

Spis treści

1. Wstęp	4
2. Rozproszona weryfikacja na urządzeniach IoT	6
2.1. Urządzenia IoT	6
2.2. Biometria twarzy	6
2.3. Wysokopoziomowy schemat systemu	6
3. System weryfikacji użytkownika za pomocą biometrii twarzy	6
3.1. Komercyjne systemy weryfikacji twarzy	6
3.2. Procedura weryfikacji przy wykorzystaniu sieci neuronowych	6
3.3. Wstępne przetwarzanie obrazu	7
3.3.1. Normalizacja obrazu	7
3.3.2. Normalizacja pozy	7
3.4. Neuronowy ekstraktor cech	7
3.4.1. Klasyfikacja twarzy	8
3.4.2. FaceNet	8
3.5. Implementacja systemu weryfikacji twarzy	8
3.5.1. Zbiory danych	8
3.5.2. Ewaluacja	9
3.5.3. Wyniki	9
4. Federated Learning	10
4.1. Projekt systemu	11
4.1.1. Protokół	11
4.1.2. Urządzenie IoT	12
4.1.3. Serwer	12
4.2. Algorytm optymalizacji	12
4.3. Implementacja optymalizacji FL	12
4.3.1. Sieć konwolucyjna	12
4.3.2. Zbiory danych	13
4.3.3. Protokół treningowy	13
4.3.4. Ewaluacja	14
5. Protokołu treningowy FL do zadania weryfikacji twarzy	15
5.1. Protokół treningowy	15
5.2. Generowanie przykładów negatywnych	15
5.2.1. StyleGAN	15
5.2.2. Przykładowe twarze	15
5.3. Zbiory danych	16
5.4. Eksperymenty	16
5.4.1. Metoda pre-treningu modelu	16
5.4.2. Online Hard sample mining	16

0. Spis treści

6. Podsumowanie	17
Bibliografia	18
Wykaz symboli i skrótów	19

1. Wstęp

Coraz częściej urządzenia internetu rzeczy stają się głównymi urządzeniami komputerowymi coraz większej liczby użytkowników (cite). Często gromadzą one wrażliwe dane i dostęp do takich urządzeń przez niewłaściwe osoby grozi nieodwracalnymi stratami dla ich właściciela. Nowe przyrządy wyposażone w odpowiednie sensory pozwalają na uwierzytelnienie dostępu już nie tylko za pomocą hasła ale również przez weryfikację biometryczną. Zabezpieczenia biometryczne mogą się opierać na rozpoznawaniu linii papilarnych, głosu, skanowaniu żył, czy też tęczówki lub siatkówki oka. W szczególności popularnym rozwiązaniem jest weryfikacja użytkownika przez biometrię twarzy (jakiś cytat).

Metody weryfikacji twarzy (ją jakieś) wymagają wyspecjalizowanych i dokładnych kamer (need fact check). Z jednej strony montowanie drogich i nowoczesnych kamer na urządzeniach IoT do celów weryfikacji jest nieopłacalne finansowo, a z drugiej z perspektywy użytkownika pożądane jest posiadanie możliwie dokładnego systemu weryfikacji dostępu. Najnowocześniejsze i najbardziej dokładne metody bazują w całości lub przynajmniej części na bazie sieci neuronowych (cite,fact check). Metody te pewnym stopniu niwelują potrzebę posiadania wyspecjalizowanych kamer jednak dokładność nowoczesnych metod jest bardzo uzależniona od jakości i ilość danych, które posłużyły do wytrenowania sieci neuronowej.

Sensory, w które wyposażone są te urządzenia (na przykład aparat, mikrofon, itp), w połączeniu z faktem, że są używane codziennie, gromadzą niebywałą ilość, zazwyczaj prywatnych, danych. Modele wyuczone na takich danych dają znakomitą poprawę ich użyteczności jednak ze względu na wrażliwy charakter danych wiąże się z ryzykiem i wysoką odpowiedzialnością ich przechowywania w centralizowanej lokalizacji albo nawet całkowitym brakiem dostępu do tych danych.

Federated Learning pozwala na bezpieczne dla użytkownika wykorzystanie jego prywatnych danych (zdjęć) w celu dotrenowania sieci neuronowych i w tym poprawy ich jakości. W tej pracy zostanie zbadana metoda ta metoda uczenia sieci neuronowych w implementacji systemu rozpoznawania twarzy systemu na urządzenia IoT.

Praca została podzielona na rozdziały. W rozdziale XXX została przedstawiona ogólny zarys implementowanego systemu. Opisujemy czym charakteryzują się urządzenia IoT się, na czym polega zadanie weryfikacji użytkownika i pokazujemy, że zastosowanie podejścia Federated Learningu do tego zadania jest odpowiednie.

Następnie w rozdziale XXX w szczegółach zostaje opisany problem weryfikacji użytkownika na podstawie cech biometrycznych jego twarzy. Omawiane zostają dwa współczesne podejścia wykorzystujące głębokie sieci neuronowe, ich wady i zalety względem wykorzystania w FL. Prezentujemy dokładniej wybrane przez nas podejście wraz z naszą implementacją i otrzymanymi wynikami.

Rozdział XXX został poświęcony na omówienie Federated Learningu. Opisujemy

cechy środowiska, w którym zastosowanie tej metody treningu sieci daje największy zysk, prezentujemy projekt całego systemu zaczynając od protokołu komunikacji, założen co do urządzeń końcowych oraz serwera. Omawiamy w algorytm Federated Averaging, jego implementacje oraz walidacje implementacji na syntetycznym zbiorze danych MNIST oraz CIFAR10. Ostatecznie prezentujemy zastosowanie FL do dotrenowania ekstraktora cech w rozproszonym zbiorze danych.

Na końcu w rozdziale XXX zostało zawarte podsumowanie tej pracy wraz z propozycją jej dalszego rozwoju.

Główne kontrybucje tej pracy to 1) Implementacja i weryfikacja algorytmu FL dla zadań klasyfikacji obrazów oraz weryfikacji twarzy. 2) ???

2. Rozproszona weryfikacja na urządzeniach IoT

2.1. Urządzenia IoT

2.2. Biometria twarzy

2.3. Wysokopoziomowy schemat systemu

3. System weryfikacji użytkownika za pomocą biometrii twarzy

3.1. Komercyjne systemy weryfikacji twarzy

3.2. Procedura weryfikacji przy wykorzystaniu sieci neuronowych

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

Mając na wejściu systemu dwa zdjęcia x_i oraz x_j wynik weryfikacji będzie wyznaczany następująco:

$$\text{weryfikacja jest} \begin{cases} \text{pozytywna,} & \text{jeśli } d(f(x_i), f(x_j)) < \alpha \\ \text{negatywna,} & \text{w.p.p} \end{cases} \quad (1)$$

gdzie $f(x)$ jest wektorem cech obrazu x , opisany w sekcji 3.4, α jest marginesem, $d(f_1, f_2)$ jest pewną funkcją dystansu liczoną pomiędzy wektorami wyznaczanymi przez $f(x)$.

3.3. Wstępne przetwarzanie obrazu

3.3.1. Normalizacja obrazu

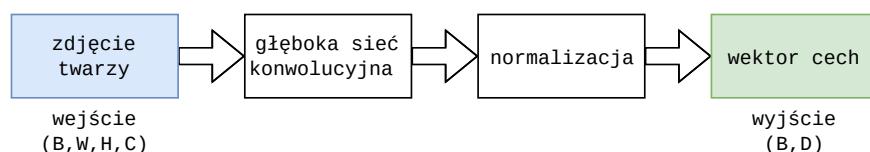
3.3.2. Normalizacja pozycji



Rysunek 1. Przykłady działania ekstraktora twarzy. Pokazane zostały tutaj wybrane przykłady prezentujące działanie ekstraktora twarzy.

3.4. Neuronowy ekstraktor cech

We współczesnych systemach weryfikacji wykorzystuje się głębokie sieci konwolucyjne. W szczegółach zostaną omówione używana przez nas architektura w sekcji XXX. Pomijając szczegóły modelu i traktując go jako czarną skrzynkę ogólna idea ekstraktora została pokazana na Rysunku 2. Głównym założeniem jest stworzenie systemu end-to-end, którego rezultatem działania będzie embedding $f(x)$, wyznaczony z obrazu wejściowego x przez rzutowanie go do pewnej przestrzeni cech \mathbb{R}^d , w taki sposób, że pewna funkcja odległości wyznaczona dla wszystkich zdjęć twarzy jest mała dla twarzy należących do tych samych osób i duża dla różnych twarzy.



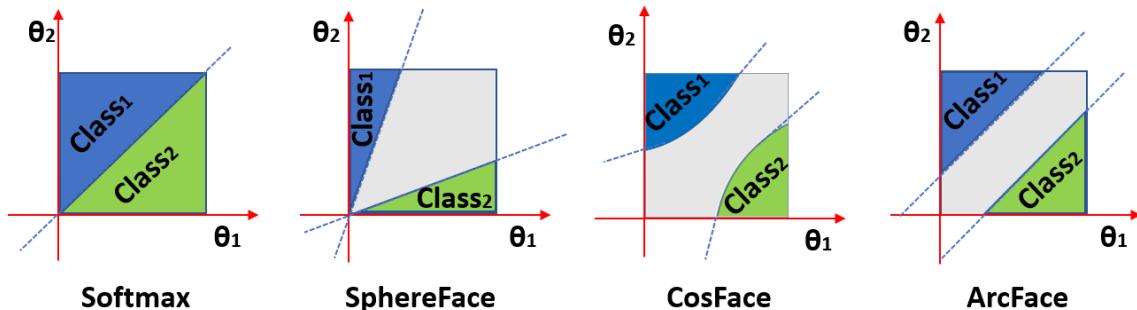
Rysunek 2. Struktura ekstraktora cech. Ekstraktor składa się z wejścia, które w ogólnym przypadku może być paczką B przetworzonych wstępnie obrazów twarzy o wymiarach $W \times H \times C$, głębokiej sieci konwolucyjnej i następującej po niej warstwie normalizacji. W rezultacie na wyjściu otrzymujemy B wektorów cech o wymiarach D .

W literaturze można znaleźć dwie rodziny algorytmów trenujących, jedna wzorująca się na klasycznym podejściu stosowanym podczas treningu klasyfikatorów obrazów. Najnowsze podejścia z tej rodziny algorytmów prezentujemy w sekcji 3.4.1. Drugi rodzaj

3. verification

algorytmów bazują na optymalizacji multi-class classification hinge loss (po pl?). Opisujemy je w szczegółach w sekcji 3.4.2

3.4.1. Klasyfikacja twarzy



Rysunek 3. Porównanie marginesów decyzyjnych

Softmax[1]

SphereFace

CosFace[2]

ArcFace[3]

3.4.2. FaceNet

3.5. Implementacja systemu weryfikacji twarzy

3.5.1. Zbiory danych

W literaturze znajduje się duża liczba zbiorów do treningu ekstraktorów cech twarzy(cytaty). Popularnym wyborem jest zbiór MS1M, posiada on dużą różnorodność twarzy oraz dużą liczbę przykładów trenujących. VggFace2 jest relatywnie nowym zbiorem i nie tak popularnym jak poprzedni ale cechuje się nie tylko dużym rozmiarem ale i dużą różnorodnością wewnętrz-klasową.

LFW jest jest niejako standardem w testowaniu systemów do weryfikacji, jednak ze względu na jego niewielkie rozmiary i niską różnorodność wewnętrz-klasową nie będzie odpowiedni do testowania systemu opartego o FL dlatego jego zastosowanie ograniczymy do walidacji pre-trenowanych modeli trenowanych w tradycyjnym sposobem na serwerze.

Papier googla twierdzi że udało im się uzyskać dobry score jednak zadne inne prace nie raportują skutecznego otrzymania podobnych rezultatów. Może to wynikać

- braku zbliżonego wielkością zbioru danych użytych do uzyskania raportowanych wyników
- brakiem podobnych zasobów obliczeniowych

Zbiór danych	# osób	# zdjęć	# zdjęć na osobę	aligned
VGGFace2-train [4]	9.1K	3.3M	80/362.6/843	Nie
MS1M-DeepGlint [5]	87K	3.9M	?/44.8/?	Tak
LFW [6]	5,749	13,233	1/2.3/530	Nie

Tabela 1. Zbiory danych.

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

3.5.2. Ewaluacja

3.5.3. Wyniki

4. Federated Learning

W tym rozdziale przedstawiony został projekt systemu, który korzystając z podejścia Federated Learning (FL) umożliwi na trening ekstraktora cech twarzy w rozproszonym środowisku urządzeń IoT.

Federated Learning Federated Learning (FL) jest podejściem zastosowania uczenia maszynowego w rozproszonym środowisku pozwalające na trening modeli on dużym zbiorze danym zdecentralizowanych prywatnych danych znajdujących się na urządzeniach końcowych użytkowników, a w szczególności urządzeniach Internetu Rzeczy. FL realizuje ideę "przyniesienie kodu do danych, zamiast danych do kodu" i adresuje fundamentalne problemy prywatności, własności i lokalności danych. Federated learning został opisany w (cite). Problemy odpowiednie do zastosowania federated learningu mają następujące właściwości:

- Trening na rzeczywistych danych gromadzonych na urządzeniach mobilnych dają znaczącą przewagę nad treningiem na ogólnie dostępnych danych proxy dostępnych w centrach danych.
- Te dane są prywatne albo są zbyt duże do przechowywania ich w centrach danych
- Dla zadań nadzorowanych, etykiety danych powstają samoistnie z interakcji użytkownika z urządzeniem.

Cechy środowiska systemu Algorytmy optymalizacji mogące być zastosowane do optymalizacji na urządzeniach IoT mają kilka cech wyróżniających je od znanych już algorytmów rozproszonej optymalizacji:

- **Non-IID** Dane trenujące on danym urządzeniu są zazwyczaj zależne od konkretnego użytkownika i dlatego lokalny zbiór danych zebrany na dowolnym urządzeniu nie będzie reprezentatywny w stosunku do dystrybucji całej populacji
- **Niezbalsansowany** Podobnie, niektórzy użytkownicy będą o wiele częściej korzystali z aplikacji aparatu niż inni, co będzie prowadziło do różnic w wielkości zebranych lokalnych zbiorów danych trenujących.
- **Masywnie rozproszony** Spodziewa się, że liczba finalnych użytkowników biorących udział w optymalizacji będzie większa niż średnia liczba przykładów trenujących przypadająca na jednego klienta.
- **Ograniczona komunikacja** Urządzenia 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 doprowadzenie systemu do działania w patologicznym przypadku środowiska danych Non-IID oraz ograniczonej i zawodnej komunikacji.

4.1. Projekt systemu

Implementowany system umożliwia trening głębszych sieci neuronowych on danych przetrzymywanych na bezpośrednio urządzeniu IoT. Dane te nigdy nie opuszczają samego urządzenia. Wagi modelu są agregowane i łączone na serwerze znajdującym się w chmurze za pomocą algorytmu Federated Averaging konstruując nowy model globalny, który zostaje przesłany z powrotem do urządzeń końcowych do inferencji. System ten został opisany pierwotnie w (cytat) i z sukcesem zaaplikowany w implementacji inteligentnej klawiatury smartphone. (cite blogaska googla).

Przedstawienie architektury systemu zaczynamy od przedstawienia protokołu według, którego przebiega cały przepływ danych. Opisujemy to w następnej sekcji.

4.1.1. Protokół

Uczestnikami protokołu są urządzenia Internetu Rzeczy oraz serwer, który jest usługą chmurową. Urządzenia anonsują serwerowi swoją gotowość do uruchomienia zadania. Zadanie jest specyficzne dla populacji urządzeń i polega na lokalnym uruchomieniu obliczeń, takich jak trening z zadanymi przez serwer parametrami albo ewaluacja wytrenowanego modelu na lokalnych dla urządzenia danych.

Protokół dzieli komunikacje na rundy, z której każda zaczyna się od selekcji. Z potencjalnie tysiący gotowych urządzeń w danym oknie czasowym, serwer wybiera ich podzbior, który dostanie zaproszenia do wzięcia udziału obliczeniach. Liczebność tego podzbioru jest parametrem algorytmu opisywanego w kolejnym rozdziale

Serwer mówi wybranym urządzeniom jakiego typu obliczenia będą wykonywane. Wraz z tą informacją przesyłany graf obliczeniowy modelu, zserializowane parametry globalnego modelu oraz pozostałe informacje niezbędne do uruchomienia obliczeń w danej rundzie. Następnie każdy z uczestników wykonuje lokalnie obliczenia bazując na globalnym stanie oraz swoim lokalnym zbiorze danych i przesyła wynik obliczeń z powrotem na serwer. W szczególności wynikiem, w przypadku rundy treningowej, będą wytrenowane parametry modelu lokalnego. Serwer wciela zebrane aktualizacje w swój stan globalny co kończy pojedyńczą rundę i proces ten się powtarza.

Opisywany protokół pozwala urządzeniom końcowym na poprawianie globalnego, pojedyńczego modelu pomiędzy rundami, z których każda składa się z trzech faz co zostało pokazane na rysunku (rysunek?).

Selekcja Cyklicznie, urządzenia, które spełniają wymagane kryteria (Patrz sekcja XXX) meldują serwerowi swoją gotowość. Serwer wybiera podzbior z dostępnych mu urządzeń. W naszej implementacji wybrano prostą, losową selekcję z rozkładem jednostajnym N urządzeń z tych, które ogłosili swoją gotowość.

Konfiguracja Serwer jest skonfigurowany na podstawie wybranego mechanizmu selekcji urządzeń oraz agregacji modeli. Urządzenia końcowe są samo-konfigurowane według

4. federated

przesłanej przez serwer informacji o typie obliczeń, grafu obliczeniowego oraz stanu globalnego.

Raportowanie Serwer czeka, aż partycypujące urządzenia zdadzą raport z wynikiem przeprowadzonych obliczeń. Jeśli runda była rundą treningową to wraz z otrzymaniem wszystkich zaktualizowanych modeli, serwer agreguje je z użyciem algorytmu Federated Averaging i informuje rapportujące urządzenia kiedy mogą ponownie zgłosić swoją gotowość. Jeśli wystarczająca liczba urządzeń zaraportuje wyniki serwerowi w pewnym oknie czasowym, runda zostanie uznana za zakończoną z powodzeniem i zagregowany model zastąpi aktualny model globalny. W przeciwnym przypadku zebrane wyniki z tej rundy zostają porzucone. Protokół jest w pewnym stopniu odporny na to, że partycypujące urządzenia z pewnych powodów nie odpowiedzą na przeslaną konfigurację albo nie zaraportują wyników.

Fazy selekcji i raportowania są dospecyfikowane przez zestaw parametrów. Podczas fazy selekcji serwer bierze pod uwagę pożądaną liczbę uczestników rundy, czasy time-outów oraz minimalną liczbę potrzebnych uczestników do rozpoczęcia rundy. Faza selekcji trwa dopóki nie zostanie wybrana docelowa liczba urządzeń albo nie skończy się czas przeznaczony na fazę selekcji. W drugim przypadku runda zostanie rozpoczęta jeśli została zebrana minimalna liczba urządzeń, w przeciwnym przypadku runda zostaje porzucona. Faza raportowania parametryzowana jest przez minimalną liczbę urządzeń, które muszą zaraportować wynik.

4.1.2. Urządzenie IoT

4.1.3. Serwer

4.2. Algorytm optymalizacji

Algorytm 1 opisany został zaimplementowany w języku Python. Do implementacji modeli neuronowych i algorytmów uczących został wykorzystany framework PyTorch [7]. Poprawność implementacji została sprawdzona na zadaniu klasyfikacji obrazów wykorzystując prostą sieć konwolucyjną oraz popularny zbiór danych CIFAR10.

4.3. Implementacja optymalizacji FL

4.3.1. Sieć konwolucyjna

Jako obiekt treningu omawianego algorytmu zastała użyta niewielka sieć neuronowa zawierająca dwie warstwy konwolucyjną z filtrami o szerokości 5x5 (pierwsza z 32 kanałami, druga z 64, po każdej dodatkowa warstwa 2x2 max pooling), po których następuje dwu-warstwowy perceptron i na końcu warstwa przekształcenia liniowego, co daje w sumie 10^6 miliona parametrów. Model został podsumowany 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

```

4.3.2. Zbiory danych

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 klasę. Zawarte są w nim obrazy o szerokości i wysokości 32 pikseli. Standardowo zbiór dzieli się na dwa zbalansowane klasowo podzbiory: testowy i treningowy zawierających odpowiednio 10000 i 50000 etykietowanych przykładów. Na rysunku 4 znajduje się 10 losowo wybranych obrazów, dla każdej z 10 klas.

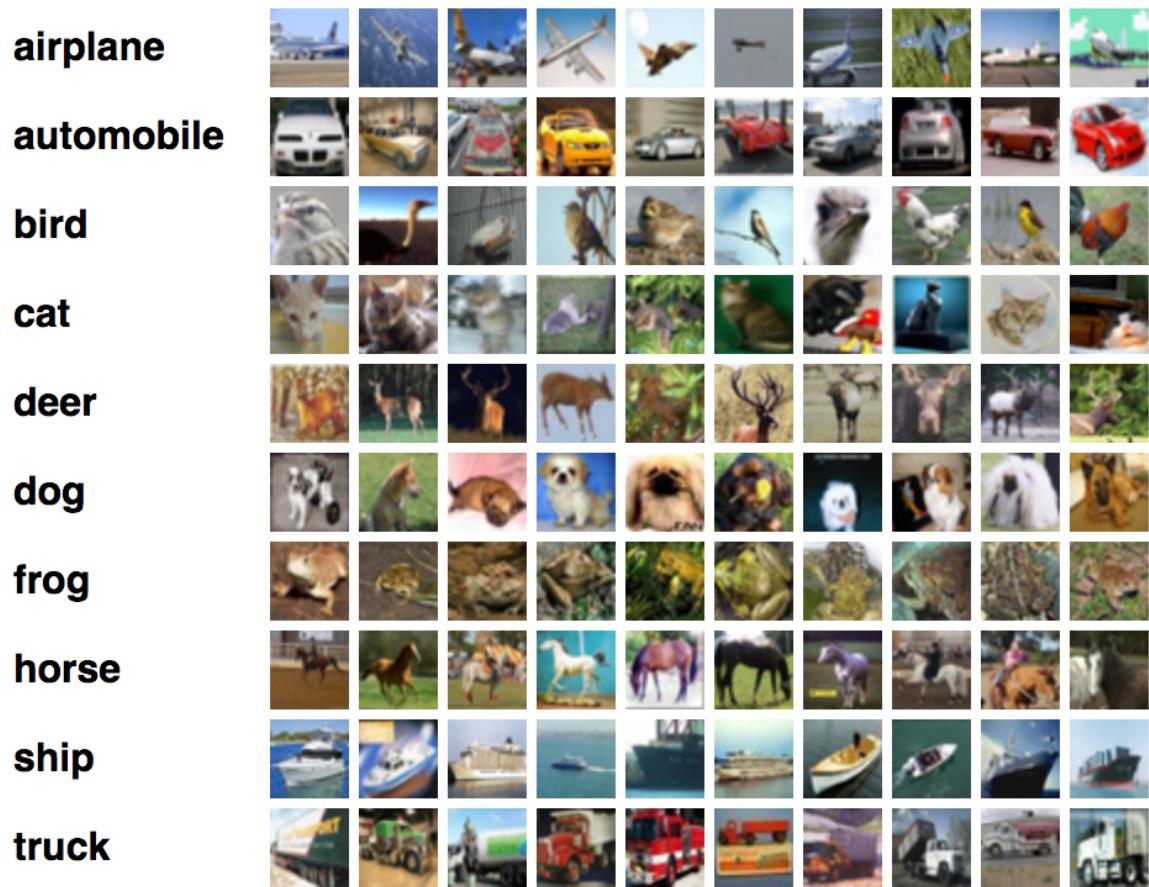
4.3.3. Protokół treningowy

Do sprawdzenia poprawności implementacji została zaimplementowana procedura treningowa wzorowana na [8]. Zbiór treningowy 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 mniej wymagający przypadek, w którym dane każdego użytkowania są zbalansowane oraz równomiernie rozdysytrybuowane.

Naszym celem była maksymalizacja dokładności z jaką model klasyfikował obrazy pochodzące ze zbioru testowego. Badanie jakości końcowego modelu globalnego odbywało się już nie w sposób rozproszony, a na serwerze stosując cały dostępny zbiór testowy, co zostało umożliwione dzięki wprowadzonym uproszczeniom co do rozkładu danych.

Obrazy uległy standardowemu przetworzeniu wstępemu i augmentacji. Przykłady trenujące zredukowano do wielkości 24x24 pikseli przez losowe obcięcie krawędzi, obrazy uległy losowemu horyzontalnemu odbiciu lustrzanemu oraz standardowej normalizacji.

Zaimplementowany algorytm został porównany do standardowego algorytmu SGD (cytat).



Rysunek 4. 10 przykładowych obrazów dla każdej z 10 klas zbioru CIFAR10

4.3.4. Ewaluacja

	Acc.	80%	82%
SGD	18000	31000	
FedAvg	280 (64.3×)	630 (49.2×)	
SGD-nasz	20500	45000	
FedAvg-nasz	510 (40.2×)	1200 (37.5×)	

Tabela 2. Liczba aktualizacji modelu globalnego i przyspieszenie względem podstawowego SGD potrzebna do otrzymania zadanej dokładności (Acc) na testowym zbiorze CIFAR10.

5. Protokołu treningowy FL do zadania weryfikacji twarzy

5.1. Protokół treningowy

5.2. Generowanie przykładów negatywnych

5.2.1. StyleGAN

5.2.2. Przykładowe twarze



Rysunek 5. Przykłady działania generatora twarzy.

5.3. Zbiory danych

Zbiór danych	# osób	# zdjęć	# zdjęć na osobę	aligned
MS1M-DeepGlint [5]	87K	3.9M	?/44.8/?	Tak
VGGFace2-test-train	500	163.3K	?/361/?	Nie
StyleGAN-0.7	1M	1M	1	Nie
VGGFace2-test-test	500	18.1K	?/361/?	Nie

Tabela 3. **Zbiory danych** MS1M-DeepGlint posłuży do wytrenowania ekstraktora cech, a VGGFace2-test po podzieleniu do treningu oraz testowania treningu FL.

5.4. Eksperymenty

5.4.1. Metoda pre-treningu modelu

5.4.2. Online Hard sample mining

6. Podsumowanie

Bibliografia

- [1] K. He, X. Zhang, S. Ren i J. Sun, **Deep Residual Learning for Image Recognition**, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, s. 770–778, 2015.
- [2] H. J. Wang, Y. Wang, Z.-F. Zhou, X. Ji, Z. Li, D. Gong, J. Zhou i W. Liu, **CosFace: Large Margin Cosine Loss for Deep Face Recognition**, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, s. 5265–5274, 2018.
- [3] J. Deng, J. Guo i S. Zafeiriou, **ArcFace: Additive Angular Margin Loss for Deep Face Recognition**, w CVPR, 2018.
- [4] Q. Cao, L. Shen, W. Xie, O. M. Parkhi i A. Zisserman, **VGGFace2: A dataset for recognizing faces across pose and age**, w *International Conference on Automatic Face and Gesture Recognition*, 2018.
- [5] <http://trillionpairs.deeplint.com/overview..>
- [6] G. B.H. E. Learned-Miller, **Labeled Faces in the Wild: Updates and New Reporting Procedures**, University of Massachusetts, Amherst, spraw. tech. UM-CS-2014-003, 2014.
- [7] 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.
- [8] 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