nameAlgorytm

Spis treści

1. Wstęp

Coraz częściej urządzenia internetu rzeczy stają się głównymi urządzeniemi komputerowymi(cytat). Sensory, w które wyposarzone są te urządznia (takie jak aparat, mikrofon, GPU), w połączniu z faktem, że urządznia te są używane codzienie, gromadzą niebywałą ilość, zazwyczaj prywatnych, danych. Modele wyuczone na takich danych dadzą znakomitą poprawę użyteczności jednak ze względu na wrażliwy charakter danych wiąze się z ryzykiem i wysoką odpowiedzialnością ich przechechowywania w scentralizowanej lokalizacji albo nawet całkowitym brakiem dostępu do tych danych.

Urządznia IoT często gromadzą wrażliwe dane i dostęp do takich urządznien przez niewłaściwe osoby grozi nieodwracalnymi stratami dla właściciela urządznia. Nowe urządzania wyposarzone w odpowiednie sensory pozwalają na uwirzytelnienie dostępu nie tylko po haśle ale i przez weryfikacje biometryczną. Zabezpieczenia biometryczne mogą się opierać również na rozpoznawaniu linii papilarnych, głosu, skanowaniu żył, czy też tęczówki lub siatkówki oka. W szeczólności popularnym rozwiązaniem jest weryfikcja urzytkownika przez biometrie twarzy (jakis cytat).

W tej pracy zostanie zbadana metoda uczenia opisana w (cite) w implementacji systemu rozpoznawania twarzy systemu na urządznia IoT.

Główne kontrybucje tej pracy to 1) Implementacja i weryfikacja algorytmu `FederatedAveraging` dla zadań klasyfikacji obrazów oraz weryfikacji twarzy

2. Rozproszone uczenie na urządzeniu końcowym

Federated Learning Problemy odpowiednie do zastosowania federated learningu moją następujące właściwości:

1) Trening na rzeczywistych danych gromadzonych na urządzniach mobilnych dają znaczą przewagę nad treningiem na ogólno dostępnych danych proxy dostępnych w centrach danych.

2) Te dane są prywatne albo są zbyt duże do przetrzywywnia ich w centrach danych

3) Dla zadań nadzorowanych, etykiety danych powstają samoistnie z interakcji użytkownika z użrządzeniem.

Optymalizacja Algorytmy optymalizacji mogące być sastosowane do optymalizajic na urządzniach IoT mają kilka cech wyróżniających je od znanych już algorytmów rozporoszonej optymalizacji:

- **Non-IID** Dane trenujące on danym urządzniu są zazwyczaj zależne od konkretnego użytkownika i dlatego lokalny zbiór danych zebrany na dowolnym urządzeniu nie będzie reprezentatywny w stosunku do dystrybucji całej populacji
- **Niezbalansowany** Podobnie, niektórzy urzytkownicy będą o wiele częściej korzystali z aplikacji aparatu niż inni, co będzie prowadziło do różnic w wielkości zebranych lokalnych zbiorów danych trenujących.
- **Masywnie rozporszony** Spodziewa się, że liczba finalnych użytkowników biorąca udział w optymalizacji będzie większa niż średnia liczba przykładów trenujących przypadająca na jednego klienta.
- **Ograniczona komunikacja** Urządznia IoT są pomimo założenia, że mają dostęp do internetu mogą być ograniczone wolnym albo kosztownym łączem sieciowym.

W tej pracy główna uwaga zostanie poświęcona na rozwiązanie doprowadznie systemu do działania w środowisku danych Non-IID oraz ograniczonej komunkiacji.

2.1. Implementacja

Algorytm 1 opisany został zaimplementowany w języku Python. Do implementacji modeli neuronowych i algorytmów uczących został wykorzystany framework PyTorch [1]. Poprawność implementacji została sprawdzona na zadaniu klasyfikacji obrazów wykorzystując prostą sieć konwolucyją oraz popularny zbiór danych CIFAR10.

Sieć konwolucyjna Jako obiekt treningu omawianego algorytmu zastała użyta niewielka sieć neuronowa zawierająca dwie warstwy konwolucyjne z filtrami o szerokości 5x5 (pierwsza z 32 kanałami, druga z 64, po każdej dodatkowa warstwa 2x2 max pooling), po których następuje dwuwarstowy percepron i na końcu wartswa przeształcenia liniowego, co daje w sumie 10^6 miliona parametrów. Model został modsumowany w tabeli

Algorytm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate (k, w)**:** // *Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

CIFAR10 CIFAR10 jest popularnym syntetycznym zbiorem danych. Zbiór danych składa się z 60 000 kolorowych obrazów podzielonych na 10 klas, z 6000 obrazami przypadającymi na jedną klase. Zawarte są w nim obrazy o szerokości i wyskości 32 pikseli. Standordowo zbiór dzieli się na dwa zbalansowane klasowo podzbiory: testowy i trenigowy zawierających odpowiednio 10000 i 50000 oetykietowanych przykładów. Na rysunku 1 znajduje się 10 losowo wybranych obrazów, dla każdej z 10 klas.

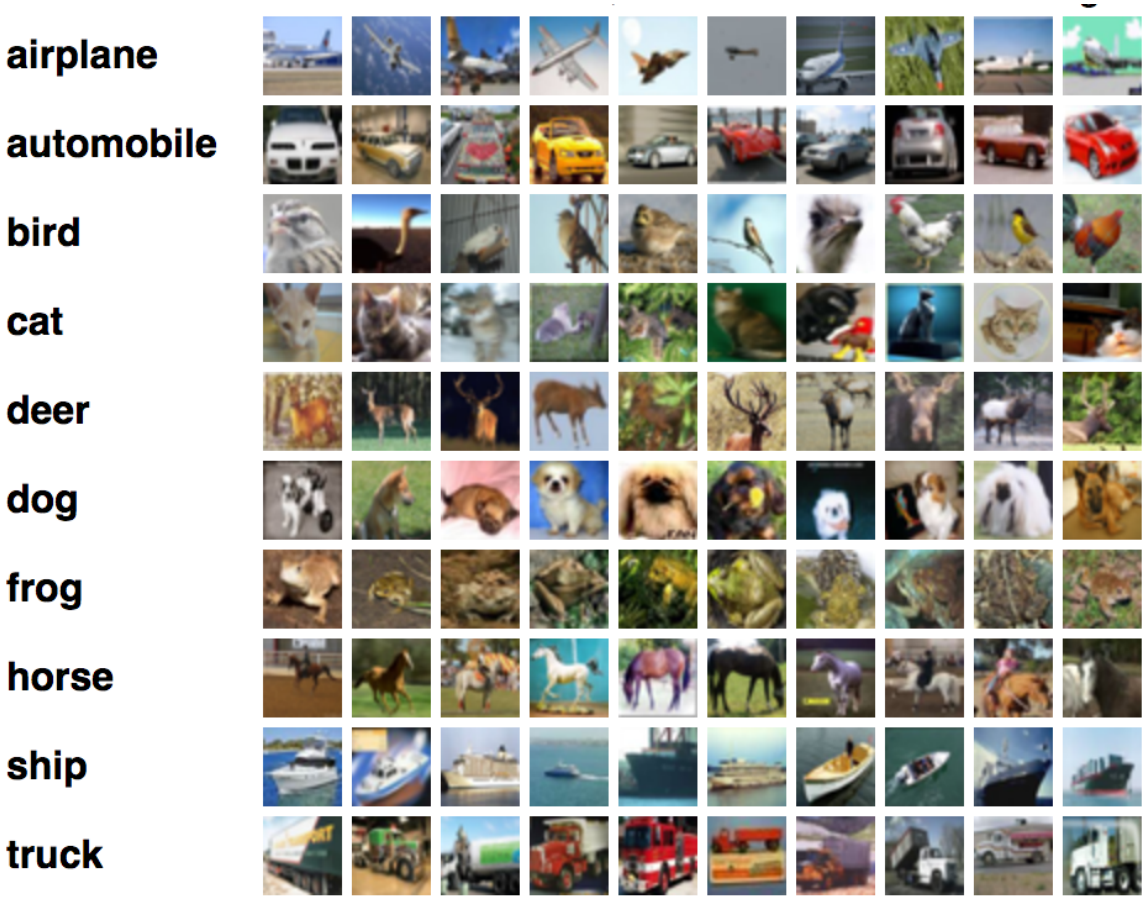
Protokół treningowy Do sprawdzenia poprawności implementacji została zaimplementowana procedura treningowa wzorowana na [2]. Zbiór trengowy został podzielony pomiędzy 100 użytkowników tak żeby każdy zawierał po 500 przykładów trenujących. Z powodu braku naturalnego podziału danych na tak dużą liczbę klientów rozważany jest tutaj nieco mnie wymagający przypadek, w którym dane każdego użytkowanika są zbalansowane oraz IID.

Naszym celem była maksymalizacja dokładności z jaką model klasyfikował obrazy pochodzące ze zbioru testowego. Badanie jakości końcowego modelu globalengo odbywało się już nie w sposób rozproszony, a na serwerze stosując cały dostępny zbiór testowy.

Obrazy uległy standardowemu przetworzeniu wstępnemu, który się składał z losowemu obcinanu obrazów do wielkości 24x24 pikselu, loswemu odbiciu lustrzanemu oraz standardowej normalizacji.

Zaimplementowany algorym został porównany do standardowego algorytmu SGD.

Ewaluacja



Rysunek 1. 10 przykładowych obrazów dla każdej z 10 klas zbioru CIFAR10

3. System weryfikacji użytkownika

Zadaniem systemu jest weryfikacja użytkownika, cz zostały przedstiawione w następnej sekcji.

3.1. Wstępne przetwarzanie obrazu

3.2. Weryfikacja użytkownika

Weryfikacja twarzy jest zadaniem przyrównania twarzy kandydata to innej, i weryfikacja czy nastąpiło ich dopasowanie. Jest to mapowanie jeden-do-jednego: należy sprawdzić czy jest to ta sama osoba.

3.2.1. Procedura weryfikacji

3.2.2. Ekstraktor cech

tutaj opisz jaki jest pipeline wchodzi zdjęcie i wychodzi embedding

3.2.3. Zbiory danych

Są jakieś zbiory daych 1) VggFace2 2) MS1M 3) MegaFace 4) LWF

Ze względu na popularność dwóch ostatnich zbiorów w ewaluacji systemów rozpoznających twarze zostaną one właśnie wykorzystane do ewaluacji, a dwa pierwsze tylko do treningu.

3.2.4. Metody weryikacji twarzy

Wyjściem z modelu jest wektor(embedding), który pozwala na odróżnienie jednej twarzy od drugiej za pomocą porównanie embedingPapier googla twierdzi ze udało im sie uzyskać dobry score jednak nie rzadne inne prace nie raportują sukcesywngeo otrzymania podobnego wyniku. Może to wynikać z 1) braku zbliżonego wielkością zbioru danych użytych do uzyskania wyniku raportowangeo przez googla 2) brakiem podobnych zasobów obliczeniowychu jednej twarzy do embedingu drugiej twarzy. Coś o tym, że są tak jakby dwie rodziny algorytmów. Jedna bazująca na klasycznym podjeściu stosowanym podczas klasyfikacji obrazów i druga bazująca na optymalizacji multi-class classiication hindge loss(jak to do uja wafla przetłumaczyć?).

Papier googla twierdzi ze udało im sie uzyskać dobry score jePapier googla twierdzi ze udało im sie uzyskać dobry score jednak nie rzadne inne prace nie raportują sukcesywngeo otrzymania podobnego wyniku. Może to wynikać z 1) braku zbliżonego wielkością zbioru danych użytych do uzyskania wyniku raportowangeo przez googla 2) brakiem podobnych zasobów obliczeniowychdnak nie rzadne inne prace nie raportują sukcesywngeo otrzymania podobnego wyniku. Może to wynikać z 1) braku zbliżonego wielkością zbioru danych użytych do uzyskania wyniku raportowangeo przez googla 2) brakiem podobnych zasobów obliczeniowych

Tripletloss jest o wiele lepszy dla zastosowań federated learningu procedura trenowania

Classification loss Pokazać że jest kilka rodzajów takich lossów 1) Centre loss 2) Cosine loss 3) Arc loss

Implementacja Papier googla twierdzi ze udało im sie uzyskać dobry score jednak zadne inne prace nie raportują sukcesywngeo otrzymania podobnego wyniku. Może to wynikać z 1) braku zbliżonego wielkością zbioru danych użytych do uzyskania wyniku raportowangeo przez googla 2) brakiem podobnych zasobów obliczeniowych

Została podjęta próba implementacji wytrenowania modelu stosując w.w.y metodę wykorzystując zbiór danych VggFace2.

W celach pokazania, że implementacja jest poprawna porównujemy następujące modele

Porównanie: dwa poejścia: dwa datasety, modele niekoniecznie moje ale testowane na własnej ewaluacji

triplet(google report), wytrenowanych na danych prywatnych triplet(own), centre(external source) wytrenowanych na VggFace2 triplet(own), centre(external source) wytrenowanych na MS1m -> wyjdzie, że triplet loss jest o wiele mniej skuteczna

see this: https://arxiv.org/pdf/1901.08616.pdf https://arxiv.org/pdf/1709.02940.pdf

3.2.5. Rozporszona weryfikacja twarzy

porównanie:

centre loss ms1m

globalny model arcface uczony na ms1m, testowany na lwf i megaface globalny model arcface uczony na ms1m, dotrenowany tripletem na VggFace2, testowany

globalny model arcface dotrenowanie na lwf ale negatywne przykłady samplujemy z generatora globalny model arcface dotrenowanie na megaface ale negatywne przykłady samplujemy z generatora

rozporoszonej optymalizacji:

- **Non-IID** Dane trenujące on danym urządzniu są zazwyczaj zależne od konkretnego użytkownika i dlatego lokalny zbiór danych zebrany na dowolnym urządzeniu nie będzie reprezentatywny w stosunku do dystrybucji całej populacji
- **Niezbalansowany** Podobnie, niektórzy urzytkownicy będą o wiele częściej korzystali z aplikacji aparatu niż inni, co będzie prowadziło do różnic w wielkości zebranych lokalnych zbiorów danych trenujących.
- **Masywnie rozporszony** Spodziewa się, że liczba finalnych użytkowników biorąca udział w optymalizacji będzie większa niż średnia liczba przykładów trenujących przypadająca na jednego klienta.
- **Ograniczona komunikacja** Urządznia IoT są pomimo założenia, że mają dostęp do internetu mogą być ograniczone wolnym albo kosztownym łączem sieciowym.

W tej pracy główna uwaga zostanie poświęcona na rozwiązanie doprowadznie systemu do działania w środowisku danych Non-IID oraz ograniczonej komunkiacji.

Bibliografia

[1] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga i A. Lerer, **?**Automatic Differentiation in PyTorch**?**, w *NeurIPS Autodiff Workshop*, 2017.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson i B. A. y Arcas, *Communication-Efficient Learning of Deep Networks from Decentralized Data*, 2016. arXiv: 1602.05629 [cs.LG].

Wykaz symboli i skrótów

EiTI – Wydział Elektroniki i Technik Informacyjnych
PW – Politechnika Warszawska