AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

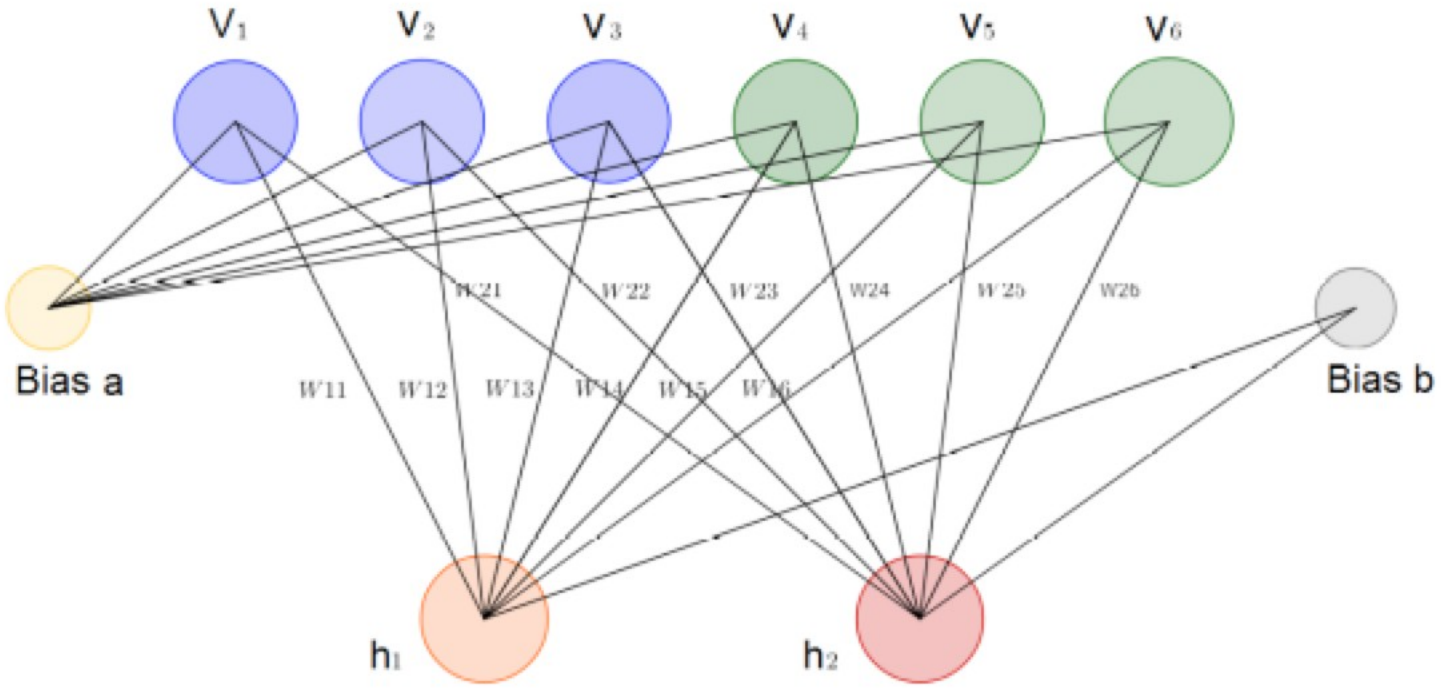
Autoencoders

Dariusz Kucharski
Katedra Automatyki i Robotyki

Kraków, 21.11.2019

Ograniczona maszyna Boltzmanna

- **Ogólna architektura sieci**



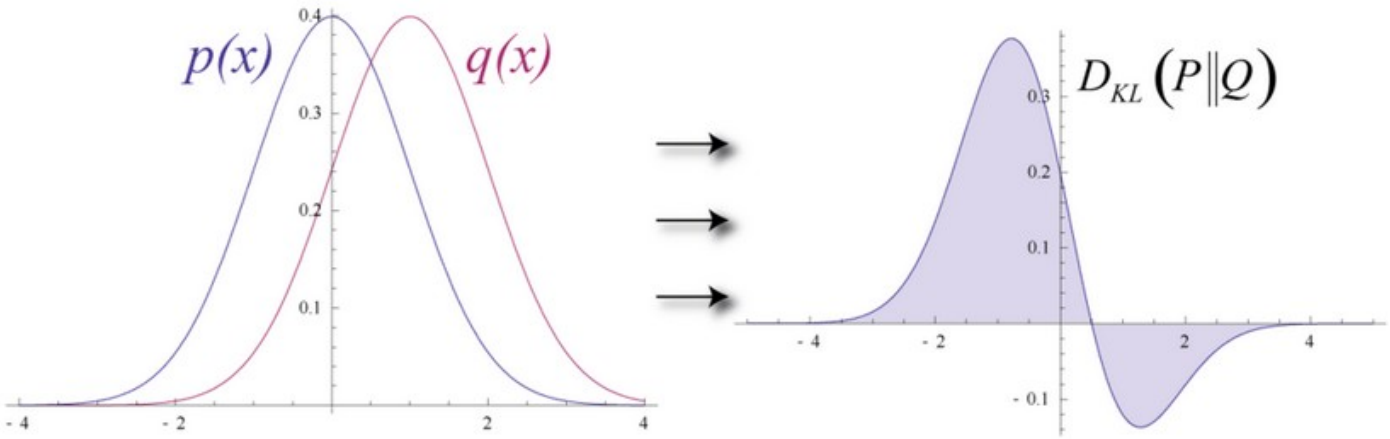
Architektura ograniczonej maszyny Boltzmanna [2]

Ograniczona maszyna Boltzmanna

- Odpowiedzi warstwy ukrytej interpretujemy jako prawdopodobieństwo p(a|x; w)
- Przy obliczeniu rekonstrukcji możemy wynik rekonstrukcji interpretować jako p(x|a; w)
- Suma tych prawdopodobieństw to wspólny rozkład prawdopodobieństwa p(a, x) (joint probability)

Ograniczona maszyna Boltzmanna

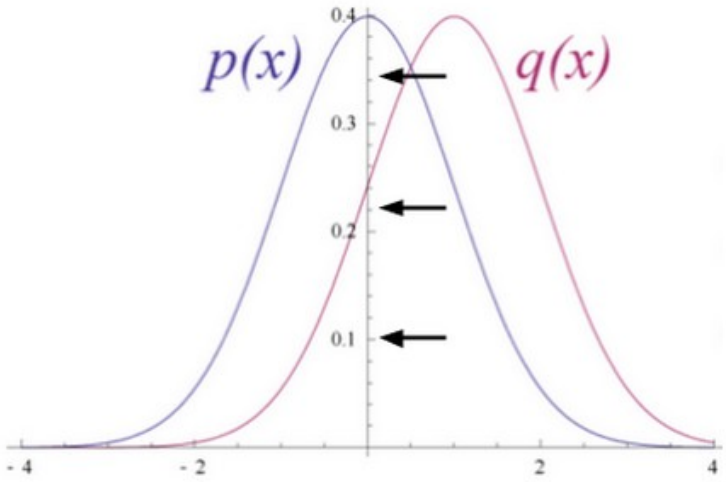
- **Jeśli klasyfikacja to odgadywanie klasy (discriminative learning), regresja to szacowanie liczby, to rekonstrukcja to szukanie rozkładu prawdopodobieństwa wejścia (generative learning)**



Różnica w rozkładzie prawdopodobieństwa sygnału oryginalnego i rekonstrukcji [1]

Ograniczona maszyna Boltzmanna

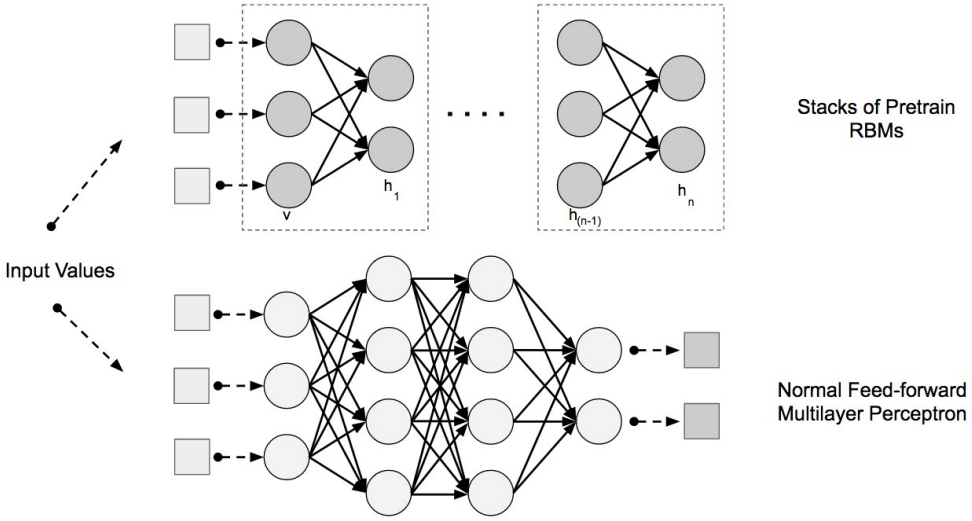
- **Jeśli klasyfikacja to odgadywanie klasy (discriminative learning), regresja to szacowanie liczby, to rekonstrukcja to szukanie rozkładu prawdopodobieństwa wejścia (generative learning)**



Dążymy to minimalizacji tej rekonstrukcji [1]

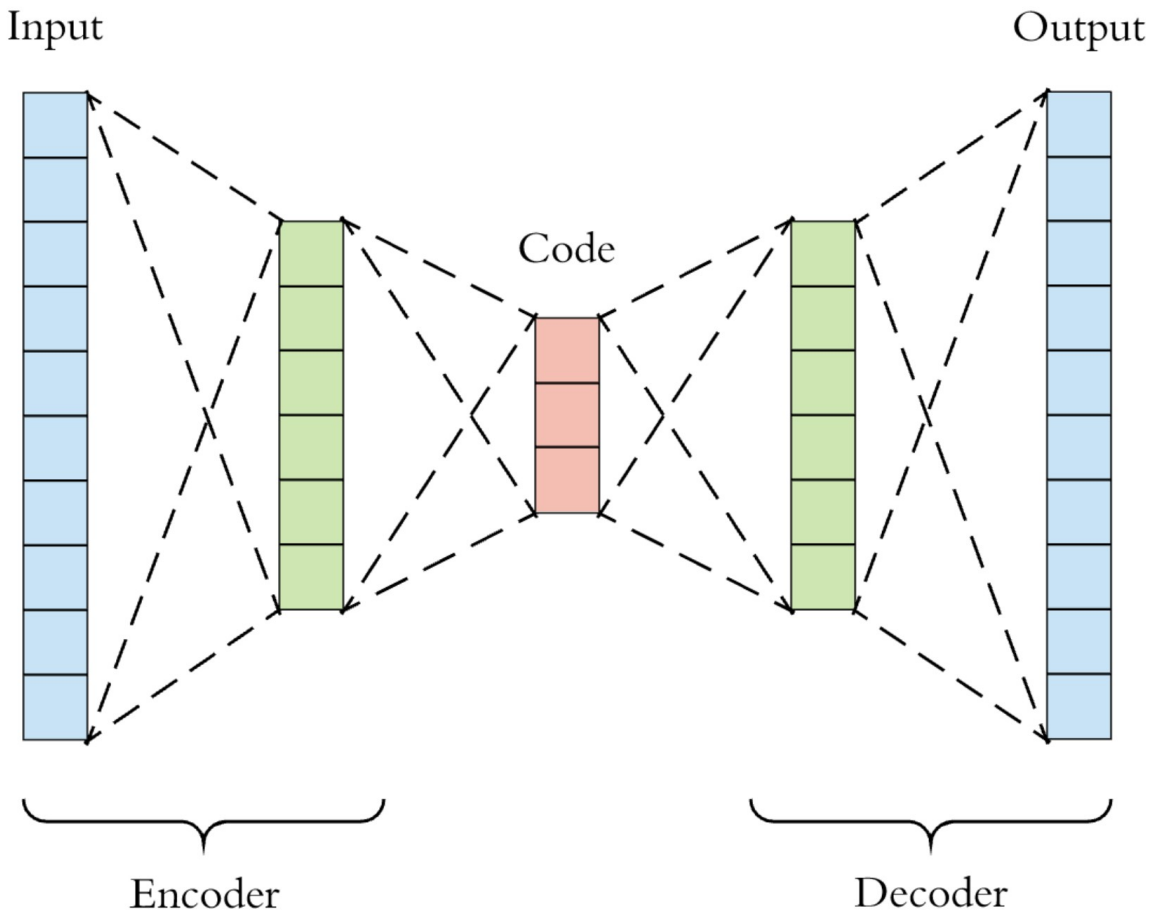
Deep Belief Network

- Powstaje poprzez wstępnej uczenie Ograniczonych Maszyn Boltzmanna
- Wejście na kolejną maszynę boltzmana jest wyjściem z poprzedniej
- Po wyuczeniu odpowiedniej ilości warstw, ograniczone maszyny boltzmana łączone są w sieć głęboką (często dodaje się też klasyfikator) i następuję fine tuning – douczenie sieci, aby otrzymać jeszcze lepszą skuteczność



Przykład DBN [4]

Autoenkodery



Przykład autoenkodera[1]

Autoenkodery

Sieć neuronowa, używana do efektywnego kodowania danych wejściowych (odnalezienia reprezentacji charakteryzującej dany zbiór, często o mniejszym wymiarze niż wejście) w sposób nienadzorowany.

Możemy wyróżnić enkoder (część sieci odpowiedzialna za kodowanie), który dokonuje transformacji wejścia do pewnej ukrytej reprezentacji z (stanowiącej pewnego rodzaju kod):

$$z = f_\theta(x) = \sigma(Wx + b)$$

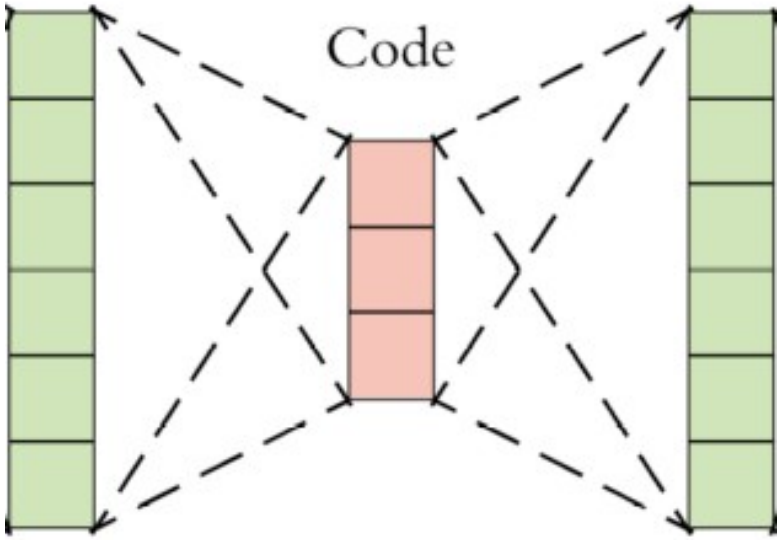
Oraz dekoder, który próbuje otworzyć te dane wejściowe na podstawie ich ukrytej reprezentacji z

$$x' = f_{\theta'}(h) = \sigma'(W'z + b')$$

Minimalizując funkcję błędu między wejściem i rekonstrukcją.

W, W', b, b' - parametry sieci; σ, σ' – funkcje aktywacji

Autoenkodery

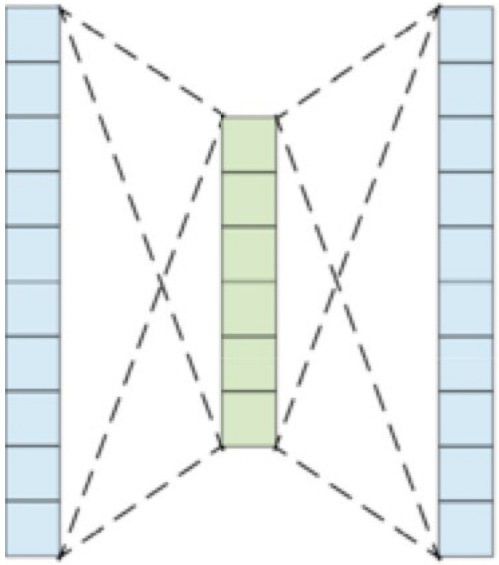


Przykład autoenkodera[1]

Dla W’ = W^T w początkowej fazie uczenia – połowa mniej parametrów do optymalizacji

Stacked autoencoders

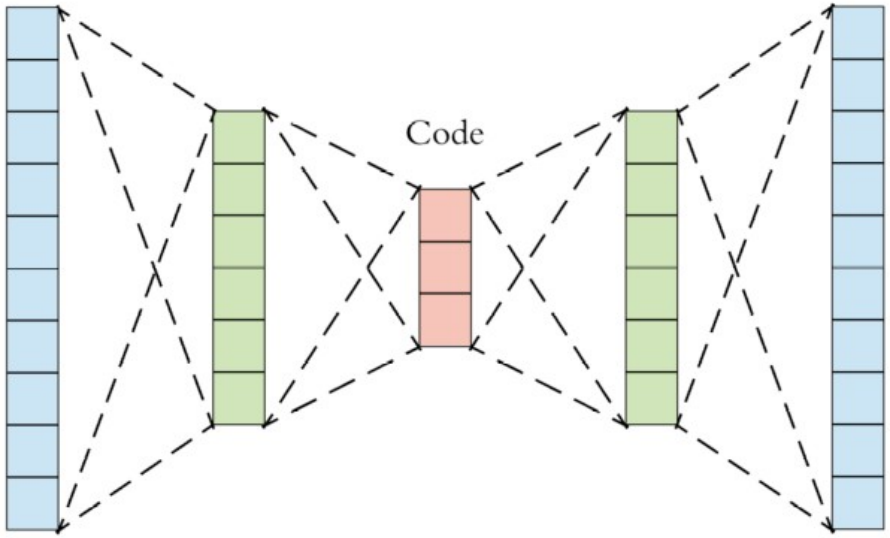
- **Uczenie autoenkoderów wielowarstwowych**
  - **Uczenie sieci z jedną warstwą ukrytą**



Przykład autoenkodera[1]

Stacked autoencoders

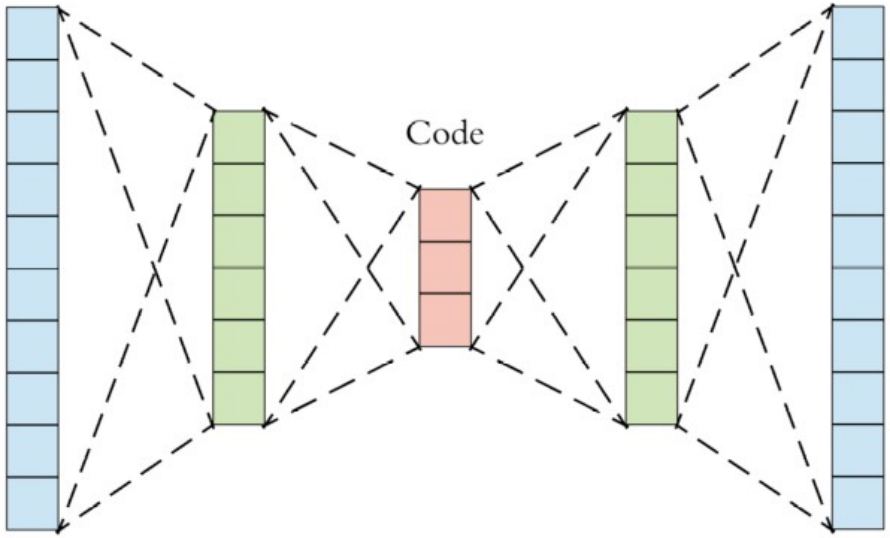
- **Uczenie autoenkoderów wielowarstwowych**
  - **Uczenie sieci z jedną warstwą ukrytą**
  - **Dodanie kolejnej warstwy ukrytej, zablokowanie wag warstwy już wyuczonej, uczenie nowej warstwy ukrytej**



Przykład autoenkodera[1]

Stacked autoencoders

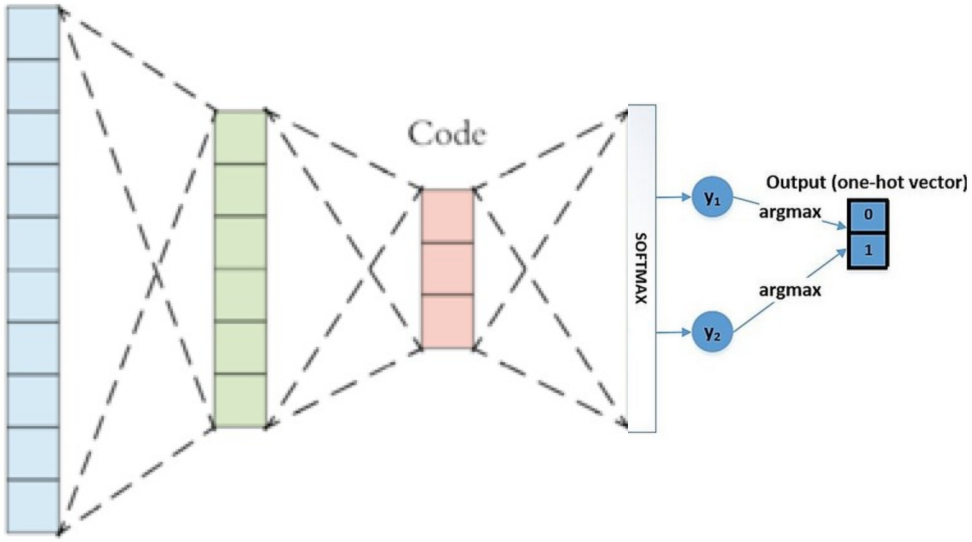
- **Uczenie autoenkoderów wielowarstwowych**
  - **Uczenie sieci z jedną warstwą ukrytą**
  - **Dodanie kolejnej warstwy ukrytej, zablokowanie wag warstwy już wyuczonej, uczenie nowej warstwy ukrytej**
  - **Odblokowanie wag i douczenie calej sieci**



Przykład autoenkodera[1]

Stacked autoencoders

- **Zamiana dekodera na klasyfikator, douczenie go z zablokowanymi wagami enkodera a następnie wykonanie fine tuningu to tzw. uczenie pół nadzorowane**



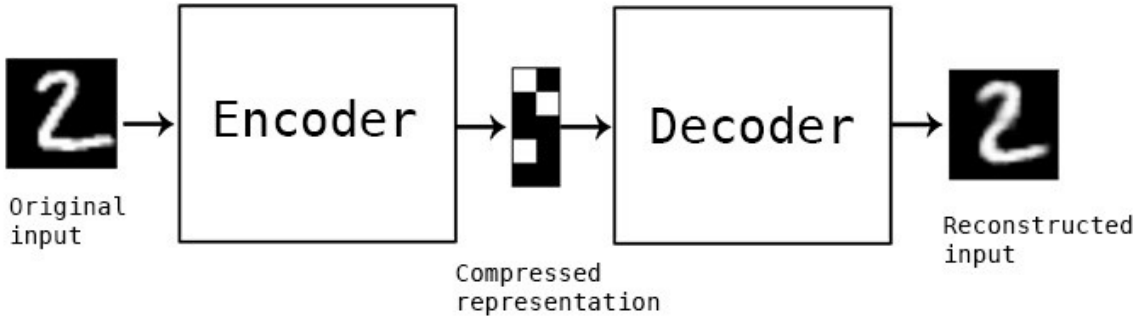
Uczenie półnadzorowane [1], [https://www.researchgate.net/profile/Babak_Bashari_Rad/publication/326914586/figure/fig1/AS:657671334150145@1533812473734/Proposed-neural-network-classifier-with-softmax-output-function-and-a-bias-unit.png]

Stacked autoencoders

- **Należy pamiętać o dobraniu odpowiedniej funkcji kosztu do zadanego problemu**
  - **Średni kwadrat**
  - **Entropia krzyżowa**
  - **Inne?**

Autoencoder

- **Wejście: 28x28x1 → 784**
- **Warstwy ukryte: 32**



Schemat autoenkodera dla zbioru MNIST[2]

Autoencoder

- **Wejście: 28x28x1 → 784**
- **Warstwy ukryte: 32**

Przykładowy wynik autoenkodera z jedną warstwą ukrytą [2]

Deep Autoencoder

- **Wejście: 28x28x1 → 784**
- **Warstwy ukryte: 128 → 64 → 32**

Przykładowy wynik autoenkodera typu deep [2]

Convolutional Autoencoder

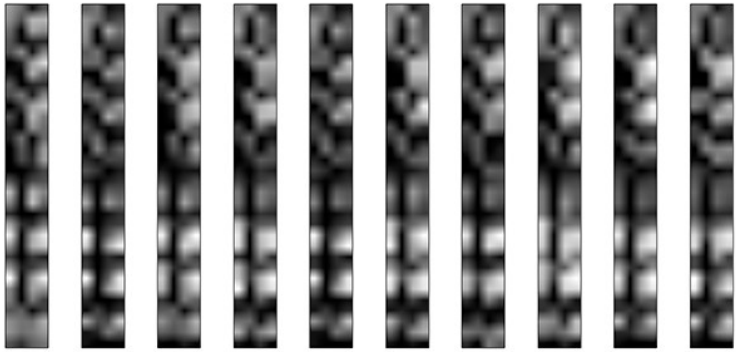
- **Wejście: 28x28x1 → 784**
- **Warstwy ukryte: conv 16 (3, 3) → mp (2, 2) → conv 8 (3, 3) → mp (2, 2) → conv 8 (3, 3) → mp (2, 2)**

Przykładowy wynik autoenkodera konwolucyjnego [2]

Convolutional Autoencoder

- **Wejście: 28x28x1 → 784**
- **Warstwy ukryte: conv 16 (3, 3) → mp (2, 2) → conv 8 (3, 3) → mp (2, 2) → conv 8 (3, 3) → mp (2, 2)**

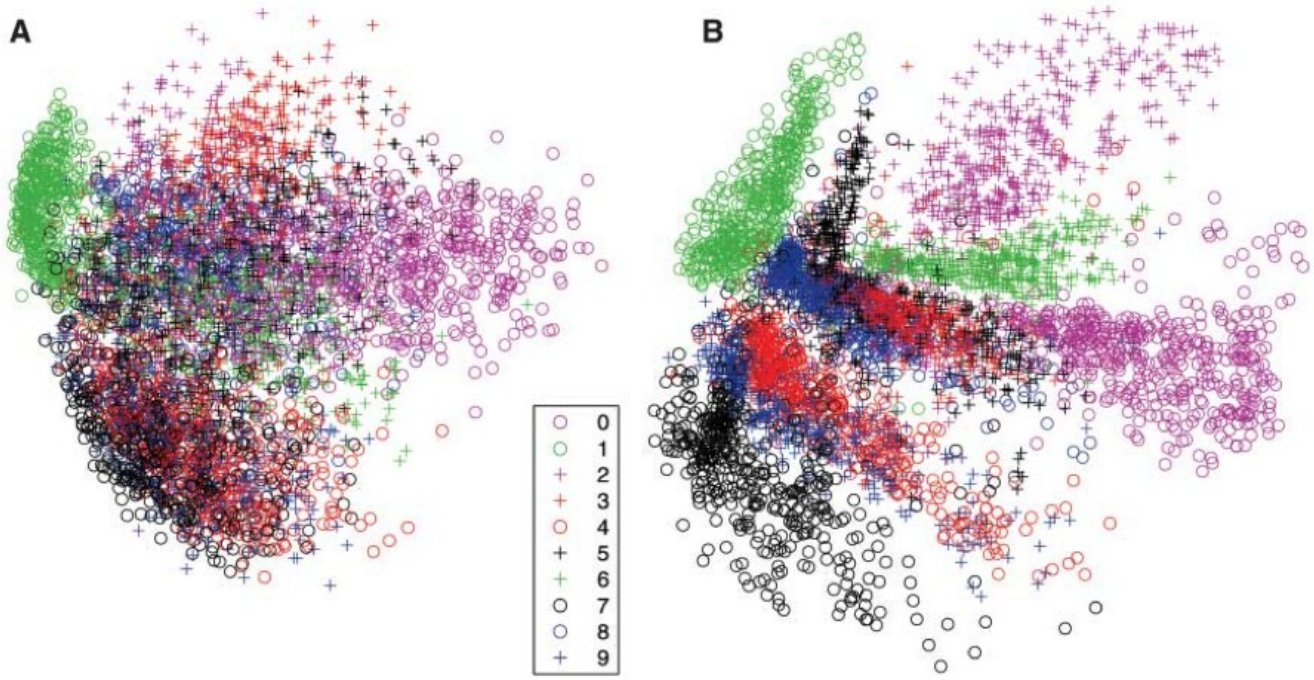
Przykładowy wynik autoenkodera konwolucyjnego [2]



Odpowiedź warstwy latent (8x4x4 → 4x32) [2]

Przykładowy autoenkoder dla zbioru mnist

- **Wejście: 28x28x1 → 784**
- **Warstwy ukryte: 100 → 500 → 250 → 2**



Porównanie dwóch pierwszych składowych PCA (A) oraz zastosowaniu autoenkodera z dwoma neuronami w warstwie latent (B) [3]

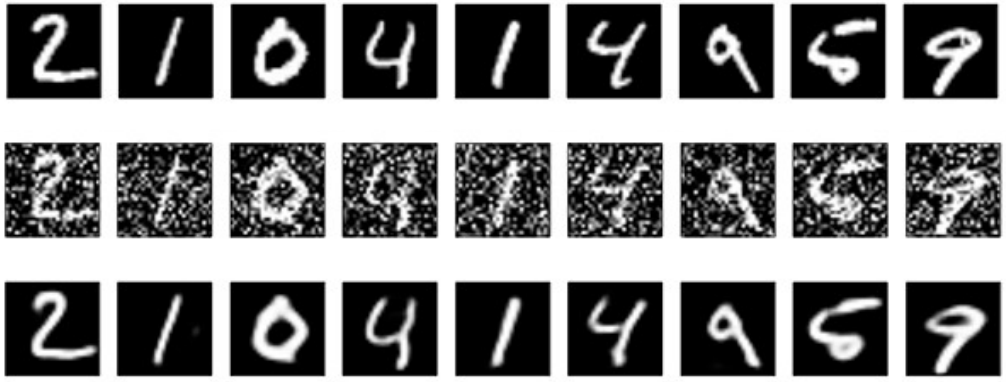
Denoising autoencoders

- **Autoenkodery przygotowane w celu uzupełnienia pewnych informacji bądź usunięcia tych niechcianych**
- **Zakłada się, że poprzez odpowiednie uczenie, sieć neuronowa nauczy się jedynie tych cech, które potrzebne są do rekonstrukcji bez zakłóceń**
- **Czasem stosuje się tę właściwość gdy przy danym zbiorze i pewnej architekturze klasyczny autoenkoder „przepisuje” dane na wyjście (brak wyuczenia ukrytej reprezentacji)**

Denoising autoencoders

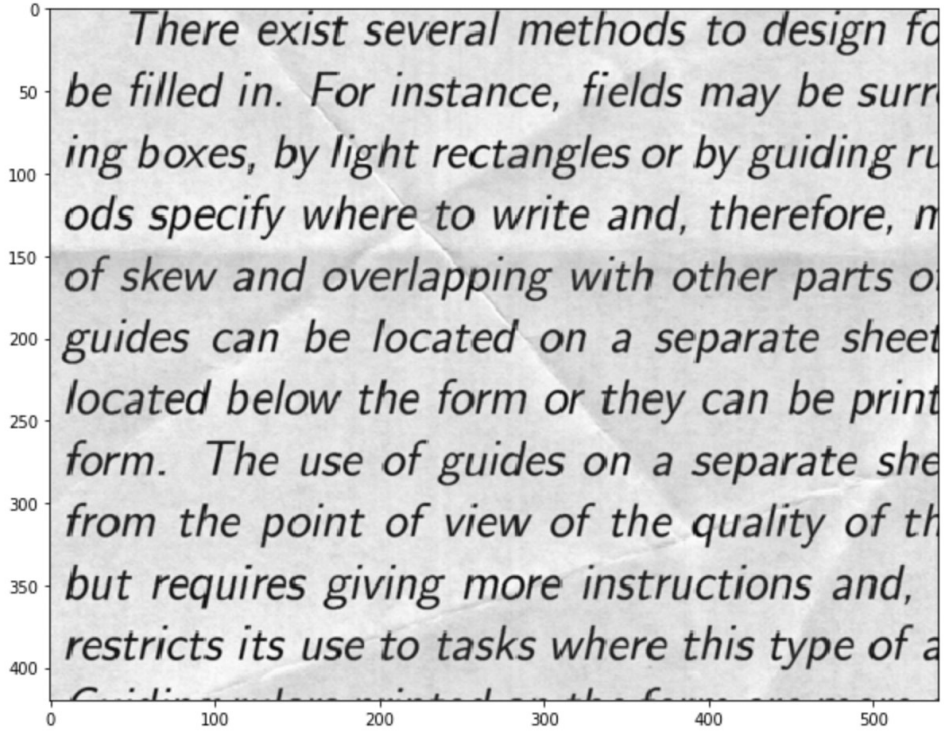
- **Uczenie następuje poprzez zakłócenie oryginalnych danych**
  - **Poprzez zastosowanie dropoutu (30%-50% jednostek wyłączonych)**
  - **„ręcznego” zerowania niektórych wartości**
  - **Zastosowanie szumu (np. salt and pepper)**
- **Błąd rekonstrukcji mierzy się między wyjściem a oryginalnym wejściem (przed dodaniem szumu)**

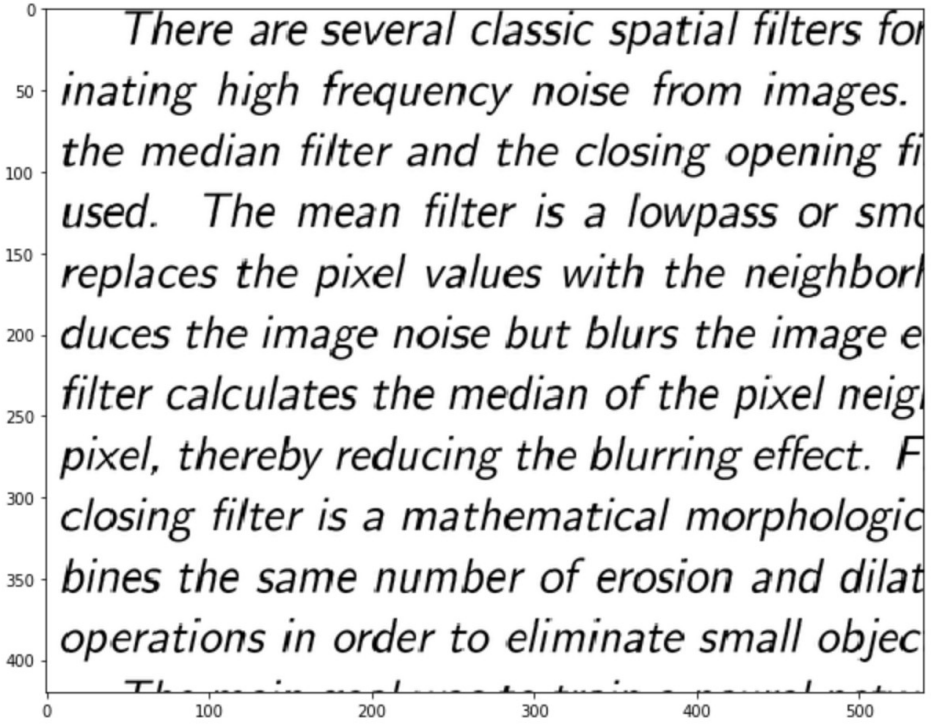
Denoising autoencoders



Przykład odszumienia wejścia przez autoenkoder, funkcja kosztu liczona między obrazem przed zaszumieniem a rekonstrukcją [2]

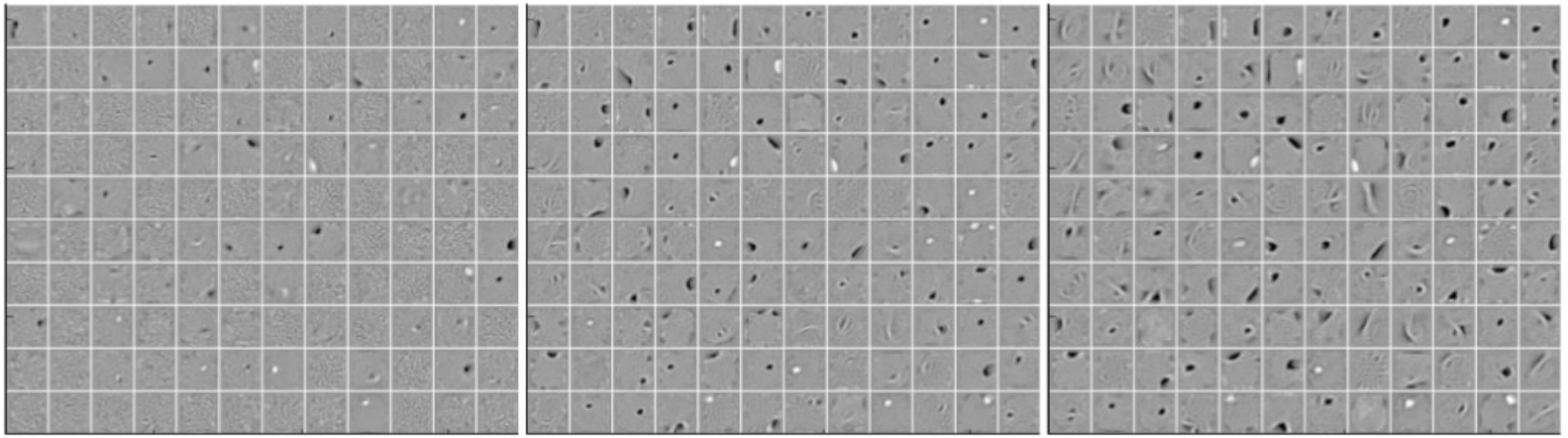
Denoising autoencoders



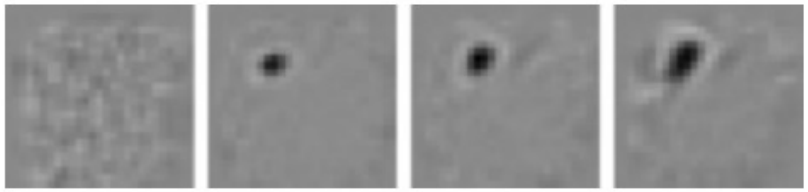


Inny przykład odszumiania [1]

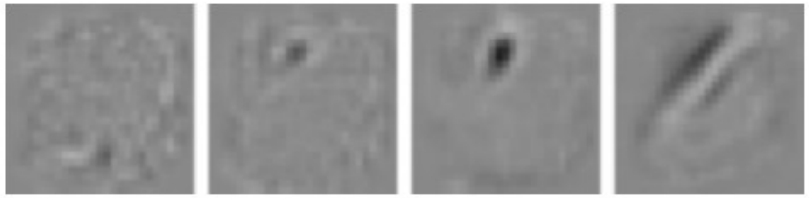
Denoising autoencoders



(a) No destroyed inputs (b) 25% destruction (c) 50% destruction



(d) Neuron A (0%, 10%, 20%, 50% destruction)



(e) Neuron B (0%, 10%, 20%, 50% destruction)

Zachowanie neuronów dla zaszumionego wejścia [4]

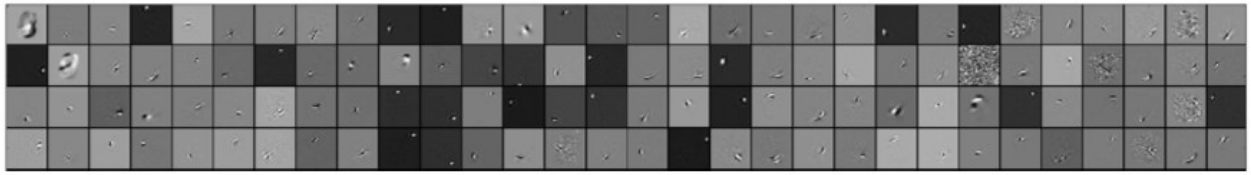
Sparsity autoencoders

- **Warstwa latent może być większa od wejścia**
- **Sieć aktywuje tylko nieliczne neurony dla danych przykładów**
- **Definiowana przez współczynniki sparsity (*ρ*)**
  - **Dodanie regularyzatora za odstępstwo od założonego sparsity (ρ) (w uproszczeniu, sieć jest karana za aktywację zbyt dużej lub zbyt małej ilości neuronów)**

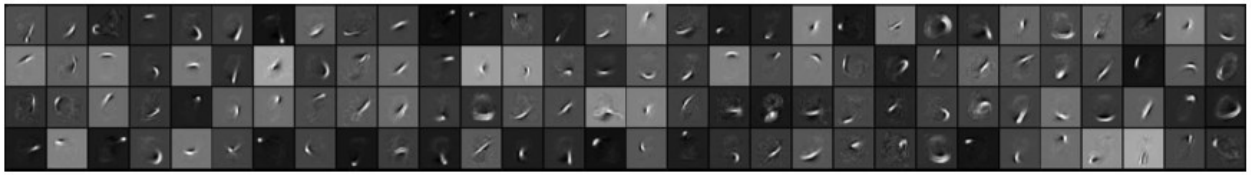
k-Sparse Autoencoders

- Obliczenie odpowiedzi warstwy latent
- Wskazanie k największych aktywacji neuronów
  - Reszta aktywacji jest zerowana
- Rekonstrukcja z wykorzystaniem k wskazanych neuronów z warstwy latent
- Aktualizacja wag

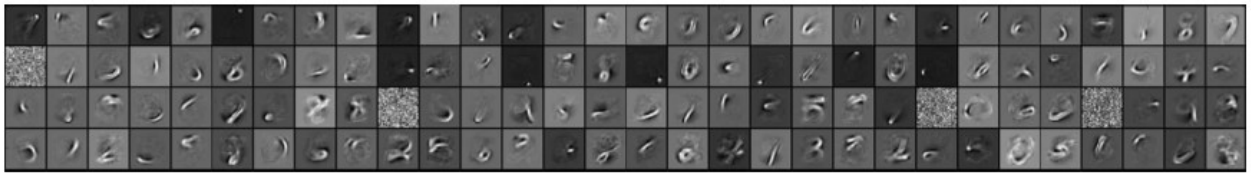
k-Sparse Autoencoders



(a) $k = 70$



(b) $k = 40$



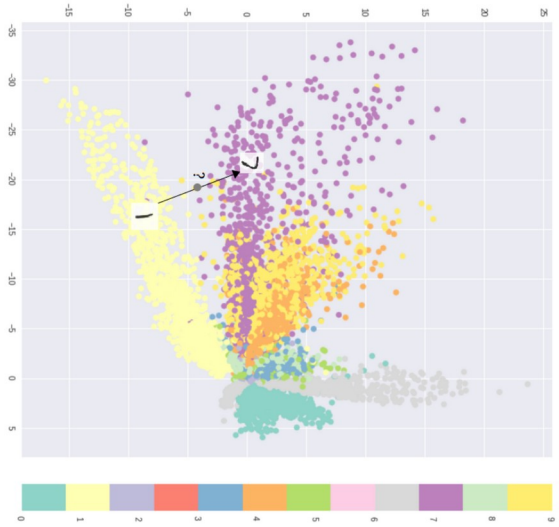
(c) $k = 25$

(d) $k = 10$

Wyuczona informacja przez sieć w zależności od współczynnika k [5]

Variational Autoencoders

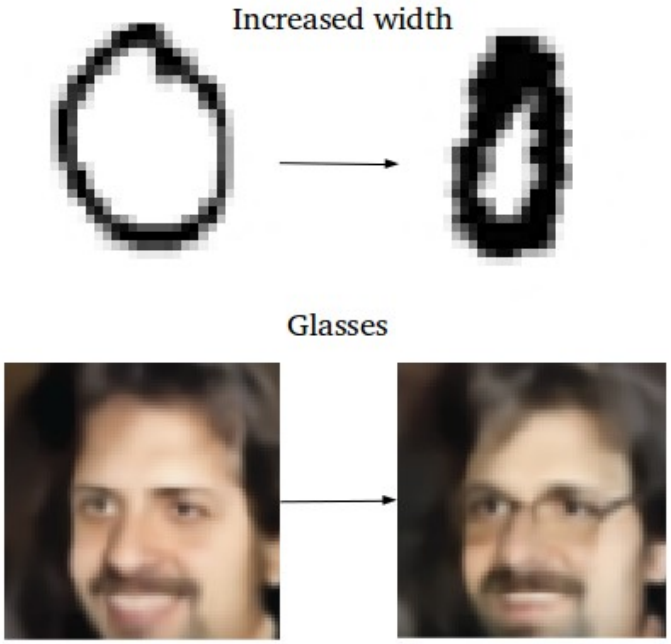
- **Problem z klasycznymi autoenkoderami polega na tym, że nie ma reguły co tak naprawdę przedstawia warstwa latent**
- **Odległości i zależności przestrzenne między aktywacjami dla danych przykładów często nie mają ze sobą związku – przestrzeń aktywacji warstwy latent nie jest ciągła**



Brak informacji o potencjalnej aktywacji między 7 a 1 [6]

Variational Autoencoders

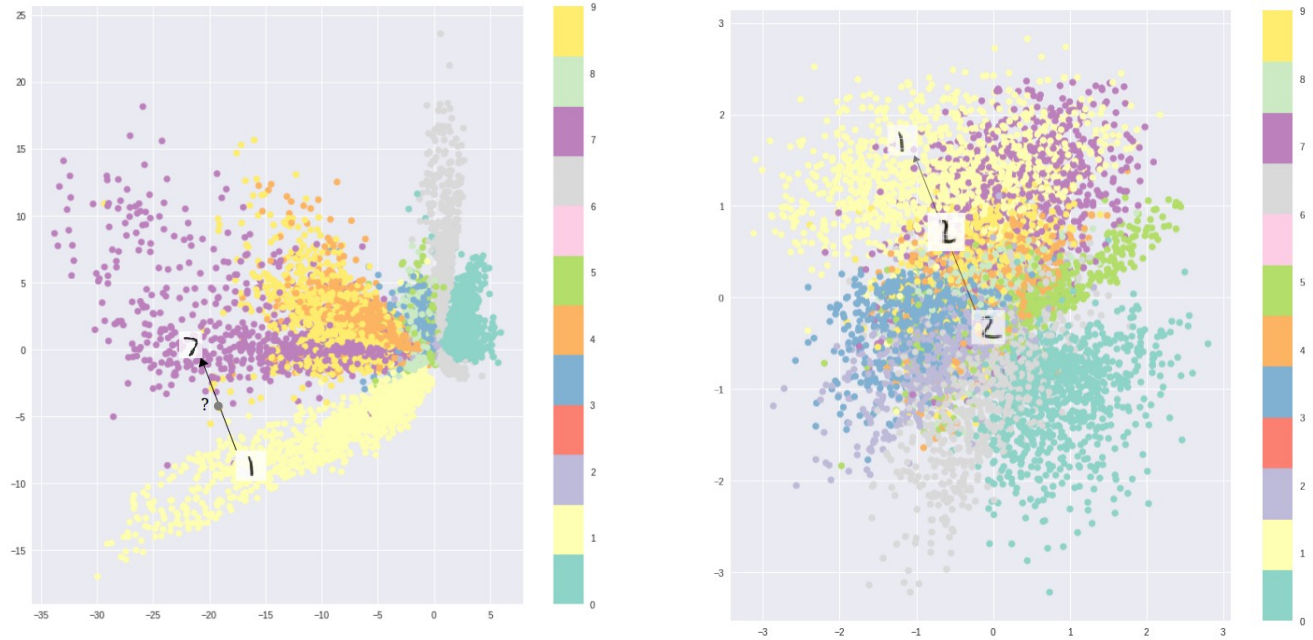
» **Możliwość generowania (zmieniania) nowych danych podobnych do tych które zostały użyte do uczenia**



Przykład wprowadzonej zmiany w przykładzie ze zbioru [6]

Variational Autoencoders

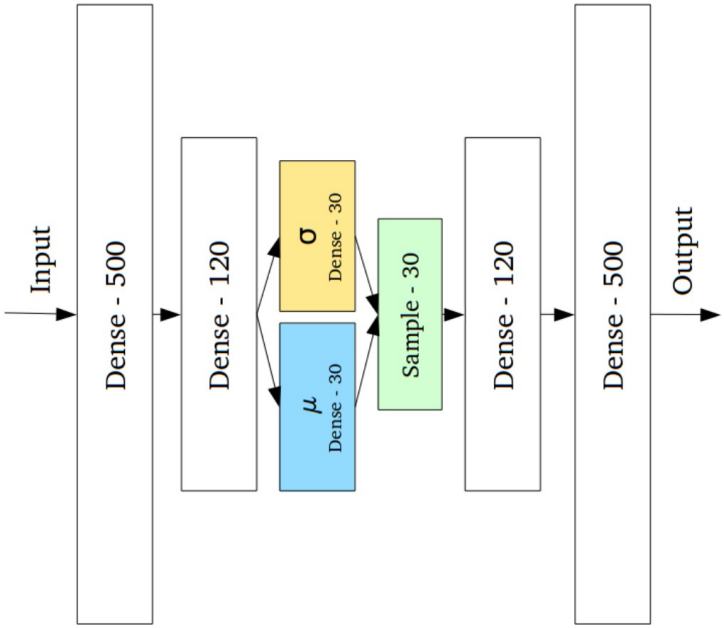
- **Założeniem autoenkoderów wariacyjnych jest ciągła przestrzeń aktywacji warstwy latent**
- **Spełnienie założenia umożliwia „zmuszenie” sieci do zakodowania przykładów w postaci rozkładu Gaussa**



Porównanie przestrzeni latent autoenkodera klasycznego i autoenkodera wariacyjnego[6]

Variational Autoencoders

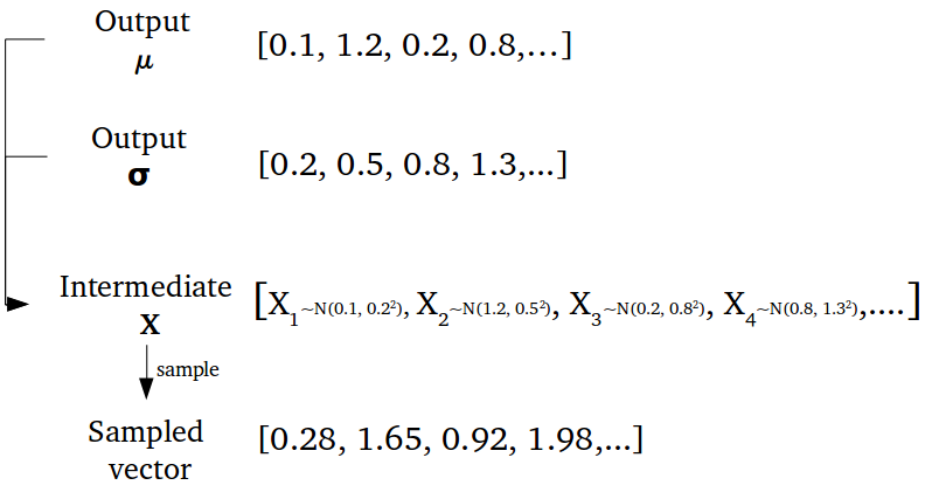
- **Założeniem autoenkoderów wariacyjnych jest ciągła przestrzeń aktywacji warstwy latent**
- **Spełnienie założenia umożliwia „zmuszenie” sieci do zakodowania przykładów w postaci rozkładu Gaussa**



Warstwa latent jest reprezentowana przez wektor średnich i odchyleń[6]

Variational Autoencoders

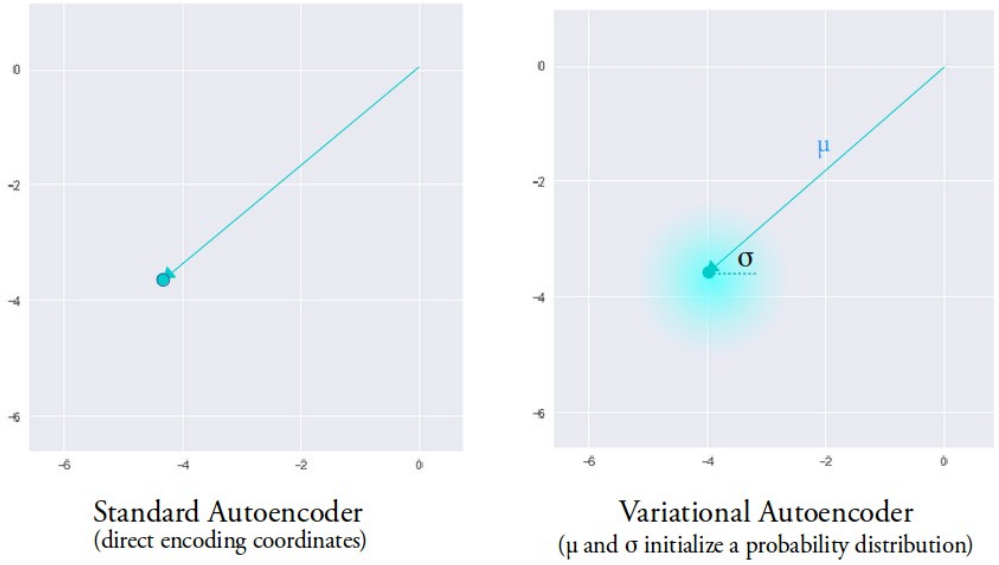
- **Aktywacja jest obliczana poprzez losowanie próbki (przejście stochastyczne)**
- **Przez co wartość aktywacji może być różna dla tego samego przykładu**



Przykład przejścia przez warstwę latent [6]

Variational Autoencoders

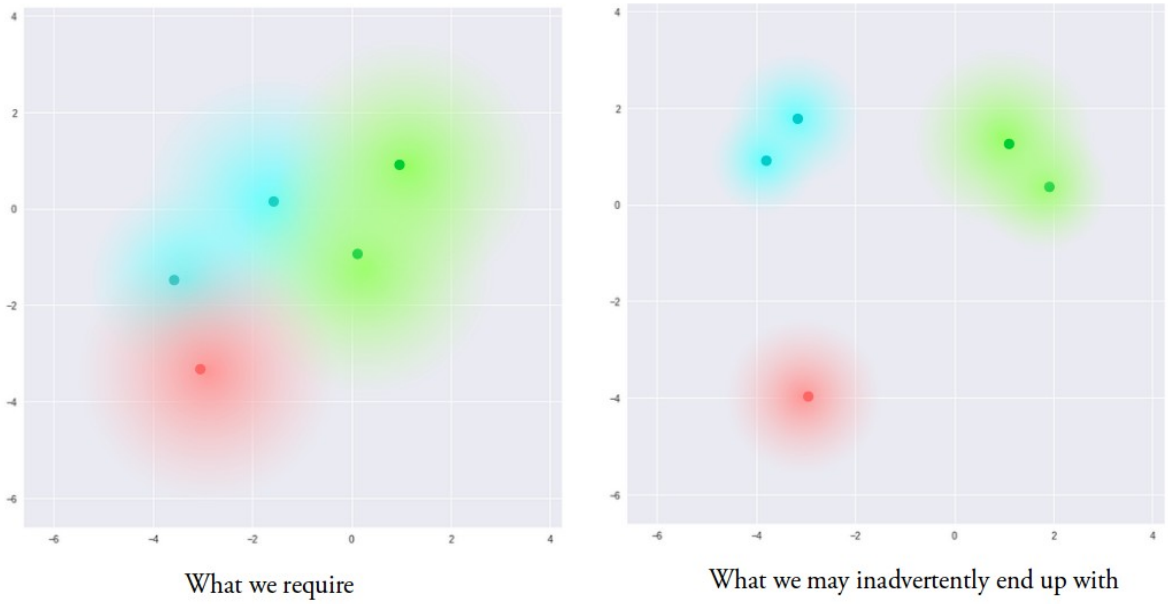
- **Średnia wskazuje gdzie zakodowany przykład ma być umiejscowiony w warstwie latent**
- **Odchylenie natomiast wskazuje obszar w jakim powinien znajdować się przykład z danego typu**



Dany przykład kodowany jest na otoczeniu [6]

Variational Autoencoders

» **Aby uniknąć sytuacji w której przejścia między kodowaniami nie są ciągłe, wprowadza się dywergencję KL do funkcji kosztu**



Po lewej stronie rezultat oczekiwany [6]

Variational Autoencoders

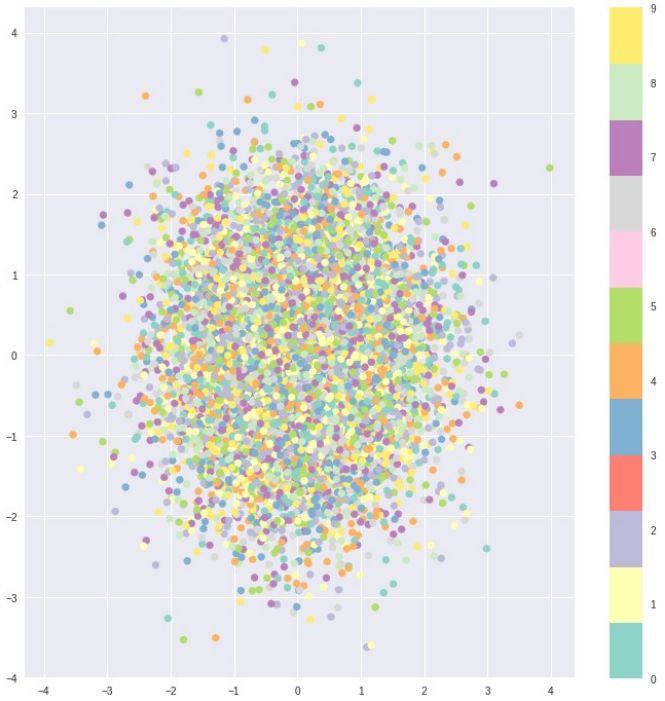
$$\sum_{i=1}^{n} \sigma_i^2 + \mu_i^2 - \log(\sigma_i) - 1$$

Dywergencja KL [6]

- **Można interpretować jako sumę wszystkich dywergencji KL między rozkładem przykładu wejściowego (X_i) a wartością rozkładu normalnego dla średniej i odchylenia**
- **Intuicyjnie, taka funkcja kosztu „zmusza” vae do rozprzestrzenienia wartości aktywacji równo wokół środka przestrzeni latent**

Variational Autoencoders

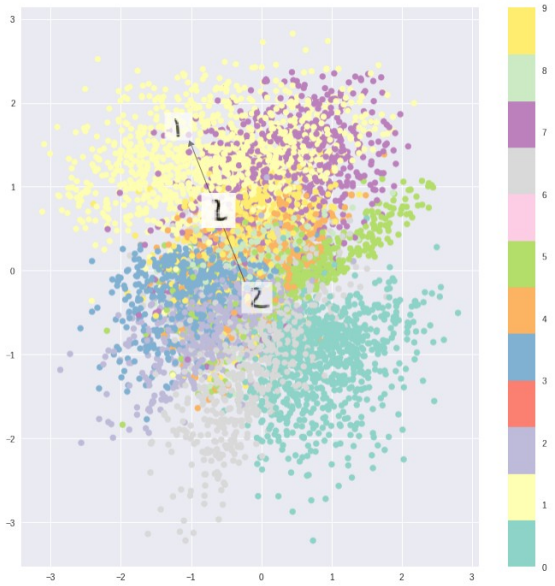
» **Zastosowanie tylko sumy dywergencji KL jako funkcji kosztu spowodowałoby jednak losowe skupienie wszystkich kodowań wokół środka**



Przykład latent na zbiorze MNIST w przypadku zastosowania samej dywergencji KL jako funkcji kosztu [6]

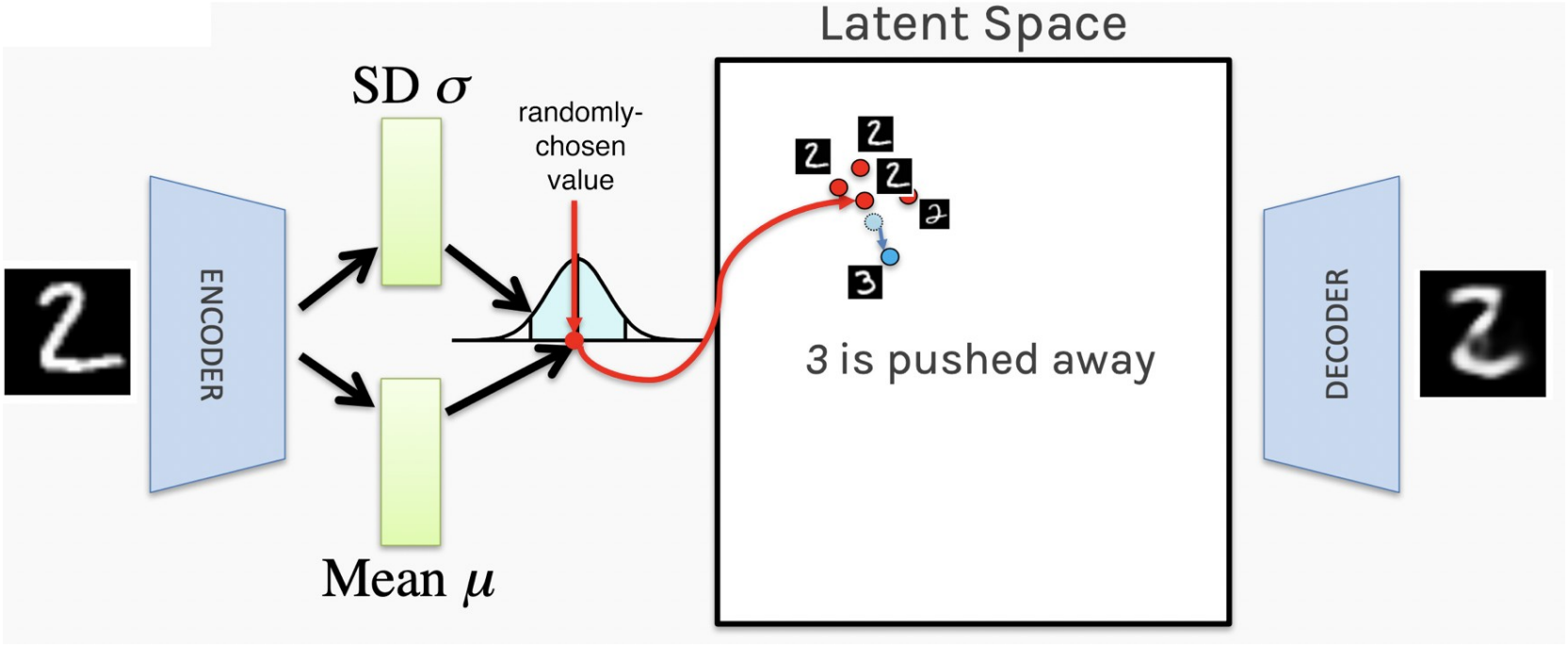
Variational Autoencoders

- **Dodanie błędu rekonstrukcji do funkcji kosztu powoduje klasteryzowanie kodowań**
- **Dywergencja KL odpowiada za skupienie wartości wokół centrum, błąd rekonstrukcji odpowiada za klasteryzację**



Aktywacje warstwy latent w przypadku zastosowania funkcji kosztu jako sumy dywergencji KL oraz błędu rekonstrukcji[6]

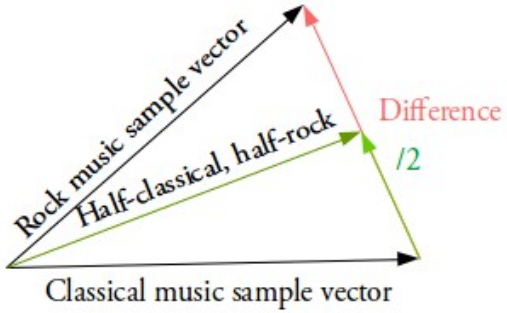
Variational Autoencoders



W procesie uczenia, elementy o podobnych cechach są skupiane blisko siebie[1]

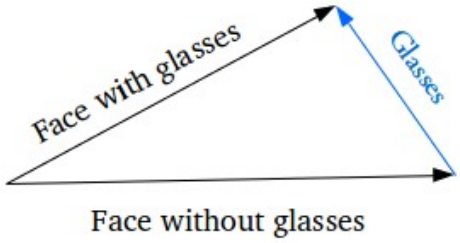
Variational Autoencoders

- **Generowanie przykładów poprzez wyznaczenie średniej z dwóch wektorów**



Jeśli dany przykład ma nosić cechy dwóch różnych przykładów, należy obliczyć średnią z wartości ich średnich [6]

- **Odejmując dwa wektory jesteśmy w stanie wygenerować ich różnicę**



Różnica dwóch kodowań pozwoli natomiast wygenerować różniące ich cechy [6]

Bibliografia

[1] https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368

[2] https://blog.keras.io/building-autoencoders-in-keras.html

[3] Hinton G. E., Salakhutdinov R. R, Reducing the Dimensionality of Data with Neural Networks, 2006

[4] Vincent et al., Extracting and Composing Robust Features with Denoising Autoencoders, 2008

[5] Makhzani A., Frey B., k-Sparse Autoencoders, 2013

[6] https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf