Deep Learning 1 (na 29.10.23)

1. *Matematyka w uczeniu głębokim.*
   Utworzyć tablice w numpy (numpy.array()), sprawdzić metody/pola: dtype, ndim, shape, type(object). Pokazać operacje na tensorze: wybór określonych elementów (tzw. slicing), reshape, broadcasting, operacje elementarne (np. dodanie dwóch tensorów).
2. *Zarządzanie obrazami.*
   Obrazy są danymi sieci konwolucyjnych. Proszę zapoznać się z dowolnym API zarządzania obrazami (np. najbardziej popularna biblioteka Pillow) i pokazać w kodzie metody: załadowanie obrazu, wyświetlenie, zapis do pliku, wypisanie informacji o obrazie (format, tryb koloru, rozmiar), zmiana rozmiaru, operacje: obrót, odwrócenie, wycinek; przygotowanie obrazu do sieci neuronowej: skalowanie, centrowanie, standaryzacja, itp.
3. *Data augmentation.*
   Przeanalizować kodem działanie *keras.preprocessing.image.ImageDataGenerator*
   Jest to automat do „tworzenia" dodatkowych obrazów. Jak używać tego generatora podczas uczenia?
4. *Wizja komputerowa.*
   Wizja komputerowa to dziedzina, która polega na filtrowaniu obrazów w celu znajdowania na nich pożądanych artafaktów.
   a. Proszę znaleźć gotowe metody filtrów i zastosować na obrazie.
   b. Proszę napisać kod w Pythonie, który ładuje i wizualizuje obraz, wykonuje na obrazie konwolucję zadanym filtrem (lub zestawem filtrów), na koniec wizualizuje i zapisuje obraz wynikowy do pliku. Filtry zapisać jako macierze kwadratowe o nieparzystej długości boku. Jeśli filtr da się zadawać analitycznie, dodatkowym parametrem wywołania powinien być rozmiar. Obraz wynikowy powinien być tej samej wielkości, co obraz wejściowy – jeśli to niemożliwe, należy rozszerzyć obraz wierszami/kolumnami zer na zewnątrz (tzw. 0-padding).
   c. Zaznajomić się z wykładem „Sieci konwolucyjne" i zademonstrować w kodzie działanie konwolucji jako metody (analiza uczenia sieci CNN na danych jest dalej). Uwzględnić hiperparametry: Stride (S), 0-Padding (P), rozmiar maski filtru (F). Przy wybranym F i S algorytm ma sam ustalić najmniejsze P, tak, aby obraz wynikowy miał ten sam rozmiar, co wejściowy.
      Do kodu dołączyć min. dwa przykładowe obrazy wejściowe.
5. *Szablon uczenia: komponenty.*
   Proszę przygotować szablon uczenia sieci neuronowej za pomocą biblioteki Keras: model sekwencyjny, funkcjonalny, własna podklasa klasy model. Kto woli, proszę przygotować taki szablon w bibliotece Pytorch.
6. *Pojedynczy neuron.*
   Zamodelować za pomocą biblioteki Keras (ew. Pytorch) pojedynczy neuron (testować różne funkcje aktywacji) i wyuczyć go klas liniowo separowalnych (na danych z dowolnego publicznego zbioru, lub wygenerowanych w przestrzeni 2D lub 3D).
7. *Sieć MLP.*
   Dla wybranego przez siebie zbioru danych proszę wyuczyć model sieci neuronowej (w zależności od danych: klasyfikacja binarna/wieloklasowa, regresja). Sieć MLP składa się z warstw *Dense*. Proszę dobrać hiperparametry (liczbę warstw ukrytych, liczbę neuronów na każdej warstwie, aktywacje, optymalizator) oraz funkcję straty i metrykę/i. Wizualizować funkcję straty/metrykę, co epokę.
8. *Sieć CNN.*
   Dla wybranego przez siebie zbioru obrazów proszę znaleźć dobry model CNN. Model należy budować przyrostowo, obserwując krzywe uczenia (wykres straty i dowolnej metryki w funkcji numeru epoki).

Sieć CNN składa się z pewnej liczby bloków typu: Conv2D, MaxPooling2D, Dropout, BatchNormalization. Warstwy te i ich hiperparametry zostały przedstawione na wykładzie „Sieci konwolucyjne”.

9. *Transfer learning.*
   Dla wybranego zbioru obrazów proszę przetestować wydajność sieci CNN o znanej architekturze (keras.io/api/applications). Dostosowanie topologii wstępnie wyuczonej sieci do naszego zbioru danych polega na:
   a. dostosowaniu rozmiaru wejścia do konkretnej sieci,
   b. zmianie wyjścia (adekwatnie do liczby klas problemu),
   c. douczeniu sieci na wybranym przez nas zbiorze.