Dokumentacja Wstępna

Treść zadania

Dla problemu klasyfikacji obrazów (np. zbiór FashionMNIST lub docelowo CIFAR-10 lub CIFAR-100), należy znaleźć podzbiór najbardziej charakterystycznych przykładów z każdej klasy które są wystarczającym do zbudowania poprawnie działającego klasyfikatora. Czyli które zdjęcia są w stanie zapewnić najlepszą możliwą separację?

Zespół

Mateusz Ostaszewski 325203
Michał Sadowski 325221

Algorytmy

- Różne metody wybierania podzbioru:
  - Strategia ewolucyjne $(\mu + \lambda)$ - krzyżowanie, mutacja, selekcja
  - Strategia ewolucyjne (μ, λ) - krzyżowanie, mutacja, selekcja
  - Strategia ewolucyjna $(1 + 1)$ - mutacja
  - Algorytm wspinaczkowy
  - Algorytm wspinaczkowy z tabu (z kolejką FIFO)
- Algorytmy ewolucyjne zaimplementowane z wykorzystaniem biblioteki DEAP.
- Algorytmy wspinaczkowe implementowane samodzielnie.
- Wszystkie eksperymenty będą przeprowadzane na **FashionMNIST**, a na koniec zostanie wykonana ewaluacja dwóch najlepszych technik na **CIFAR-100**.

Reprezentacja

Osobnik reprezentowany jako np.ndarray(dtype=bool) o długości pełnego zbioru danych.

Sąsiedztwo

Sąsiadem osobnika jest dowolny inny osobnik który rózni się pod jednym dowolnym bitem.

Krzyżowanie

Krzyżownaie uniform - Dla każdego bitu losowo wybieramy, od którego rodzica ma być skopiowany.

```
def uniform_crossover(parent1, parent2):
    mask = np.random.randint(0, 2, size=len(parent1), dtype=bool)
    offspring1 = np.where(mask, parent1, parent2)
```

```
    offspring2 = np.where(mask, parent2, parent1)
    return offspring1, offspring2
```

DEAP `tools.cxUniform`

Mutacja

Losowo zmieniamy wybrane bity z True na False lub odwrotnie.

```
def flip_mutation(individual, mutation_rate):
    random_mask = np.random.random(size=individual.shape) < mutation_rate

    individual[random_mask] = ~individual[random_mask]
    return individual
```

DEAP `tools.mutFlipBit`

Selekcja Ruletkowa

```
def roulette_selection(population, fitnesses):
    probabilities = fitnesses / np.sum(fitnesses)
    return population[np.random.choice(len(population), p=probabilities)]
```

DEAP `tools.selRoulette`

Modele

Po przetestowaniu modeli jak radzą sobie z detekcją(2-3 epoki na calym zbiorze treningowym) na podstawie wyników wybraliśmy:

- **Prosty Model własny** dla FashionMNIST:
  - Model z bardzo prostą architekturą sieci konwolucyjnej

```
class SimpleCNN(nn.Module):
def __init__(self):
    super(SimpleCNN, self).__init__()
    self.conv1 = nn.Conv2d(1, 16, kernel_size=3, padding=1)
    self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
    self.fc1 = nn.Linear(32 * 7 * 7, 128)
    self.fc2 = nn.Linear(128, 10)

def forward(self, x):
    x = F.relu(self.conv1(x))
    x = F.max_pool2d(x, 2)
    x = F.relu(self.conv2(x))
    x = F.max_pool2d(x, 2)
    x = x.view(x.size(0), -1)
    x = F.relu(self.fc1(x))
```

```
        x = self.fc2(x)
        return x
```

  - Został wybrany aby szybko testować metody wybierania podzbiorów na mniej złożonym zbiorze danych.
  - Po wstępnych testach otrzymujemy około 90% accuracy na Fashion-MNIST
- **MobileNetV3-Small** dla CIFAR-100:
  - Model z domyślnymi wagami z PyTorch.
  - Modyfikacja pierwszej warstwy, aby obsługiwała obrazy czarno-białe (1 kanał).
  - Zmiana ostatniej warstwy dla liczby klas (100).
  - Po wstępnych testach otrzymujemy około 60% accuracy na CIFAR-100

link do testów

Analiza danych

W projekcie będziemy korzystać ze zbiorów: - FashionMNIST - CIFAR-100

Zbiory te są już podzielone na zbiory treningowe i testowe. Podział ten zachowuje zrównoważenie klasowe. Skorzystamy z tego podziału oraz dodatkowo ze zbioru testowego wydzielmy zbiór walidacyjny.

Funkcja celu

Funkcja celu:

$$J(S) = \alpha \cdot \text{Balanced Accuracy}(S) - \beta \cdot \frac{|S|}{|D|}$$

Początkowe wartości α i β zostały dobrane jako neutralne (1), aby zrównoważyć znaczenie jakości klasyfikacji i rozmiaru podzbioru. W przypadku, gdy jeden ze składników będzie dominował, rozważymy dostrojenie tych parametrów za pomocą eksperymentów.

Metryki

Dodatkowo zastosujemy standardowe metryki, aby zweryfikować, czy funkcja celu nie poszła w skrajność:
- **Balanced Accuracy**.
- **Confusion Matrix**

Sposób podsumowywania wyników

- Każdy eksperyment należy przeprowadzić kilkukrotnie dla każdego algorytmu i wyciągnąć średnią.
- Porównywać metody wybierania podzbiorów, korzystając z krzywej **ECDF** (Empirical Cumulative Distribution Function) - na naszej funkcji loss.
- Raportować wyniki, korzystając z **MlFlow**.