# k-NN Sampling for Visualization of Dynamic Data Using LION-tSNE - analysis

**Gędłek Paweł**
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza
Kraków
gedlek@student.agh.edu.pl

**Wójtowicz Patryk**
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza
Kraków
wojtowicz@student.agh.edu.pl

**ABSTRAKT**

Obecnie rośnie zapotrzebowanie na metody wizualizacji dynamicznie zmieniających się dużych zbiorów danych. Taka potrzeba pojawia się między innymi w medycynie, gdy aktualne dane (np. stan pacjenta) zmienia się na bieżąco. Co za tym idzie istotnym elementem analizy danych, jest nie tylko dobór odpowiedniej metody wizualizacji, ale także sposobu próbkowania. W niniejszym raporcie prezentujemy analizę oraz wizualizację danych opartą na metodzie Local Interpolation with Outlier coNtrol t-distributed stochastic neighbor embedding (LION tSNE) wraz z wykorzystaniem idei kNN samplingu. Jako próbę kontrolną metody LION tSNE użyliśmy metody tSNE, a także losowo wybranych próbek oraz selektywnie wybranych próbek metodą k najbliższych sąsiadów. Testy zostały przeprowadzone na czterech różnych datasetach oraz wydajność metod została zmierzona z użyciem wiarygodnej metryki.

## Contents

# 1 Metoda tSNE

## 1.1 Czym właściwie jest tSNE?

Algorytm tSNE(t-Distributed Stochastic Neighbor Embedding) którego autorami są Laurens van der Maaten oraz Geoffrey Hinton bazuje na metodzie SNE, której głównym założeniem jest reprezentacja wielowymiarowych danych w możliwy do zobrazowania dla człowieka dwu- lub trzy-wymiarowej przestrzeni. Osiąga się to poprzez modelowanie wysoko wymiarowych obiektów poprzez dwu- lub trzy-wymiarowe punkty w taki sposób, że zbliżone obiekty modelowane są poprzez bliskie sobie punkty, a oddalone obiekty modelowane są poprzez oddalone od siebie punkty z dużym prawdopodobieństwem.

## 1.2 Algorytm tSNE - podstawy matematyczne

Algorytm tSNE w dużym uproszczeniu sprowadza się do następujących kroków:

- Algorytm tSNE konwertuje odległości między parami punktów w funkcję rozkładu prawdopodobieństwa określająca podobieństwo pomiędzy parami punktów.
- Rozbieżność między podobieństwem wysoko wymiarowych danych z nisko-wymiarowymi danymi jest mierzona poprzez dywergencje Kullbacka-Leiblera i minimalizowana metodą gradientową poszukiwania minimum lokalnego

Mamy dany zbiór wejściowy $X = \{x_1, x_2...x_n\}$ gdzie dla każdego $x_i \in \mathbb{R}^D$ jest D-wymiarowym wektorem. Zbiór ten zostanie przekształcony do postaci $Y = \{y_1, y_2...y_n\}$ gdzie każde $y_i \in \mathbb{R}^d$ jest d-wymiarowym wektorem oraz $d \ll D$ (zazwyczaj d = 2 lub 3). Podobieństwo pomiędzy parą punktów wejściowych $x_i$ oraz $x_j$ oznaczamy poprzez $p_{j/i}$, które jest prawdopodobieństwem wybrania $x_j$ jako sąsiada $x_i$ według funkcji gęstości prawdopodobieństwa na rozkładzie normalnym gdzie $x_i$ stanowi centrum. $p_{j/i}$ definiujemy jako:

$$p_{j|i} = \frac{exp(\frac{-d(x_i,x_j)^2)}{2\sigma_i^2})}{\sum_{k\neq i}^{n} exp(\frac{-d(x_i,x_k)^2)}{2\sigma_i^2}))} \quad p_{i|i} = 0 \quad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

gdzie:
$d(x_i, x_j)$ - odległość pomiędzy punktami $x_i$ oraz $x_j$ w oryginalnym wymiarze
$\sigma_i$ - wariancja dla punktu $x_i$

Aby orzymać zbiór wyjściowy Y, losujemy $n$ punktów w docelowym wymiarze i dla każdego z nich wyznaczamy podobną funkcję gęstości prawdopodobieństwa (tym razem jest to rozkład Studenta):

$$q_{ij} = \frac{(1 + d(y_i, y_j)^2)^{-1}}{\sum_{k\neq l}^{n}((1 + d(y_k, y_l)^2)^{-1})}$$

gdzie:
$d(y_i, y_j)$ - odległość pomiędzy punktami $y_i$ oraz $y_j$ w docelowym wymiarze

W ten sposób otrzymujemy łączone rozkłady gęstości prawdopodobieństwa P i Q dla wszystkich punktów ze zbiorów odpowiednio X i Y. Podobieństwo między nimi (a właściwie dowolnymi 2 rozkładami) określa dywergencja Kullbacka-Leiblera:

$$C = KLDIV(P||Q) = \sum_{i}^{n}\sum_{j}^{n} p_{ij} log\frac{p_{ij}}{q_{ij}}$$

która staje się naszą funkcją kosztu, którą chcemy zminimalizować, robi się to algorytmem spadku po gradiencie (*Gradient Descent*). Pochodna cząstkowa C dla $y_i$ to:

$$\frac{\delta C}{\delta y_i} = 4\sum_{j}^{n}(p_{ij} - q_{ij})(y_i - y_j)(1 + d(y_i - y_j)^2))^{-1}$$

### 1.3 Sposób wyboru wariancji

Wariancję $\sigma_i$ dla punktu $x_i$ wybiera się na podstawie parametru algorytmu ustawianego przez użytkownika zwanego *Perplexity*. Definjujemy:

$$Perp(i) = 2^{H(P_i)} \quad H(P_i) = \sum_{j} p_{j|i} log(\frac{1}{p_{j|i}})$$

gdzie:
$H(P_i)$ - entropia Shannona dla zmiennej losowej $P_i$

Dla rozkładu normalnego im większa entropia, tym większa wariancja, tym "grubsze ogony" funkcji dzwonowej, tym większe prawdopodobieństwo wybrania bardziej odległych sąsiadów punktu $x_i$. Zazwyczaj dla wszystkich punktów $Perp(i)$ ustawiane jest na taką samą wartość $p$. Im mniej "gęsty" jest nasz zbiór danych tym $Perp$ powinno być większe.

### 1.4 Workflow metody tSNE

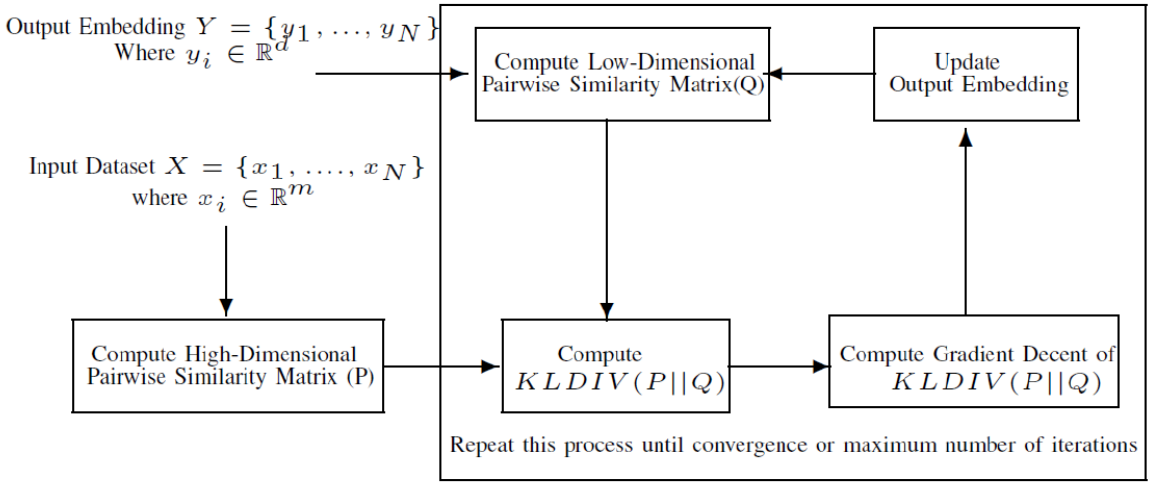


Figure 1: Algorytm tSNE - workflow modelu

### 1.5 Pseudokod metody tSNE

## 2 Metoda LION tSNE

Algorytm t-SNE nie odpowiada na pytanie jak dodawać nowe dane (lub wizualizować dynamiczne zmiany danych) do utworzonego modelu. Z pomocą przychodzi metoda LION tSNE (Local Interpolation with Outlier coNtrol t-Distributed Stochastic Neighbor Embedding). Korzysta ona z 2 metod dodawania nowych punktów:

- Dla inlierów czyli punktów które mogą potencjalnie należeć do jakiegoś klastra - Inverse Distance Weight Interpolation (IDW)
- Dla outlierów specjalna heurystyka (Outlier Placement) oszacowania ich pozycji na wizualizacji zapewniająca odpowiednią odległość od innych punktów

**Algorithm 1**: Simple version of t-Distributed Stochastic Neighbor Embedding.

**Data**: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
**Result**: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
**begin**
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  **for** $t$=1 **to** $T$ **do**
    compute low-dimensional affinities $q_{ij}$ (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  **end**
**end**

Figure 2: Algorytm tSNE - pseudokod

### 2.1 Pseudokod metody LION tSNE

Z oryginalnego datasetu wybierane są **losowo** punkty i na ich podstawie tworzone jest mapowanie do niższego wymiaru (za pomoca standardowego algorytmu t-SNE). Następnie:

**Algorithm 1** LION-tSNE - General Approach

**function** LION-TSNE($x$, $p$, $r_x$, $r_close$, $X_{train}$, $Y_{train}$)
  $neighbor\_indices = select\_neighbors\_in\_radius(x, X_{train}, r_x)$
  $X_{neighb} = X_{train}[neighbor\_indices]$
  $Y_{neighb} = Y_{train}[neighbor\_indices]$
  **if** $len(neighbor\_indices) > 1$ **then**
    $y = local\_IDW\_interpolation(X_{neighb}, Y_{neighb})$
  **else if** $len(neighbor\_indices) == 1$ **then**
    $y = single\_neighbor\_placement(X_{neighb}, Y_{neighb})$
  **else**
    $y = outlier\_placement()$
  **end ifreturn** $y$
**end function**

Figure 3: Algorytm tSNE - pseudokod

### 2.2 LION tSNE - IDW

Gdy algorytm zadecyduje, że dany punkt jest inlierem, stosowana jest technika Inverse Distance Weight Interpolation (IDW). Dla nowego punktu $x$ jego pozycja F(x) w nowym wymiarze jest wyliczana następująco:

$$F(x) = \sum_{d(x-x_i) \leq r_x} w_i(x) y_i, gdzie \quad w_i(x) = \frac{d(x-x_i)^{-p}}{\sum_{d(x-x_j) \leq r_x} d(x-x_j)^{-p}}$$

$r_x$ - promień odległości dla lokalnego sąsiedztwa w którym jest dodawany nowy punkt, parametr wywołania
$p$ - parametr wywołania

Zauważmy, że gdy $x \to x_i$ to $d(x - x_i)^{-p} \to \infty$ i $w_i(x) \to 1$ i $F(x) \to y_i$

### 2.3 LION tSNE - outlier placement

Główną ideą outlier placementu jest następująca: jeśli x jest outlierem to odpowiadająca jej po zmapowaniu wartość y powinna także być zwizualizowana jako outlier. Aby odnaleźć takie wartości, musimy znaleźć położenie y, takie że nie ma żadnych sąsiadów w promieniu $r_y$. Promień $r_y$ jest jednym z parametrów algorytmu i jeśli zostanie wybrana zbyt duża wartość, wtedy z racji na duże odległości między punktami, zmniejsza się czytelność wykresu natomiast jeśli promień zostanie wybrany zbyt mały klastry i wartości odstające mogą być nierozróżnialne. Dlatego też wartość $r_y$ może być wyznaczona na podstawie pewnego percentylu rozkładu odległości najbliższych sąsiadów w przestrzeni y (np. 95-ty lub 99-ty percentyl).

## 3 kNN sampling

### 3.1 kNN sampling w LION tSNE

Autorzy artykułu zasugerowali, że sposób próbkowania danych zastosowany w tSNE jest niewystarczający chociażby przy dynamicznie zmieniających się danych. Jednocześnie przedstawili kilka kroków potrzebnych do zrealizowania idei k-NN samplingu.

1. Czyszczenie danych za pomocą eliminacji redundantnych punktów i zainicjalizowanie pustych zmiennych odpowiednimi wartościami.
2. Dobór właściwego zbioru treningowego poprzez k-NN sampling.
3. Rzutowanie zbioru treningowego na nisko wymiarową przestrzeń oraz dodanie nowych punktów do modelu tSNE.
4. Dla nowych danych interpolacja ich do istniejącego modelu za pomocą LION-tSNE
5. Wyliczenie precyzji k-NN samplingu dla całego zbioru danych.

### 3.2 Wybór zbioru treningowego poprzez k-NN sampling

Zaproponowana idea k-NN samplingu opiera się na wyliczeniu Nearest Neighbor score (NNscore) oraz Mutual Nearest Neighbor score (MNNscore). Mamy dany graf skierowany $G = (V, E)$, gdzie krawędź $E(v_1, v_2)$ oznacza, że $v_2$ jest sąsiadem $v_1$, natomiast sąsiedztwo $v_1$ oznaczamy jako $N_{v1}$. Stopień wychodzący każdego wierzchołka jest równy k, a stopień wchodzący zależy od wartości współczynnika sąsiedztwa innych wierzchołków. W celu wyznaczenia optymalnego zbioru treningowego dobieramy odpowiednie k oraz zbiór punktów wejściowych mapujemy na wierzchołki grafu k-NN.

Wyznaczamy NN_score, który odpowiada stopniu wchodzącemu wierzchołka $x_i$:

$$NNscore(x_i) = |\{x_j | x_i \in N_{x_j}\}|. \forall_{j \neq i} x_j \in X$$

Gdzie X jest zbiorem danych wejściowych a $N_{x_j}$ sąsiedztwem $x_j$ Następnie wyliczamy MNNscore, który dla $x_i$ jest równy conajwyżej k:

$$MNNscore(x_i) = |\{x_j | x_i \in N_{x_j} \wedge x_i | x_j \in N_{x_i}\}|. \forall_{j \neq i} x_j \in X$$

Ostatecznie dobór punktu $x_i$ jako próbki treningowej oraz jego sąsiedztwo jest dany jako:

$$train\_sample = first\_index\{argmax_{x_i \in X}\{NNscore(x_i)\} \cap argmax_{x_i \in X}\{MNNscore(x_i)\}\}$$

### 3.3 Dodanie nowych punktów do modelu tSNE

W LION-tSNE stosujemy IDW i Outlier Placement. Nowe dane mogą być dodawane do modelu tSNE na dwa sposoby, na podstawie wyliczonych parametrów $r_{xNN}, r_{yNN}$ oraz $r_{close}$. Wartość $r_{xNN}$ oznacza minimalny promień zbioru wejściowego, co z pomocą pewnej heurystyki pozwala na wskazanie czy dany punkt jest wartością właściwą czy odstającą. Parametr $r_{close}$ pozwala na zidentyfikowanie próbek odstających powiązanych z wartościami odstającymi znajdującymi się w modelu tSNE. Natomiast $r_{yNN}$ wyznacza minimalną odległość pomiędzy punktami ze zbioru wejściowego, a próbkami odstającymi oraz dla wartości odstających pomiędzy nimi.

### 3.4 Wyliczenie precyzji k-NN samplingu

Nowo dodane próbki są umieszczane pośród próbek treningowych o podobnej charakterystyce. Aby to zbadać, dla każdej nowo dodanej próbki, wyliczany jest stopień precyzji oraz obserwowany jest procent k sąsiadów o podobnej charakterystyce. Precyzja k-NN zazwyczaj zależy od parametru k, który jest zwykle ustawiony na stałe. Mała wartość k oznacza dobrą precyzje, natomiast jeśli wartość k zaczyna rosnąć to wtedy precyzja maleje. Dlatego też odpowiedni dobór k stanowi ważny element k-NN samplingu.

## 4 Wyniki projektu

W pierwszej części projektu zdecydowaliśmy się na analizę działania metody LION tSNE i zwracanych przez nią rezultatów. Ważnym elementem okazało się sprawdzenie jaki wpływ na wizualizację ma odpowiednie próbkowanie danych wejściowych. Serię eksperymentów przeprowadziliśmy na zbiorach IRIS [5] oraz MNIST [6], a jako próbkę kontrolną wybraliśmy tradycyjne tSNE. W połowie eksperymentów mieliśmy do czynienia z losowo wybranym zbiorem testowym natomiast w drugiej połowie posłużyliśmy kNN samplingiem będącym jedną z części naszego projektu. W dalszej fazie eksperymentów posłużyliśmy się nieco bardziej złożonymi zbiorami danych, a mianowicie Fashion MNIST [7] oraz Reuters [8].

Spróbujmy odpowiedziec na pytanie dlaczego odpowiednie próbkowanie danych jest takie ważne? Losowe próbkowanie polega na pseudolosowym doborze rekordów z wybranego datasetu, co jest obarczone możliwością wyboru nierównych podzbiorów danych klas oraz ryzykiem dużego stopnia wariancji danych.

```
mnist_random_df['target'].value_counts()

Out[ ]:

7    230
1    228
9    216
3    206
5    193
0    193
8    190
6    188
2    186
4    170
Name: target, dtype: int64
```

(a) losowe próbkowanie

```
mnist_knn_df['target'].value_counts()

Out[ ]:

2    200
0    200
3    200
6    200
9    200
1    200
8    200
7    200
4    200
5    200
Name: target, dtype: int64
```

(b) próbkowania k najbliższych sąsiadów

Figure 4: Podział na poszczególne klasy dla zbioru MNIST

W przypadku przeprowadzonego przez nas kNN samplingu, zauważyliśmy, że zwraca dość obiecujące wyniki, a mianowicie polega na tym, że wyliczamy za pomocą Nearest Centroid Classifier (pochodzącego z biblioteki scikit-learn) centroidy dla poszczególnych klas. Następnie dla każdej klasy wyszukujemy k najbliższych sąsiadów centroida klasy i umieszczamy je w zbiorze testowym. Tak przeprowadzone próbkowanie rozwiązuje oba problemy wspomniane powyżej (występujące w losowym próbkowaniu).

### 4.1 Przeprowadzone eksperymenty

Poszczególne wizualizacje przedstawione poniżej zostały uporządkowane w następopujący sposób: górny wiersz - losowy sampling, dolny wiersz - knn sampling, a także odpowiednio od lewej tSNE, LION tSNE, PCA + LION tSNE, MDS + LION tSNE.

- IRIS Dataset [5]

  Jest to stosunkowo mały zbiór zawierający opisy czterech własności, trzech różnych gatunków irysów. Z racji na niewielki rozmiar, zbiór okazał się przydatny do testowania samplingu danych (wybieraliśmy 120

spośród 150 dostępnych rekordów), jednak wydaję się zbyt mały dla niektórych metod, aby stworzyć w pełni wiarygodny model.

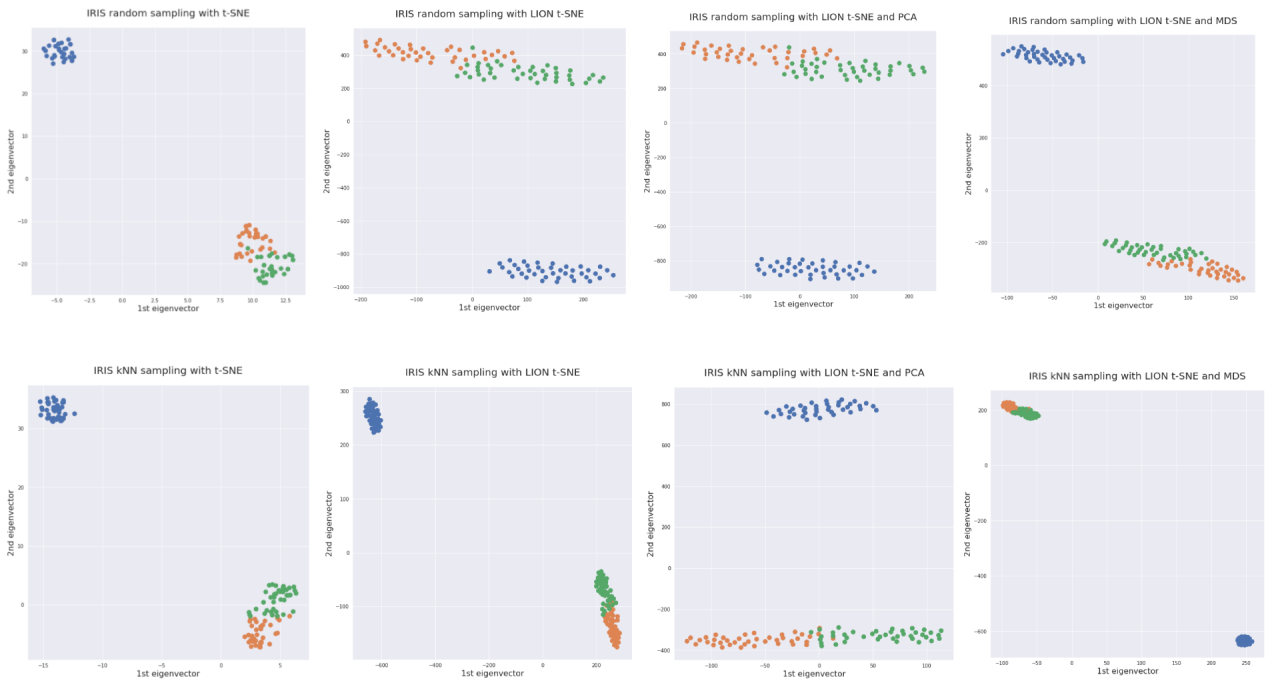


Figure 5: Wizualizacja IRIS Dataset za pomocą różnych metod

- MNIST Dataset [6]

  Dataset zawierający opisy cyfr pisanych reprezentowanych jako obrazki 28x28 pikseli. W przypadku eksperymentów na tym zbiorze danych posłużyliśmy zbiorem testowym zawierającym 2000 rekordów (mniej więcej 200 obrazków na daną klasę).

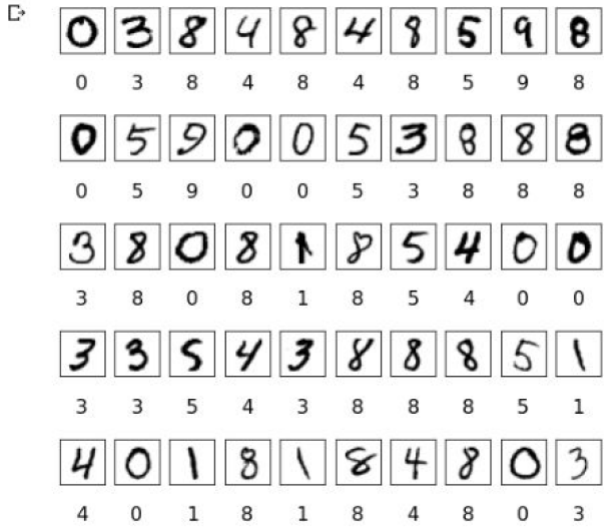


Figure 6: Wizualizacja przykładowych danych ze zbioru MNIST

- Fashion MNIST Dataset [7]

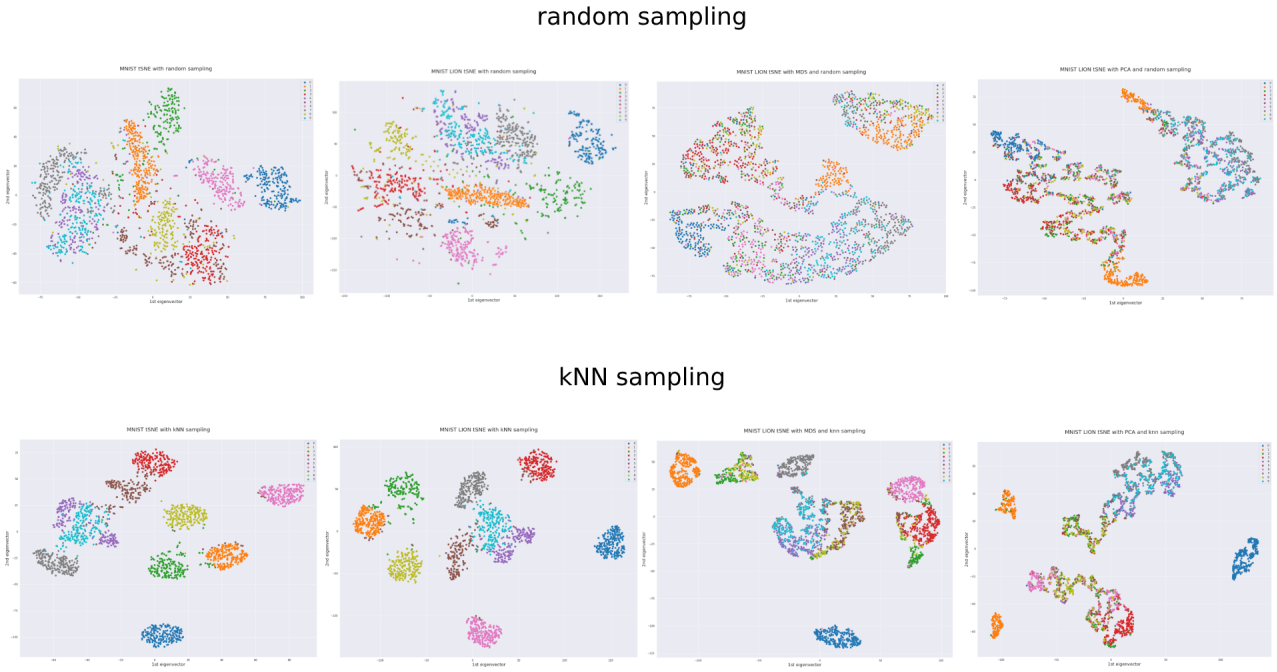


Figure 7: Wizualizacja MNIST Dataset za pomocą różnych metod

Zbiór danych Fashion MNIST zawiera tysiące zdjęć ubrań pochądzących z zasobów sklepu Zalando. Każde zdjęcie zostało przekonwertowane na macierz 28x28 pikseli, w której wartości są z przedziału 0-255, co odpowiada skali szarości tych obrazków. Co więcej, mamy tu do czynienia z 10 różnymi rodzajami ubrań, a mianowicie:

1. T-shirt/top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boo

Zatem zbiór ten wzoruję się częściowo na zbiorze MNIST jednak jest zdecydowanie bardziej skomplikowany, co jest niejednokrotnie pożądane w niektórych zadaniach związanych z uczeniem maszynowym. Oto kilka przykładowych zdjęć z tego zbioru:

- Reuters Dataset [8]

Dataset stworzony z dokumentów pochodzącejz kolekcji Reuters-21578, która ukazała się w głównym kanale publikacji agencji Reuters w 1987. Dokumenty zostały złożone i zindeksowane na kategorię przez personel agencji Reuters Ltd. (Sam Dobbins, Mike Topliss, Steve Weinstein) oraz Carnegie Group, Inc. (Peggy Andersen, Monica Cellio, Phil Hayes, Laura Knecht, Irene Nirenburg) w 1987.

W 1990 dokumenty zostały opublikowane przez Reuters i CGI do celów badawczych dla Information Retrieval Laboratory (W. Bruce Croft, Director) z wydziału Informatyki Uniwersytetu Massachusetts w Amherst. Formatowanie dokumentów i stworzenie zwiuązanych z tym plików zostało wykonane w 1990 przez Davida D. Lewisa and Stephena Hardinga z Information Retrieval Laboratory.

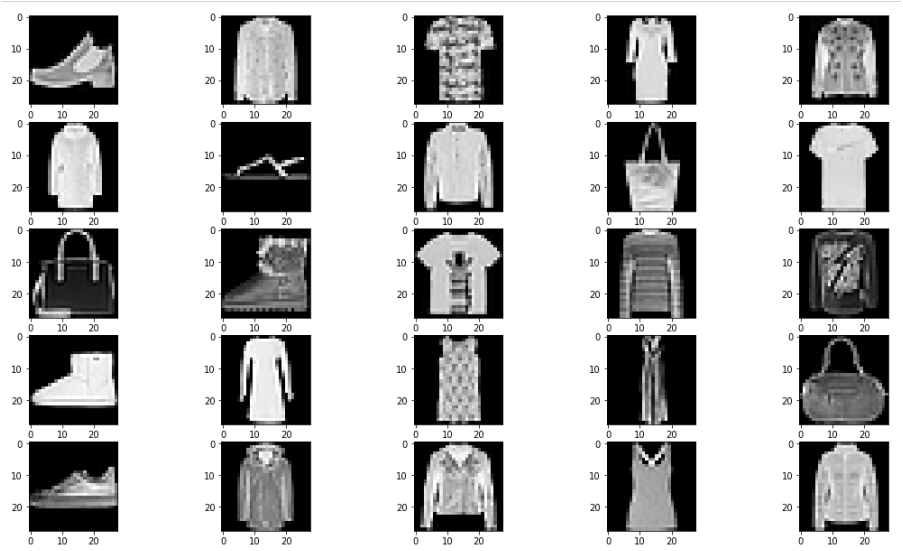

Figure 8: Wizualizacja przykładowych danych ze zbioru Fashion MNIST

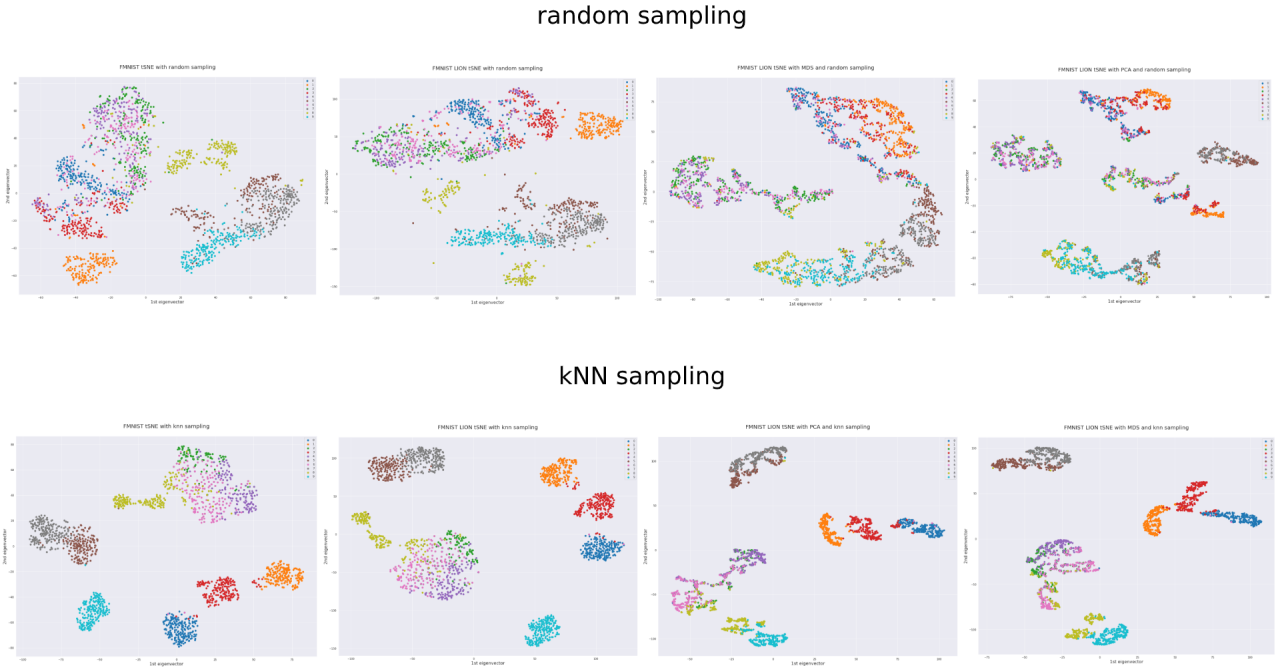


Figure 9: Wizualizacja Fashion MNIST Dataset za pomocą różnych metod

### 4.2 Wykorzystane metryki

TODO

## 5 Wnioski

TODO

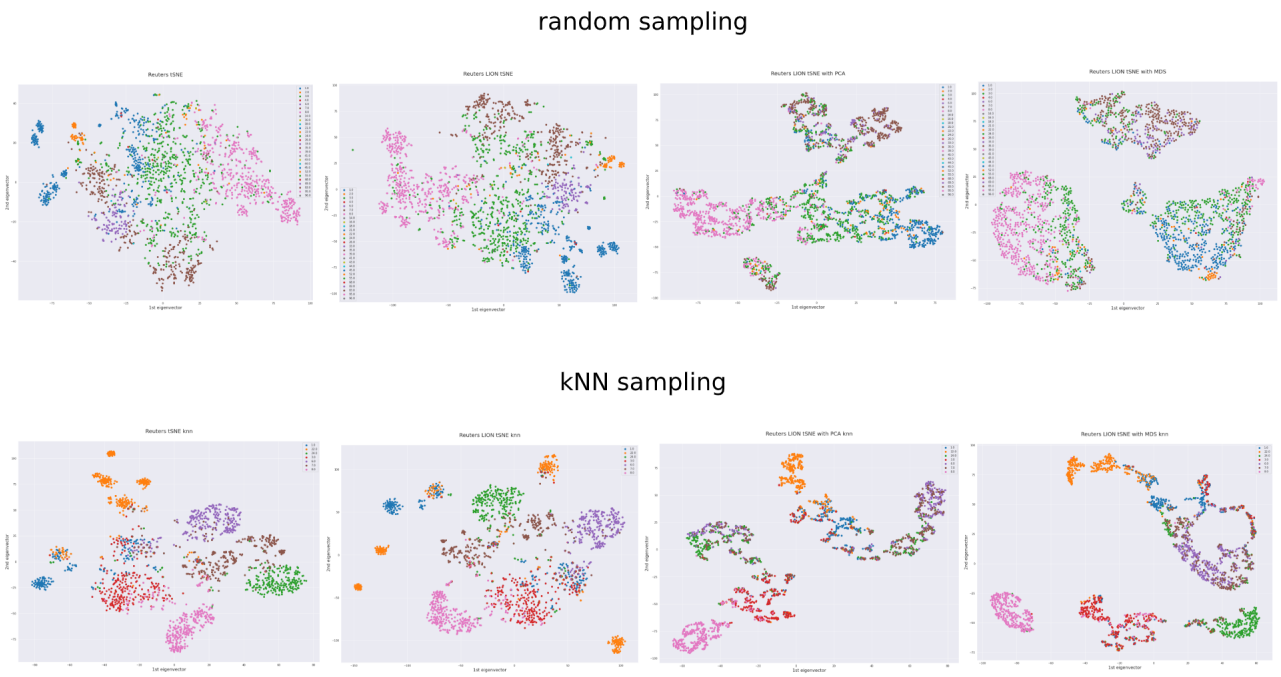


Figure 10: Wizualizacja Reuters Dataset za pomocą różnych metod

## Źródła


[1] Bheekya Dharamsotu ; K. Swarupa Rani ; Salman Abdul Moiz ; C. Raghavendra Rao Paper: k-NN Sampling for Visualization of Dynamic Data Using LION-tSNE. *https://ieeexplore.ieee.org/abstract/document/8990391*

[2] Laurens van der Maaten ; Geoffrey Hinton Paper: Visualizing Data using t-SNE *http://www.cs.toronto.edu/ hinton/absps/tsne.pdf*

[3] Andrey Boytsov ; François Fouquet ; Yves Le Traon Paper: Visualizing and Exploring Dynamic High-Dimensional Datasets with LION-tSNE. *https://github.com/andreyboytsov/LION-tSNE*

[4] Andrey Boytsov LION tSNE Github Repository. *https://github.com/andreyboytsov/LION-tSNE*

[5] IRIS Dataset. *https://archive.ics.uci.edu/ml/datasets/iris*

[6] MNIST Dataset. *http://yann.lecun.com/exdb/mnist/*

[7] FASHION MNIST Dataset. *https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/*

[8] REUTERS Dataset. *https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection*