POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Praca dyplomowa magisterska

Segmentacja obrazów hiperspektralnych przy użyciu trójwymiarowych konwolucyjnych sieci neuronowych

autor: Marcin Pszczółka

kierujący pracą: dr inż. Jakub Nalepa

Gliwice, grudzień 2019

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 6 grudnia 2019

.....
(podpis)

.....
(poświadczenie wiarygodności
podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Segmentacja obrazów hiperspektralnych przy użyciu trójwymiarowych konwolucyjnych sieci neuronowych" spełnia wymagania formalne pracy dyplomowej magisterskiej.

Gliwice, dnia 6 grudnia 2019

.....

(podpis promotora)

Spis treści

Rozdział 1

Wstęp

W obecnych czasach dzięki rozwojowi techniki możliwe jest gromadzenie coraz to większej ilości danych. Aparatura pomiarowa staje się bardziej dokładna, tańsza oraz łatwiej dostępna, a sam proces pozyskiwania danych często bywa automatyzowany. W odpowiedzi na ciągle rosnącą ilość danych powstają coraz to bardziej efektywne sposoby ich przetwarzania. Najczęściej bazują one na uczeniu maszynowym, czyli dziedzinie łączącej w sobie takie obszary jak informatyka, statystyka czy robotyka. Podkategorią uczenia maszynowego pozwalającą na znajdowanie w danych bardziej złożonych zależności przy jednoczesnym mniejszym zaangażowaniu ekspertów domenowych jest uczenie głębokie. Wykorzystuje ono między innymi sztuczne sieci neuronowe, czyli struktury inspirowane działaniem ludzkiego mózgu. Do wzrostu popularności tej techniki w ostatnich latach przyczynił się także rozwój procesorów graficznych [25].

Uczenie głębokie znalazło zastosowanie między innymi w takich poddziedzinach nauki jak rozpoznawanie mowy [26], rozpoznawanie obrazów [19], przetwarzanie języka naturalnego [44], silniki rekomendacji [11], czy analiza obrazów medycznych [41].

Dzięki rozpoznawaniu mowy możliwa jest obsługa urządzeń za pomocą głosu, co często jest wykorzystywane przez osoby niepełnosprawne, dla których inne interfejsy interakcji z urządzeniem są niedostępne. Innym przykładem zastosowania może być odbywająca się w czasie rzeczywistym translacja komunikatów głosowych pomiędzy językami naturalnymi (ang. *speech-to-speech*).

Silniki rekomendacji zostały spopularyzowane między innymi przez serwisy dostarczające szeroko pojętą rozrywkę (np. muzykę, seriale, czy filmy). W oparciu o wcześniejsze zachowania i wybory użytkownika, pozwalają one z dużym prawdopodobieństwem przewidzieć, jakimi innymi treściami użytkownik może być zainteresowany.

W analizie tekstu pisanego, głębokie uczenie jest wykorzystywane już od wielu lat. Metody eksploracji tekstu (ang. *text mining*) pozwalają odkryć wzorce między innymi w reklamacjach przesyłanych przez klientów czy oświadczeniach lekarskich, co pozwala na ich automatyczne przetwarzanie.

W kontekście analizy obrazów medycznych, metody głębokiego uczenia wspierają takie czynności jak monitorowanie stanu zdrowia pacjentów, badania przesiewowe zdjęć rentgenowskich, czy analiza obrazów mikroskopowych. Dzięki takim narzędziom lekarze są w stanie pracować sprawniej i szybciej diagnozować potencjalne zagrożenia.

Niniejsza praca skupia się na wykorzystaniu głębokich sieci neuronowych w analizie obrazów hiperspektralnych. W odróżnieniu od standardowego aparatu fotograficznego, który rejestruje informacje o natężeniu światła dla trzech długości fal elektromagnetycznych, odpowiadających kolorom czerwonemu, zielonemu i niebieskiemu, przy fotografii hiperspektralnej rejestrowanych może być od kilkudziesięciu do kilkuset różnych długości fali z różnych zakresów promieniowania elektromagnetycznego, np. światła widzialnego, dalekiej i bliskiej podczerwieni czy ultrafioletu. Dodatkowe informacje mogą okazać się bardzo pomocne przy identyfikacji fotografowanych obiektów i rozpoznawaniu ich charakterystyki. Same zdjęcia hiperspektralne zajmują jednak znacznie więcej przestrzeni dyskowej, a ich przetwarzanie stanowi wyzwanie i jest praktycznie niemożliwe bez wykorzystania komputera [40].

1.1 Cele pracy

Wstępna analiza rozważanego w niniejszej pracy zagadnienia związanego z analizą obrazów hiperspektralnych pozwoliła na podzielenie ogólnego zadania na kilka mniejszych, których zrealizowanie zostało uznane za kluczowe. Cele niniejszej pracy podsumowano w poniższych punktach:

- Dokonanie przeglądu literatury opisującej próby wykorzystania technik uczenia maszynowego oraz głębokiego do rozpoznawania obiektów znajdujących się na obrazach hiperspektralnych.
- Zbadanie, w jaki sposób różne metody podziału danych na zbiór treningowy i testowy wpływają na jakość rozpoznawania obiektów obecnych na obrazach hiperspektralnych.
- Zaprojektowanie odpowiedniej architektury głębokiej sieci neuronowej.
- Wybór metryk pozwalających na ewaluację oraz rzetelne porównanie otrzymanych wyników.
- Implementacja wybranych algorytmów oraz przeprowadzenie eksperymentów z wykorzystaniem różnych hiperparametrów sieci głębokich.
- Przeanalizowanie wyników eksperymentów oraz porównanie ich z wynikami znanymi z literatury.
- Zdefiniowanie możliwych kierunków dalszego rozwoju bądź usprawnień do zaproponowanego rozwiązania.

1.2 Układ pracy

Praca została podzielona na pięć rozdziałów. Rozdział drugi zawiera analizę literatury związanej z tematem pracy i ma na celu przybliżenie elementarnych zagadnień związanych z omawianą dziedziną nauki oraz zwięzłe przedstawienie znanych rozwiązań rozważanego w pracy zagadnienia. W rozdziale trzecim opisano zaproponowane podejście do badanego problemu, rozważono różne metody służące ocenie jego skuteczności oraz uzasadniono wybór konkretnych technik. Rozdział czwarty przedstawia metodyki wykorzystane podczas badań, opisy zbiorów danych, na których przeprowadzono eksperymenty oraz wyniki będące efektem tychże eksperymentów. W rozdziale piątym zebrano wnioski wyciągnięte z przeprowadzanych badań oraz zdefiniowano potencjalne kierunki dalszego rozwoju pracy wraz z propozycjami ulepszeń do wykorzystanych algorytmów.

Rozdział 2

Przegląd literatury

W niniejszym rozdziale znajduje się opis wszelkich pojęć oraz algorytmów niezbędnych do zrozumienia istoty problemu rozważanego w niniejszej pracy. Wyjaśnione oraz doprecyzowane są w nim przede wszystkim terminy występujące w tytule pracy, takie jak segmentacja obrazów, obrazy hiperspektralne, konwolucyjne sieci neuronowe, czy różnice między sieciami dwu- i trójwymiarowymi. W dalszej części rozdziału znajduje się przegląd oraz analiza literatury związanej z tematem wraz z krótkim opisem istniejących rozwiązań.

2.1 Obrazowanie hiperspektralne

Obrazowanie hiperspektralne (ang. *hyperspectral imaging*) polega na jednoczesnym rejestrowaniu wielu zakresów widma spektralnego (kanałów) dla fotografowanego obszaru. W odróżnieniu od fotografii barwnej, gdzie dla obrazu rejestrowane są tylko trzy zakresy odpowiadające kolorowi czerwonemu, zielonemu, oraz niebieskiemu (RGB, od ang. *red, green, blue*), w fotografii hiperspektralnej rejestruje się kilkadziesiąt, albo nawet kilkaset długości fal. Zazwyczaj wykraczają one poza spektrum światła widzialnego i obejmują także podczerwień lub ultrafiolet [40].

Obraz hiperspektralny jest trójwymiarową strukturą danych posiadającą dwa wymiary przestrzenne i jeden wymiar spektralny. Obraz ten składa się z $W \times H \times B$ pikseli, gdzie $W \times H$ to rozmiar obrazu w ramach jednego kanału, natomiast B reprezentuje liczbę kanałów. Innymi słowy obraz hiperspektralny składa się z B

obrazów monochromatycznych o wysokości H pikseli i szerokości W pikseli [27].

Posiadając informację o ilości odbitego promieniowania elektromagnetycznego, w zależności od częstotliwości fali, można dla każdego miejsca w obrazie wyznaczyć charakterystykę spektralną. Pozwala ona na klasyfikację obiektów znajdujących się na obrazowanej scenie. Jej dokładność zależy między innymi od zdolności badanego materiału do absorpcji promieniowania elektromagnetycznego w zadanym spektrum, zakresu częstotliwości promieniowania wykorzystanego w badaniu, rozdzielczości spektralnej, czy stosunku sygnału do szumu użytego sensora hiperspektralnego. Różne skanery wykonują zdjęcia hiperspektralne przy użyciu różnych zakresów częstotliwości, przy czym zakresy te są zazwyczaj ciągłe. Ważne jest, aby odpowiednio dobrać aparaturę pomiarową do badanych obiektów. Fotografie hiperspektralne wykonuje się zarówno dla obiektów w skali mikroskopowej, jak i makroskopowej. Stan skupienia substancji, dla której stosuje się takie obrazowanie, również nie stanowi ograniczeń. Technika ta jest efektywna zarówno dla ciał stałych, cieczy, jak i gazów. Obrazowanie hiperspektralne znajduje bardzo szerokie zastosowanie w takich dziedzinach jak geografia, geologia, meteorologia, rolnictwo, archeologia, kryminalistyka czy ratownictwo [8].

Mimo wielu zalet obrazowanie hiperspektralne wiąże się także z pewnymi wyzwaniami. Ze względu na zastosowania dane hiperspektralne często pozyskiwane są z sensorów umieszczonych na samolotach lub dronach. W takich przypadkach mogą wystąpić liczne zakłócenia prowadzące do zmniejszenia stosunku sygnału do szumu, takie jak zanieczyszczenia atmosfery gazami i aerozolami (rozpraszającymi i absorbującymi promieniowanie elektromagnetyczne) oraz niekontrolowane warunki oświetlenia sensorów lub fotografowanych obiektów. Zbieranie danych w takich warunkach jest o wiele trudniejsze niż w laboratorium, gdzie są one zwykle optymalne, niezmienne i dobrze kontrolowane [38].

Zdjęcia hiperspektralne przez swoją trójwymiarową strukturę są trudne do interpretacji nawet przez specjalistę bez wykorzystania dodatkowych narzędzi lub programów. Najprostszy sposób wizualizacji takich danych sprowadza się do generowania trójkanałowych obrazów. Kanały wykorzystane do stworzenia takiego zdjęcia nie muszą jednak odpowiadać tym wykorzystywanym w fotografii kolorowej. Spektra wybierane są więc losowo, bądź dobierane są te, które mogą nieść najistotniejsze informacje. W celu łatwiejszej analizy każdemu z wybranych kana-

łów przypisuje się jakąś barwę. Kolory te nie oddają jednak rzeczywistych barw, a mają na celu jedynie ułatwienie interpretacji [8].

2.2 Segmentacja obrazu

Segmentacja obrazu jest to operacja polegająca na wyodrębnieniu z obrazu obszarów, które są jednorodne pod względem pewnego wybranego kryterium [18]. Definiując obszar jako zbiór punktów obrazu R pozostających w relacji spójności, to jest takich że $\forall p_1, p_2 \in R$ istnieje ścieżka p łącząca p_1 z p_2 taka, że $p \in R$, segmentacje można określić jako podział obrazu R na rozłączne obszary $R_1, R_2, ..., R_N$ takie, że:

- $R_1, R_2, ..., R_N$ tworzą podział zbioru R, tj.
  - $\sum_{i=1}^{n} R_i = R$,
  - Obszary są parami rozłączne: $i \neq j \Rightarrow R_i \cap R_j = 0$.
- Obszary R_i są spójne.
- Każdy obszar powinien spełniać pewien test jednolitości (predykat logiczny P).
- Dla żadnej pary przyległych obszarów ich suma nie może spełniać P.

Kryterium jednorodności dobierane jest w zależności od cech, które w przetwarzanym obrazie uważane są za istotne. Odpowiedni dobór tego kryterium bywa kluczowy i często wymaga dużej wiedzy domenowej [20].

Wynikiem segmentacji jest nowy obraz o tych samych wymiarach, będący uogólnieniem obrazu wejściowego. Przykładowy wynik segmentacji zaprezentowany jest na rysunku 2.1. Liczba obszarów w obrazie wynikowym posiadających różne cechy może zależeć od parametru danego algorytmu segmentacji bądź wybranego kryterium jednorodności. W najprostszym przypadku, gdy takie obszary są tylko dwa, mówi się o binaryzacji obrazu.

Segmentację przeprowadza się w celu redukcji nadmiarowych informacji w obrazie, dzięki czemu obraz staje się prostszy do interpretacji oraz lepiej przystosowany do dalszej analizy [22]. Aby dokonać oceny poprawności segmentacji stosuje

Rysunek 2.1: Wynik segmentacji (po prawej) przeprowadzonej dla obrazu wejściowego (po lewej) zestawiony z obszarem zdefiniowanym przez eksperta (środkowy obraz). Źródło: [17].

się odpowiednie metryki. Jedną z prostszych do wyznaczenia jest współczynnik DICE (od ang. *dice similarity coefficient*). Można go wyliczyć z poniższego wzoru:

$$S(R_A, R_B) = \frac{2|R_A \cap R_B|}{|R_A \cup R_B|}, \tag{2.1}$$

gdzie R_A i R_B to obszary znalezione przez dwa algorytmy segmentacji A i B lub obszar zdefiniowany przez eksperta (ang. *ground truth*, A) i obszar znaleziony przez algorytm (B).

W kontekście obrazów hiperspektralnych, segmentację obrazów często wykonuje się poprzez przyporządkowanie każdemu pikselowi etykiety reprezentującej przynależność do jednej z wielu klas [8]. Granice poszczególnych obszarów wyznacza się następnie według kryteriów opisanych wcześniej w tym podrozdziale.

2.3 Sztuczne sieci neuronowe

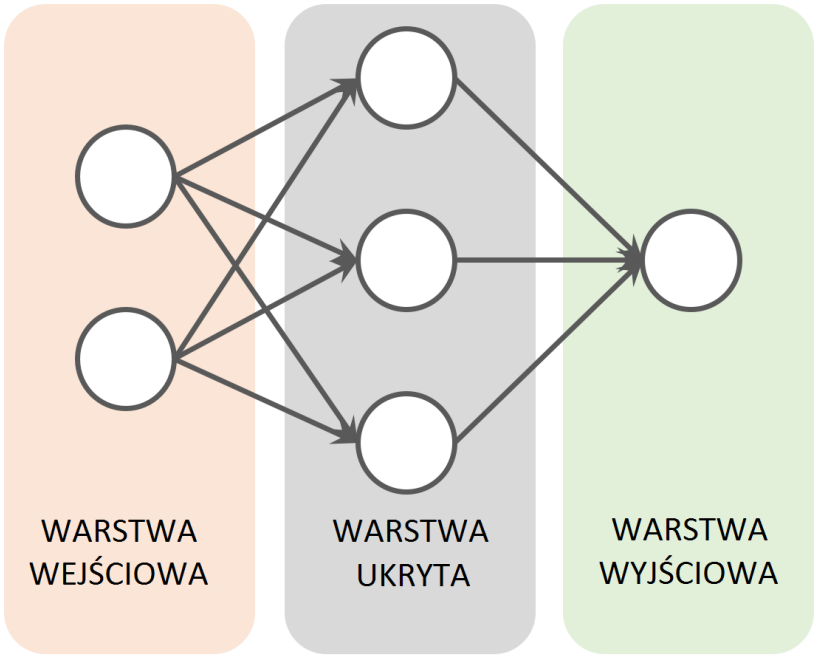
Powstanie sztucznych sieci neuronowych zostało zainspirowane badaniami nad działaniem biologicznych neuronów tworzących mózgi ludzi i zwierząt. Pierwsze matematyczne modele tych komórek oraz próby łączenia ich w proste sieci powstały już w pierwszej połowie XX wieku. Koncepcja ta, rozwinięta i udoskonalona, znajduje obecnie szerokie zastosowania w wielu dziedzinach życia, sprawdzając się przede wszystkim tam, gdzie zawodzą algorytmy deterministyczne, oparte na sztywno określonych regułach [25].

2.3.1 Budowa sztucznych sieci neuronowych

Podstawowym elementem przetwarzającym sygnał w sztucznych sieciach neuronowych jest neuron. Posiada on n wejść x_i, przy czym $i = 1, ..., n$ oraz tylko jedno wyjście y. Działanie neuronu można opisać w następujący sposób. Najpierw wartość każdego sygnału wejściowego x_i jest przemnażana przez odpowiadającą mu wagę w_i, a następnie wszystkie wyniki są sumowane. Przy tym działaniu bardzo często uwzględnia się jeden dodatkowy składnik b zwany progiem (ang. *bias*). Suma jest przekazywana jako argument do funkcji aktywacji f, która decyduje o tym, jaki sygnał pojawi się na wyjściu neuronu. Matematyczny model działania neuronu posiadającego n wejść, odpowiadający powyższemu opisowi można zapisać jako:

$$y = f(b + \sum_{i=1}^{n} w_i x_i). \tag{2.2}$$

Neurony połączone w acykliczny graf skierowany tworzą sztuczną sieć neuronową. W takiej strukturze wyjścia niektórych neuronów stają się wejściami dla innych. Aby zwiększyć łatwość korzystania oraz wydajność sieci neuronowych, neurony w ich obrębie grupuje się w warstwy. Kolejne warstwy łączy się ze sobą zazwyczaj w taki sposób, że wyjścia neuronów warstwy poprzedniej podawane są na wejścia każdego neuronu warstwy następnej. Połączone w ten sposób neurony wraz z nazwami kolejnych warstw zaprezentowane zostały na rysunku 2.2. Pierwsza warstwa, nazywana warstwą wejściową, przyjmuje na swoje wejścia dane dostarczone do sieci. Oznacza to, że liczba neuronów w tej warstwie odpowiada liczbie jednocześnie wprowadzanych do sieci wartości. Na wyjściach ostatniej warstwy (warstwy wyjściowej) pojawiają się wyniki działania sieci. Wszystkie warstwy między wejściową a wyjściową nazywa się warstwami ukrytymi. Ich zadaniem jest realizacja kolejnych etapów przetwarzania danych. Może istnieć dowolna liczba warstw ukrytych N. W zależności od niej mówi się o N-warstwowych sieciach neuronowych. Sieci, które posiadają więcej niż parę warstw ukrytych, nazywa się sieciami głębokimi [10].



Rysunek 2.2: Przykład struktury sieci neuronowej złożonej z warstwy wejściowej, jednej warstwy ukrytej oraz warstwy wyjściowej.

2.3.2 Uczenie sztucznych sieci neuronowych

W dziedzinie uczenia maszynowego można wskazać kilka sposobów nauki w zależności od tego, czy istnieje możliwość zweryfikowania poprawności odpowiedzi udzielonej przez model. W przypadku, gdy jest to niemożliwe, mówi się o uczeniu nienadzorowanym (ang. *unsupervised learning*), które bazuje wyłącznie na strukturach i zależnościach obecnych w danych. Algorytmy uczenia nienadzorowanego wykorzystuje się na przykład do grupowania danych. O uczeniu nadzorowanym (ang. *supervised learning*) mówi się wtedy, gdy dane, na podstawie których model uczy się wnioskować, są opisane, to znaczy, dla każdego przykładu ze zbioru treningowego istnieje zdefiniowana oczekiwana odpowiedź modelu. Te dodatkowe informacje pozwalają algorytmowi uczącemu na wypracowanie reguł, dzięki którym możliwa jest predykcja odpowiedniej wartości dla każdego nowego przykładu (nieprzetwarzanego wcześniej przez model). Jeżeli przewidywana zmienna przyjmuje wartość kategoryczną, proces ten nazywa się klasyfikacją, natomiast jeśli zmienna przyjmuje wartość ciągłą (numeryczną), mówi się o regresji [25].

Uczenie sieci jest procesem opartym na prezentacji przypadków uczących (przykładów prawidłowo rozwiązanych zadań), w trakcie której następuje stopniowa poprawa zdolności sieci do rozwiązywania postawionego przed nią zadania. Jest to możliwe dzięki porównywaniu odpowiedzi udzielanych przez sieć z odpowiedziami wzorcowymi. Wprowadzana korekta błędu powoduje, że sieć po każdej prezentacji zwiększa szansę udzielenia odpowiedzi bardziej zbliżonej do odpowiedzi wzorcowej. Zmierza się również do tego, aby sieć uogólniła wyuczoną umiejętność na inne, podobne zadania, nieprezentowane w trakcie uczenia (generalizacja).

Uczenie wielowarstwowych sieci neuronowych jest najczęściej realizowane przy pomocy algorytmu wstecznej propagacji błędu (ang. *backpropagation*). Można go postrzegać jako iteracyjny proces „przejścia" i „powrotu" przez wszystkie warstwy sieci. „Przejście" jest tu propagacją informacji w przód (ang. *forward pass*), a „powrót" to propagacja informacji w tył (ang. *backward pass*) [37].

Faza propagacji w przód ma miejsce, gdy przykłady szkoleniowe przechodzą przez sieć neuronową w celu wyznaczenia przewidywanych dla niech etykiet. Każdy neuron znajdujący się w ramach tej samej warstwy przeprowadza pewną transformację informacji, które otrzymał od neuronów warstwy poprzedniej, a następnie przetworzony sygnał przesyła do neuronów warstwy następnej. Gdy odpowiednie obliczenia zostaną wykonane przez wszystkie neurony, a dane podane na wejściu pokonają każdą warstwę, wtedy na wyjściach ostatniej warstwy otrzyma się przewidywane dla tych danych etykiety [10].

Zestawiając przewidziane przez sieć etykiety z faktycznymi dokonuje się oceny modelu. Wykorzystuje się do tego tak zwaną funkcją straty (ang. *loss function*). Jest to funkcja, która oblicza koszt popełnionego przez model błędu (stratę). Jest ona minimalizowana podczas treningu. Dzięki niej można określić, czy model zachowuje się w oczekiwany sposób. Mimo że istnieje wiele sposobów realizacji tej funkcji, jej wyniki mogą być interpretowane w taki sam sposób. W idealnym przypadku, to znaczy takim, w którym wartość funkcji wynosi zero, można mówić o braku rozbieżności między wartościami szacunkowymi a oczekiwanymi. Jeżeli natomiast wartość jest wysoka, oznacza to, że model wymaga dalszego uczenia (czyli dalszego dostosowywania wag) [37].

Po obliczeniu straty jest ona propagowana wstecz. Począwszy od warstwy wyjściowej, informację o stracie przesyła się do wszystkich neuronów poprzedzającej

jej warstwy ukrytej, które przyczyniły się do obliczenia wartości na wyjściu sieci. Neurony te otrzymują jednak tylko ułamek całkowitego sygnału straty wyznaczany w oparciu o względny wkład, w jakim każdy z nich przyczynił się do określenia wyjścia. Proces ten powtarza się warstwa po warstwie, aż wszystkie neurony w sieci otrzymają sygnał straty opisujący ich względny udział w całkowitej stracie.

Gdy informacje o stracie są już rozpowszechnione wewnątrz sieci, wagi połączeń między neuronami są dostosowywane w taki sposób, aby następnym razem, gdy sieć zostanie wykorzystana do przewidywania, wartość funkcji straty była jak najbliższa zeru. Jest to więc problem optymalizacji, do rozwiązania którego wykorzystuje się jeden z algorytmów optymalizacyjnych (ang. *optimization procedure*). Zaktualizowanie wag sieci jest ostatnim etapem pojedynczej iteracji treningu zwanej epoką (ang. *epoch*) [37].

2.3.3 Hiperparametry sztucznych sieci neuronowych

Przedstawiony powyżej opis w dosyć ogólny sposób tłumaczy działanie algorytmu używanego podczas uczenia sztucznych sieci neuronowych. Jego rzeczywisty przebieg może być nieco bardziej złożony, ponieważ jest on uzależniony od wielu parametrów oraz wybranych sposobów realizacji niektórych etapów uczenia nazywanych ogólnie hiperparametrami. Określane one są zawsze podczas konstruowania modelu. Najistotniejsze z nich zostały już wspomniane oraz krótko scharakteryzowane, ale wymagają bardziej dogłębnego wyjaśnienia.

Funkcja aktywacji

Przytoczona podczas omawiania budowy neuronu funkcja aktywacji określa sposób obliczania wartości wyjściowego sygnału neuronu na podstawie wartości sumarycznego pobudzenia (sumy wszystkich wejść pomnożonych przez odpowiadające im wagi). Jej obecność pozwala na wprowadzenie nieliniowości do modelu, co skutkuje możliwością rozwiązywania przez sieć bardziej skomplikowanych zadań. Dobranie odpowiedniej funkcji aktywacji jest więc bardzo ważne. Poniżej opisano parę najczęściej wykorzystywanych funkcji aktywacji [10].

Funkcja Sigmoidalna (ang. *Sigmoid*) jest określona wzorem $f(x) = \frac{1}{1+e^{-x}}$. Niezależnie od tego, jak duża będzie łączna suma aktywacji, zostanie ona przekształ-

cona na liczbę z zakresu $\langle 0, 1\rangle$. W przeszłości bywała standardowym wyborem podczas implementacji sztucznej sieci neuronowej. Obecnie ze względu na swoje wady jest coraz rzadziej spotykana. Gdy wartość przez nią zwracana jest bliska 0 lub 1, wtedy jej gradient osiąga wartość bliską zeru. W znaczny sposób utrudnia to naukę danego neuronu oraz sprawia, że podczas propagacji wstecznej neuron nie przesyła sygnału do wcześniejszych warstw. Neurony, których wartość na wyjściu jest zawsze równa 0 lub 1, nazywane są „martwymi". Przy nieodpowiednio dobranych pozostałych parametrach sieci łatwo o sytuację, w której wiele neuronów osiągnie taki stan.

Funkcja Softmax jest funkcją wykładniczą, której wartość zostaje dodatkowo znormalizowana w taki sposób, by suma wszystkich wyjść dla całej warstwy była równa 1. Używana jest najczęściej w warstwie wyjściowej wielowarstwowych sieci neuronowych dostosowanych do rozwiązywania problemów klasyfikacyjnych. Dzięki zastosowanej normalizacji wartości wyjściowe mogą być interpretowane jako oszacowania prawdopodobieństw przynależności danego sygnału wejściowego do poszczególnych klas [12].

Funkcja ReLU (od ang. *Rectified Linear Units*) dana jest wzorem $f(x) = max(0, x)$. Neuron, wewnątrz którego zastosowana jest ta funkcja aktywacji, przekazuje sygnał dalej tylko wtedy, gdy suma jego aktywacji jest większa od 0. Gdy ten warunek jest spełniony, wartość wyjściowego sygnału jest równa sumie sygnałów wejściowych. Do korzyści płynących z wykorzystania tej funkcji w sieciach neuronowych zalicza się między innymi przyśpieszenie procesu uczenia oraz ograniczenie czasu potrzebnego do otrzymania przewidywań dla nowych przykładów w stosunku zarówno do funkcji sigmoid, jak i wielu innych funkcji aktywacji. Obecnie jest to jedna z najczęściej wykorzystywanych funkcji aktywacji. Nie jest jednak pozbawiona wad, podobnie jak w przypadku funkcji sigmoid, funkcja ReLU też może sprawić, że niektóre neurony staną się „martwe". Można spotkać się z modyfikacjami tej funkcji starającymi się rozwiązać ten problem [21].

Funkcja straty

Funkcja straty, inaczej zwana funkcją kosztu, jest powiązana z procesem uczenia modelu. O jej doborze może decydować natura zadania, które tworzona sieć

neuronowa ma rozwiązywać. W przypadku, gdy przewidywane przez model wartości są ciągłe, funkcja ta będzie zdefiniowana w inny sposób, niż gdyby problemem była klasyfikacja binarna (klasyfikacja przykładu do jednej z dwóch klas), czy wieloklasowa. Jednak nawet w obrębie jednego typu zadań spotyka się różne formuły służące do obliczenia wartości funkcji kosztu. Wybór najbardziej dopasowanej funkcji kosztu do rozwiązywanego problemu wymaga jego zrozumienia i określenia, jakiego typu błędy są akceptowalne, a które powinny być eliminowane. Poniżej opisano parę najczęściej wykorzystywanych funkcji straty.

Suma-Kwadratów jest sumą kwadratów różnic pomiędzy wartościami zadanymi i wartościami otrzymanymi na wyjściach każdego neuronu wyjściowego. Jest to standardowa funkcja błędu stosowana w trakcie uczenia sieci neuronowych. Jej wybór jest z pewnością najwłaściwszą decyzją w większości problemów regresyjnych.

Entropia wzajemna (pojedyncza i wielokrotna) jest sumą iloczynów zadanych wartości oraz logarytmów błędów dla każdego neuronu wyjściowego. Funkcja ta występuje w dwóch wersjach: jedna z nich przeznaczona jest dla sieci z pojedynczym neuronem wyjściowym (rozróżniającym dwie klasy), zaś druga dla sieci z wieloma neuronami wyjściowymi. Funkcja błędu oparta na entropii wzajemnej została specjalnie zaprojektowana dla problemów klasyfikacyjnych, gdzie jest stosowana łącznie z występującą w warstwie wyjściowej sieci logistyczną funkcją aktywacji (gdy przewidywane jest pojedyncze wyjście sieci) lub funkcją aktywacji typu Softmax (gdy w sieci występuje większa liczba wyjść) [4].

Optymalizacja funkcji kosztu

Algorytmy optymalizacyjne są używane podczas aktualizowania wag modelu. Przy ich pomocy redukowane są błędy popełniane przez sieć. Aby zrozumieć różnice w poszczególnych podejściach, należy przedstawić jeden dodatkowy parametr – współczynnik uczenia (ang. *learning rate*). Algorytm uczenia wskazuje, w jakim kierunku należy zmienić wagi, żeby błąd popełniany przez sieć zmalał, natomiast współczynnik uczenia decyduje o tym, jak bardzo te wagi we wskazanym kierunku się zmienią. W przypadku, w którym współczynnik uczenia jest zbyt mały, proces uczenia może trwać bardzo długo. Jeśli jednak zastosowany współczynnik uczenia

jest zbyt duży, to na skutek niemonotonicznej charakterystyki funkcji błędu może się zdarzyć, że minimum funkcji błędu zostanie pominięte. W efekcie błąd po wykonaniu aktualizacji wag może być większy a nie mniejszy niż przed jej wykonaniem. Przyjmując za kryterium to, czy współczynnik uczenia jest modyfikowany podczas nauki czy też nie, można wyróżnić dwie kategorie algorytmów optymalizacji funkcji kosztu [12].

- Algorytmy o stałym współczynniku nauki – współczynnik nauki jest stały przez cały proces uczenia.
- Algorytmy o adaptacyjnym współczynniku nauki – współczynnik nauki może ulegać zmianie na przestrzeni całego procesu uczenia.

Jedną z metod należących do pierwszej kategorii jest metoda stochastycznego gradientu prostego (ang. *gradient descent*). Ta technika wykorzystuje do modyfikacji wag pochodne cząstkowe (gradient) funkcji straty, dzięki czemu jest w stanie określić kierunek, w którym funkcja ta maleje najszybciej. Podczas procesu uczenia, wagi sieci aktualizowane są po każdym przetworzonym przykładzie uczącym. Takie podejście wprowadza sporą fluktuację. By ją ograniczyć, stosuje się modyfikację tego algorytmu polegającą na aktualizacji wag dopiero, gdy zostanie przetworzona określona liczba przykładów (ang. *batch size*). Pozwala ona ponadto na przyśpieszenie treningu sieci, ponieważ ogranicza niezbędną do wyuczenia modelu liczbę wykonywania opisanego algorytmu [7].

Algorytm Momentum jest modyfikacją metody stochastycznego gradientu prostego pozwalającą na jej przyśpieszenie. Podczas aktualizacji wag brany jest pod uwagę nie tylko gradient obliczony w danej chwili, ale również ten wyliczony w poprzednim kroku. Zmiana wprowadzana w wagach jest większa, jeżeli oba gradienty mają ten sam kierunek. W przeciwnym wypadku (gdy gradienty mają przeciwny kierunek) zmiana jest mniejsza [7].

Algorytm Adam (od ang. *Adaptive Moment Estimation*) jest obecnie najczęściej wykorzystywanym optymalizatorem funkcji kosztu. Jest to algorytm o adaptacyjnym współczynniku nauki obliczanym dla każdego parametru sieci osobno, wykorzystującym ponadto koncept obecny w metodzie Momentum. Charakteryzuje się dużą szybkością odnajdowania minimum (w stosunku do innych optymalizatorów) [12].

Regularyzacja

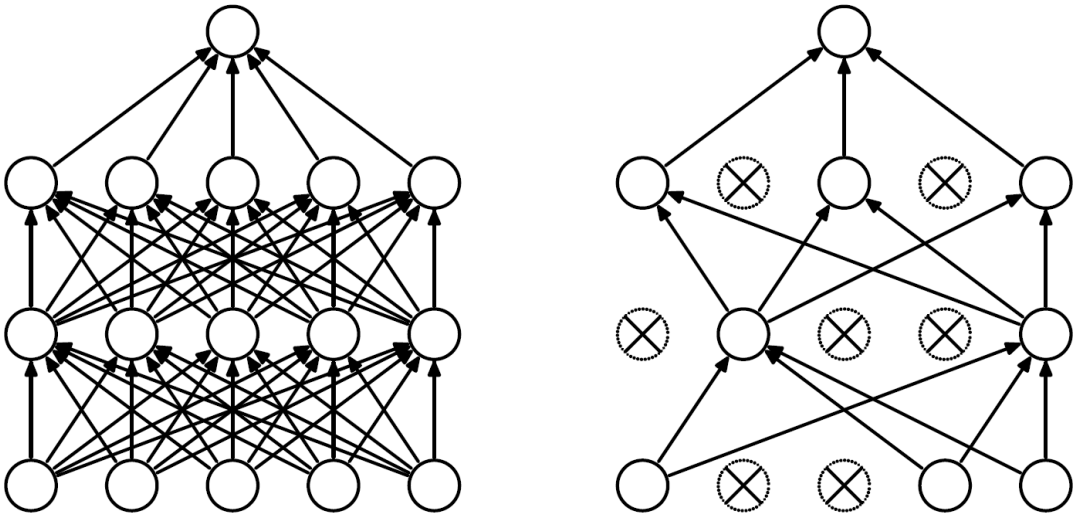
Jednym z fundamentalnych wyzwań obecnych w dziedzinie uczenia maszynowego jest pokonanie problemu przeuczania modelu (ang. *overfitting*), czyli jego zbyt dużego dopasowania do danych treningowych. Występuje ono, gdy model osiąga dobre wyniki będąc ocenianym na podstawie danych pochodzących ze zbioru treningowego, jednocześnie charakteryzując się niską dokładnością w przypadku ewaluacji wykorzystującej zbiór testowy. W takim przypadku, mówi się również, że sieć nie posiada zdolności generalizacji zdobytej wiedzy. Swego rodzaju przeciwieństwem tego zjawiska jest niedouczenie (ang. underfitting), które ma miejsce gdy model nie jest w stanie uchwycić wzorców zarówno w danych szkoleniowych, jak i testowych. Przeuczenie występuje zwykle, gdy model jest nadmiernie złożony, w szczególnie gdy posiada zbyt wiele parametrów do dostosowania. Różnica w wartości błędu obliczonej na różnych zbiorach może wynikać na przykład z tego, że model zaczyna błędnie postrzegać szum obecny w danych treningowych jako wzorce i cechy charakterystyczne [32].

Jednym ze skutecznych sposobów radzenia sobie z tym problemem jest zwiększenie liczby przykładów treningowych. Pozyskanie ich jest niestety często niemożliwe bądź bardzo kosztowne. Inną standardową metodą stosowaną aby zapobiegać przeuczeniu modelu jest użycie dodatkowego zbioru (walidacyjnego) podczas uczenia sieci i zatrzymanie treningu w momencie gdy błąd sieci obliczony na tym dodatkowym zbiorze zaczyna rosnąć. Jest to jednak metoda wejściowo-wyjściowa, traktująca algorytm uczenia jako czarną skrzynkę i może okazać się niewystarczająca aby w pełni rozwiązać problem.

Istnieją również techniki wbudowane w algorytm uczenia pozwalające na uniknięcie zbytniego dopasowania modelu do danych treningowych. Taką techniką jest regularyzacja. Jedną z bardziej popularnych jest regularyzacja L_2, która polega na dodaniu dodatkowego składnika do funkcji kosztu. Jest on obliczany na podstawie wartości wszystkich wag modelu (im większe wagi, tym większą ma wartość) i nazywany karą (ang. *penalty*). Celem tej operacji jest sprawienie by wagi występujące w sieci były mniejsze, ponieważ to duże wagi są powodem dużych zmian na wyjściu przy małych zmianach na wejściu. Stopień w jakim obliczona kara wpływa na całkowitą funkcję kosztu może być regulowany za pomocą parametru metody

[43].

Inną metodą regularyzacji jest *Dropout* (pojęcie nie posiada polskiego odpowiednika). Polega na wielokrotnym, tymczasowym usuwaniu losowych neuronów w trakcie uczenia sieci. Po zakończeniu jednej fazy uczenia z usuniętymi neuronami, są one przywracane. W kolejnej fazie losowany jest nowy zestaw usuwanych neuronów, a uczenie jest kontynuowane. Ostatecznie wytrenowany model wykorzystuje wszystkie neurony. Dzięki takiemu podejściu neurony uczą się w bardziej niezależny sposób [35]. Schemat sieci po jednokrotnym usunięciu wybranych neuronów został przedstawiony na rysunku 2.3.



Rysunek 2.3: Przykładowa struktura sieci neuronowej przed (lewa strona obrazu) oraz po (prawa strona obrazu) zastosowaniu operacji *Dropout*.

2.4 Konwolucyjne sieci neuronowe

Konwolucyjne sieci neuronowe nazywane także splotowymi są to głębokie sieci neuronowe wyspecjalizowane w przetwarzaniu wszelkiego rodzaju sygnałów (w niniejszej pracy zostaną omówione głownie w kontekście przetwarzania obrazów). Działają one na zasadzie ekstrakcji cech i ich transformacji w celu osiągnięcia hierarchii cech wtórnych. Próbują określić bardziej skomplikowane cechy na podstawie prostszych cech. Ograniczona w odpowiedni sposób liczba połączeń między

neuronami znajdującymi się w sąsiadujących warstwach pozwala na szybsze trenowanie sieci oraz zmniejsza jej skłonności do przeuczenia się, czyli nadmiernego dopasowania do danych treningowych [1].

Najistotniejszymi elementami konwolucyjnych sieci neuronowych, od których pochodzi ich nazwa, są warstwy konwolucyjne. Konwolucją nazywa się operację na dwóch funkcjach, której wynikiem jest inna, która może być interpretowana jako zmodyfikowana wersja oryginalnych funkcji.

Ogólny przypadek konwolucji można zapisać jako:

$$h(t) = (f * g)(t) \int_0^t f(x)g(t-x)dx, \tag{2.3}$$

gdzie:

- h – wynik operacji konwolucji,
- f – funkcja poddawana operacji konwolucji,
- g – funkcja określająca wagi dla funkcji f (filtr),

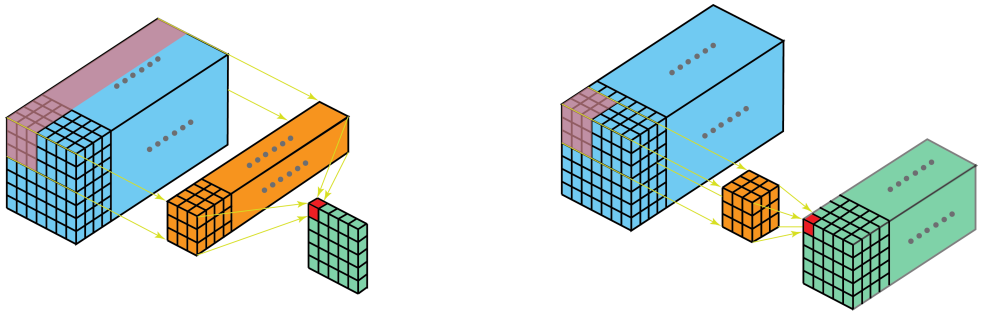
który dla przypadku dyskretnego przyjmuje postać:

$$h(t) = (f * g)[n] = \sum_m^n f[m]g[n-m]. \tag{2.4}$$

W kontekście analizy obrazów istota konwolucji polega na przesuwaniu okna z wartościami z g zwaną kernelem bądź filtrem „wzdłuż" sygnału f oraz przemnażaniu odpowiadających wartości i sumowaniu wynikowych iloczynów. W przypadku dwuwymiarowym filtr będzie przesuwany z lewej do prawej strony, a następnie z góry na dół. Konwolucje dla takiego przypadku opsuje poniższa formuła:

$$h[m,n] = (f * g)[m,n] = \sum_j \sum_k f[j,k]g[m-j,n-k]. \tag{2.5}$$

W przypadku konwolucji trójwymiarowej zasada działania jest bardzo podobna, ale filtr jest „przesuwany" wzdłuż dodatkowego wymiaru. Wynika to z faktu, że głębokość kernela jest mniejsza od głębokości obrazu (dla konwolucji dwuwymiarowej głębokości te są równe). Wynikiem takiej operacji jest trójwymiarowa struktura (w przypadku konwolucji dwuwymiarowej wynikiem jest macierz) [2]. Różnice między konwolucją dwuwymiarową i trójwymiarową przedstawione są na rysunku 2.4.



(a) Konwolucja dwuwymiarowa (b) Konwolucja trójwymiarowa

Rysunek 2.4: Przykład przeprowadzania operacji konwolucji. Kolorem niebieskim oznaczono obraz wejściowy, kolorem pomarańczowym filtr, natomiast kolorem zielonym mapę cech. Źródło: [2].

Formalny zapis operacji konwolucji trójwymiarowej wygląda następująco:

$$h[m,n,p] = (f * g)[m,n,p] = \sum_j \sum_k \sum_l f[j,k,l]g[m-j,n-k,p-l]. \tag{2.6}$$

2.4.1 Budowa konwolucyjnych sieci neuronowych

Pierwszą warstwą w przypadku konwolucyjnych sieci neuronowych jest warstwa konwolucyjna. Przyjmuje ona na swoje wejścia obraz przedstawiony w postaci trójwymiarowej macierzy, w której pierwszy oraz drugi wymiar odpowiadają wysokości i szerokości obrazu, natomiast trzeci odpowiada za liczbę kanałów (jeżeli jednocześnie na wejścia sieci podawane jest wiele obrazów, to struktura ta posiada jeden dodatkowy wymiar). Fotografie czarno-białe posiadają tylko jeden kanał, zdjęcia kolorowe trzy (odpowiadające wartościom RGB), natomiast obrazy hiperspektralne mogą ich mieć nawet kilkaset.

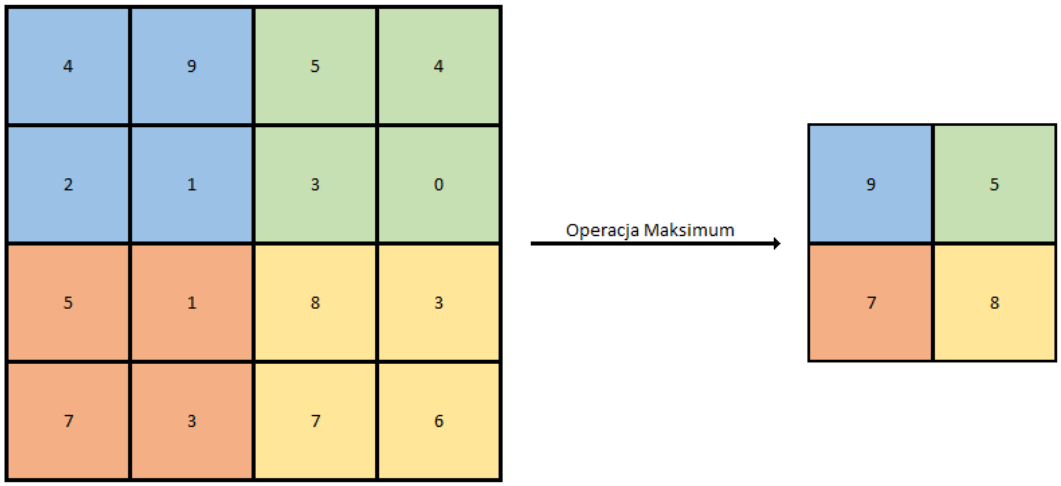
Warstwa konwolucyjna wykorzystuje omówioną wcześniej w tym rozdziale operację splotu w celu wytworzenia liniowych aktywacji. Dwuwymiarowe lub trójwymiarowe macierze, których rozmiar jest jednym z hiperparametrów, nazywane są jądrem, kernelem bądź filtrem. Filtr reprezentuje pewną cechę charakterystyczną, która występuje w obrazie. Jest on przemieszczany po wejściowym obrazie, a na kolejnych polach macierzy wykonywana jest operacja splotu. Wartości przechowywane w polach kernala to wagi. Każde kolejne przesunięcie filtra w granicach

obrazu daje nowy wynik mnożenia splotowego. Zestawione ze sobą wyniki tego działania dla całego obrazu tworzą mapę cech (mapę aktywacji). Im większa wartość znajduje się w danym miejscu mapy aktywacji, tym bardziej prawdopodobne jest, że cecha reprezentowana przez filtr występuje w tym miejscu na obrazie [1].

Wykorzystanie operacji konwolucji do przetwarzania obrazów jest powszechne również w klasycznych metodach wizji komputerowej, w których nie są używane sieci neuronowe. Filtry w nich stosowane są najczęściej predefiniowane lub tworzone przez ekspertów [36]. Tym, co sprawia, że konwolucyjne sieci neuronowe są tak potężnym narzędziem pomocnym w przetwarzaniu obrazów, jest ich zdolność do samodzielnego dobierania filtrów (ekstrakcji cech) podczas procesu uczenia sieci.

Bardzo często między kolejnymi warstwami konwolucyjnymi okresowo umieszczana jest warstwa grupowania danych (ang. *pooling layer*). Jej głównym zadaniem jest stopniowa redukcja rozmiaru danych, co pozwala na przyspieszenie obliczeń oraz zredukowanie prawdopodobieństwa przeuczenie się sieci. Warstwa grupowania danych wykorzystuje zazwyczaj operację Maksimum niezależnie dla każdego obszaru wejściowego i przeskalowuje go przestrzennie (zmniejszając jego rozmiar). Operacja Maksimum polega na ograniczeniu map aktywacji poprzez wybranie z analizowanego obszaru najwyższej wartości aktywacji i przekazanie jej do nowej macierzy aktywacji, tak jak zaprezentowane jest to na rysunku 2.5. Alternatywną metodą jest na przykład obliczenie średniej wszystkich wartości obszaru i umieszczanie jej w nowej macierzy aktywacji [1].

Warstwy konwolucyjne (podobnie jak łączące) często pojawiają się w konwolucyjnych sieciach neuronowych wielokrotnie. Mogą więc one przyjmować na swoje wejścia nie tylko surowe dane z obrazu wejściowego, ale również wyniki pochodzące z warstw poprzednich. Warstwa, która działa bezpośrednio na wartościach pikseli obrazu wejściowego, jest zazwyczaj w stanie wyciągnąć tylko tak zwane cechy niskopoziomowe (np. pionowa lub pozioma krawędź). Kolejna warstwa, która operuje na wynikach poprzedniej, dzięki kombinacji niskopoziomowych cech, jest w stanie uchwycić bardziej złożone cechy (np. kształty). Proces ten postępuje z każdą następną warstwą, pozwalając w końcu na odszukanie cech istotnych z punktu widzenia rozwiązywanego problemu (np. detekcja zwierząt, czy wykrywanie twarzy). Wraz z głębokością sieci pole odbiorcze następujących po sobie warstw pośrednio opisuje coraz większy obszar oryginalnego obrazu [12].



Rysunek 2.5: Przykład działania warstwy łączącej wykorzystującej operacje Maksimum z oknem 2×2 na obrazie wejściowym o rozmiarach 4×4.

Ostatnim elementem występującym w sieciach konwolucyjnych, rozwiązujących problem klasyfikacji, jest warstwa w pełni połączona. Jest to zazwyczaj sieć neuronowa, w której wszystkie neurony warstwy poprzedniej są połączone ze wszystkimi neuronami warstwy następnej. Jej zadaniem jest interpretacja rezultatów wcześniej wykonanych operacji i zdefiniowanie na ich podstawie wektora wag odpowiadających poszczególnym klasom. Wykorzystując wcześniej wyekstrahowane wysokopoziomowe cechy oraz wagi im przypisane, warstwa w pełni połączona określa prawdopodobieństwo przynależność przetwarzanego obrazu do każdej z klas.

2.5 Istniejące rozwiązania problemu segmentacji obrazów hiperspektralnych

Większość metod segmentacji obrazów hiperspektralnych sprowadza się do przeprowadzenia klasyfikacji dla wszystkich pikseli. Oznacza to, że każdemu pikselowi w obrazie przypisywana jest etykieta klasowa [8].

Metodą, z którą można się spotkać w wielu publikacjach dotyczących uczenia maszynowego i głębokiego, wykonywanym jeszcze przed przystąpieniem do właściwej analizy, jest redukcja wymiarowości danych. Mimo utraty pewnych informacji

w analizowanym zbiorze pozwala ona na jego łatwiejsze przetwarzanie, co może wiązać się z takimi korzyściami jak uproszczenie modelu, ograniczenie czasu potrzebnego do wytrenowania sieci, czy zmniejszenie zasobów niezbędnych do przeprowadzenia klasyfikacji. W niektórych przypadkach zredukowanie danych może prowadzić do zwiększenia dokładności klasyfikacji. Natura danych może jednak ten proces utrudniać lub nawet całkowicie go uniemożliwiać [30].

W pracach dotyczących segmentacji obrazów hiperspektralnych redukcja danych najczęściej polega na zmniejszeniu głębokości obrazu. Bardzo często wykorzystywana jest w tym celu analiza głównych składowych (ang. principal component analysis, PCA). Ideą stojącą za PCA jest zredukowanie wymiarowości danych składających się z dużej liczby powiązanych ze sobą zmiennych, przy jednoczesnym zachowaniu w nich jak największej wariancji. Osiąga się to poprzez przekształcenie zmiennych pierwotnych w nowy zestaw zmiennych zwanych głównymi składowym, które nie są ze sobą skorelowane, a następnie uporządkowanie ich w taki sposób, aby to początkowe czynniki wyjaśniały najwięcej zmienności. Przed przeprowadzeniem PCA należy dokonać normalizacji zmiennych. Jej pominięcie w przypadku zmiennych o dużym rozrzucie zakresu wartości może spowodować fałszywy wzrost znaczenia składowych o dużych wartościach [16].

Konwencjonalne metody klasyfikacji pojedynczego piksela wykorzystują wyłącznie informacje spektralne. Typowe klasyfikatory oparte są na obliczaniu odległości [9], metodzie k najbliższych sąsiadów [33] i regresji logistycznej [23]. Zazwyczaj dokładność tych metod jest niezadowalająca z powodu niewystarczającej liczby próbek treningowych dla tak dużej liczby pasm widmowych. Ta nierównowaga między dużą wymiarowością spektralną a ograniczoną liczbą próbek treningowych znana jest jako „przekleństwo wymiarowości". Kolejną wadą algorytmów wykorzystujących jedynie informacje spektralne jest występująca w danych redundancja (pewne pasma widmowe mogą być silnie skorelowane). Przy takim podejściu nie jest również możliwe uchwycenie istotnej zmienności przestrzennej obecnej w przypadku danych o wysokiej rozdzielczości. Ma to ogólny wpływ na niższą dokładność klasyfikacji. Zaprojektowanie klasyfikatorów wykorzystujących zarówno informacje spektralne, jak i przestrzenne (związane z kształtem oraz wielkością różnych struktur) może poprawić skuteczność klasyfikacji.

Konwencjonalne metody klasyfikacji widmowo-przestrzennej można podzielić

na dwie kategorie. W pierwszej znajdują się techniki wykorzystujące informację spektralną i przestrzenną osobno. Inaczej mówiąc, najpierw za pomocą różnych technik (np. wykorzystujących operacje morfologiczne [28]), znajdowane są zależności przestrzenne. Następnie są one łączone z cechami spektralnymi w celu przeprowadzenia klasyfikacji pikselowej. Druga kategoria zazwyczaj łączy informacje przestrzenne z cechami spektralnymi, tworząc połączenie cechy. Na przykład generowane w różnych skalach i częstotliwościach zestawy trójwymiarowych filtrów falkowych [29], trójwymiarowych filtrów Gabora [34] lub innych rodzajów filtrów są stosowane, aby wyodrębnić kombinacje widmowo-przestrzenne.

Większość konwencjonalnych metod ekstrahowania cech opiera się na ich ręcznym tworzeniu (wykorzystując do tego wiedzę domenową) i „płytkich" modelach uczących się. Taki sposób odnajdowania cech nie zawsze się sprawdza ze względu na liczbę szczegółów zawartych w rzeczywistych danych. Ostatnio w obszarze segmentacji obrazów hiperspektralnych duży potencjał dostrzeżono w technikach wykorzystujących uczenie głębokie. Zamiast polegać na ręcznie zaprojektowanych filtrach, głębokie sieci neuronowe są w stanie automatycznie uczyć się hierarchii cech (od nisko do wysoko poziomowych) z surowych danych wejściowych. Podobnie jak wcześniej przytoczone metody, techniki wykorzystujące uczenie głębokie mogą klasyfikować piksele tylko na podstawie informacji spektralnych lub łącząc je z przestrzennymi.

Jednowymiarowe sieci konwolucyjne działające na poziomie pikseli znalazły szerokie zastosowanie w dziedzinie teledetekcji. W [14] zestawiono ze sobą wyniki klasyfikacji otrzymane przy wykorzystaniu jednowymiarowej konwolucyjnej sieci neuronowej, głębokiej sieci neuronowej oraz klasyfikatora bazującego na maszynie wektorów podpierających. Największą dokładność klasyfikacji wykazała pierwsza z wymienionych metod.

Podobne rozwiązania sprawdziły się również w innych dziedzinach. Celem [15] było ocenienie skuteczności nieinwazyjnej metody wykrywania zarażonych pasożytniczym grzybem kłosów pszenicy. Zdjęcia roślin zostały wykonane skanerem hiperspektralnym z odległości około 30 cm. Ich analiza z wykorzystaniem jednowymiarowej konwolucyjnej sieci neuronowej okazała się być bardziej skuteczna niż metoda bazująca na klasycznych algorytmach uczenia maszynowego.

Również w [42] użyto podobnej metody przetwarzania fotografii hiperspektral-

nych w celu rozróżnienia kilku gatunków chryzantem. W przypadku w którym spektralny wymiar danych nie został wstępnie zredukowany, najbardziej satysfakcjonujące wyniki otrzymano przy wykorzystaniu metody bazującej na głębokich konwolucyjnych sieciach neuronowych.

Mimo wielu zalet w stosunku do klasycznych metod uczenia maszynowego oraz udokumentowanych zastosowań w rozwiązywaniu rzeczywistych problemów, metody oparte na sieciach konwolucyjnych klasyfikujących piksele wyłącznie na podstawie informacji spektralnych nie zawsze okazują się być najlepszą z dostępnych możliwości. Ich skuteczność może być ograniczona (szczególnie w domenie związanej z teledetekcją), np. poprzez negatywny wpływ szumu mogącego występować w danych [14]. Istnieją też inne rozwiązania, również bazujące na uczeniu głębokim, które są w stanie zapewnić jeszcze większą skuteczność i dokładność klasyfikacji.

W [24] można znaleźć alternatywne podejście do klasyfikacji pikseli w obrazach hiperspektralnych wykorzystujące konwolucyjne sieci neuronowe. Dla danych wejściowych stosowany jest algorytm PCA do projekcji danych hiperspektralnych na 3-kanałowy obraz, którego klasyfikacja jest przeprowadzana z wykorzystaniem dwuwymiarowej konowlucyjnej sieci neuronowej. Wada tej metody polega na tym, że traktuje ona obrazy hiperspektralne podobnie do RGB. Przetwarzając dane hiperspektralne z wykorzystaniem znanych rozwiązań wizji komputerowej można utracić specyficzne właściwości tego typu danych.

Próbując obejść ten problem, w [45] zaproponowano inne podejście. Do przetworzenia hipersześcianu wykorzystane zostały dwa osobne modele. Pierwszy wyodrębnia cechy widmowe wzdłuż wymiaru radiometrycznego, natomiast drugi, oparty na dwuwymiarowej konwolucyjnej sieci neuronowej wydobywa cechy przestrzenne (w podobny sposób jaki opisano we wcześniejszym akapicie). Funkcje dwóch modeli są następnie łączone poprzez wykorzystanie ich wyników przez klasyfikator. W taki sposób przeprowadzana jest klasyfikacja przestrzenno-spektralna.

Zrozumienie istotnego znaczenia informacji spektralnej prowadzi również do innych sposobów analizy obrazów hiperspektralnych np. takich, w których dane hiperspektralne są przetwarzane bezpośrednio i kompleksowego w procesie głębokiego uczenia. Aby rozwiązać problem spektralnej redukcji wymiarów, w kilku pracach zaproponowano naprzemienne stosowanie konwolucji przestrzennych i widmowych, w celu regularnego zmniejszania rozmiaru mapy cech. W szczególności

w [3] przedstawiono konwolucyjną sieć neuronową, która bierze pod uwagę oba typy sąsiedztwa (przestrzenne i spektralne) dla przetwarzanego piksela. Danymi podawanymi na wejścia sieci są wycinki obrazu hiperspektralnego (ang. *patches*). Pierwsze warstwy zmniejszają wymiar spektralny przy użyciu jądra $1 \times 1 \times n$ (głębokość n dobierano w taki sposób, aby była mniejsze od głębokości obrazu), a następne redukują wymiary przestrzenne wykorzystując filtr $k \times k \times 1$ (przyjmując za k wartość 3 lub 5). Proces ten jest powtarzany, by ostatecznie, dwie w pełni połączone warstwy mogły wykonać ostatni etap klasyfikacji.

Chociaż metody przestrzenno-spektralne osiągną często wystarczająco satysfakcjonujące wyniki, to wymagają one wysokiego poziomu inżynierii przy ich konstruowaniu. W podejściu zaprezentowanym w [13] sześcian hiperspektralny również jest przetwarzany kompleksowo przez pojedynczy model. Wykorzystana do tego została trójwymiarowa sieć konwolucyjna, które działa jednocześnie na trzech wymiarach. To koncepcyjnie prostsze podejście jest w stanie wprowadzić nieznaczną poprawę w wynikach klasyfikacji w odniesieniu do rozwiązań wykorzystujących dwa modele (osobny do ekstrakcji cech przestrzennych oraz osobny do ekstrakcji cech spektralnych). Zaproponowano wiele architektur trójwymiarowych splotowych sieci neuronowych w celu zbadania jak technik głębokiego uczenia dobrze znane z wizji komputerowej radzą sobie w przypadku bardziej złożonych danych. Trójwymiarowe mapy aktywacji tworzone przez takie sieci wydają się być przynajmniej teoretycznie bardziej odpowiednie dla danych hiperspektralnej.

W [5] jest szczególnie widoczne, że wykorzystanie trójwymiarowych konwolucyjnych sieci neuronowych do klasyfikacji obrazów hiperspektralnych pozwala na uzyskanie lepszych wyników niż w przypadkach gdy do tego samego zadania używane są sieci dwuwymiarowe. W odniesieniu do architektur jednowymiarowych, lub łączących podejścia spektralne z przestrzennym za pomocą dwóch sieci, trójwymiarowe sieci konwolucyjne łączą obie strategie rozpoznawania wzorców w ramach pojedynczego filtra, dzięki czemu mogą wymagać mniej parametrów i warstw.

Rozdział 3

Segmentacja obrazów hiperspektralnych przy użyciu trójwymiarowych konwolucyjnych sieci neuronowych

W tym rozdziale spisano rozważania dotyczące wpływu hiperparametrów oraz poszczególnych elementów budujących konwolucyjne sieci neuronowe na skuteczność segmentacji obrazów hiperspektralnych, aby ostatecznie przyjąć, które z nich zostaną przebadane w ramach tej pracy. W dalszej części znajduje się uzasadnienie wyboru konkretnych metryk służących do oceny modelu podczas przeprowadzonych badań eksperymentalnych. W ostatniej części rozdziału opisano sposób, w jaki z dostępnych dane zostaną wyodrębnione te służące do nauki modelu oraz te służące do jego ewaluacji.

3.1 Dobór architektury i hiperparametrów konwolucyjnej sieci neuronowej

Z dostępnej literatury wynika, że to właśnie trójwymiarowe konwolucyjne sieci neuronowe są w stanie osiągać najwyższą skuteczność przy segmentacji obrazów hiperspektralnych. Do tej pory zaproponowano wiele takich sieci. Dwie sieci różnią

się od siebie architekturą jeżeli posiadają one inną liczbę warstw, bądź warstwy z których są skonstruowane są innego typu. Dla przykładu, w niektórych modelach w ogóle nie występują warstwy łączące, w innych taka warstwa występuje tylko jedna (zazwyczaj na końcu), ale zdarzają się też takie, w których warstwy konwolucyjne i łączące są ze sobą przeplatane. Różnice w konwolucyjnych sieciach neuronowych mogą też być bardziej subtelne i wprowadzać zmiany wyłącznie w obrębie hiperparametrów. Zazwyczaj sieci działające na surowych danych hiperspektralnych różnią się od tych przetwarzających dane ze zredukowaną wcześniej ilością wymiarów spektralnych. W ramach badań sprawdzona została możliwość użycia takiej samej architektury konwolucyjnej sieci neuronowej dla obu tych przypadków. Jako że kolejne warstwy w ramach sieci zmniejszają rozmiary obrazu wejściowego, obiecującą metodą radzenia sobie z dużą ilością wymiarów wydaje się być dodawanie kolejnych warstw, a w szczególności użycie warstw łączących.

Ponieważ wykorzystywane w teledetekcji skanery hiperspektralne różnią się specyfikacją i generują dane o odmiennych strukturach, algorytmy używane do ich przetwarzania bywają odpowiednio dostosowywane do analizowanego zbioru. Niektóre parametry sieci, takie jak kształt danych wejściowych (a przynajmniej wielkość wymiaru spektralnego), czy ilość klas wyjściowych, zależą bezpośrednio od zbioru danych i są raczej niemożliwe do uogólnienia, jednak architektura i inne hiperparametry mogły by teoretycznie pozostać stałe. W ramach niniejszej pracy wykonano badania sprawdzające na ile satysfakcjonujące wyniki można uzyskać przy użyciu tej samej architektury do analizy różnych zdjęć hiperpektralnych oraz czy dostosowywanie hiperparametrów sieci dla każdego zbioru danych skutkuje znacznym polepszeniem wyników.

Proces badawczy zakładał ciągłe modyfikowanie struktury i hiperparametrów sieci neuronowej. Architekturę początkowego modelu, którego wyniki stanowiły punkt wyjścia dla kolejnych eksperymentów przedstawiono na rysunku 3.1. Model składał się z dwóch warstw konwolucyjnych (na schemacie są one zaznaczone strzałkami), jednej warstwy w pełni połączonej oraz warstwy wyjściowej (warstwa w pełni połączona została przedstawiona na schemacie wraz z warstwą wyjściową). Każda warstwa konwolucyjna posiadała cztery filtry. Rozmiar filtrów wynosił $3 \times 3 \times 7$ dla warstwy pierwszej, natomiast dla warstwy drugiej $3 \times 3 \times 5$. Funkcją aktywacji w ostatniej warstwie była funkcja *Softmax*, natomiast wszyst-

kie pozostałe warstwy korzystały z funkcji *ReLU*. Regularyzacji dokonano przy użyciu metody *L2* z parametrem 0.01. Rozmiar pojedynczego przykładu podawanego na wejścia sieci był równy $5 \times 5 \times 15$ (wysokość, szerokość, liczba pasm) dla danych, których wymiarowość została zredukowana przy pomocy algorytmu PCA (na rysunku przedstawiono wejście sieci przy założeniu przetwarzania właśnie takich danych). Przykłady utworzone bezpośrednio z surowych danych posiadały tę samą wysokość i szerokość, natomiast liczba pasm była różna w zależności od zbioru (103, 200 lub 224). W warstwie w pełni połączonej umieszczono 144 neurony. Liczba neuronów warstwy wyjściowej odpowiadała liczbie klas obecnej w przetwarzanym zbiorze danych (9 dla *Pavia University* i 16 dla *Salinas* i *Indian Pines*, oznaczona na schemacie jako N). Maksymalna liczba epok została ustalona na poziomie czterdziestu, natomiast w przypadku gdy przez cztery epoki model nie był w stanie osiągnąć lepszego wyniku (co rozumie się poprzez uzyskanie większej dokładności dla zbioru walidacyjnego) proces uczenia był przerywany wcześniej. Liczebność partii wyniosła 128. Do optymalizacji funkcji kosztu wybrano algorytm *Adam*. Przyjęta funkcja kosztu to kategoryczna entropia krzyżowa.

3.2 Ewaluacja modelu

Dobranie odpowiedniej topologi sieci często zależy od tego, jaka metryka jest uważana za kluczową. Dokładniejsze modele mogą wymagać dłuższego czasu treningu natomiast te klasyfikujące szybciej mogą cechować się zbyt dużym dopasowaniem do danych. Używanie bardziej złożonych modeli może czasem okazać się niemożliwe ze względu na wyższe wymagania sprzętowe. Dla niektórych zastosowań ważniejsza od ogólnej dokładności klasyfikatora może okazać się dokładność dla pewnej wybranej klasy. Można tu mówić o innym koszcie w zależności od liczby pikseli poprawnie zaklasyfikowanych do wybranej klasy (ang. True Positive, TP), liczby pikseli nie należących do wybranej klasy, które zostały poprawnie zaklasyfikowanych do innej niż wybrana (ang. True Negative, TN), liczby pikseli zaklasyfikowanych do wybranej klasy, podczas gdy w rzeczywistości należą do innej (ang. False Positive, FP), bądź liczby pikseli zaklasyfikowanych do jednej z pozostałych klas, podczas gdy w rzeczywistości należą do wybranej klasy (ang. False Negative, FN). Badania wykonane w ramach tej pracy, nie narzucają kontekstu w

jakim przeprowadzana jest klasyfikacja, dlatego też dla każdego z przeprowadzonych eksperymentów zmierzone zostały następujące cechy wykorzystanego w nim modelu:

- Dokładność – (ang. *accuracy*) wyraża stosunek poprawnie dokonanych klasyfikacji do ilości wszystkich dokonanych klasyfikacji. Wyrażana zazwyczaj w procentach. Jest dosyć ogólna i często wykorzystuje się ją gdy dla danego zadania nie wybrano jeszcze najistotniejszej metryki. W przypadku gdy zbiory reprezentujące poszczególne klasy są niezrównoważone, może okazać się zwodnicza.

- Współczynnik kappa – informuje gdzie w skali pomiędzy klasyfikacją losową a idealną, znajduje się dany klasyfikator. Jego wartości mieszczą się w przedziale $\langle 0, 1\rangle$, gdzie 0 oznacza model losowy a 1 idealny klasyfikator. Przyjmując oznaczenie T dla liczby przykładów w zbiorze testowym, A_k dla dokładności klasyfikatora i A_l dla dokładności klasyfikatora losowego, współczynnik kappa można wyrazić jako $kappa = \frac{A_k - A_l}{T - A_l}$.

- Czas treningu - czas, który był niezbędny do wypracowania odpowiednich wag połączeń między neuronami (nauki modelu) przy wykorzystaniu zbioru treningowego.

- Czas predykcji - czas, który był niezbędny do dokonania predykcji dla wszystkich przykładów ze zbioru testowego.

Ponadto, dla każdej klasy występującej w zbiorze testowym zapisane zostały wartości dla poniższym metryk:

- Precyzja – (ang. *precision*) poprzez stosunek poprawnie dokonanych klasyfikacji dla danej klasy do wszystkich (poprawnych i niepoprawnych) klasyfikacji dla tej klasy określa stopień w jakim można zaufać pozytywnym predykcjom w danej klasie (obliczana ze wzoru $Precyzja = \frac{TP}{TP+FP}$).

- Czułość – (ang. *recall*) za pomocą stosunku poprawnie dokonanych klasyfikacji dla danej klasy do liczby wszystkich przykładów należących do tej klasy określa ile obserwacji zostało „zgubionych” dla danej klasy (obliczana ze wzoru Czułość $= \frac{TP}{TP+FN}$).

- F1 – (ang. *F1-score*) pozwala na wyrażenie precyzji i czułości w ramach jednej wartości, będącej średnią harmoniczną obu (obliczana ze wzoru $F1 = 2\frac{precyzja \times \text{czułość}}{precyzja + \text{czułość}}$).
- Wsparcie – (ang. *support*) ilość przykładów reprezentujących daną klasę w zbiorze treningowym.

3.3 Podział danych na zbiory treningowe i testowe

Metody bazujące na uczeniu maszynowym bądź uczeniu głębokim, a w szczególności te wykorzystujące sieci neuronowe są nierozerwalnie związane z takimi pojęciami jak zbiór treningowy oraz zbiór testowy.

Zbiorem treningowym nazywa się zastaw danych, które są wykorzystane podczas uczenia modelu. To właśnie występujące w nich zależności i cechy charakterystyczne muszą zostać odszukane, a następnie utrwalone w postaci wag. Często jeden przykład ze zbioru treningowego jest wykorzystany w procesie uczenia modelu wielokrotnie.

Zbiór testowy natomiast, służy do ostatecznej oceny modelu. Składa się z przykładów, które nie zostały użyte w treningu. Dzięki temu można zweryfikować w czy dany model będzie w wystarczająco skuteczny w przypadku przetwarzania nowych danych, co jest istotą uczenia maszynowego.

Aby upewnić się, że sieć nie dopasowuje się zanadto do danych treningowych, bardzo często wprowadza się jeszcze jeden zbiór - walidacyjny. Najczęściej tworzy się go poprzez wyodrębnienie pewnej liczby przykładów ze zbioru treningowego (zbiór treningowy, testowy i walidacyjny są więc rozłączne). Zbiór walidacyjny, podobnie jak treningowy, jest używany podczas uczenia. Nie ma on jednak bezpośredniego wpływu na aktualizację wag sieci. Po każdym, jednorazowym wykorzystaniu wszystkich przykładów uczących (epoce), model jest oceniany. Ewaluacja bazująca wyłącznie na zbiorze treningowym może okazać się niewystarczająca z powodu wspomnianego wcześniej dopasowania do danych treningowych, dlatego w celu otrzymania bardziej wiarygodnych wyników, przeprowadza się ją na osobnym zbiorze. Opierając się na metrykach wyznaczonych przy pomocy zbioru walidacyj-

nego, można w odpowiedni sposób reagować na zmiany zachodzące w modelu np. poprzez wcześniejsze zakończenie procesu uczenia. Otrzymując zadowalające wyniki dla zbioru walidacyjnego, można oczekiwać, że gdy zupełnie nowy przykłady zostanie przedstawiony sieci, sklasyfikuje go ona z podobną dokładnością.

Pojedynczym przykładem podawanym na wejścia modelu jest kostka danych, będąca wycinkiem obrazu hiperspektralnego. Jej szerokość oraz wysokość dobierane są w taki sposób, aby były mniejsze od odpowiadających rozmiarów przetwarzanego obrazu co najmniej o rząd wielkości. Wszystkie informacje spektralne są jednak zachowane, a więc głębokość wycinka odpowiada głębokości obrazu. Etykietą dla takiego przykładu jest klasa, która została przypisana przez eksperta środkowemu pikselowi.

W pracach o podobnej tematyce proces podziału danych na zbiór testowy i treningowy wygląda zazwyczaj bardzo podobnie. Pierwszym etapem jest stworzenie wspólnego zbioru zawierającego wszystkie możliwe wycinki obrazu o zadanej wielkości wraz z odpowiadającymi im etykietami. Oznacza to, że dla każdego piksela w obrazie tworzony jest wycinek, dla którego rozpatrywany piksel stanowi środek. Piksele, których nie przyporządkowano do żadnej z klas są pomijane. Wynika to z faktu, że piksele stanowiące tło posiadają zazwyczaj bardzo różną charakterystykę spektralną (reprezentują dowolne obiekty znajdujące się na ziemni, nie będące przedmiotem zainteresowania). W przypadku gdy tworzony jest wycinek dla piksela, którego odległość od najbliższej krawędzi obrazu jest mniejsza od połowy długości boku tego wycinka, brakujące wartości uzupełnia się zerami, bądź poprzez skopiowanie wartości pikseli znajdujących się na skraju obrazu. Następnie, ze wszystkich przykładów utworzonych w ten sposób, losowane są te, które znajdą się w zbiorze testowym, oraz te, które trafią do zbioru treningowego. Opcjonalnie, część przykładów może stanowić zbiór walidacyjny. Stosunek przykładów, które mają trafić do poszczególnych zbiorów jest podawany jako parametr.

Metoda ta ma jednak pewną niedoskonałość, która znaczenie uwydatnia się wraz ze wzrostem stosunku liczebności zbioru treningowego do liczebności zbioru testowego. Rozpatrując dwa wycinki utworzone dla pary bezpośrednio sąsiadujących ze sobą pikseli, można zauważyć, że zazwyczaj przedstawiany przez nie obszar jest bardzo podobny. Wynika to z tego, że jeden piksel to zbyt mała odległość, aby kolejne z generowanych przykładów znacząco się od siebie różniły. Sytuacja ta,

nieco uproszczona, została zobrazowana na rysunku 3.2. W ukazanym przykładzie dwa wycinki różnią się tylko wartością jednego piksela. Ponadto, oba te przykłady otrzymają taką samą etykietę (odpowiadającą klasie reprezentowaną przez kolor zielony). W przypadku gdy oba te przykłady znajdą się w tym samym zbiorze (treningowym, walidacyjnym lub testowym) nie będzie miało to większego wpływu na wyniki klasyfikacji. Jednak gdy jeden z tych przykładów znajdzie się w zbiorze treningowym, a drugi w zbiorze testowym (co jest możliwe, gdy podział ten jest wykonywany według algorytmu opisanego w poprzednim akapicie), ocena modelu przedstawiona za pomocą różnych miar może być nieco zawyżony. Jest to skutkiem tego, że sieć „widziała" już bardzo podobny przykład, do tego na którym jest testowana. Ponadto, sytuacja w której dwa podobne przykłady znajdują się w dwóch osobnych zbiorach nie oddaje rzeczywistej. Jednym z potencjalnych scenariuszy wykorzystujących sztuczną sieć neuronową do segmentacji obrazów hiperspektralnych jest rozpoznawanie obszarów o wcześniej wyuczonej charakterystyce na nowych terenach. Można więc przyjąć, że dane analizowane po zakończonym treningu modelu będą różnić się w bardziej znaczący sposób.

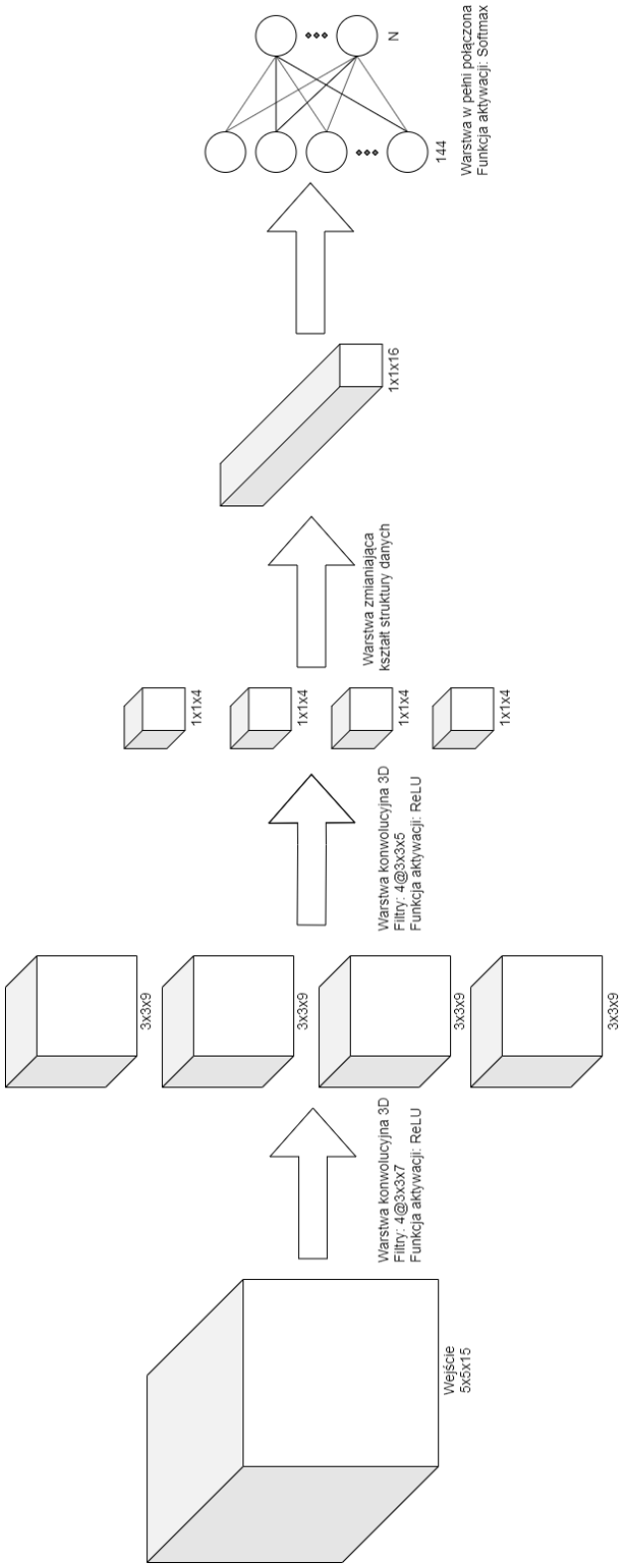
Użyta podczas badań metoda stara się rozwiązać opisany w poprzednim akapicie problem i ma lepiej odwzorowywać rzeczywisty scenariusz użycia głębokich sieci neuronowych do segmentacji obrazów hiperspektralnych w domenie związanej z teledetekcją. Zaproponowany algorytm podziału danych na zbiory treningowy, testowy oraz walidacyjny opiera się na koncepcji usuwania z obrazu pod-obszarów (większych od docelowych przykładów) i tworzenia na ich podstawie zbioru testowego oraz walidacyjnego. Przez pod-obszar rozumie się część obrazu o kształcie kwadratu, którego wysokość i szerokość są definiowane przez parametr algorytmu, przy czym wymiar ten powinien być co najmniej dwa razy większy od wymiarów wycinka (pojedynczego przykładu). Podczas tworzenia zbioru testowego wyznacza się taką liczbę pod-obszarów, które łącznie będą pokrywać określony procent obrazu (ten procent również jest parametrem algorytmu). Określenie położenia każdego pod-obszaru realizowane jest w sposób losowy przy czym pod-obszary nie mogą na siebie nachodzić (jeden piksel może znajdować się tylko w jednym pod-obszarze). Następnie z każdego pod-obszaru wyznaczane są wycinki. Do zbioru testowego trafiają jednak tylko te, dla których odległość środkowego piksela od krawędzi pod-obszaru jest większa lub równa od wymiaru wycinka. Podejście to

zapewnia, że przykłady znajdujące się w zbiorze testowym nie będą współdzielić ze zbiorem treningowym ani walidacyjnym żadnego piksela. Podział pikseli w obrębie jednego pod-obszaru oraz tych znajdujących się w jego bliskim sąsiedztwie został przedstawiony na rysunku 3.3. Zastosowanie pewnego marginesu (białe piksele na wspomnianym rysunku), pozwalającego na separacje zbiorów wiąże się jednak z pewnym kosztem. Część pikseli nie zostanie przekształcona na przykłady, to znaczy, że w żadnym ze zbiorów nie pojawią się wycinki, dla których te piksele stanowiłyby środek. Zbiór walidacyjny tworzony jest za pomocą pod-obszarów o takich samych rozmiarach oraz z takim samym marginesem wewnętrznym. Podobnie jak w ma to miejsce podczas tworzenia zbioru treningowego, tak też w tym przypadku ich umiejscowienie pod-obszarów wybierane jest w sposób losowy. Jeżeli pod-obszar walidacyjny nachodzi na pod-obszar testowy to pod-obszar walidacyjny jest w odpowiedni sposób ograniczany. Taka sytuacja (dla przykładów o rozmiarach 3×3 piksele) została przedstawiona na rysunku 3.4. Widać, że część pikseli w ramach pod-obszaru walidacyjnego musi zostać odrzucona ze względu na znajdujący się niedaleko pod-obszar testowy. Mimo, że zmniejsza to ostateczną liczbę przykładów, które znajdą się w zbiorze walidacyjnym, zdecydowano się na takie podejście, ponieważ w znacznym stopniu ułatwia to generowanie pod-obszarów. Ponadto stosowanie zbioru walidacyjnego o znacznie mniejszej liczebności od zbioru treningowego jest powszechną praktyką. Liczba pod-obszarów walidacyjnych określana jest na podstawie obliczonej wcześniej liczbie pod-obszarów testowych (stanowi połowę tej wartości). Zbiór treningowy powstaje z wszystkich pikseli, które pozostały w obrazie po usunięciu z niego pod-obszarów testowych i walidacyjnych.

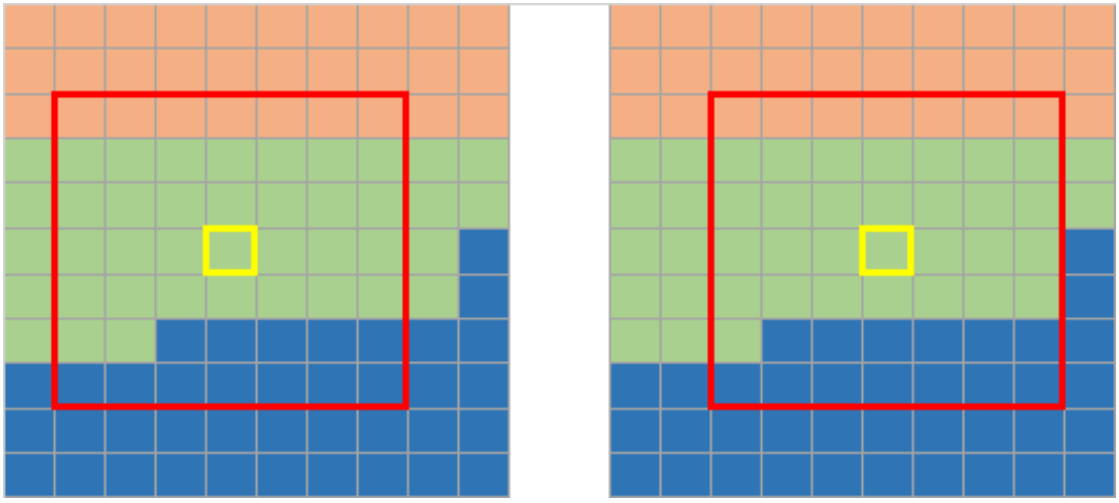
Opisana metoda, chociaż rozwiązuje pewne problemy związane z podziałem danych, sama nie jest wolna od wad. Wspomniany wcześniej koszt „utraconych" pikseli jest tylko jedną z niedoskonałości. Czas działania algorytmu jest nieco dłuższy niż w przypadku standardowej metody podziału. Ponadto, przy nieodpowiednio dobranych parametrach, metoda ta może w ogóle nie zadziałać. Miejsca pod-obszarów testowych losowane są w sposób iteracyjny (jeden po drugim). W przypadku, gdy kolejny pod-obszar testowy miałby pokrywać część istniejącego już pod-obszaru tego samego typu, algorytm stara się znaleźć inne miejsce dla nowego pod-obszaru (losując je). Jeżeli po określonej liczbie prób, nadal nie jest możliwe wygenerowanie kolejnego pod-obszaru, algorytm przerywa działanie. Taka sytu-

acja może mieć miejsce, kiedy pod-obszary testowe będą miały stanowić znaczący procent całego obrazu. Wspomniany procent pokrycia obrazu przez pod-obszary testowe (będący parametrem algorytmu) nie gwarantuje też analogicznego stosunku w liczebności poszczególnych zbiorów. Wpływa na to z faktu, że do zbioru treningowego, testowego i walidacyjnego nie trafiają przykłady, których środkowy piksel stanowi tło. Jeżeli stosunek pikseli opisanych jedną z klas do pikseli będących tłem jest znacząco większy dla pod-obszarów testowych niż dla reszty obrazu, wtedy liczebność zbioru testowego może również być większa od oczekiwanej.

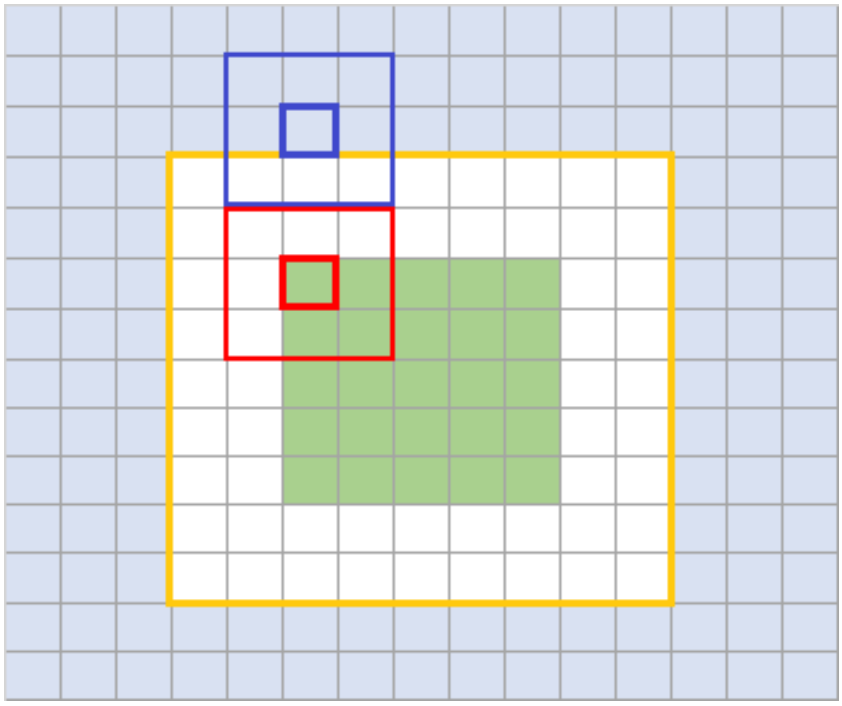
Rozważając wady i zalety obu metod opisanych w tym podrozdziale zdecydowano, by w badaniach porównać wpływ obu sposobów podziału danych na wyniki klasyfikacji. Większość badań, zostanie przeprowadzona jednak przy wykorzystaniu metody mającej lepiej oddawać prawdziwy przypadek użycia.



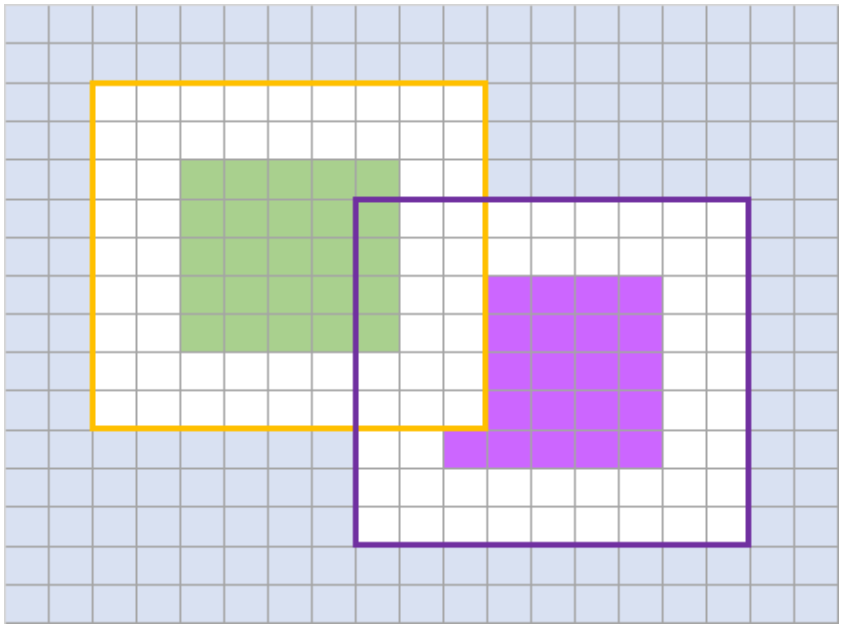
Rysunek 3.1: Schemat konwolucyjnej sieci neuronowej wykorzystanej w badaniach.



Rysunek 3.2: Dwa przykładowe wycinki utworzone dla sąsiadujących pikseli. Klasyfikowany piksel zaznaczono żółtą ramką. Zawartość wycinka zaznaczono czerwoną ramką. Kolory pomarańczowy, zielony i niebieski symbolizują różne klasy.



Rysunek 3.3: Pod-obszar testowy (żółta obwódka), piksele będące środkami wycinków ze zbioru testowego (zielone), wycinek ze zbioru testowego (czerwona obwódka), piksele będące środkami wycinków ze zbioru treningowego (jasnoniebieskie), wycinek ze zbioru treningowego (niebieska obwódka).



Rysunek 3.4: Pod-obszar testowy (żółta obwódka), piksele będące środkami wycinków ze zbioru testowego (zielone), pod-obszar walidacyjny (fioletowa obwódka), piksele będące środkami wycinków ze zbioru walidacyjnego (fioletowy), piksele będące środkami wycinków ze zbioru treningowego (jasnoniebieskie).

Rozdział 4

Badania

Rozdział ten zawiera opis bibliotek i narzędzi wykorzystanych do implementacji wybranych algorytmów, informacje na temat metody prowadzenia badań eksperymentalnych oraz zwięzłą charakterystykę danych wykorzystanych podczas przeprowadzonych eksperymentów. W dalszej części tego rozdziału umieszczone zostały wyniki badań wraz z ich analizą.

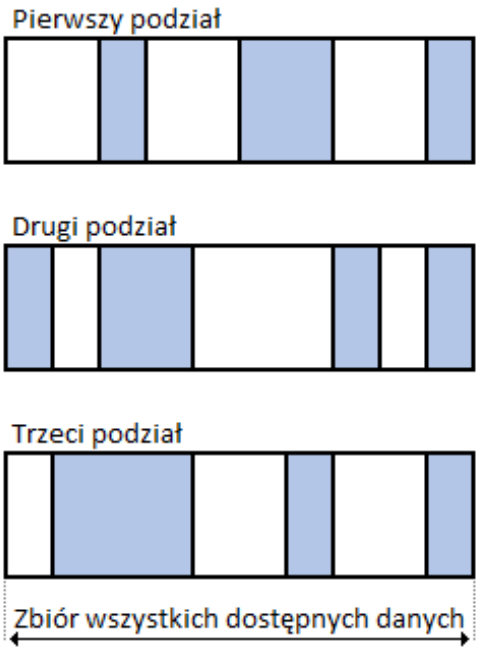
4.1 Implementacja

Wszystkie badania eksperymentalne wykonane w ramach niniejszej pracy zostały przeprowadzone przy użyciu programu napisanego w języku `Python` w wersji 3.7.3. Wykorzystano przy tym zewnętrzną bibliotekę `Keras` w wersji 2.2.4, która znacznie ułatwia tworzenie aplikacji wykorzystujących techniki głębokiego uczenia. Ponadto umożliwia ona wykorzystanie procesora graficznego do przyśpieszenia obliczeń wykonywanych podczas trenowania modelu. Inne zewnętrzne pakiety wykorzystane podczas implementacji to między innymi `sckitlearn` w wersji 0.21.2 (użyty do przeprowadzenia wstępnej redukcji danych za pomocą algorytmu PCA), czy `spectral` w wersji 0.19 (użyty do wizualizacji zdjęć hiperspektralnych). Prezentowane w dalszej części pracy wyniki są efektem obliczeń przeprowadzonych na stacji roboczej wyposażonej w procesor Intel Core i5-2500 (4 rdzenie, taktowanie 3.3 GHz – 3.6 GHz), 8GB RAM oraz kartę graficzną NVIDIA GeForce GT 1030.

4.2 Metodyka badań

Opisany w w poprzednim rozdziale algorytm podziału danych na zbiór treningowy, testowy i walidacyjny w dość dużym stopniu opiera się na losowości. Przy niekorzystnym losowaniu może dojść do sytuacji, w której wszystkie przykłady danej klasy znajdą się w poza zbiorem treningowym nie dając modelowi szansy na poznanie ich struktury. Może to w znaczący sposób zaniżyć oceną takiego modelu. Możliwa jest też sytuacja odwrotna, w której model będzie sprawiał wrażenie lepszego, jeżeli wybrane zastawy danych będą odpowiednio korzystne. Aby zredukować prawdopodobieństwo wystąpienia takich zachowań, podczas badań została zastosowana walidacja krzyżowa monte carlo (ang. *monte carlo cross-validation*). Jej ogólne działanie polega na wyborze spośród wszystkich dostępnych danych, losowych próbek, które trafią do zbioru testowego (losowanie odbywa się bez zwracania) oraz umieszczenia pozostałych w zbiorze treningowym. Proces ten powtarzany jest wiele razy, za każdym razem generując nowy zestaw zbiorów treningowych i testowych. Jako, że podział dokonywany jest niezależnie dla każdego przebiegu, ta sama próbka może pojawić się w wielu zbiorach treningowych bądź testowych. Oceny modelu dokonuje się na podstawie wszystkich takich zestawów. Oznacza to, że model zostanie wytrenowany i przetestowany wielokrotnie. Dla każdego modelu zostaną też obliczone wszystkie metryki służące do jego oceny, których wartości zostaną następnie uśrednione. Przykładowy wynik działania tego typu walidacji krzyżowej tworzącej trzy różne podziały został zobrazowany na rysunku 4.1.

Opisana techniki walidacji krzyżowej jest tylko ogólnym sposobem postępowania i nic nie stoi na przeszkodzie by podczas generowania zbiorów, część przykładów umieścić w zbiorze walidacyjnym, bądź by sposób wybierania próbek był zdefiniowany w nieco inny sposób (o ile nadal jest oparty o losowość). Jak najbardziej da się więc przenieść tę koncepcję na opisany w poprzednim rozdziale algorytm podziału danych. Dodatkową korzyścią płynącą z takiego podejścia jest łatwa możliwość zapisania wszystkich zbiorów danych lub informacji potrzebnych do ich odtworzenia. Pozwala to na ewentualne powtórzenie eksperymentów, jeżeli z jakiegoś powodu zajdzie taka potrzeba.



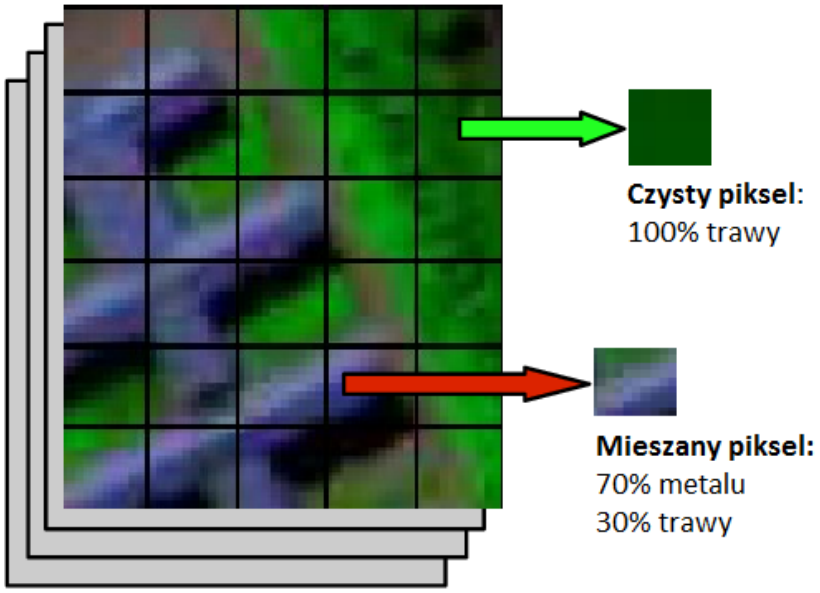
Rysunek 4.1: Wynik walidacji krzyżowej monte carlo tworzącej trzy zestawy zbiorów treningowych (oznaczone kolorem niebieskim) i testowych (oznaczone kolorem białym).

4.3 Zbiory danych

Z obrazami satelitarnymi, bądź tymi wykonywanymi z drona, czy samolotu, wiążą się nieodłącznie takie pojęcia jak rozdzielczość spektralna oraz rozdzielczość przestrzenna. Rozdzielczość spektralna mówi, jaka liczba kanałów jest rejestrowana podczas jednej aktywacji, natomiast rozdzielczość przestrzenna opisuje wielkość, jakiej jeden piksel (czyli pojedynczy element obrazu) odpowiada w rzeczywistości. Często zdarza się, że jeden piksel pokrywa więcej niż jeden typ pokrycia terenu (ang. *mixed pixel*). Przykład takiego piksela jest widoczny na rysunku 4.2. Im więcej takich pikseli znajduje się na analizowanym zdjęciu, tym trudniejsza jest jego analiza oraz interpretacja [39].

Wszystkie opisane w tym podrozdziale zbiory danych zostały wykorzystane przy ewaluacji zaproponowanego modelu. Przez swoją szeroką dostępność oraz róż-

norodność są one wykorzystywane w wielu publikacjach dotyczących segmentacji obrazów hiperspektralnych. Różnice występują głównie w rozdzielczości spektralnej i przestrzennej, ilości klas oraz rozkładzie klas.



Rysunek 4.2: Przykład piksela zawierającego więcej niż jedną klasę oraz piksela reprezentującego tylko jedną klasę. Źródło: [39].

Pavia University

Dane zostały zebrane z wykorzystaniem skanera ROSIS. Fotografowany obszar obejmuje tereny znajdujące się w pobliżu Uniwersytetu w Pawii (północne Włochy). Obraz zawiera informacje o 103 kanałach (w zakresie 430 do 860 nm) dla 610×340 pikseli. Rozdzielczość przestrzenna wynosi 1,3m. W obszarze zdefiniowanym przez eksperta 42776 pikseli zostało przyporządkowanych do dziewięciu różnych klas. Pozostałe (niesklasyfikowane) piksele stanowią tło. Dane zapisane są w formacie charakterystycznym dla programu MATLAB (`*.mat`).

Salinas

Zdjęcie zostało wykonane w Salinas Valley znajdującego się w Kalifornii, w Stanach Zjednoczonych, przy użycia sensora AVRIS. Dla fotografowanego obszaru

zarejestrowane zostały 224 kanały. Wysokość obrazu wynosi 512, natomiast szerokość 217 pikseli. Dwadzieścia kanałów znajdujących się w oryginalnych danych, dla których woda wykazuje się dużą absorpcją promieniowania elektromagnetycznego, zostało odrzuconych przed dalszą analizą. Długości rejestrowanych fal obejmuje zakres od 400 do 2500 nm. Rozdzielczość przestrzenna wynosi 3.7 m. Obszar zdefiniowany przez eksperta zawiera 16 klas, które przedstawiają głównie różne rodzaje zbóż i roślin uprawnych. Wizualizacje tego zbioru danych przedstawia rysunek 4.3.

Rysunek 4.3: Zdjęcie tworzące zbiór danych Salinas. Po lewej stronie fotografowana scena zwizualizowana za pomocą sztucznych kolorów (użyto kanałów numer 9, 19 i 21 z dostępnych 204), po prawej wyszczególnione przez eksperta obszary wraz z legendą.

Indian Pines

Fotografia została wykonany przy użyciu sensora AVIRIS. Przedstawia Indian Pines w północno-wschodniej Indianie (USA). Długości fali wykorzystanej do stwo-

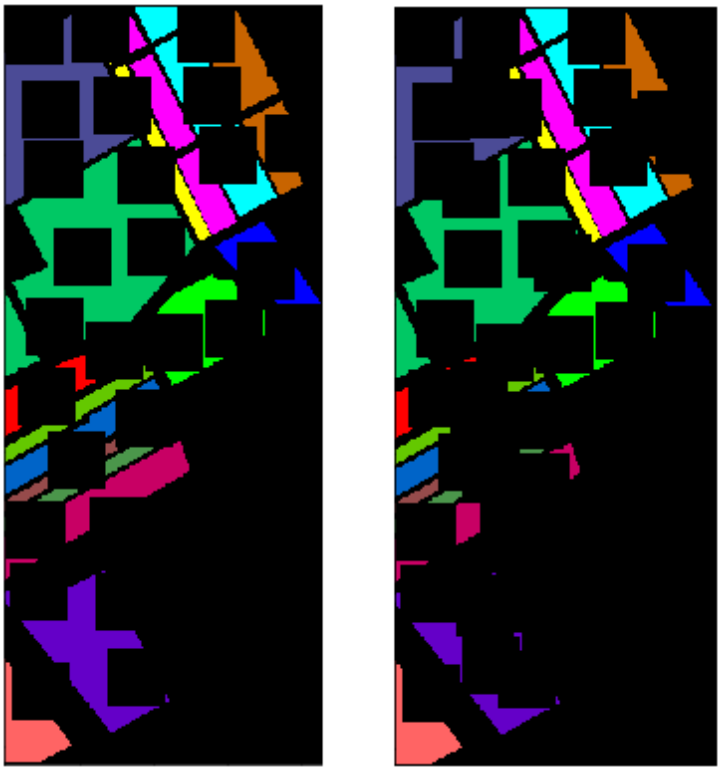
rzenia tych danych waha się od 400 do 2500 nm. Informacje spektralne zostały zapisane w 224 kanałach z których 12 zostało usuniętych z powodu dużego szumu. Rozmiar obrazu wynosi 145 × 145 pikseli. Dwie trzecie sceny Indian Pines pokrywają tereny rolnicze, a jedna trzecia przedstawia lasy lub inne naturalnie występujące rośliny wieloletnie. Na zdjęciu znajdują się również dwie linie autostrady, linia kolejowa, a także kilka domów oraz mniejsze drogi. Ponieważ zdjęcie zostało wykonane w czerwcu, niektóre z obecnych na nim upraw (np. kukurydza czy soja), znajdują się we wczesnych stadiach wzrostu z pokryciem mniejszym niż 5%. Obszar zdefiniowany przez eksperta podzielony został na szesnaście klas.

Przygotowanie danych

Z każdego z opisanych powyżej zdjęć hiperspektralnych wygenerowano pięć zestawów danych (każy złożony ze zbioru treningowego, walidacyjnego i testowego). Podziału dokonano wykorzystując algorytm działający w oparciu o pod-obszary (opisany w rozdziale trzecim). W każdym ze zbiorów treningowych znalazło się około 40% wszystkich przykładów (nie uwzględniając pikseli, które musiały zostać odrzucone z powodu zbyt małej odległośći od krawędzi pod-obszarów). Liczebność zbiorów testowych była podobna, natomiast dla zbiorów walidacyjncyh stanowiła około 20% dostępnych przykładów. Margines błędu, wynikający ze sposobu działania algorytmu dzielącego dane wynosił około 5% wszystkich oznaczonych pikseli występujących w ramach jednego zdjęcia hiperspektralnego. Na rysunku 4.4 zaprezentowano zbiór danych Salinas pozbawiony kolejnych typów pod-obszarów. Wszystkie przykłady, które pozostały na obrazie po usunięciu obu rodzajów pod-obszarów (prawa strona rysunku) stanowią zbiór treningowy.

4.4 Wyniki i analiza

Wyniki pierwszego eksperymentu (wykorzystującego architekturę opisaną w rozdziale trzecim) umieszczone zostały w tabeli 4.1. Wskazują na znacznie wyższą skuteczność modelu (uwzględniając każdą z wykorzystanych metryk) uczonego i testowanego na danych ze wstępnie zredukowaną liczbą wymiarów przestrzennych. Mimo wysokich wyników dla zbiorów *Salinas* i *Pavia University*, nie udało się



Rysunek 4.4: Zbiór danych Salinas. Obszar oznaczony przez eksperta po usunięciu pod-obszarów testowych (lewa strona) oraz walidacyjncyh (prawa strona).

uzyskać podobnej skutecznością przy przetwarzaniu zbioru *Indian Pines*. Może to wynikać z charakterystyki samego zbioru. Z racji mniejszych rozmiarów posiada on mniejszą liczbę przykładów, na których sieć mogłaby zdobywać wiedzę. Fakt, że przyjęta metoda podziału danych na zbiory treningowe, testowe i walidacyjne jeszcze bardziej ją ogranicza, również jest nie bez znaczenia. Kolejną kwestią mającą wpływ na słabsze wyniki uzyskane na tym zbiorze może być brak przykładów reprezentujących niektóre z klas w zbiorze treningowym.

Analiza wyników dla zbioru *Pavia University* otrzymanych przez sieć działającą na danych niezredukowanych, pozwala zaobserwować, że dokładność nie zawsze jest wystarczająco dobrym wskaźnikiem do oceny modelu. Jej średnia wartość dla tego zbioru wyniosła około 35%, co mogłoby być uznane za nie najgorszy punkt wyjścia. W rzeczywistości, każdy z modeli działających na tym zbiorze przypisywał wszystkim przykładom tę samą etykietę. Nie jest to pożądanym zachowaniem z

Tablica 4.1: Uśrednione wyniki modelu referencyjnego obliczone na zbiorze testowym z uwzględnieniem typu danych, na których przeprowadzono trening (dane poddane PCA oraz dane nie zredukowane).

	Salinas		Pavia University		Indian Pines	
	PCA	bez PCA	PCA	bez PCA	PCA	bez PCA
Dokładność	90.704	35.407	94.209	34.639	44.462	1.987
Kappa	89.481	25.215	92.329	0.000	35.369	0.000
Śr. precyzja	0.939	0.298	0.928	0.038	0.369	0.002
Śr. czułość	0.934	0.321	0.911	0.111	0.335	0.079
Śr. F1	0.930	0.288	0.916	0.055	0.316	0.003
Cz. treningu	14.539	55.981	12.813	18.270	7.044	6.643
Cz. predykcji	0.87	2.092	0.623	1.186	0.152	0.404

punktu widzenia rozwiązywanego problemu. W tabeli 4.2 przedstawiono dokładne wyniki pojedynczego eksperymentu przeprowadzonego na tym zbiorze (pozostałe miały podobny charakter). Wszystkie przykłady ze zbioru treningowego zostały oznaczone jako klasa *Meadows*. Jako że jest to najliczniejsza klasa ze wszystkich dostępnych w zbiorze treningowym, taka klasyfikacja skutkowała dokładnością na poziomi 46,53%.

Zerowe wartości uśrednionych współczynników Kappa dla modeli operujących na danych nie zredukowanych w zbiorach *Pavia University* oraz *Indian Pines* mogą świadczyć o niepoprawnym działaniu sieci. Warto jednak zauważyć, że w przypadku trzeciego zbioru wartość ta była niezerowa (ponieważ dwa z pięciu wyuczonych modeli wykazywały pewną zdolność do rozwiązania problemu klasyfikacji). W tabeli 4.3 przedstawiono szczegółowe wyniki lepszego modelu (wyższy współczynnik Kappa i dokładność). Chociaż wyniki nie były tak dobre jak w przypadku modeli działających na przykładach ze zredukowaną liczbą wymiarów przestrzennych, to uzyskany współczynnik Kappa na poziomie 78.45 i dokładność równa 75.64% pozwoliły na na dostrzeżenie pewnego potencjału w tym rozwiązaniu. Porównanie przebiegu treningu wszystkich pięciu modeli (Salinas, bez PCA), wykazało, że przyczyną braków postępów w nauce mógł być wysoki koszt początkowy. Osiągał on wartości większe od 11 dla sieci których obliczony współczynnik Kappa wynosił zero (te sieci przypisywały wszystkim przykładom jedną etykietę). Na wykresie 4.5

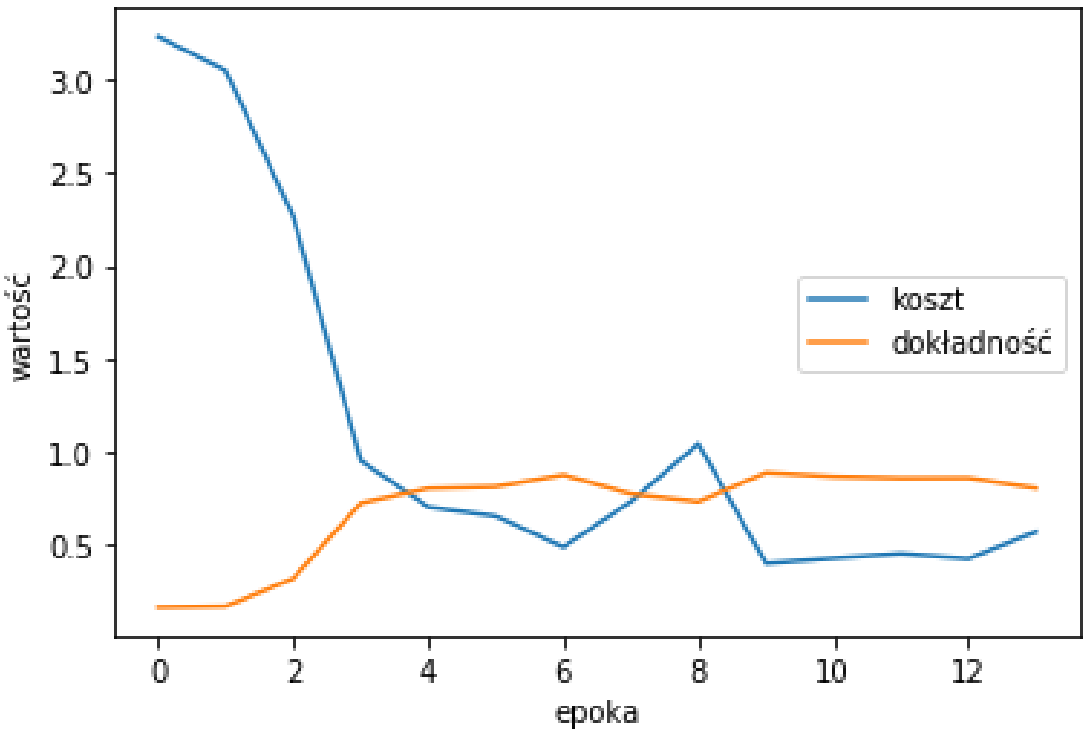
Tablica 4.2: Wyniki sieci przypisującej wszystkim przykładom testowym tę samą etykietę. Zbiór Pavia University. Dane nie zredukowane.

	precyzja	czułość	F1	wsparcie
Asphalt	0.00	0.00	0.00	1717
Meadows	0.47	1.00	0.64	6629
Gravel	0.00	0.00	0.00	576
Trees	0.00	0.00	0.00	975
Painted metal sheets	0.00	0.00	0.00	316
Bare Soil	0.00	0.00	0.00	2048
Bitumen	0.00	0.00	0.00	424
Self-Blocking Bricks	0.00	0.00	0.00	1183
Shadows	0.00	0.00	0.00	378

Tablica 4.3: Wyniki jednej z sieci użytej podczas eksperymentu referencyjnego. Zbiór Salinas. Dane nie zredukowane.

	precyzja	czułość	F1	wsparcie
Brocoli green weeds 1	1.00	0.96	0.98	654
Brocoli green weeds 2	0.91	0.98	0.94	936
Fallow	0.99	0.86	0.92	613
Fallow rough plow	0.91	0.99	0.95	561
Fallow smooth	0.92	0.92	0.92	863
Stubble	0.98	0.87	0.92	1440
Celery	1.00	0.92	0.96	1228
Grapes untrained	0.54	0.89	0.67	3614
Soil vinyard develop	1.00	0.97	0.98	2268
Corn senesced green weeds	0.45	0.80	0.57	374
Lettuce romaine 4wk	0.98	0.88	0.93	373
Lettuce romaine 5wk	0.93	0.96	0.95	525
Lettuce romaine 6wk	0.88	0.99	0.93	377
Lettuce romaine 7wk	0.91	0.87	0.89	381
Vinyard untrained	0.67	0.25	0.36	2916
Vinyard vertical trellis	0.94	0.32	0.47	696

przedstawiono informacje na temat zmian wartości kosztu oraz dokładności wraz z kolejnymi epokami treningu modelu wykazującego największy potencjał (spośród wspomnianych pięciu). Zauważono, że stosunkowo niska początkowa wartość kosztu pozwoliła na jego dalszą redukcję z każdą kolejną epoką. W modelach nie klasyfikujących poprawnie takie zachowanie nie występowało. Jednym ze sposobów na uzyskanie niskiej wartości kosztu podczas treningu jest wykorzystanie normalizacji wsadowej. Pozwala ono również zmniejszyć prawdopodobieństwo utknięcia w minimum lokalnym. Z tego też powodu zdecydowano się na zastosowanie tej techniki przy kolejnej serii eksperymentów.



Rysunek 4.5: Zmiany wartości współczynnika Kappa oraz dokładności podczas procesu uczenia jednego z modeli. Zbiór Salinas. Dane niezredukowane.

4.4.1 Normalizacja wsadowa

W kolejnej serii eksperymentów zastosowano normalizację wsatową dla każdej warstwy konwolucyjnej oraz dla warstwy w pełni połączonej. Z powodów opisanych

Tablica 4.4: Wyniki eksperymentów związanych z wpływem normalizacji wsadowej na skuteczność klasyfikacji.

	Salinas		Pavia University		Indian Pines	
	PCA	bez PCA	PCA	bez PCA	PCA	bez PCA
Dokładność	92.317	85.227	94.615	87.833	47.326	34.775
Kappa	91.305	83.050	92.881	83.344	38.177	25.519
Śr. precyzja	0.947	0.904	0.929	0.859	0.372	0.249
Śr. czułość	0.952	0.863	0.925	0.815	0.317	0.281
Śr. F1	0.948	0.860	0.925	0.820	0.301	0.214
Cz. treningu	36.890	49.629	29.815	27.564	15.955	14.673
Cz. predykcji	1.442	2.642	1.164	1.328	0.249	0.479

w [6] zdecydowano się odstąpić od metod regularyzacji (ponieważ w obecności normalizacji wsadowej zaczyna ona pełnić inną funkcję). Z wyników umieszczonych w tabeli 4.4 wywnioskowano, że normalizacja wsadowa rzeczywiście pozwoliła na rozwiązanie problemu wysokiej wartości kosztu. Żaden z modeli przetwarzających dane niezredukowane nie wykazywał już tendencji do przypisywania wszystkim przykładom tej samej etykiety, a ich średnia dokładność, kappa, precyzja, czułość oraz F1 znacząco wzrosły. Wydłużył się natomiast czas potrzebny do treningu tych modeli, co było spowodowane faktem, że warunek przerwania uczenia następował po większej liczbie epok (sieci w końcu się uczyły). Wyniki modeli uczonych na danych zredukowanych również uległy poprawie (chociaż wzrost był nieduży). Podczas badania wpływu kolejnego hiperparametru, postanowiono zachować zmiany związane z wprowadzeniem normalizacji wsadowej.

4.4.2 Liczba warstw konwolucyjnych

W kolejnej serii eksperymentów zbadano wpływ dodatkowej warstwy konwolucyjnej na wyniki klasyfikacji, które zostały zestawione w tabeli 4.5. Dodana warstwa posiadała taką samą liczbę filtrów jak pozostałe warstwy tego samego typu (to znaczy 4), natomiast rozmiar filtra wynosił $3 \times 3 \times 3$. Dzięki dodatkowej warstwie, modele działające na danych o niezmodyfikowanej liczbie kanałów odnotowały nieco lepsze wyniki (na podstawie Kappa i dokładności), postanowiono

Tablica 4.5: Wyniki eksperymentów związanych z wpływem dodatkowej warstwy konwolucyjnej na skuteczność klasyfikacji.

	Salinas		Pavia University		Indian Pines	
	PCA	bez PCA	PCA	bez PCA	PCA	bez PCA
Dokładność	91.218	85.070	93.004	92.275	36.475	35.030
Kappa	90.076	83.254	90.799	90.032	25.068	27.353
Śr. precyzja	0.934	0.927	0.906	0.916	0.258	0.250
Śr. czułość	0.937	0.895	0.919	0.912	0.236	0.232
Śr. F1	0.931	0.890	0.907	0.905	0.201	0.197
Cz. treningu	42.733	131.421	40.993	62.529	18.359	33.874
Cz. predykcji	1.496	4.197	0.989	2.030	0.239	0.733

więc zachować te zmianę dla modeli tego typu. Największa poprawa obecna była dla zbioru *Pavia University*. Można więc wnioskować, że w celu osiągnięcia jak najlepszych wyników na poszczególnych zbiorach, architektura sieci oraz hiperparametry powinny być dobierane dla nich indywidualnie. Dodatkowa warstwa nie przyniosła jednak korzyści w przypadku danych przetworzonych przez algorytm PCA, dlatego w tym przypadku zmiana ta została odrzucona. W obu wariantach wzrósł średni czas treningu, co jest spowodowane większą liczbą parametrów wewnątrz sieci.

4.4.3 Liczba kanałów w danych zredukowanych

Kolejna seria eksperymentów dotyczyła wyłącznie modeli działających na danych, których wymiarowość została zredukowana za pomocą algorytmu PCA. Zbadano, czy zwiększenie docelowej liczby wymiarów do 20 przełoży się na wzrost wartości poszczególnych metryk. Wyniki dla wszystkich trzech zbiorów danych zestawiono w tabeli 4.6. Ich analiza wykazała korzystny wpływ na wyniki klasyfikacji (na podstawie kappa i dokładności), więc zmianę tę postanowiono zachować.

Tablica 4.6: Wyniki eksperymentów związanych z wpływem docelowej liczby kanałów w danych zredukowanych na skuteczność klasyfikacji.

	Salinas	Pavia University	Indian Pines
Dokładność	93.337	96.527	47.607
Kappa	92.466	95.415	39.273
Śr. precyzja	0.955	0.948	0.375
Śr. czułość	0.951	0.952	0.332
Śr. F1	0.952	0.949	0.302
Cz. treningu	29.691	27.553	20.271
Cz. predykcji	1.376	1.104	0.264

4.4.4 Warstwa redukująca

W kolejnej serii eksperymentów zbadano wpływ dodatkowej warstwy redukującej na wyniki klasyfikacji (tabela 4.7). Została ona umieszczona zaraz po ostatniej warstwie konwolucyjnej. W przypadku modeli działających na danych zredukowanych zanotowano drobny spadek jakości klasyfikacji. Sieci przetwarzające dane niezredukowane okazały się nieco skuteczniejsze dla zbiorów *Salinas* oraz *Pavia University*, natomiast spadek jakości klasyfikacji nastąpił dla zbioru *Indian Pines*. Dla tego typu modeli, postanowiono zachować tę dodatkową warstwę. Był to kolejny przykład sytuacji, w której zmiana wprowadzona w modelu miała różny wpływ na wyniki w zależności od zbioru danych.

4.4.5 Wymiary przykładów uczących

Celem kolejnej serii eksperymentów było zbadanie wpływu rozmiaru przykładów podawanych na wejścia sieci na wyniki klasyfikacji. Testowany wariant zakładał powiększenie wysokości i szerokości przykładów o dwa piksele (z 3×3 na 5×5) przy jednoczesnym zachowaniu obecnej głębokości. Wyniki zestawiono w tabeli 4.8. Średni czas potrzebny na trening modeli uczonych na danych niezredukowanych wydłużył się ponad dwukrotnie podnosząc jednak nieznacznie dokładność i współczynnik Kappa. Z tego względu zdecydowano się na zachowanie tej zmiany dla tego typu sieci. W przypadku modeli przetwarzających dane zredukowane poprawa nie

Tablica 4.7: Wyniki eksperymentów związanych z wpływem dodatkowej warstwy redukującej na skuteczność klasyfikacji.

	Salinas		Pavia University		Indian Pines	
	PCA	bez PCA	PCA	bez PCA	PCA	bez PCA
Dokładność	91.577	86.622	93.522	93.838	48.082	31.287
Kappa	90.487	84.776	91.489	91.929	38.005	22.016
Śr. precyzja	0.945	0.937	0.924	0.946	0.360	0.193
Śr. czułość	0.944	0.904	0.932	0.941	0.302	0.215
Śr. F1	0.940	0.904	0.926	0.940	0.294	0.174
Cz. treningu	29.934	125.708	42.886	82.786	15.485	27.893
Cz. predykcji	1.436	4.337	1.141	2.253	0.272	0.747

nastąpiła.

4.4.6 Liczebność partii

W kolejnej serii eksperymentów zbadano wpływ zmniejszenia liczebności partii (ze 128 do 64) na wyniki klasyfikacji (tabela 4.9). W przypadku modeli działających na danych niezredukowanych zaobserwowano poprawę o parę punktów procentowych względem poprzedniego rozwiązania (dokładność i Kappa). Postanowiono więc zachować tę zmianę. W przypadku sieci przetwarzających dane zredukowane nie odnotowano korzystnego wpływu zmniejszenia liczebności partii.

4.4.7 Liczba filtrów w warstwach konwolucyjncyh

Ostatnim z badanych hiperparametrów była liczba filtrów w ramach warstw konwolucyjnych. Wyniki klasyfikacji uzyskane po zwiększeniu liczby filtrów w każdej takiej warstwie z 4 do 8 zostały zapisane w tabeli 4.10. W przypadku modeli działających na danych niezredukowanych zauważono poprawę w klasyfikacji danych ze zbioru *Pavia University* (na podstawie dokładności i Kappa) wraz z jednoczesnym jej spadkiem w pozostałych zbiorach. Ponadto średni czas treningu wzrósł jeszcze bardziej.

Tablica 4.8: Wyniki eksperymentów związanych z wpływem rozmiarów przykładów na skuteczność klasyfikacji.

	Salinas		Pavia University		Indian Pines	
	PCA	bez PCA	PCA	bez PCA	PCA	bez PCA
Dokładność	93.109	87.130	96.415	93.794	47.619	43.820
Kappa	92.214	85.127	95.268	91.810	37.863	30.698
Śr. precyzja	0.954	0.926	0.942	0.932	0.338	0.267
Śr. czułość	0.950	0.907	0.944	0.914	0.277	0.255
Śr. F1	0.949	0.892	0.941	0.916	0.281	0.235
Cz. treningu	26.623	472.714	28.229	147.323	12.392	52.654
Cz. predykcji	1.244	5.726	0.884	2.506	0.215	1.042

Tablica 4.9: Wyniki eksperymentów związanych z wpływem liczebności partii na skuteczność klasyfikacji.

	Salinas		Pavia University		Indian Pines	
	PCA	bez PCA	PCA	bez PCA	PCA	bez PCA
Dokładność	93.275	87.982	96.210	94.127	46.801	48.257
Kappa	92.395	86.610	95.014	92.203	37.580	38.480
Śr. precyzja	0.954	0.892	0.944	0.943	0.383	0.341
Śr. czułość	0.951	0.915	0.945	0.957	0.328	0.288
Śr. F1	0.949	0.881	0.943	0.947	0.309	0.273
Cz. treningu	35.165	319.304	54.592	222.699	18.813	74.501
Cz. predykcji	1.455	6.350	1.209	2.668	0.273	1.069

Tablica 4.10: Wyniki eksperymentów związanych z wpływem liczby filtrów w warstwach konwolucyjnych na skuteczność klasyfikacji.

	Salinas		Pavia University		Indian Pines	
	PCA	bez PCA	PCA	bez PCA	PCA	bez PCA
Dokładność	92.733	83.879	96.873	96.170	45.293	45.718
Kappa	91.788	81.484	95.870	94.877	36.389	34.664
Śr. precyzja	0.951	0.924	0.961	0.956	0.392	0.304
Śr. czułość	0.954	0.866	0.955	0.954	0.350	0.301
Śr. F1	0.950	0.868	0.958	0.954	0.328	0.259
Cz. treningu	28.205	488.417	30.738	268.638	9.405	105.469
Cz. predykcji	1.424	8.054	1.053	3.431	0.238	1.437

4.4.8 Porównanie wyników z dostępnymi w literaturze

Ze względu na wykorzystaną w przeprowadzonych badaniach metodę podziału danych na zbiory treningowe, testowe oraz walidacyjne, bezpośrednie porównanie wyników z innymi dostępnymi w literaturze pracami (które wykorzystują standardowe metody podziału) uznano za mało miarodajne. Zamiast tego postanowiono wybrać najskuteczniejszy spośród przetestowanych modeli, wytrenować go w sposób klasyczny i dopiero wtedy porównać wartości poszczególnych metryk.

W [31] autorzy również wykorzystali zbiory *Indian Pines*, *Salinas* oraz *Pavia University* do oceny jakości klasyfikacji konwolucyjnej sieci neuronowej. Dzięki połączeniu trójwymiarowych warstw konwolucyjnych z jedną dwuwymiarową byli w stanie klasyfikować piksele znajdujące w zbiorze testowym z dokładnością przewyższającą 99,75%. Jest to jeden z lepszych rezultatów opisanych w literaturze. Zbiór treningowy stanowił 30% wszystkich przykładów dostępnych w danym zbiorze, natomiast zbiór treningowy 70%. Zrezygnowano z osobnego zbioru walidowacyjnego. Zaproponowanej przez siebie architekturę autorzy nadali nazwę *HybridSN*.

Z przeprowadzonych w niniejszej pracy badań najskuteczniejszy modele otrzymano w rezultacie eksperymentu czwartego. Operował on na danych z wymiarem spektralnym zredukowanym do dwudziestu pasm. Od architektury wyjściowej (opisanej na początku tego podrozdziału) odróżniał go brak regularyzacji oraz zastosowana w ramach każdej warstwy normalizacja wsadowa.

Tablica 4.11: Porównanie najskuteczniejszego modelu otrzymanego w ramach niniejszej pracy z model opisanym w literaturze.

	Salinas	
	HybridSN	wbrany model
Dokładność	100.00	98.71
Kappa	100.00	98.57

Tablica 4.12: Porównanie najskuteczniejszego modelu otrzymanego w ramach niniejszej pracy z model opisanym w literaturze.

	Pavia University	
	HybridSN	wbrany model
Dokładność	99.98	99.14
Kappa	99.98	98.86

W przytoczonej wcześniej pracy przedstawiono wyłącznie informacje dotyczące obliczonej dokładności oraz współczynnika Kappa, dlatego też tylko te dwie metryki zostały ze sobą porównane. Zestawienie znajduje się w tabelach 4.11, 4.12 oraz 4.13. Skuteczność klasyfikacji zaproponowanego modelu nie była tak wysoka jak w przypadku architektury *HybridSN*, mimo to dla zbiorów *Salinas* oraz *Pavia University* różnica nie była duża. Dysproporcje w skuteczności klasyfikacji były bardziej widoczne w wynikach otrzymanych na zbiorze *Indian Pines*. Dla modelu wybranego podczas badań przeprowadzonych w ramach niniejszej pracy zanotowano wpływ zmiany algorytmu podziału danych na wartości dokładności oraz współczynnika Kappa. Pozwala to przypuszczać, że sieć *HybridSN* mogłaby się okazać mniej skuteczna podczas wykorzystania jej w najbardziej oczywistych przypadkach użycia.

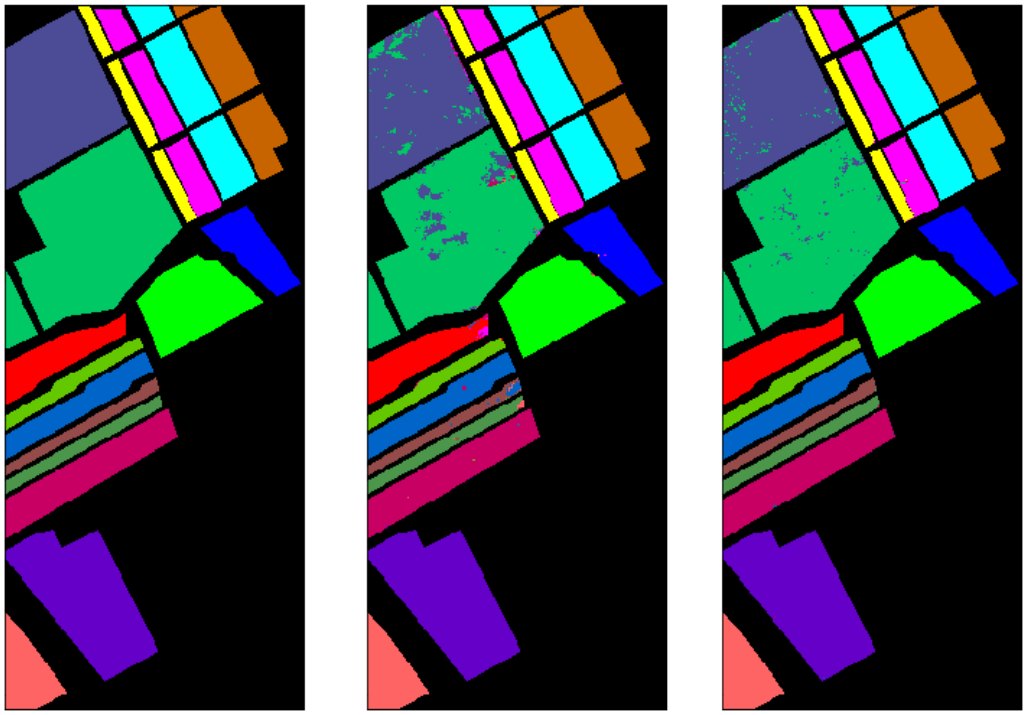
4.4.9 Ostateczne wyniki segmentacji

Poza porównaniem współczynników i metryk zdecydowano się również przedstawić możliwości zaproponowanego modelu do rozwiązania problemu segmentacji obrazów hiperspektralnych w bardziej przystępny sposób. W tym celu dokonano

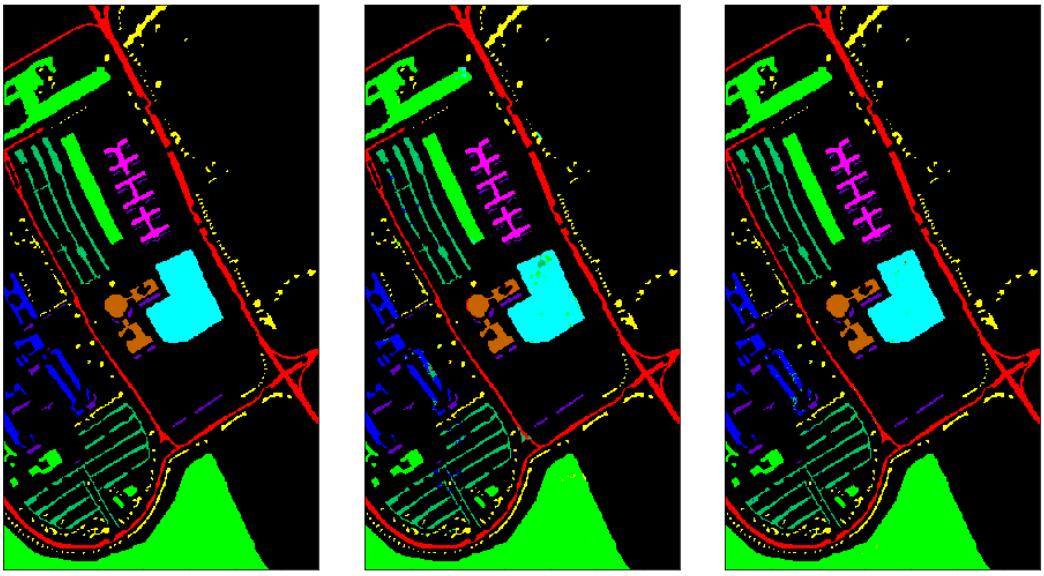
Tablica 4.13: Porównanie najskuteczniejszego modelu otrzymanego w ramach niniejszej pracy z model opisanym w literaturze.

	Indian Pines	
	HybridSN	wbrany model
Dokładność	99.75	84.48
Kappa	99.71	82.31

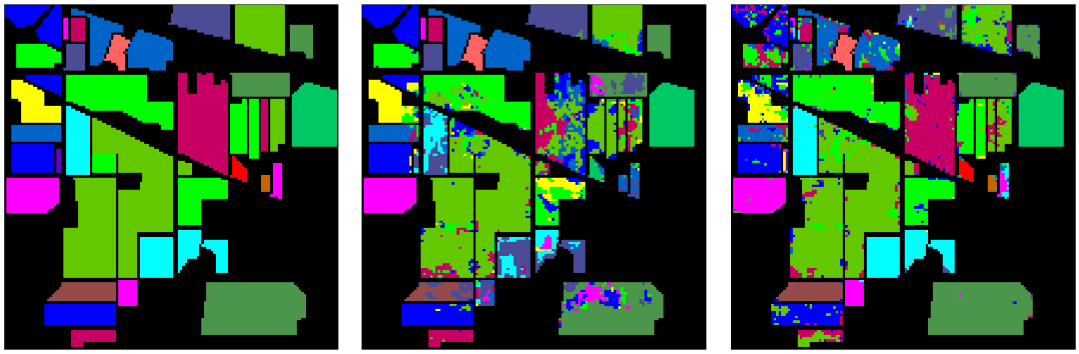
klasyfikacji każdego piksela (nie będącego tłem) w każdym z przetwarzanych zbiorów danych. Wyniki zestawiono z obszarami oznaczonymi przez eksperta. Poza modelem uczonym i testowanym na danych podzielonych za pomocą algorytmu wykorzystującego pod-obszary, uwzględniono również ten wytrenowany podczas porównania z siecią *HybridSM* (wykorzystujący sposób standardowy). Wyniki umieszczono na rysunkach 4.6, 4.7 oraz 4.8.



Rysunek 4.6: Wyniki segmentacji z wykorzystaniem zaproponowanego modelu. Obraz środkowy został uzyskany w wyniku działania modelu uczonego na danych pochodzących z wydzielonych pod-obszarów. Obraz po prawej stornie został uzyskany w wyniku działania modelu uczonego na przykładach dobieranych w sposób losowy. Po lewej obszar oznaczony przez eksperta. Zbiór Salinas.



Rysunek 4.7: Wyniki segmentacji z wykorzystaniem zaproponowanego modelu. Obraz środkowy został uzyskany w wyniku działania modelu uczonego na danych pochodzących z wydzielonych pod-obszarów. Obraz po prawej stornie został uzyskany w wyniku działania modelu uczonego na przykładach dobieranych w sposób losowy. Po lewej obszar oznaczony przez eksperta. Zbiór Pavia University.



Rysunek 4.8: Wyniki segmentacji z wykorzystaniem zaproponowanego modelu. Obraz środkowy został uzyskany w wyniku działania modelu uczonego na danych pochodzących z wydzielonych pod-obszarów. Obraz po prawej stornie został uzyskany w wyniku działania modelu uczonego na przykładach dobieranych w sposób losowy. Po lewej obszar oznaczony przez eksperta. Zbiór Indian Pines.

Rozdział 5

Podsumowanie

Przeprowadzone w ramach niniejszej pracy badania pozwalają stwierdzić, że wstępna redukcja wymiarowości danych hiperspektrlnych ma znaczący wpływ na skuteczność klasyfikacji. Chociaż udało się uzyskać model, który był w stanie w zadowalającym stopniu (w odniesieniu do innych rozwiązań znanych z literatury) analizować zarówno dane surowe, jak i wstępnie przetworzone, to nie był on tak skuteczny jak rozwiązania wyspecjalizowane w konkretnym typie danych. Zaobserwowano również, że dobranie odpowiedniej architektury sieci oraz jej hiperparametrów specjalnie pod jeden zestaw danych może prowadzić do otrzymania jeszcze lepszych wyników dla danego zbioru. Model taki staje się jednak mniej ogólny i może skutkować spadkiem jakości klasyfikacji podczas przetwarzania innych zdjęć hiperspektralnych. Mimo że wykorzystany algorytm podziału danych wykorzystujący koncept pod-obszarów skutkuje gorszymi wynikami oceny modelu, to zdaje się lepiej odwzorowywać rzeczywiste przypadki użycia i jego zastosowanie powinno być rozważone podczas badań nad rozwiązaniami problemu segmentacji obrazów hiperspektralnych.

Uczenie głębokie jest cały czas rozwijającą się dziedziną nauki. Dostępne metody i algorytmy są cały czas ulepszane, bądź wypierane przez bardziej skuteczne rozwiązania. Nie zawsze jest możliwe określenie ze stuprocentową pewnością, jakie modyfikacje powinny zostać wdrożone w modelu, aby poprawić skuteczność jego klasyfikacji. Działanie niektórych metod jest potwierdzone wyłącznie eksperymentalnie, pewne zastosowane techniki mogą mieć nieoczywisty wpływ na

działanie innych (np. jednoczesne stosowanie normalizacji wsadowej oraz regularyzacji). Wszystkie te czynniki wraz z dużą liczbą hipermarametrów występującą w strukturach, jakimi są knowlucyjne sieci neuronowe, powodują, że znalezienie optymalnego rozwiązania stanowi prawdziwe wyzwanie. Z drugiej strony w wielu zastosowaniach oczekuje się od rozwiązania, by było „wystarczająco dobre", a wiele stopni swobody pozwala na odpowiednie dostosowanie modelu w zależności, czy priorytetem jest czas treningu, ogólna dokładność klasyfikacji, czy może precyzja uzyskana dla jednej z klas.

Temat segmentacji obrazów hiperspektralnych przy wykorzystaniu konwolucyjnych sieci neuronowych jest bardzo rozległy i niniejsza praca go nie wyczerpuje. Jednym z potencjalnych kierunków dalszego jej rozwoju jest kontynuacja poszukiwań optymalnych hiperparametrów modelu rozwiązującego problem klasyfikacji poszczególnych pikseli do wielu klas. Podczas przeprowadzonych badań rozpatrzono zaledwie kilka z wszystkich możliwych konfiguracji.

Algorytm dzielący dane na zbiory służące do treningu, walidacji oraz testowania modelu również pozostawia sporo miejsca na wprowadzenie usprawnień. Dla przykładu, liczba niewykorzystanych pikseli może zostać ograniczona poprzez odpowiednie łączenie sąsiadujących pod-obszarów. W niektórych okolicznościach pomocną mogłaby się okazać modyfikacja, która gwarantowałaby, że w zbiorze treningowym znajdą się przykłady reprezentujące każdą z klas. Wykorzystany jako parametr procent pokrycia przetwarzanego obrazu pod-obszarami nie zawsze skutkuje podobnym stosunkiem liczebności poszczególnych zbiorów. Możliwość podania dokładnych proporcji podziału pozwoliłaby na łatwiejsze porównywanie zarówno przebiegu uczenia, jak i wyników modelu.

W literaturze zaproponowano wiele różnych rozwiązań problemu segmentacji obrazów hiperspektralnych. Duża część z nich bierze pod uwagę sąsiedztwo klasyfikowanego piksela. Wykorzystanie do treningu oraz testów tych modeli algorytmu podziału danych bazującego na pod-obszarach pozwoliłoby na zbadanie ich skuteczności w warunkach bardziej zbliżonych do rzeczywistych zastosowań.

Wykorzystane zbioru danych stanowią jedynie niewielką część wszystkich dostępnych zdjęć hiperspektralnych wykonanych z pokładu samolotu bądź drona. Użycie innych danych mogłoby pomóc w odkryciu potencjalnych słabości zaproponowanego modelu oraz ich wyeliminowaniu, co skutkowałoby jeszcze wyższym

poziomem klasyfikacji przykładów.

Bibliografia

[1] Saad Albawi, Tareq Abed Mohammed, Saad ALZAWI. Understanding of a convolutional neural network. 08 2017.

[2] Bai, Kunlun. A comprehensive introduction to different types of convolutions in deep learning. URL: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215.

[3] Amina Ben Hamida, Alexandre Benoit, Patrick Lambert, Chokri Ben Amar. 3-d deep learning approach for remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, PP:1–15, 04 2018.

[4] Brownlee, Jason. How to choose loss functions when training deep learning neural networks. URL: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/.

[5] Yushi Chen, Hanlu Jiang, Chunyang Li, Xiuping Jia, Pedram Ghamisi. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 54:1–20, 07 2016.

[6] David Wu. L2 regularization and batch norm. URL: https://blog.janestreet.com/l2-regularization-and-batch-norm/.

[7] E. Dogo, O. Afolabi, N Nwulu, Bhekisipho Twala, Clinton Aigbavboa. A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. 12 2018.

[8] Drzewiecki, Wojciech. Teledetekcja. URL: `http://home.agh.edu.pl/~galia/students/NS/teledetekcja_w_skrocie.pdf`.

[9] Qian Du, Chein-I Chang. A linear constrained distance-based discriminant analysis for hyperspectral image classification. *Pattern Recognition*, 34:361–373, 02 2001.

[10] Crescenzio Gallo. *Artificial Neural Networks: tutorial*. 01 2015.

[11] Tianhan Gao, Lei Jiang, Xibao Wang. *Recommendation System Based on Deep Learning*, strony 535–543. 01 2020.

[12] Grochowski, Marek. Głębokie uczenie. URL: `http://www.is.umk.pl/~grochu/wiki/lib/exe/fetch.php?media=zajecia:nn_2018_1:nn-wyklad.pdf`.

[13] Mingyi He, Bo Li, Huahui Chen. Multi-scale 3d deep convolutional neural network for hyperspectral image classification. strony 3904–3908, 09 2017.

[14] Wei Hu, Yangyu Huang, Li Wei, Fan Zhang, Hengchao Li. Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors*, 2015:1–12, 07 2015.

[15] Xiu Jin, Lu Jie, Shuai Wang, Hai Qi, Shao Li. Classifying wheat hyperspectral pixels of healthy heads and fusarium head blight disease using a deep neural network in the wild field. *Remote Sensing*, 10:395, 03 2018.

[16] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[17] Jordan, Jeremy. An overview of semantic image segmentation. URL: `https://www.jeremyjordan.me/semantic-segmentation/`.

[18] Am Khan, Ravi Srisha. Image segmentation methods: A comparative study. *International Journal of Soft Computing and Engineering*, strony 2231–2307, 09 2013.

[19] Michał Koziarski, Boguslaw Cyganek. Image recognition with deep neural networks in presence of noise – dealing with and taking advantage of distortions. *Integrated Computer-Aided Engineering*, 24:1–13, 08 2017.

[20] Krawiec, Krzysztof. Segmentacja obrazu. URL: `https://www.cs.put.poznan.pl/kkrawiec/wiki/uploads/Zajecia/po5.pdf`.

[21] Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[22] G V S Kumar. Review on image segmentation techniques. *International Journal of Scientific Research Engineering Technology (IJSRET)*, 3:992–997, 09 2014.

[23] Jun Li, Jose Bioucas-Dias, Antonio Plaza. Semisupervised hyperspectral image segmentation using multinomial logistic regression with active learning. *Geoscience and Remote Sensing, IEEE Transactions on*, 48:4085 – 4098, 12 2010.

[24] Konstantinos Makantasis, Konstantinos Karantzalos, Anastasios Doulamis, Nikolaos Doulamis. Deep supervised learning for hyperspectral data classification through convolutional neural networks. strony 4959–4962, 01 2015.

[25] Andreas C. Müller, Sarah Guido. *Introduction to Machine Learning with Python*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2016.

[26] Ali Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, PP:19143–19165, 02 2019.

[27] Samuel Ortega, Himar Fabelo, Dimitris Iakovidis, Anastasios Koulaouzidis, Gustavo Marrero Callico. Use of hyperspectral/multispectral imaging in gastroenterology. shedding some–different–light into the dark. *Journal of Clinical Medicine*, 8:36, 01 2019.

[28] Antonio Plaza, Pablo Martinez, Rosa Perez, Javier Plaza. A new approach to mixed pixel classification of hyperspectral imagery based on extended morphological profiles. *Pattern Recognition*, 37:1097–1116, 06 2004.

[29] Yuntao Qian, Minchao Ye, Jun Zhou. Hyperspectral image classification based on structured sparse logistic regression and three-dimensional wavelet texture features. *IEEE Transactions on Geoscience and Remote Sensing*, 51:2276–2291, 04 2013.

[30] Raj, Judy T. A beginner's guide to dimensionality reduction in machine learning. URL: `https://towardsdatascience.com/dimensionality-reduction-for-machine-learning-80a46c2ebb7e`.

[31] Swalpa Roy, Gopal Krishna, Shiv Ram Dubey, Bidyut Chaudhuri. Hybridsn: Exploring 3d-2d cnn feature hierarchy for hyperspectral image classification. 02 2019.

[32] Shaeke Salman, Xiuwen Liu. Overfitting mechanism and avoidance in deep neural networks. 01 2019.

[33] Luis Samaniego, András Bárdossy, Karsten Schulz. Supervised classification of remotely sensed imagery using a modified k-nn technique. *Geoscience and Remote Sensing, IEEE Transactions on*, 46:2112 – 2125, 08 2008.

[34] Linlin Shen, Sen Jia. Three-dimensional gabor wavelets for pixel-based hyperspectral imagery classification. *IEEE T. Geoscience and Remote Sensing*, 49:5039–5046, 12 2011.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 06 2014.

[36] Sullivan, Josephine. Handcrafted image features. URL: `http://www.csc.kth.se/utbildning/kth/kurser/DD2427/bik14/DownloadMaterial/Lectures/Lecture2.pdf`.

[37] Torres, Jordi. Learning process of a neural network. URL: https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7.

[38] Pramod K. Varshney, Manoj K. Arora. *Advanced Image Processing Techniques for Remotely Sensed Hyperspectral Data*. Springer-Verlag, Berlin Heidelberg New York, 2004.

[39] Alberto Villa, Jocelyn Chanussot, Jon Benediktsson, Christian Jutten. Spectral unmixing for the classification of hyperspectral images at a finer spatial resolution. *IEEE Journal of Selected Topics in Signal Processing*, 5, 07 2011.

[40] Sowmya Vishvanathan, Soman Kp, M. Hassaballah. *Hyperspectral Image: Fundamentals and Advances*, strony 401–424. 01 2019.

[41] Guorong Wu, Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19, 03 2017.

[42] Na Wu, Chu Zhang, Xiulin Bai, Xiaoyue Du, Yong He. Discrimination of chrysanthemum varieties using hyperspectral imaging combined with a deep convolutional neural network. *Molecules*, 23:2831, 10 2018.

[43] Zongben Xu, Hai Zhang, Yao Wang, Xiangyu Chang, Yong Liang. L 1/2 regularization. *Science China Information Sciences*, 53, 06 2010.

[44] Tom Young, Devamanyu Hazarika, Soujanya Poria, Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13:55–75, 08 2018.

[45] Wenzhi Zhao, Shihong Du. Spectral–spatial feature extraction for hyperspectral image classification: A dimension reduction and deep learning approach. *IEEE Transactions on Geoscience and Remote Sensing*, 54:4544–4554, 04 2016.

Dodatki

Zawartość dołączonej płyty

- praca w formacie `pdf`,
- źródła programu,
- zbiory danych użyte w eksperymentach.

Spis rysunków

Spis tablic