*Stanisław OSOWSKI**

DOI: 10.15199/59.2018.5.2

Głębokie sieci neuronowe i ich zastosowania w eksploracji danych

Deep neural networks in application to data mining

Artykuł dotyczy podstaw działania typowych sieci neuronowych głębokich, do których zalicza się sieci konwolucyjne (CNN), autoenkoder czy sieć LSTM. Omówiono struktury tych sieci, ich algorytmy uczenia oraz dokonano przeglądu podstawowych zastosowań owych sieci w rozwiązywaniu różnego rodzaju zadań eksploracji danych, między innymi klasyfikacji, regresji, segmentacji danych, rekonstrukcji danych i wielu innych. Przedstawiono między innymi wyniki prac dotyczących bioinżynierii, w szczególności rozpoznawanie i klasyfikację obrazów mammograficznych.

Słowa kluczowe: głębokie uczenie, sieć CNN, autoenkoder, sieć LSTM, klasyfikacja, predykcja

The paper presents the theoretical fundamentals of deep neural networks. The basic deep structures are discussed. They include convolutional neural networks (CNN), autoencoder (AE) and recurrent network called LSTM (Long Short-Term Memory). This paper is concerned on presentation of their typical structures and learning algorithms. The review of some chosen applications of these solutions in such tasks as classification, regression, segmentation of data, reconstruction, text and speech recognition, etc., are discussed. Some chosen numerical results concerning classification problems will be also presented and discussed.

Key words: deep learning, CNN, LSTM, autoencoder, classification, prediction

W ostatnich latach ogromny postęp w eksploracji danych dokonał się za pośrednictwem tak zwanego głębokiego uczenia. Głębokie uczenie dotyczy wielowarstwowych sieci neuronowych, które pełnią jednocześnie funkcję generatora cech diagnostycznych dla analizowanego procesu oraz finalną funkcję klasyfikatora bądź układu regresyjnego. Uzyskuje się w ten sposób doskonałe narzędzie zastępujące człowieka przede wszystkim w trudnej dziedzinie opisu procesu za pomocą specjalizowanych deskryptorów, których stworzenie wymaga dużych zdolności eksperckich. Okazuje się przy tym, że takie podejście do bezinterwencyjnej metody generacji cech jest o wiele skuteczniejsze od stosowanych tradycyjnie metod generacji deskryptorów. Umożliwia przy tym poprawę dokładności działania systemu. Z tego powodu technologia sieci głębokich stała się ostatnio bardzo szybko jednym z najbardziej popularnych obszarów w dziedzinie nauk komputerowych.

Za protoplastę tych sieci można uznać zdefiniowany na początku lat dziewięćdziesiątych wielowarstwowy neocognitron Fukushimy [1]. Prawdziwy rozwój tych sieci zawdzięcza się jednak profesorowi LeCun [2], który zdefiniował podstawową strukturę i algorytm uczący specjalizowanej sieci wielowarstwowej, zwanej *Convolutional Neural Network* (**CNN**). Obecnie CNN stanowi podstawową strukturę stosowaną na szeroką skalę w przetwarzaniu obrazów. Tymczasem powstało wiele odmian sieci, będących modyfikacją struktury podstawowej CNN (np. UNN), jak również sieci różniących się zasadniczo od CNN. Przykładem mogą być autoenkoder (AE), jako wielowarstwowe, nieliniowe uogólnienie liniowej sieci PCA [3], sieci rekurencyjne typu **LSTM** (*Long Short-Term Memory*) [4], stanowiące skuteczne rozwiązanie problemu propagacji wstecznej w czasie lub ograniczona (wielowarstwowa) maszyna Boltzmanna **RBM** (*Restricted Boltzmann Machine*) używana w sieciach głębokiej wiarygodności **DBN** (*Deep Belief Network*) [5].

Cechą wspólną tych rozwiązań, zwłaszcza w przypadku CNN, jest wielowarstwowość ułożenia neuronów i ogromna (sięgająca milionów) liczba połączeń wagowych między neuronami. W celu uniknięcia problemu lawinowego narastania liczby adaptowanych wag, stosuje się powszechnie połączenia typu lokalnego. W tego typu rozwiązaniach neuron jest zasilany nie przez wszystkie sygnały neuronowe (piksele obrazów) warstwy poprzedzającej, jak w sieciach klasycznych, ale przez wybraną małą grupę neuronów (pikseli) tej warstwy, tworzących maskę filtrującą. Analiza całego obrazu następuje przez przesuwanie tej maski z ustalonym krokiem (*stride*) wzdłuż i wszerz obrazu. Charakterystyczną cechą jest przy tym używanie identycznych wartości wag przesuwającej się maski filtracyjnej.

Każdy rodzaj sieci głębokich związany jest z problemem adaptacji ogromnej liczby parametrów sieci w akceptowalnym czasie. Skuteczna implementacja i rozwój koncepcji LeCuna stały się możliwe dzięki olbrzymiemu postępowi w rozwoju technologii informatycznych, umożliwiającemu ogromne przyśpieszenie obliczeń. Połączenie najnowszych technologii z proponowanymi rozwiązaniami sieci głębokich stworzyło podstawy zastosowań sztucznej inteligencji w życiu codziennym.

Realne stało się zaprojektowanie pojazdów samosterujących, których podstawą są rozpoznawanie obrazów i obiektów prezentowanych na tych obrazach w czasie rzeczywistym, sterowanie robotami, skuteczne rozpoznawanie głosu, automatyczna generacja tekstu na podstawie wypowiedzi głosowej, segmentacja złożonych obrazów, zwłaszcza biomedycznych, przewidywanie szeregów czasowych (na przykład zapotrzebowania na energię elektryczną). Ważną sferą zastosowań jest wspomaganie diagnostyki medycznej dzięki skutecznym algorytmom przetwarzania obrazów medycznych, na przykład mikroskopowych, histologicznych itp.

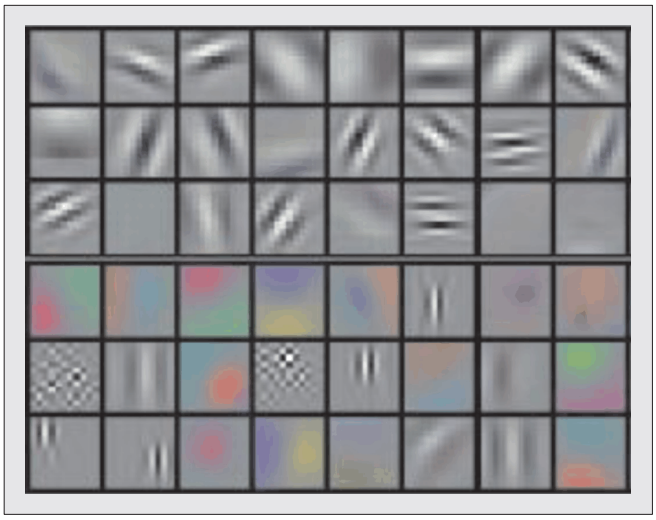
W artykule ograniczono się do przedstawienia trzech najbardziej charakterystycznych rozwiązań głębokich, do których należą sieci CNN, AE oraz LSTM. Przedstawiono zasady ich działania, jak również podstawowe przykłady zastosowań w rozwiązywaniu zagadnień praktycznych.

SIECI KONWOLUCYJNE CNN

Sieci konwolucyjne powstały jako narzędzie analizy i rozpoznawania obrazów wzorowane na sposobie działania naszych zmysłów. Wyeliminowały one kłopotliwy i trudny dla użytkownika etap manualnego opisu cech charakterystycznych obrazów, stanowiących atrybuty wejściowe dla końcowego etapu klasyfikatora bądź układu regresyjnego. W tym rozwiązaniu sieć sama

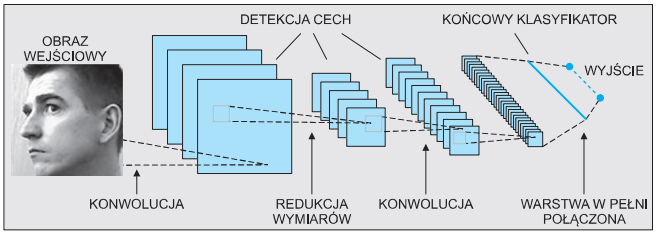
* **Politechnika Warszawska, Wojskowa Akademia Techniczna, e-mail: sto@iem.pw.edu.pl**

odpowiada za generacje cech. Poszczególne warstwy sieci CNN przetwarzają obrazy z warstwy poprzedzającej (na wstępie jest to zbiór obrazów oryginalnych), poszukując prymitywnych cech (np. grupy pikseli o podobnym stopniu szarości, krawędzie, przecinające się linie itp.). Kolejne warstwy ukryte generują pewne uogólnienia cech z warstwy poprzedzającej, organizowane w formie obrazów. Przykład wyników takiego przetworzenia obrazu kolorowego w warstwie ukrytej sieci, zawierającej 48 neuronów (każdy o różnych wartościach wag), pokazano na rys. 1.



Rys. 1. Przykład wyników przetwarzania obrazu kolorowego w cechy diagnostyczne w jednej warstwie ukrytej o 48 neuronach przez sieć CNN [6]

Obrazy te w niczym nie przypominają oryginałów, ale reprezentują cechy charakterystyczne dla nich, wyrażone w formie szarości pikseli. W efekcie ostatnia warstwa konwolucyjna generuje obrazy stosunkowo niewielkich wymiarów, reprezentujące cechy charakterystyczne dla przetwarzanego zbioru. Obrazy te są reprezentowane przez tensory, których elementy ulegają następnie przetwarzaniu na postać wektorową, będącą początkiem układu w pełni połączonego, stanowiącego właściwy klasyfikator bądź układ regresyjny. Przykład takiej struktury CNN przedstawiono na rys. 2.



Rys. 2. Przykład struktury CNN, zawierającej 3 warstwy konwolucyjne o połączeniu lokalnym i w pełni połączoną strukturę końcową stanowiącą właściwy klasyfikator

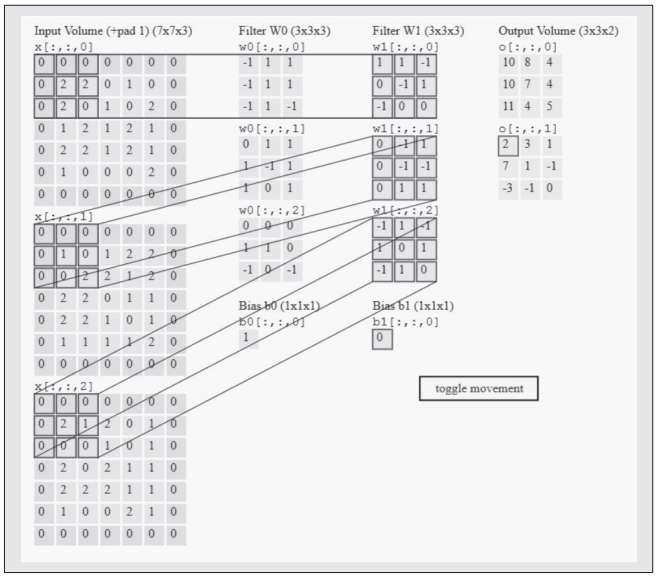
Kolejne warstwy sieci mają za zadanie wygenerowanie w sposób automatyczny (bez udziału człowieka) zbioru cech charakterystycznych dla analizowanych wzorców obrazowych, który następnie podlega klasycznemu przetworzeniu w decyzję klasyfikacyjną bądź regresyjną.

Nazwa sieci CNN pochodzi od operacji splotu (konwolucji), stanowiącej istotny element procesu obliczeniowego. W przypadku obrazów tablica danych jest reprezentowana przez dwuwymiarową macierz **I** o elementach reprezentujących stopnie jasności pikseli $I(m,n)$, a jądro **K** jest dwuwymiarowe. Operacja splotu dwuwymiarowego jest wówczas zapisana w postaci [6]:

$$\mathbf{Y}(i,j) = \mathbf{I}(i,j) * \mathbf{K}(i,j) = \sum_m \sum_n \mathbf{I}(m,n)\mathbf{K}(i-m, j-n) = \\ = \sum_m \sum_n \mathbf{I}(i-m, j-n)\mathbf{K}(m,n) \,. \quad (1)$$

W sieciach CNN wejściem dla pierwszej warstwy ukrytej jest reprezentacja RGB obrazów ustawionych w formie tensora. Operacja konwolucji w pierwszej warstwie konwolucyjnej obejmuje wszystkie trzy kanały RGB obrazu (suma poszczególnych kanałów), każdy z innymi przyjętymi wartościami wag jądra w postaci filtru liniowego stanowiącego neuron analizujący. W dalszych warstwach operacja ta obejmuje wiele (ustalonych przez użytkownika) obrazów z warstwy poprzedzającej.

Przykład operacji konwolucyjnej dla kroku przesunięcia filtru równego 2 przedstawiono na rys. 3 [6]. Dane wejściowe stanowią 3 kanały RGB obrazu. Każdy kanał jest przetwarzany przez odpowiedni dla niego filtr (neuron), a wyniki przetworzenia podlegają sumowaniu z uwzględnieniem polaryzacji, tworząc obraz wynikowy. W efekcie w każdej warstwie konwolucyjnej filtr jest wielowymiarowy, realizujący operacje tensorowe, przy czym wymiar maski filtrującej jest równy $n_x \times n_y$, a liczba tych masek tworzących jeden filtr jest równa liczbie obrazów wejściowych dla danej warstwy podlegających przetworzeniu (maksymalnie liczba obrazów poprzedniej warstwy). W przypadku pierwszej warstwy konwolucyjnej i obrazu reprezentowanego przez 3 kanały RGB każdy filtr składa się z trzech jednostek (filtr W0 lub W1). Liczba obrazów wyjściowych po konwolucji jest równa liczbie zastosowanych filtrów wielokanałowych (w przykładzie są dwa filtry W0 i W1, stąd dwa obrazy wyjściowe reprezentowane przez *Output Volume*).



Rys. 3. Przykład konwolucji obrazu reprezentowanego przez kanały RGB z 3 jednostkami tworzącymi filtr W0 (wynik obrazowy w postaci macierzy o(:,:,0)) oraz 3 jednostkami filtru W1 (wynik obrazowy o(:,:,1)) [6]

W przetwarzaniu danych operację konwolucji stosowaną w CNN wyróżniają ważne zalety w stosunku do zwykłych operacji macierzowych w sieciach klasycznych. Należą do nich: lokalność połączeń, wspólne (powtarzalne) wartości wag połączeń filtru przesuwającego się po obrazie i powstała dzięki temu niezmienniczość (ekwiwariancja) względem przesunięcia.

Każdy sygnał wyjściowy filtru podlega działaniu nieliniowej funkcji aktywacji, która w przypadku CNN przybiera najczęściej postać **ReLU** (*Rectified Linear Unit*) opisaną wzorem [3].

$$y(x) = \begin{cases} x & \text{dla} \quad x \geq 0 \\ 0 & \text{dla} \quad x < 0 \end{cases} \quad (2)$$

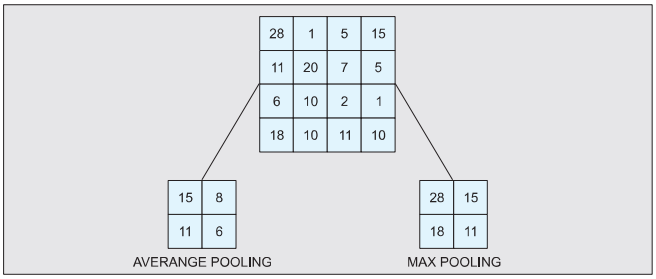
Funkcja ta charakteryzuje się nieciągłą pochodną. Ten fakt utrudnia uczenie sieci metodami gradientowymi. Stąd zazwyczaj zastępuje się ją aproksymacją zwaną *softplus*, zdefiniowaną następująco [3]:

$$y(x) = \ln(1+e^{x}). \quad (3)$$

Jej pochodna jest ciągła i reprezentuje sigmoidę unipolarną. W wyniku konwolucji i działania funkcji ReLU uzyskuje się wartości szarości pikseli obrazu wynikowego.

Liczba neuronów konwolucyjnych, filtrujących poszczególne obszary lokalne obrazu, jest ustalana przez użytkownika. Każdy neuron ma określoną przez użytkownika liczbę wag podlegających doborowi i łączących go z określonym rejonem lokalnym obrazu. Dla typowych wielkości maski 3x3 (małe obrazy) lub 15x15 (duże obrazy) liczba wag neuronu staje się więc równa 9 lub 225 (plus waga polaryzacji). Przesuwna maska analizy wędrująca po obrazie wejściowym dostarcza do wyjścia neuronu sygnał sumacyjny, będący sumą iloczynów jasności pikseli i odpowiednich, połączonych z nimi, wag neuronu na zasadzie konwolucji. Neuron ze swoimi wagami stanowi więc filtr przesuwany po obrazie z określonym krokiem. Przesuwanie maski filtracyjnej neuronu sterowane jest przez wybór wielkości kroku *stride*. Wielkość *stride*=1 oznacza, że nowa maska przesuwana jest o jeden piksel obrazu (w kierunku *x* lub *y*) w stosunku do pozycji poprzedniej maski. Rozmiar obrazu wynikowego zależy od wybranego kroku *stride* i wielkości maski filtrującej.

W następnym kroku powstały obraz wynikowy poddaje się działaniu operacji łączenia sąsiednich wyników, tzw. operacji *pooling*. Funkcja ta przekształca wynik nieliniowego działania filtracyjnego neuronu w określonym rejonie obrazu przez zdefiniowaną statystykę dotyczącą najbliższych mu pod względem lokalizacji wyników wyjściowych. Typowe funkcje to: *max pooling*, zwracająca wartość maksymalną wyników neuronów znajdujących się w sąsiedztwie prostokątnym aktualnego wyniku oraz *average pooling* zwracająca wartość średnią wyników w tym obszarze. Przykład takiej operacji uwzględniającej obszar maski 2x2 przedstawiono na rys. 4.



Rys. 4. Przykład wyników działania operacji *average pooling* i *max pooling* z maską 2x2 i krokiem *stride* =2 na obrazie wynikowym o wymiarach 4x4

Operacja *pooling* – poza zmniejszeniem wymiaru obrazu – umożliwia uzyskanie reprezentacji bardziej inwariantnej względem niewielkiego przemieszczenia danych. Stanowi pewne uogólnienie cech reprezentowanych przez sąsiednie grupy pikseli. Często dla zachowania odpowiednich wymiarów obrazu stosuje się operację uzupełnienia brzegów zerami, tzw. *zero padding*. Zwykle jest to uzupełnienie symetryczne z obu stron obrazu. Dodatkowo stosuje się niekiedy również normalizację stopnia szarości obrazów powstałych w wyniku konwolucji.

W poszczególnych warstwach ukrytych obrazy wynikowe tworzą tensory (strukturę trójwymiarową: dwa wymiary obrazu i liczba obrazów). Obrazy (macierze) ostatniej warstwy konwolucyjnej są przetwarzane na postać wektorową, stanowiącą początek sieci w pełni połączonej, w której każdy neuron jest połączony z każdym sygnałem warstwy poprzedniej – połączenia globalne (*Fully Connected layer* – **FC**). Neurony w warstwie o pełnym połączeniu mają unikalne wagi, podlegające adaptacji w procesie uczenia.

Przejście z reprezentacji tensorowej (wiele obrazów 2-wymiarowych) na wektorową (pojedyncze sygnały tworzące wejście dla właściwego klasyfikatora) może odbywać się wieloma sposobami. Jednym z nich jest przyjęcie wszystkich wartości pikselowych poszczególnych obrazów jako cech diagnostycznych i ustawienie ich w formie wektora przez operację tzw. *reshape*. Przy zbyt dużych wymiarach obrazów może być stosowane zmniejszenie rozdzielczości obrazów tensora. Innym rozwiązaniem jest zastosowanie operacji *pooling* z odpowiednio dużym krokiem *stride*. W niektórych przypadkach można zaprojektować sieć w taki sposób, że ostatnia warstwa konwolucyjna zawiera tylko pojedynczą wartość dla każdego kanału.

W większości rozwiązań sieci CNN zamiast klasycznej wielowarstwowej sieci neuronowej stosuje się klasyfikator typu *softmax*, w którym sygnały wejściowe są bezpośrednio przetwarzane na prawdopodobieństwo przynależności do określonej klasy [3]. Liczba neuronów wyjściowych jest równa liczbie klas, przy czym każdy neuron podlega adaptacji wag w klasyczny sposób przez optymalizację funkcji celu. Wartość sygnału sumacyjnego neuronu *i*-tego określona jest wówczas wzorem:

$$u_i(\mathbf{x}) = \sum_j w_{ij} x_j + w_0 . \quad (4)$$

Prawdopodobieństwo przynależności wektora **x** do *i*-tej klasy (*i*=1, 2, …, *M*) zależy od wartości funkcji *softmax*, obliczanej dla każdej składowej wektora wyjściowego według wzoru:

$$\mathrm{softmax}(\mathbf{u})_i = \frac{\exp(u_i)}{\sum_{j=1}^{M} \exp(u_j)} \quad (5)$$

Wartość największa wyznacza przynależność do klasy określonej wskaźnikiem *i*.

Ważnym elementem procesów występujących w CNN jest duża niezależność wyniku od rodzaju przetwarzanych obrazów, gdyż niezależnie od typu obrazu system przetwarzania koncentruje się na elementach prymitywnych, występujących w każdym obrazie. Stało się to przyczyną powstania specjalnego sposobu wstępnego uczenia sieci CNN, tzw. *Transfer Learning* na dowolnie dobranym zestawie danych uczących, który podlega jedynie douczeniu na danych użytkownika. Dzięki takiemu rozwiązaniu możliwe jest ogromne skrócenie czasu uczenia, a przy tym znaczne polepszenie zdolności generalizacyjnych sieci.

Takim prototypem dostępnym w Matlabie (poczynając od wersji z roku 2017) jest *AlexNet* [7], zaprojektowany przez grupę naukowców: Alex Krizhevsky, Ilya Sutskever and Geoff Hinton. W implementacji Matlaba struktura sieci AlexNet jest następująca [8].

```
1 ‚data' Image Input 227x227x3 images with ‚zero-
center' normalization
2 'conv1' Convolution 96 11x11x3 convolutions
with stride [4 4] and padding [0 0]
3 'relu1' ReLU
4 'norm1' Cross Channel Normalization with 5
channels per element
5 'pool1' Max Pooling 3x3 max pooling with stride
[2 2] and padding [0 0]
6 'conv2' Convolution 256 5x5x48 convolutions
with stride [1 1] and padding [2 2]
7 'relu2' ReLU
8 'norm2' Cross Channel Normalization with 5
channels per element
9 'pool2' Max Pooling 3x3 max pooling with stride
[2 2] and padding [0 0]
10 'conv3' Convolution 384 3x3x256 convolutions
with stride [1 1] and padding [1 1]
11 'relu3' ReLU
12 'conv4' Convolution 384 3x3x192 convolutions
with stride [1 1] and padding [1 1]
13 'relu4' ReLU
```

```
14 'conv5' Convolution 256 3x3x192 convolutions
with stride [1 1] and padding [1 1]
15 'relu5' ReLU
16 'pool5' Max Pooling 3x3 max pooling with stride
[2 2] and padding [0 0]
17 'fc6' Fully Connected 4096 fully connected
layer
18 'relu6' ReLU
19 'drop6' Dropout 50% dropout
20 'fc7' Fully Connected 4096 fully connected
layer
21 'relu7' ReLU
22 'drop7' Dropout 50% dropout
23 'fc8' Fully Connected 1000 fully connected
output layer
24 'prob' Softmax softmax
25 'output' Classification 1000 classes
```

W sumie w strukturze tej można wyróżnić 25 podwarstw (włączając w nie konwolucję liniową, operację ReLU, normalizację i *pooling*). Sieć zawiera 5 warstw konwolucyjnych i 3 warstwy FC. Pierwsza warstwa FC połączona z operacją ReLU jest warstwą zawierającą 4096 cech diagnostycznych. Druga warstwa FC (również połączona z operacją ReLU) jest warstwą ukrytą klasyfikatora końcowego zasilającą warstwę wyjściową 1000 neuronów (zapewnia rozpoznanie 1000 klas). Zastosowany w strukturze klasyfikator to *softmax*. Stosując transfer *learning* można wyodrębnić sygnały z pierwszej warstwy FC jako cechy diagnostyczne i zasilić nimi dowolnie wybrany klasyfikator, np. SVM czy MLP.

Istnieje obecnie wiele innych dostępnych struktur CNN, wytrenowanych wstępnie i dostosowanych do operacji *transfer learning*. Można tu wymienić między innymi następujące.

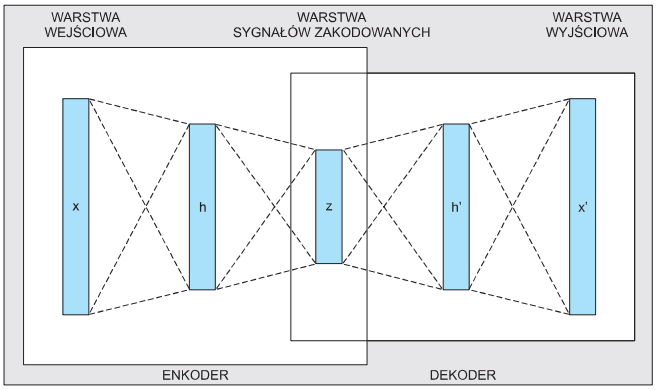
- **ZFNet** ukształtowany przez Matthew Zeilera i Roba Fergusa [9]. Jest to ulepszona wersja AlexNet uzyskana przez rozszerzenie wymiarów środkowych warstw konwolucyjnych i zmniejszenie parametru *stride* i wymiarów filtru w pierwszej warstwie.
- **GoogLeNet** – zmniejszona liczba parametrów podlegających uczeniu (z 60 milionów w AlexNet do 4 milionów) przez wprowadzenie modułu wstępnego przetwarzania. Poza tym stosuje *Average Pooling* zamiast *Max Pooling*, co zmniejsza utratę informacji w kolejnych etapach przetwarzania.
- **VGGNet** ukształtowany przez Karen Simonyana i Andrew Zissermana [10]. Sieć zawiera 15 CONV/FC warstw stosujących zuniformowaną strukturę stosującą filtry konwolucyjne 3x3 i filtry 2x2 w operacji *pooling*. Model dostępny w bibliotece Caffe.
- **ResNet** ukształtowany przez zespół Kaiming He [11]. Główną nowością jest pomijanie pewnych powiązań między sąsiednimi warstwami i występowanie powiązań z dalszą warstwą, jak również zastosowanie tzw. *batch normalization*, czyli normalizacji używanej do małego zbioru próbek uczących (wartość średnia zerowa i jednostkowa wariancja). Taka normalizacja zdecydowanie przyspiesza proces uczenia. Architektura unika również warstw końcowych globalnie połączonych.
- **U-NET** – struktura sieci CNN opracowana przez zespół Ronnebergera [12] dostosowana specjalnie do zadań segmentacji obrazów.

W bibliotece CAFFE [16] są dostępne programy komputerowe wybranych struktur sieci CNN, gotowych do użycia w trybie *transfer learning*.

SIEĆ AUTOENKODERA

Autoenkoder jest specjalnym rozwiązaniem głębokiej sieci neuronowej, którego zadaniem jest skopiowanie danych wejściowych w wyjściowe z akceptowalnym błędem [3]. Jest to wielowarstwowy układ autoasocjacyjny stowarzyszający ze sobą dane wejściowe. Układ składa się z dwu następujących po sobie części: układu kodującego rzutującego dane wejściowe w postaci wektora **x** w dane zakodowane, reprezentowane przez wektor **h**=**f**(**x**) i dekodera wykonującego działanie odwrotne, rekonstruujące dane wejściowe **x'**=**g**(**h**). Przykład struktury autoenkodera o jednej warstwie sygnałów zakodowanych przedstawiono na rys. 5. Dane wejściowe w postaci wektora **x** są kodowane w wektor **z** za pośrednictwem wektora sygnałów ukrytych **h** (część tworząca enkoder). Dekodowanie odbywa się w strukturze symetrycznej, odtwarzając za pośrednictwem wektora **h'** wektor **x'** (część zwana dekoderem).

Podstawą jego działania jest kompresja danych polegająca na wychwyceniu głównych cech procesu reprezentowanego przez



Rys. 5. Przykład struktury autoenkodera zawierający część kodującą (enkoder) i rekonstrukcję sygnałów wejściowych (dekoder)

zbiór wektorów **x**. Oznacza to, że dane zrekonstruowane **x'** nie reprezentują dokładnych kopii danych wejściowych, ale ich aproksymację. Wektor **h**, stanowiący podstawę kodowania i następnie rekonstrukcji, reprezentuje zatem cechy główne procesu, eliminując elementy nieistotne (szumowe) zawarte w zbiorze danych wejściowych **x**. Autoenkoder jest uogólnieniem dwuwarstwowej sieci liniowej PCA. Różni go od niej liczba warstw, która może być dowolna, jak również nieliniowa funkcja aktywacji neuronów stosowana w przypadku ogólnym.

Bezpośrednie, dokładne kopiowanie w siebie danych uczących **x** nie ma sensu praktycznego. Nie jest możliwe bowiem wychwycenie najważniejszych cech procesu zawartych w zbiorze danych uczących. Dopiero ograniczenie liczby neuronów warstwy ukrytej definiującej kodowanie danych wejściowych zmusza proces uczenia do wychwycenia cech najważniejszych i pominięcia cech drugorzędnych, niemających większego wpływu na wynik rekonstrukcji danych. Proces uczenia sieci może być zdefiniowany jako dobór parametrów (wag neuronowych) prowadzący do minimalizacji funkcji kosztu $E=E(\mathbf{x},\mathbf{g}(\mathbf{h}(\mathbf{x}))$.

Podstawowa definicja funkcji kosztu przyjmowana jest zwykle w postaci wartości błędu średniokwadratowego po wszystkich zmiennych $n = 1, 2,\ldots,N$ dla danych uczących przy $k = 1, 2, \ldots, p$, przy czym N jest wymiarem wektora wejściowego, a p liczbą danych uczących (obserwacji).

$$E = \frac{1}{p}\sum_{n=1}^{N}\sum_{k=1}^{p}\left(x_n^{(k)} - x'^{(k)}_n\right)^2 = \frac{1}{p}\sum_{n=1}^{N}\sum_{k=1}^{p}\left(x_n^{(k)} - g_n(h(\mathbf{x}^{(k)})\right)^2 \tag{6}$$

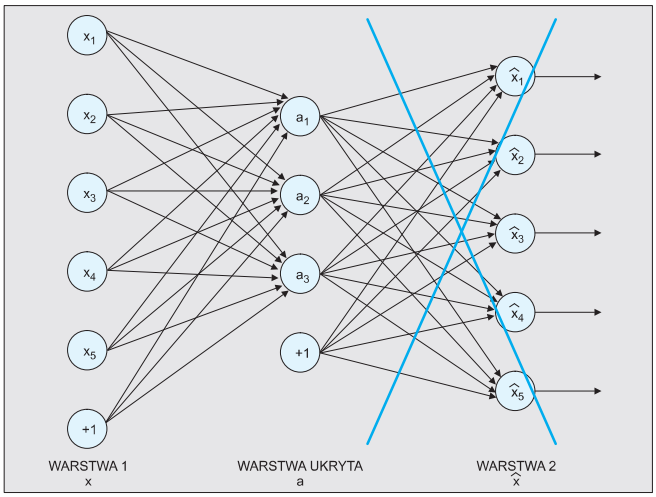
W celu uzyskania lepszej odporności autoenkodera na szum w uczeniu stosuje się również wersję zmodyfikowaną, w której zamiast dopasowywać sygnały wyjściowe **g**(**h**(**x**)) do sygnałów wejściowych **x** dopasowanie dotyczy odpowiedzi autoenkodera na wersje zaszumione sygnałów wejściowych $\mathbf{g}(\mathbf{h}(\mathbf{x}_s))$, gdzie $\mathbf{x}_s$ reprezentuje wersję zaszumioną **x.** Układ przetwarza wówczas zbiór wektorów $\mathbf{x}_s$ w taki sposób, aby uzyskać na wyjściu możliwie najlepszą aproksymację wektorów oryginalnych **x**.

Zwiększenie zdolności generalizacji autoenkodera może być osiągnięta w procesie uczenia przez włączenie do funkcji kosztu czynnika kary za nadmiernie rozbudowaną strukturę. Wprowadza się również regularyzację, uwzględniającą pochodne sygnałów

warstwy ukrytej względem elementów wektora wejściowego **x.** Taki rodzaj regularyzacji zmusza algorytm uczący do wytworzenia sygnałów wyjściowych sieci, które zmieniają się niewiele przy niewielkich zmianach wartości sygnałów wejściowych. Inne podejście do regularyzacji autoenkodera uwzględnia zgodność średniej statystycznej aktywacji neuronów $\hat{\rho}_j$ w aktualnym stanie uczenia z jej wartością pożądaną ρ_j. Mała wartość średniej aktywacji neuronu oznacza, że neuron ten aktywuje się jedynie dla niewielkiej ilości danych wejściowych, specjalizując się w ich reprezentacji. Dodając taki czynnik do minimalizowanej funkcji kosztu, zmusza się neurony do reprezentowania określonych (specyficznych) cech procesu.

W efekcie w procesie uczenia minimalizacji podlega funkcja kosztu zmodyfikowana o poszczególne rodzaje czynników regularyzacyjnych. Taka postać wymusza kompromis między zgodnością zrekonstruowanego sygnału wyjściowego układu z wartościami wejściowymi a złożonością tego układu, uwzględniając jednocześnie wrażliwość na niewielkie zmiany mogące nastąpić w rozkładzie wartości sygnałów wejściowych w trybie odtwarzania.

Istnieją różne implementacje algorytmu uczącego. Najbardziej popularne podejście polega na stopniowym uczeniu w następujących po sobie warstwach (rys. 6). Proces rozpoczyna się w warstwie pierwszej, dla której sygnałami wejściowymi są sygnały oryginalne opisane wektorem **x**, a sygnały wyjściowe oznaczone są jako $\hat{\mathbf{x}}$. Celem uczenia jest dobór liczby neuronów ukrytych i wag tych neuronów, które zapewnią właściwe (zgodnie z regularyzowaną funkcją celu) odtworzenie sygnałów wejściowych w warstwie wyjściowej. W tej sytuacji sygnały warstwy ukrytej stanowią kod sygnałów wejściowych.



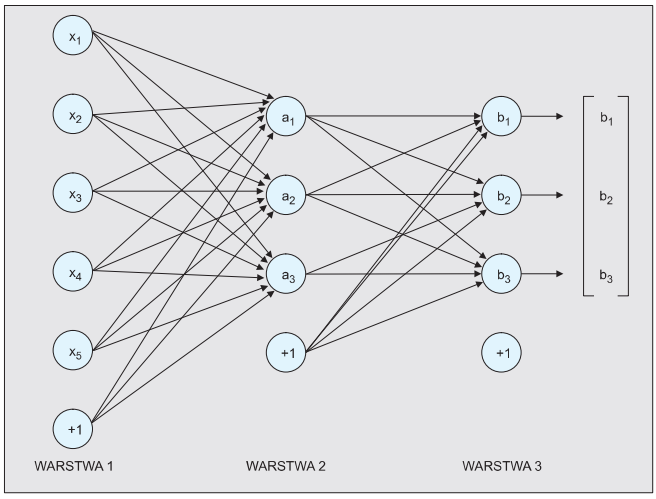
Rys. 6. Ilustracja sposobu uczenia pierwszej warstwy ukrytej

Po doborze parametrów warstwy ukrytej odrzuca się warstwę rekonstrukcji i sygnały z warstwy ukrytej stają się wejściowymi dla ukształtowania następnej warstwy **b** (rys. 7), trenowanej w sposób identyczny jak warstwa a.

Proces uczenia kontynuuje się dla kolejnych warstw, dobierając ich parametry w taki sposób, aby uzyskać możliwie najlepsze odwzorowanie sygnałów wejściowych w sygnały wyjściowe. Ostatnia warstwa ukryta autoenkodera zawiera kodowanie na najwyższym poziomie, zwykle o najbardziej ograniczonej liczbie neuronów (po wyeliminowaniu warstw rekonstrukcyjnych). Sygnały wyjściowe tej warstwy stanowią cechy diagnostyczne wyselekcjonowane automatycznie przez strukturę układu w procesie uczenia z udziałem zbioru danych uczących. Mogą one stanowić atrybuty wejściowe dla końcowego stopnia klasyfikatora bądź układu regresyjnego o dowolnej konstrukcji.

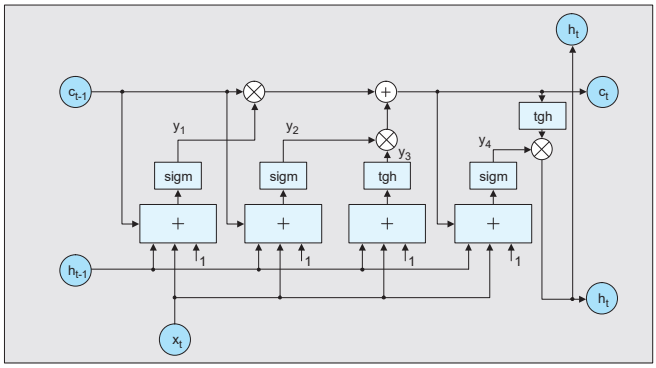
SIEĆ LSTM

Sieć **LSTM** (*Long Short-Term Memory*) jest siecią rekurencyjną charakteryzującą się długą pamięcią krótkich wzorców [4, 13,



Rys. 7. Kolejny etap uczenia następnej warstwy autoenkodera

14]. Jest znakomitym narzędziem do przetwarzania złożonych szeregów czasowych, w tym do przetwarzania mowy, generacji tekstu itp. Składa się z wielu rekurencyjnie połączonych bloków, zwanych blokami pamięciowymi. Każdy taki blok zawiera trzy bramki multiplikatywne: wejściową, wyjściową oraz pamięci, pełniące role sterowanych zaworów. Typową strukturę komórki bloku pamięciowego używanej w LSTM przedstawiono na rys. 8 [15].



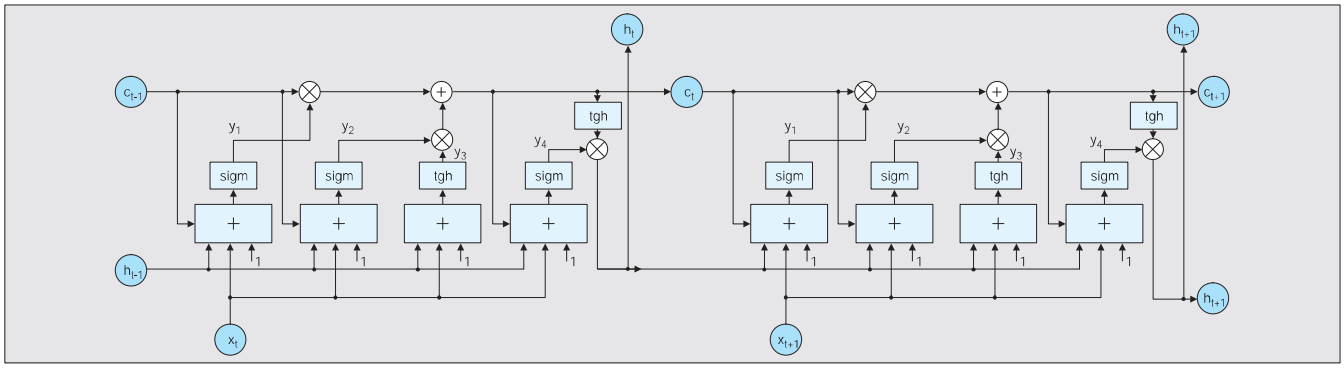
Rys. 8. Typowa struktura bloku pamięciowego sieci LSTM

Komórka pamięciowa zawiera trzy wejścia. Są to:
X_t – sygnał wejściowy w aktualnej chwili czasowej *t,*
C_{t-1} – sygnał pamięci z chwili *t*-1 poprzedniej komórki pamięci,
h_{t-1} – sygnał wyjściowy z chwili *t*-1 poprzedniej komórki pamięci.

Jednocześnie komórka ta wytwarza dwa rodzaje sygnałów wyjściowych istotnych z punktu widzenia współdziałania komórek. Są to:
C_t – sygnał pamięci w aktualnej chwili czasowej *t* danej komórki pamięci,
h_t – sygnał wyjściowy w aktualnej chwili czasowej *t* danej komórki pamięci.

Komórka przetwarza nieliniowo zarówno aktualny sygnał wejściowy podany na jej wejście, jak i sygnały wyjściowe oraz pamięci z chwili poprzedniej wszystkich bloków, z którymi jest sprzężona. Struktura przepływu sygnałów w czasie dla sieci LSTM złożonej z kilku komórek może być przedstawiona jak na rys. 9.

Istotną rolę w przetwarzaniu sygnałów odgrywają bramki multiplikatywne, oznaczone symbolem mnożenia ×. Górna bramka z lewej strony komórki pełni funkcję sterowanego zaworu pamięci. Decyduje o tym, jaka część pamięci z poprzedniej chwili czasowej przejdzie do chwili następnej. Stanowi zatem regulowany system zaworowy, który przepuszcza w określonym stopniu (od zera do 1) informację zawartą w pamięci z chwili poprzedniej. O poziomie przejścia tej zawartości decyduje sygnał wytworzony



Rys. 9. Model przepływu sygnałów w czasie dla sieci LSTM złożonej z trzech komórek pamięci [15]

przez warstwę neuronową sigmoidalną, widoczną poniżej z lewej strony modelu.

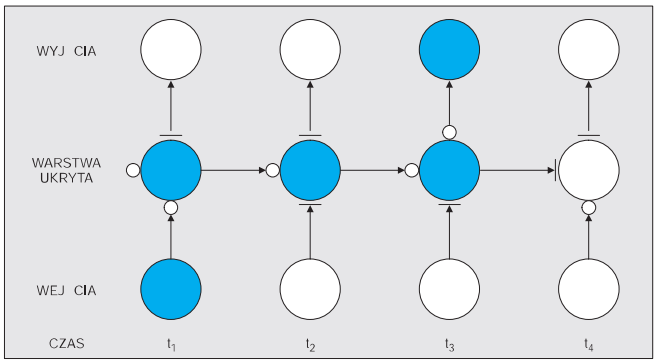
Druga bramka (symbol środkowy ×) jest następnym regulowanym zaworem przy tworzeniu nowej pamięci i uczestniczy w wytworzeniu zawartości pamięci C_t dla aktualnej chwili czasowej. Stanowi sterowane połączenie nowo wytworzonego stanu pamięci i pamięci poprzedniej C_{t-1}, która została przepuszczona przez zawór pierwszy. Połączenie tych informacji następuje w bloku sumatora. W efekcie takiej operacji stara zawartość pamięci C_{t-1} zostaje zamieniona na nową C_t.

Poszczególne zawory mają swoje wymuszenia i sygnały wyjściowe. Zawór pamięci jest sterowany przez jednowarstwową sieć neuronową o sigmoidalnej (jednopolarnej) funkcji aktywacji. Sygnały wejściowe dla tej warstwy stanowią: X_t – sygnał wejściowy w aktualnej chwili czasowej t, C_{t-1} – sygnał pamięci z chwili t-1 poprzedniej komórki pamięci, h_{t-1} – sygnał wyjściowy z chwili t-1 poprzedniej komórki pamięci oraz sygnał polaryzacji. Sumacyjny sygnał wyjściowy zaworu pamięci po przetworzeniu sigmoidalnym mnożony jest z sygnałem pamięci z chwili poprzedniej i stanowi dla niej współczynnik, z jakim poprzednia pamięć przechodzi do następnej chwili czasowej.

Drugi zawór, reprezentowany przez środkowy mnożnik ×, decyduje o nowym stanie pamięci C_t w chwili t i jest nazywany zaworem nowej pamięci. Składa się z dwu równolegle działających warstw neuronowych. Pierwsza warstwa sigmoidalna jednopolarna przyjmuje identyczne wymuszenia jak warstwa neuronowa w zaworze pamięci (X_t, C_{t-1}, h_{t-1} i polaryzacja) oraz stanowi przetworzoną zawartość starej pamięci. Druga warstwa operuje jedynie sygnałami X_t, h_{t-1} oraz polaryzacją, a więc nie zależy od zawartości starej pamięci. Funkcja aktywacji tej warstwy jest sigmoidą bipolarną, realizowaną przez funkcję tangensa hiperbolicznego. Mnożnik decyduje o wpływie starej pamięci na stan wyjściowy tego zaworu, który będzie następnie dodany do zawartości starej pamięci, wychodzącej z bloku pamięci (symbol sumatora), tworząc aktualny (w chwili t) stan pamięci C_t.

Ostatnią operacją komórki pamięciowej jest wytworzenie sygnału wyjściowego h_t w aktualnej chwili czasowej t. Sygnał wyjściowy komórki jest iloczynem dwu sygnałów: aktualnego sygnału pamięci C_t przepuszczonego przez funkcję sigmoidalną bipolarną oraz sygnału wyjściowego z warstwy neuronowej zasilonej poprzez X_t, h_{t-1} oraz polaryzację. System ten stanowi zawór wyjściowy komórki, decydujący o tym, jak duża porcja nowej pamięci zostanie przekazana na wyjście komórki, czyli sygnału komórki, który zasili komórkę w następnej chwili czasowej.

W efekcie wysterowania poszczególnych zaworów możliwe jest przepuszczenie lub zablokowanie określonej porcji informacji z chwil poprzednich. Na rys. 10 zilustrowano możliwy wpływ informacji dostarczonej do sieci w chwili $t = 1$ na stan komórki w następnych chwilach czasowych [13]. Symbol „o" oznacza pełne otwarcie zaworu, a symbol „–„ pełne zamknięcie. Stan wyjścia komórek w poszczególnych chwilach czasowych



Rys. 10. Ilustracja przepływu informacji z chwili $t = 1$ do wyjścia w kolejnych chwilach czasowych w zależności od stanu wysterowania zaworów

w zależności od informacji z chwili $t = 1$ jest funkcją otwarcia lub zamknięcia odpowiednich zaworów i jest ilustrowany ciemnym stanem węzła (informacja przechodzi) lub jasnym (informacja zablokowana).

W procesie uczenia doborowi podlegają wagi poszczególnych połączeń sieci przez minimalizację funkcji błędu na danych uczących w postaci par (wielkość wejściowa **x** i wielkość zadana na wyjściu **d**), określonych w kolejnych chwilach czasowych. Minimalizacja odbywa się metodą gradientową przy generacji gradientu przez zastosowanie propagacji wstecznej. Sygnały wejściowe przesyłane są najpierw wprzód (w obrębie przyjętego okna czasowego T), a następnie błąd na wyjściu w poszczególnych chwilach czasowych przesyłany jest w kierunku odwrotnym (od wyjścia do wejścia). Na tej podstawie generowane są składniki gradientu względem poszczególnych wag sieci, podobnie jak odbywało się to w klasycznych sieciach neuronowych.

Przyjmijmy oznaczenia wskaźnikowe jak na rys. 8. Poszczególne warstwy neuronowe charakteryzowane są przez macierze wagowe opisane za pomocą dwu wskaźników: pierwszy pochodzi od numeru przypisanego warstwie, drugi od symbolu wielkości wejściowej dla tej warstwy. Przykładowo $\mathbf{W}_{1h}$ oznacza macierz wagową pierwszego zaworu, przetwarzającą sygnały wyjściowe komórki $\mathbf{h}_{t-1}$ z poprzedniej chwili wagowej. Funkcja sigmoidalna oznaczona jest symbolem sigm. Wektory sygnałowe w poszczególnych punktach komórki są wyrażone wzorami [13]:

$$y_1 = sigm(\mathbf{W}_{1c}\mathbf{c}_{t-1} + \mathbf{W}_{1h}\mathbf{h}_{t-1} + \mathbf{W}_{1x}x_t + \mathbf{w}_{10} \quad (7)$$

$$y_2 = sigm(\mathbf{W}_{2c}\mathbf{c}_{t-1} + \mathbf{W}_{2h}\mathbf{h}_{t-1} + \mathbf{W}_{2x}x_t + \mathbf{w}_{20} \quad (8)$$

$$y_3 = tgh(\mathbf{W}_{3h}\mathbf{h}_{t-1} + \mathbf{W}_{3x}x_t + \mathbf{w}_{30} \quad (9)$$

$$y_4 = sigm(\mathbf{W}_{4c}\mathbf{c}_t + \mathbf{W}_{4h}\mathbf{h}_{t-1} + \mathbf{W}_{4x}\mathbf{x}_t + \mathbf{w}_{40} \quad (10)$$

Na tej podstawie tworzone są: stan pamięciowy komórki c_t oraz sygnał wyjściowy komórki h_t (oba w chwili aktualnej t). Są one określone wzorami:

$$\mathbf{c}_t = \mathbf{y}_2 \cdot \mathbf{y}_3 + \mathbf{c}_{t-1}\,\mathbf{y}_1 \quad (11)$$

$$\mathbf{h}_t = \mathbf{y}_4 \cdot tgh(\mathbf{c}_t) \quad (12)$$

Przetwarzanie wsteczne sygnałów odbywa się w identyczny sposób, jak w metodzie klasycznej propagacji wstecznej (odwrócenie kierunku przepływu przy tych samych wartościach wag gałęzi liniowej i zastąpieniu funkcji aktywacji przez wartość pochodnej w punkcie pracy sieci o normalnym kierunku przepływu). W przypadku węzła realizującego mnożenie korzysta się z zależności pochodnej iloczynu, zgodnie z którą, jeśli sygnał y jest iloczynem dwu sygnałów v_1 i v_2, to:

$$\frac{dy}{dw} = v_1 \frac{dv_2}{dw} + v_2 \frac{dv_1}{dw}. \quad (13)$$

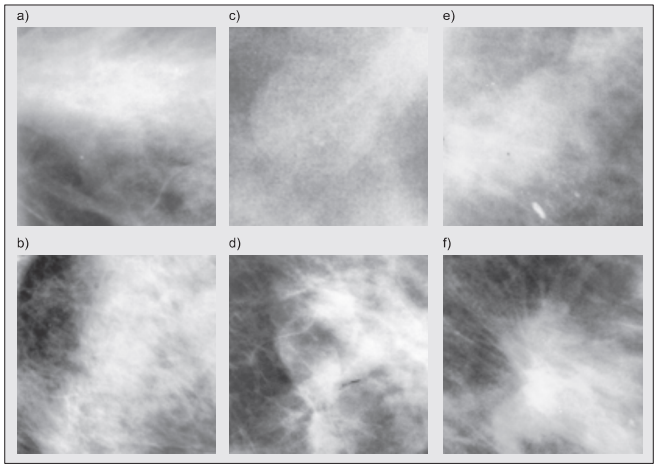
Pełna informacja o gradiencie jest sumą pochodnych po wszystkich krokach czasowych wykonanych wewnątrz okna pomiarowego T. W uczeniu wykorzystuje się z reguły metodę największego spadku z momentem rozpędowym.

PRZYKŁADY ZASTOSOWAŃ

Uczenie głębokie znalazło wiele zastosowań w eksploracji danych. Można je zaliczyć bądź do zadania klasyfikacji (wielkości zadane w formie binarnej) bądź do regresji (wielkości zadane w postaci liczb rzeczywistych). Każdy rodzaj zadania znalazł swoją aplikację w rozwiązywaniu różnorodnych zadań z dziedziny inżynierii, ekonomii, finansów, automatycznego przetwarzania głosu, tekstu itp. Szczególne znaczenie odgrywają sieci głębokie w inżynierii biomedycznej, wspomagając aktywnie diagnostykę medyczną w tak trudnych zadaniach, jak rozpoznawanie i segmentacja obrazów biomedycznych, zarówno mikroskopowych, jak i histologicznych.

Pierwszy przykład zastosowania CNN będzie dotyczyć trudnego problemu rozpoznania obrazów mammograficznych piersi, w szczególności odróżnienia przypadku zdrowego od nowotworu, jak również odróżnienia nowotworów złośliwych od nowotworów łagodnych. Badania zostały przeprowadzone na bazie danych *Digital Database for Screening Mammography* (**DDSM**) przygotowanej na Uniwersytecie Floryda [17]. Baza ta zawiera obrazy przedstawiające rejony zainteresowań lekarzy (ROI) o wymiarze 128×128, podlegające podejrzeniu wystąpienia zmian nowotworowych. Skład bazy danych zawierał: 8254 obrazy tkanek zdrowych, 862 obrazy ROI opisane przez lekarzy jako zmiany łagodne i 1052 obrazy ROI reprezentujące nowotwory złośliwe.

Baza danych jest więc mocno niezbilansowana pod względem udziału poszczególnych klas, stąd trudna w przetwarzaniu. Druga trudność wynika z dużego podobieństwa reprezentantów różnych klas przy jednoczesnym znacznym zróżnicowaniu obrazów tworzących jedną klasę. Ilustrują to przykłady obrazów reprezentujących wszystkie trzy klasy przedstawione na rys. 11.

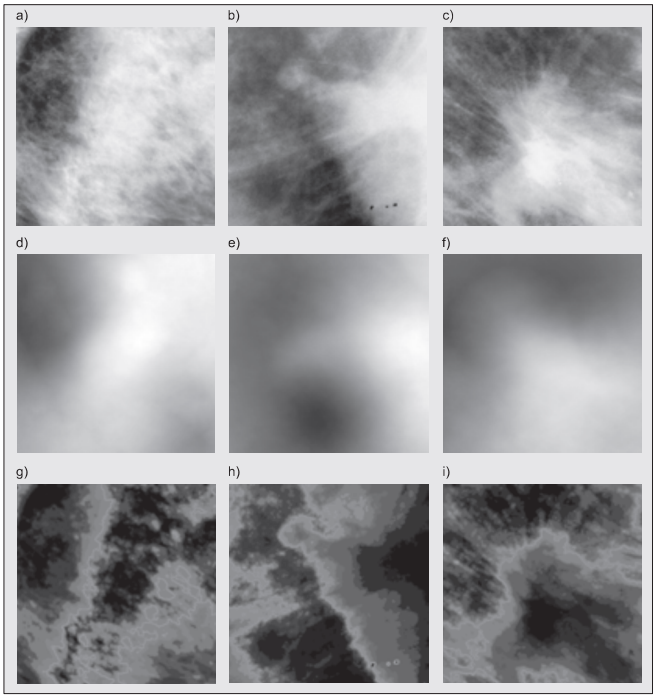


Rys. 11. Przykłady mammogramów (ROI) reprezentujących przypadki: zdrowy (a,b), nowotwór łagodny (c,d) i nowotwór złośliwy (e,f)

Dodatkowym problemem jest stosunkowo mała liczba obrazów w bazie DDSM w stosunku do liczby wag sieci CNN podlegających adaptacji. Aby zwiększyć różnice międzyklasowe w sposób efektywny (nie powielając danych oryginalnych), zastosowano dodatkowo dwie różne transformacje obrazów [18].

Pierwsza polegała na użyciu nieujemnej faktoryzacji macierzy (NMF) [19]. Część obrazów reprezentujących przypadki zdrowe została zdekomponowana na dwie macierze składowe o nieujemnych wartościach i użyta jako podstawa transformacji pozostałych danych (zarówno zdrowych, jak i nowotworowych). W ten sposób obrazy reprezentujące nowotwór podlegały większym zmianom niż przypadki normalne. Jednocześnie tak zmodyfikowana baza zwiększyła liczbę danych uczestniczących w uczeniu.

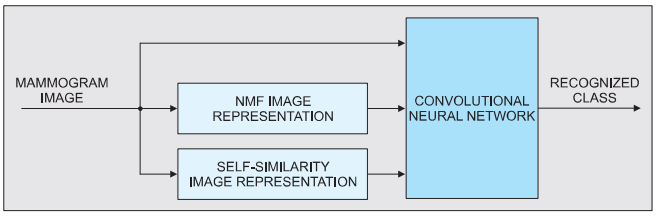
W drugiej metodzie tworzenia dodatkowych obrazów wykorzystano pojęcie statystycznego samopodobieństwa, rozumianego jako stopień podobieństwa fragmentów obrazu do obrazu pełnego [18]. Oryginalne obrazy ROI przekształcone do wymiaru 1024×1024 podzielono na podobszary 5×5. Dla każdego z tych podobszarów obliczano odległość Kołmogorowa-Smirnowa (KS) od całego obrazu. W ten sposób każdy mały rejon obrazu był reprezentowany przez względną miarę KS o nasyceniu pikseli z przedziału (0, 1) tworząc nową reprezentację przetransformowaną do oryginalnego wymiaru 128×128. W ten sposób powstała nowa rodzina reprezentacji obrazów oryginalnych wzbogacona o informacje statystyczne. Jednocześnie obie zastosowane transformacje zwiększyły bazę danych podlegających analizie. Przykłady obrazów reprezentujących trzy klasy odpowiadające danym oryginalnym (wiersz górny), przekształconym przez NMF (wiersz środkowy) i przez samopodobieństwo (wiersz dolny) przedstawiono na rys. 12.



Rys. 12. Ilustracja wpływu zastosowanych transformacji na wygląd obrazów ROI. Kolumna z lewej strony reprezentuje przypadek normalny, kolumna środkowa przypadek nowotworu łagodnego, kolumna z prawej strony przypadek nowotworu złośliwego. Wiersz górny przedstawia obrazy oryginalne, środkowy – obrazy utworzone na podstawie NMF, wiersz dolny – obrazy samopodobieństwa

Suma tych obrazów tworzy potrojoną bazę danych użytą w eksperymentach numerycznych uczenia i walidacji sieci CNN (rys. 13) [18].

Sieć CNN tworzona była od podstaw z zastosowaniem programu w Matlabie. Ostateczna struktura sieci, zdefiniowana po wielu eksperymentach wstępnych, zawierała trzy warstwy konwolucyjne:



Rys. 13. Struktura zadawania danych na sieć CNN [18]

- pierwsza warstwa złożona z 32 neuronów, pole recepcyjne filtru 5×5, *stride* 1×1, *zero-padding* 2×2, ReLU i *max pooling*,
- druga warstwa złożona z 32 neuronów, pole recepcyjne filtru 5×5, *stride* 1×1, *zero-padding* 2×2, ReLU i *max pooling*,
- trzecia warstwa zawierała 64 neurony, pole recepcyjne filtru 5×5, *stride* 1×1, *zero-padding* 2×2, ReLU i *average pooling*

oraz dwie warstwy w pełni połączone: pierwsza o 64 neuronach i druga zawierająca dwa neurony reprezentujące dwie rozpoznawane klasy obrazów.

Sieć podlegała uczeniu metodą propagacji wstecznej z zastosowaniem tzw. *mini-batch*, czyli losowego doboru niewielkiej grupy danych uczących w poszczególnych krokach uczenia. W tabeli 1 przedstawiono wyniki rozpoznania w dwu przypadkach: klasa przypadków zdrowych względem chorych oraz klasa reprezentująca nowotwór złośliwy względem klasy nowotworów łagodnych. W tabeli podano wartość powierzchni AUC pod krzywą ROC, czułość rozpoznania klasy, specyficzność oraz dokładność. Wszystkie dane odnoszą się do danych walidacyjnych nieuczestniczących w uczeniu, uzyskanych w trybie 10-krotnej walidacji krzyżowej (wartości średnie z 10 prób walidacji).

Tabela 1. Wyniki analizy bazy mammograficznej dla dwu przypadków rozpoznania klas

	AUC	Czułość	Specyficzność	Dokładność
Klasa przypadków nowotworowych względem normalnych	0,919	82,28%	86,59%	85,82%
Klasa nowotworów złośliwych względem łagodnych	0,909	73,36%	86,01%	84,75%

Uzyskane wyniki należą do grupy najlepszych dla tej bazy danych i są o kilka punktów procentowych lepsze od zastosowania sieci klasycznych przy użyciu cech diagnostycznych definiowanych w sposób manualny przez użytkownika.

Inny przykład wykazujący wyższość metod głębokich nad klasycznymi dotyczy rozpoznawania obrazu twarzy [20]. Obrazy twarzy 50 osób płci obojga przy różnym jej ustawieniu i oświetleniu były zarejestrowane przy użyciu światła widzialnego oraz podczerwieni. Każda osoba była reprezentowana przez 20 obrazów różniących się znacznie od siebie. Przykład tych różnic pokazano na rys. 14 [20].

Tym razem zastosowano technikę *transfer learning* z użyciem programu ALEXNET Matlaba [8]. Sieć ta stosuje strukturę CNN wstępnie nauczoną na zbiorze ponad miliona obrazów zaczerpniętych z Internetu. Zadaniem użytkownika jest dopasować działanie tej sieci do własnego zadania. Osiąga się to przez adaptację końcowej struktury stanowiącej klasyfikator typu *softnet*. Sieć klasyfikatora w pełni połączonego składa się z 4096 neuronów warstwy pierwszej, 2500 neuronów warstwy drugiej i 50 neuronów warstwy wyjściowej *softnetu*, reprezentujących 50 klas rozpoznawanych obrazów twarzy. Poniżej przedstawiono szczegółową strukturę CNN według standardów Matlaba.

Image Input:227x227x3
Convolution1: 96 11x11x3 convolutions with stride [4 4] and zero-padding [0 0]
ReLU
Cross Channel Normalization with 5 channels per element
Max Pooling: 3x3 max pooling with stride [2 2] and zero-padding [0 0]
Convolution2: 256 5x5x48 convolutions with stride [1 1] and zero-padding [2 2]
ReLU
Cross Channel Normalization with 5 channels per element
Max Pooling: 3x3 max pooling with stride [2 2] and zero-padding [0 0]
Convolution3: 384 3x3x256 convolutions with stride [1 1] and zero-padding [1 1]
ReLU
Convolution4: 384 3x3x192 convolutions with stride [1 1] and zero-padding [1 1]
ReLU
Convolution5: 256 3x3x192 convolutions with stride [1 1] and zero-padding [1 1]
ReLU
Max Pooling: 3x3 max pooling with stride [2 2] and zero-padding [0 0]
Fully Connected Layer: 4096 elements of the vector fully connected to next layer
ReLU
Dropout: 50%
Fully Connected Layer: 2500 fully connected layer
ReLU
Dropout 50%
Fully Connected Layer: 50 fully connected neurons
Softmax classifier
Output layer: 50 neurons representing 50 classes

Sieć zawiera 5 warstw konwolucyjnych współpracujących z ReLU, normalizacją i *Max pooling*. Opis warstwy konwolucyjnej w notacji Matlaba (np. 96 11x11x3 dla warstwy wejściowej) zawiera w kolejności: liczbę neuronów (obrazów wynikowych w warstwie – 96), wielkość pola recepcyjnego neuronu filtrującego (11x11) i liczbę obrazów analizowanych jednocześnie przez neurony (np. 3 dla warstwy wejściowej). Zastosowana operacja *Max pooling* o wymiarze pola recepcyjnego 3x3 redukuje więc informację przekazywaną z tego pola do następnej warstwy w stosunku 1 : 9. Uczenie takiej struktury sieciowej jest stosunkowo szybkie. Przy zastosowaniu procesora graficznego proces uczenia trwał zaledwie kilka minut.

Rys. 14. Przykłady obrazów twarzy jednej osoby użytej w eksperymentach numerycznych. Wiersz górny – rejestracja w świetle widzialnym, wiersz dolny – rejestracja w podczerwieni [20]

Wyniki rozpoznania 50 klas reprezentujących poszczególne osoby przy zastosowaniu sieci CNN zostały porównane z klasycznym podejściem do problemu przy zastosowaniu metod przetwarzania obrazów uznanych w świecie za skuteczne (PCA, KPCA, tSNE) [20] i przedstawione w tabeli 2. Widocz-

Tabela 2. Wyniki rozpoznania obrazów twarzy 50 osób przy zastosowaniu sieci głębokiej CNN i klasycznego podejścia [20] do rozpoznania twarzy przy zastosowaniu różnych technik generacji cech diagnostycznych (PCA, KPCA i tSNE). Dane przedstawiają średnią dokładność rozpoznania klasy ± standardowe odchylenie uzyskane w 10 próbach testowania systemu [20]

	CNN [%]	PCA [%]	KPCA [%]	tSNE [%]
Światło widzialne	4,42±1,35	13,30±1,5	12,96±1.4	15,84 ±1,6
Podczerwień	5,89+/-1,84	14,21+/-1,7	14,32±1.8	16,14±1,9

na jest zdecydowana przewaga sieci CNN. Średnia wartość czułości rozpoznania klas w przypadku światła widzialnego, osiągnięta przez sieć CNN, była równa 97,33% i 94,23% dla podczerwieni.

Sieci neuronowe znalazły szerokie zastosowanie w przetwarzaniu i analizie mowy. Typowe jest zastosowanie krótkookresowej transformacji Fouriera (STFT) przedstawiającej sygnał mowy jako funkcję jednocześnie czasu i częstotliwości w postaci obrazu spektrogramu. Spektrogram zmienia się wraz z kolejnymi frazami wypowiedzi. Głęboka sieć neuronowa w formie LSTM jest w stanie rozpoznać i zapamiętać sekwencje tych obrazów i przekształcić je do postaci konkretnych słów. Na bazie LSTM opracowywane są nowoczesne rozwiązania tłumaczy tekstów językowych. Translacja jest możliwa bez specjalnego *preprocessingu* tekstu. Sieć uczy się zależności występujących między kolejnymi słowami i transformuje na inny język. Opracowany ostatnio system **GNMT** (*Google Neural Machine Translation*) stosuje sieci rekurencyjne LSTM do tłumaczenia całych zdań z jednego języka na drugi.

Ogromnym polem zastosowań uczenia głębokiego staje się przetwarzanie języka naturalnego. Występuje tu problem zrozumienia języka lub jego generacji. Zrozumienie języka polega na wyciągnięciu znaczenia wypowiedzianych sekwencji wyrazów lub zdań. Zadaniem systemu jest nauczenie się zrozumienia wypowiedzi, podobnie jak to czyni człowiek. Z kolei generacja języka będzie odpowiedzią systemu automatycznego na wypowiedziany tekst odpowiednio do zrozumienia jego treści.

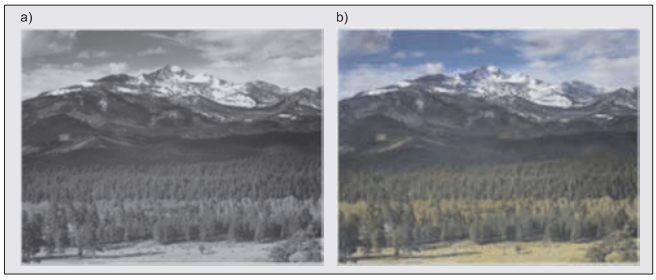
Ciekawym polem zastosowań głębokiego uczenia jest generacja nowego tekstu na podstawie analizy wielu tekstów zaczerpniętych z różnych źródeł. W przypadku stosowania sieci rekurencyjnej LSTM zapamiętywane są określone sekwencje wyrazów, tworzących zdania. Na tej podstawie możliwe jest utworzenie nowego zwartego tekstu, spójnego pod względem treści. Wiele przykładów takiego procesu z wykorzystaniem pakietu *Tensorflow* można obecnie znaleźć w Internecie.

Głębokie uczenie znalazło wiele zastosowań w problemach inwersyjnych, do których można zaliczyć odtwarzanie obrazów na podstawie znanych fragmentów, rekonstrukcję obrazów na podstawie jego zakodowanego modelu, optymalne pod względem zniekształceń zwiększanie rozdzielczości obrazów, redukcję i eliminację szumu z danych pomiarowych itp.

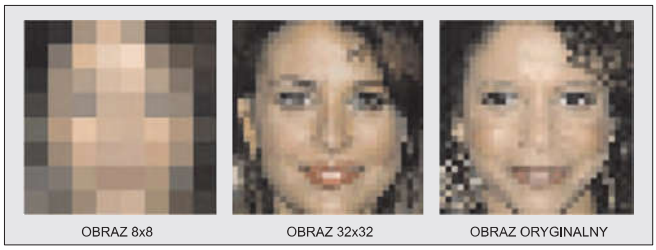
Interesującym polem zastosowań głębokiego uczenia jest przemysł filmowy. Sieci te używane są w procesie kolorowania filmów czarno-białych, w automatycznym dodawaniu dźwięków do filmów niemych itp. W tym ostatnim przypadku zadaniem sieci jest stowarzyszenie ramek wideo z odpowiadającymi im nagraniami dźwiękowymi przygotowanymi w bazie danych.

Przykład efektu zamiany obrazu reprezentowanego w skali szarości na obraz kolorowy przedstawiono na rys. 15 [21]. Efekt ten uzyskano stosując rozbudowaną strukturę sieciową składającą się z kilku segmentów: układu generującego cechy niskiego, średniego i globalnego poziomu, jak również układu nakładającego kolory na poszczególne fragmenty obrazu.

Google opracował technikę rekonstrukcji obrazu niskiej rozdzielczości w obraz o rozdzielczości wyższej, stosując odpowiednią strukturę głębokiej sieci PixelCNN. Metoda nosi nazwę *Pixel Recursive Super Resolution* i umożliwia odtworzenie obrazu bardzo niskiej rozdzielczości w obraz o rozdzielczości satysfakcjonującej użytkownika. Przykład wyniku działania programu rekonstruującego obraz 32x32 na podstawie prototypu 8x8 pokazano na rys. 16 [21].



Rys. 15. Przykład wyniku kolorowania obrazu oryginalnego danego w skali szarości przy zastosowaniu uczenia głębokiego [21]



Rys. 16. Przykład wyników rekonstrukcji obrazu twarzy o niskiej rozdzielczości 8x8 (obraz z lewej strony) w obraz o wyższej rozdzielczości 32x32 (środkowy). Z prawej strony pokazany jest obraz oryginalny, z którego utworzony został obraz 8x8 [21]

Rys. 17. Przykład naśladownictwa stylu obrazu oryginalnego (a) przeniesionego na obraz stworzony przez sieć głęboką (b) [22]

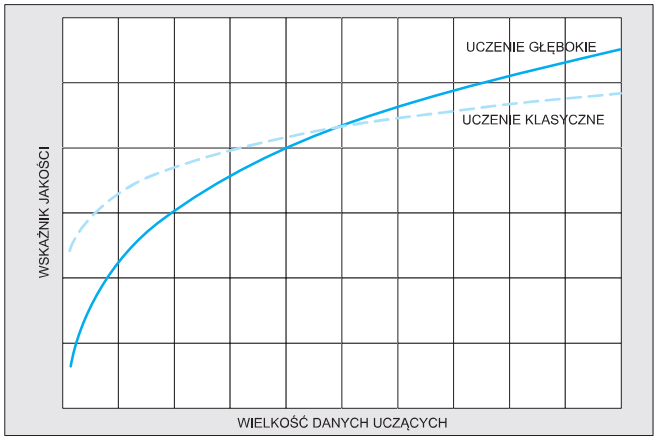
Sieci głębokie mogą przejawiać własności twórcze w dziedzinie sztuki. Na podstawie zbioru uczącego, zawierającego wiele arcydzieł malarstwa, są w stanie stworzyć własne obrazy charakteryzujące się określonym stylem zaczerpniętym z prototypu oryginalnego. Przykład wyniku takiej operacji przedstawiono na rys. 17.

Ogromne pole zastosowań uczenia głębokiego powstało w dziedzinie gier komputerowych, sterowania robotami czy opracowaniem samosterujących się samochodów. W tych dziedzinach sieci głębokie są w stanie przewyższyć zdolności ludzkie. W większości gier komputerowych sieci typu *DeepLearning* uzyskiwały lepsze wyniki niż najbardziej doświadczeni użytkownicy tych gier [22].

* * *

Głębokie sieci neuronowe i związane z tym głębokie uczenie stworzyły nowe perspektywy rozwoju sztucznej inteligencji. Dzięki mechanizmom samoorganizacji zarówno w sensie generacji cech diagnostycznych, jak i zastosowania końcowej jednostki wykonawczej (klasyfikatora lub układu regresyjnego), sieci są w stanie przetwarzać dane oryginalne bez potrzeby wstępnej analizy eksperckiej. Mają lepsze zdolności uogólniania wiedzy nabytej w uczeniu niż sieci klasyczne. Zastosowanie odpowiednio dużych zbiorów danych uczących umożliwia uzyskanie bardzo dobrej zdolności działania na danych nieuczestniczących w ucze-

niu. W wyniku wielu eksperymentów stwierdzono wyższość pod tym względem sieci głębokich nad rozwiązaniami klasycznymi. W sieciach klasycznych obserwuje się pewne ograniczenie stopnia generalizacji, którego nie obserwuje się w sieciach głębokich. Typowe porównanie jakości uczenia uzależnione od dostępnego wolumenu danych uczących przedstawiono na rys. 18.



Rys. 18. Porównanie zdolności generalizacyjnych sieci głębokich i klasycznych w zależności od wielkości zbioru danych uczących

Sieci głębokie stanowią narzędzie przetwarzania danych stosunkowo łatwe w zastosowaniach praktycznych. Dzięki zaimplementowanej strategii automatycznej generacji cech diagnostycznych, są stosowane do rozwiązania wielu różniących się problemów. Wprawdzie olbrzymia liczba adaptowanych parametrów sieci wymaga zastosowania bardzo dużych zbiorów uczących, jednak w praktyce użytkownik adaptuje tylko wagi połączeń ostatnich kilku warstw, co znacznie przyspiesza proces uczenia i redukuje wymagania dotyczące bazy uczącej, umożliwiając jednocześnie uzyskanie dobrych zdolności generalizacyjnych.

LITERATURA

[1] Fukushima K.: "Neocognitron – a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". *Biological Cybernetics* 1980, vol.. 36, No 4, pp. 193–202. doi:10.1007/bf00344251.

[2] LeCun Y., Y. Bengio: "Convolutional networks for images, speech, and time-series". 1995, in Arbib M. A. (editor), *The Handbook of Brain Theory and Neural Networks*. MIT Press, Massachusetts.

[3] Goodfellow I., Y. Bengio, A. Courville : *Deep learning* 2016, MIT Press, Massachusetts.

[4] Schmidhuber J.: *Deep learning in neural networks: An overview*. Neural Networks 2015, vol. 61, pp. 85-117.

[5] Hinton G. E, S. Osindero, and Y. W. Teh,: *A fast learning algorithm for deep belief nets*. Neural Computation 2006, vol. 18, pp. 1527-1554.

[6] Lecture CS231n. *CS231n: Convolutional Neural*. 2017, Stanford Vision Lab, Stanford University.

[7] Krizhevsky A., I. Sutskever, G. Hinton: *Image net classification with deep convolutional neural networks*. Advances in Neural Information Processing Systems 2012, vol. 25, pp. 1-9.

[8] Matlab 2017b, Math Works, Natick, USA, 2017.

[9] Zeiler M. D., R. Fergus: *Visualizing and Understanding Convolutional Networks*. 2013, pp. 1-11, https://arxiv.org/abs/1311.2901.

[10] Simonyan K., A. Zisserman: *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014, pp. 1-14, arXiv:1409.1556.

[11] He K., X. Zhang, S. Ren, J. Sun: *Deep Residual Learning for Image Recognition*. 2015, http://arxiv.org/abs/1512.03385.

[12] Ronneberger O., P. Fischer, T. Brox: *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015, arXiv:1505.04597.

[13] Graves A.: *Supervised sequence labelling with recurrent neural networks*. 2016 (preprint).

[14] Greff K., R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber: „LSTM: A search space odyssey". *IEEE Transactions on Neural Networks and Learning Systems 2017*, vol. 28, No 10, pp. 2222-2232.

[15] Shi Y.: *Understanding LSTM and its diagrams*. 2016, https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e...

[16] https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet.

[17] Heath D. K. M., K.W. Bowyer: "Current status of the digital database for screening mammography". 1998, *Proceedings of the Fourth International Workshop on Digital Mammography*, Kluwer Academic Publishers, Amsterdam, pp. 457–460.

[18] Świderski B., J. Kurek, S. Osowski, M. Kruk, W. Barhoumi.: "Deep learning and non-negative matrix factorization in recognition of mammograms". Proc. SPIE10225B 2017, Eighth Int. Conf. Graphic and Image Processing, 2017; doi:10.1117/12.2266335, http://dx.doi.org/10.1117/12.2266335.

[19] Cichocki A., R. Zdunek, A. H. Phan, S. I. Amari: *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. 2009 Wiley, New York.

[20] Siwek K., S. Osowski: "Deep neural networks and classical approach to face recognition – comparative analysis". *Przegląd Elektrotechniczny* 2018, R. 94, No 4, pp. 1-4.

[21] http://www.yaronhadad.com/deep-learning-most-amazing-applications/

[22] http://hi.cs.waseda.ac.jp/~iizuka/projects/colorization/en/

Portal Informacji Technicznej

www.sigma-not.pl

największa baza publikacji technicznych on-line