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 został zaimplementowany w języku Python. Do implementacji modeli neuronowych i algorytmów uczących został wykorzystany framework PyTorch [1]. Poprawność implementacji została sprawdzona wykorzystując popularny zbiór danych CIFAR10. Model został

Protokół treningowy Do sprawdzenia poprawności implementacji została zaimplementowana procedura treningowa opisana w [2]

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

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

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.

Ewaluacja

3. System weryfikacji użytkownika

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

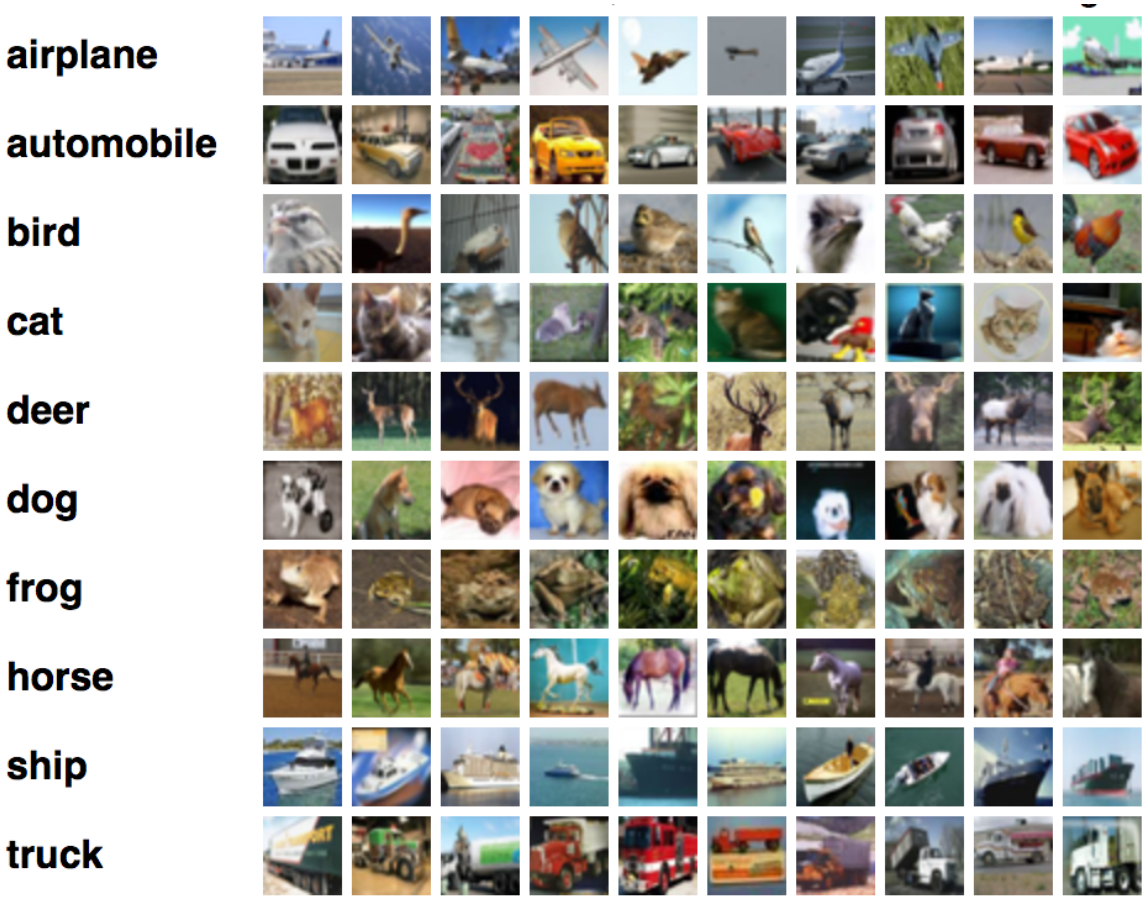
3.1. Wstępne przetwarzanie obrazu

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



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

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