WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ
POLITECHNIKI RZESZOWSKIEJ

Gabriel Lichacz

Rozpoznawanie rysunków grafów

Praca magisterska

Promotor:
dr Paweł Bednarz

Rzeszów, 2024

Spis treści

Wykaz symboli

G - graf

$V(G)$ - zbiór wierzchołków grafu G

$E(G)$ - zbiór krawędzi grafu G

C_n - cykl n-wierzchołkowy

D - digraf

$G(V_1, V_2)$ - graf dwudzielny

K_n - graf pełny

N_n - graf bezkrawędziowy

P_n - ścieżka n-wierzchołkowa

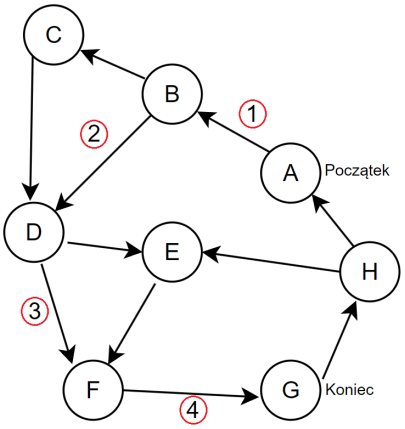
W_n - koło n-wierzchołkowe

1. Wstęp

W matematyce, grafy można zdefiniować jako graficzną reprezentację danych, w której wartości są przedstawione w pewien uporządkowany sposób, zwykle w relacji do siebie nawzajem. „*Stanowią wygodny aparat do modelowania różnych obiektów, (...) i odpowiednio interpretowane - mogą zawierać pewne informacje*"[12].

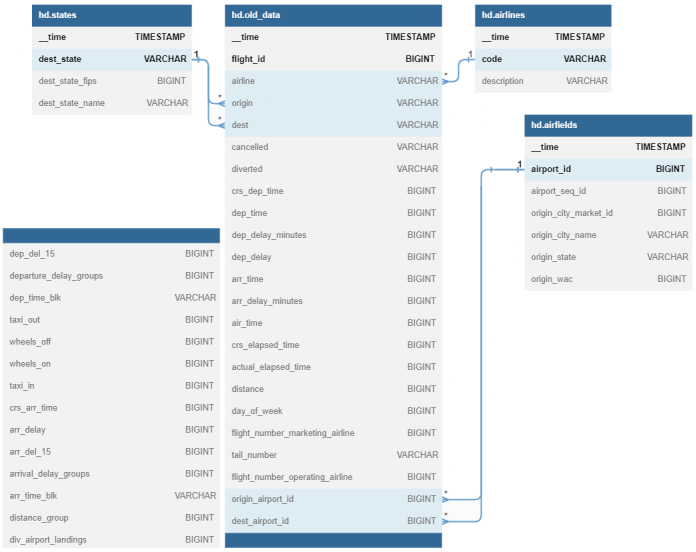
Teoria grafów to dziedzina matematyki zajmująca się badaniem właściwości grafów, będąca bardzo ważnym narzędziem w wielu „*dziedzinach od rachunku operacyjnego, chemii, po genetykę, lingiwistykę oraz od elektroniki i geografii po socjologię i architekturę*"[11]. Grafy dają możliwość zobrazowania pewnych modeli, co jest szczególnie korzystne w analizie wzorców. W kontekście grafów warto podkreślić ich zastosowanie poza teoretycznymi analizami.

W dziedzinach informatycznych, grafy stanowią fundament wielu algorytmów, takich jak algorytmy przechodzenia, algorytmy najkrótszej ścieżki, drzew rozpinających, czy modeli sieci. Przykładem może być tutaj wyszukiwanie najkrótszej trasy, chociażby w nawigacji GPS, gdzie wierzchołki odpowiadają skrzyżowaniom, a krawędzie drogom. W przypadku znajdowania najbardziej optymalnych tras, warto wymienić takie algorytmy jak $A*$, Bellmana-Forda czy Dijkstry.



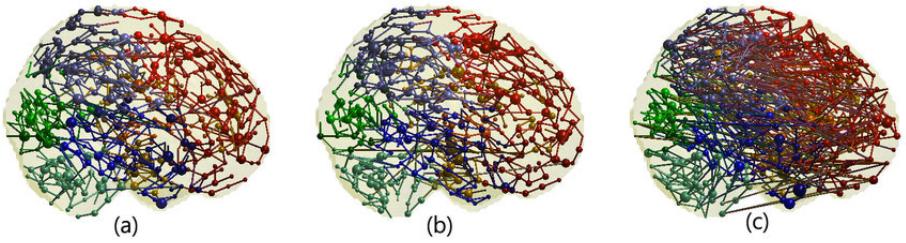
Rysunek 1.1: Przykład grafu z wyznaczoną najkrótszą ścieżką od wierzchołka A do G

Duże znaczenie mają również w reprezentacji i modelowaniu struktur danych, takich jak bazy danych. Najczęściej stosowane bazy danych, tj. relacyjne, zbudowane są w sposób, który grafy mogą doskonale zobrazować - wierzchołki odpowiadają kolumnom w tabelach a połączenia między nimi to krawędzie, reprezentujące relacje.



Rysunek 1.2: Przykładowy schemat relacyjnej bazy danych

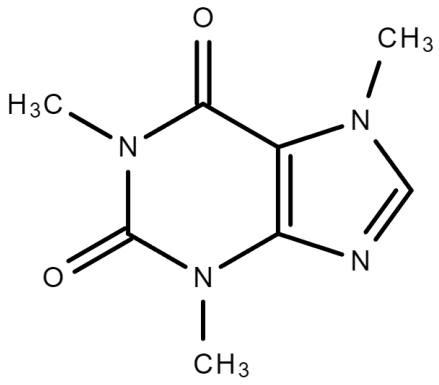
W biologii, grafy pełnią ważną rolę w modelowaniu układu nerwowego, sieci białek, szlaków metabolicznych oraz interakcji między genami. W genetyce wykorzystuje się je między innymi do analizy drzew filogenetycznych, co pozwala chociażby na śledzenie relacji ewolucyjnych między organizmami, z liśćmi reprezentującymi żywe organizmy, a wierzchołkami pośrednimi jako ich wspólnymi przodkami [4].



Rysunek 1.3: Przykłady minimalnych drzew rozpinających sieci neuronowych ludzkiego mózgu. Źródło: Chung M.K.: Graph Theory in Brain Networks, University of Wisconsin-Madison, 2021

Natomiast w chemii, grafy służą do reprezentacji struktury molekularnej związków chemicznych, umożliwiając naukowcom analizę ich właściwości i reaktywności. Znaczenie teorii grafów dla chemii wynika głównie z istnienia zjawiska izomeryzmu, które jest uzasadnione przez teorię struktury chemicznej. Wszystkie wzory strukturalne związków o wiązaniach kowalencyjnych są grafami, które nazywane są grafami

molekularnymi [2].

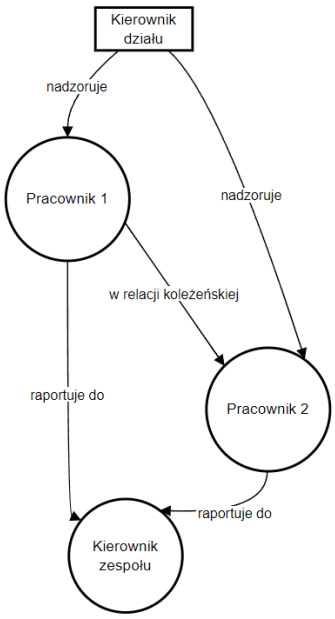


Rysunek 1.4: Struktura molekularna kofeiny

W lingwistyce, przy pomocy grafów możliwe jest modelowanie struktury języka, analiza morfologiczna czy syntaktyczna oraz wiele innych dziedzin, takich jak gramatyka generatywna, będąca prawdopodobnym kandydatem na teoretyczną podstawę biolingwistyki. Wykorzystywała ona notację drzewa jako pomocnicze narzędzie do wyrażania struktur językowych, lecz takie podejście zostało podważone w kolejnych pracach [1]. Dzięki grafom, możliwe jest również lepsze zrozumienie i przetwarzanie języka naturalnego przez komputery, co stanowi podstawę technologii takich jak tłumaczenie automatyczne czy rozpoznawanie mowy.

Teoria grafów znajduje także zastosowanie w analizie sieci społecznych, gdzie pomagają w badaniu relacji między ludźmi. Wielu psychologów i socjologów zajmuje się tematem struktur wynikających z relacji między różnymi podmiotami. Przykładami takich zależności mogą być sieci komunikacyjne między ludźmi, relacje dominacji i uległości w grupie, wpływ lub władza jednych podmiotów nad innymi, czy relacje między różnymi aspektami pola psychologicznego danej osoby lub jej osobowości [7]. Bardzo dużym polem jest również analiza mediów społecznościowych, które to wpływają coraz bardziej na przeciętnego człowieka. Poprzez gromadzenie i analizę danych dotyczących połączeń między użytkownikami, wzorców interakcji i zachowań komunikacyjnych, analiza mediów społecznościowych pozwala zauważyć pewne struktury społeczne i zidentyfikować wzorce leżące u podstaw interakcji w ich obrębie. Wszystko to możliwe jest do modelowania za pomocą struktur znanych z teorii grafów [9].

Rozpoznawanie wzorców, nam ludziom, pozwala na szybszą naukę przez roz-



Rysunek 1.5: Przykład sieci relacji pomiędzy pracownikami w dziale danej firmy

poznawanie czegoś, co już wcześniej widzieliśmy. W bardzo dużym uproszczeniu, algorytmy uczenia maszynowego działają w podobny sposób. Gdy model zostanie prawidłowo nauczony na pewnych danych, jest w stanie rozpoznawać podobne wzorce w innych, nigdy wcześniej nie widzianych.

Podsumowując, grafy są niezwykle wszechstronnym narzędziem, które znajduje zastosowanie w bardzo wielu dziedzinach nauki i technologii. Ich zdolność do reprezentowania skomplikowanych struktur i relacji w sposób zrozumiały, przystępny i czytelny jest nieoceniona. Dzięki nim możliwe jest również analizowanie i przetwarzanie informacji w efektywniejszy sposób, niż informacji nieustrukturyzowanych.

Celem pracy jest zobrazowanie owej zależności, na przykładzie nauczenia sieci neuronowej, w taki sposób, by po wytrenowaniu na kilku typach grafów stworzonych sztucznie, model był w stanie rozpoznać dane wzorce i je nazwać, w przestrzeni rzeczywistej.

2. Podstawowe definicje teorii grafów

2.1. Definicje

Definicje zostały zaczerpnięte z literatury, z pozycji [12], [11] oraz [13].

Definicja 1. Grafem nieskierowanym, skończonym G nazywamy parę (V, E), gdzie

$V = V(G)$ jest zbiorem skończonym, niepustym, natomiast $E = E(G)$ jest rodziną mogących się powtarzać dwuelementowych podzbiorów niekoniecznie różnych elementów ze zbioru V. Zbiór $V(G)$ nazywamy zbiorem wierzchołków, a elementy tego zbioru nazywamy wierzchołkami i oznaczamy symbolami: x, y, x_i, y_i, 1, 2, ... Zbiór $E(G)$ nazywamy zbiorem krawędzi grafu G. Mówimy, że krawędź $\{v, w\}$ łączy wierzchołki v i w, i na ogół oznaczamy ją krócej symbolem vw. W wielu zagadnieniach nazwy wierzchołków są nieistotne, więc je pomijamy i mówimy wtedy, że graf jest nieoznakowany.

Definicja 2. Jeżeli w grafie G istnieją co najmniej dwie krawędzie xy, to krawędź tę nazywamy krawędzią wielokrotną.

Definicja 3. Krawędź xy w grafie G nazywamy pętlą.

Definicja 4. Graf mający krawędzie wielokrotne nazywamy multigrafem.

Definicja 5. Graf, który nie ma krawędzi wielokrotnych i pętli, nazywamy grafem prostym.

Definicja 6. Graf zawierający pętle nazywamy pseudografem.

Definicja 7.

Definicja 8. Graf G taki, że $E(G) = \emptyset$, nazywamy grafem bezkrawędziowym. Jeżeli $|V(G)| = n$, to graf bezkrawędziowy oznaczony symbolem N_n. Każdy wierzchołek grafu bezkrawędziowego jest wierzchołkiem izolowanym.

Definicja 9.

Definicja 10. Graf prosty G taki, że każde dwa wierzchołki są sąsiednie, nazywamy grafem pełnym. Jeżeli $|V(G)| = n$, to graf pełny oznaczamy K_n.

Definicja 11. Graf G, którego zbiór wierzchołków można podzielić na dwa rozłączne, niepuste podzbiory V_1 i V_2 tak, że jeżeli $xy \in E(G)$, to $x \in V_1 \vee y \in V_2$ nazywamy grafem dwudzielnym.

Definicja 12. Drogą w grafie jest skończony ciąg naprzemiennie występujących wierzchołków i krawędzi, rozpoczynający się i kończący wierzchołkami, taki, że każde dwie kolejno po sobie następujące krawędzie mają wspólny wierzchołek.

Definicja 13.

3. Uczenie maszynowe

Uczenie maszynowe, znane również jako machine learning, to specjalistyczna gałąź sztucznej inteligencji, która koncentruje się na konstruowaniu modeli i algorytmów umożliwiajcych komputerom samodzielne uczenie się z dostępnych danych.

W przeciwieństwie do systemów, które są bezpośrednio programowane do wykonania określonych zadań, systemy uczenia maszynowego analizują dane, rozpoznają wzorce i podejmują decyzje oparte na zdobytej w ten sposób wiedzy.

3.1. Definicje

Definicje zostały zaczerpnięte z literatury, z pozycji [5], [6], [8] oraz [10].

Definicja 1. Zbiór treningowy - zbiór danych, który jest używany do trenowania modelu uczenia maszynowego.

Definicja 2. Zbiór walidacjny - zbiór danych, który jest używany do sprawdzenia wydajności modelu uczenia maszynowego.

Definicja 3. Zbiór testowy - zbiór danych używany do oceny wydajności modelu uczenia maszynowego po przeszkoleniu go na zbiorze treningowym i ocenie na zbiorze walidacyjnym.

Definicja 4. Klasyfikacja to proces polegający na przypisaniu obiektów do wcześniej zdefiniowanych klas na podstawie ich cech.

Definicja 5. Regresja liniowa - metoda, w której model liniowy przewiduje wyniki na podstawie ważonej sumy cech wejściowych oraz stałej, nazywanej punktem obciążenia lub punktem przecięcia.

Definicja 6. Walidacja krzyżowa to proces, w którym dane dzielone są na kilka części (przyjmujemy k) zwanych "złożeniami"(lub "foldami stąd nazwa k-Fold Cross-Validation). Model jest trenowany na $k-1$ złożeń, a testowany na pozostałym z nich. Proces ten jest powtarzany k razy, za każdym razem używając innego złożenia do testowania, a pozostałych do treningu. Jeżeli różnica w wydajności jest znacząca, uzasadniony jest sceptyzm odnośnie pojedynczych wyników pomiaru wydajności systemu. Z drugiej strony, jeżeli wszystkie wyniki są podobne, można mieć dużą dozę pewności, że niezależnie od konkretnego podziału na dane testowe i treningowe, wydajność systemu będzie podobna. Końcowa ocena modelu jest uzyskiwana poprzez uśrednienie wyników z każdej iteracji.

Definicja 7. Warstwa Rescaling mnoży każde wejście przez ustalony współczynnik skalujący. Zastosowanie tej techniki jest przydatne, gdy różne cechy danych wejściowych mają różne zakresy wartości. Poprzez jednolite skalowanie, model może efektywniej uczyć się wzorców, a proces optymalizacji staje się stabilniejszy.

Definicja 8. Warstwa Conv2D tworzy jądro splotu, które jest nakładane na dane

wejściowe w jednym wymiarze przestrzennym (lub czasowym), aby wygenerować tensor danych wyjściowych. Dodatkowo, jeśli stosowana jest funkcja aktywacji, jest ona stosowana również do danych wyjściowych.

Definicja 9. Warstwa MaxPooling2D dokonuje redukcji wymiarów danych wejściowych wzdłuż ich wymiarów przestrzennych (wysokości i szerokości), wybierając maksymalną wartość z każdego okna o rozmiarze określonym przez wybrany współczynnik *pool_size*, dla każdego kanału danych wejściowych. Okno to jest przesuwane o określoną liczbę kroków wzdłuż obu wymiarów.

Definicja 10. Warstwa Dropout losowo zeruje jednostki wejściowe z prawdopodobieństwem określonym przez wybrany współczynnik dropout na każdym etapie treningu, co pomaga unikać przeuczenia modelu. Jednostki, które nie zostały wyzerowane, są skalowane w górę przez mnożenie przez $\frac{1}{1-współczynnikDropout}$, aby suma wartości wejściowych pozostała niezmieniona.

Definicja 11. Bias (błąd obciążenia) to błąd wynikający z niepoprawnych założeń w procesie uczenia maszynowego. Oznacza różnicę między przewidywaną wartością modelu a rzeczywistą wartością.

Definicja 12. Funkcje aktywacji są nieliniowe, co pozwala na modelowanie złożonych funkcji i wprowadza nieliniowość do sieci. Równanie matematyczne opisujące działanie sieci neuronowej ma postać: $Y' = g(W_o + X^T * W)$, gdzie: Y' to przewidywana wartość wyjściowa, W_o to wartość bias, X^T to transpozycja macierzy wejściowej X, W to przypisane wagi, a g to funkcja aktywacji.

Definicja 13. Funkcja ReLU (ang. Recified Linear Unit) - funkcja aktywacji. Jest ciągła, ale nieróżniczkowalna w punkcie z = 0, a jej pochodna dla z < 0 wynosi 0. Spisuje się bardzo dobrze w modelowaniu złożonych funkcji, a dodatkowym aututem jest jej szybkość przetwarzania. Nie ma maksymalnej wartości wyjściowej.

Definicja 14. Warstwa Flatten (spłaszczona) ma za zadanie przekształcić każdy obraz wejściowy w tablicę jednowymiarową. Nie zawiera żadnych parametrów, a jej jedynym celem jest proste, wstępne przetworzenie danych.

Definicja 15. Przeuczenie to sytuacja, w której algorytm dopasowuje się zbyt dokładnie do danych treningowych, co prowadzi do modelu, który nie potrafi dokładnie prognozować ani wnioskować na podstawie nowych danych spoza zbioru treningowego.

Definicja 16. Warstwa Dense (w pełni połączona) zarządza samodzielnie swoją macierzą wag, zawierającą wszystkie wagi połączeń między neuronami a wejściami

do nich, oraz wekorem obciążeń. Zawiera najczęściej bardzo dużo parametrów, dzięki czemu model uzyskuje swobodę w dopasowaniu do danych treningowych. Jednocześnie, grozi mu również przez to ryzyko przetrenowania, zwłaszcza w przypadku korzystania z mniejszych zestawów danych.

Definicja 17. Regularyzacja to technika stosowana w celu zapobiegania przeuczeniu modelu. Działa poprzez dodanie kary do funkcji kosztu, co penalizuje zbyt złożone modele.

Definicja 18. Regularyzacja L2 służy do ograniczania wag sieci neuronowych, natomiast regularyzacja L1 przydaje się do tworzenia modeli rzadkich (w których wiele wag ma wartość równą 0). Zazwyczaj powinno się stosować ten sam typ regularyzatora we wszystkich wartstwach sieci.

Definicja 19. Epoka to pełny cykl przez cały zbiór danych treningowych, w której model przetwarza wszystie dostępne dane treningowe. Liczba epok określa ile razy model przejdzie przez cały zbiór danych treningowych.

Definicja 20. Dokładność modelu to stosunek oznaczonych prawidłowo wartości do przykładów sklasyfikowanych nieprawidłowo.

Definicja 21. Koncepcja macierzy pomyłek polega na zliczaniu przypadków zaklasyfikowania próbek z klasy A jako przykładów należących do klasy B. Aby utworzyć taką macierz, należy uzyskać zbiór prognoz, które porównywane są z rzeczywistymi wartościami docelowymi.

Definicja 22. Strata modelu to wartość, która wskazuje, jak bardzo prognozy modelu różnią się od rzeczywistych wartości dla pojedynczych przykładów. Idealnie przewidziane wartości mają stratę równą zeru, natomiast im większa różnica między prognozami a rzeczywistością, tym wyższa jest strata.

Definicja 23. Algorytm t-SNE (t-Distributed Stochastic Neighbor Embedding) to technika redukcji wymiarowości, która szczególnie dobrze nadaje się do wizualizacji wielowymiarowych zbiorów danych. Można ją zaimplementować przy użyciu aproksymacji Barnesa-Huta, co umożliwia stosowanie jej na dużych, rzeczywistych zbiorach danych.

3.2. Rodzaje uczenia maszynowego

Według [6], uczenie maszynowe można sklasyfikować na podstawie kilku kryteriów. Jest to nadzór człowieka w procesie trenowania, możliwość modelu do uczenia

się w czasie rzeczywistym oraz sam sposób pracy (nauka z przykładów lub modelu). Kryteria te nie wykluczają się wzajemnie - można je dowolnie łączyć. Za przykład może posłużyć filtr antyspamowy, który ciągle się uczy, wykorzystując model sieci neuronowej i analizując wiadomości email. Taki system można określić przyrostowym, opartym na modelu i nadzorowanym.

Dodatkowe kryteria oceny rodzaju uczenia maszynowego:

- Uczenie nadzorowane (ang. supervised learning) to podejście, w którym model jest szkolony na danych, które są już odpowiednio oznaczone (np. rekordy mają przypisane odpowiednie klasy). Celem jest odkrycie funkcji, która przekształca dane wejściowe w oczekiwane wyjścia. Znajduje zastosowanie w klasyfikacji i regresji. Przykład: Klasyfikacja wiadomości e-mail jako spam lub nie-spam.

- Uczenie nienadzorowane (ang. unsupervised learning) - w tym przypadku model bada nieoznaczone dane, aby odkryć pewne wzorce lub struktury. Najczęściej stosowane w klasteryzacji, czy redukcji wymiarowości. Przykłady:

  * Klasteryzacja klientów w celu segmentacji rynku, gdzie klienci są grupowani na podstawie ich zachowań zakupowych.

  * Redukcja wymiarowości w celu wizualizacji danych wysokowymiarowych, np. za pomocą algorytmu t-SNE.

- Uczenie przez wzmacnianie (ang. reinforcement learning) to przypadek, gdzie model uczy się poprzez interakcję ze swoim otoczeniem, podejmując decyzje, które maksymalizują pewną nagrodę. Przykłady:

  * Algorytmy sterujące robotami, które uczą się poruszać w nieznanym terenie.

  * Programy grające w gry, takie jak AlphaGo (chińska gra Go), które uczą się strategii gry poprzez rozgrywanie wielu partii.

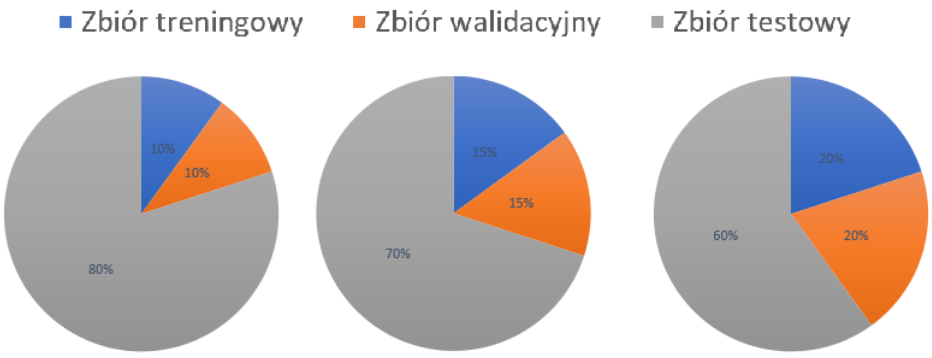
3.3. Proces uczenia maszynowego

Proces uczenia maszynowego można podzielić na kilka etapów, które są niezbędne do stworzenia skutecznego modelu zdolnego do samodzielnej nauki na podstawie zebranych danych.

Należy rozpocząć od zgromadzenia danych z odpowiednich źródeł. Mogą obejmować bazy danych, API, pliki CSV, czujniki, logi systemowe czy nawet wpisy z mediów

społecznościowych. Dane mogą być ustrukturyzowane (np. tabele w bazach danych) lub nieustrukturyzowane (np. obrazy, tekst).

Dalej, konieczne jest przygotowanie danych do odpowiedniego formatu. Obejmuje to usunięcie brakujących, pustych oraz błędnych wartości, radzenie sobie z duplikatami i anomaliami, skalowanie cech, kodowanie zmiennych kategorycznych, normalizację danych oraz podzielenie danych na zbiory treningowe, walidacyjne i testowe.



Rysunek 3.6: Przykład podziału danych na zbiory treningowe, walidacyjne i testowe

Najczęściej, podział na zbiory dokonuje się w proporcjach 70-80% na trening, 10-15% na walidację i 10-15% na testy. Jest to zależne od specyfiki problemu, dlatego konieczne jest odpowiednie przygotowanie i zbadanie danych przed podjęciem decyzji.

Wybór modelu to proces, który zależy od rodzaju problemu (np. regresja, klasyfikacja, klasteryzacja) oraz charakterystyki danych, gdzie najpopularniejsze modele to drzewa decyzyjne, lasy losowe, maszyny wektorów nośnych (SVM), sieci neuronowe, k-najbliższych sąsiadów (k-NN) i regresja liniowa/logistyczna. Trenowanie modelu to kolejny etap, który polega na dostosowaniu parametrów modelu do danych treningowych, w tym dostosowaniu hiperparametrów modelu (parametrów, które nie są uczone, np. liczba warstw w sieci neuronowej) poprzez metodę walidacji krzyżowej lub inne techniki optymalizacji.

Ewaluacja obejmuje ocenę modelu za pomocą pewnych metryk, takich jak dokładność, precyzja, recall, F1-score, błąd średniokwadratowy (MSE), błąd absolutny (MAE), a także analizy wydajności modelu w przypadku klasyfikacji binarnej. Optymalizacja modelu to kolejny etap, który obejmuje dalsze dostosowanie hyperparametrów, wybór cech, które najbardziej wpływają na wynik modelu, próby różnych architek-

tur modelu oraz zastosowanie technik takich jak L1, L2, dropout, które zapobiegają przeuczeniu modelu.

Implementacja modelu jest procesem, w którym wdrażany jest model w środowisku produkcyjnym. Zakłada ona przeprowadzenie integracji z aplikacjami zewnętrznymi, tworzenie API serwujących dane, zautomatyzoawanie decyzji, czy też śledzenie wydajności modelu w czasie rzeczywistym, aby wykryć ewentualne pogorszenie jakości (drift danych) i regularne aktualizacje modelu. Aktualizacja i utrzymanie modelu to kolejny etap, który obejmuje regularne aktualizowanie modelu na podstawie nowych danych, aby utrzymać jego dokładność i skuteczność, ciągłe monitorowanie, aby zapewnić, że model działa zgodnie z oczekiwaniami i nie występują niepożądane zachowania. Proces uczenia maszynowego jest iteracyjny i wymaga ciągłej interakcji między danymi, modelem i wynikami, aby osiągnąć optymalne rezultaty.

4. Wykorzystywane technologie

Praca opiera się na wykorzystaniu języka R oraz Python do generowania zbiorów danych, wszelkich manipulacji na nich oraz ich klasyfikacji.

4.1. Język R

Rysunek 4.7: Logo R [18]

Język R to szeroko stosowany w statystyce, analizie danych oraz naukach przyrodniczych język interpretowalny. Nie ma on skomplikowanej składni i jest przystosowany do bycia jak najbardziej przyjaznym dla nowego użytkownika. Oprócz dużych możliwości obliczeniowych, jest również świetnym narzędziem do wizualizacji danych, co spowodowało, że został wybrany do stworzenia zbioru danych. Grafy wygenerowane zostały przy pomocy biblioteki igraph w wersji 2.0.3. Jest to pakiet do tworzenia i analizy struktur sieci, a co za tym idzie oferuje bogaty wybór funkcji do generowania losowych i regularnych grafów oraz ich wizualizacji.

4.2. Język Python

Rysunek 4.8: Logo Python [17]

Język Python jest jednym z najpopularniejszych języków wysokopoziomowych ogólnego przeznaczenia. Zawdzięcza to swojej wszechstronności oraz prostocie składni. Znaczna liczba bibliotek pozwala na wykorzystywanie Pythona od prostych skryptów, przez analizę danych, aż po rozbudowane aplikacje, takie jak całe systemy największych gigantów technologicznych, np. Google. Język ten jest szeroko wykorzystywany w dziedzinie Data Science do wizualizacji, analizy i przetwarzania danych oraz w uczeniu maszynowym. Ostatnie z wymienionych zastosowań zadecydowało o wyborze języka Python jako narzędzia do stworzenia modelu klasyfikacji grafów. Wykorzystana została biblioteka Keras z pakietu Tensorflow.

4.3. Stanowisko pracy

Całość pracy, tj. generacja danych, modele oraz testy, została przygotowana na komputerze osobistym o parametrach:

- CPU: i5-10400F 2.9 GHz
- RAM: 32 GB 3200 MHz
- GPU: ADM Radeon RX 5600 XT 6GB
- Dysk: 2 x 1 TB HDD, 1 TB NVMe, 120 GB SSD
- System operacyjny: Windows 10

System nie posiada karty graficznej zoptymalizowanej pod zastosowania uczenia maszynowego. Biblioteki języka Python obsługują jednak karty graficzne AMD, co umożliwia pracę.

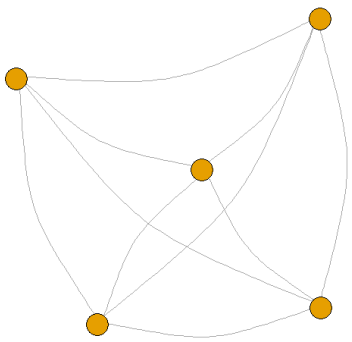
5. Testy

Model uczenia maszynowego jaki został wykorzystany w testach to sieć neuronowa.

Do testów stworzone zostało kilka modeli sieci neuronowych, wytrenowanych na rysunkach grafów stworzonych za pomocą skryptów R. Implementacja została wykonana biblioteką TensorFlow oraz Keras w języku Python. Modele są w stanie rozpoznawać rysunki grafów i przypisywać im odpowiednie klasy. Celem było również przetestowanie modeli na rzeczywistych zdjęciach, zawierających wzorce przypominające grafy, bądź rysunkach grafów narysowanych ręcznie.

Stworzone zostały 3 modele: - wytrenowany na danych ze stałą liczbą wierzchołków - wytrenowany na danych ze stałą liczbą wierzchołków oraz walidacją krzyżową - wytrenowany na danych ze zmienną liczbą wierzchołków

5.1. Generacja danych



Rysunek 5.9: Przykładowy wygenerowany graf pełny

Dane wygenerowane zostały przy pomocy skryptu stworzonego w języku R oraz biblioteki igraph. Skrypt został zaprojektowany funkcyjnie, by osiągnąć możliwie największą automatyzację testów. Rysunki grafów tworzone są o wielkości 800x600 pikseli, na białym tle, z wierzchołkami w kolorze pomarańczowym, bez jakichkolwiek oznaczeń wierzchołków oraz zapisywane są w odpowiednich katalogach, odpowiadających klasie grafu. Przygotowane zostały funkcje tworzące ścieżki, cykle, grafy pełne, grafy bezkrawędziowe oraz drzewa binarne. W każdej z funkcji możliwy jest wybór liczby

generowanych grafów, liczba wierzchołków grafu oraz współczynnik odpowiadający za zakrzywienie krawędzi na rysunkach.

```
#' Rysuj graf
#'
#' @param graph Graph - Graf do narysowania
#' @param pathName string - Sciezka
#' @param fileName string - Nazwa pliku
#' @param vertexNo int - liczba wierzcholkow
#' @param i int - Numer iteracji
#' @param plotCurve float
#' @return void
#'
plotGraphHelper <- function(graph, pathName, fileName,
 vertexNo, i, plotCurve)
{
  png(file.path(pathName, paste0(fileName, "-", vertexNo, "-",
  i, ".png")), width = 800, height = 600)
  plot(graph, vertex.label = NA, edge.curved = plotCurve)
  dev.off()
}
```

Listing 1: Listing skryptu rysującego grafy

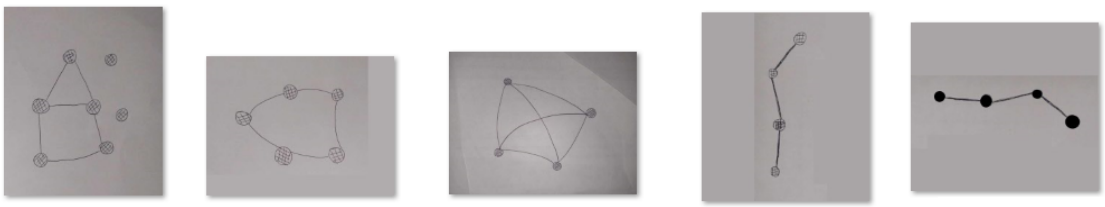
```
#' Graf sciezka N wierzcholkow, nieskierowany
#'
#' @param N int - liczba rysunkow
#' @param vertexNo int - liczba wierzcholkow
#' @param plotCurve float
#' @return void
#'
plotPaths <- function(N, vertexNo, plotCurve)
{
  fileName <- 'path'
  pathName <- createDir(vertexNo, fileName)
  definition <- c()
  for (index in 1:(vertexNo-1))
  {
    definition <- c(definition, index, index + 1)
  }
  definitionMatrix <- matrix(definition, ncol = 2, byrow =
 TRUE)

  for (i in 1:N)
  {
    graph <- graph_from_edgelist(definitionMatrix, directed =
 FALSE)
    E(graph)$weight <- runif(ecount(graph))
    plotGraphHelper(graph, pathName, fileName, vertexNo, i,
 plotCurve)
  }
}
```

Listing 2: Listing funkcji tworzącej ścieżkę

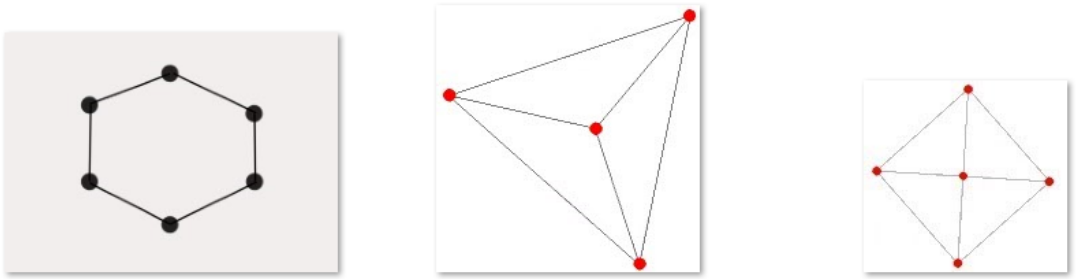
W testach wykorzystane zostały wszystkie wybrane typy grafów. Każdy z nich, czyli dana liczba wierzchołków i typ grafu, wygenerowany został w liczbie 500 sztuk oraz ze stałą krzywizną krawędzi, wynoszącą 0,3.

5.2. Dane zewnętrzne

Obrazy testowe, które nazwane są tutaj danymi zewnętrznymi, są rysunkami grafów pochodzącymi spoza przygotowanego testu. Dzielą się na obrazy pobrane z internetu, obrazy wygenerowane przez skrypt w R, ale nie używane w treningu oraz rysunki odręczne grafów.



Rysunek 5.10: Przykładowe zewnętrzne rysunki grafów narysowane odręcznie



Rysunek 5.11: Przykładowy zewnętrzne rysunki grafów pobrane z internetu

5.3. Opis skryptu

5.3.1. Przygotowanie

Skrypt rozpoczyna się od przygotowania środowiska do trenowania modelu. Najpierw ustawia ścieżki do katalogów z wygenerowanymi grafami oraz do katalogów na dane treningowe i walidacyjne. Następnie sprawdza, czy te katalogi istnieją, a jeśli nie, tworzy je. Dalej, definiuje parametry dotyczące wielkości obrazów oraz wielkości partii

danych, które będą używane podczas treningu. Dla każdej wartości liczby wierzchołków ustawia ścieżkę do katalogu z wygenerowanymi grafami, pobiera listę podkatalogów oraz obrazów w każdym z nich, a następnie dzieli obrazy na zestawy treningowe i walidacyjne w stosunku 80:20. Tworzy odpowiednie katalogi dla tych zestawów, jeśli jeszcze nie istnieją, i kopiuje obrazy do właściwych lokalizacji, segregując je na treningowe i walidacyjne.

5.3.2. Model

Na początku dane zostały przygotowane ze zbioru obrazów, znajdującego się w katalogu lokalnym. Dokonany został podział na zbiory treningowe i walidacyjne. Dla każdego przejścia walidacji krzyżowej, dane zostały podzielone inaczej. Po wczytaniu danych, zostały przeskalowane i przekształcone do odcieni szarości.

```
n_splits = 5
kfold = KFold(n_splits=n_splits, shuffle=True, random_state=42)
history = []
all_images = [os.path.join(dp, f) for dp, dn, filenames in os.walk(data_dir) for f in filenames if os.path.splitext(f)[1] == '.png']

for train_index, val_index in kfold.split(all_images):
  train_images = [all_images[i] for i in train_index]
  validation_images = [all_images[i] for i in val_index]

  # Dane treningowe
  train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    labels='inferred',
    label_mode='int',
    image_size=(img_height, img_width),
    batch_size=batch_size)
  class_names = train_ds.class_names
  train_ds = train_ds.map(lambda x, y: (rgb_to_grayscale(x), y))

  # Dane walidacyjne
  val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    data_dir,
    labels='inferred',
    label_mode='int',
    image_size=(img_height, img_width),
    batch_size=batch_size)
  val_ds = val_ds.map(lambda x, y: (rgb_to_grayscale(x), y))

  # Model
  model = tf.keras.models.Sequential([
    tf.keras.layers.Rescaling(1./255),
```

```
32        tf.keras.layers.Conv2D(32, 3, activation='relu'),
33        tf.keras.layers.MaxPooling2D(),
34        tf.keras.layers.Conv2D(32, 3, activation='relu'),
35        tf.keras.layers.MaxPooling2D(),
36        tf.keras.layers.Conv2D(32, 3, activation='relu'),
37        tf.keras.layers.MaxPooling2D(),
38        tf.keras.layers.Flatten(),
39        tf.keras.layers.Dense(128, activation='relu',
    kernel_regularizer=tf.keras.regularizers.l2(0.01)),
40        tf.keras.layers.Dropout(0.2),
41        tf.keras.layers.Dense(len(class_names))
42      ])
43
44      # Kompilacja modelu
45      model.compile(
46        optimizer='adam',
47        loss=tf.losses.SparseCategoricalCrossentropy(from_logits=
    True),
48        metrics=['accuracy']
49      )
50
51      # Uczenie modelu
52      history.append(model.fit(
53        train_ds,
54        validation_data=val_ds,
55        epochs=75
56      ))
```

Listing 3: Listing jedenego ze skryptów tworzących model

Model sieci neuronowej został zdefiniowany jako sekwencyjny stos warstw. Pierwsza warstwa to warstwa Rescaling, która normalizuje wartości pikseli do zakresu [0, 1]. Następne trzy warstwy to Conv2D z wybraną liczbą filtrów, z których każda jest poprzedzona warstwą MaxPooling2D. Warstwy konwolucyjne używają różnych funkcji akytwacji, np. ReLU. Po warstwach konwolucyjnych znajduje się warstwa Flatten, która przekształca mapy cech 2D w wektor 1D. Następnie dodana jest w pełni połączona (Dense) warstwa z wybraną liczbą jednostek i wybraną funkcją aktywacji, wraz z warstwą dropout. Zastosowana jest tam również regularyzacja L2 (zmniejszanie wag) z ustaloną siłą regularyzacji wynoszącą. Warstwa wyjściowa zawiera tyle jednostek, ile występuje klas. Zależnie od danego testu, może być to różna liczba.

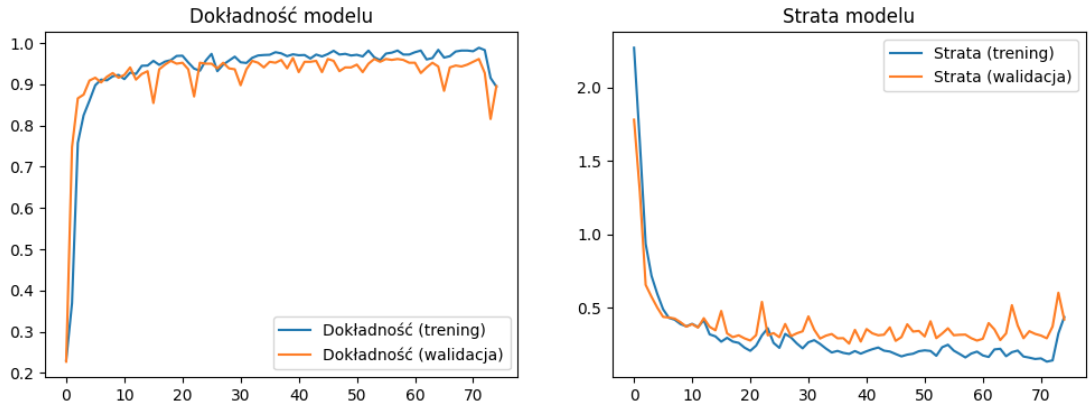
Dla standaryzacji danego testu ustalono K-Fold z liczbą podziałów równą 5. Dla wszystkich warstw wybrana została funkcja aktywacji ReLU. Liczba epok wyniosła 75. W przypadku warstw konwolucyjnych, wybrano 32 filtry, a dla warstwy w pełni połączonej zastosowano 128 jednostek. Regularyzacja została zastosowana z siłą 0,01, a współczynnik dropout - 0,2.

5.3.3. Wyniki

Po wytrenowaniu modelu, skrypt dokonuje wizualizacji dokładności i straty modelu w następujących krokach. Najpierw wyświetla w konsoli wartości dokładności dla obu zbiorów z historii treningu. Dalej tworzy wykresy, gdzie na pierwszym z nich pokazuje dokładność na zbiorze treningowym i walidacyjnym, a na drugim wykresie prezentuje stratę modelu dla obu zbiorów.

```
Dokładność na zbiorze treningowym:  [0.23068182170391083, 0.3693181872367859, 0.7579545378684998, 0.824999988079071, 0.8602272868156433, 0.897727251
Dokładność na zbiorze walidacyjnym:  [0.22727273404598236, 0.7477272748947144, 0.8659090995788574, 0.875, 0.9090909361839294, 0.9159091114997864, 0.
```

Rysunek 5.12: Przykładowe wartości dokładności dla zbioru treningowe i walidacyjnego



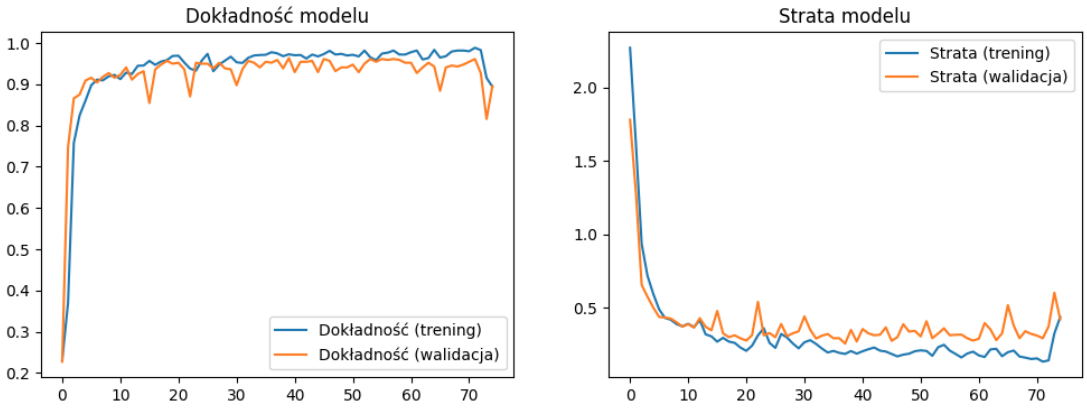
Rysunek 5.13: Przykładowa wizualizacja dokładności i straty wytrenowanego modelu

5.3.4. Testy na danych zewnętrznych

Po wyświetleniu dokładnosci modelu skrypt przeszukuje katalog z danymi i jego podkatalogi, by przygotować obrazy zewnętrzne. Następnie ustawia ścieżkę do katalogu z obrazami testowymi i pobiera ich listę. Dla każdego obrazu w tej liście wczytuje go, przeskalowuje do odpowiedniego rozmiaru i konwertuje do skali szarości Następnie model przewiduje klasę obrazu, a wynik jest wyświetlany w konsoli.

5.4. Testy modeli

5.4.1. Model podstawowy



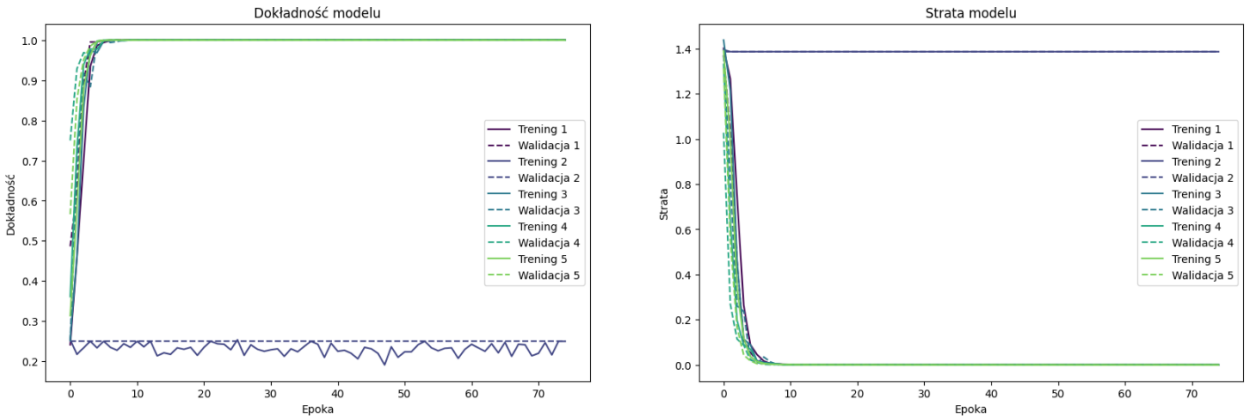
Rysunek 5.14: Wyniki testów dla modelu podstawowego

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\drawn\connected-drawn-1.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
|- test_graphs\drawn\cycle-drawn-1.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\drawn\full-drawn-1.jpg -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\drawn\full-drawn-2.jpg -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\drawn\full-drawn-3.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
|- test_graphs\drawn\path-drawn-1.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\drawn\path-drawn-2.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\drawn\path-drawn-3.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\generated\cycle-45.png -| najprawdopodobniej należy do klasy |- cycle -| z prawdopodobieństwem 56.76 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\generated\full-113.png -| najprawdopodobniej należy do klasy |- bipartite -| z prawdopodobieństwem 65.54 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\generated\path-78.png -| najprawdopodobniej należy do klasy |- path -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\internet\internet-cycle-1.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\internet\internet-full-1.jpg -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\internet\internet-full-2.jpg -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 99.97 procent.
```

Rysunek 5.15: Klasyfikacja obrazów zewnętrznych dla modelu ze zmienną liczbą wierzchołków

5.4.2. Model z walidacją krzyżową

W przypadku standardowego modelu z walidacją krzyżową model bardzo szybko uległ przeuczeniu. Już po szóstej iteracji dokładność na zbiorze walidaycjnym wyniosła 100%, co nie jest realistycznie możliwe. Została podjęta próba ograniczenia przeuczenia poprzez zwiększenie zbioru danych, zmiany liczby epok w modelu oraz manipulacji współczynnikami dropout i regularyzacji. W każdym przypadku model zwracał niezadowalające wyniki wynoszące 100% po jednej z początkowych iteracji.



Rysunek 5.16: Wyniki testów dla modelu z walidacją krzyżową

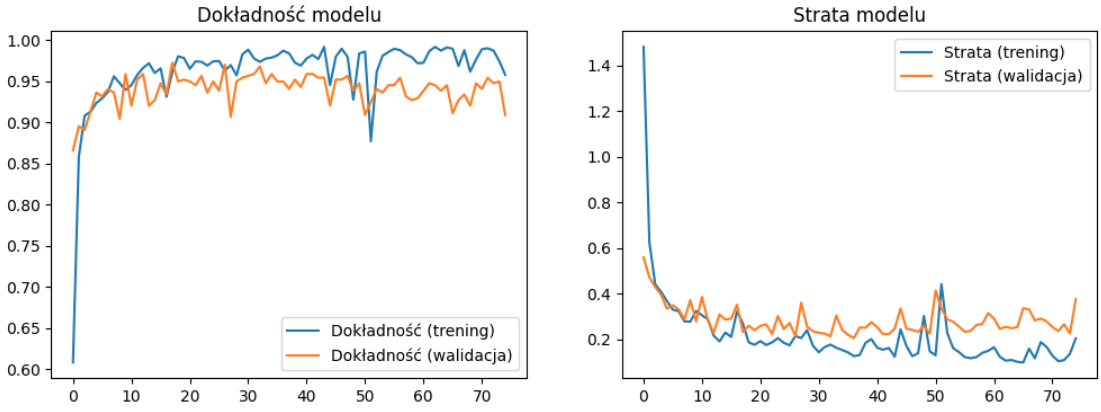
Z powodu przeuczenia model nie radził sobie z zewnętrznymi obrazkami testowymi. Większość grafów określił jako grafy pełne, co nie jest zgodne ze stanem rzeczywistym.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 104ms/step
|- test_graphs\drawn\connected-drawn-1.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
|- test_graphs\drawn\cycle-drawn-1.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
|- test_graphs\drawn\full-drawn-1.jpg -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\drawn\full-drawn-2.jpg -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\drawn\full-drawn-3.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\drawn\path-drawn-1.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
|- test_graphs\drawn\path-drawn-2.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\drawn\path-drawn-3.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\generated\cycle-45.png -| najprawdopodobniej należy do klasy |- cycle -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
|- test_graphs\generated\full-113.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\generated\path-78.png -| najprawdopodobniej należy do klasy |- path -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\internet\internet-cycle-1.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\internet\internet-full-1.jpg -| najprawdopodobniej należy do klasy |- empty -| z prawdopodobieństwem 99.93 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
|- test_graphs\internet\internet-full-2.jpg -| najprawdopodobniej należy do klasy |- empty -| z prawdopodobieństwem 100.00 procent.
```

Rysunek 5.17: Klasyfikacja obrazów zewnętrznych dla modelu z walidacją krzyżową

5.4.3. Model ze zmienną liczbą wierzchołków

Najlepsze wyniki pod względem rozpoznawania zewnętrznych obrazków testowych oraz realistycznej dokładności na zbiorze walidacyjnym, zostały uzyskane przy użyciu modelu sieci neuronowej uczonej na rysunkach grafów z różną liczbą wierzchołków. Było to odpowiednio 4, 5, 6 oraz 7 wierzchołków.



Rysunek 5.18: Wyniki testów dla modelu ze zmienną liczbą wierzchołków

Model nie poradził sobie zbyt dobrze z obrazami zewnętrznymi, lecz znacznie lepiej niż model z walidacją krzyżową. Poprawnie wskazanych klas grafów było 5 z 14 wszystkich rysunków. Mimo, że model jest w stanie poprawnie określić niektóre typy grafów poprawnie, wciąż jest to dokładność niższa niż 50%.

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 120ms/step
|- test_graphs\drawn\connected-drawn-1.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 61.37 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\drawn\cycle-drawn-1.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 64.43 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\drawn\full-drawn-1.jpg -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\drawn\full-drawn-2.jpg -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 99.80 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\drawn\full-drawn-3.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
|- test_graphs\drawn\path-drawn-1.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 62.29 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\drawn\path-drawn-2.png -| najprawdopodobniej należy do klasy |- connected -| z prawdopodobieństwem 99.46 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\drawn\path-drawn-3.png -| najprawdopodobniej należy do klasy |- full -| z prawdopodobieństwem 69.90 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\generated\cycle-45.png -| najprawdopodobniej należy do klasy |- bipartite -| z prawdopodobieństwem 55.28 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step
|- test_graphs\generated\full-113.png -| najprawdopodobniej należy do klasy |- bipartite -| z prawdopodobieństwem 56.59 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\generated\path-78.png -| najprawdopodobniej należy do klasy |- path -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
|- test_graphs\internet\internet-cycle-1.png -| najprawdopodobniej należy do klasy |- cycle -| z prawdopodobieństwem 100.00 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\internet\internet-full-1.jpg -| najprawdopodobniej należy do klasy |- path -| z prawdopodobieństwem 99.71 procent.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
|- test_graphs\internet\internet-full-2.jpg -| najprawdopodobniej należy do klasy |- path -| z prawdopodobieństwem 39.16 procent.
```

Rysunek 5.19: Klasyfikacja obrazów zewnętrznych dla modelu z walidacją krzyżową

5.4.4. Model ze zmienną liczbą wierzchołków i walidacją krzyżową

Rysunek 5.20: Wyniki testów dla modelu ze zmienną liczbą wierzchołków i walidacją krzyżową

Rysunek 5.21: Klasyfikacja obrazów zewnętrznych dla modelu ze zmienną liczbą wierzchołków i walidacją krzyżową

5.5. Wnioski

W przypadku uczenia modeli z wykorzystaniem grafów pełnych, najczęściej dominowały one cały zbiór danych, przez co modele w kolejnych testach klasyfikowały większość testowych grafów rysowanych odręcznie jako właśnie grafy pełne.

Testy z wykorzystaniem stałej liczby wierzchołków grafów okazały się mniej owocne niż testy z rysunkami grafów o zmiennej liczbie wierzchołków.

Wystąpiła tendencja do niepoprawnego określania innych grafów, grafami dwudzielnymi, jeśli takie znajdowały się w zbiorze danych treningowych.

6. Podsumowanie i wnioski końcowe

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Załączniki

- Skrypt generujący obrazy grafów
- Skrypt testowy z modelem podstawowym
- Skrypt testowy z modelem, z walidacją krzyżową
- Skrypt testowy z modelem dostosowanym do nauki grafów o różnej liczbie wierzchołków
- Skrypt testowy z modelem, z walidacją krzyżową, dostosowanym do nauki grafów o różnej liczbie wierzchołków

Literatura

[1] Arikawa K.: Graph Theory Teaches Us Something About Grammaticality. The Prague Bulletin of Mathematical Linguistics No. 112, 2019, pp. 55-82

[2] Balaban A.T.: Applications of Graph Theory in Chemistry. Department of Organic Chemistry, Polytefhnic Institute. 76206 Bucharest, Roumania, 1985

[3] Chung M.K.: Graph Theory in Brain Networks, University of Wisconsin-Madison, 2021

[4] Erciyes K.: Graph-Theoretical Analysis of Biological Networks: A Survey. Computation 2023, 11, 188, DOI: https://doi.org/10.3390/computation11100188

[5] Fenner M.E.: Uczenie maszynowe w Pythonie dla każdego. Helion SA, Gliwice 2020.

[6] Géron A.: Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow. Helion SA, Gliwice 2020.

[7] Harary F., Norman R.Z.: Graph Theory as a Mathematical Model in Social Science, Research Center for Group Dynamics, University of Michigan, 1953

[8] Seenappa M.G.: Graph Classification using Machine Learning Algorithms. Master's Projects. 725, San Jose State University 2019, DOI: https://doi.org/10.31979/etd.b9pm-wpng

[9] Umami M.H., Prihandini R.M., Agatha A.B.: Application of Graph Theory to Social Network Analysis, Department of Mathematics Educations, University of Jember, Jember, Indonesia, 2024

[10] L.J.P. van der Maaten, Hinton G.E.: Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008

[11] Wilson R.J.: Wprowadzenie do teorii grafów. PWN, Warszawa 2012.

[12] Włoch A., Włoch I.: Matematyka dyskretna. Podstawowe metody i algorytmy teorii grafów. Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów 2008.

[13] Wojciechowski J., Pieńkosz K.: Grafy i sieci. PWN, Warszawa 2013.

[14] https://cran.r-project.org/web/packages/igraph/index.html. Dostęp 10.03.2024.

[15] https://developers.google.com/machine-learning. Dostęp 20.07.2024.

[16] https://www.ibm.com/topics. Dostęp 20.07.2024.

[17] https://www.python.org/. Dostęp 07.08.2024.

[18] https://www.r-project.org/. Dostęp 07.08.2024.

[19] http://student.krk.pl/026-Ciosek-Grybow/rodzaje.html. Dostęp 26.03.2024.

[20] https://www.tensorflow.org/api_docs. Dostęp 21.07.2024.

[21] http://wms.mat.agh.edu.pl/~md/ang-pol.pdf. Dostęp 29.03.2024.

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza Rzeszów, 2024
Wydział Matematyki i Fizyki Stosowanej

STRESZCZENIE PRACY DYPLOMOWEJ MAGISTERSKIEJ

ROZPOZNAWANIE RYSUNKÓW GRAFÓW

Autor: Gabriel Lichacz, nr albumu: 164174
Opiekun: dr Paweł Bednarz
Słowa kluczowe: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po polsku

RZESZOW UNIVERSITY OF TECHNOLOGY Rzeszow, 2024
The Faculty of Mathematics and Applied Physics

MSC THESIS ABSTRACT

RECOGNITION OF GRAPHS

Author: Gabriel Lichacz, nr albumu: 164174
Supervisor: Paweł Bednarz PhD
Key words: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po angielsku