Uczenie nienadzorowane

Autoenkodery i RBM

Uczenie nienadzorowane

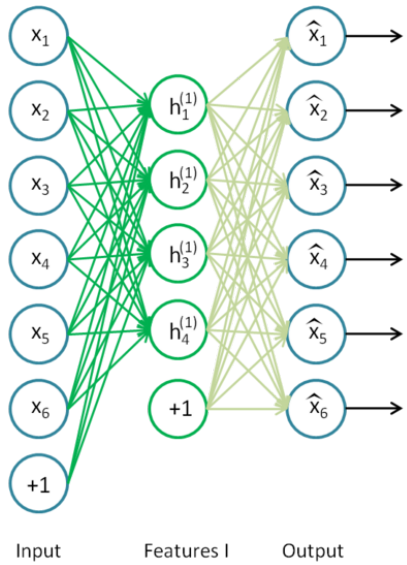
- Uczenie **nadzorowane** (*supervised*) - dane wejściowe $\mathbf{X}$ posiadają spodziewane wyjście (etykiety) $(\mathbf{X}, \mathbf{Y})$
- Uczenie **nienadzorowane** (*unsupervised*) używa wyłącznie danych wejściowych $\mathbf{X}$. Dane bez etykiet są powszechnie występujące.
  - trening automatycznie wykrywa istotne cechy w danych,
  - modeluje rozkład pr. danych
- Modele sieci:
  - autoenkodery (AE)
  - Restricted Boltzman Machines (RBM)
  - rzadkie kodowanie, sieci dekonwolucyjne
- Zastosowanie:
  - kodowanie sygnału, wykrywanie istotnych cech, usuwanie szumu i rekonstrukcja sygnału
  - inicjowanie głębokich sieci DNN
  - generowanie sygnałów

Autoenkodery

Encoder:

$$\mathbf{h}(\mathbf{x}) = f(\mathbf{x})$$

$$= \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$$



Decoder:

$$\hat{\mathbf{x}} = g(\mathbf{h})$$

$$= \sigma(\mathbf{W}'\mathbf{h}(\mathbf{x}) + \mathbf{b}')$$

- **Autoencoder** - sieć jednokierunkowa, której celem jest rekonstrukcja sygnału wejściowego
- warstwa ukryta $\mathbf{h}$: *code*, detektor cech, skompresowana reprezentacja danych
- często $\mathbf{W}' = \mathbf{W}^T$ współdzielone wagi

Źródło grafiki: `http://ufldl.stanford.edu`

Koszt rekonstrukcji

Funkcja rekonstrukcji $L(\mathbf{x}, \hat{\mathbf{x}})$

- MSE dla danych wejściowych rzeczywistych

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{2} \sum_k (\hat{x}_k - x_k)^2$$

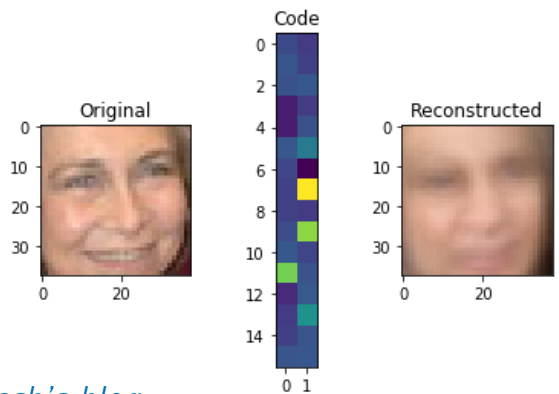
  - liniowa aktywacja warstwy wyjściowej $\hat{\mathbf{x}} = \mathbf{W}'\mathbf{h} + \mathbf{b}'$
- Cross Entropy dla danych wejściowych binarnych

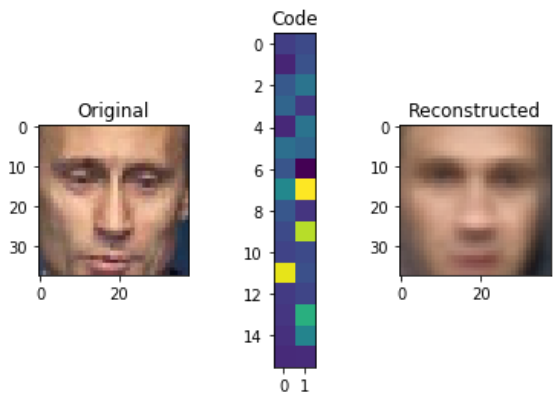
$$L(\mathbf{x}, \hat{\mathbf{x}}) = -\sum x_k \log \hat{x}_k + (1 - x_k) \log (1 - \hat{x}_k)$$

- trening wsteczną propagacją błędu

Undercomplete AE

- **Undercomplete AE** gdy warstwa **h** mniejsza od rozmiaru wejścia,
- kompresja sygnału do rozmiaru **h**
- nie generalizuje, jest w stanie kodować wyłącznie rozkład danych treningowych
- dla liniowego dekodera jest równoważny z PCA
- Przykład: input-output: 32x32 , hidden (code size): 32





src: ☞ *Manash's blog*

Regularyzowany AE

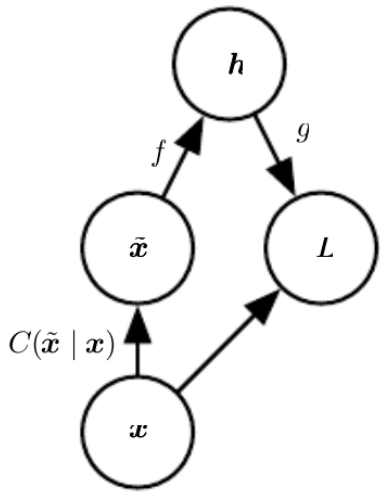
- wielkość **h** decyduje o pojemności ale za duża pojemność sprawia, że AE nie jest w stanie wyekstrahować wartościowej informacji o rozkładie sygnału wejściowego (wejścia mogą być kopiowane na wyjście)
- *Overcomplere* AE gdy rozmiar **h** większy od **x**,
  - brak kompresji
  - nie gwarantują uzyskania wartościowych reprezentacji bez dodania odpowiedniej regularyzacji wymuszającej odpowiednią reprezentację w **h** (np. rzadkość)
- **Sparse autoencoder** z karą za nie rzadką reprezentację

$$L(\mathbf{x}, \hat{\mathbf{x}}) + \lambda \sum |h_i|$$

  - można też użyć ReLU (Glorat, 2011)

Denoise autoencoder (DAE)

- **Denoise autoencoder (DAE)** usuwa szum (lub inne zniekształcenia) sygnału wejściowego.
- Sygnał wejściowy jest kopią **x** z dodanym szumem (lub innymi deformacjami) a AE uczy się odtwarzać oryginalny sygnał bez szumu



src: ☞ *Building Autoencoders in Keras*

Contractive AE

- **Contractive autoencoders (CAE)**

$$L(\mathbf{x}, \hat{\mathbf{x}}) + \lambda ||\nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x})||_F^2$$

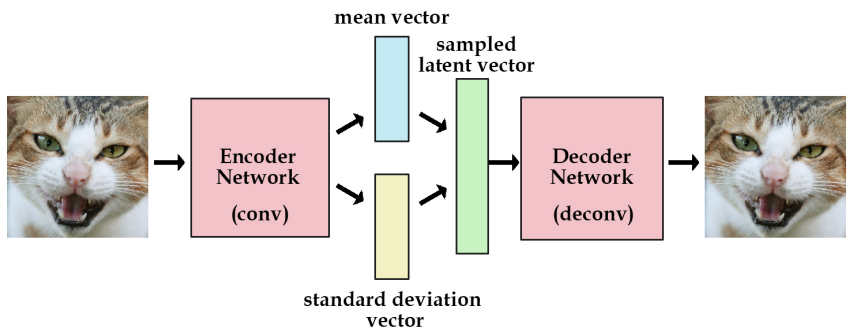
- kara za dużą zmienność reprezentacji przy zmianie sygnału (norma Frobeniusa)

$$||\nabla_{\mathbf{x}} \mathbf{h}(\mathbf{x})||_F^2 = \sum_i \sum_j \left(\frac{\partial h_i(x)}{\partial x_j} \right)^2$$

S. Rifai, et. al, (2011) Contractive Auto-Encoders:Explicit Invariance During Feature Extraction

Autoenkodery c.d.

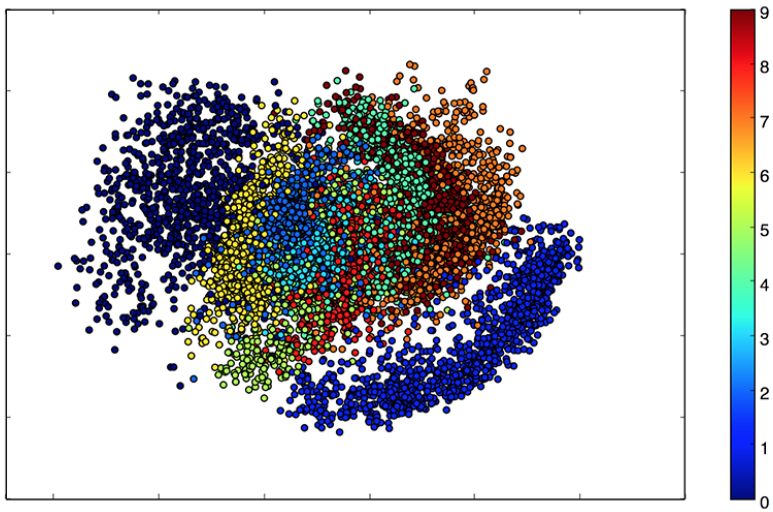
- **Variational autoencoders (VAE)** dodaje człon regularyzujący wymuszający odpowiedni rozkład sygnału w warstwie kodującej



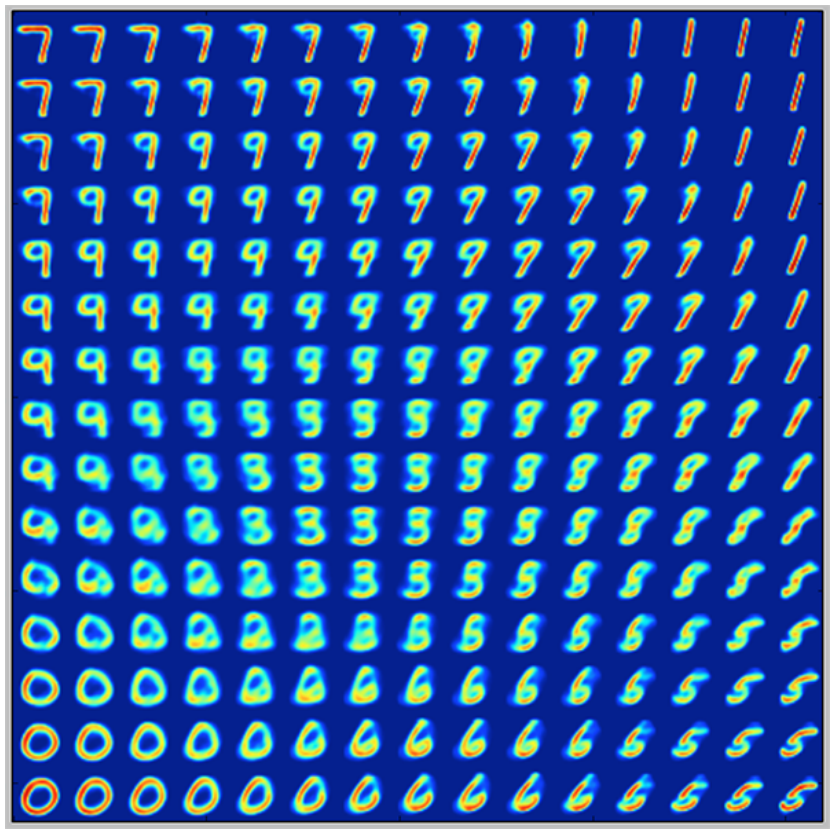
- Convolutional AE, dekoder jest symetryczną do enkodera siecią dekonwolucyną
- sequence-to-sequence AU (z użyciem np. LSTM)

VAE na MNIST

wizualizacja danych treningowych



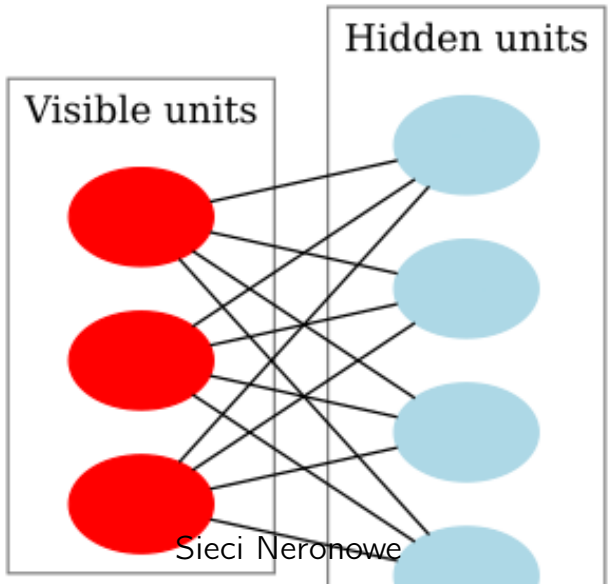
generowanie cyfr



src: ☞ *Building Autoencoders in Keras*

Restricted Boltzman Machine

- **Restricted Boltzman Machines (RBM)** (Hinton 2000), wcześniej Harmonium (P. Smolensky, 1986), to szczególny przypadek Maszyny Boltzmana
- stochastyczna sieć neuronowa modelująca rozkład prawdopodobienstw wejść
- *restrictive* - brak połączeń wewnątrz warstwy
- algorytm uczenia *contrastive dicergence* (Hinton)



Źródło grafiki: Wikipedia

Budowa RBM

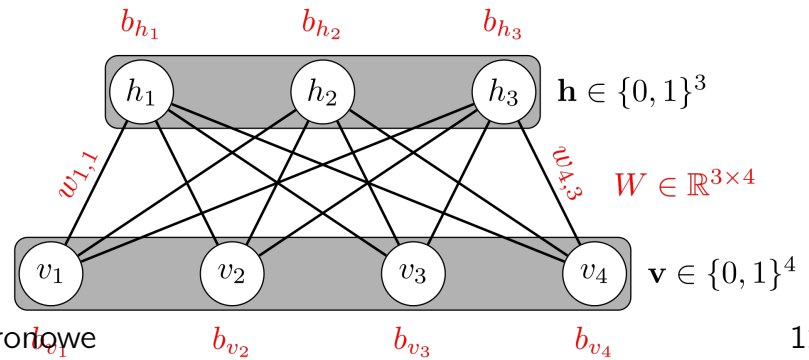
- binarne jednostki umieszczone w dwóch warstwach: widzialnej **v** i ukrytej **h**
- sygnał wejściowy podawany do jednostki widzialnej $\mathbf{v} = \mathbf{x}$
- energia układu

$$E(\mathbf{x}, \mathbf{h}) = -\sum a_i x_i - \sum b_i h_i - \sum\sum x_i w_{ij} h_j$$

gdzie w_{ij} to wagi połączeń między warstwami, a_i i b_i to wyrazy wolne związane z warstwą widzialną i ukrytą

- prawdopodobieństwo rozkładu

$$p(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{x},\mathbf{h})}$$



src: https://kwonkyo.wordpress.com/

RBM

- prawdopodobieństwo marginalne

$$P(\mathbf{x}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{x},\mathbf{h})}$$

- brak połaczeń wewnątrz warstw, więc stany w wartstwach są niezależne

$$P(\mathbf{h}|\mathbf{x}) = \prod_{j=1}^{n} P(h_j|\mathbf{x}), \qquad P(\mathbf{x}|\mathbf{h}) = \prod_{i=1}^{m} P(v_i|\mathbf{h})$$

$$P(h_i = 1|\mathbf{x}) = \sigma(\mathbf{w}_i\mathbf{x} + b), \qquad P(x_k = 1|\mathbf{h}) = \sigma(\mathbf{h}^T\mathbf{w}_k + c)$$

Contrastive divergence CD

- maksymalizacja prawdopodobieństwa $P(\mathbf{x})$

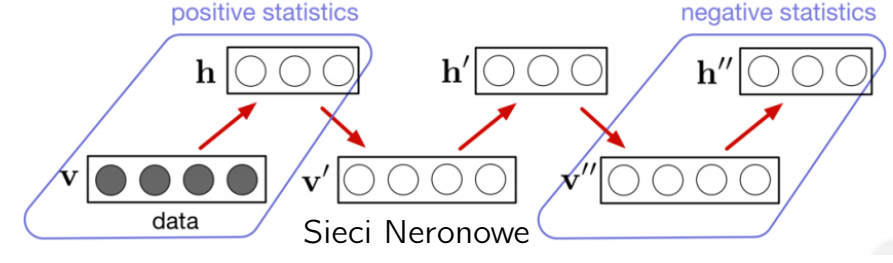
$$L = \frac{1}{N} \sum \log P(\mathbf{x})$$

- *Contrastive divergence* (Hinton) mtoda przybliżona kożystająca z samplowania Gibbsa
  - dla każdego $\mathbf{x}$ wejściowego wygeneruj $\mathbf{x}'$ z k krotnego samplowania Gibbsa
  - aktualizacja

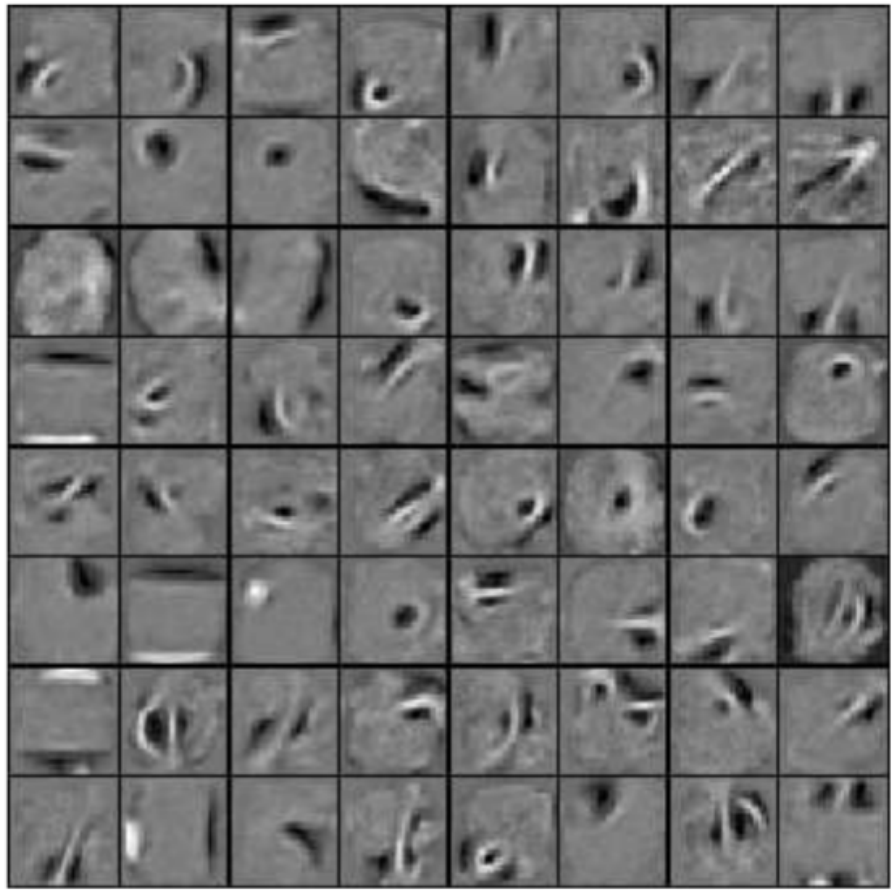
$$\Delta \mathbf{W} = \epsilon(\mathbf{x}\mathbf{h}^T - \mathbf{x}'\mathbf{h}'^T)$$

$$\Delta \mathbf{a} = \epsilon(\mathbf{x} - \mathbf{x}'), \qquad \Delta \mathbf{b} = \epsilon(\mathbf{h} - \mathbf{h}')$$

- w praktyce $k = 1$ daje dobre cechy

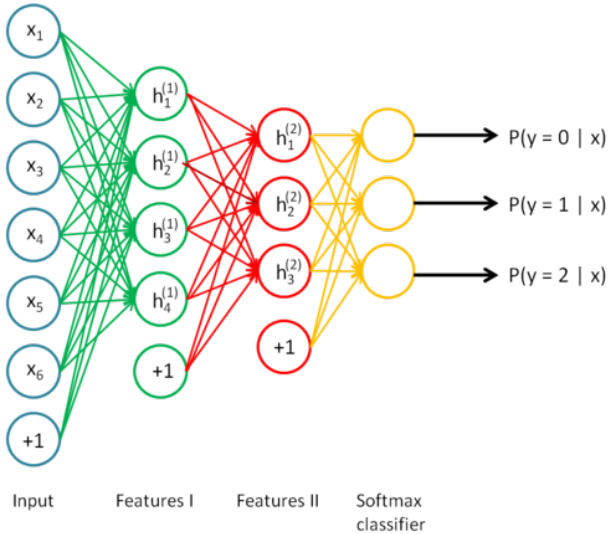


Filtry RBM na MNIST



Erhan, Bengio, (2010) Why Does Unsupervised Pre-training Help Deep Learning?

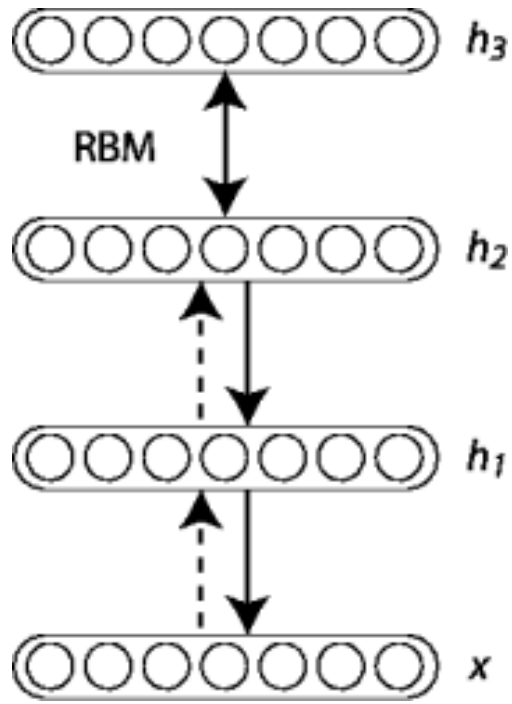
Stacked autoencoders



- Sygnał na wyjściu warstwy ukrytej jest wejściem kolejnego autoenkodera
- Hierarchia reprezentacji: pierwsza warstwa koduje najprostsze warstwy, kolejne bardziej ogólne relacje, itd.
- Podobna architektura: wielowarstwowy RBM to Deep Belief Networks

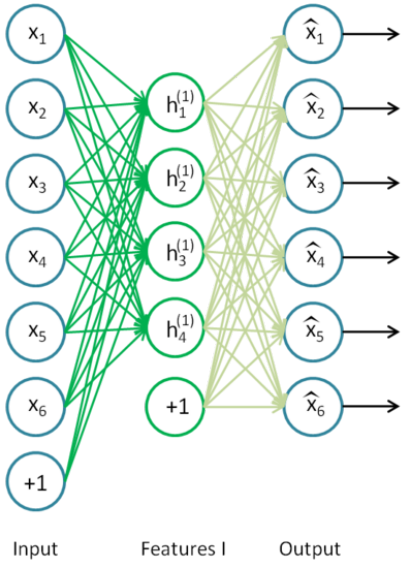
Deep Belief Networks (DBN)

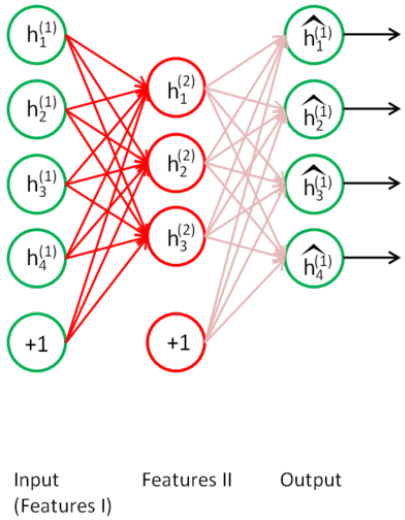
- DBN (Teh, Hinton 2006) wielowarstwowy RBM,
- uczy się głębokiej, hierarchicznej reprezentacji danych, każda kolejna warstwa analizuje sygnał widoczny w warstwie ukrytej poprzedniego RBM
- uczenie zachłanne: kolejne warstwy dodawane i uczone pojedynczo
- pierwszy efektywny model głęboki.
- to nie jest sieć jednokierunkowa

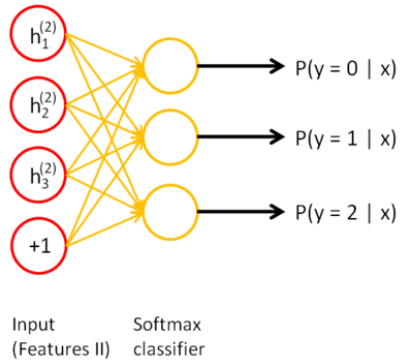


src: *http://deeplearning.net/tutorial/DBN.html*

Inicjalizacja sieci DNN



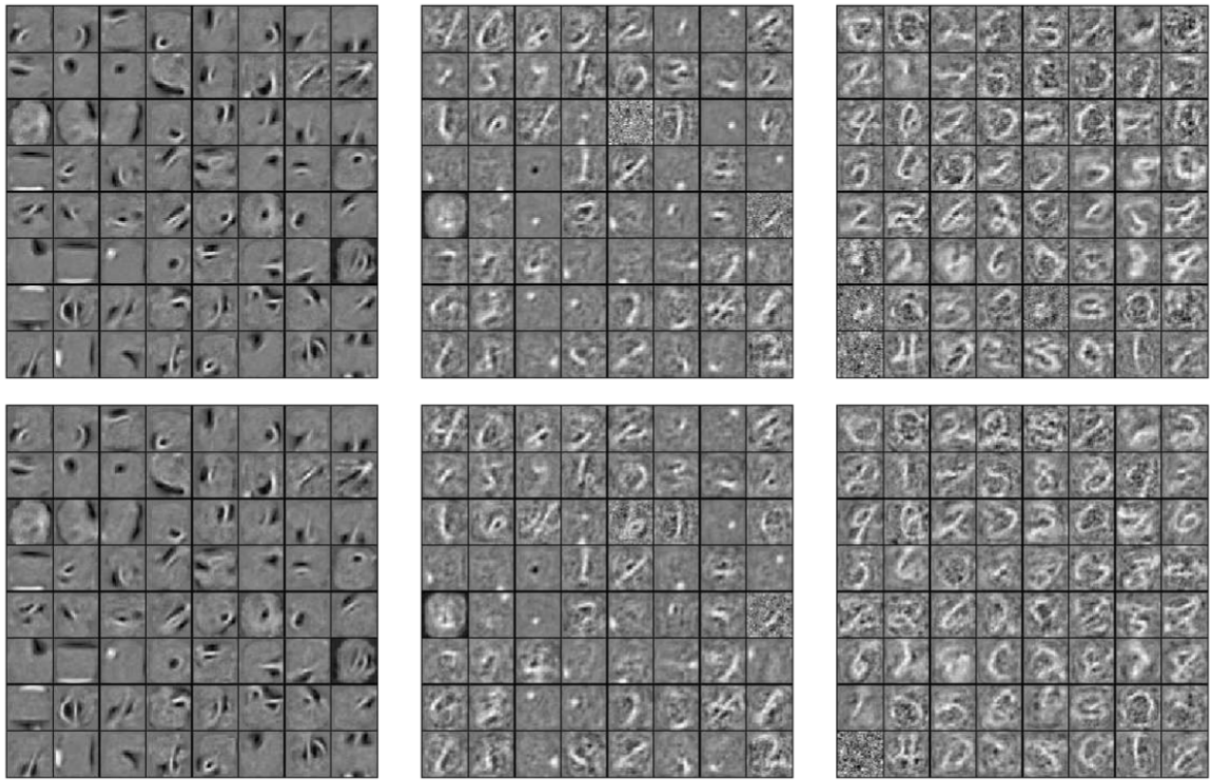




1. *Greedy layer-wise pre-training*:
   - trening nienadzorowany autoenkodera wsteczną propagacją
   - usuwamy warstwę wyjściową (dekoder) i uczymy kolejny AE, którego wejściem jest sygnał z warstwy ukrytej poprzedniego AE przy niezmiennych wartościach wag w poprzednich warstwach
   - powtarzamy procedurę dla każdej kolejnej warstwy
2. *fine tuning*: dodajemy w pełni połączoną warstwę wyjściową (lub kilka warstw) i uczymy całą sieć w sposób nadzorowany

Filtry DBN na MNIST

pre-training



fine tune

Erhan, Bengio, (2010) Why Does Unsupervised Pre-training Help Deep Learning?

Wyniki MNIST, Bengio 2007

	train.	valid.	test
DBN, unsupervised pre-training	0%	1.3%	1.4%
Deep net, auto-associator pre-training	0%	1.4%	1.4%
Deep net, supervised pre-training	0%	1.75%	2.0%
Deep net, no pre-training	.004%	2.1%	2.4%
Shallow net, no pre-training	.004%	1.8%	1.9%

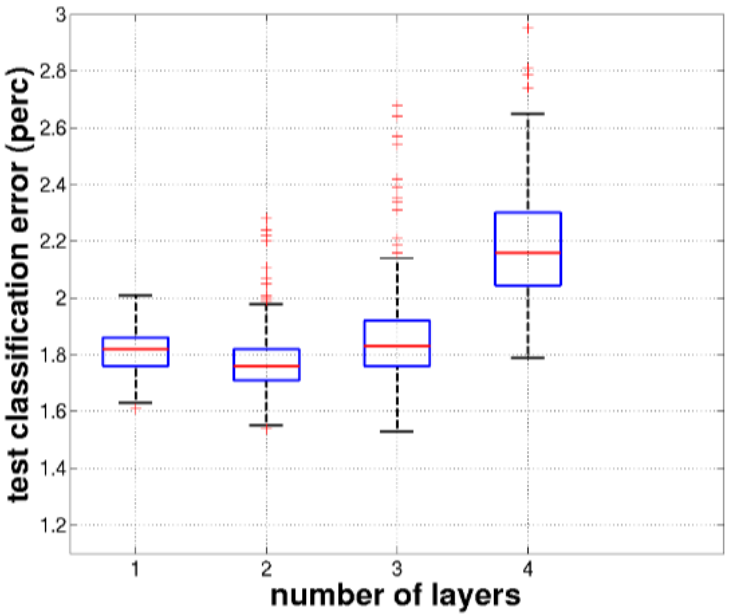
Table 2: *Classification error on MNIST training, validation, and test sets, with the best hyper-parameters according to validation error, with and without pre-training, using purely supervised or purely unsupervised pre-training.*

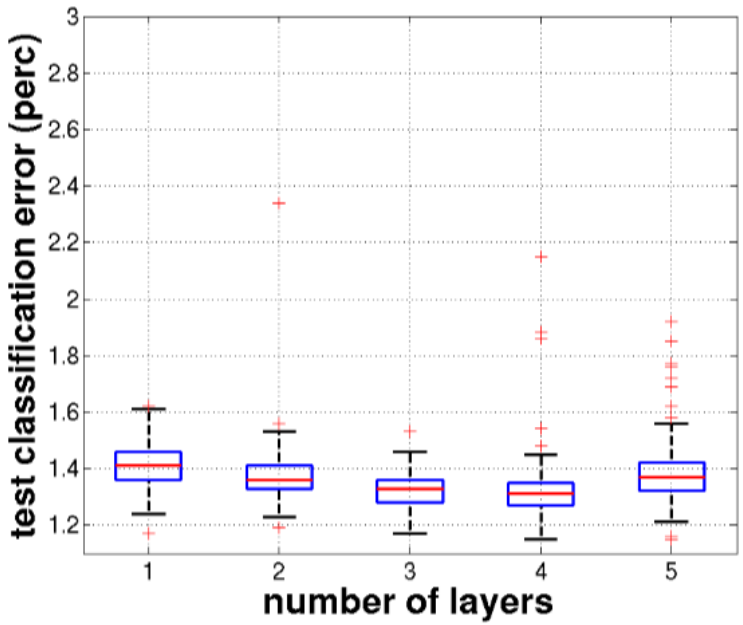
- architektura: 784 wejść, 10 wyjść, 3 warstwy ukryte
- na MNIST 0.1% statystycznie istotna różnica

Bengio, et al., Greedy layer-wise training of deep networks, 2007.

Pre-trening DNN

Pre-trening poprawia generalizację, działa jak regularyzacja

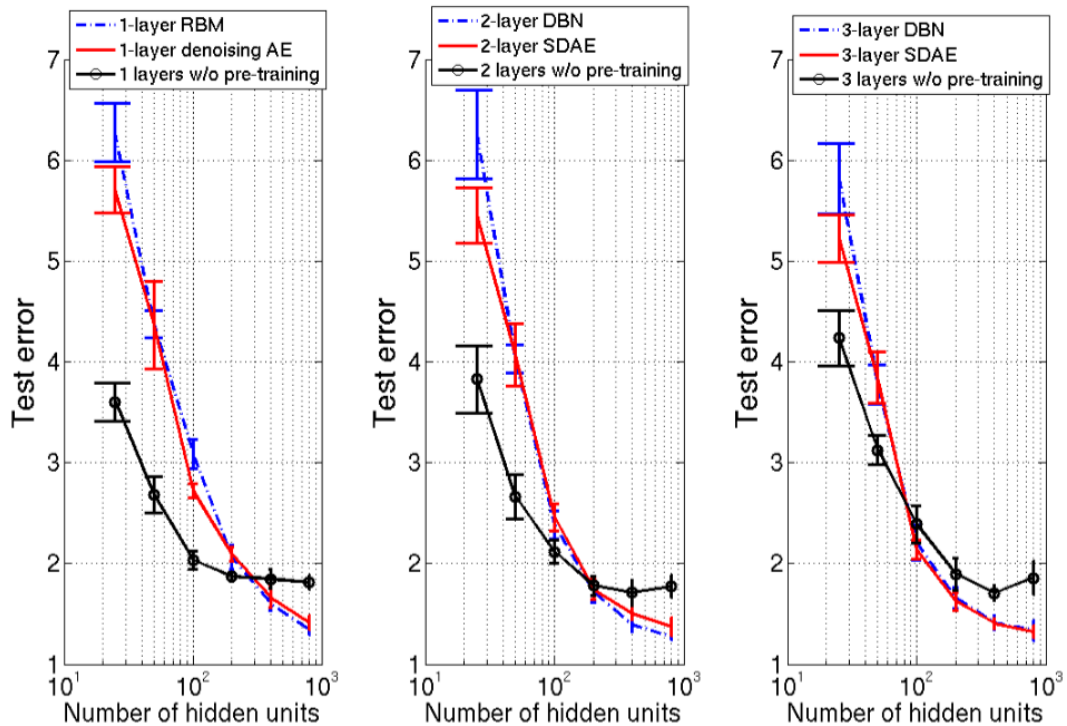




Erhan, Bengio, (2010) Why Does Unsupervised Pre-training Help Deep Learning?

Pre-trening DNN

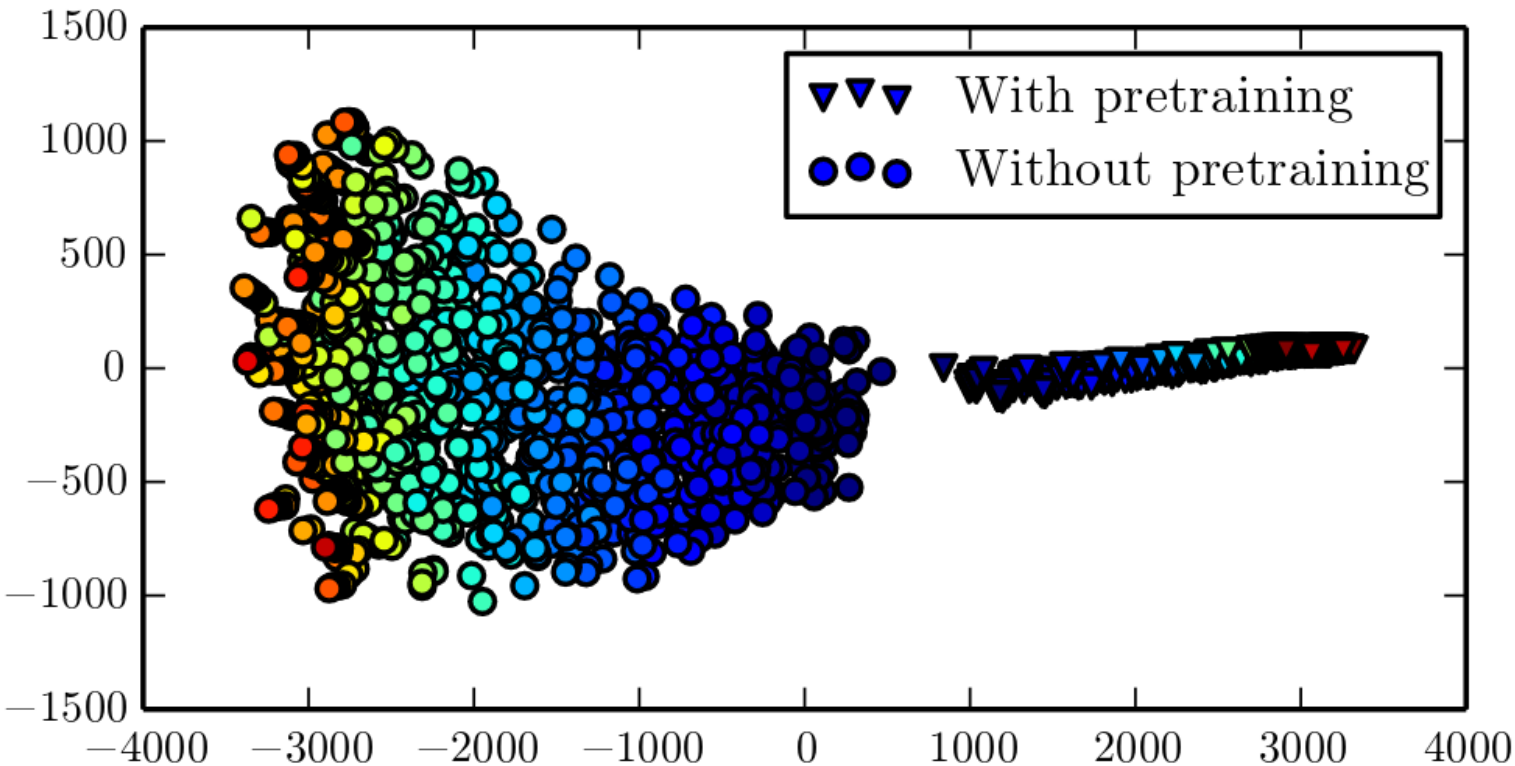
Wartość pre-treningu wzrasta z liczbą wartsw



Erhan, Bengio, (2010) Why Does Unsupervised Pre-training Help Deep Learning?

Pre-training DNN

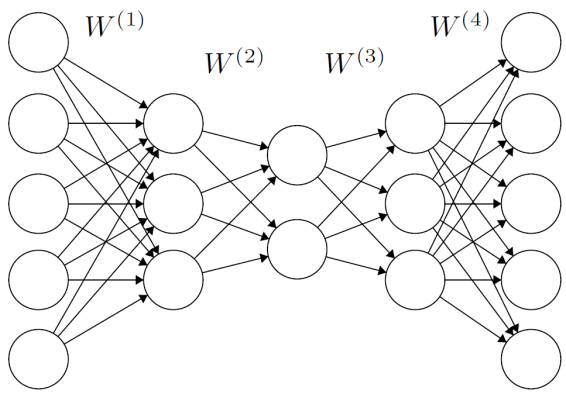
Pre-training ma wpływ na trajektorię uczenia



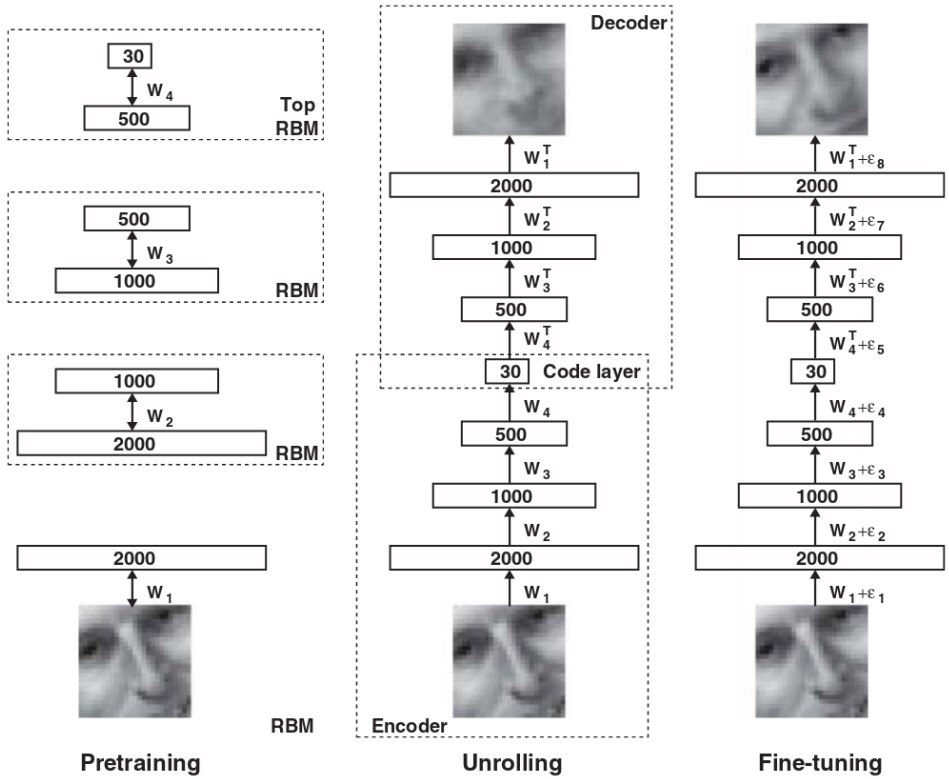
Goodfelow, 2016

Głębokie autoenkodery

- dodatkowa warstwa ukryta zwiększa możliwości enkodera w odwzorowaniu kodu (uniwersalny aproksymator jest w stanie nauczyć się dowolnego kodowania)
- dodanie warstw pozwala zmniejszyć złożoność reprezentacji trudnych problemów
- głębokie AE pozwalają uzyskać większą kompresję (Hinton 2006)
- niekompletne głębokie AE - żadna warstwa ukryta nie powinna być mniejsza niż warstwa kodująca



Pretraining dla głebokich AE



- RBM używany do inicjacji kodera i dekodera

G. Hinton, R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science 2006

Komresja obrazów: Deep AE vs. PCA

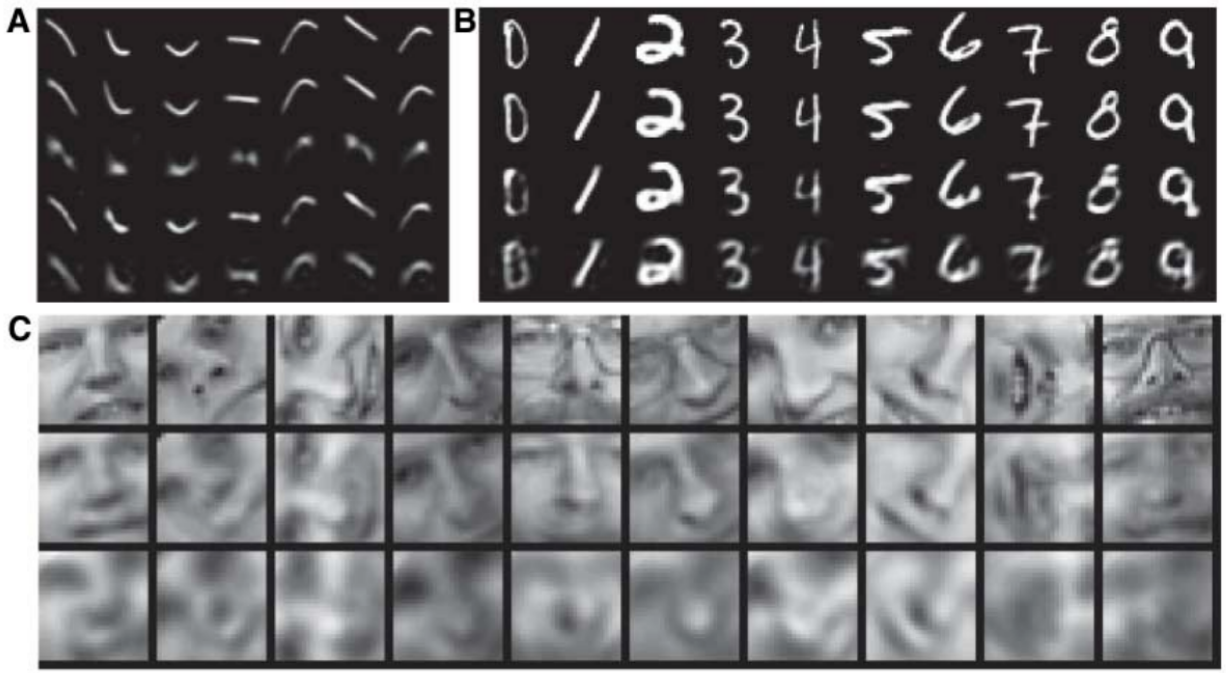
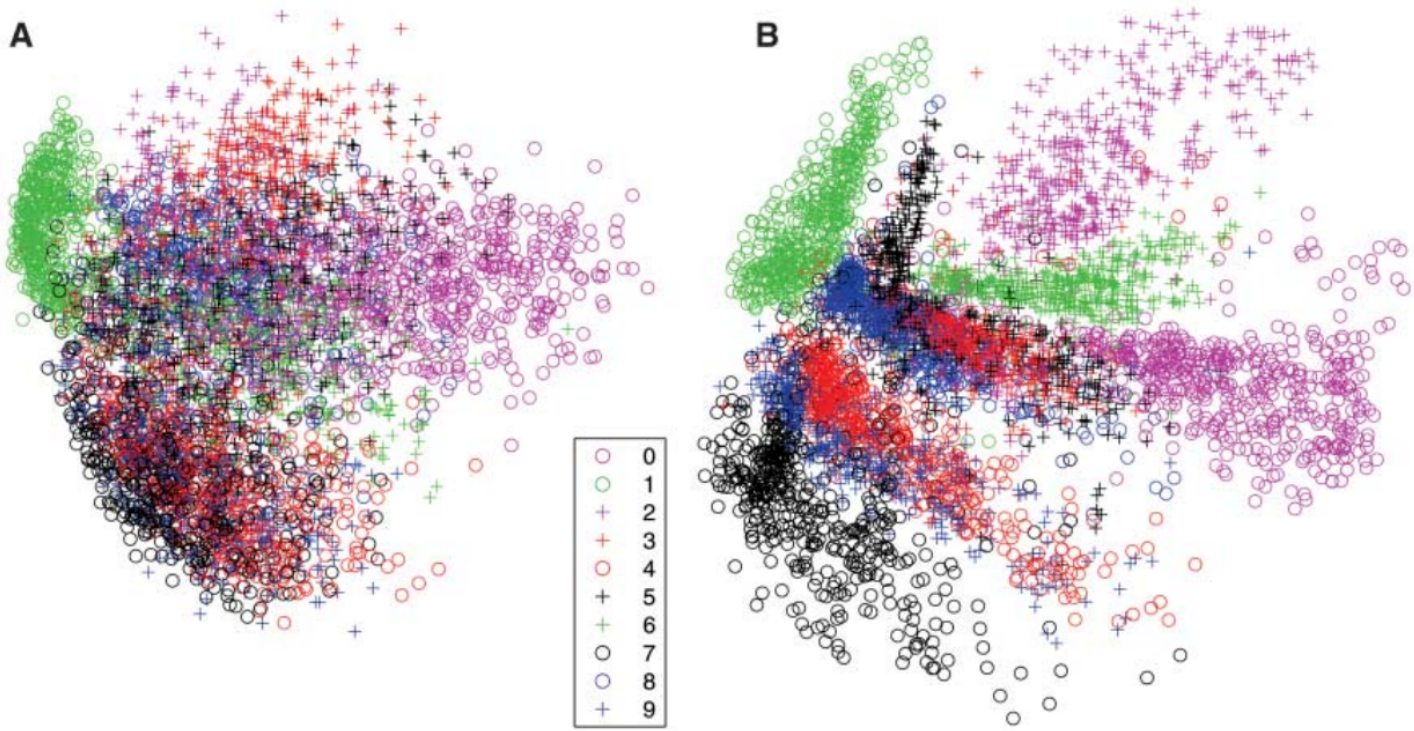


Fig. 2. (**A**) Top to bottom: Random samples of curves from the test data set; reconstructions produced by the six-dimensional deep autoencoder; reconstructions by "logistic PCA" (*8*) using six components; reconstructions by logistic PCA and standard PCA using 18 components. The average squared error per image for the last four rows is 1.44, 7.64, 2.45, 5.90. (**B**) Top to bottom: A random test image from each class; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional logistic PCA and standard PCA. The average squared errors for the last three rows are 3.00, 8.01, and 13.87. (**C**) Top to bottom: Random samples from the test data set; reconstructions by the 30-dimensional autoencoder; reconstructions by 30-dimensional PCA. The average squared errors are 126 and 135.

G. Hinton, R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science 2006

Projekcja do 2D (dwa neurony kodujące)

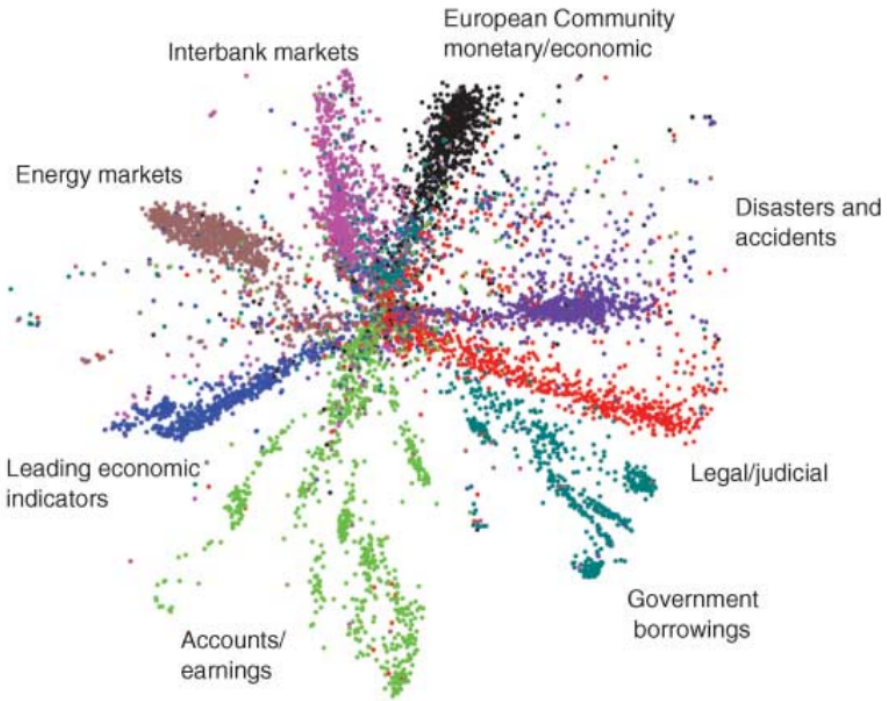
MNIST: PCA vs. Deep AE (784-1000-500-250-2)



G. Hinton, R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science 2006

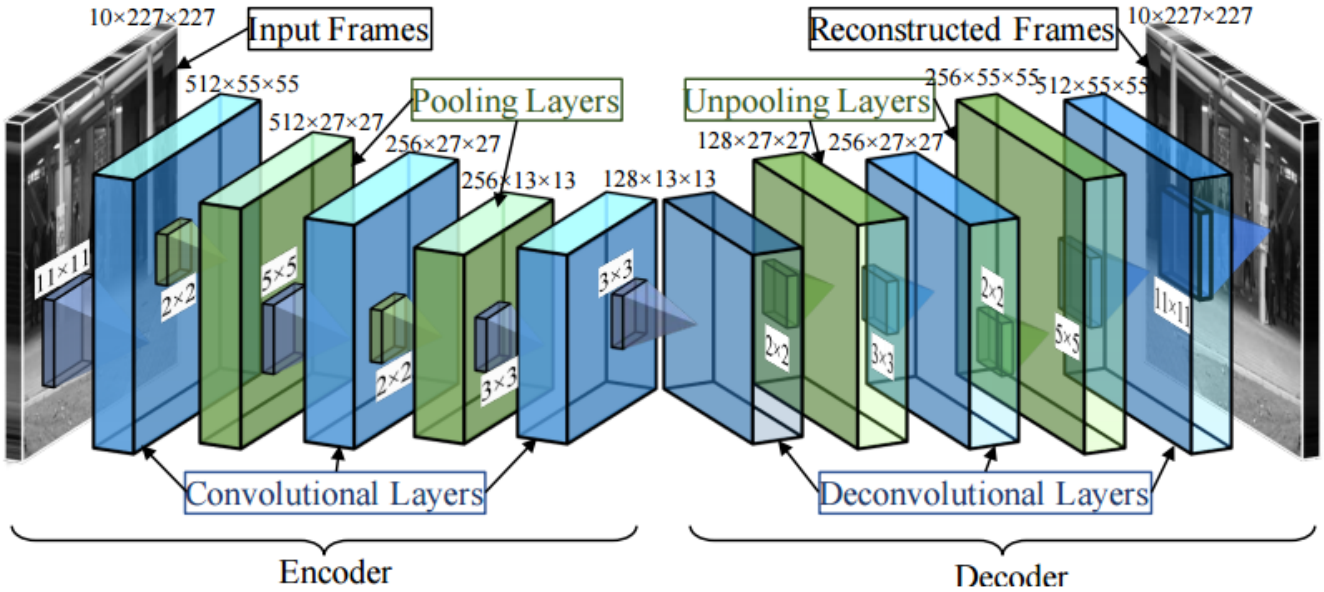
Projekcja do 2D (dwa neurony kodujące)

Wizualizacja dokumentów: architektura 2000-500-250-125-2



G. Hinton, R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science 2006

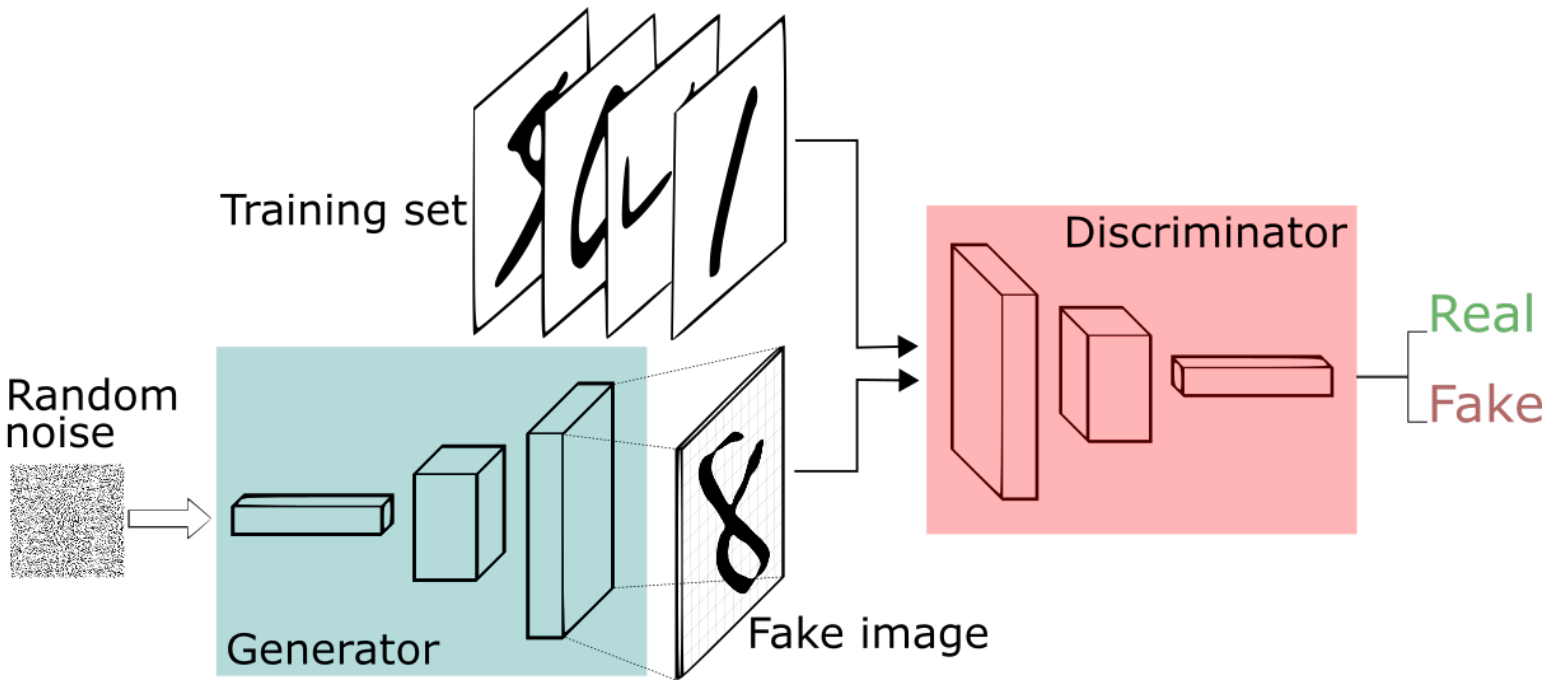
Splotowe autoenkodery



Generative Adversal Network (GAN)

- GAN (Goodfelow 2014) - dwie sieci (generująca i oceniająca) rywalizujące ze sobą w grze o sumie zerowej
- zastosowanie: generowanie zdjęć o realistycznych cechach
- sieć oceniająca (dyskryminująca, np. CNN) stara się odróżnić prawdziwy sygnał od wygenerowanego przez siec generująca
- sieć generująca (np. sieć dekonwolucyjna) tworzy sygnał z pewnego rozkładu starając się „oszukać" sieć oceniającą, dąży do maksymalizacji błędu dyskryminacji
- obie sieci uczone wsteczną propagacją, sieć generująca twory coraz bardziej realistyczne obrazy, sieć oceniająca specjalizuje się w rozróżnianiu coraz subtelniejszych różnic pomiędzy obrazami prawdziwymi i wygenerowanymi

GAN



src: https://deeplearning4j.org/generative-adversarial-network

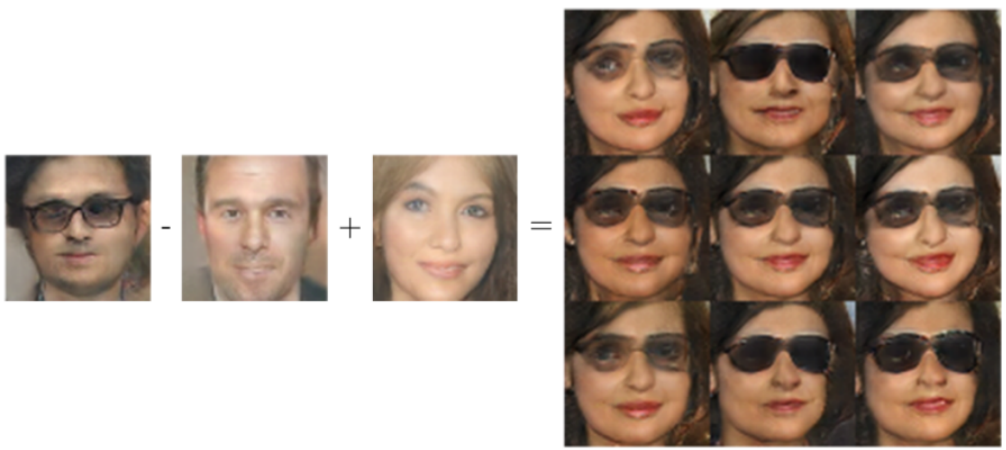
Przykłady

DCGAN (Denton et al., 2015) generowanie obrazów wysokiej rozdzielczości

Goodfwlow, Deep Learning, 2016

Ukryta reprezentacja pojęć

- utworzona ukryta reprezentacja koduje pojęcia
- arytmetyczne operacja na wektorach kodujących odpowiadają relacjom semantycznym pomiędzy pojęciami



Radford et al., 2015

☞ AI i uczenie maszynowe przewidują wyniki MS 2018