Uczenie nienadzorowane

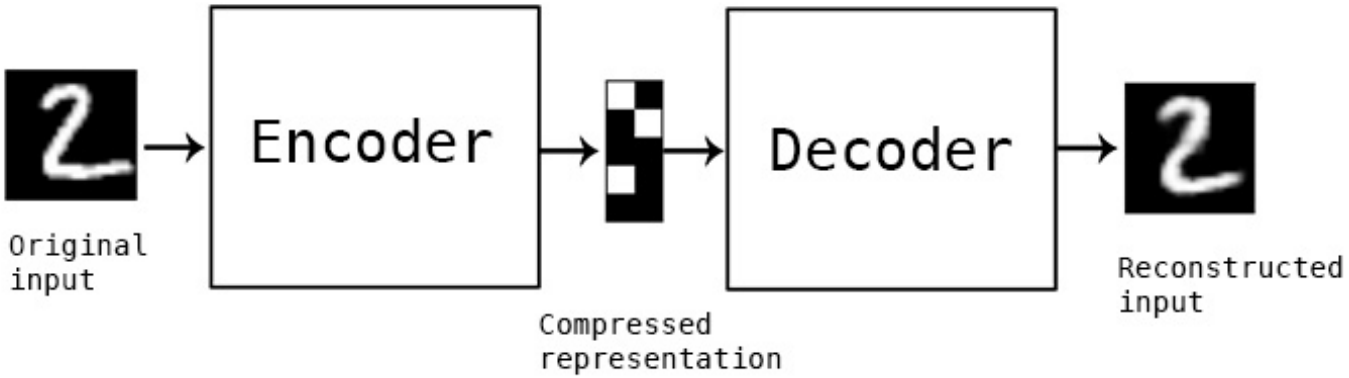
Autoenkodery

Modele generatywne

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 prawdopodobieństwa danych
- Modele sieci:
  - autoenkodery (AE)
  - Restricted Boltzman Machines (RBM)
  - GAN
- Zastosowanie:
  - kodowanie sygnału, wykrywanie istotnych cech, usuwanie szumu i rekonstrukcja sygnału
  - generowanie sygnałów
  - inicjowanie głębokich sieci DNN (stacked autoencoders)

Autoenkodery



- autoenkodery uczą się odtwarzać sygnał wejściowy
- warstwy pośrednie kodują specyficzne cechy danych
- kompresja stratna - wymagamy aby model nie był jednostkowym odwzorowaniem
- AE uczą się reprezentacji danych (ekstrakcja cech), modelują rozkład prawdopodobieństwa sygnału
- koder i dekoder mogą być sieciami neuronowymi, całość uczona wsteczną propagacją błędu

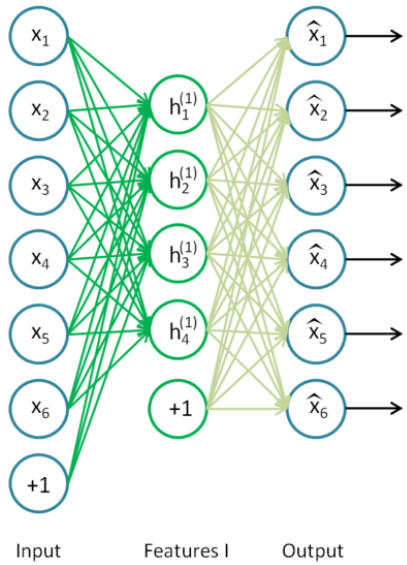
Rys: ☞ *Building Autoencoders in Keras*

Autoenkodery

Encoder:

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

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



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 $\mathbf{h}$: koduje sygnał wejściowy, detektor cech, skompresowana reprezentacja danych
- często wymaga się aby $\mathbf{W}' = \mathbf{W}^T$, współdzielone wagi kodera i dekodera

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 $\hat{\mathbf{x}} = \mathbf{W}'\mathbf{h} + \mathbf{b}'$

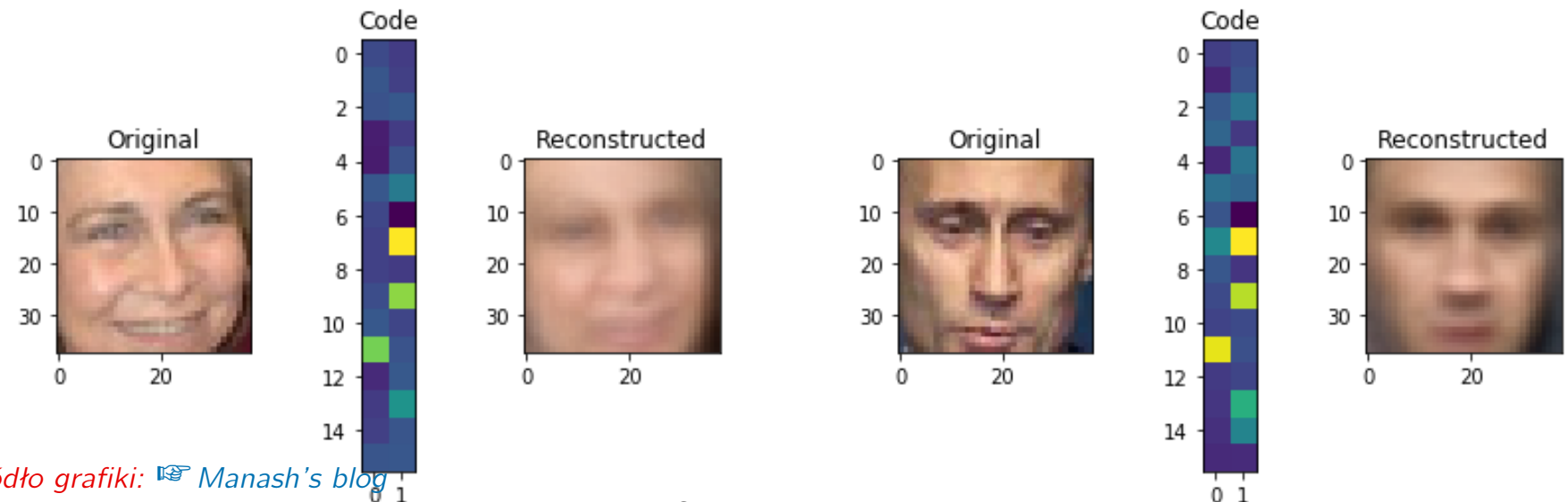
- Cross Entropy dla danych wejściowych binarnych

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

- trening wsteczną propagacją błędu
- ograniczenia zapobiegające tworzeniu odwzorowania jednostkowego: niekompletny AE, regularyzowany AE, kurczliwy AE, ...

Niekompletny autokoder

- **Niekompletny AE** (*undercomplete*) - rozmiar warstwy **h** mniejszy od rozmiaru wejścia,
- kompresja sygnału do rozmiaru **h**
- dla liniowego dekodera wynik jest równoważny z PCA, wagi neuronów tworzą składowe główne maksymalizujące wariancję
- Przykład: input-output: 32x32 (1024 piksele) , hidden (code size): 32



Źródło grafiki: ☞ *Manash's blog*

Regularyzowany AE

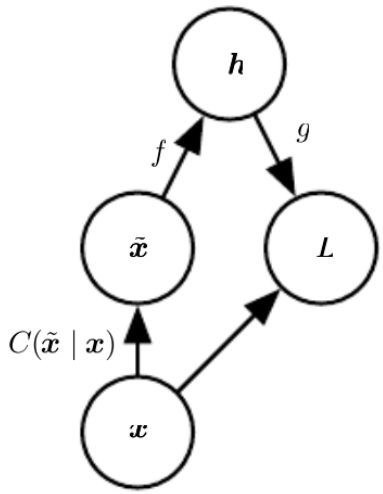
- wielkość **h** decyduje o pojemności sieci i jakości odwzorowania. Dla niekompletnych AE jest ograniczenie zwiększania rozmiaru warstwy ukrytej
- **Nadkompletny** AE (*overcomplete*) gdy rozmiar **h** większy od rozmiaru sygnału wejsciowego **x**,
  - brak kompresji
  - nie gwarantują uzyskania wartościowych reprezentacji bez dodania odpowiedniej regularyzacji wymuszającej odpowiednią reprezentację w **h** (np. rzadkość)
- **Rzadki autoencoder** z czynnikiem kary za brak rzadkiej reprezentacji

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

  - w celu uzyskania rzadkich aktywacji można też użyć ReLU (Glorat, 2011)

Autoenkoder z odszumianiem (DAE)

- **Denoise autoencoder (DAE)** usuwa szum (lub inne zniekształcenia) sygnału wejściowego.
- Sygnał wejściowy $\tilde{\mathbf{x}}$ jest zniekształconą kopią $\mathbf{x}$
- AE uczy się odtwarzać oryginalny sygnał bez deformacji



src: ☞ *Building Autoencoders in Keras*

Kurczliwy AE

- **Contractive autoencoder (CAE)**

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

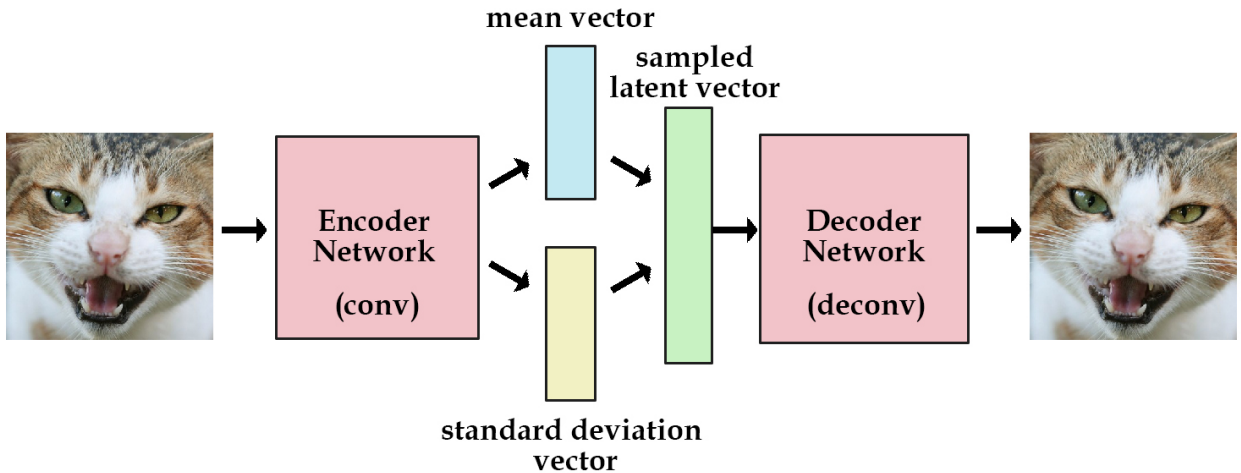
- kara za dużą zmienność reprezentacji przy zmianie sygnału (norma Frobeniusa)

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

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

Wariacyjny AE

- **Variational autoencoders (VAE)** jawnie modeluje rozkład gęstości danych, kodowana reprezentacja to parametry funkcji gęstości, np. σ_i, μ_i rozkładu normalnego $N(\mu_i, \sigma_i)$



- sygnał wejściowy dekodera jest próbkowany z rozkładu określonego przez parametry warstwy kodującej
- model generatywny - umożliwia próbkowanie danych wyjściowych, tworzenie nowych sygnałów o charakterystyce zbliżonej do danych treningowych

Kullback–Leibler divergence

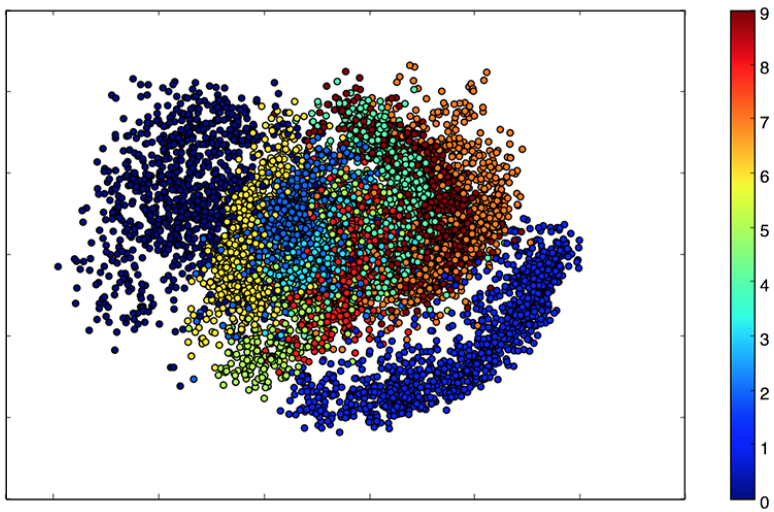
- czynnik regularyzujący, dodawany do funkcji kosztu (np. MSE), wymuszający odpowiedni rozkład danych w warstwie kodującej (KL divergence), np. preferujący rozkład $N(0, 1)$

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

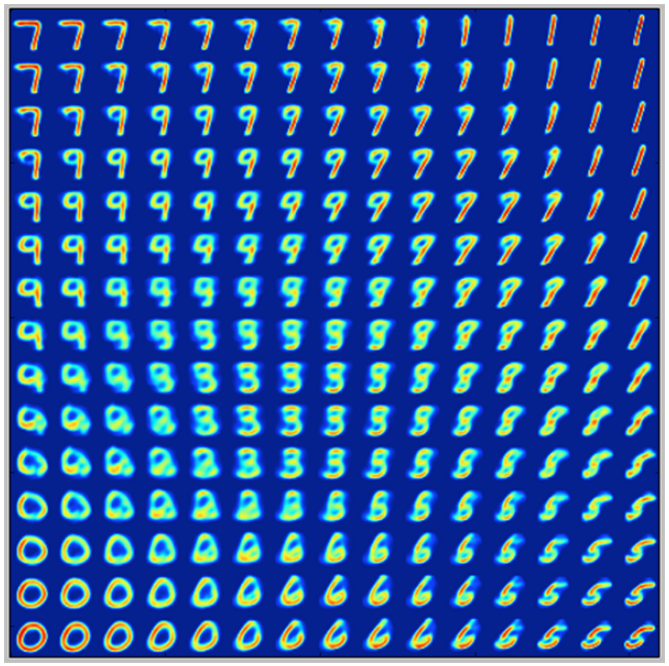
- osiąga minimum dla $\sigma_i = 1, \mu_i = 0$
- wymusza rozkład zakodowanych danych w centrum przestrzeni ukrytej, minimalizacja kosztu rekonstrukcji konkuruje z regularyzajcą próbując rozdzielić grupy różnych danych

VAE na MNIST

wizualizacja danych treningowych w 2 wymiarowej warstwie ukrytej kodera



generowanie cyfr dla siatki 15x15 wartości 2 wymiarowej warstwy ukrytej dekodera

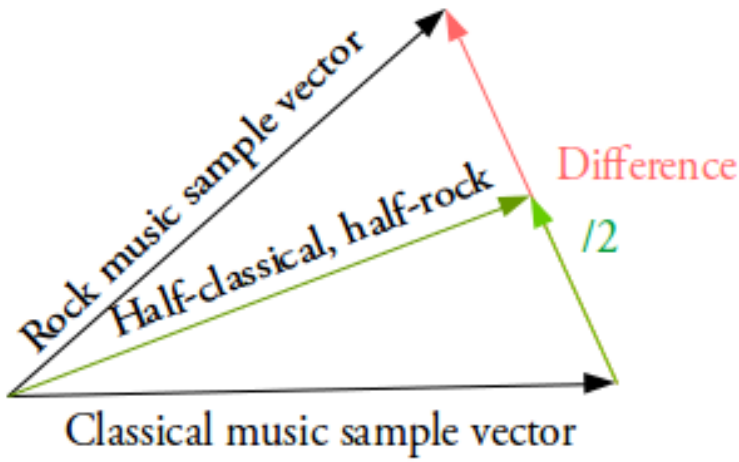


Architektura: Input-512-2-(μ, σ)-2-512-Output

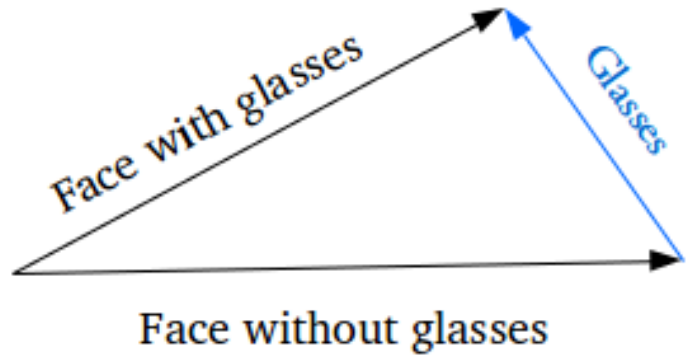
src: ☞ *Building Autoencoders in Keras*

Interpolowanie i generowanie pojęć

Arytmetyka w przestrzeni zmiennych utajonych



Środek między wektorami μ kodującymi pojęcia

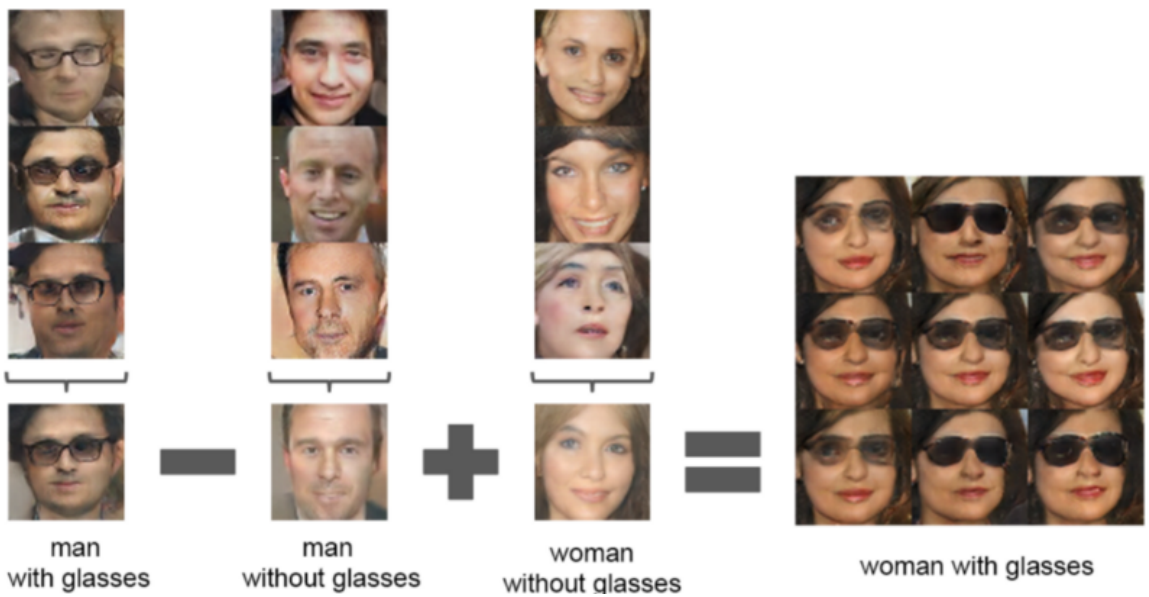


Róznica pomiędzy pojęciami może być dodana do innego pojęcia

☞ *Irhum Shafkat, Intuitively Understanding Variational Autoencoders*

Ukryta reprezentacja pojęć

- wyuczona reprezentacja danych koduje pojęcia
- arytmetyczne operacja na wektorach kodujących odpowiadają relacjom semantycznym pomiędzy pojęciami



Przykład: generowanie twarzy

Wejście

Rekonstrukcja

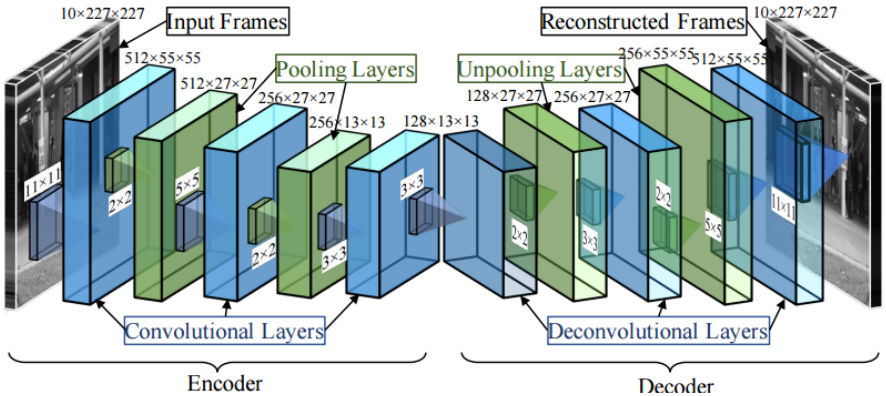
Losowe

☞ Deep Feature Consistent Variational Autoencoder

☞ *Variational auto-encoder trained on celebA*

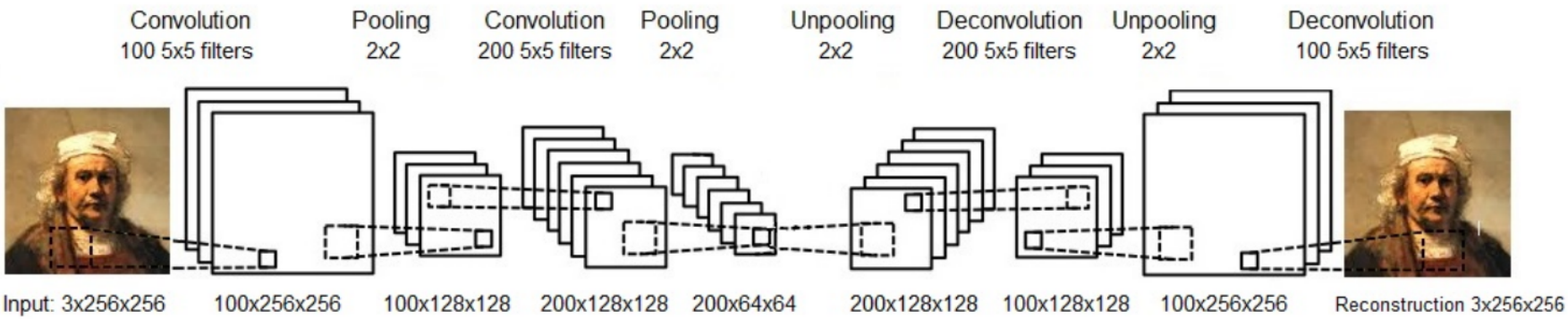
Inne typy autoenkoderów

- Convolutional AE, dekoder jest symetryczną do enkodera siecią dekonwolucyną

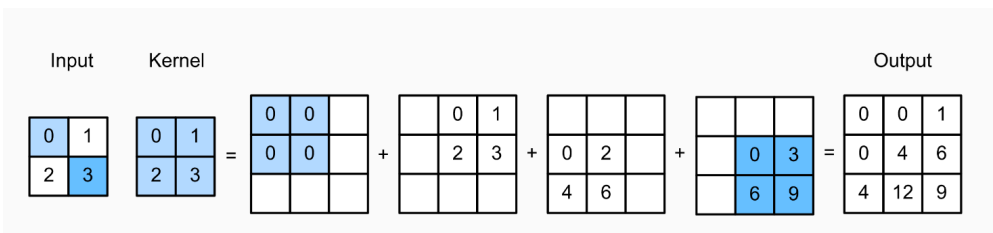


- sequence-to-sequence AU z użyciem jednostek rekurencyjnych (np. LSTM)
- głębokie autoenkodery - więcej warstw sprzyja kompresji danych o złożonych rozkładach
- stacked autoencoders - tworzenie głębokich sieci z uczonych sekwencyjnie autoenkoderów (metoda inicjalizacji)

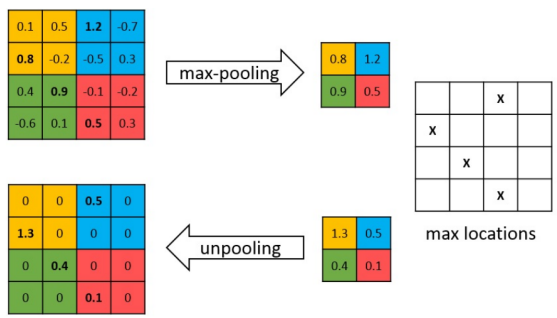
Deep Convolutional Autoencoder



deconvolution
transpozycja splotu



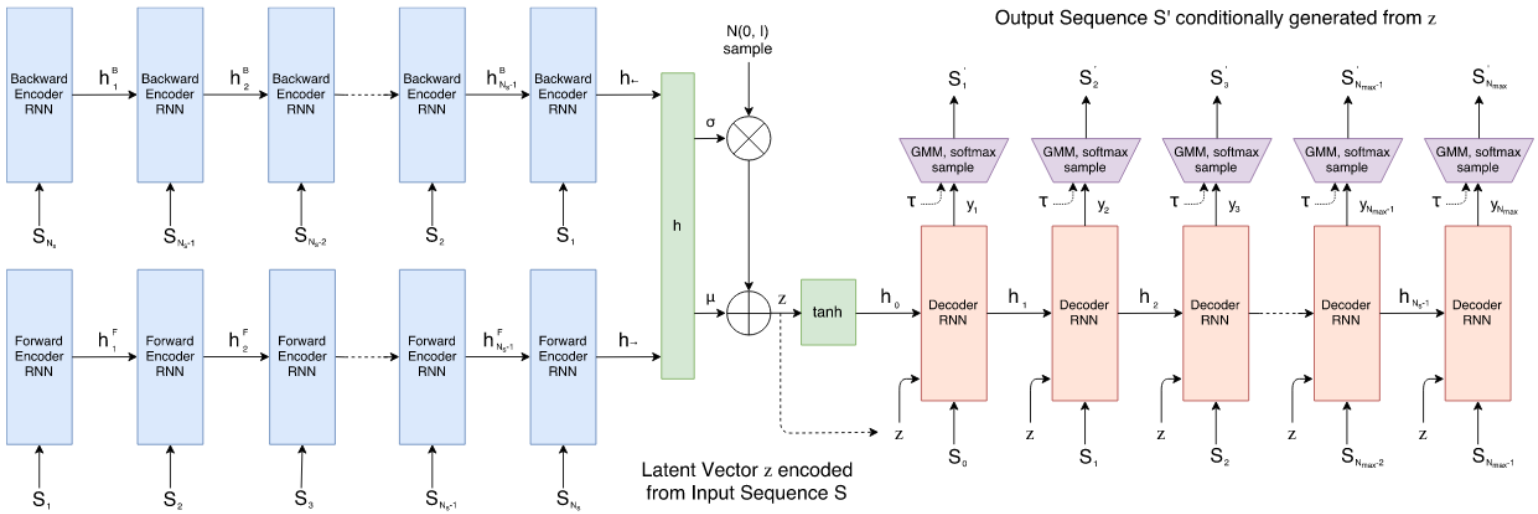
unpooling



Omid E. David, Nathan S. Netanyahu, DeepPainter: Painter Classification Using Deep Convolutional Autoencoders

Przykład: Sketch-RNN

Sequence-to-Sequence Variational Autoencoder do trenowania reprezentacji rysowania odręcznego

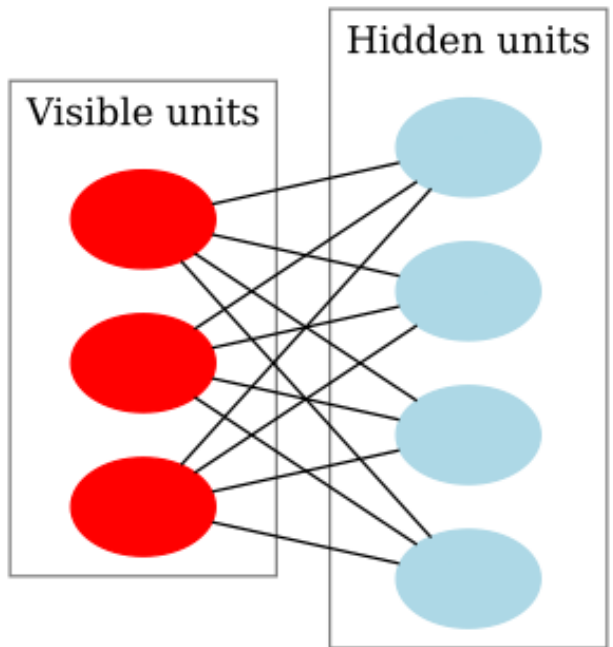


☞ Draw Together with a Neural Network

David Ha, Douglas Eck, A Neural Representation of Sketch Drawings

Restricted Boltzman Machine

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



Źródło grafiki: Wikipedia

Budowa RBM

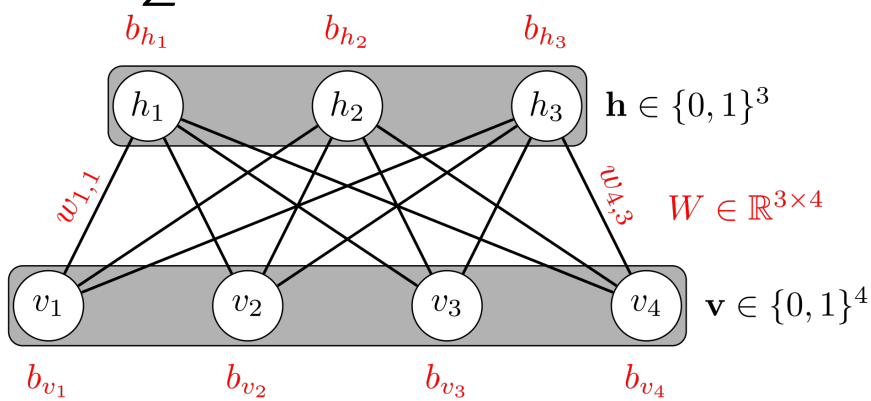
- binarne jednostki umieszczone w dwóch warstwach: widzialnej **v** i ukrytej **h**
- sygnał wejściowy podawany do jednostki widzialnej $\mathbf{v} = \mathbf{x}$
- energia układu

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

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 połączeń wewnątrz warstw, więc stany w warstwach są niezależne

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

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

Contrastive divergence CD

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

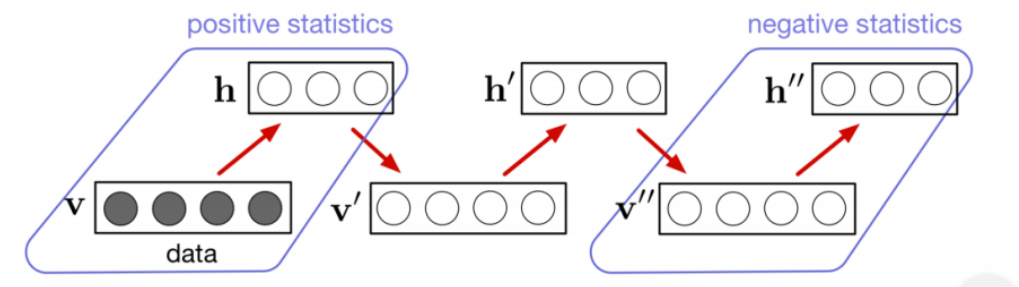
$$L = \frac{1}{N} \sum \log P(\mathbf{x})$$

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

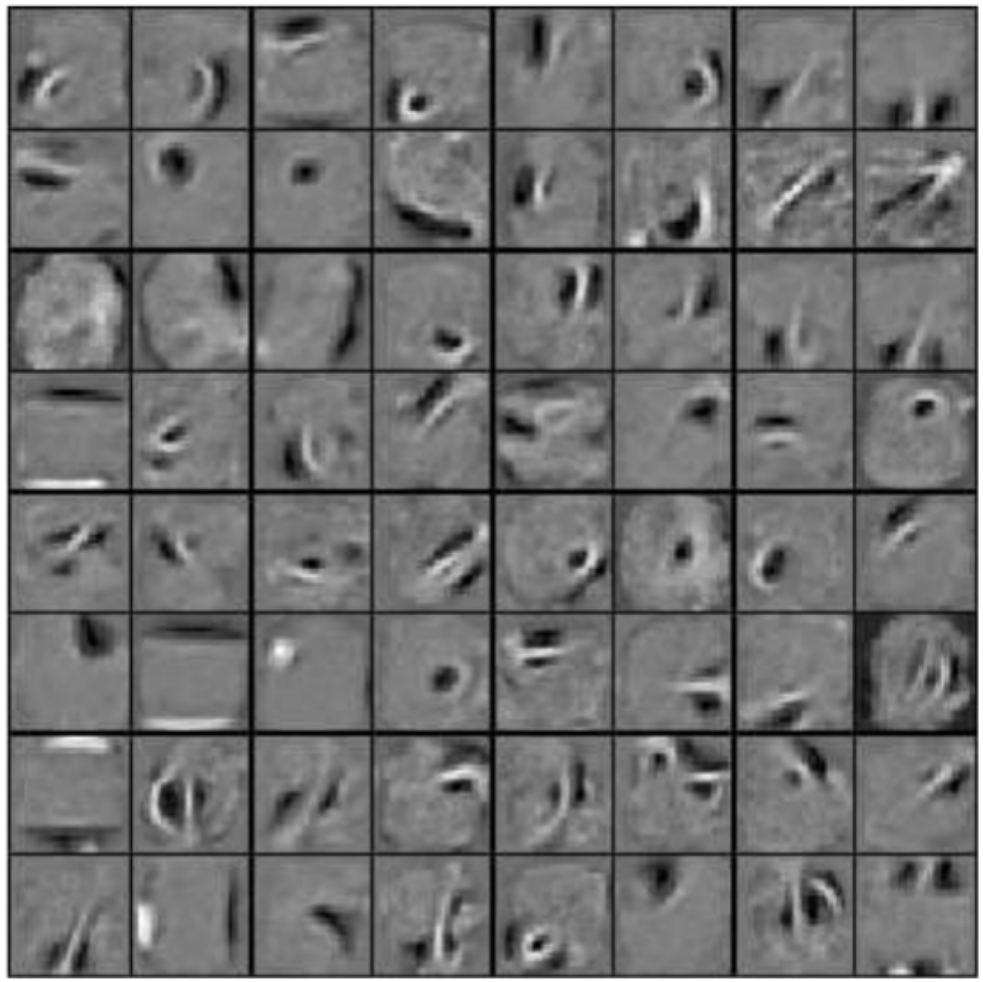
$$\Delta \mathbf{W} = \epsilon(\mathbf{x}\mathbf{h}^T - \mathbf{x}'\mathbf{h}'^T)$$

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

- w praktyce $k = 1$ daje dobre rezultaty

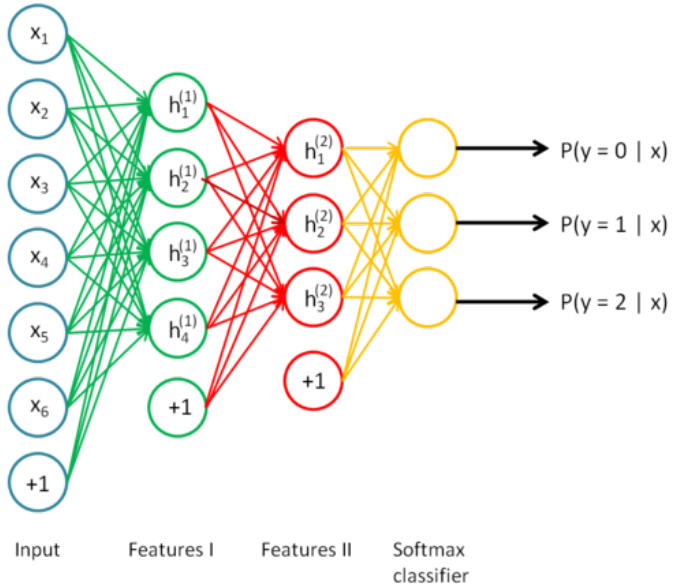


Filtry RBM na MNIST



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

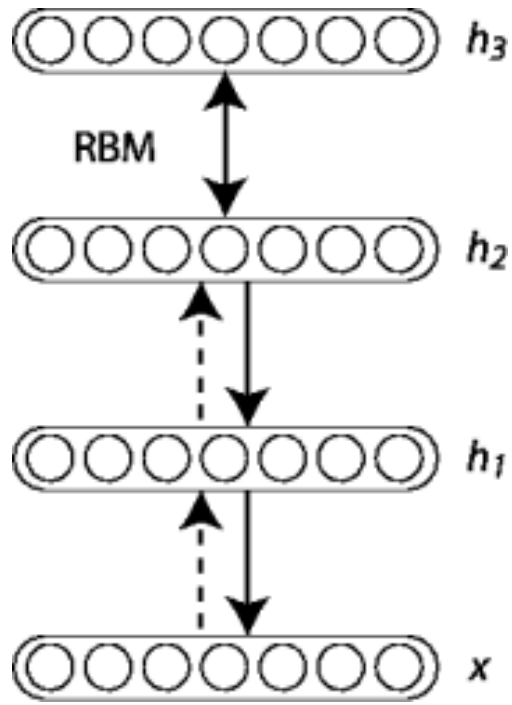
Stacked autoencoders



- Sygnał na wyjściu warstwy ukrytej autoenkodera staje się wejściem kolejnego autoenkodera
- Hierarchia reprezentacji: pierwsza warstwa koduje najprostsze warstwy, kolejne bardziej ogólne relacje, itd.
- Podobna architektura z wielowarstwowym RBM to Deep Belief Networks (DBN)

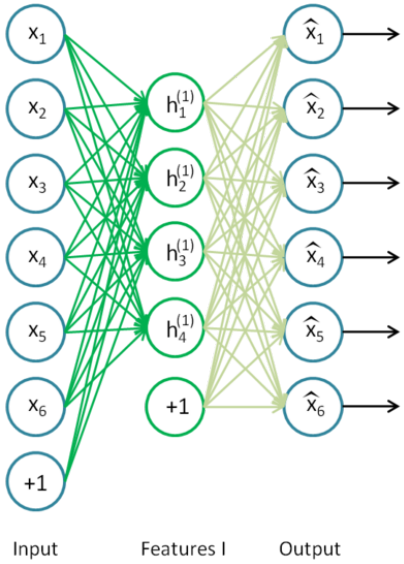
Deep Belief Networks (DBN)

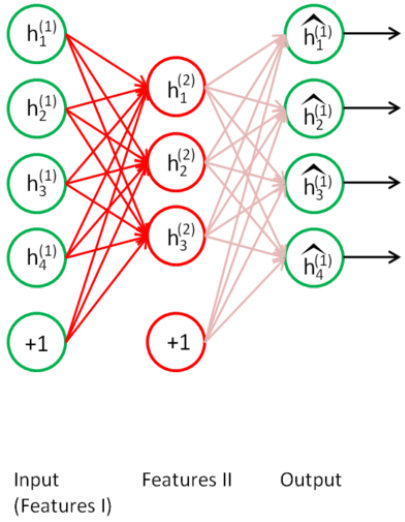
- DBN (Teh, Hinton 2006) wielowarstwowy RBM,
- uczy się głębokiej, hierarchicznej reprezentacji danych, każda kolejna warstwa analizuje sygnał widoczny w warstwie ukrytej poprzedniego RBM
- uczenie zachłanne: kolejne warstwy dodawane i uczone pojedynczo
- pierwszy efektywny model głęboki
- to nie jest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

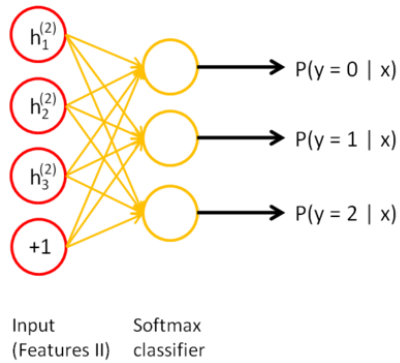


src: *http://deeplearning.net/tutorial/DBN.html*

Inicjalizacja sieci DNN



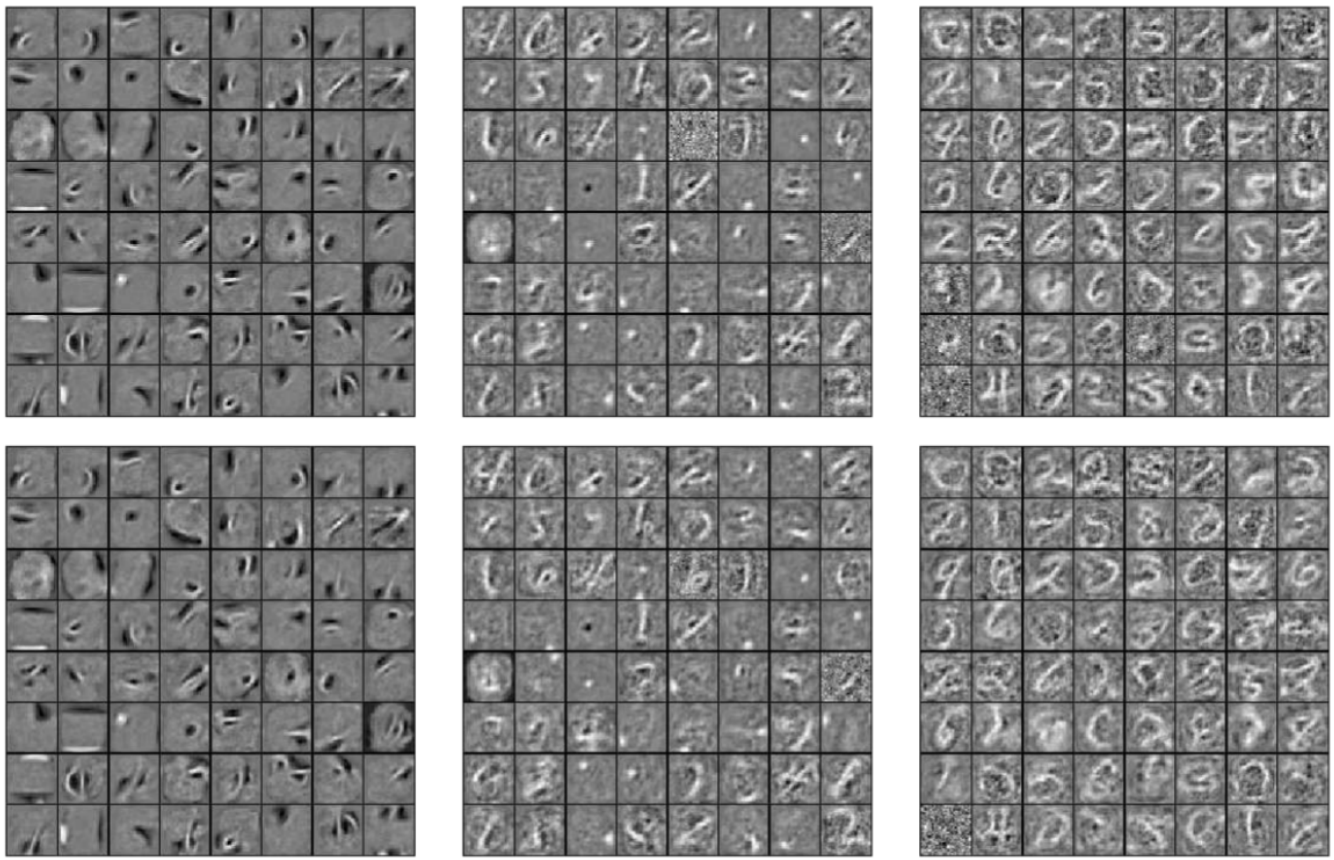




1. *Greedy layer-wise pre-training*:
   - trening nienadzorowany autoenkodera wsteczną propagacją
   - usuwamy warstwę wyjściową (dekoder) i uczymy kolejny AE, którego wejściem jest sygnał z warstwy ukrytej poprzedniego AE przy niezmiennych wartościach wag w poprzednich warstwach
   - powtarzamy procedurę 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

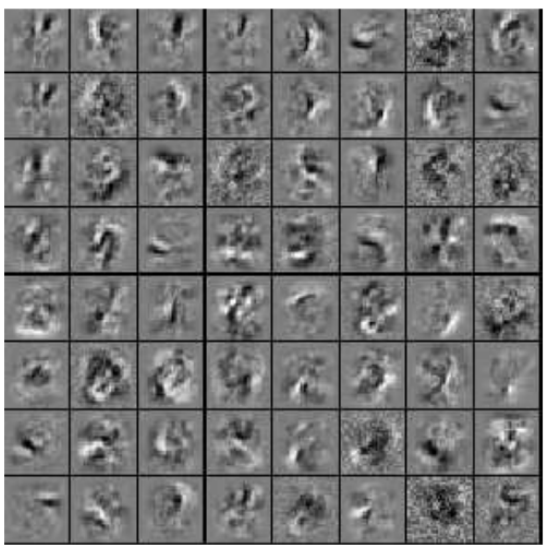
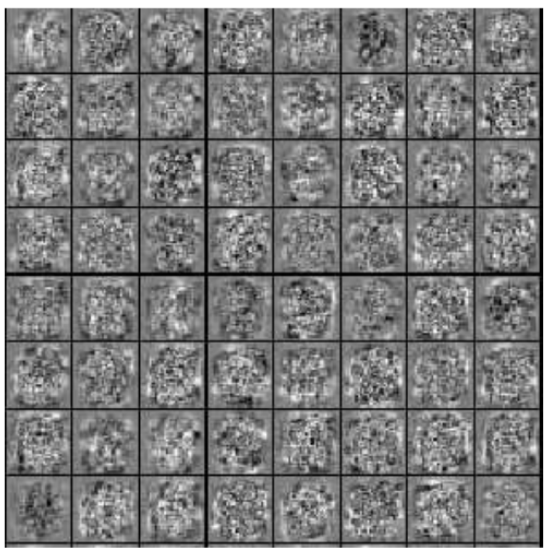
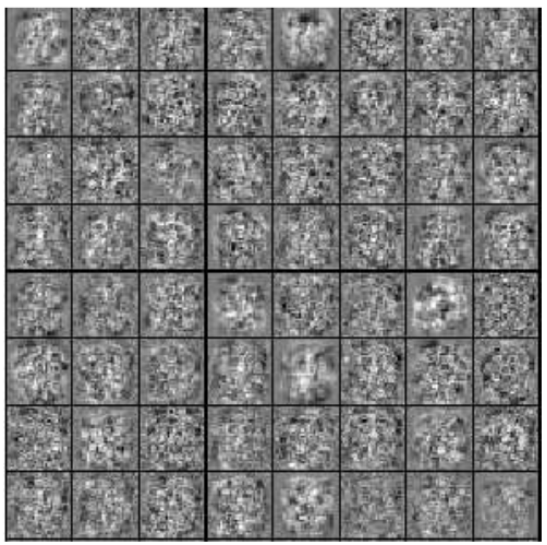


Erhan, Bengio, (2010) Why Does Unsupervised Pre-training Help Deep Learning?

fine tune

Filtry DBN na MNIST

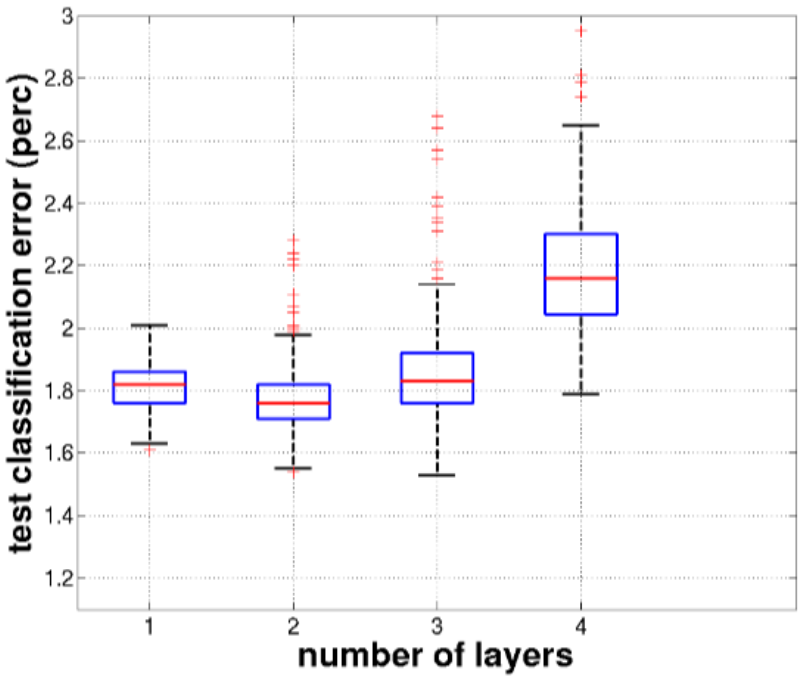
without pre-training

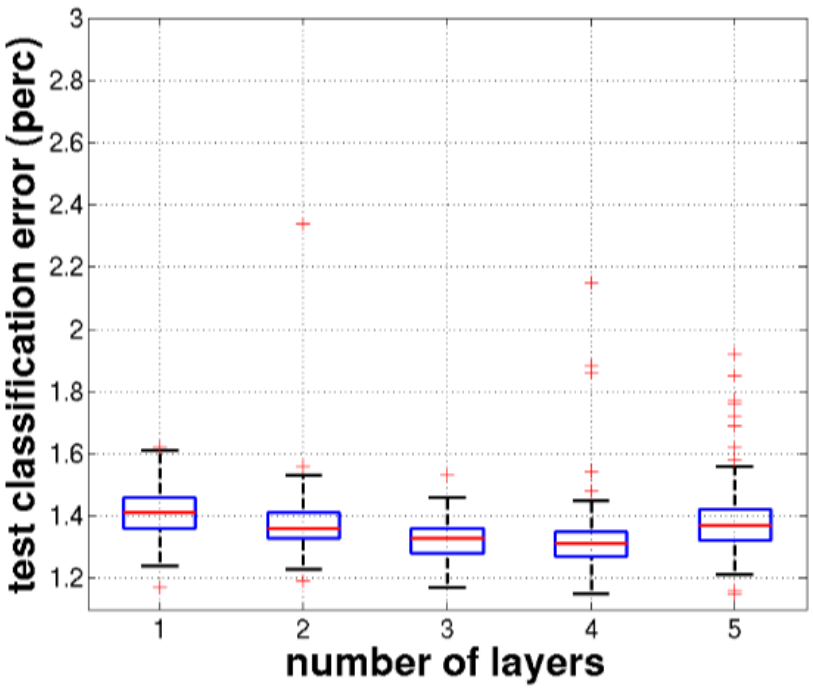




Erhan, Bengio, (2010) Why Does Unsupervised Pre-training Help Deep Learning?

Pre-trening DNN

Pre-trening poprawia generalizację, działa jak regularyzacja

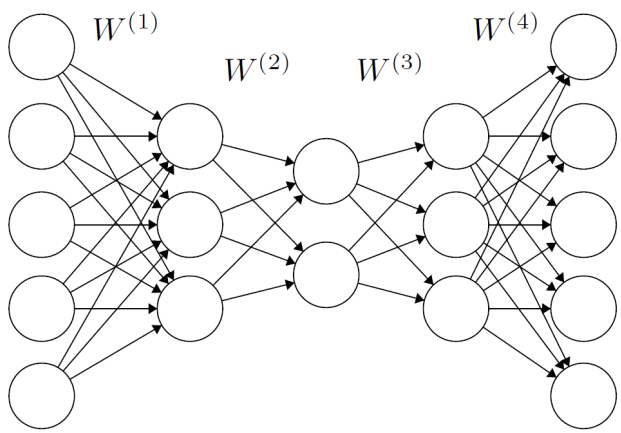




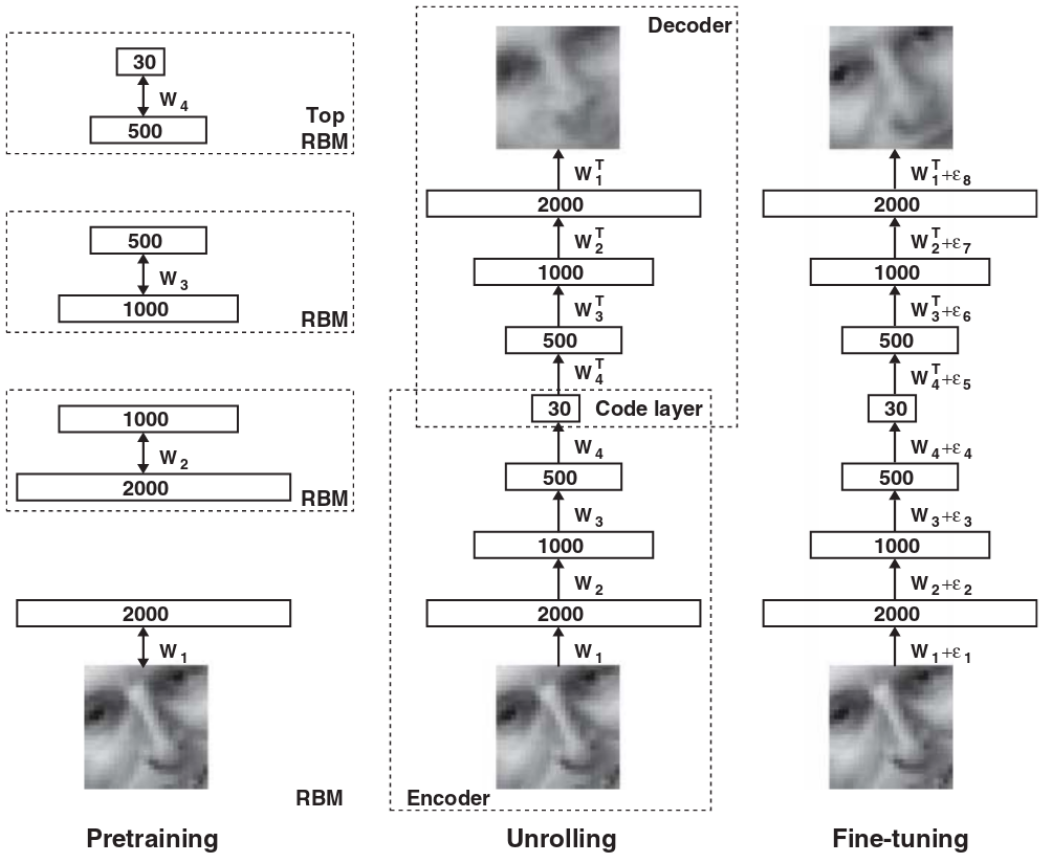
Erhan, Bengio, (2010) Why Does Unsupervised Pre-training Help Deep Learning?

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 głębokich AE



- RBM używany do inicjacji kodera i dekodera

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

Kompresja obrazów: Deep AE vs. PCA

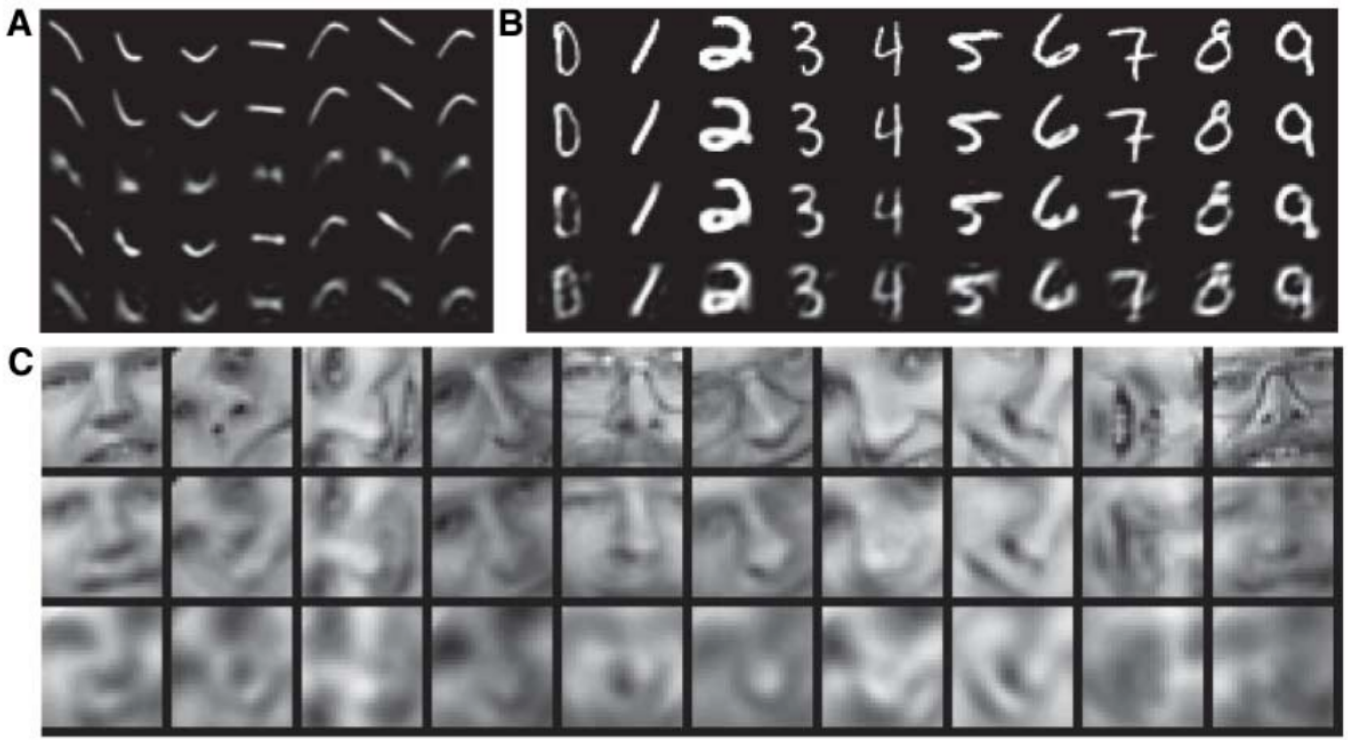
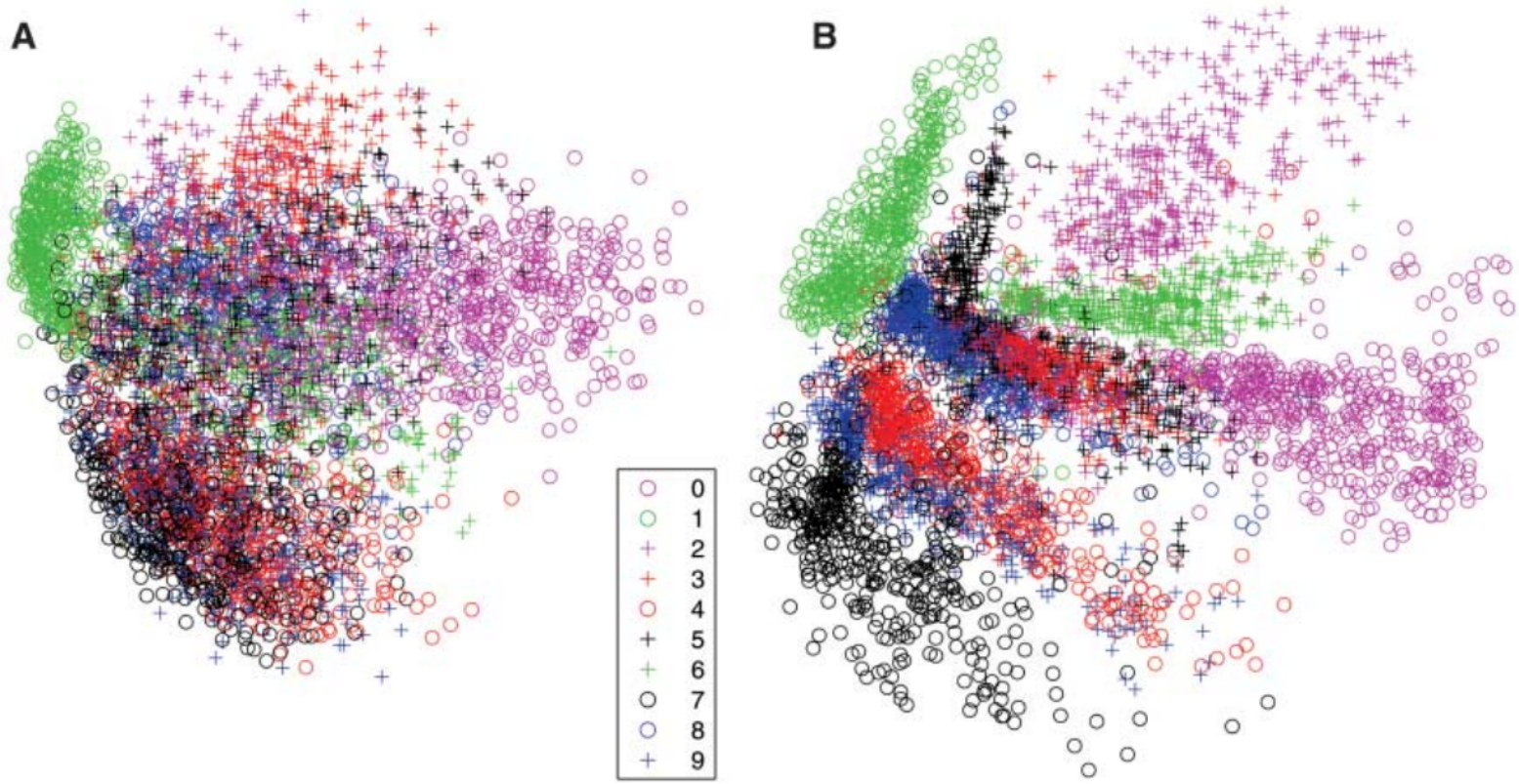


Fig. 2. (**A**) Top to bottom: Random samples of curves from the test data set; reconstructions produced by the six-dimensional deep autoencoder; reconstructions by "logistic PCA" (*8*) using six components; reconstructions by logistic PCA and standard PCA using 18 components. The average squared error 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)

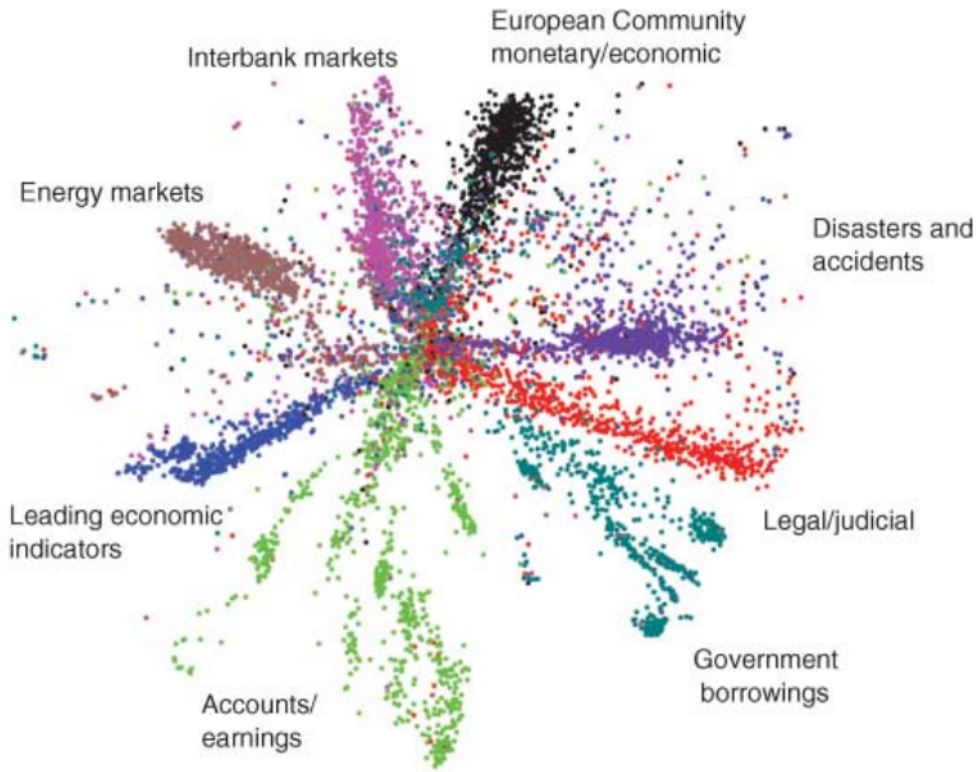
MNIST: PCA vs. Deep AE (784-1000-500-250-2)



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

Projekcja do 2D (dwa neurony kodujące)

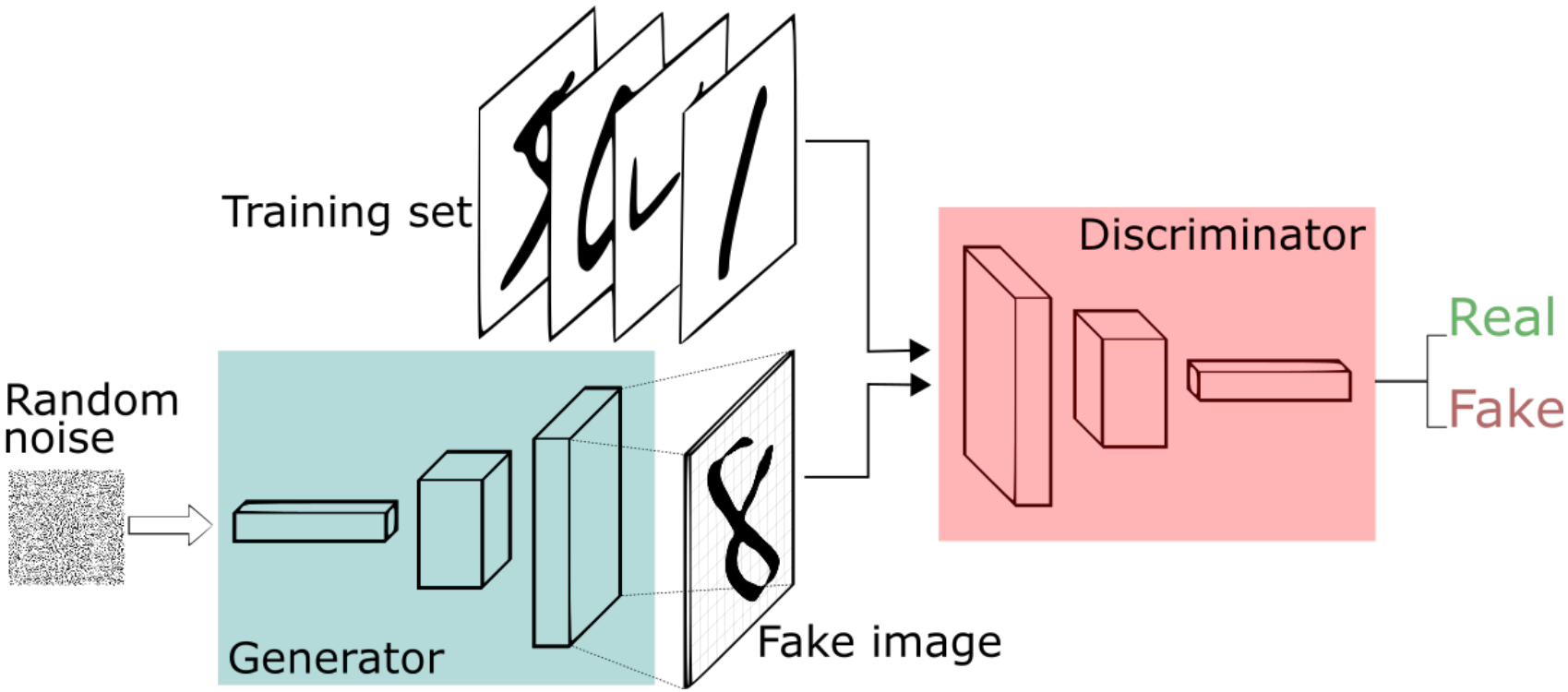
Wizualizacja dokumentów: architektura 2000-500-250-125-2



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

Generative Adversal Network (GAN)

Goodfelow 2014



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

Generative Adversal Network (GAN)

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

Trening GAN

Funkcja kosztu dyskryminatora - binarna entropia krzyżowa (spodziewana odpowiedź 1 dyskryminatora dla obrazu prawdziwego)

$$L_D = -\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{ data (x)}}} \log D(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$

gdzie $p_{data}(\mathbf{x})$ rozkład danych prawdziwych, $p_z(\mathbf{z})$ rozkład danych generowanych

Funkcja kosztu generatora

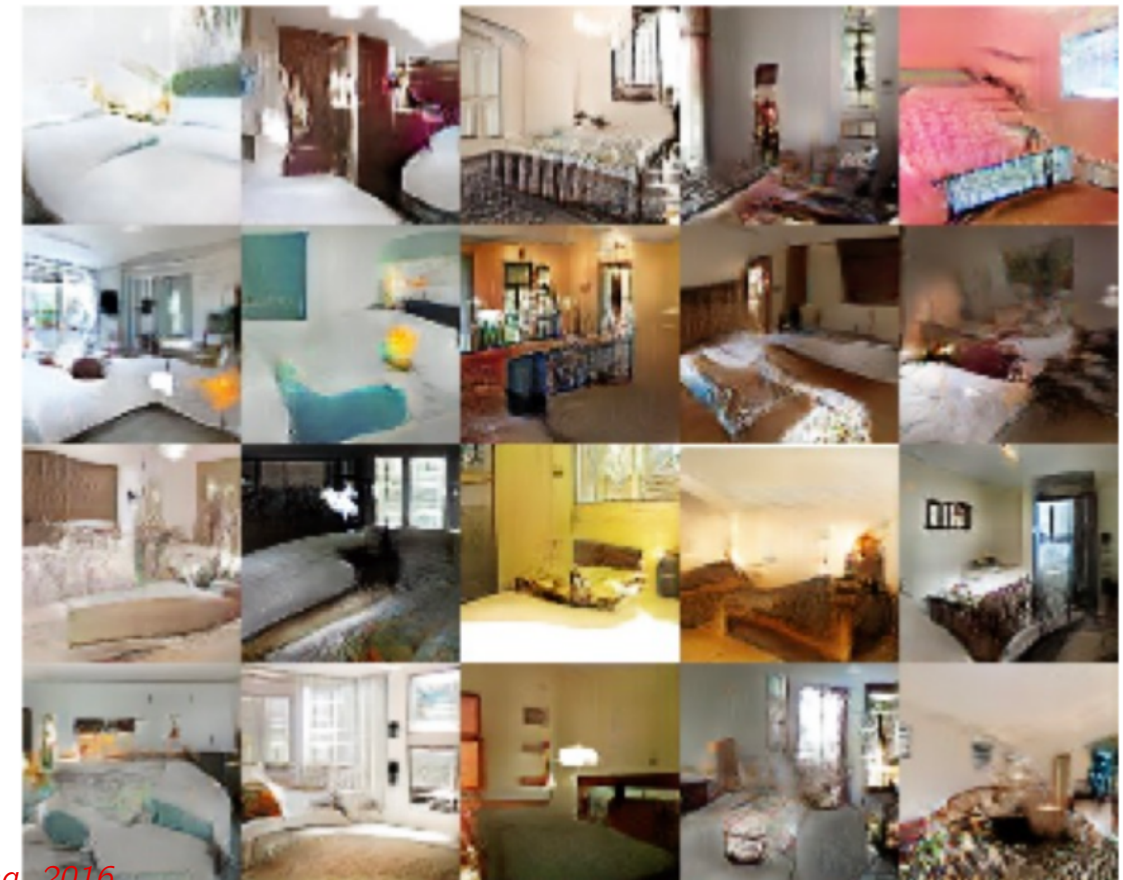
$$L_G = -\frac{1}{2}\mathbb{E}_z \log D(G(\mathbf{z}))$$

maksymalizuje prawdopodobieństwo popełnienia błędu przez dyskryminator (spodziewana odpowiedź 1 dyskryminatora dla obrazu fałszywego)

DCGAN - deep conv GAN

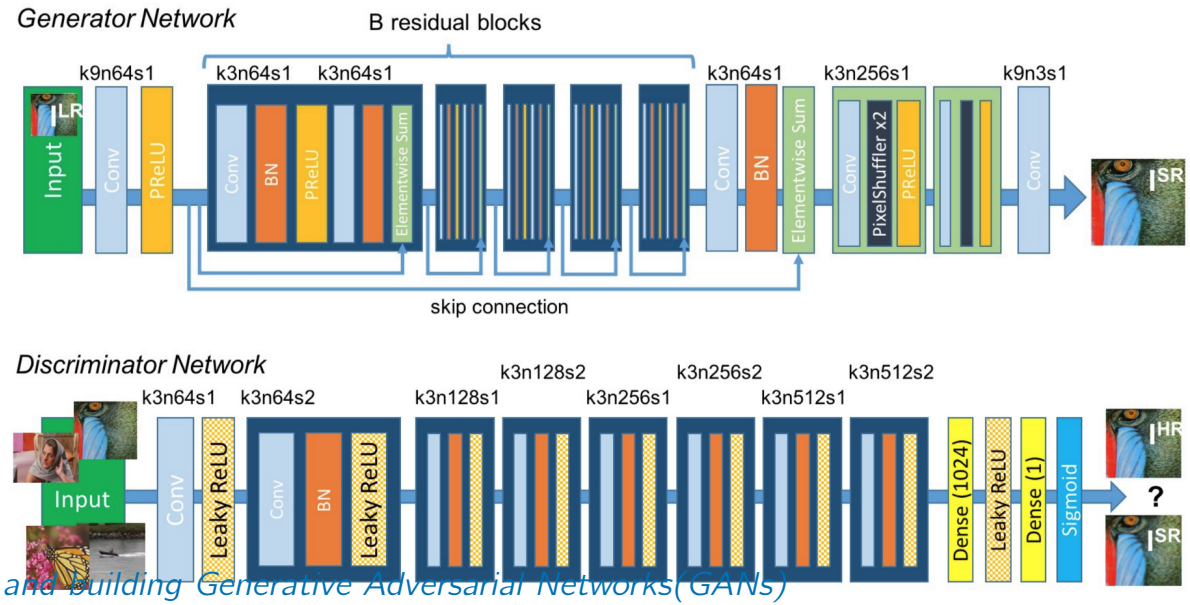
Denton et al., 2015

Generowanie obrazów wysokiej rozdzielczości



Goodfwlow, Deep Learning, 2016

SRGAN - super resolution GAN

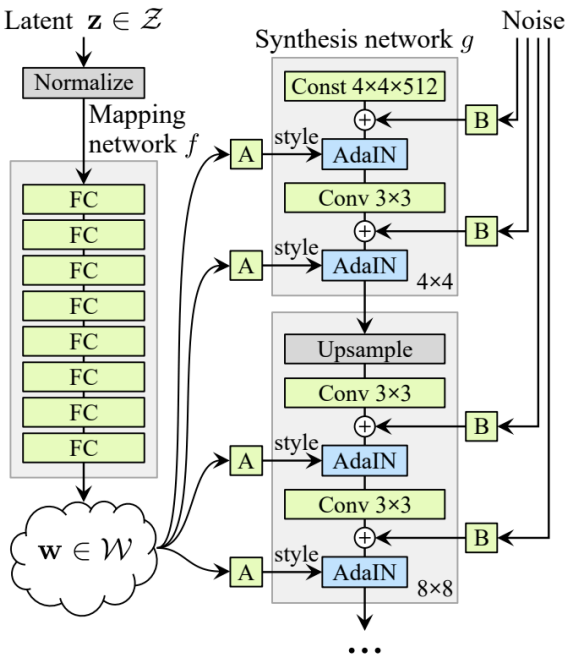


Rys: ☞ *Understanding and building Generative Adversarial Networks(GANs)*

Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

StyleGAN

Generator



T. Karras, S. Laine, T. Aila (2019) A Style-Based Generator Architecture for Generative Adversarial Networks

Inne zastosowania

- rekonstrukcja obrazów 3D ze zdjęć
- generowanie tekstu, synteza mowy, ...
- modyfikacje obrazów: zmiana twarzy, mimiki, fryzury, postarzanie osób na zdjęciach
- transfer stylu
- generowanie scen w grach wideo, zwiększanie rozdzielczości tekstur
- bezpieczeństwo: podnoszenie skuteczności algorytmów w wykrywaniu ataków cybenetycznych, wykrywanie fałszerstw, wykrywanie anomalii
- generowanie dane na potrzeby uczenia maszynowego
- generowanie obrazów z tekstu, generowanie nagrań wideo
- ☞ Curated list of awesome GAN applications and demo