Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

INSTYTUT ELEKTROTECHNIKI TEORETYCZNEJ I SYSTEMÓW
INFORMACYJNO-POMIAROWYCH
ZAKŁAD ELEKTROTECHNIKI TEORETYCZNEJ I INFORMATYKI STOSOWANEJ

Praca dyplomowa inżynierska

na kierunku Automatyka i Robotyka

Metody prognozowania obciążeń elektroenergetycznych przy wykorzystaniu sieci CNN

Maria Ogryczak
Nr albumu 270774

promotor
dr hab. inż. Krzysztof Siwek

WARSZAWA 2018

Tytuł pracy: Metody prognozowania obciążeń elektroenergetycznych przy wykorzystaniu sieci CNN.

W niniejszej pracy wykonano badania dotyczące predykcji obciążeń elektroenergetycznych w Krajowym Systemie Elektroenergetycznym (KSE) wykorzystując do tego celu konwolucyjne sieci neuronowe (CNN).

Pierwszym etapem było przygotowanie sygnału tak, aby mógł posłużyć za wejście do sieci. Do uzyskania sygnału w postaci 2D wykorzystano trzy metody: spektrogram, transformację falkową oraz budowanie macierzy z fragmentów wektora (sześć różnych wariantów).

Następnie opracowano dwanaście konwolucyjnych sieci neuronowych, w których testowano zarówno wpływ ilości warstw konwolucyjnych, liczby i wielkości filtrów, jak i działania dodatkowych operacji takich jak dropout i maxpooling. Specyfika tych sieci polega na użyciu operacji splotu (konwolucji), która pozwala na wyznaczenie odpowiednich cech z obrazu w sposób automatyczny. Służą one potem za wejście do tradycyjnej sieci neuronowej.

Po przygotowaniu wszelkich niezbędnych elementów rozpoczęto badania eksperymentalne. Polegały na nauczeniu sici neuronowej przewidywania obciążeń elektroenergetycznych na dany dzień mając do dyspozycji dane z określonej liczby dni wstecz. Wyjściem sieci były dwadzieścia czery wartości chwilowych obciążeń na dany dzień. Następnie liczono błąd średni, maksymalny oraz absolutny i na ich podstawie określano jakość wyników otrzymanych korzystając z danej metody i sieci.

Najlepszą kombinacją okazała się być sieć prosta, złożona z tylko jednej warstwy konwolucyjnej i przetworzenie sygnału za pomocą spektrogramu. Wyniki były wyraźnie lepsze niż w pozostałych przypadkach, lecz inne metody również dawały dobre wyniki. Zauważalny wpływ na wyniki miała ilość dni wstecz branych pod uwagę przy dokonywaniu prognozy. Dla najlepszych wariantów wartości błędów średnich były rzędu 2-3%.

Po przeanalizowaniu wyników badań eksperymentalnych porównano przewidywania uzyskane najlepszą metodą dla wybranego dnia z rzeczywistymi wartościami obciążeń. Otrzymane prognozy były bardzo dobre, spełniające oczekiwania. Należy jednak zauważyć, że nie mogą one służyć przy rozwiązywaniu problemów technicznych, ponieważ pobrane dane nie były wystarczająco dokładne.

Ze względu na to, że macierze wejściowe miały małe rozmiary oraz ich ilość nie była zbyt duża, zastosowanie dwóch warstw konwolucyjnych, operacji dropout i maxpooling nie przyniosło porządanych efektów. Spodziewano się, że znacząco poprawią one wyniki, lecz w efekcie uległy one pogorszeniu – zostało wyciągnięte wiele cech, które były zbyt szczegółowe.

Znaczącym elementem badań okazał się czas potrzebny na nauczenie poszczególnych sieci. Im bardziej były one skomplikowane, tym więcej czasu było potrzebne. Jednak odczuwalnie możnaby go skrócić stosując komputer o znacznie lepszych parametrach, wyposażony w procesor graficzny (GPU).

Słowa kluczowe: konwolucyjne sieci neuronowe, predykcja, obciążenia elektroenergetyczne, zmiana postaci sygnału z 1D na 2D, sektrogram, transformacja falkowa

Title of the thesis: Predicting methods of electric power load using CNN networks.

In the following thesis the research pertains to prediction of electric power load in country electric power system (KSE) were made. To achieve this goal the convolutional neural networks were used.

The first step was to prepare signal to be used as an input of the network. To get a signal in 2D form three methods were used: spectrogram, wave transform and building matrix from vector extracts (six different variants).

Afterwards twelve convolutional neural networks were mapped out, in which influence of number of convolutional layers, quantity and size of filters were tested. Moreover, the action of additional operations such as dropout and maxpooling was observed. Profile of this networks is use of convolution, which enables appointing appropriate features of image in an automatic way, which later serve as an input to the traditional neural network.

After preparing all required elements, experimental researches were started. Its purpose was to teach a neural network to predict electric power load for the chosen day, having at disposal data from the specific number of previous days. In the output of the network were twenty-four values of load for the given day. Afterwards a mean error, maximal error and absolute mean error were calculated. The quality of results that were achieved using the given network and method was determined taking these errors into consideration.

Simple network that has only one convolution layer appeared to be the best one. As it was forseen, the best method of signal processing was spectrogram. The results were definitely better than in other cases, but different methods also were satisfying. Noticable influence at results had the amount of previous days taken into consideration during making a forecast. In the few best cases values of mean error were about 2-3%.

After resolving the results of experimental research, the predicted values gained from the best method were compared with the real values of load. Given predictions were very good, fulfilling the expectations. It should not be missed that they can not be used to solve technical problems because downloaded data were not accurate enough.

The fact that, the sizes of input matrixes were small and there was not enough of them caused that two convolutin layers, dropout and maxpooling operations did not bring expected effects. It was anticipated that they will dramatically improve the results, however the errors increased – too many features were chosen, that were too specific.

An important element of research turned out to be the time needed to teach some of the networks. The more complicated they were, the more time it took. Nevertheless, it is considered that if the computer used during research had better parameters and the graphical processor unit (GPU), the time would be much shorter.

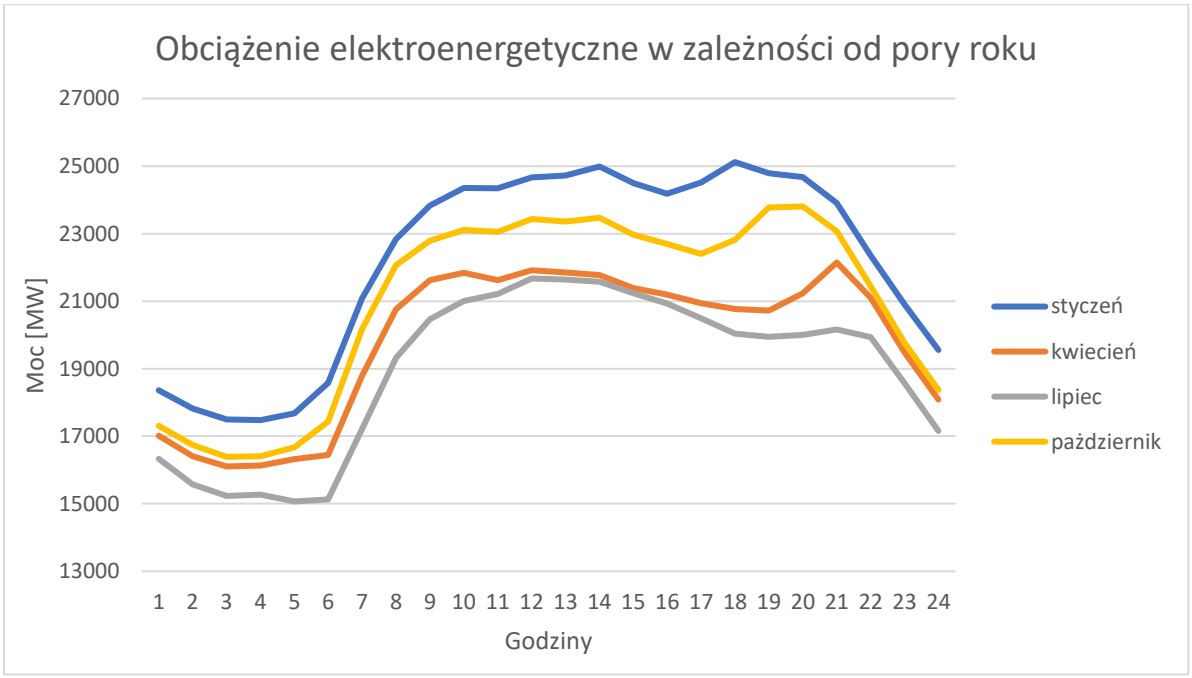
Key-words: convolutional neural networks, prediction, electric power load, change of signal form from 1D to 2D, spectrogram, wave transform

Spis treści

1. Wstęp

W niniejszej pracy przedmiotem badań są metody przetwarzania sygnałów 1D do postaci 2D oraz różne architektury sieci CNN (Convolutional Neural Networks) do prognozowania obciążeń elektroenergetycznych.

Krajowy System Elektroenergetyczny (KSE) to zespół jednostek wytwórczych i odbiorczych połączonych ze sobą, wzbogacony o elementy odpowiadające za przesyłanie, przetwarzanie i rozdzielanie energii elektrycznej na terenie całego kraju. Celem systemu jest dostarczenie do użytkowników energii, w zależności od potrzeb, bez względu na zakłócenia. Liczy się nie tylko jej ilość, ale także jakość. Istotnym elementem są również punkty sterujące zajmujące się bezpieczeństwem i regulacją systemu. Chwilowe wartości obciążeń zależą między innymi od pory roku (rys. 1.1), pogody, dnia tygodnia, godziny czy też szczególnych wydarzeń w kraju. Bardziej szczegółowo, a jednak przystępnie system elektroenergetyczny objaśnił Zdzisław Jankiewicz [2].



Rysunek 1.1. Zapotrzebowanie na moc w Krajowym Systemie Elektroenergetycznym zależnie od pory roku dla 25-tego dnia każdego z przedstawionych miesięcy.

Na podstawie przeprowadzonych badań będzie można prognozować obciążenia elektroenergetyczne. Dzięki temu możliwe będzie przygotowanie systemu na skrajnie wysokie

zapotrzebowanie bądź wykorzystanie niskich poziomów zapotrzebowania mocy np. w celu przepompowania wody w elektrowniach szczytowo – pompowych. Może to służyć nie tylko zapobiegnięciu awarii zasilania (blackout), ale również zrównoważeniu systemu tak, aby był jak najbardziej efektywny.

Wszelkie badania zostały przeprowadzone w środowisku MatLab. Wybrano to narzędzie ze względu na jego szeroki zakres funkcji i obszerną dokumentację.

Dane dotyczące zapotrzebowania mocy Krajowego Systemu Energetycznego zostały pobrane ze strony internetowej Polskich Sieci Elektroenergetycznych. Pod uwagę brane były wartości od stycznia 2007 roku do października 2017 roku. Nie uwzględniono wszystkich dostępnych danych, ponieważ system energetyczny uległ zmianom na przestrzeni lat, więc wcześniejsze dane mogłyby spowodować dodatkowe błędy. Z każdego dnia dostępnych było 96 wartości – pomiary zbierane były co 15 minut, zaczynając od godziny 00.15.

Dostępne dane zostały przetworzone do postaci 2D, ponieważ takiej wymaga sieć CNN. Badaniom zostały poddane trzy główne metody: spektrogram, transformacja falkowa oraz układanie macierzy z fragmentów wektora, z czego trzecia miała 6 różnych wariantów. Na podstawie błędów otrzymanych po przetestowaniu sieci neuronowej określone zostało, która z powyższych metod najlepiej sprawdza się przy zadaniu dotyczącym predykcji.

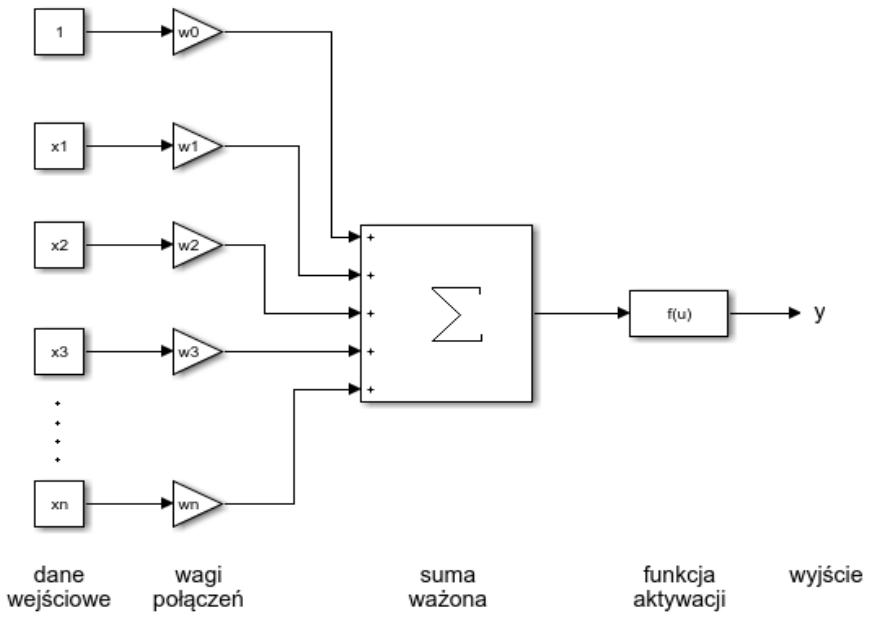
Wszystkie przebadane metody zmiany postaci sygnału oraz stworzone sieci zostały porównane. Wybrano najlepszą konfigurację i porównano ją z rzeczywistymi chwilowymi zapotrzebowaniami.

2. Sztuczne sieci neuronowe

Koncepcja sztucznych sieci neuronowych powstała w oparciu o ludzki układ nerwowy. Ideą było stworzenie programu, który będzie się uczył sam i nie będzie miał sztywnych granic, jak to jest w klasycznym programowaniu.

Podstawowy model sieci neuronowej został wymyślony przez McCullocha i Pittsa w 1943 r. Jego schemat przedstawiony jest na rys. 2.1. Polegał on na tym, że każdy sygnał (x_i) ma swoją określoną wagę (ważność), przez którą zostaje przemnożony. Następnie wszystkie otrzymane wartości są sumowane. Dodaje się do nich również bias, czyli wartość progu, po przekroczeniu którego dany neuron może wysłać impuls. Następnie wynik przekazywany jest do funkcji aktywacji, która może przybierać różne formy. Wartość otrzymana na końcu jest pożądanym wyjściem.

$$y = f(\sum_{i=1}^{N} w_i x_i + b)$$



Rysunek 2.1. Model neuronu w sztucznej sieci neuronowej.

Najpopularniejszą, tradycyjną siecią neuronową jest sieć MLP (Multi Layer Perceptron). Jest to sieć składająca się z kilku warstw, gdzie wyjście neuronu z jednej warstwy staje się wejściem neuronu z

kolejnej warstwy. Zazwyczaj połączenia te są między wszystkimi neuronami warstw sąsiadujących, używa się wtedy określenia, że jest to w pełni połączona sieć MLP (fully - connected MLP).

Różnica między otrzymanym wynikiem a wartością oczekiwaną nazywana jest błędem sieci neuronowej. Błędy sieci neuronowej zostaną opisane bardziej szczegółowo w dalszej części pracy.

Więcej o starszych, klasycznych sieciach neuronowych można dowiedzieć się w książce profesora Osowskiego pt. „Sieci neuronowe do przetwarzania informacji" [7]. Jest w niej sporo informacji na temat różnych rodzajów sieci, lecz nie obemuje ona dziedziny deep learningu, której dotyczyć będzie ta praca.

2.1. Deep Learning, czyli uczenie głębokie

Głębokie sieci neuronowe powstały w wyniku rozwoju uczenia maszynowego i w dzisiejszych czasach są coraz częściej używane. Dąży się do tego, aby stały się jak najwierniejszą kopią ludzkiego mózgu w kwestiach poznawczych. Umożliwiają one nie tylko rozpoznawanie obrazów czy ludzkiej mowy, ale także np. wyznaczanie trendów na rynkach finansowych.

Pojęcie głębokości, użytej w nazwie tego rodzaju sieci odnosi się do ich budowy. Głębokość jest najdłuższą drogą, jaką musi przebyć impuls od neuronu wejściowego do wyjściowego, przechodząc przez sieć. Jednak jest to raczej pojęcie intuicyjne, ponieważ wielkość ta nie jest jednoznacznie określona. Przeważnie przyjmuje się, że jeśli sieć ma więcej niż dwie warstwy ukryte (na rys. 2.2 oznaczone kolorem niebieskim), zaliczana jest do sieci głębokich, ale w skutek dynamicznego rozwoju uczenia maszynowego granica ta może zostać przesunięta.

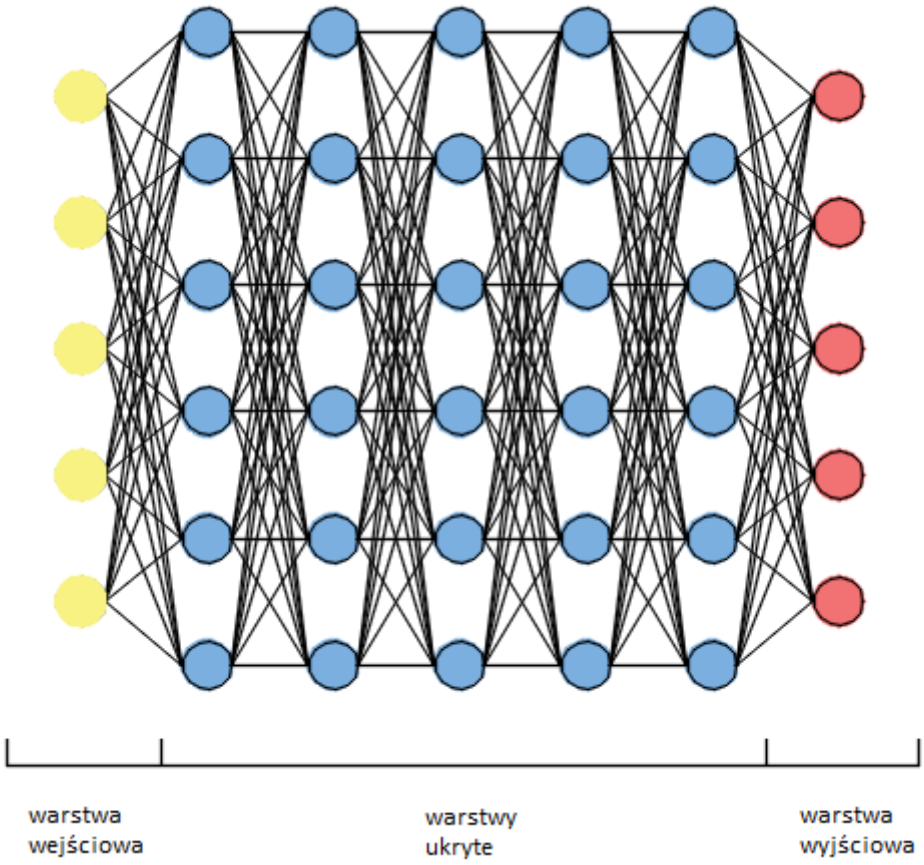
Sieci głębokie składają się z wielu warstw, z których początkowe mają za zadanie wybrać cechy, na podstawie których dane są później klasyfikowane czy rozpoznawane. Jest to proces automatyczny, aczkolwiek nadzorowany. Idea wyboru cech jest taka, że w każdej kolejnej warstwie wybierane cechy są coraz bardziej abstrakcyjne. Następnie trafiają one do klasycznej sieci neuronowej.

Łatwo wyciągnąć z tego wniosek, że nie do każdego problemu dobrym rozwiązaniem będą sieci głębokie. Przy zagadnieniach prostych lub nie mających budowy hierarchicznej lepiej mogą sprawdzić się sieci płytkie.

Skorzystanie z głębokich sieci neuronowych pozwala wyodrębnić cechy w sposób automatyczny, opisujących badany proces bądź zjawisko, co zwalnia użytkownika z obowiązku zdefiniowania cech.

Warto wyjaśnić również pojęcie uczenia nadzorowanego (supervised learning). Polega ono na tym, że na wejście sieci dostarczane są pary uczące, dla uproszczenia oznaczone zostaną jako {x,d}, gdzie x to dane wejściowe, natomiast d to cel, jaki powinien zostać uzyskany, co pozwala później na łatwe

wyliczenie błędu uzyskanego wyniku i ułatwia sieci uczenie. Inaczej sytuacja wygląda w przypadku uczenia nienadzorowanego (unsupervised learning). Na wejście sieci dostarczane są tylko dane wejściowe, bez pożądanego rezultatu, a zadaniem sieci jest samodzielne zrozumienie schematu danych i wyodrębnienie odpowiednich cech. Wyniki sieci uczonej w sposób nadzorowany są przeważnie lepsze. Jednak nie w każdym wypadku mamy dostęp do oczekiwanych wyników, wtedy uczenie nienadzorowane staje się dobrą alternatywą.



Rysunek 2.2. Model sieci głębokiej.

Ciekawe jest to, że pomysł głębokich sieci neuronowych pojawił się już na początku lat 70. XX wieku, jednak ich rozwój był znacznie wolniejszy niż w dzisiejszych czasach. Miało to dwie główne przyczyny:

Po pierwsze, trzeba zauważyć, że do stosowania głębokich sieci neuronowych wymagane są duże zbiory danych, do których dostęp nie zawsze jest powszechny. Dla przykładu, w zrobieniu systemu rozpoznającego tablice rejestracyjne, głównym problemem może być zebranie odpowiedniej ilości danych.

Drugim problemem jest potrzebna moc obliczeniowa i czas uczenia się głębokich sieci neuronowych. Im sieć taka jest większa (składa się z większej liczby warstw), tym bardziej czas uczenia się wydłuża. Stosowana w tym badaniu sieć jest bardzo mała, więc czas ten jest stosunkowo krótki. W dużych sieciach, takich jakie stosuje np. firma Google, czas uczenia sieci, która składa się z dużo większej liczby warstw, znacznie się wydłuża.

Choć w tym momencie te dwa aspekty wciąż stanowią utrudnienie dla twórców sieci głębokich, to kilka dekad temu stanowiły problem nie do ominięcia. Nie da się ukryć, że rozwój technologiczny, a co za tym idzie – znacząco lepsze procesory o mocach obliczeniowych dawniej niewyobrażalnych, a także wykorzystanie procesorów graficznych (GPU) umożliwiły rozkwit tej dziedziny. Ważną rolę odegrał również Internet i ogólnie pojęta cyfryzacja. Dało to dostęp do olbrzymiej ilości baz danych, które mogą służyć za wejście do głębokiej sieci neuronowej.

2.2. Sieci CNN

Jedną z gałęzi deep learningu są sieci CNN (Convolutional Neural Networks – konwolucyjne sieci neuronowe). Charakteryzują się tym, że główną operacją, na jakiej oparte jest działanie sieci jest konwolucja, czyli splot.

$$y(n) = x(n) * w(n) = \sum_{i=-\infty}^{\infty} x(i)w(n-i) = \sum_{i=-\infty}^{\infty} x(n-i)w(i)$$

W tym wypadku **x** to dane uczące, natomiast **w** to przypisane im wagi. Ze względu na to, że zarówno sygnał jak i jądro przekształcenia są w tym wypadku macierzami, dane przedstawiono jako macierz **A**, natomiast wagi jako macierz **B**.

$$Y(i,j) = A(i,j) * B(i,j) = \sum_m \sum_n A(m,n)B(i-m, j-n) =$$

$$= \sum_m \sum_n A(i-m, j-n)B(m,n)$$

Splot jest jednym z podstawowych sposobów przetwarzania sygnału. W prostym tłumaczeniu jest to złożenie dwóch sygnałów (nie wolno pomylić z dodawaniem lub mnożeniem sygnałów) i otrzymanie trzeciego. Szerzej ten temat wytłumaczył Steven Smith [9].

Wejściem do sieci CNN są obrazy w reprezentacji RGB, co daje trzy kanały. Każdy z nich przetwarzany jest przez odpowiednie filtry. Następnie są one sumowane i tak powstaje obraz końcowy. Takich warstw konwolucyjnych, zawierających odrębne filtry, może być wiele. Bardzo duże sieci CNN składają się z nawet kilkunastu takich warstw.

Jedna część sieci CNN składa się zazwyczaj z trzech warstw: filtracji, funkcji aktywacji oraz statystycznego łączenia wyników np. pooling. Najczęściej używaną funkcją aktywacji jest funkcja ReLU (Rectified Linear Unit). W wyniku konwolucji bardzo szybko zmienia się rozmiar obrazu. W celu uniknięcia sytuacji, w której obraz po przejściu przez różne warstwy jest bardzo mały, stosuje się operację zero-padding. Polega ona na dopisywaniu zer na brzegach macierzy po konwolucji.

Wyższość sieci CNN nad klasycznymi sieciami przejawia się głównie w trzech cechach:

- połączenia są lokalne – nie ma tutaj połączeń między wszystkimi neuronami; są one dzielone na grupy na które nakładane są maski (zawsze mniejsze od wymiarów obrazu), co skutkuje mniejszą ilością połączeń, a co za tym idzie, zajmuje mniej pamięci systemu. Takie podejście wzoruje się na ludzkim nerwie wzrokowym, gdzie pole widzenia człowieka podzielone jest na fragmenty i odpowiednie neurony aktywują się zależnie od tego, czy wystąpi zmiana w danym obszarze.

- wagi połączeń są wspólne dla wielu neuronów – wagi połączeń dobierane są tylko dla połączeń lokalnych, dzięki czemu proces uczenia jest efektywniejszy

- niezmienniczość funkcji względem przesunięcia – zapewnia, że zmiana wejścia skutkuje taką samą zmianą wyjścia.

2.2.1. Struktura

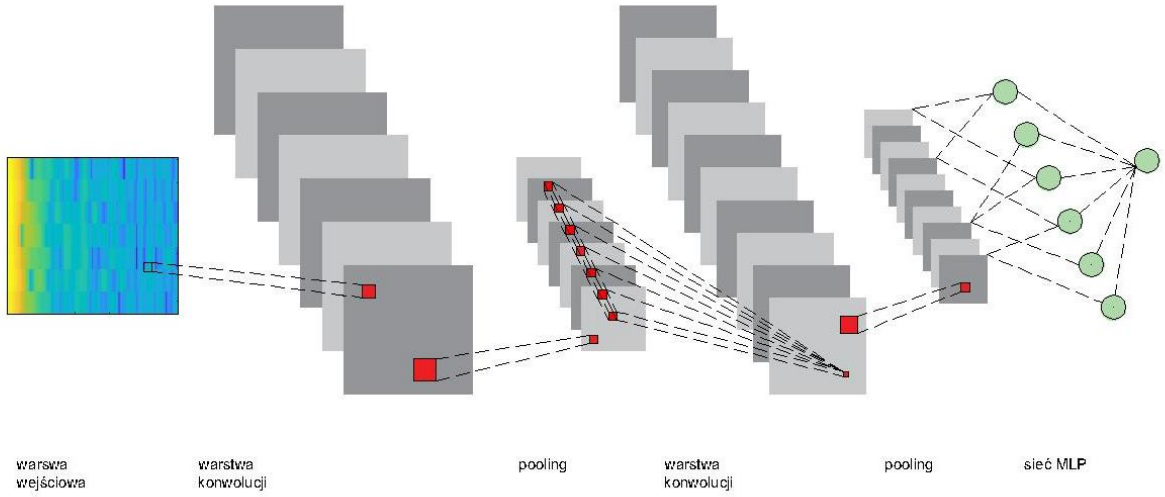
Struktura sieci CNN składa się z kilku warstw konwolucyjnych, warstw głosujących oraz tradycyjnej sieci neuronowej (przeważnie MLP). Każda z warstw konwolucyjnych ma za zadanie wyłonić najważniejsze cechy obrazu, które będą wejściem do tradycyjnej sieci. Dzięki takiemu działaniu wyniki są niezależne, ponieważ system skupia się na najprostszych, najbardziej widocznych cechach.

Pełna konwolucyjna sieć neuronowa (rys. 2.3.) składa się z:

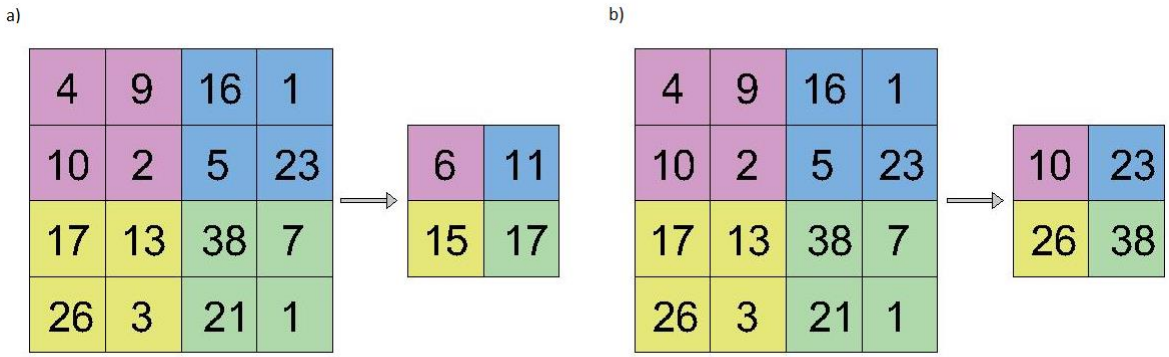
- warstwy wejściowej – zbiór danych wejściowych podany w postaci sygnałów dwuwymiarowych – obrazów.

- warstwy konwolucyjnej – w warstwie tej nakładane są na obraz filtry, które powstają poprzez wybranie cechy – jednej wspólnej wagi dla wielu neuronów. Filtry te mają postać dwuwymiarową, podobnie do obrazów wejściowych. Zamiast tradycyjnego modelu neuronu stosuje się tutaj operację splotu, ponieważ daje ona lepsze wyniki przy wyodrębnianiu cech.

- warstwy głosującej – stosowany jest tutaj pooling, czyli wyciąganie z danej macierzy kwadratowej (fragmentu całego obrazka) liczby największej (max – pooling) bądź średniej (average – pooling). Zadaniem tej warstwy jest zmniejszenie wymiarów danych; szczególnie w większych sieciach jest to bardzo przydatne.



Rysunek 2.3. Model konwolucyjnej sieci neuronowej.



Rysunek 2.4. Zasada działania operacji: a) average – pooling b) max – pooling.

- warstwy pełnej – jest to tradycyjna sieć neuronowa (przeważnie MLP, opisana już wcześniej). Wszelkie wcześniejsze operacje mają za zadanie przygotować dane i cechy na wejście do tej właśnie sieci.

- warstwy wyjściowej – ostatniej warstwy, z której otrzymujemy pożądany wynik.

Liczba warstw konwolucyjnych, głosujących jak i warstw sieci tradycyjnej dobierana jest przez użytkownika, lecz nie powinna być przypadkowa. Należy znaleźć taką ilość, która umożliwi

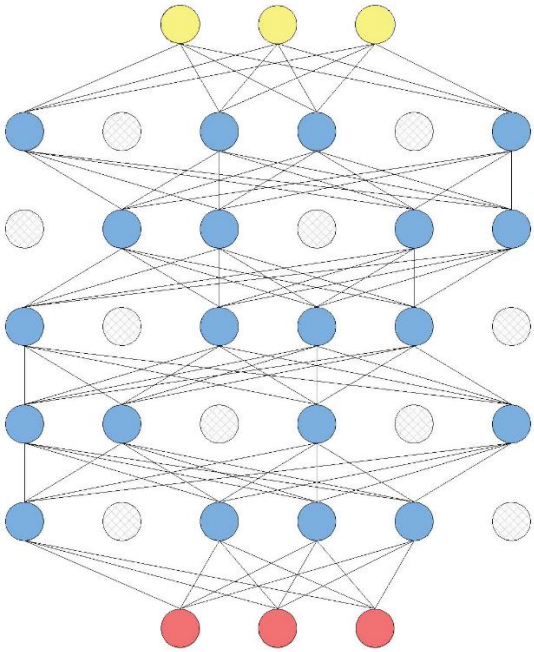
odpowiednie przetworzenie obrazu, ale jednocześnie nie będzie zbyt duża i nie spowoduje zafałszowania wyników.

Dodatkowo, operacją, którą można wykorzystać dla polepszenia wyników sieci jest dropout – szerzej opisana w publikacji w Journal of Machine Learning Research [10] . Może ona skutecznie zapobiegać problemowi przetrenowania sieci, który przybliżony zostanie później. Operacja dropout polega na losowym usunięciu niektórych neuronów z sieci (białe neurony na rys. 2.5.). Dzieje się to, gdy na każdą warstwę wyjściową nałożona zostanie maska m_k. Gdy sieć jest bardzo duża takie działanie może okazać się niezwykle przydatne. Przy każdym uczeniu sieci mogą być usunięte różne neurony, a ich ilość zależy od podanego parametru lub jest ustalona jako stała wartość – najczęściej 0,5, co oznacza, że połowa neuronów zostaje wykluczona z dalszego uczenia. Funkcja dropout może być również traktowana jako dadanie szumu do każdej warstwy sieci. Taki zabieg skłania sieć do skupienia się na najistotniejszych cechach.

$$m_k \sim Bernoulli\ (p)$$

$$y'_k = m_k * y_k$$

$$y_{k+1} = f(w_k y'_k + b_k)$$



Rysunek 2.5. Ilustracja operacji dropout.

Warto zwrócić również uwagę na to, że warstwa konwolucyjna różni się od warstwy sieci tradycyjnej przede wszystkim tym, że w tej drugiej, połączenia między neuronami są globalne, a wagi dobierane są do każdego polączenia, dlatego warto nieco zmniejszyć obraz podczas procesu konwolucji, szczególnie w wypadku bardzo dużych (pod względem ilości pikseli) obrazów.

2.2.2. Proces uczenia

Uczenie każdej sieci neuronowej polega na dobraniu odpowiednich wag neuronów. W przypadku sieci CNN do tego celu należy dobrać odpowiedni rozmiar masek filtracyjnych. W praktyce jednak jest to problem wymagający rozważenia wielu aspektów. Tworząc sieci neuronowe na potrzeby tego badania kierowano się wskazówkami umieszczonymi w książce *Deep Learning* [1] w rozdziale 11 – *Practical Methodology*.

Pierwszą rzeczą, jaka jest potrzebna do nauczenia sieci są dane. Na wejście sieci muszą zostać dostarczone dane uczące oraz testujące. Mogą być to pary {x, d}, jeśli będzie to uczenie nadzorowane, bądź same dane x w wypadku uczenia nienadzorowanego.

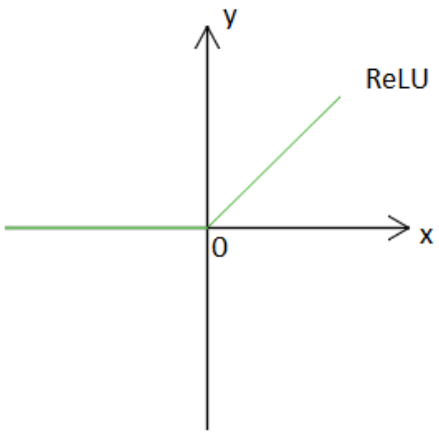
Kolejnym krokiem jest określenie błędów, jakie będą liczone w celu zbadania czy sieć spełnia swoją funkcję, a także określenie wartości błędów, które będą satysfakcjonujące. Nie da się stworzyć sieci, która będzie miała wszystkie błędy równe zeru. Jednak w zależności od problemu wymagania dotyczące jednych błędów będą większe, a innych mniejsze. Dobrą ilustracją problemu może być przytoczony w książce *Deep Learning* [1] przykład dotyczący selekcji spośród wiadomości e-mail tych, które kwalifikują się jako spam. Sytuacja, w której e-mail będący spamem trafia do skrzynki odbiorczej jest niepożądana, lecz odwrotna sytuacja, gdy wiadomość, która powinna trafić do adresata, przekierowywana jest do folderu spam, jest niedopuszczalna. Istotne jest więc, aby na tym etapie postawić odpowiednie pytania i wyznaczyć cele dotyczące powstającej sieci tak, aby w dalszym budowaniu sieci móc dążyć do ich osiągnięcia.

Następnie można przystąpić do budowy sieci. Dobrze jest zacząć od niezbyt skomplikowanej architektury i później, w razie potrzeby, stopniowo ją rozbudowywać. Należy zdefiniować ilość warstw konwolucyjnych oraz rozmiar i liczbę filtrów w każdej z nich. Jest to bardzo ważna kwestia, ponieważ w tych warstwach sieć dokonuje analizy obrazu weściowego i w nich przebiega kluczowe uczenie.

Ponadto należy wybrać funkcję aktywacji. Istnieje wiele różnych rodzajów tej funkcji – np. liniowa, progowa, sigmoidalna, tanh, Softplus, Softmax bądź ReLU. Ze względu na to, że w tym badaniu zastosowana zostanie funkcja ReLU (Rectified Linear Unit), zostanie ona dokładniej opisana. Nie da się ukryć, że jest ona bardzo prosta. W przedziale argumentów mniejszych od zera jest ona równa zero, więc jeśli jakieś wartości są ujemne, ustawiane są one na zero. Natomiast w przedziale większym bądź równym zeru funkcja ReLU przyjmuje postać funkcji liniowej, więc jej wartościom przypisywany jest odpowiedni argument. Jednak fakt, że nie jest ona symetryczna wyróżnia ją na tle wcześniej

stosowanych funkcji aktywacji i przedstawia się jako zaleta przy badaniu zbieżności i wyników doświadczeń, dokładne badania na ten temat opisano w publikacji *Delving Deep into Rectifiers: Surpassing Human – Level Performance on ImageNet Classification* [3]. Często dla polepszenia wyników trenowania sieci stosuje się pretrenowanie, ale użycie funkcji ReLU i dropout pozwala na ominięcie tego etapu.

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$



Rysunek 2.6. Wykres funkcji ReLU.

Dobieranie odpowiednich parametrów i warstw sieci jest dość żmudnym i czasochłonnym zajęciem, więc dobrze jest wzorować się na istniejących już architekturach, zamiast tworzyć sieć w zupełnie przypadkowy sposób. Istnieją też metody polegające na użyciu wytrenowanej już dużej sieci, a następnie douczenie jej w odpowiedni sposób tak, aby umiała rozwiązać problem. Popularną do tych celów siecią jest sieć AlexNet [5], która została wytrenowana na ogromnym zbiorze danych i służy głównie do rozpoznawania obrazu. Można ją dotrenować tak, aby rozpoznawała nie tylko np. kota z różnych zdjęć, ale miała bardziej wyspecyfikowane zadania, np. wybrać kota konkretnej rasy spośród zdjęć kotów.

Po uruchomieniu, sieć uczy się automatycznie: sama wybiera odpowiednie cechy obrazów wejściowych, uczy się ich i na wyjściu podaje wynik. Później należy tylko obliczyć błędy i w zależności od tego czy są satysfakcjonujące czy nie, zmienić budowę sieci.

Bardzo ważną rzeczą, o której trzeba pamiętać jest odpowiednia ilość epok (jednostek uczących). W wypadku, gdy będzie ich za mało sieć będzie niedotrenowana, może się to objawiać np. wyższym błędem danych uczących niż testujących, co z logicznego punktu widzenia nie powinno mieć miejsca. Natomiast w odwrotnej sytuacji, gdy liczba ta jest zbyt duża, błąd uczący będzie się zmniejszał, lecz błąd testujący będzie rósł, a to również nie zapewnia zadowalających wyników.

Błędów, które można policzyć w celu ocenienia jakości sieci jest wiele, lecz w tej pracy opisane zostaną tylko cztery, ze względu na to, że będą rozpatrywane.

Błąd średni absolutny (MAE – Mean Absolute Error) jest to realna wartość, o jaką różni się otrzymany wynik od rzeczywistego. Daje on dokładną informację dla użytkownika, o ile wynik sieci różni się od rzeczywistego w odpowiednich jednostkach. Dla człowieka jest to lepsze zobrazowanie niż błąd średni.

$$err_{mae} = \frac{1}{N}\sum_{i=1}^{N}|y_i - d_i|$$

Błąd średniokwadratowy (MSE – Mean Squared Error) jest to średnia wartość kwadratu różnicy otrzymanej wartości i rzeczywistej. Daje on lepszy obraz jeśli chodzi o różnice w błędach. Podczas gdy błąd średni rośnie liniowo, błąd średniokwadratowy rośnie parabolicznie, co daje dużo lepszy pogląd na otrzymane dane. Nie był brany pod uwagę w procesie oceniania sieci, lecz liczony był automatycznie w warstwie regresyjnej w sieci.

$$err_{mse} = \frac{1}{N}\sum_{i=1}^{N}(y_i - d_i)^2$$

Błąd średni jest to średnia różnica między otrzymaną wartością, a rzeczywistą podzielona przez wartość rzeczywistą. Ten błąd jest zdecydowanie najważniejszy w tym badaniu. Na jego podstawie będzie określana jakość sieci.

$$err_{mean} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - d_i}{d_i}\right| * 100\%$$

Błąd maksymalny jest to największa różnica między otrzymaną wartością, a rzeczywistą podzielona przez wartość rzeczywistą. Jest to błąd, który będzie używany w ramach dodatkowej kontroli.

$$err_{max} = \max\left(\left|\frac{y_i - d_i}{d_i}\right|\right) * 100\%$$

2.2.3. Zastosowanie sieci konwolucyjnych

Zastosowanie konwolucyjnych sieci neuronowych jest bardzo szerokie w sytuacjach gdy mają zastąpić dwa z pięciu ludzkich zmysłów – wzrok i słuch. Można je wykorzystywać do rozpoznawania obrazów lub ich klasyfikacji, a także analizy dokumentów czy pisma. Także wykraczają one poza nieruchomy obraz i mogą być stosowane do np. wykrywania ruchu człowieka czy analizy video. Natomiast jeśli chodzi o dźwięk, może to być rozpoznawanie dźwięku np. piosenek, słów bądź mówcy [6] albo analiza mowy. Rzadziej konwolucyjne sieci neuronowe stosuje się do predykcji, aczkolwiek nie ma przeciwskazań do korzystania z nich w tym dziale, ponieważ sprowadzają się one do nauczenia pewnych wzorców. Biorąc pod uwagę fakt, że potrzebne jest wiele danych do dokonania w miarę trafnych przewidywań, a człowiek z przeanalizowaniem takiej ilości danych mógłby mieć spory problem, można pokusić się o stwierdzenie, że w tej dziedzinie maszyna ma nie tylko dorównać człowiekowi, lecz go przewyższyć.

W celu stosowania sieci CNN do klasyfikacji obrazu potrzebna jest ogromna liczba danych i w zależności od stopnia skomplikowania obrazów – odpowiednio duża sieć. Badania bardzo dużej sieci wykorzystywanej do takich właśnie celów prowadzili Krizhevsky, Sutskever i Hinton z Uniwersytetu w Toronto [4].

Z kolei jeśli chodzi o rozpoznawanie obrazu, sukcesami mogą pochwalić się Simonyan i Zisserman [8], którzy prowadzili badania na temat rozpoznawania obrazu na dużą skalę, lecz przy użyciu bardzo głębokich sieci, a o małych wymiarach filtrów – takich, jakie użyte zostaną przy tym badaniu. Wyniki potwierdziły stwierdzenie, że dla bardzo dużych zbiorów danych wraz z głębokością sieci wzrasta jej efektywność.

3. Przetwarzanie sygnału

Pojęcie przetwarzania sygnału obejmuje wszystkie operacje, jakich można dokonać na danym sygnale. Generalnie sygnały podzielić można na analogowe oraz dyskretne i wedle potrzeb odpowiednio je przetworzyć. Robi się to w celu zbadania pewnych właściwości sygnału bądź zmiany jego postaci, do takiej, która jest potrzebna.

3.1. Metody preprocesingu sygnałów 1D do postaci 2D.

Metod przetwarzania sygnałów z postaci wektorowej do macierzowej jest wiele, lecz w tym badaniu zostały wybrane tylko trzy. Są nimi: spektrogram, transformacja falkowa oraz zwykłe podzielenie wektora na części i złożenie ich razem w macierz (6 wariantów). Etap przetworzenia obrazu w tym badaniu jest ważny ze względu na to, że sieć CNN wymaga na wejściu obrazów (macierzy), a dane są zapisane w postaci wektora.

3.1.1. Spektrogram

Jednym z przekształceń, którym można poddać dany sygnał jest transformacja Fouriera, która pozwala zaobserwować rozkład częstotliwości w danym sygnale. Dla lepszego zobrazowania, transformata Fouriera ilustruje funkcje sinus i cosinus, z których składa się sygnał. Bardzo dobrze sprawdza się ona do analizowania globalnych cech sygnału.

$$X(j\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}\,dt$$

Może ona jednak być stosowana tylko do funkcji ciągłych, natomiast do funkcji dyskretnych wykorzystuje się dyskretną transformację Fouriera DFT. Umożliwia ona przekształcenie funkcji dyskretnej poprzez zmianę operacji całkowania na sumowanie. Sygnały, w znacznej mierze są teraz w postaci cyfrowej (przetwarzane przez komputery), dlatego taka postać transformacji Fouriera była niezbędna.

$$X(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi\frac{nm}{N}}$$

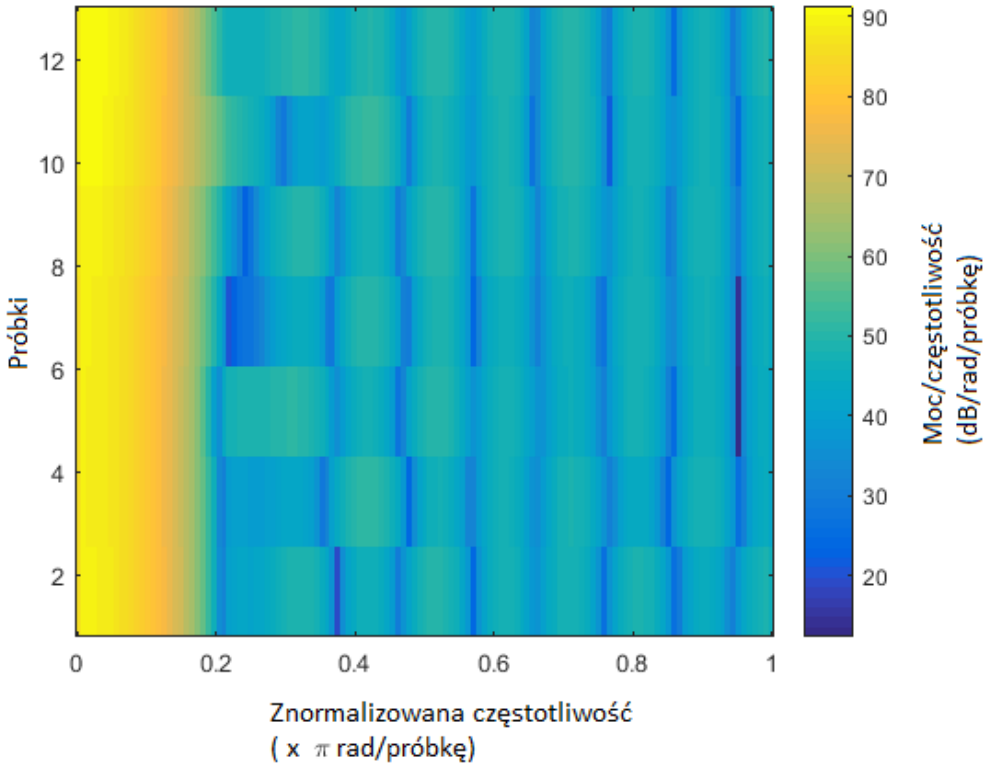
Jednak ze względu na jej bardzo dużą złożoność obliczeniową – N^2 działań arytmetycznych, zdecydowanie częściej korzysta się z szybkiej transformacji Fouriera FFT, podczas której wykonywane jest jedynie $\frac{N}{2}\log_2 N$ działań. Szybka transformata Fouriera nie jest uproszczeniem DFT i zachowuje wszystkie właściwości dyskretnej transformaty Fouriera.

$$W_N = e^{-j\frac{2\pi}{N}}$$

$$X(m) = \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_{\frac{N}{2}}^{nm} + W_N^m \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_{\frac{N}{2}}^{nm}$$

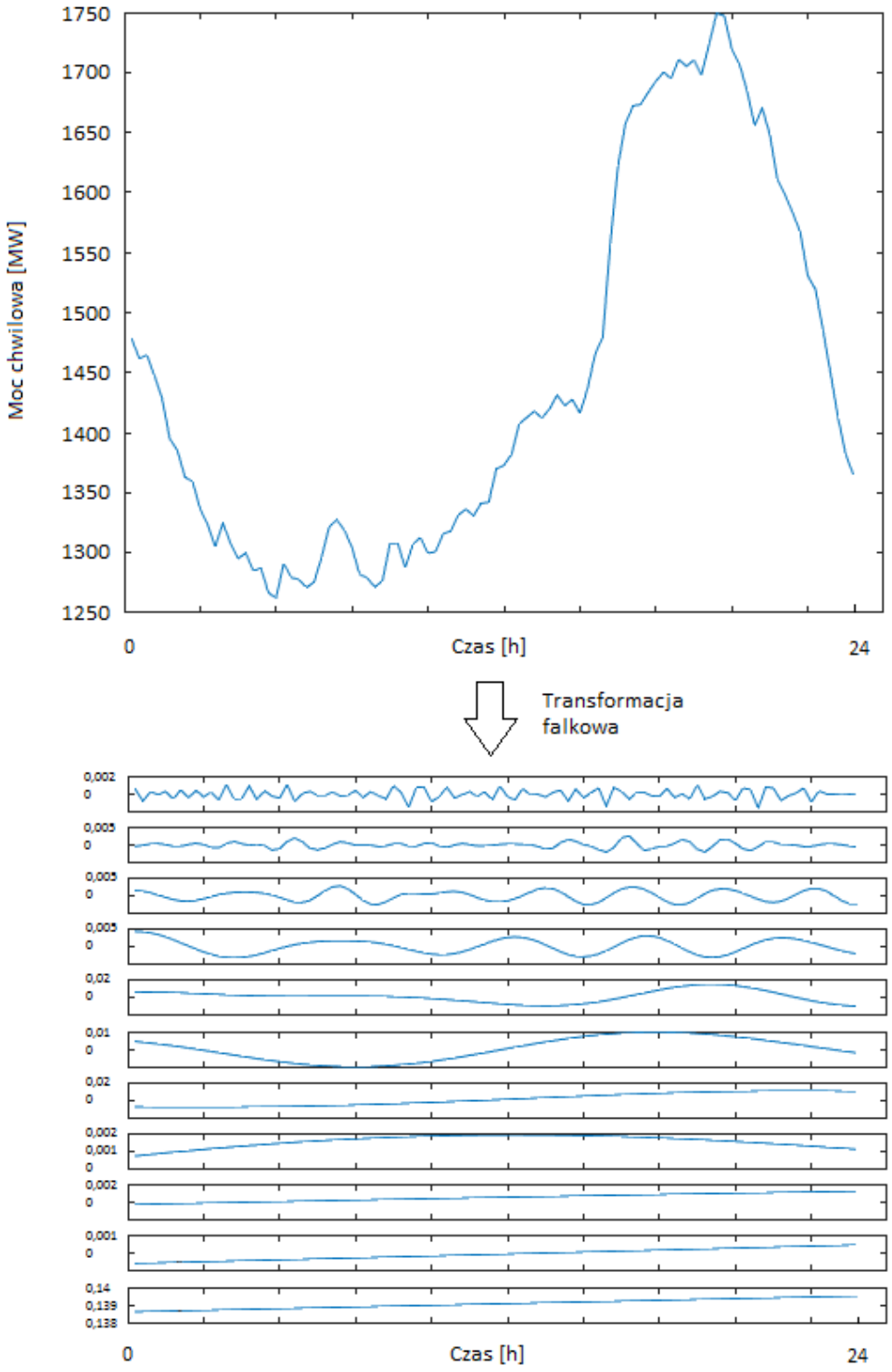
Wizualną reprezentacją szybkiej transformaty Fouriera jest spektrogram. Ze względu na to, że FFT bardzo dużo mówi o danym sygnale, dobrze byłoby zastosować ją jako metodę przetworzenia obrazu w tym wypadku. Spektrogram doskonale to umożliwia, ponieważ spełnia warunki, aby być wejściem do sieci CNN – jest obrazem.

W celu uzyskania spektrogramu zostanie użyta funkcja *spectrogram(x)*, dostępna w MatLabie, gdzie x jest sygnałem wejściowym – wektorem. Funkacja ta ma dodatkowo wiele atrybutów, które nie zostaną wykorzystane.



Rysunek 3.1. Interpreacja graficzna szybkiej transformaty Fouriera – spektrogram.

3.1.2. Metoda falkowa



Rysunek 3.2. Rozkład sygnału na falki dla losowo wybranego dnia.

Kolejnym przekształceniem, które zostanie użyte w tym badaniu jest rozkład sygnału na falki. Podczas, gdy w transformacie Fouriera sygnał składa się z sinusów i cosinusów, w transformacie falkowej jego cząstkami są falki, które nie są tak jednoznacznie określone. Jest jednak kilka kryteriów, które muszą spełniać:

- wartość średnia falki musi być równa 0

$$\int_{-\infty}^{+\infty} \varphi(t)dt = 0$$

- jej pasmo przenoszenia musi być skończone

- musi być znormalizowana $||\varphi|| = 1$

- skupiona musi być wokół $t = 0$

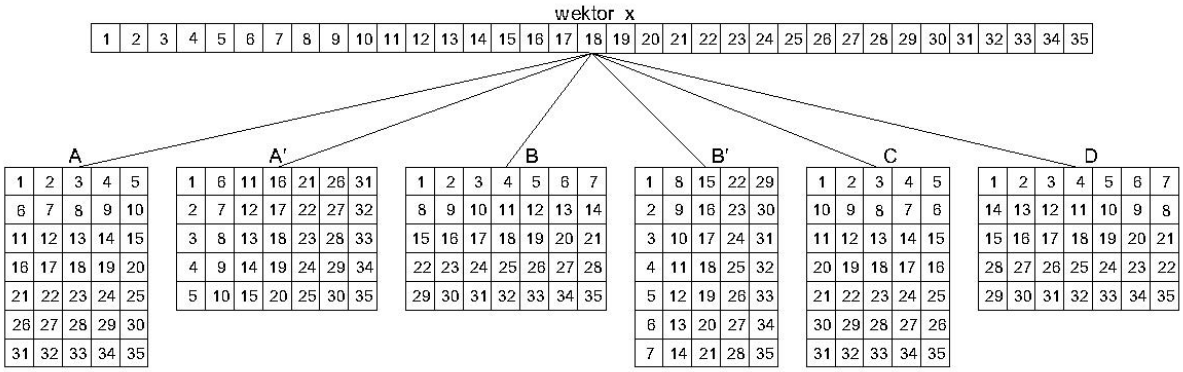
$$Wf(u,s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{+\infty} f(t)\, \Psi\left(\frac{t-u}{s}\right) dt$$

Transformację falkową powszechnie stosuje się w przetwarzaniu obrazu. Może być to zarówno monitorowanie statków w oparciu o dane z satelit czy obrazów z radarów jak i badanie dzieł sztuki. Jest także wykorzystywana do analiz sygnałów z badań zjawisk przyrodniczych.

3.1.3 Metoda układania macierzy z fragmentów wektora

Metoda ta polega na przekształceniu sygnału w postaci wektora na macierz dwuwymiarową składając ją z fragmentów tego wektora. Można to zrobić na różne sposoby: spisywać dane rzędami po kolei od lewej do prawej bądź też jeden rząd od lewej do prawej, a drugi odwrotnie. Analogicznie można postąpić spisując te dane kolumnami. Z tego powodu w tej metodzie mamy wiele wariantów użytych w trakcie badań. Są one bardzo podobne, ale nie takie same, więc spodziewać się można różnych wyników. Uzyskane macierze są danymi wejściowymi do sieci.

Na rys. 3.3 przedstawiono sposoby tworzenia macierzy z wektora danych. Nazwy macierzy odpowiadają nazwom użytym w badaniu, lecz liczba komórek, wierszy i kolumn jest inna i ma jedynie zobrazować sposób zmiany postaci sygnału 1D na 2D. Wariantów tworzenia macierzy z wektora jest bardzo dużo, tutaj zostało wybranych kilka najprostszych.



Rysunek 3.3. Sześć możliwości tworzenie macierzy z wektora – zamiana postaci sygnału 1D na 2D.

4. Badania eksperymentalne

Przed rozpoczęciem badań należało wykonać kilka czynności mających na celu przygotowanie danych do użycia ich przy uczeniu sieci. Dane zostały pobrane ze strony internetowej Polskich Sieci Elektroenergetycznych w 10 plikach programu Excel – w każdym zawierały się chwilowe wartości zapotrzebowania na moc w Polsce z danego roku, zbierane co 15 minut. Dane zebrane zostały z lat 2007-2017, dostępne były co prawda dane już od 2002, ale w tym wypadku wykorzystanie ich wiązałoby się z większymi błędami, ponieważ średnie zapotrzebowanie na moc w Polsce rosło na przestrzeni lat i starsze dane mogłyby znacząco zafałszować wyniki. Trzebabyło więc wybrać granicę, która dostarczyłaby wystarczającą liczbę danych, ale jednocześnie byłyby one na tyle aktualne, że nie wprowadzałyby dodatkowych szumów do sieci.

Zarówno obróbka danych jak i budowa oraz testowanie sieci zostały wykonane w programie MatLab. Wszystkie programy i funkcje, z których korzystano zostały załączone do niniejszej pracy.

4.1. Dane wejściowe

Pierwszym etapem, który odbywał się w programie głównym (program_glowny.m) było wpisanie dwóch parametrów dotyczących uczenia sieci:

- metoda – nr metody przetwarzania sygnału używany w danym przypadku.

- numofdays – liczba dni wstecz, na podstawie których sieć ma ustalić, jakie będzie obciążenie elektroenergetyczne kolejnego dnia.

Następnie wywoływana była funkcja, w której odczytywane były dane z plików programu Excel i tworzony był wektor P z wszystkimi wartościami obciążeń, który później był normalizowany do łatwiejszej dalszej obróbki. Fragment kodu realizującego te zadania:

```
Braw=[];
for r = 2007:2017
plikcsv = sprintf('%d.csv',r);
[num,txt,raw] = xlsread( plikcsv );
Braw = [Braw; raw(2:end,:)];
end

C = Braw(:,4);
S = sprintf('%s*', C{:});
P = sscanf(S, '%f*');
P = P./max(P);                          % znormalizowany wektor obciążeń
```

Następnie, zależnie od wybranej metody, wywoływana była funkcja, gdzie tworzone były pary uczące i testujące, będące wejściem do sieci neurownowej. Poniżej zamieszczony został fragment kodu odpowiadający za utworzenie par metodą III (macierz A).

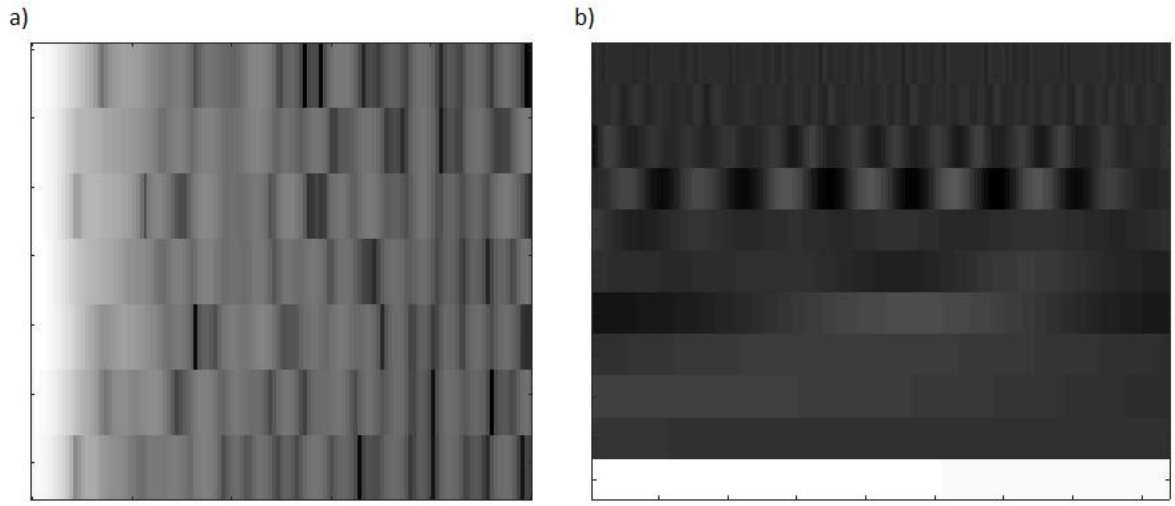
```
function [] = macierzA(numofdays,P)
lprob = 96;                                % liczba probek w ciągu doby
if (numofdays>5)                           % wymiary powstającej macierzy
    k=2*numofdays;
    j=12;
else
    k=4*numofdays;
    j=6;
end
[lw,lk] = size(P);                         % rozmiar wektora obciążeń
xnum = floor(((lw-((numofdays*lprob)+lprob))/lprob)+1);    %liczba par
sizeu = ceil(5*xnum/6);                    % liczba par uczących
sizet = xnum-sizeu;                        % liczba par testujących
xu = zeros(k,j,1,sizeu);
xt = zeros(k,j,1,sizet);
xtemp = zeros(k,j,1,xnum);                 % pary tymczasowe
dtemp = [];
du = [];
dt = [];
count=1;
for i = (numofdays*lprob+1):lprob:lw-lprob+1          % tworzenie par
    d = P(i:lprob/24:i+lprob-1)';
    x = reshape(P(i-1:-lprob/24:i-numofdays*lprob),k,j);
    xtemp(:,:,1,count) = x;
    dtemp = [dtemp;d];
    count = count+1;
end
idx = randperm(xnum);                    % wektor losowych wartości z przedziału
1:xnum
xu = xtemp(:,:,1,idx(1:sizeu));          % tworzenie losowych par uczących i
testujących
xt = xtemp(:,:,1,idx(sizeu+1:end));
du = dtemp(idx(1:sizeu),:);
dt = dtemp(idx(sizeu+1:end),:);
siec_cnn_regr(xu,xt,du,dt)                 % przejście do funkcji, w której
stworzona została sieć neuronowa
end
```

W plikach odpowiadających za inne metody zmieniał się jedynie fragment zawarty w pętli for, gdzie do macierzy x wstawiany był obraz powstały przez skorzysatnie z funkcji *spectrogram*, bądź *rozklad_na_falki*, którego kod został przedstawiony poniżej.

W tych wypadkach zbędny był rówież proces liczenia wymiarów powstającej macierzy.

```
function [falki] = rozklad_na_falki(sig, falka, N);
[C,L]=wavedec(sig,N,falka);
detale=[1:N];
for k=detale
   up2=wrcoef('d',C,L,falka,k);
   zwrot(k,:)=up2';
end
zwrot(k+1,:)=wrcoef('a',C,L,falka,N)';
falki = zwrot;
```

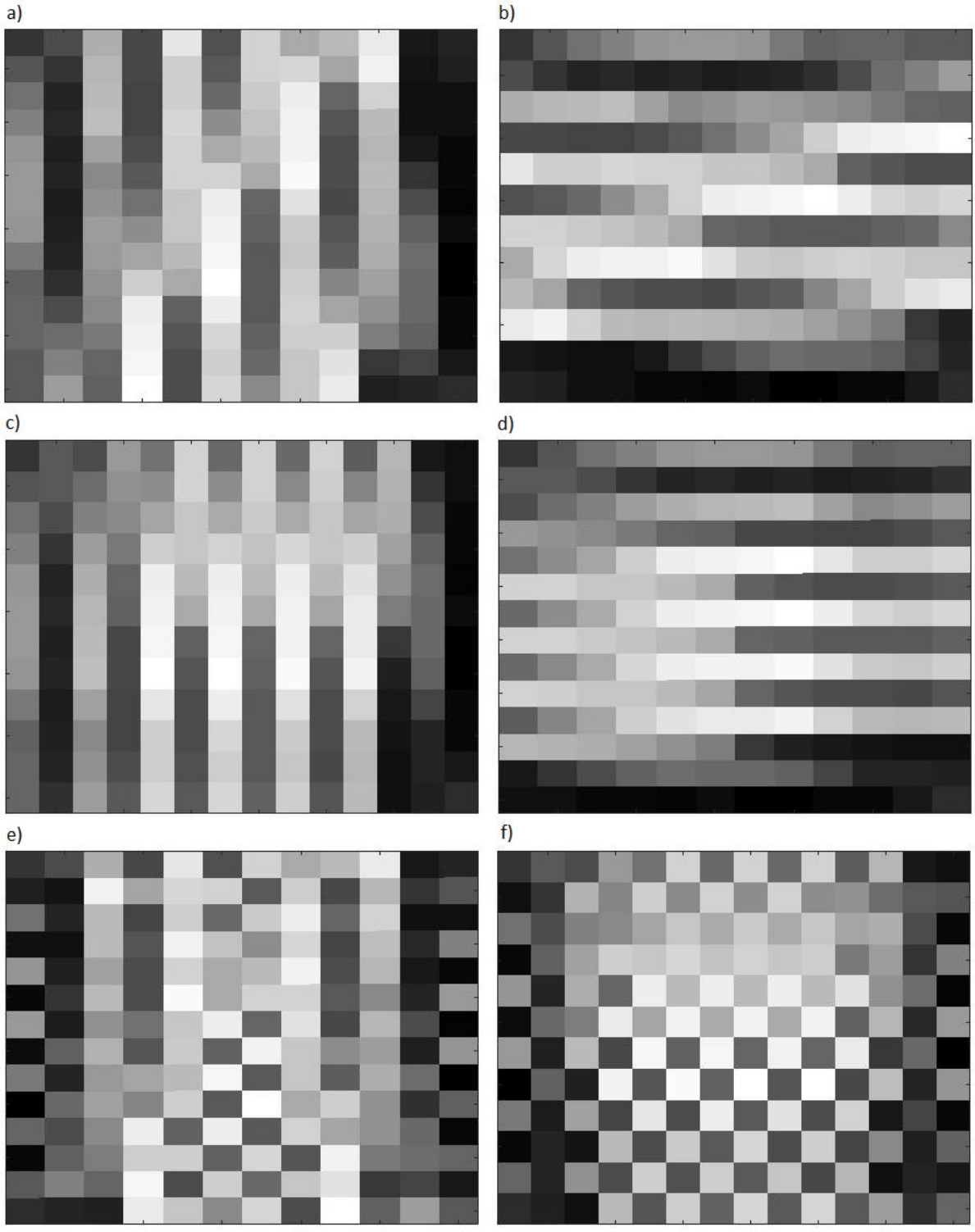
Jak już wcześniej wspomniano, w badaniu przedstawiono trzy różne metody przetwarzania sygnału. Wszystkie zaprezentowane przykłady zostały wykonane dla tych samych siedmiu dni. Dwie pierwsze to spektrogram i transformacja falkowa, których wyniki przedstawiono na rys. 4.1.



Rysunek 4.1.Wizualizacja sygnałów wejściowych w postaci 2D a) spektrogram b) falki.

Stworzenie obrazu pierwszą metodą wymagało jedynie wywołania funkcji *spectrogram(x)* w programie MatLab, gdzie x jest wektorem danych wejściowych (fragmentem wektora P o długości zależnej od liczby dni wstecz branych pod uwagę). Z kolei obraz powstały metodą transformacji falkowej jest przedstawieniem falek za pomocą prostokątów o różnej intensywności koloru. Każdy wiersz tego obrazu jest inną falką, co powoduje, że ostatnia z nich jest bardzo jasna - wartości tego przebiegu są najwyższe.

Trzecia metoda miała sześć różnych wariantów. Interpretacje graficzne tych macierzy zostały przedstawione na rys. 4.2. Jak widać, macierze są do siebie parami podobne: macierz A i Ap, B i Bp, C i D, co wynika z podobnego sposobu ułożenia ich wierszy i kolumn, który został dokładnie przedstawiony na rys. 3.3. Do tworzenia macierzy została wykorzystana funkcja *reshape(x,k,j)*, gdzie x jest wektorem danych, natomiast k i j to wymiary tworzonej macierzy. Z kolei w przypadkach e) i f) z rys. 4.2 potrzebna



Rysunek 4.2. Interpretacja graficzna macierzy utworzonych z wektora x podzielonego na części: a) macierz A b) macierz A’ c) macierz B d) macierz B’ e) macierz C f) macierz D (patrz rys. 3.3).

była również funkcja *fliplr(A(l ,:))*, która pozwalała na zmienienie kolejności komórek w danym wierszu/kolumnie. Zamiast rosnąć od lewej do prawej, rosły od prawej do lewej (macierz C z rys. 3.3).

Istotną rzeczą jest fakt, że mimo, iż wstępne dane zawierały chwilowe zapotrzebowanie na moc zbierane co 15 minut, dane przekazywane do sieci były okrojone do danych z godzinnymi odstępami. Takich też wyników oczekiwano od sieci.

Przy tworzeniu macierzy zawierających dane uczące i testujące posłużono się funkcją *randperm(xnum)*, która umożliwiła losowe wybranie tych danych. Układa ona wartości z danego zakresu (1:xnum) w przypadkowej kolejności.

Następnie uzyskane macierze były umieszczane w czterowymiarowych macierzach xu(k, j, 1, sizeu) i xt(k, j, 1, sizet), gdzie pierwsze dwa wymiary to wymiary jednej macierzy (obrazka), kolejny wymiar mówi o tym, że obrazy są monochromatyczne, a czwarty jest wielkością tej macierzy – ilością obrazów wejściowych. Macierze du i dt są macierzami dwuwymiarowymi, gdzie pierwszy wymiar jest równy ilości tych wektorów, takiej samej jak ilość obrazków w macierzach xu(sizeu,24) i xt(sizet,24), natomiast drugi wynosi 24, więc tyle, ile wektor oczekiwanych wartości.

4.2. Opracowane sieci konwolucyjne

Wszystkie macierze były również danymi wejściowymi do funkcji *siec_cnn_regr*, która zawierała odpowiednio modyfikowaną sieć neuronową CNN. Podstawowa jej konfiguracja zawierała nastęujące elementy:

- imageInputLayer([rozmx rozmy 1]) – warstwę wejściową, której argumentami był rozmiar obrazków umieszczonych w wektorze xu.

- convolution2dLayer(n,m) – warstwę konwolucyjną, najważniejszą, ponieważ na niej opiera się działanie tej sieci. Jej argumentami były wielkość filtrów, która zawsze jest kwadratowa i równa n x n oraz ilość tych filtrów równa m. Ilość tych warstw i jej argumentów ulegała zmianie podczas badań.

- reluLayer – warstwa odpowiadająca za przejście sygnału przez funkcję aktywaci ReLU. Została ona umieszczona w kilku miejscach tej sieci.

- fullyConnectedLayer(n) – warstwa pełna, zawierająca neurony uczące się na podstawie wcześniej wyekstraktowanych cech. Jej argumentem jest liczba neuronów – n, która w wypadku, gdy warstwa ta jest przedostatnia musi być równa 24 – liczbie wartości wyjściowych sieci.

- regressionLayer – warstwa wykorzystująca funkcję strat w postaci RMSE (root mean square error) i pozwalająca wybrać elementy kluczowe do wyliczenia wartości wyjściowych.

Warianty sieci wykorzystywanych w badaniach:

1. imageInputLayer ([rozmx rozmy 1]
   convolution2dLayer (3,20)
   reluLayer
   fullyConnectedLayer (24)
   reluLayer
   regressiornLayer];

2. imageInputLayer ([rozmx rozmy 1]
   convolution2dLayer (3,55)
   reluLayer
   fullyConnectedLayer (24)
   reluLayer
   regressionLayer];

3. imageInputLayer ([rozmx rozmy 1]
   convolution2dLayer (3,55)
   reluLayer
   fullyConnectedLayer (24)
   reluLayer
   fullyConnectedLayer (24)
   regressionLayer];

4. imageInputLayer ([rozmx rozmy 1])
   convolution2dLayer (3,55)
   reluLayer
   fullyConnectedLayer (48)
   reluLayer
   fullyConnectedLayer (24)
   regressionLayer];

5. imageInputLayer ([rozmx rozmy 1])
   convolution2dLayer (3,55)
   reluLayer
   maxPooling2dLayer(2,'Stride',2)
   fullyConnectedLayer (48)
   reluLayer
   fullyConnectedLayer (24)
   regressionLayer];

6. imageInputLayer ([rozmx rozmy 1])
   convolution2dLayer (3,55)
   reluLayer
   maxPooling2dLayer (2)
   fullyConnectedLayer (48)
   reluLayer
   fullyConnectedLayer (24)
   regressionLayer];

7. imageInputLayer ([rozmx rozmy 1])
   convolution2dLayer (3,55)
   reluLayer
   fullyConnectedLayer (48)
   reluLayer
   dropout (0.5)
   fullyConnectedLayer (24)
   regressionLayer];

8. imageInputLayer ([rozmx rozmy 1])
   convolution2dLayer (3,55)
   reluLayer
   maxPooling2dLayer (2)
   fullyConnectedLayer (48)
   reluLayer
   dropout (0.5)
   fullyConnectedLayer (24)
   regressionLayer];

9. imageInputLayer ([rozmx rozmy 1])
   convolution2dLayer (3,55)
   reluLayer
   convolution2dLayer (1,55)
   reluLayer
   fullyConnectedLayer (48)
   reluLayer
   fullyConnectedLayer (24)
   regressionLayer];

10. imageInputLayer ([rozmx rozmy 1])
    convolution2dLayer (3,55)
    reluLayer
    convolution2dLayer (1,100)
    reluLayer
    fullyConnectedLayer (48)
    reluLayer
    fullyConnectedLayer (24)
    regressionLayer];

11. imageInputLayer ([rozmx rozmy 1])
    convolution2dLayer (3,55)
    reluLayer
    maxPooling2dLayer (2, 'Stride',2)
    convolution2dLayer (1,55)
    reluLayer
    fullyConnectedLayer (48)
    reluLayer
    fullyConnectedLayer (24)
    regressionLayer];

12. imageInputLayer ([rozmx rozmy 1])
    convolution2dLayer (3,55)
    reluLayer
    maxPooling2dLayer (2)
    convolution2dLayer (1,55)
    reluLayer
    fullyConnectedLayer (48)
    reluLayer
    dropout (0.5)
    fullyConnectedLayer (24)
    regressionLayer];

Pierwsza z uczonych sieci miała być siecią odniesienia – najprostszym wariantem, do którego później miały być porównywane kolejne przypadki. Zdecydowano się na filtry 3x3, ponieważ obrazy wejściowe są małe (macierze mają małe wymiary) i większe filtry mogłyby nie wyciągnąć wystarczająco dużo informacji. Liczba filtrów – 20 została dobrana eksperymentalnie. Dla mniejszych wartości wyniki były znacznie słabsze.

W kolejnym wariancie sieci liczba ta została zwiększona tak, aby można było sprawdzić wpływ ilości filtrów na jakość sieci. Ze względu na obserwacje, iż liczba filtrów równa 55 daje lepsze wyniki niż wersja wcześniejsza, zdecydowano się na pozostawienie takiej ilości filtrów w kolejnych badaniach.

W trzecim przypadku została dodana druga warstwa pełna z taką samą liczbą neuronów, co w pierwszej. Spróbowano również zmienić tę liczbę. Mniejsza dała dużo gorsze efekty, natomiast większa poprawiła wyniki, więc zdecydowano się na stworzenie czwartego wariantu, badającego właśnie ten element sieci.

Następnym krokiem było dodanie funkcji mającej teoretycznie poprawić wyniki – MaxPooling. W piątym wariancie została ona użyta z parametrami (2, 'Stride', 2). Pierwszy z nich oznacza wielkość okna, z którego brane są wartości – w tym przypadku 2x2, natomiast kolejna wielkość – krok, również wybrana jako 2, oznacza, że przesunięcie między pierwszym a drugim oknem wynosi dwie komórki. Analogicznie, jeśli chodzi o kolejne okna.

W wariancie szóstym zmianie uległy argumenty funkcji maxpooling. Zostały one zmienione na (2), oznacza to, że wielkość okna wciąż wynosiła dwa, natomiast krok został zmieniony na 1 – wartość domyślną.

W kolejnym przypadku zrezygnowano z funkcji maxpooling i zbadano wpływ warstwy z funkcją dropout. Jej argumentem jest część wszystkich neuronów, które zostają tymczasowo usunięte z sieci, dla lepszych wyników. Wartość ta została ustawiona na domyślną – 0,5. Natomiast wariant ósmy miał za zadanie sprawdzić działanie sieci z zastosowaniem zarówno operacji maxpooling, jak i dropout. Argument pierwszej z tych operacji ustawiono na 2, natomiast drugiej na 0,5.

Ze względu na to, że badaniu podlegała konwolucyjna sieć neuronowa, kolejnym przypadkiem była sieć, w której znajdowały się dwie warstwy. Budując te sieci wzorowano się na sieci AlexNet [5], która jest zdecydowanie większa i przeznaczona do znacznie bardziej skomplikowanych problemów, lecz jej struktura podobna jest do tej, którą chciano uzyskać w tym zadaniu. Jest to jeden z dwóch powodów, dla jakich w drugiej warstwie konwolucyjnej fitry są wymiarów 1x1, natomiast drugą przyczyną był błąd pojawiający się przy kompilacji programu przy innych wartościach. Liczba filtrów wybrana została taka sama, jak w pierwszej warstwie.

Z kolei w następnym (dziesiątym) przypadku właśnie ilość filtrów uległa zmianie. Sprawdzono czy otrzymywane wyniki są lepsze z mniejszą czy z większą liczbą filtrów i zdecydowano się na drugi wariant (100 filtrów).

Następny przypadek był poszerzony o operację MaxPooling z argumentami jak w przypadku piątym, a w ostatnim pojawiła się jeszcze operacja dropout, tak aby sprawdzić jaki wpływ mają te funkcje na sieć o dwóch warstwach konwolucyjnych. Ze względu na to, że przypadek, gdzie w drugiej warstwie konwolucyjnej było 100 filtrów nie przyniósł lepszych efektów – zrezygnowano z niego w dwóch ostatnich badaniach i ilość filtrów ustawiono na 55.

Badania na wszystkich sieciach i metodach wykonano dwukrotnie – raz dla jednego dnia poprzedzającego dzień przewidywany, drugi raz dla siedmiu dni wstecz.

Kolejnym krokiem było ustawienie opcji treningowych sieci. Nie skupiano się na tym etapie, wybrano tylko podstawowe opcje. Być może zagłębienie się w te ustawienia mogłoby przynieść jeszcze lepsze efekty. Opcja 'sgdm' (Stochastic Gradient Descent with Momentum) oznacza uaktualnianie przez sieć parametrów – wag i biasów, aby zminimalizować błąd sieci, posuwając się drobnymi krokami w kierunku ujemnego gradientu.

$$\theta_{n+1} = \theta_n - \alpha \nabla E(\theta_n)$$

Natomiast drugi parametr to liczba epok, która domyślnie ustawiona jest na 30, lecz była to zdecydowanie za mała liczba, ponieważ sieć była niedouczona. Eksperymentalnie wybrano więc liczbę 500 epok, lecz nie dla wszystkich przypadków była ona wystarczająca. W tabelach w dalszym etapie tej pracy oznaczono żółtym kolorem przypadki, gdy liczba epok została ustawiona na 600, ponieważ liczba 500, była wciąż za mała. Pojawił się wtedy problem związany z czasem uczenia sieci. Podczas gdy konfiguracje mało skomplikowanych sieci z prostą metodą przetwarzania – macierzową – uczyły się kilkanaście minut na komputerze osobistym o przeciętnych parametrach, czas uczenia tych najbardziej skomplikowanych wydłużał się do pięciu godzin, co stanowiło spore utrudnienie i przedłużenie badań. Warto jednak wspomnieć, że przy wykorzystaniu do uczenia sieci komputera osobistego o znacznie lepszych parametrach czas z pięciu godzin skracał się do trzydziestu minut. Widać więc, jak niewielka zmiana sprzętu może wpłynąć na wydajność badań.

Wracając jednak do sieci, po wybraniu opcji treningu, rozpoczynało się trenowanie sieci wywołane poleceniem *net = trainNetwork(xu,du,layers,options)*, którego argumenty były wcześniej opisywane. Po nauczeniu sieci następowało przewidywanie wartości *yt= predict(net,xt)* oraz wyliczenie błędów: średniego, maksymalnego oraz rzeczywistego. Ostatnim etapem było wyświetlenie wyników na wykresie.

Wyniki (wartości błędów) zebrano w tabelach, dla każdej metody zamiany sygnału oddzielnie. W tabelach kolorem jasno-pomarańczowym oznaczono sieci z jedną warstwą konwolucyjną, natomiast ciemno-pomarańczowym te z dwiema warstwami konwolucyjnymi, zaś kolorem brązowym sieci, które wymagały 600 epok uczenia, zamiast 500. Kolorem zielonym oznaczono sieci o najmniejszych błędach średnich danych testujących zarówno dla przypadku jednej, jak i dwóch warstw konwolucyjnych. Na czerwono zaś najmniejsze błędy maksymalne również dla danych testujących. Skrótem errt oznaczono błąd testujący, natomiast erru – błąd uczący. Z kolei błąd średni – err_mean, maksymalny – err_max, a absolutny – err_mae.

4.3. Metoda I – spektrogram

Pierwszą metodą zamiany sygnału 1D na 2D był spektrogram. Błędy sieci nauczonych danymi powstałymi w ten sposób, spisano w tabelach 4.1 (dane biorące do przewidywań jeden dzień wstecz) oraz 4.2 (dane biorące do przewidywań siedem dni wstecz).

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	4,55	4,17	59,79	63,78	797,42	747,58
2	4,06	3,92	71,98	60,02	718,73	697,74
3	3,62	3,50	40,90	47,99	637,41	619,05
4	3,81	3,69	39,76	47,05	682,01	661,02
5	4,48	4,41	46,53	49,13	818,41	800,05
6	3,83	3,71	47,23	51,12	679,38	663,64
7	4,18	4,04	40,47	49,49	734,47	713,48
8	4,02	4,01	47,54	4568,00	710,86	710,86
9	4,09	3,97	39,29	52,02	734,47	705,61
10	3,78	3,62	51,07	51,09	666,27	640,04
11	5,05	4,92	54,54	49,57	902,35	881,36
12	4,59	4,36	52,21	55,25	810,54	768,57

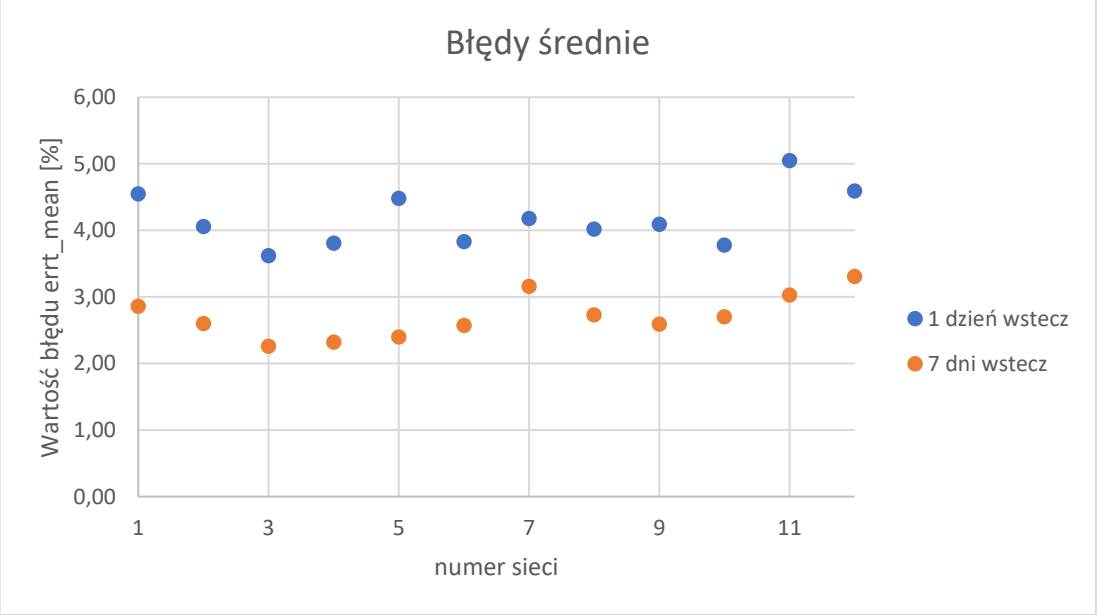
Tabela 4.1. Błędy sieci uczonej spektrogramem sygnału dla jednego dnia wstecz.

Z otrzymanych wyników można wywnioskować, że najlepszą siecią dla tej właśnie metody jest bardzo prosta sieć o jednej warstwie konwolucyjnej i dwóch warstwach pełnych o tej samej liczbie neuronów. Trzeba przyznać, że mimo iż dla wszystkich sieci (zależnie od ilości dni wstecz branych pod uwagę) wyniki są bardzo zbliżone, ta metoda dla obydwu przypadków prezentuje się najlepiej. Wyniki uzyskane po nauczeniu sieci z dwiema sieciami konwolucynymi nie są złe, jednak zauważalnie gorsze.

Warto zwrócić również uwagę na fakt, że najmniejsze błędy maksymalne nie zawsze pokrywają się z najmniejszymi błędami średnimi. Nie jest to zaskakujące, gdyż między tymi błędami istnieje mała zależność. Jednak wartości tych błędów są dość niepokojące. Mimo, iż na tle innych metod są dość niskie to w rzeczywistości błąd rzędu 40-50% jest niedopuszczalny. Powoduje, że przewidywania nie mają żadnej wartości. Trzeba jednak pamiętać, że wartość taka wystąpiła tylko raz, a jednak w większości przewidywanych danych była ona znacznie mniejsza.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	2,86	2,35	45,84	44,84	493,32	408,09
2	2,60	2,44	43,13	48,76	451,17	422,32
3	2,26	1,97	35,57	37,85	398,71	343,63
4	2,32	2,01	40,56	38,01	411,83	354,12
5	2,40	2,35	51,53	42,39	419,70	409,20
6	2,57	2,11	47,62	39,49	448,55	367,23
7	3,16	2,73	42,81	49,66	540,36	469,53
8	2,73	2,68	42,60	47,64	474,78	464,29
9	2,59	2,18	48,72	46,29	443,30	377,73
10	2,70	2,27	51,45	38,73	466,91	398,71
11	3,03	2,91	46,72	54,07	527,24	508,88
12	3,31	3,16	48,95	49,14	582,33	558,72

Tabela 4.2. Błędy sieci uczonej spektrogramem sygnału dla siedmiu dni wstecz.



Rysunek 4.3. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę przy zastosowaniu spektrogramu.

Z założenia wartością satysfakcjonującą, jeśli chodzi o błąd średni, byłoby 4%, jak widać tą metodą udało się uzyskać dużo bardziej zadawalający wynik.

Zdecydowanie lepsze wyniki otrzymano dla większej ilości dni wstecz branych pod uwagę przy przewidywaniu (rys 4.3). Spodziewano się takiego wyniku. Być może, gdyby jeszcze zwiększyć tę liczbę, można było uzyskać lepsze wyniki, aczkolwiek na pewno nie jest to zależność liniowa, a mogłoby to niekorzystnie wpłynąć na czas uczenia. W takim przypadku, jak tutaj, gdy dostęp do danych jest nieograniczony, warto skorzystać z tak prostego sposobu na polepszenie jakości sieci.

Jeśli chodzi o ilość warstw konwolucyjnych w sieci, to wyniki są zupełnie inne niż oczekiwano. Przede wszystkim zakładano, że sieć o dwóch warstwach konwolucyjnych poradzi sobie z zadanym problemem lepiej niż ta o jednej warstwie. Okazało się jednak, że obrazy wejściowe były mało skomplikowane i kolejna warstwa z tyloma filtrami wyodrębniała cechy, które nie były już tak istotne, a co za tym idzie, pogarszała wyniki. Okazało się również, że operacje maxpooling i dropout, które z założenia miały pomóc, wręcz przeciwnie – pogorszyły wyniki. Problem polegał na tym, że danych było stosunkowo mało, a te operacje mają znacznie lepsze zastosowanie przy dużych zbiorach danych.

4.4. Metoda II – falki

Następną metodą zamiany sygnały był rozkład na falki (transformacja falkowa). Wartości błędów sieci wytrenowanych na obrazach powstałych tym sposobem przedstawiono w tabelach 4.3 i 4.4.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	5,07	4,93	52,93	60,43	902,35	878,74
2	4,76	4,71	56,52	59,40	844,64	834,15
3	4,30	4,14	60,77	56,75	755,45	737,09
4	4,15	3,98	53,68	49,09	726,60	697,74
5	6,48	6,35	61,76	51,90	1132,14	1112,53
6	4,41	4,31	44,09	60,44	778,73	764,62
7	4,89	4,60	55,84	52,03	860,38	813,16
8	4,55	4,41	53,97	59,73	802,67	773,81
9	10,29	10,36	74,26	77,56	1775,84	1744,36
10	10,54	9,99	58,72	77,81	1804,69	1736,49
11	10,23	10,04	74,20	77,50	1754,85	1746,98
12	10,12	10,07	53,62	77,52	1765,35	1746,98

Tabela 4.3. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na falki dla jednego dnia wstecz.

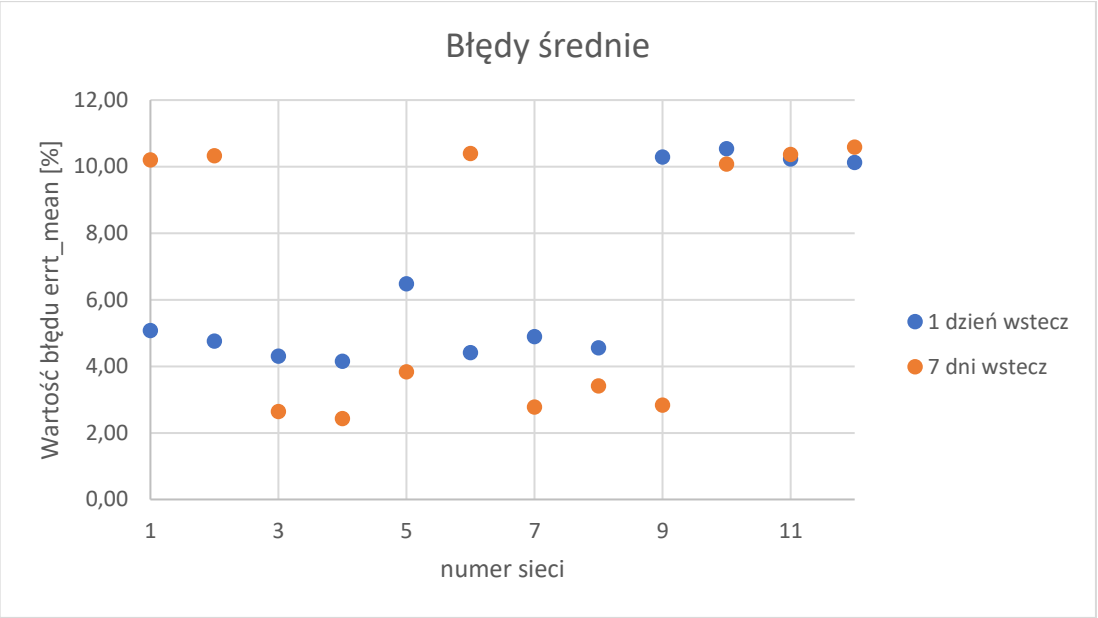
Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	10,20	10,00	74,10	77,40	1775,60	1745,10
2	10,33	10,03	57,41	77,49	1794,20	1741,74
3	2,64	2,42	43,83	46,86	459,04	422,32
4	2,43	2,37	42,19	51,75	430,19	417,07
5	3,83	3,66	47,89	53,50	658,40	634,79
6	10,39	10,02	77,68	67,03	1778,46	1744,36
7	2,77	2,77	49,01	52,17	485,27	480,03
8	3,41	3,24	40,61	52,76	600,69	566,59
9	2,83	2,87	55,40	49,86	495,77	501,01
10	10,08	10,07	58,42	77,47	1762,72	1746,98
11	10,36	10,02	63,99	77,47	1802,07	1739,12
12	10,59	9,99	63,06	78,01	1799,45	1739,12

Tabela 4.4. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na falki dla siedmiu dni wstecz.

W tej metodzie błędy są dużo bardziej zróżnicowane. Można zauważyć, że dla większości sieci z zastosowaniem dwóch warstw konwolucyjnych, błędy średnie wynoszą około 10%. Jest to bardzo duża wartość – niedopuszczalna, dlatego od razu dyskwalifikuje te sieci przy zastosowaniu rozkładu na falki. Jest to najprawdopodobniej spowodowane tym, że obraz powstały tą metodą przenosi stosunkowo mało informacji, więc zastosowanie kolejnej warstwy, która wybiera cechy, mimo że takich o dużym znaczeniu jest mało, skutkuje bardzo dużym wzrostem błędów.

Trzeba jednak przyznać, że dla sieci nr 9 z tabeli 4.3 wyniki są jak najbardziej zadowalające. To ukazuje pewną ciekawą właściwość przy uczeniu sieci. Cały proces uczenia, z naciskiem na wybór cech, jest automatyczny i w pewien sposób losowy. Użytkownik nie ma kontroli nad tym, co dzieje się podczas uczenia. Skutkuje to różnymi wynikami przy kilkukrotnym uczeniu sieci od nowa. Nie jest to regułą, ale być może po nauczeniu ponownie sieci, dla których wyniki były bardzo złe, możnaby uzyskać znacząco mniejsze błędy, lecz mogłoby to zadziałać również w drugą stronę. W wypadkach, gdy błędy były zbyt duże bądź sieć okazywała się być niedouczona (błąd uczący większy od testującego) badania przeprowadzano ponownie, lecz przeważnie uzyskiwane wyniki były zbliżone (nigdy wartości nie były dokładnie takie same).

Dla rozkładu na falki najlepiej sprawdziła się sieć z pojedynczą warstwą konwolucyjną, lecz o dwóch warstwach pełnych, ale o różnych liczbach neuronów. Wyniki nie odbiegają znacząco od pozostałych, ale są dobre, szczególnie dla przypadku, gdzie branych pod uwagę jest więcej dni wstecz do uczenia sieci.



Rysunek 4.4. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę przy zastosowaniu transformacji falkowej.

Ciekawe jest jednak porównanie błędów średnich dla różnych ilości dni wstecz branych pod uwagę. Podczas gdy w przypadku spektrogramu odległości pomiędzy odpowiednimi punktami były zbliżone (przebiegi równoległe), tutaj są one zupełnie porozrzucane i trudno doszukać się takiej prawidłowości. Znacznie więcej jest punktów mocno oddalonych – dużych błędów. Nie ulega jednak wątpliwości, że ponownie lepsze wyniki są dla sieci, w których danymi uczącymi jest kilka dni wstecz.

4.5. Metoda III – macierze

Trzecią i ostatnią metodą, którą przetwarzano sygnał 1D na 2D jest tworzenie macierzy, której jest aż sześć wariantów. Błędy sieci uczonej macierzami typu A (rys.3.3) zostały zestawione w tabelach 4.5 i 4.6.

Metoda tworzenia macierzy jest bardzo prosta, ponieważ nie wymaga żadnej transformacji, ale skutkuje to również brakiem analizy sygnału, a co za tym idzie mniejszą ilością danych, z których warstwy konwolucyjne mogą wyodrębniać cechy. Spośród wybranych najlepszych sieci nie można zauważyć żadnego schematu. Zdecydowanie sieci z jedną warstwą radzą sobie lepiej. Jak wcześniej wspomniano, w tej metodzie jest mało danych w prostej formie, a wybieranie kolejnych, drobnych

cech pogarsza wyniki. Bardzo odznacza się to przede wszystkim tam, gdzie do przewidywań brany jest tylko jeden dzień wstecz. Po dodaniu warstwy konwolucyjnej, błąd wzrósł ponad dwukrotnie.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	4,56	4,45	58,59	66,73	789,55	779,06
2	4,66	4,39	56,73	61,01	804,67	770,91
3	4,65	4,46	63,77	60,38	828,90	792,18
4	4,59	4,47	53,61	60,21	815,78	800,05
5	5,51	5,44	52,51	55,31	965,30	960,05
6	5,47	5,28	53,43	56,25	952,19	925,95
7	5,07	5,01	52,74	54,19	891,85	878,74
8	5,56	5,35	57,20	54,50	970,55	936,45
9	10,50	9,99	58,44	77,49	1823,05	1733,87
10	10,61	9,92	73,74	77,07	1830,92	1726,00
11	10,15	10,06	63,94	77,42	1775,84	1744,36
12	10,53	9,99	74,60	77,90	1791,58	1739,12

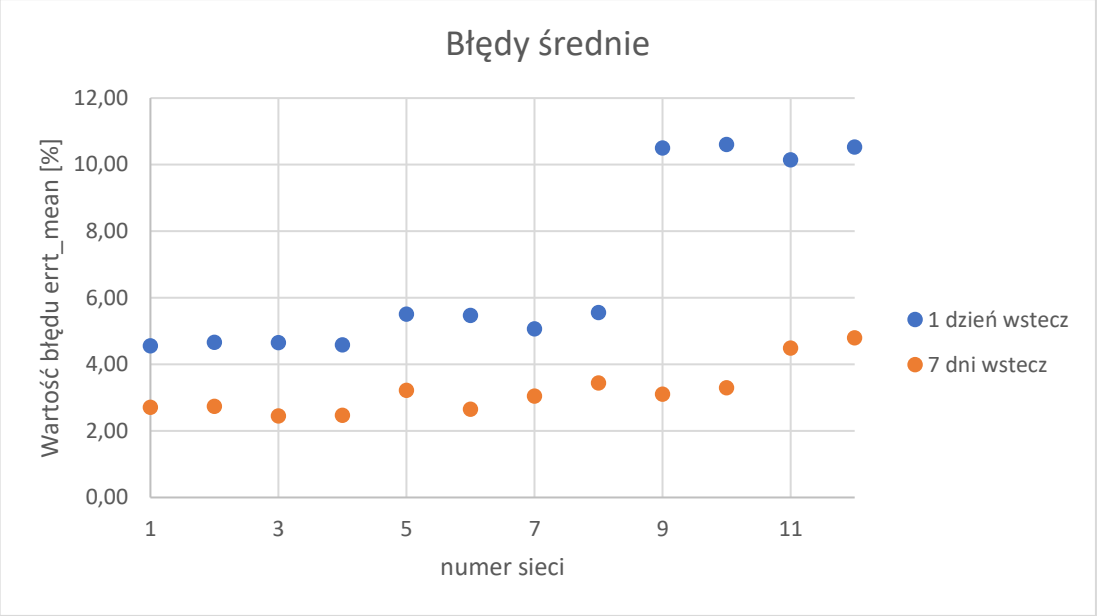
Tabela 4.5. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz A) dla jednego dnia wstecz.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	2,71	2,51	52,54	43,65	468,16	438,61
2	2,74	2,46	44,15	51,50	474,26	426,74
3	2,45	2,32	50,40	43,33	426,11	405,96
4	2,47	2,38	37,38	44,48	435,36	419,19
5	3,22	2,99	54,18	55,26	567,50	523,44
6	2,65	2,49	47,07	46,27	458,15	434,09
7	3,05	3,00	53,34	47,99	527,75	520,31
8	3,44	3,34	41,27	50,81	584,92	569,39
9	3,11	3,04	46,45	51,20	548,67	531,46
10	3,30	3,11	48,59	56,27	555,16	533,53
11	4,49	4,38	57,62	66,29	778,06	765,80
12	4,80	4,71	64,72	69,02	821,03	815,78

Tabela 4.6. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz A) dla siedmiu dni wstecz.

Metoda ma na pewno gorsze wyniki niż dwie poprzednie i podczas gdy dla pierwszego przypadku, wartość błędów przestaje być zadowalająca, dla drugiego wariantu wyniki wciąż są bardzo dobre.

Ponownie, wbrew oczekiwaniom funkcje dropout oraz maxpooling sprawiły, że wyniki tylko się pogorszyły.



Rysunek 4.5. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę dla macierzy A.

Gdy porównane zostaną błędy średnie ze względu na ilość dni wstecz branych pod uwagę przy przewidywaniu, widać niemalże idealną zależność. Zwiększenie tej liczby skutkuje przesunięciem punktów w dół (zmniejszeniem błędu), lecz zależności między poszczególnymi sieciami zostają bardzo wiernie zachowane.

Kolejnym sposobem tworzenia macierzy była macierz B (rys.3.3). Wyniki tej metody zostały przedstawione w tabelach 4.7 oraz 4.8.

Podobnie jak w przypadku macierzy A wyniki drugiego wariantu są znacznie lepsze i zadowalające, natomiast błędy uzyskane w pierwszym badaniu wykraczają poza założony próg. Trudno wybrać, która sieć sprawdza się w tym wypadku najlepiej. Zdecydowanie można powiedzieć, że sieć nr 4 (jedna warstwa konwolucyjna i dwie pełne o różnej liczbie neuronów) przyniosła najlepszy wynik, jednak jest on niewiele lepszy od sieci nr 2. Z całą pewnością można stwierdzić, że metoda ta nie jest zła, ale zasługą niskich błędów jest branie pod uwagę większej liczby dni wstecz, a nie specjalna budowa sieci.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	4,67	4,47	61,99	72,39	815,78	786,93
2	4,60	4,37	62,43	69,67	805,29	771,19
3	4,63	4,46	50,88	63,84	80791,48	786,93
4	4,73	4,51	64,92	58,37	836,77	800,05
5	5,40	5,31	57,27	54,80	944,32	933,82
6	5,26	5,13	52,05	57,51	915,46	897,10
7	5,10	5,02	47,90	53,77	904,97	891,85
8	5,30	5,30	58,77	61,91	933,82	925,95
9	10,07	10,07	66,83	77,40	1754,85	1746,98
10	6,11	6,28	50,84	54,53	1067,60	1093,83
11	10,10	10,07	62,61	77,47	1752,23	1749,61
12	10,44	10,02	74,53	77,84	1781,08	1741,74

Tabela 4.7. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz B) dla jednego dnia wstecz.

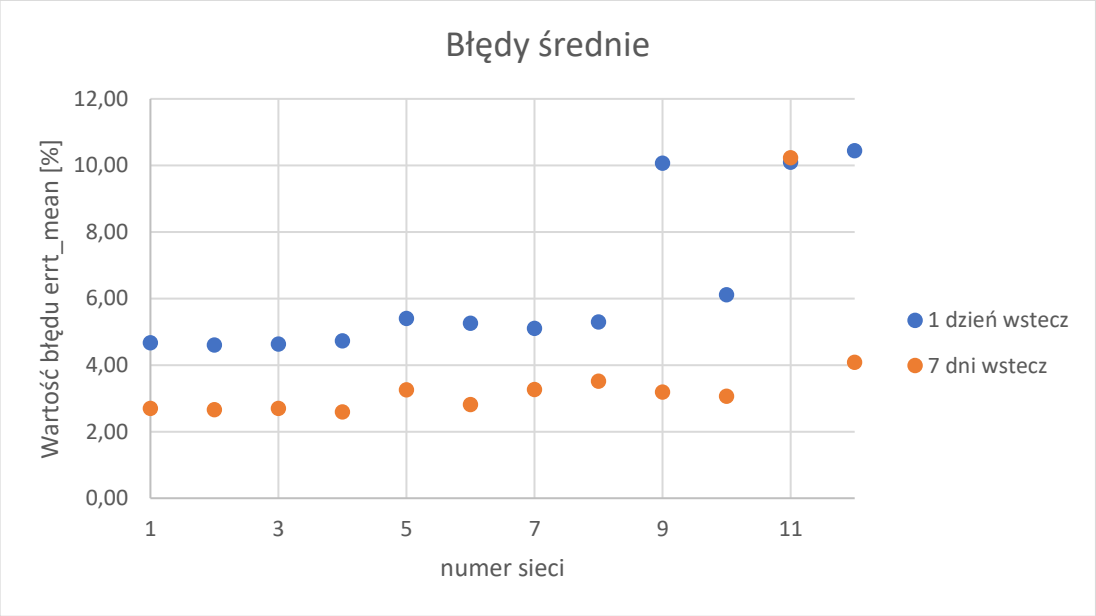
Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	2,70	2,62	44,61	52,54	471,76	462,78
2	2,66	2,50	54,78	44,49	462,27	435,37
3	2,70	2,45	41,61	50,55	470,81	426,85
4	2,59	2,63	33,53	45,66	464,05	467,73
5	3,26	3,24	49,75	49,54	574,19	572,85
6	2,81	2,72	49,88	45,84	498,99	478,48
7	3,27	3,08	49,98	52,44	557,68	526,37
8	3,52	3,42	47,86	51,28	624,87	601,33
9	3,19	3,10	41,54	47,49	564,38	551,42
10	3,06	2,89	52,66	52,75	526,86	499,90
11	10,23	10,05	62,80	77,67	1752,27	1745,97
12	4,08	4,02	58,36	61,38	710,61	695,78

Tabela 4.8. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz B) dla siedmiu dni wstecz.

W najlepszej sieci, jaką udało się uzyskać w tym badaniu, błąd maksymalny jest niski w porównaniu do pozostałych, lecz jest to raczej przypadek niż przewidziane działanie sieci. Z kolei mamy tutaj do

czynienia również z bardzo wysokimi błędami – około 77%. Mimo, że jest on jednorazowy dla każdej sieci, to trzeba przyznać, że sieć o tak piorunująco dużym błędzie jest bezużyteczna.

Także w metodzie tej nie można się doszukać dobroczynnego wpływu operacji dropout i maxpooling. Jednak jeśli chodzi o sieci z dwiema warstwami konwolucyjnymi, zdecydowanie najlepiej spisała się sieć, gdzie w drugiej warstwie było znacznie więcej filtrów. Jest to dość zaskakujące, ponieważ jest tutaj niewiele, mało skomplikowanych danych, więc taki wariant nie powinien się sprawdzić. Mimo, że spisała się najlepiej; jej błędy są zbyt duże. Mając do dyspozycji sieci o błędach kilka procent niższych, nie ma sensu sięgać po tę sieć.



Rysunek 4.6. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę dla macierzy B.

Na wykresie błędów średnich w zależności od ilości dni wstecz branych pod uwagę, podobnie jak przy wykorzystaniu macierzy A, widać zależności między metodami. Zasada ta została zachwiana dla sieci o dwóch warstwach konwolucyjnych, lecz dla pozostałych sieci jest bardzo dobrze zauważalna.

Kolejnym sposobem zamiany sygnału było tworzenie macierzy A’ (rys.3.3). Wyniki sieci uzyskane tą metodą spisano w tabelach 4.9 oraz 4.10.

Po przeanalizowaniu danych widać, że są one bardzo podobne do dwóch poprzednich przypadków. Drobne różnice wynikają raczej z różnych cykli uczenia, a nie innych metod. Spodziewano się, że dla każdego rodzaju macierzy wyniki będą zróżnicowane i będzie można wybrać metodę najlepszą. Jednak okazało się, że niezależnie od rodzaju macierzy, metoda ta zwraca podobne wyniki.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	4,58	4,50	57,13	61,76	802,67	789,55
2	4,94	4,35	59,41	60,87	855,13	768,57
3	4,33	4,24	54,14	55,49	773,81	752,83
4	4,48	4,42	45,47	57,48	792,18	784,31
5	5,38	5,32	50,71	58,85	949,56	928,58
6	5,36	5,31	55,71	55,44	925,95	931,20
7	5,35	4,96	53,94	55,44	928,58	878,74
8	5,44	5,40	47,40	54,35	946,94	944,32
9	10,22	10,04	54,13	77,29	1770,59	1746,98
10	6,42	6,33	49,65	55,65	1106,95	1109,57
11	10,40	10,02	64,25	77,76	1786,33	1741,74
12	10,59	9,97	66,98	77,57	1833,55	1731,25

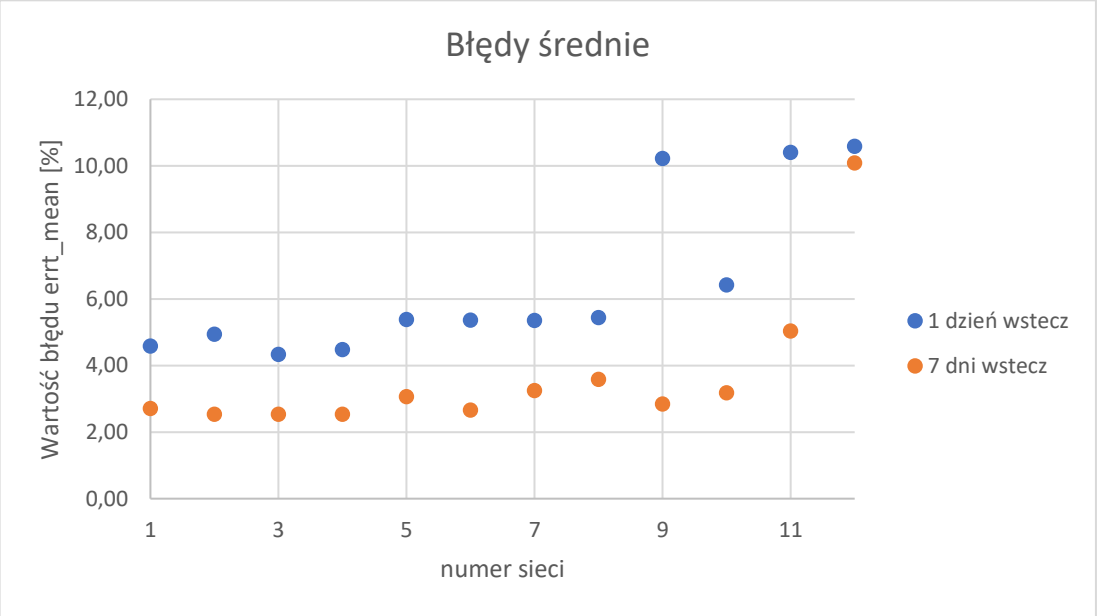
Tabela 4.9. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz Ap) dla jednego dnia wstecz.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	2,71	2,65	54,42	46,92	479,89	463,79
2	2,54	2,53	44,11	53,11	445,62	441,09
3	2,54	2,37	43,39	47,35	443,08	415,55
4	2,54	2,34	38,23	52,17	437,87	404,49
5	3,06	2,99	53,27	47,75	544,49	518,74
6	2,66	2,53	45,74	48,33	459,63	437,19
7	3,25	3,13	46,51	54,97	567,56	544,19
8	3,58	3,37	46,51	51,47	616,71	586,32
9	2,84	2,79	56,84	48,12	493,99	482,12
10	3,18	3,01	52,34	45,81	546,57	523,76
11	5,04	5,00	70,69	70,51	862,56	864,76
12	10,09	10,00	77,32	19,20	1719,54	1754,60

Tabela 4.10. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz Ap) dla siedmiu dni wstecz.

Jak w poprzednich przypadkach, wyniki dla pierwszego wariantu są znacznie gorsze, nie spełniające założeń. Natomiast gdy zwiększona została liczba dni wstecz branych pod uwagę, wyniki znów się poprawiły.

Tak, jak w poprzednich przypadkach, wykres błędów średnich przedstawia równe zależności między metodami – odpowiednio lepsze wyniki są dla drugiego wariantu.



Rysunek 4.7. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę dla macierzy Ap.

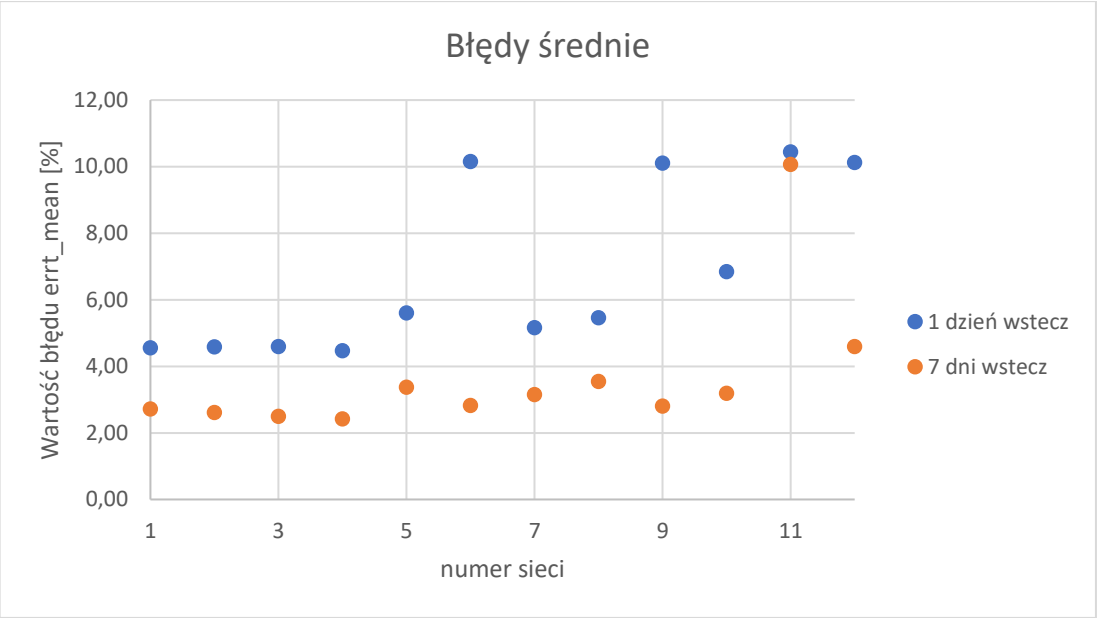
Ze względu na to, że wszystkie warianty metody macierzowej są do siebie bardzo podobne, pierwsze trzy zostały szczegółowo omówione osobno, lecz kolejne zostaną opisane wspólnie po przedstawieniu wszystkich tabel (4.11 – 4.16) i wykresów (4.8 - 4.10).

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	4,55	4,51	51,67	57,70	807,91	797,42
2	4,58	4,44	59,81	67,45	807,91	784,31
3	4,59	4,48	63,10	58,17	802,67	784,31
4	4,47	4,39	44,70	59,39	784,31	773,81
5	5,60	5,21	51,70	60,78	975,79	912,84
6	10,15	10,05	63,75	77,20	1767,97	1746,98
7	5,16	4,99	48,95	55,95	891,85	873,49
8	5,46	5,39	54,09	61,85	965,30	946,94
9	10,10	10,08	74,32	77,62	1744,36	1749,61
10	6,84	6,61	49,82	55,04	1196,13	1162,03
11	10,44	10,00	63,84	77,32	1807,32	1739,12
12	10,12	10,07	77,55	66,95	1731,25	1752,23

Tabela 4.11. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz Bp) dla jednego dnia wstecz.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	2,72	2,62	47,85	52,09	473,66	460,81
2	2,61	2,58	44,80	51,94	458,97	452,50
3	2,50	2,33	38,91	52,03	432,42	405,40
4	2,42	2,32	36,79	49,40	417,90	403,49
5	3,37	3,02	64,05	61,86	571,53	520,27
6	2,82	2,71	48,74	50,03	488,08	471,25
7	3,15	3,02	49,88	51,06	543,77	528,34
8	3,54	3,32	52,65	56,21	612,08	572,75
9	2,80	2,77	48,70	53,23	486,20	484,87
10	3,19	2,91	49,24	51,02	538,49	502,01
11	10,07	10,04	77,22	62,38	1755,09	1745,40
12	4,59	4,39	64,71	60,01	789,30	757,78

Tabela 4.12. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz Bp) dla siedmiu dni wstecz.



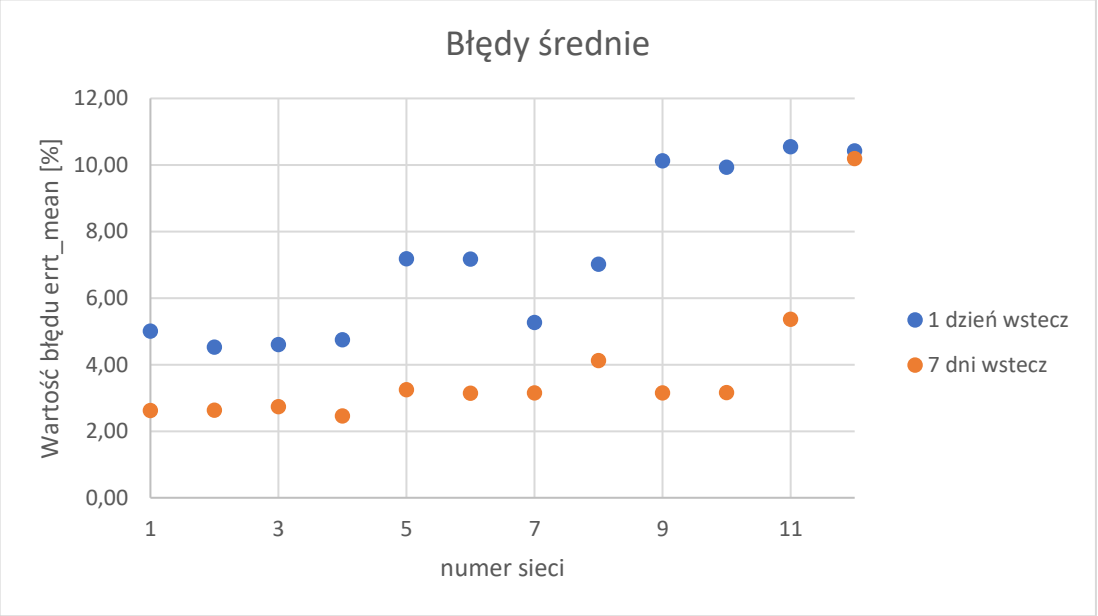
Rysunek 4.8. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę dla macierzy Bp.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	5,01	4,68	62,32	69,14	883,98	831,52
2	4,53	4,45	60,87	60,52	802,67	786,93
3	4,60	4,45	55,50	57,32	810,54	786,93
4	4,75	4,52	52,22	60,33	839,39	807,91
5	7,18	6,70	62,30	59,40	1227,61	1167,28
6	7,17	6,74	53,58	62,72	1240,73	1177,77
7	5,27	5,06	52,52	54,49	912,84	886,61
8	7,02	6,89	52,69	60,94	1222,36	1206,63
9	10,12	10,08	64,31	77,96	1739,12	1749,61
10	9,93	9,85	73,02	76,40	1718,13	1710,26
11	10,55	9,98	77,44	74,14	1817,81	1736,49
12	10,42	10,01	62,83	77,63	1794,20	1739,12

Tabela 4.13. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz C) dla jednego dnia wstecz.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	2,62	2,56	38,68	50,52	462,11	449,13
2	2,63	2,55	41,14	51,03	454,70	447,21
3	2,74	2,58	51,42	43,97	478,63	454,27
4	2,46	2,39	36,14	50,42	428,56	415,92
5	3,25	3,24	52,45	55,03	571,63	558,47
6	3,14	2,88	49,26	53,56	537,22	496,26
7	3,15	3,06	52,91	53,31	547,31	532,08
8	4,12	3,74	55,75	55,07	710,70	647,86
9	3,15	3,12	52,04	54,10	547,77	542,77
10	3,16	3,05	54,24	53,38	550,12	533,38
11	5,36	5,33	66,12	65,68	936,89	931,13
12	10,19	10,06	64,05	77,60	1759,84	1744,98

Tabela 4.14. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz C) dla siedmiu dni wstecz.



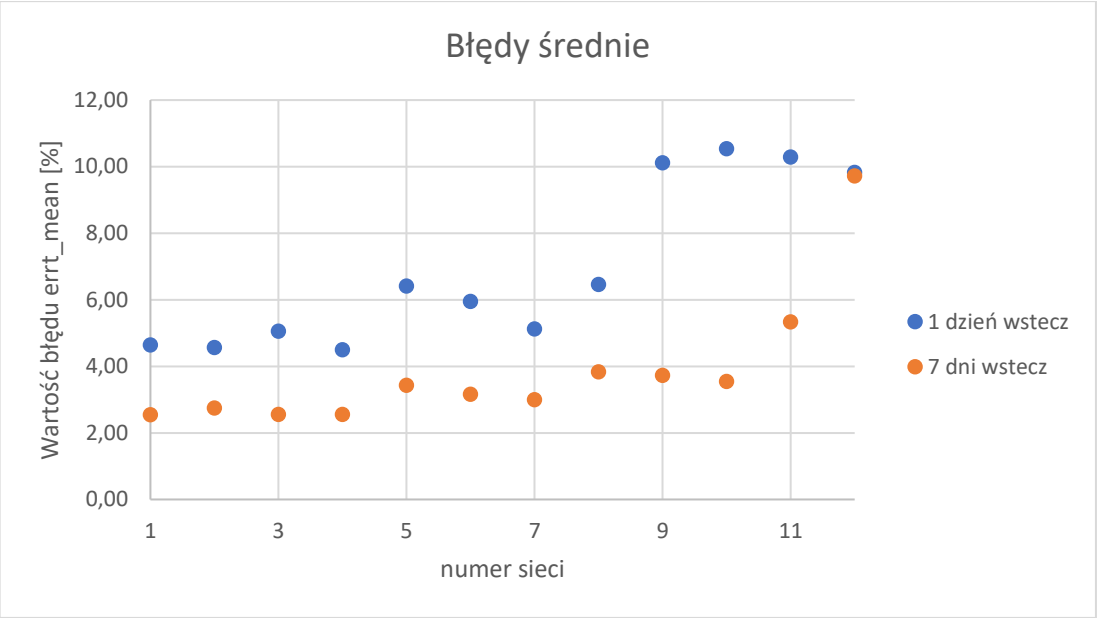
Rysunek 4.9. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę dla macierzy C.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	4,64	4,61	56,75	63,74	823,65	818,41
2	4,56	4,31	62,12	63,53	797,42	763,32
3	5,05	4,97	60,30	56,44	878,74	876,12
4	4,50	4,44	43,38	58,96	807,91	789,55
5	6,41	6,35	43,92	54,85	1143,67	1127,93
6	5,95	5,88	54,90	56,10	1043,99	1038,75
7	5,12	5,02	53,03	55,14	894,48	881,36
8	6,46	6,42	55,77	55,88	1125,31	1125,31
9	10,11	10,06	52,65	77,27	1765,35	1746,98
10	10,54	9,98	74,32	77,64	1799,45	1736,49
11	10,29	10,04	67,85	77,85	1762,72	1746,98
12	9,83	10,13	74,28	77,58	1702,39	1757,48

Tabela 4.15. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz D) dla jednego dnia wstecz.

Sieć	errt_mean [%]	erru_mean [%]	errt_max [%]	erru_max [%]	errt_mae [MW]	erru_mae [MW]
1	2,54	2,52	44,28	53,96	437,96	439,24
2	2,75	2,55	51,27	54,97	468,71	440,85
3	2,55	2,39	42,50	53,24	445,05	415,20
4	2,55	2,36	51,63	41,57	448,70	411,38
5	3,43	3,22	50,36	56,17	587,68	554,67
6	3,16	3,01	47,36	51,31	545,22	523,67
7	3,00	2,99	53,74	44,42	527,07	520,35
8	3,83	3,71	53,32	53,97	658,31	633,05
9	3,73	3,18	53,60	57,74	640,92	547,86
10	3,54	3,22	47,47	55,25	625,32	562,36
11	5,33	5,23	59,86	62,39	941,27	913,00
12	9,72	9,33	75,07	53,75	1684,05	1624,54

Tabela 4.16. Błędy sieci uczonej obrazami powstałymi poprzez rozkład sygnału na macierze (macierz D) dla siedmiu dni wstecz.



Rysunek 4.10. Błędy średnie dla poszczególnych sieci w zależności od ilości dni wstecz branych pod uwagę dla macierzy D.

Niezależnie od drobnych różnic między poszczególnymi wariantami metody macierzowej, jednoznacznie można stwierdzić, że sposób ułożenia poszczególnych komórek z wartościami obciążenia nie ma znaczenia w tej metodzie. Z badań wynika, że najmniejszy błąd uzyskano stosując typ macierzy Bp, lecz należy być ostrożnym ze stawianiem tezy, że jest to zasługa konkretnego ułożenia danych, a bardziej przypadek wynikający z danego cyklu uczenia. Wysoce prawdopodobne, że gdyby przeprowadzić badania ponownie, wartości delikatnie uległyby zmianie i inny wariant mógłby okazać się korzystniejszy.

Z obserwacji wynika, że metoda macierzowa daje bardzo dobre rezultaty, gdy bierze się pod uwagę siedem dni wstecz, zamiast jednego. Wyniki są porównywalne z metodą falkową. Jednak tutaj objawia się wyższość metody macierzowej – jej czas uczenia jest znacznie krótszy niż pozostałych.

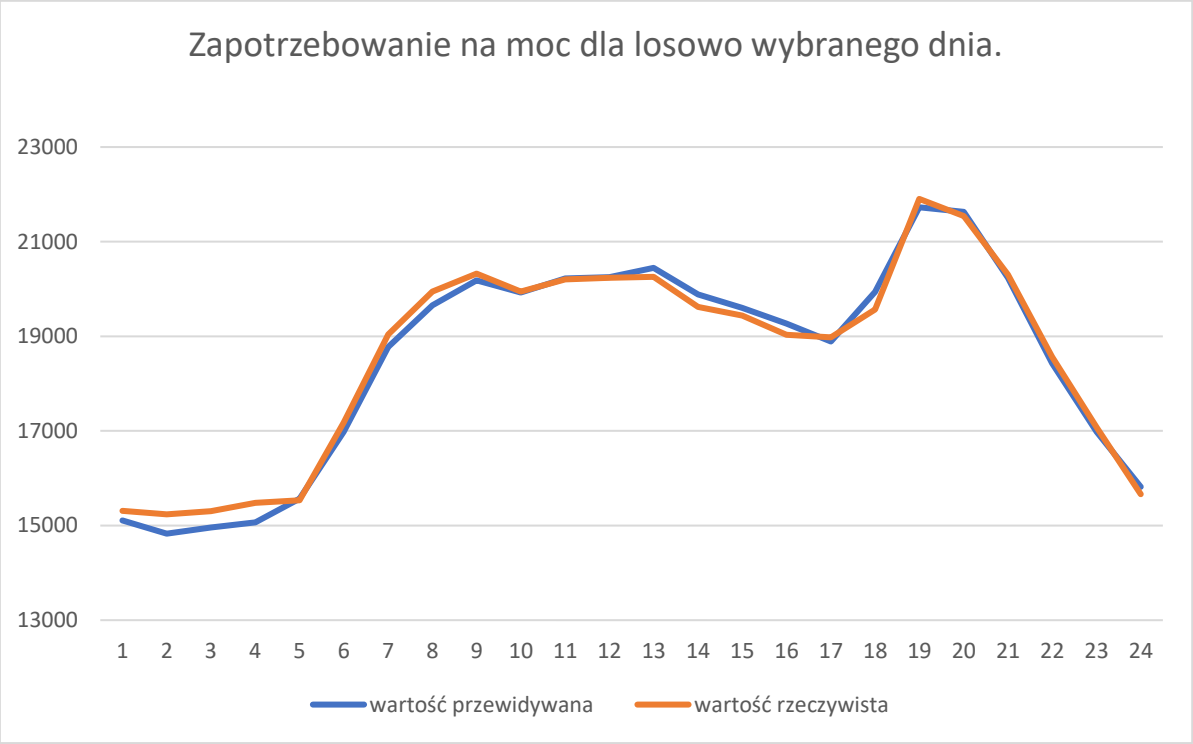
Na wykresach (4.5 - 4.10) widać również jasne zależności między różnymi sieciami. Najczęściej sieć nr 4 dawała najlepsze wyniki. Wydaje się być najbardziej odpowiednia do tej właśnie metody.

Błędy maksymalne nie wyróżniają się na tle pozostałych. Wartości te krążą wokół 50%, lecz dla lepszych sieci schodzą do 40%.

Jak już wcześniej zauważono, dodatkowa warstwa konwolucyjna nie przyniosła żadnych korzyści. Nic w tym dziwnego, ponieważ w tej metodzie wprowadzane obrazy są bardzo proste i zawierają mało informacji, więc kolejne filtry doszukują się mało istotnych szczegółów. Z tego samego powodu operacje dropout i maxpooling nie mają tutaj zastosowania. W żadnym przypadku nie poprawiły znacząco wyników.

4.6. Wyniki

Celem pracy było przebadanie różnych sieci i metod przetwarzania sygnału, a następnie wykonanie prognozy zapotrzebowania na energię w Polsce na dany dzień. Po przeprowadzeniu badań i wybraniu najlepszej wariacji – sieci oraz metody umożliwiono wprowadzenie do programu danych z dowolnie wybranych siedmiu dni i wykonanie prognozy. Dokonać tego można w funkcji *siec_cnn_regr*. Przykładowa prognoza dla losowo wybranego dnia została przedstawiona na rysunku 5.1, natomiast w tabeli 5.1 umieszczono dokładne wartości zarówno przewidywań jak i rzeczywistego chwilowego zapotrzebowania. Została ona wykonana za pomocą najlepszej sieci wykorzystującej metodę spektrogramu (tabela 4.2)



Rysunek 5.1. Zapotrzebowanie na moc dla losowo wybranego dnia.

Patrząc na uzyskane wyniki, zdecydowanie można stwierdzić, że są one zadowalające. Maksymalny błąd absolutny dla tego przypadku wynosi 416 MW, co daje jedynie 1,9% maksymalnej wartości zapotrzebowania. Największe różnice widoczne są na początku wykresu (pierwsze cztery godziny), natomiast w dalszej części wykresy się pokrywają, nie licząc delikatnych odchyleń. Nie da się zaobrserwować czy wartości przewidywane są generalnie wyższe czy niższe od rzeczywistych, więc

można stwierdzić, że wartości prognozowane oscylują wokół tych rzeczywistych. Dzięki temu wartość średnia obciążeń przewidywanych i rzeczywistych będzie zbliżona, a może być ona wykorzystana do przewidzenia ogólnej wartości zapotrzebowania na moc.

Wartość rzeczywista	Wartość przewidywana
15310	15109
15237	14829
15306	14962
15482	15066
15535	15571
17176	16981
19038	18775
19943	19656
20323	20184
19944	19923
20201	20226
20234	20248
20258	20449
19620	19885
19437	19604
19034	19272
18980	18891
19570	19938
21904	21727
21541	21632
20306	20238
18568	18431
17077	16983
15664	15820

Tabela 5.1. Chwilowe rzeczywiste wartości zapotrzebowń na moc oraz ich przewidywania dla losowo wybranego dnia.

5. Wnioski i podsumowanie

Praca dotyczyła rozwiązania problemu predykcji obciążeń elektroenergetycznych w KSE przy użyciu sieci konwolucyjnych. Opracowano dwanaście modeli sieci konwolucyjnych wykorzystujących trzy metody tworzenia macierzy z wektora obciążeń elektroenergetycznych. Przeprowadzono szereg badań eksperymentalnych na rzeczywistych danych pomiarowych z KSE opisujących wcześniej wspomniane obciążenia.

Należy zaznaczyć, że wykorzystywane dane nie są wystarczająco dokładne, aby móc otrzymane przewidywania wykorzystywać do celów technicznych – np. opierać na nich działanie elektrowni. Mogą więc posłużyć np. jako poglądowe przewidywania powszechnie udostępnione.

Otrzymane wyniki przewidywań zdecydowanie spełniają oczekiwania, jednak nie zawsze są one tak dobre, jak na rysunku 5.1. Zdarza się bowiem, że prognoza znacznie różni się od wartości rzeczywistych. Taka sytuacja może mieć miejsce np. gdy dzień wolny od pracy wypada w ciągu tygodnia pracy, a sieć nie ma takich informacji, ponieważ nie da się tego zaobserwować na podstawie wartości zapotrzebowania z siedmiu wcześniejszych dni. Może się również zdarzyć, że wystąpi bardzo duży błąd maksymalny w danym przewidywaniu i okaże się ono bezużyteczne. Jednak średnie wyniki przewidywań sięci są bardzo dobre.

Zgodnie z oczekiwaniami metoda przetwarzania obrazu za pomocą spektrogramu okazała się być najlepsza. Nic w tym dziwnego, ponieważ obraz jest w niej najbardziej zróżnicowany, jest największych rozmiarów i zawiera najwięcej informacji. Warto jednak zwrócić uwagę na to, że mimo, iż wyniki były dla tej metody zauważalnie lepsze, nie była to diametralna różnica. Niestety użycie tej metody skutkowało dłuższym czasem uczenia. Zależnie od potrzeb, możnaby również skorzystać z metody trzeciej, aby zaoszczędzić czas przeznaczony na uczenie.

Z kolei najlepsza architektura sieci okazała się być znacznie prostsza od spodziewanej. Wbrew oczekiwaniom, druga warstwa konwolucyjna znacznie pogorszyła wyniki we wszystkich metodach. Najwidoczniej macierze wejściowe miały zbyt małe rozmiary dla takiej sieci CNN i dodatkowe wyciąganie cech powodowało większe szkody – skupianie się na mało istotnych szczegółach.

Bardzo ciekawą rzeczą jest również fakt, że operacje, które powinny korzystnie wpłynąć na wyniki sieci, nie przyniosły zbyt dobrych rezultatów. Najwidoczniej zmniejszanie wymiarów dość małych już obrazów wejściowych i eliminowanie neuronów nie wpływają dobrze na tak nieskomplikowaną sieć.

Czas uczenia poszczególnych sieci wahał się od kilku minut do nawet pięciu godzin. Istotny wpływ na długość uczenia miała nie tylko metoda przetwarzania sygnału, ale również wielkość sieci. Sieci o dwóch warstwach konwolucyjnych uczyły się zdecydowanie dłużej. Po przeprowadzaniu tych samych badań na lepszym komputerze czas uczenia zdecydowanie się zmniejszył, natomiast gdyby zastosować do obliczeń procesor graficzny, czas uczenia mógłby znacząco się skrócić.

Bibliografia

[1] Goodfellow I., Bengio Y., Courville A., Deep Learning, MIT Press, 2016. [dostęp: 26.11.2017]. URL: http://www.deeplearningbook.org/.

[2] Jankiewicz Z., System elektroenergetyczny, 2016, [dostęp: 19.12.2017]. URL: http://www.bezel.com.pl/index.php/system-elektroenergetyczny/system-elektroenergetyczny.

[3] Kaiming H., Xianyu Z., Shaoqing R., Jian S., Delving Deep into Rectifiers: Surpassing Human – Level Performance on ImageNet Classification. [dostęp: 16.12.2017]. URL: https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/He_Delving_Deep_into_ICCV_2015_paper.pdf.

[4] Krizhevsky A., Sutskever I., Hinton G., ImageNet Classification with Deep Convolutional Neural Networks, [dostęp: 05.12.2017]. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[5] Mathworks, Alexnet, [dostęp: 11.12.2017]. URL: https://www.mathworks.com/help/nnet/ref/alexnet.html.

[6] Odrzywołek K., Wykorzystanie głębokich sieci neuronowych w weryfikacji mówcy, Akademia Górniczo-Hutnicza, 2016.

[7] Osowski S., Sieci neuronowe do przetwarzania informacji, praca magisterska, Oficyna Wydawnicza Politechniki Warszawskiej, 2013.

[8] Simonyan K., Zisserman A., Very Deep Convolutional Networks for Large – Scale Image Recognition, ICLR, 2015, [dostęp: 06.12.2017]. URL: https://arxiv.org/pdf/1409.1556.pdf

[9] Smith S., Cyfrowe przetwarzanie sygnałów, BTC, 2007.

[10] Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, 15, 1929-1958, 2014, [dostęp: 11.12.2017]. URL: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

Spis symboli i skrótów

A – macierz powstała metodą jak na rys. 3.3.

Ap – macierz powstała metodą jak na rys. 3.3.

B – macierz powstała metodą jak na rys. 3.3.

Bp – macierz powstała metodą jak na rys. 3.3.

C – macierz powstała metodą jak na rys. 3.3.

D – macierz powstała metodą jak na rys. 3.3.

errt_mae – błąd średni absolutny testujący

errt_max – błąd maksymalny testujący

errt_mean – błąd średni testujący

erru_mae – błąd średni absolutny uczący

erru_max – błąd maksymalny uczący

erru_mean – błąd średni uczący

{x,d} – para danych uczących

Spis rysunków

Spis tabel

5.1. Chwilowe rzeczywiste wartości zapotrzebowń na moc oraz ich przewidywania dla losowo wybranego dnia.

Spis załączników

1. Płyta CD z programami realizowanymi podczas badań.