SIGK - Projekt 3
Generacja danych

Autor: Łukasz Dąbała

1 Wymagania projektu

W ramach projektu należy stworzyć program, który będzie realizował opisane w temacie funkcje. Projekt jest zadaniem zespołowym, gdzie każdy zespół składa się z 2 osób.

Głównym językiem programowania powinien być język Python wraz z frameworkiem przeznaczonym do sieci neuronowych: Pytorch.

Za projekt można uzyskać maksymalnie $x \times 10p.$, gdzie x to liczba osób w zespole. Każdy z członków zespołu może dostać maksymalnie 10 punktów.

Ocenie w ramach projektu podlegają:

1. Działanie programu - realizacja funkcji (7 p.)
2. Dokumentacja dokonanych eksperymentów oraz wizualizacja wyników (3 p.)

Projekt uznaje się za oddany w momencie prezentacji go prowadzącemu.

2 Generacja danych

Obecnie zapotrzebowanie na dane treningowe jest ogromne. W związku z tym, w tym ćwiczeniu sprawdzimy możliwości generatywnej sztucznej inteligencji do tworzenia zbiorów danych oraz ich późniejszego wykorzystanie.

Metoda Celem projektu jest stworzenie prostej sieci generatywnej, dzięki której będzie możliwość stworzenia wielu obrazów, które następnie mogą być wykorzystane w innych zadaniach. Żeby sprawdzić jakość generacji i też przydatność wygenerowanych danych, należy też napisać klasyfikator oraz sprawdzić jego skuteczność na różnych kombinacjach:

1. klasyfikator trenowany na danych oryginalnych, testowany na wygenerowanych
2. klasyfikator trenowany na danych wygenerowanych, testowany na oryginalnych
3. klasyfikator trenowany i testowany na połączeniu dwóch zbiorów z odpowiednim ich podziałem

Dataset Zbiorem danych wykorzystanych w tym ćwiczeniu jest MedMNIST[1]. W ramach ćwiczenia należy wykorzystać jeden z jego podzbiorów (należy pracować na obrazkach 64×64):

1. BloodMNIST (łatwy)
2. DermaMNIST

Oryginalne podzbiory są już odpowiednio podzielone na dane treningowe, walidacyjne i testowe.

Przykładowe użycie zbioru danych: https://github.com/MedMNIST/MedMNIST/blob/main/examples/getting_started.ipynb. Do klasyfikacji należy użyć Transfer Learningu z pretrenowanym na ImageNet modelem Resnet50 - do zadania dopasowywana jest ostatnia warstwa w pełni połączona (kod przykładowy poniżej).

```
import torch
import torch.nn as nn
from torchvision import models

class MedModel(nn.Module):
    def __init__(self, num_labels, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # Get ResNet50 model with ImageNet weights
        self.model = models.resnet50(weights='IMAGENET1K_V2')
        num_ftrs = self.model.fc.in_features
        # Fully connected layer with num_labels classes
```

```
        self.model.fc = nn.Linear(num_ftrs, num_labels)

    def forward(self, x):
        return self.model(x)

if __name__ == "__main__":
    model = MedModel(5)
    print(model)
```

Literatura

[1] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.