classify_image():
    if 'image' not in request.files:
        return redirect(url_for('index'))

    image = request.files['image']
    if image.filename == '':
        return redirect(url_for('index'))

    try:
        # Zapisz wczytany obraz do katalogu UPLOAD_FOLDER
        image_path = os.path.join(app.config['UPLOAD_FOLDER'], image.filename)
        image.save(image_path)

        # Klasyfikacja
        wyjscie_klasyfikatora = recognize_image(image_path)
        return render_template("result.html", image_path=image_path, predictions=wyjscie_klasyfikatora)
    except Exception as e:
        return str(e), 500

if __name__ == "__main__":
    app.run(debug=True)
```

5. Utwórz katalog `templates`
6. Utwórz w tym katalogu plik `index.html`

   Będzie w nim kod źródłowy strony internetowej oferującej interfejs użytkownika w postaci strony www. Kod został rozdzielony na dwa kolejne slajdy dla czytelności, ale jest częścią jednego pliku. Możesz pobrać kod z wklejki https://pastebin.pl/view/c85c9733

   W podanym przykładzie zobaczysz język znaczników HTML i JavaScript, dzięki któremu wczytany obrazek od razu (czyli dynamicznie) zostanie osadzony na stronie, bez konieczności naduszenia jakiegokolwiek przycisku.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Image Classifier</title>
    <style>
        #preview {
            max-width: 500px;
            max-height: 500px;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1>Wczytaj obraz do klasyfikacji</h1>
    <form action="/classify" method="POST" enctype="multipart/form-data">
        <input type="file" name="image" id="imageInput" accept="image/*" required>
        <br>
        <img id="preview" src="#" alt="Image Preview" style="display: none;">
        <br>
        <button type="submit">Klasyfikuj</button>
    </form>
```

```
    <script>
        const imageInput = document.getElementById('imageInput');
        const preview = document.getElementById('preview');

        imageInput.addEventListener('change', function(event) {
            const file = event.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onload = function(e) {
                    preview.src = e.target.result;
                    preview.style.display = 'block';
                };
                reader.readAsDataURL(file);
            } else {
                preview.style.display = 'none';
            }
        });
    </script>
</body>
</html>
```

7. Utwórz w katalogu `templates` kolejny plik `result.html`

   Umieść w nim poniższy kod (możesz skorzystać z wklejki https://pastebin.pl/view/6f0a85e9)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Wyniki klasyfikacji</title>
</head>
<body>
    <h1>Wyniki klasyfikacji</h1>
    <img src="{{ image_path }}" alt="Uploaded Image" style="max-width: 500px; max-height: 500px;">
    <h2>5 najlepszych trafień:</h2>
    <ol>
        {% for label, confidence in predictions %}
        <li>{{ label }}: {{ confidence | round(2) }}</li>
        {% endfor %}
    </ol>
    <a href="/">Powtórz</a>
</body>
</html>
```

8. Uruchom program `app.py`.

Zobaczysz coś w ten deseń:

```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 216-124-670
```

Oznacza to, że w tle na lokalnej maszynie na porcie numer 5000 działa serwer z *usługą webową* i nasłuchuje żądań (ang. *request*)

8. Połącz się z serwerem

wpisując w przeglądarce adres http://127.0.0.1:5000 lub najeżdżając kursorem w terminalu na ten adres, trzymając wciśnięty CTRL i klikając myszką.

Wciśnięcie przy aktywnym terminalu CTRL+C zamyka serwer.