Dokumentacja końcowa – ISI 17Z

Adam Małkowski

1. Temat

W ramach projektu zaimplementowana została sieć neuronowa typu autoencoder oraz została wykorzystana do zmniejszenia wymiarowości zdjęć (28x28 pikseli) ze zbioru danych MNIST, reprezentujących odręcznie napisane cyfry, do 2 wymiarów. Wizualizacja wyników.

2. Rozwinięcie zadania

2.1 Autoencoder

Sieci neuronowe typu autoencoder są to głębokie sieci neuronowe posiadające określoną strukturę:
- warstwa wejściowa i wyjściowa są równoliczne,
- n-ta warstwa sieci (gdzie sieć ma m warstw takich, że $m > n$) jest najmniejsza i nazywana jest warstwą kodującą,
- co do zasady sieci te są symetryczne tj., jeżeli liczność warstwy i oznaczymy l_i to $l_i = l_{m-i+1}\ dla\ i = 1 \dots n-1$ oraz $m = 2n-1$.

Ich istotą jest możliwość redukcji wymiarowości analizowanych przestrzeni w następujący sposób:
- Sieć uczona jest bez nadzoru i podczas procesu uczenia dobiera się taki wagi, aby sieć na wyjściu zwracała to samo co otrzymała na wejściu.
- W przypadku nauczonej już sieci można podzielić ją na dwie części – koder i dekoder. Koder składa się z warstw od 1 do n, zaś dekoder od n do m. Punkt z dziedziny, dla której przeprowadzony został proces uczenia sieci, można podać na wejście kodera i wyjściem jest jego reprezentacja w mniejszej, zgodnej z rozmiarem warstwy kodującej, przestrzeni. Potencjalny powrót można przeprowadzić poprzez wykorzystanie analogiczne dekodera.

Pomijając specyfikacje opisaną powyżej, sieci neuronowe typu autoencoder posiadają te same własności co zwykłe głębokie sieci neuronowe – uczy je się korzystając z takich samych metod, wykorzystuje się takie same funkcje aktywacji itp.

2.2 MNIST

MNIST („Modified National Institute of Standards and Technology database") jest to baza danych zawierająca 60 000 obrazów rozmiaru 28x28 pikseli przedstawiających odręcznie napisane cyfry. Wraz z każdym obrazem występuje wartość definiująca jaka liczba jest na nim przedstawiona.

3. Implementacja

Do implementacji wykorzystany został język Python wraz z biblioteką TensorFlow – jest to narzędzie do budowania abstrakcyjnych grafów obliczeń które można przeliczać na różnych platformach – w szczególności na procesorze graficznym. W przypadku sieci neuronowych, jest to rozwiązanie bardzo przyspieszające uczenie z uwagi na macierzową naturę obliczeń w procesie uczenia i predykcji sieci neuronowych oraz fakt, że karty graficzne są zoptymalizowane do wykonywania obliczeń macierzowych. Praca na GPU zamiast CPU pozwala uzyskać przyśpieszenie około 2 rzędów wielkości.

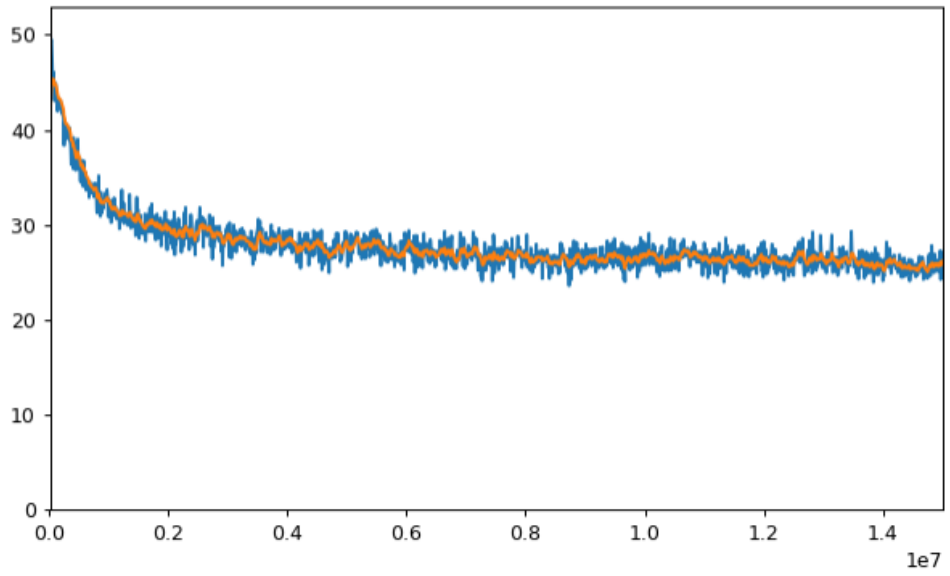
Kod został napisany i sformatowany w środowisku juptyer – notatniku wspierającym programowanie skryptowe.
Do uruchomienia wymaga zainstalowanego Pythona wraz z bibliotekami:
- tensorflow (podstawowa wersja umożliwia liczenie na CPU, wersja tensorflow-gpu wraz z odpowiednimi sterownikami kart graficznej na liczenie na GPU),
- keras,

- matplotlib,
- numpy,
- tqdm.

4. Uzyskane rezultaty

Informacje o sieci:

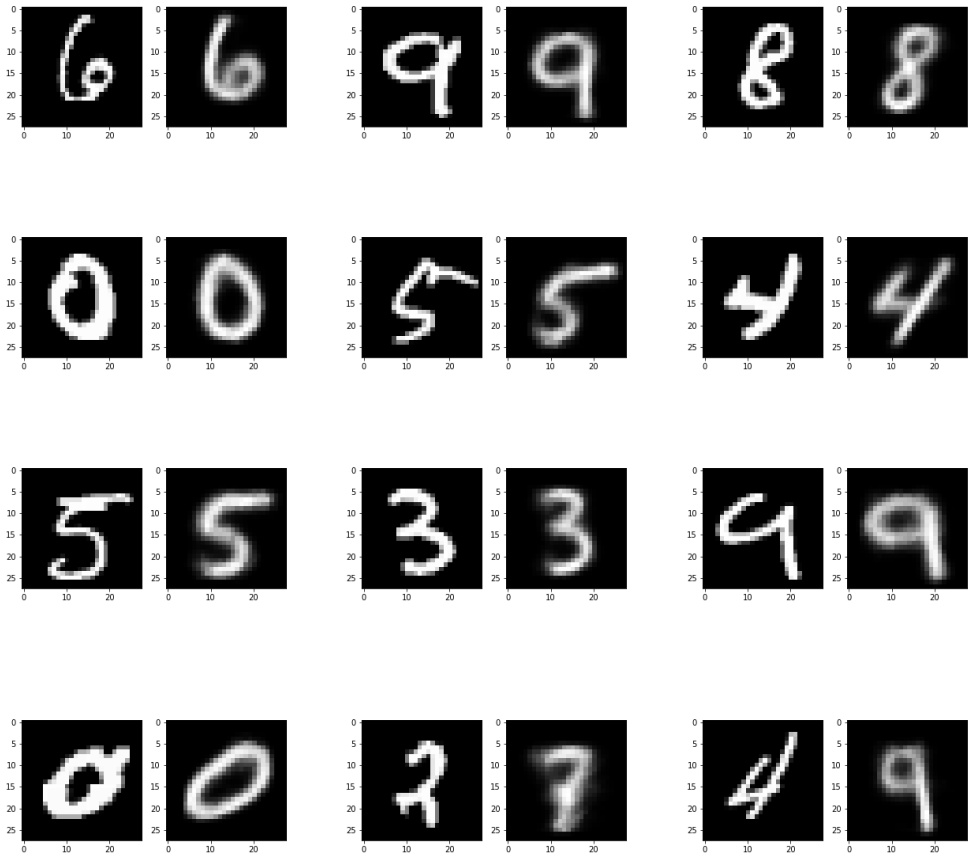
- Liczności warstw: [784, 1000, 500, 250, 2, 250, 500, 1000, 784]
- Funkcje aktywacji:
    - liniowa – warstwa kodująca
    - sigmoidalna - warstwa wyjściowa
    - selu[1] - pozostałe
- Metoda uczenia – *momentum* z parametrami: $eps = 0.0001$ oraz $moment$ zmienne w czasie, w przedziale od 0.5 do 0.99
- Przeanalizowane zostało 15 000 000 obrazków (z powtórzeniami) w partiach po 200 sztuk.



Rysunek 1 Wykres błędu średniokwadratowego na podzbiorze zbioru trenującego w trakcie procesu uczenia. Niebieska linia odpowiada dokładnej wartości błędu, pomarańczowa jest uśrednieniem.

Zauważalny jest dość wysoki poziom błędu średniokwadratowego do jakiego udało się dojść – dla porównania, identyczna sieć, jednakże z warstwą środkową liczącą 30 neuronów może odtwarzać obrazy wejściowe z błędem rzędu 1.5 (20 razy mniej).

[1] Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-Normalizing Neural Networks.;
$selu(x) = \begin{cases} x \; if \; x > 0 \\ \propto e^x - \propto \; if \; x \leq 0 \end{cases}$



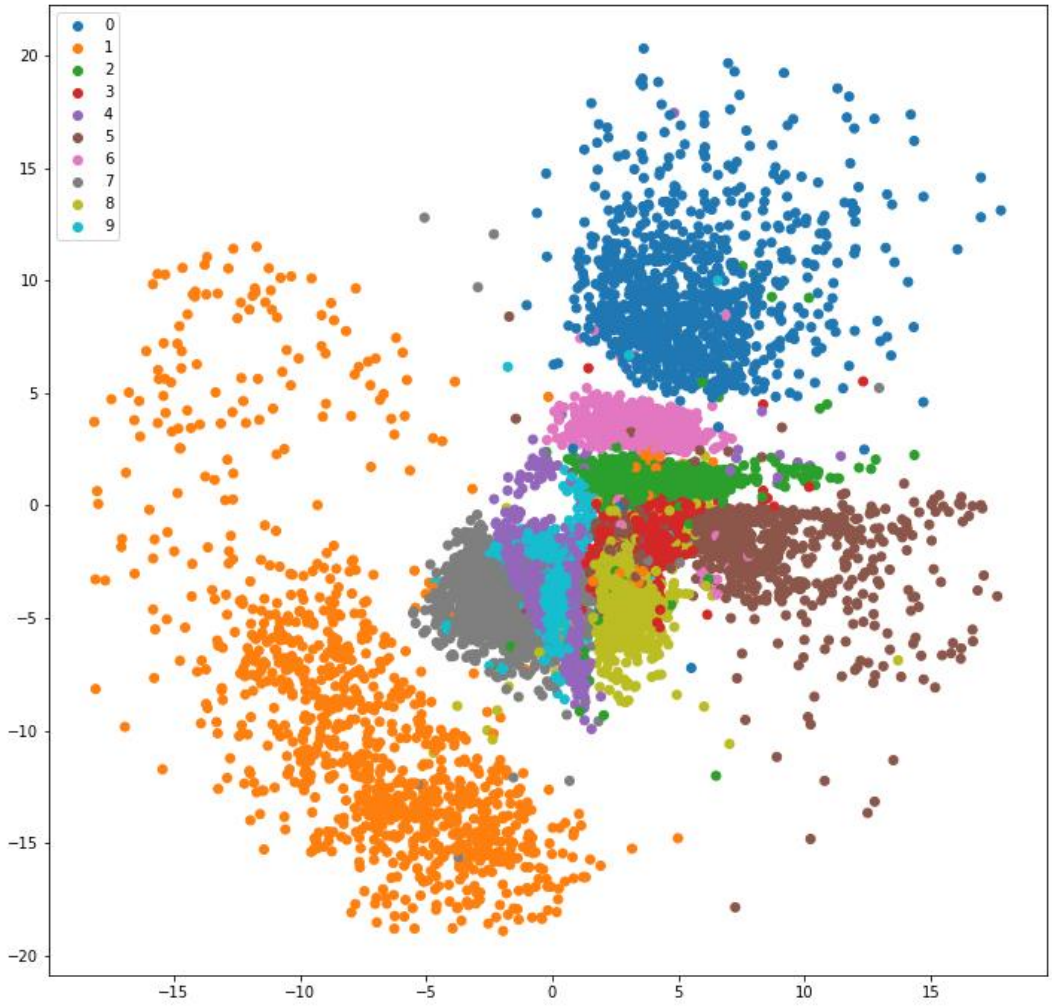
Rysunek 2 Przykłady odtworzenia obrazu wejściowego przez uzyskaną sieć – z lewej strony obraz wejściowy, z prawej wyjściowy. Więcej próbek w katalogu dołączonym do sprawozdania.

Powyższe przykłady pokazują co jest przyczyną tak wysokiego poziomu błędu – obrazy wyjściowe są „rozmazane" oraz czasami nie oddają poprawnej liczby (2 i 3 kolumna, 4 wiersz). Jednak patrząc na to z innej strony – błędy zdarzają się dość rzadko (zwłaszcza przy brzydko napisanych cyfrach) a w przypadkach w miarę poprawnego odtworzenia obrazu widać pewne „uproszczenie" cyfry – znikają niektóre cechy szczególne obrazu wejściowego (np. 3 kolumna, 3 wiersz – „9" nie jest zamknięte u góry; 4 kolumna, 1 wiersz – „0" ma nierówne granice) i obrazy wyjściowe odpowiadają cyfrom które ludzkie oko może bez trudności rozpoznać. W kontekście przetwarzania obrazów cyfr najistotniejszą informacją jest która cyfra znajduje się na rysunku – w większość zastosowań cechy szczególne konkretnego obrazu nie mają znaczenia (pomijając zagadnienia typu – rozpoznawanie autora na podstawie charakteru pisma). Wnioskiem z powyższych rozważań jest fakt, że kluczowa informacja – czyli ta, o tym jaka cyfra jest na obrazie – może być zakodowana w przestrzeni 2-wymiarowej bez konieczności klasyfikacji obrazu. Oczywiście, zagadnienie klasyfikacji dla tak prostego zbioru danych nie jest niczym trudnym – jednakże, w ogólnym przypadku pokazuje to realność zachowania kluczowych informacji przy znacznym zmniejszeniu wymiarowości problemu, w tym przypadku, z 784 na 2, co odpowiada około 40-krotnemu zmniejszeniu potrzebnej pamięci do przechowywania oraz potencjalnego przyspieszenia działania algorytmów przetwarzania rzeczonych danych – często algorytmy uczenia maszynowego albo analizy danych mają złożoność zależną od wymiarowości przestrzeni.

Jednorazowe nauczenie sieci neuronowej trwa – w przypadku opisanym w sprawozdaniu około 15 minut – jednakże można to wykonać raz, na szybkiej maszynie np. wykorzystującej proces graficzny, a następnie wykorzystywać nauczony model do redukcji wymiarowości nowych danych. Zakodowanie pojedynczego obrazu odbywa się bardzo szybko – praktycznie umożliwiając analizę online.

Rysunek 3 Kody wszystkich obrazów z MNIST na płaszczyźnie

Na rysunku 3 każdy punkt odpowiada jednemu obrazowi z analizowanego zbioru – kolor odpowiada cyfrze przedstawionej na obrazie zgodnie z legendą w lewym górnym rogu. Widoczne jest bardzo wyraźne odseparowanie „0” i „1”. Rozstrzał wartości jest bardzo duży – w obu osiach liczby są z przedziału $[-1500; 2000]$ gdzie cyfry poza „0”, „1” i „5” znajdują się w przedziale rzędu $[-100; 100]$. Dla czytelności ilustracji współrzędne (w obu wymiarach) zostały przekształcone funkcją nieliniową $x_{nowy} = x^{0.4}$.



Rysunek 4 Kody wszystkich obrazów z MNIST na płaszczyźnie – przekształcone funkcją nieliniową $x_{nowy} = x^{0.4}$

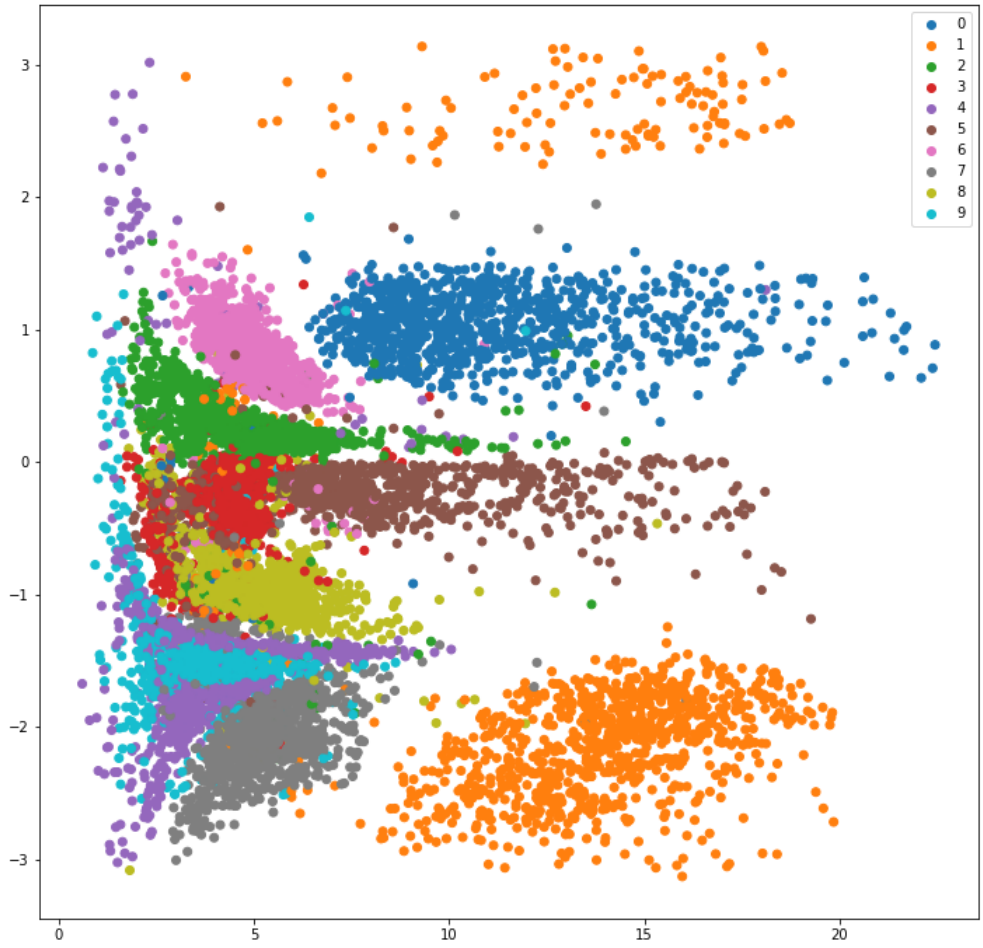
Przedziały wartości znacznie się zmniejszyły – spłaszczenie dotknęło zwłaszcza większe liczby. W efekcie, na rysunku 4 znacznie wyraźniej widać powstałe grupy. Również tutaj najlepiej odseparowane są „0”, „1” i „5”. Ale również obrazy reprezentujące inne cyfry znajdują się w swoim sąsiedztwie. Najgorzej odseparowane liczby to „4” i „9” – wynika to z częstego pojawiania się liczby „9” w zbiorze danych napisanej tak, że jest „otwarta” w górnej części – tak napisana jest bardzo podobna do „4” i w niektórych przypadkach nawet człowiek może mieć problem z poprawną jej klasyfikacją.

Rysunek 5 Kody wszystkich obrazów z MNIST na płaszczyźnie we współrzędnych radialnych

Na rysunku 5 zostały zaprezentowane wyniki po przekształceniu punktów do postaci radialnej – wartość na osi OY wskazuje na kąt podanych w radianach, zaś na OX odkładany jest moduł. Również w tym przypadku odbiór wykresu jest utrudniony z uwagi na bardzo duży rozrzut wyników, jeżeli chodzi o wartość modułu. Rysunek 6 prezentuje dane przekształcone, jeżeli chodzi o moduł, funkcją $x_{nowy} = x^{0.4}$. Wykres przedstawiony tam jest widoczniej pokazuje, że obrazy reprezentujące te same cyfry znajdują się blisko siebie. Również tutaj widać, że największy problem jest z reprezentacją „4" i „9".

Fakt, że obrazy reprezentujące poszczególne cyfry znajdują się w zbitych grupach jest o tyle istotny, że można wykorzystać tę przestrzeń do potencjalnego zadania klasyfikacji danych i oczekiwać rozsądnych wyników.

Zauważalnym jest również, że główne znaczenie posiada wartość kąta liczby – moduł o wartościach większych od 10 nie wpływa na cyfry, a i w przedziale $[0; 10]$ ma znaczenie mniejsze niż kąt.



Rysunek 6 Kody wszystkich obrazów z MNIST na płaszczyźnie we współrzędnych radialnych– przekształcone funkcją nieliniową $x_{nowy} = x^{0.4}$

5. Wnioski

Sieci neuronowe typu autoencoder z powodzeniem można wykorzystać do redukcji wymiarowości analizowanych danych – rzeczona redukcja wymiarowości posiada własności lepsze niż PCA[2] oraz po uzyskaniu nauczonej sieci neuronowej pozwala w bardzo krótkim czasie na konwersje nowych przykładów do mniej wymiarowej reprezentacji. Metoda ta, w przeciwieństwie do PCA, nie wykorzystuje jedynie informacji statystycznych, ale znajduję pewne wzorce wyróżniające grupy przykładów – w wielu zastosowaniach jest to cecha korzystna, bo wraz z redukcją wymiarowości idzie również redukcja zawartości informacyjnej polegająca na zachowaniu tylko kluczowych informacji, usuwając szumy oraz informacje drugorzędne. Pozwala to na zwiększenie jakości działania algorytmów analizy danych wykonywanych w otrzymanej przestrzeni na dwa sposoby – z uwagi na mniejszą wymiarowość działają szybciej, a z uwagi na redukcje zawartości informacyjnej mają „łatwiejsze”

[2] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

zadanie do wykonania. Oczywiście wiąże to się z tym, że dla części zastosowań powstałe informacje są zbyt ogólne (np. wspominana wcześniej analiza charakteru pisma) – ale innym zastosowaniom otwiera drzwi do szybkiego działania.