Politechnika Wrocławska

Ocena skuteczności GAE i VGAE w generowaniu reprezentacji grafów: zastosowanie do klasyfikacji oraz detekcji anomalii w molekułach

Jakub Antczak
Grupa I
268745@student.pwr.edu.pl

Joanna Wojciechowicz
Grupa III
255747@student.pwr.edu.pl

Streszczenie

W niniejszej pracy badamy, czy modele GAE i VGAE potrafią nauczyć się struktury niemutagennych cząsteczek oraz czy w połączeniu ze stratą rekonstrukcji lub modelem GMM mogą skutecznie wspomagać detekcję mutacji. Dodatkowo analizujemy jakość uzyskanych reprezentacji w zadaniu klasyfikacji. Eksperymenty pokazują, że pretrenowanie modeli na dodatkowym zbiorze niemutagennych cząsteczek - nawet z innej dziedziny - zwykle poprawia wyniki. Ponadto metoda oparta na GMM osiągała lepsze rezultaty niż podejście ze stratą rekonstrukcji.

1 Wstęp

Reprezentacja grafowa odgrywa dość ważną rolę w zastosowaniach sztucznej inteligencji w chemii i biologii, gdzie wiele struktur, takich jak molekuły, można naturalnie przedstawić jako grafy. Jakość tych reprezentacji ma bezpośredni wpływ na skuteczność modeli uczenia maszynowego w zadaniach takich jak klasyfikacja, predykcja właściwości czy detekcja anomalii. W kontekście biologii molekularnej szczególne znaczenie ma możliwość automatycznego rozpoznawania mutacji - zmiany strukturalne molekuł, które mogą mieć istotne konsekwencje biologiczne. Dlatego badanie efektywności metod generatywnych uczenia reprezentacji grafów, takich jak grafowy autoenkoder (GAE) oraz wariacyjny grafowy autoenkoder (VGAE), ma duże znaczenie praktyczne.

Motywacją do podjęcia tego tematu była potrzeba zrozumienia, na ile modele uczące się reprezentacji są w stanie uchwycić subtelne zmiany w strukturze grafu, odpowiadające rzeczywistym, biologicznie istotnym różnicom między molekułami - np. mutacjom.

W naszej pracy korzystamy z dwóch zbiorów danych: MUTAG oraz ENZYMES, Danymi wejściowymi do naszego modelu są grafowe reprezentacje molekuł zawarte w tych zbiorach. Stosujemy modele GAE i VGAE, które są trenowane w różnych wariantach: (1) bez pretrenowania, (2) z pretrenowaniem na zbiorze ENZYMES.

Celem naszego badania jest ocena, czy uzyskane reprezentacje są przydatne w detekcji anomalii - rozumianych jako mutacje - na podstawie straty rekonstrukcji grafu lub przy użyciu modelu mieszaniny rozkładów normalnych (GMM). Chcielibyśmy również zbadać, czy wstępne trenowanie modelu na zbiorze danych o zbliżonej dziedzinie tematycznej wpływa korzystnie na jego wyniki. Dodatkowo, analizujemy jakość reprezentacji w kontekście klasyfikacji molekuł (mutagenna vs niemutagenna) za pomocą regresji logistycznej, co pozwala oszacować ich użyteczność w zadaniach docelowych.

Probabilistyczne modele grafowe 2025, Politechnika Wrocławska. (templatka LateX zapożyczona z kursu Deep learning na Stanford University. Oryginalnie pochodzi z NIPS 2017.)

2 Powiązane prace

Rozwój grafowych sieci neuronowych rozpoczął się od prac takich jak [4], w którym opisano model GCN. Wprowadzono w niej propagację informacji o sąsiedztwie poprzez macierz sąsiedztwa. Metoda ta jest niepraktyczna dla dużych grafów, ze względu na operacje macierzowe, które wymagają dostępu do całej struktury grafu. Następnie Hamilton w opublikował pracę [2], której celem było umożliwienie indukcyjnego uczenia reprezentacji wierzchołków w grafach. Metoda ta stanowiła odpowiedź na ograniczenia skalowalności klasycznych ujęć grafowych sieci neuronowych (GNN), takich jak GCN. GraphSAGE proponuje alternatywne podejście oparte na losowym próbkowaniu sąsiedztwa wierzchołków oraz uczeniu funkcji agregujących, co pozwala na uogólnienie modelu na wcześniej niewidziane dane. Idea polega na obliczaniu osadzeń wierzchołków w sposób lokalny, poprzez agregację informacji pochodzących z ograniczonego podzbioru sąsiadów danego węzła. Nowa rozwiązanie umożliwiło przetwarzanie dużych grafów. W obszarze autoenkoderów grafowych pierwszym istotnym wynikiem był klasyczny GAE oraz jego wariacyjna odmiana VGAE [3]. Metody te stały się bazą dla wielu późniejszych podejść do samonadzorowanego ekstraktownia reprezentacji grafów.

Klasyczne podejścia do wykrywania cząsteczek mutagennych opierają się głównie na testach biologicznych, takich jak test Amesa [8]. Metody, mimo wysokiej dokładności, są czasochłonne, kosztowne i często wymagają użycia organizmów żywych. Obecnie nie istnieje jeden test, który dostarczałby informacji o aspektach niezbędnych do określenia potencjału mutagennego danego związku chemicznego [5]. Metoda, która pozwalałaby na ocenienie czy dana cząsteczka jest mutagenna, tylko na podstawie jej struktury byłaby przełomowa.

Wraz z rozwojem technik uczenia maszynowego pojawiły się pierwsze prace wykorzystujące modele klasyfikacyjne do przewidywania właściwości chemicznych cząsteczek. Przykładowo w pracy [6] stworzono oprogramowanie ogólnodostępne i darmowe, które pozwala na przewidywanie wartości pKa.

Przełom nastąpił wraz z wprowadzeniem głębokiego uczenia nad strukturami grafowymi. Duvenuad [1] jako jeden z pierwszych zastosował konwolucje grafowe do związków chemicznych, reprezentując cząsteczki jako grafy atomów i wiązań. Sieć ta była w stanie się uczyć reprezentacji cząsteczek bez potrzeby ręcznej inżynierii cech.

Dalsze badania nad grafowymi sieciami neuronowymi koncentrują się na ulepszaniu reprezentacji grafów, wprowadzając różne urozmaicenia, takie jak wykorzystanie mechanizmu uwagi [9]. Warto też wspomnieć, że w dostępnej literaturze, nie znaleźliśmy żadnej pracy wykorzystującej modele mikstur rozkładów normalnych (GMM) do klasyfikacji cząsteczek jako anomalii w kontekście detekcji mutagenności.

3 Zbiory danych

W projekcie skupiliśmy się na dwóch powszechnie stosowanych zbiorach grafów udostępnionych w repozytorium TUDatasets - **ENZYMES** oraz **MUTAG** [7]. Zbiór ENZYMES składa się z 600 grafów reprezentujących struktury sześciu różnych klas enzymów (po 100 grafów na klasę). Każdy wierzchołek opisany jest 21-wymiarowym wektorem cech. Z kolei MUTAG, zawiera 188 grafów cząsteczek chemicznych oznaczonych etykietami mutagenności - 125 grafów oznaczonych jako klasa 0, czyli niemutagenne oraz 63 klasy 1, czyli mutagenne. Węzły opisane są 7-wymiarowymi cechami. Ze względu na niespójność wymiaru cech między zbiorami, cechy MUTAG uzupełniliśmy zerami do wymiaru odpowiedającego liczbie cech w ENZYMES.

Do trenowania modelu w trybie mini-paczek (ang. *mini-batches*) wykorzystaliśmy mechanizm `DataLoader` z biblioteki `PyTorch Geometric`, w którym wektor `batch` pozwala na jednoczesne przetwarzanie wielu grafów przy zachowaniu informacji o przynależności węzłów do konkretnego grafu. W zadaniu wykrywania anomalii przeprowadziliśmy dwustopniowy proces uczenia. W fazie wstępnego uczenia modelu wykorzystaliśmy cały zestaw danych ENZYMES, natomiast podczas douczania posłużyliśmy się wyłącznie próbkami należącymi do klasy 0 ze zbioru MUTAG, stanowiącymi 70% wszystkich przykładów tej klasy. Zbiór walidacyjny do dostrajania hiperparametrów skomponowaliśmy z pierwszej połowy pozostałej części zbioru MUTAG, a zbiór testowy z drugiej połowy. W eksperymencie dotyczącym klasyfikacji struktura wstępnego uczenia pozostaje niezmieniona, natomiast w etapie douczania zbiór danych został podzielony w następujący sposób: 35%

próbek przeznaczono na etap trenowania, 35% na walidację, a pozostałe 30% stanowi zbiór testowy. Podczas podziału zachowano proporcje klas.

4 Metody

4.1 Grafowy Autoenkoder (GAE)

W naszym podejściu do samonadzorowanego uczenia oraz detekcji anomalii wykorzystujemy klasyczny grafowy autoenkoder GAE, zaproponowany przez Kipfa i Wellinga [3]. GAE składa się z:

- Kodera - jest to wybrana przez nas grafowa sieć neuronowa:
$$\mathbf{Z} = \text{GNN}(\mathbf{X}, \mathbf{A}),$$
gdzie $\mathbf{X}$ to cechy wierzchołków, a $\mathbf{A}$ to macierz sąsiedztwa grafu.
- Dekodera - dekoder na wejściu przyjmuje wyznaczone reprezentacje $\mathbf{Z}$, a na wyjściu oblicza rekonstrukcję danego obiektu, w naszym przypadku jest to graf. Skupiamy się wyłącznie na strukturze grafu, czyli dokonujemy rekonstrukcji krawędzi.

Aby skonstruować odpowiedni dekoder strukturalny, musimy określić metodę rozstrzygania, czy pomiędzy dwiema dowolnymi paroma wierzchołków istnieje krawędź. Często stosowanym podejściem jest wykorzystanie ich iloczynu skalarnego, który również zostanie wykorzystany w naszych badaniach. Dekoder ma zatem postać
$$\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^T),$$
gdzie

- $\hat{\mathbf{A}}$ to rekonstrukcja macierzy sąsiedztwa,
- $\sigma(\cdot)$ to sigmoidalna funkcja aktywacji.

Koder, w przypadku GAE, definiujemy jako dwuwarstwową sieć z grafową warstwą konwolucją GCNConv zaproponowaną przez Kipfa w 2016 roku [4]. Model GCN w każdej warstwie oblicza nowe cechy wierzchołków $H^{(l+1)}$ na podstawie obecnych cech $H^{(l)}$ następująco
$$H^{(l+1)} = \hat{D}^{-\frac{1}{2}} A \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)},$$
gdzie

- A to macierz sąsiedztwa grafu,
- $\hat{D}$ to macierz stopni węzłów (macierz diagonalna),
- $\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$ to tzw. symetryczna normalizacja macierzy sąsiedztwa,
- $W^{(l)}$ to macierz wyuczanych parametrów.

Ponadto, po warstwie GCNConv stosujemy nieliniową funkcję aktywacji ReLU, wprowadzamy normalizację mini-paczek (ang. *batch normalization*) oraz losowo wyłączanie grafu obliczeniowego (ang. *dropout*). Jako funkcję straty definiujemy binarną entropię krzyżową pomiędzy oryginalną, a zrekonstruowaną macierzą sąsiedztwa, z dodatkiem negatywnego próbkowania krawędzi. Dzięki takiemu podejściu model uczy się rozróżniać prawdziwe krawędzie od losowo wygenerowanych negatywów.

4.2 Wariacyjny grafowy autoenkoder (VGAE)

W naszych badaniach wykorzystujemy również wariacyjną wersję grafowego autoenkodera - *Variational Graph Autoencoder (VGAE)* - zaproponowaną przez Kipfa i Wellinga [3]. VGAE stanowi rozszerzenie klasycznego GAE, w którym zamiast punktowych reprezentacji w przestrzeni latentnej uzyskujemy parametry rozkładu normalnego (średnią i logarytm odchylenia standardowego), co umożliwia probabilistyczne kodowanie niepewności oraz lepsze właściwości generatywne modelu.

Model składa się z dwóch głównych komponentów:

- Koder - jego zadaniem jest estymacja parametrów rozkładu normalnego:

$$\boldsymbol{\mu}, \log \boldsymbol{\sigma} = \text{GNN}(\mathbf{X}, \mathbf{A}),$$

gdzie $\mathbf{X}$ to cechy wierzchołków, a $\mathbf{A}$ to macierz sąsiedztwa grafu. W naszej implementacji kodera:

  - pierwsza warstwa GCN przekształca dane wejściowe do przestrzeni ukrytej, po czym stosujemy normalizację mini-paczek (*batch normalization*), funkcję aktywacji ReLU oraz Dropout,
  - następnie osobne warstwy GCN estymują $\boldsymbol{\mu}$ oraz $\log \boldsymbol{\sigma}$.

Ostatecznie reprezentacja $\mathbf{Z}$ w przestrzeni latentnej uzyskiwana jest za pomocą próbkowania z rozkładu normalnego przy użyciu triku reparametryzacji:

$$\mathbf{Z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, I).$$

- Dekoder – jego celem jest rekonstrukcja struktury grafu wyłącznie na podstawie reprezentacji latentnych. Podobnie jak w przypadku GAE, rekonstrukcja macierzy sąsiedztwa opiera się na iloczynie skalarnym reprezentacji wierzchołków:

$$\hat{\mathbf{A}} = \sigma(\mathbf{Z}\mathbf{Z}^T),$$

gdzie $\sigma(\cdot)$ to funkcja sigmoidalna, a $\hat{\mathbf{A}}$ to rekonstrukcja macierzy sąsiedztwa.

Jako funkcję straty stosujemy sumę dwóch składników:

- straty rekonstrukcji - mierzonej za pomocą binarnej entropii krzyżowej między oryginalną, a zrekonstruowaną macierzą sąsiedztwa, z użyciem negatywnego próbkowania krawędzi,
- dywergencji Kullbacka-Leiblera (KL) - penalizującej rozbieżność pomiędzy rozkładem latentnym $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$, a rozkładem normalnym standardowym $\mathcal{N}(0, I)$.

Ostateczna funkcja straty ma postać:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \frac{1}{N}\mathcal{L}_{\text{KL}},$$

gdzie N to liczba wierzchołków w grafie.

Model trenujemy w identyczny sposób, jak w GAE. Warto dodać, że architektura kodera w modelach GAE i VGAE jest identyczna - różnica dotyczy tylko sposobu modelowanie przestrzeni ukrytej oraz funkcji straty.

4.3 Modele mikstur rozkładów normalnych (GMMs)

W celu przeprowadzenia detekcji anomalii na osadzonych reprezentacjach grafów wykorzystujemy modele mikstur rozkładów normalnych (ang. *Gaussian Mixture Models*). Modele te należą do klasy probabilistyczny model zmiennych ukrytych i zakładają, że dane obserwowane są generowane z mieszanki skończonej liczby komponentów, z których każdy ma rozkład normalny o nieznanych parametrach. W kontekście naszego projektu GMM jest wykorzystywany do identyfikacji anomalii na przestrzeni reprezentacji osadzonych, generowanych przez autoenkodery grafowe. Intuicja polega na założeniu, że punkty znacznie odbiegające od rozkładów modelowany przez komponenty GMM mogą być traktowane jako anomalie. W przeprowadzonych eksperymentach model GMM został wytrenowany wyłącznie na próbkach należących do klasy 0 ze zbioru MUTAG. Na podstawie wartości logarytmicznej wiarygodności (log-likelihood) uzyskanych dla próbek ze zbioru treningowego, ustaliliśmy próg klasyfikacji anomalii według wzoru

$$\mathbf{threshold} = Q_3 + 1.5 \cdot \text{IQR}, \tag{1}$$

gdzie

- Q_3 - kwantyl rzędu $3/4$ wartości logarytmicznej wiarygodności na zbiorze treningowym,
- IQR - rozstęp międzykwartylowy.

Ponieważ nie chcemy opierać się na konkretnym progu decyzyjnym, którego wartość silnie zależy od wielkości dostępnego zbioru danych (a ENZYMES i MUTAG to niewielkie zbiory), w wynikach skupiamy się na metryce AUC. AUC jest niezależne od wyboru progu, co czyni je bardziej stabilnym i porównywalnym wskaźnikiem skuteczności w klasyfikacji binarnej.

4.4 Strata rekonstrukcji

W celu oceny przydatności modeli GAE oraz VGAE w detekcji anomalii w strukturach molekularnych, analizujemy wartość funkcji straty rekonstrukcji. Wysoka wartość tej straty może wskazywać, że dana próbka (molekuła) odbiega od rozkładu danych treningowych, co czyni ją potencjalną anomalię.

Po zakończeniu treningu, dla każdej próbki walidacyjnej (bądź na samym końcu testowej) obliczamy zakodowaną reprezentację $\mathbf{Z}$ oraz odpowiadającą jej wartość straty rekonstrukcji. Aby określić, które próbki stanowią anomalie, wyznaczamy próg analogicznie jak w 1. W tym przypadku również opierać się będziemy na metryce AUC.

5 Eksperymenty i wyniki

Przeprowadziliśmy eksperymenty mające na celu ocenę wpływu wybranych hiperparametrów na skuteczność detekcji mutacji oraz klasyfikację grafów jako mutagennych lub niemutagennych. Analizowane hiperparametry to: współczynnik uczenia $[10^{-4}, 10^{-3}, 10^{-2}]$, rozmiar mini-paczki $[8, 16]$, liczba wymiarów w warstwie reprezentacji $[16, 32, 64]$ oraz współczynnik losowego wyłączenia neuronów $[0.1, 0.3, 0.5]$.

Reprezentacje grafów tworzymy na podstawie osadzeń wierzchołków, następnie agregujemy za pomocą operacji globalnego uśrednienia (ang. *global mean pooling*), aby uzyskać wektorową reprezentację całego grafu. Tak przygotowane reprezentacje wykorzystywane są zarówno w zadaniu detekcji mutacji, jak i klasyfikacji grafów.

W zadaniu detekcji mutacji zastosowano podejścia oparte na stracie rekonstrukcji oraz modelu GMM. Na podstawie zbioru walidacyjnego dobrano optymalny zestaw hiperparametrów, a następnie model wytrenowano od nowa na połączonym zbiorze treningowym i walidacyjnym (z wykluczeniem przykładów klasy 1). Ewaluację przeprowadzono na zbiorze testowym. Liczbę komponentów GMM dobrano na podstawie kryteriów AIC i BIC. Jest to 3 dla GAE, 1 dla VGAE.

W zadaniu klasyfikacji molekuł jako mutagenne/niemutagenne użyliśmy regresji logistycznej jako klasyfikatora. Model trenowano na zbiorach zgodnych z opisem w Rozdziale 3.

dUwzględniliśmy łącznie 12 wariantów najlepszych modeli: GMM z i bez pretrenowania na ENZYMES, model oparty na stracie rekonstrukcji z i bez pretrenowania, oraz klasyfikator z i bez pretrenowania (dla GAE i VGAE). Na zbiorze ENZYMES trening był wykonywany na 100 epokach, a dotrenowywanie na MUTAG na 50 epokach. Jeśli trening był wykonywany tylko na zbiorze MUTAG, to było to 100 epok.

5.1 Detekcja anomalii

Wyniki metryki AUC na zbiorze testowym (Tabela 1) pokazują, że zastosowanie GMM do detekcji anomalii przyniosło lepsze rezultaty zarówno dla GAE, jak i VGAE, gdy modele były wcześniej pretrenowane na zbiorze ENZYMES. Sugeruje to, że model skuteczniej koduje strukturę niemutagennej cząsteczki w reprezentacji, jeśli podczas treningu zetknie się z większą liczbą podobnych przykładów - nawet pochodzących z innego typu zbioru, co jest intuicyjne. Dodatkowo, spośród analizowanych konfiguracji, najlepsze wyniki osiągnął pretrenowany model GAE.

Analizując wyniki detekcji anomalii na zbiorze testowym przy użyciu straty rekonstrukcji (tabela 2), zauważamy, że pretrenowanie niewiele poprawiło skuteczność modelu GAE, natomiast w przypadku VGAE doprowadziło do pogorszenia wyników. Warto też zauważyć, że w tym zadaniu model VGAE znacząco przewyższa GAE pod względem osiąganych rezultatów.

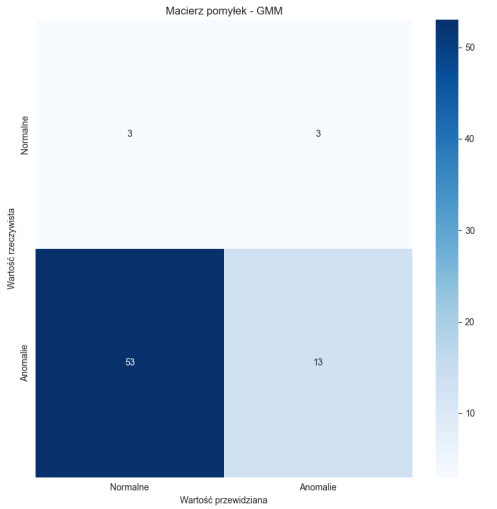
Macierze pomyłek (Rys. 1 oraz 2) przedstawiają wyniki dla najlepszych modeli detekcji anomalii opartych na GAE i VGAE, z zastosowaniem progu ustalonego według wzoru 1. W obu przypadkach wykorzystano metodę opartą na GMM, która - jak pokazują również Tabele 1 i 2 - osiągała lepsze wyniki niż podejście oparte na stracie rekonstrukcji. Zauważalna jest tendencja modeli do klasyfikowania mutagennych cząsteczek jako niemutagennych. Ponadto, we wszystkich przypadkach wartość metryki AUC była niższa niż 0.5, co wskazuje na słabą zdolność rozróżniania klas. Jednym z możliwych powodów tak niskiej skuteczności może być zbyt mały rozmiar zbioru treningowego.

Tabela 1: Porównanie wyników najlepszych modeli na zbiorze testowym dla detekcji anomalii za pomocą GMM.

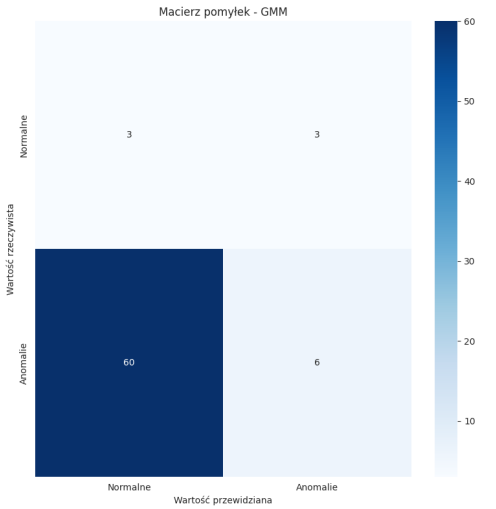
Model	**AUC score - GMM**	**Hiperparametry: lr, batch size, hidden channels, latent dim, dropout**
GAE - pretrenowany	0.43	0.01, 16, 64, 64, 0.5
GAE	0.28	0.01, 8, 64, 32, 0.5
VGAE - pretrenowany	0.39	0.01, 8, 64, 64, 0.5
VGAE	0.36	0.01, 16, 64, 64, 0.3

Tabela 2: Porównanie wyników najlepszych modeli na zbiorze testowym dla detekcji anomalii za pomocą straty rekonstrukcji.

Model	**AUC score - *recon loss***	**Hiperparametry: lr, batch size, hidden channels, latent dim, dropout**
GAE - pretrenowany	0.1	0.0001, 16, 64, 64, 0.1
GAE	0.09	0.0001, 16, 64, 64, 0.1
VGAE - pretrenowany	0.21	0.0001, 16, 64, 64, 0.5
VGAE	0.24	0 .0001, 16, 64, 64, 0.1



Rysunek 1: GAE - macierz pomyłek dla GMM przy ustalonym progu - najlepsze parametry (pretrenowany).



Rysunek 2: VGAE - macierz pomyłek dla GMM przy ustalonym progu - najlepsze parametry (pretrenowany).

5.2 Klasyfikacja

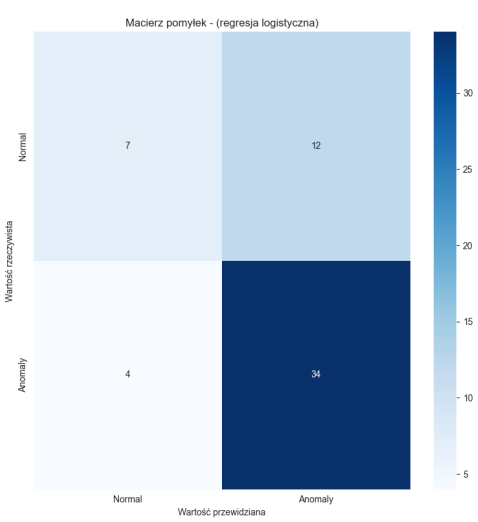
Po wytrenowaniu modeli GAE i VGAE na zbiorze zawierającym zarówno cząsteczki mutagenne, jak i niemutagenne, uzyskane reprezentacje przekazano do prostego klasyfikatora - regresji logistycznej - w celu oceny ich jakości w zadaniu klasyfikacji. Z Tabeli 3 wynika, że modele pretrenowane na zbiorze ENZYMES osiągnęły wyższe wartości metryki F1, co świadczy o lepszej jakości reprezentacji. Ogólnie model GAE uzyskał lepsze wyniki niż VGAE. Macierze pomyłek (rys. 3 i 4) przedstawiają wyniki najlepszych modeli na zbiorze testowym. Widać w nich, że modele mają trudność z prawidłowym rozpoznawaniem cząsteczek niemutagennych, często klasyfikując je błędnie jako mutagenne.

Przeciętne wyniki klasyfikacji mogą wynikać z niewystarczającej liczby danych treningowych, przez co modele GAE i VGAE nie były w stanie wyuczyć wysokiej jakości reprezentacji. Alternatywnie,

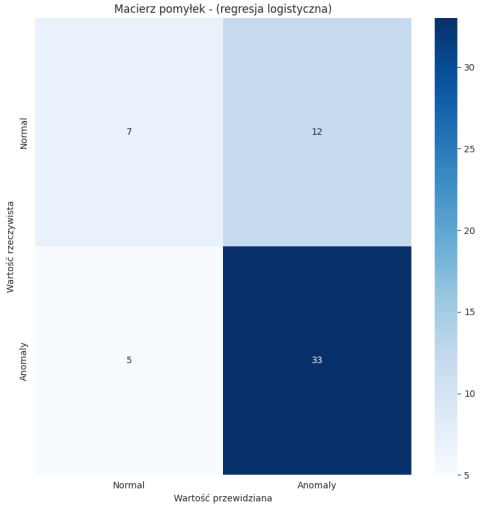
przyczyną może być również zbyt prosty klasyfikator - regresja logistyczna - który nie jest w stanie w pełni wykorzystać struktury zakodowanej w danych.

Tabela 3: Porównanie wyników najlepszych modeli na zbiorze testowym dla klasyfikacji grafów.

Model	**F1 macro score**	**Hiperparametry: lr, batch size, hidden channels, latent dim, dropout**
GAE - pretrenowany	0.64	0.01, 8, 64, 32, 0.1
GAE	0.62	0.01, 16, 64, 16, 0.1
VGAE - pretrenowany	0.62	0.01, 16, 64, 32, 0.1
VGAE	0.56	0.001, 16, 64, 32, 0.5



Rysunek 3: GAE - macierz pomyłek dla klasyfikacji grafów - najlepsze parametry (pretrenowany).



Rysunek 4: VGAE - macierz pomyłek dla klasyfikacji grafów - najlepsze parametry (pretrenowany).

6 Wnioski i kierunki dalszych prac

W pracy zbadaliśmy dwie metody detekcji anomalii w cząsteczkach (rozumianych jako mutacje): opartą na rozkładzie GMM oraz na stracie rekonstrukcji, wykorzystując reprezentacje z modeli GAE i VGAE trenowanych z pretrenowaniem lub bez niego. Celem było sprawdzenie, czy modele są w stanie nauczyć się struktury poprawnych (niemutagennych) molekuł, mając dostęp wyłącznie do takich podczas treningu.

Analiza metryki AUC wykazała, że metoda z GMM działała lepiej niż ta oparta na stracie rekonstrukcji. Pretrenowanie na zbiorze ENZYMES poprawiało wyniki dla metody GMM, co potwierdza, że większa liczba przykładów poprawnych cząsteczek sprzyja nauce ich reprezentacji. W zadaniu klasyfikacji molekuł z użyciem regresji logistycznej również zauważono poprawę po pretrenowaniu, a lepsze wyniki uzyskano na reprezentacjach z GAE niż z VGAE.

Niskie wartości AUC sugerują jednak, że modele nie były w pełni skuteczne - prawdopodobnie ze względu na mały rozmiar zbiorów treningowych. Dalsze prace powinny uwzględniać użycie większych zbiorów danych, lepsze dostrojenie progu decyzyjnego, oraz rozważenie bardziej złożonych klasyfikatorów i innych metod uczenia reprezentacji.

7 Podział prac

Jakub Antczak: implementacja GAE, implementacja klasyfikacji, implementacja GMM, funkcja pobierające dane, utworzenie eksperymentów dla GAE, raport.

Joanna Wojciechowicz: implementacja VGAE, implementacja straty rekonstrukcji, implementacja trenowania, utworzenie eksperymentów dla VGAE, raport.

Źródła

Użyte pakiety: `scikit-learn`, `matplotlib`, `seaborn`, `torch_geometric`, `torch`, `numpy`.

[1] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[2] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[3] Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[5] Stanley Maloy and Kelly Hughes. *Brenner's encyclopedia of genetics*. Academic Press, 2013.

[6] Kamel Mansouri, Neal F Cariello, Alexandru Korotcov, Valery Tkachenko, Chris M Grulke, Catherine S Sprankle, David Allen, Warren M Casey, Nicole C Kleinstreuer, and Antony J Williams. Open-source qsar models for pka prediction using multiple machine learning approaches. *Journal of Cheminformatics*, 11:1–20, 2019.

[7] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs, 2020.

[8] Sebastian Tejs. The ames test: a methodological short review. *Environmental Biotechnology*, 4(1):7–14, 2008.

[9] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.