





**Politechnika  
Śląska**

## **PRACA MAGISTERSKA**

„Zastosowanie konwolucyjnych sieci neuronowych w klasyfikacji obrazów dla urządzeń mobilnych”

**Marzena Halama**

**294024**

**Kierunek: matematyka**

**Specjalność: matematyka teoretyczna**

**PROMOTOR:**

**dr hab. inż. Joanna Domańska**

**PROMOTOR POMOCNICZY:**

**prof. dr hab. Mykola Bratiichuk**

**WYDZIAŁ MATEMATYKI STOSOWANEJ**

**GLIWICE 2024**

*Chciałabym podziękować dr inż. Konradowi Połys za wsparcie  
i obecność w tym ważnym czasie.*

*Chciałabym również wyrazić moją głęboką wdzięczność  
mojej promotorce, dr hab. inż. Joannie Domańskiej, za jej cierpliwość,  
zaangażowanie i nieustające inspiracje. Jestem niezmiernie wdzięczna  
za możliwość nauki i rozwoju pod jej kierownictwem.*

**Tytuł pracy:**

„Zastosowanie konwolucyjnych sieci neuronowych w klasyfikacji obrazów dla urządzeń mobilnych”

**Streszczenie:**

Niniejsza praca magisterska ściśle związana jest z dziedziną sztucznej inteligencji i uczenia maszynowego. W pracy szczególna uwaga została poświęcona zastosowaniu konwolucyjnych sieci neuronowych na urządzeniach mobilnych. Często, ze względu na ograniczenia w mocy obliczeniowej sprzętu, niezbędne jest stosowanie specjalnych technik, które umożliwiają wykorzystanie oraz efektywne działanie sieci konwolucyjnych. Jedną z takich metod jest wykorzystanie ekstraktorów cech modeli pierwotnie wytrenowanych na wielkoskalowych bazach danych, takich jak ImageNet, jako podstawę dla innych zadań. W ramach tych metod stosuje się między innymi techniki takie jak transfer wiedzy oraz głębokie dopasowywanie wzorców. Celem tych technik jest np. umożliwienie urządzeniom ograniczonym pod względem zasobów obliczeniowych, takich jak urządzenia mobilne, wykonanie zaawansowanych zadań m.in. rozpoznawanie wzorców i klasyfikacja obrazów, z zachowaniem odpowiedniej szybkości i dokładności. Wyniki przedstawione w pracy pokazują, że metody te dają możliwość działania aplikacji mobilnych wykorzystujących sztuczną inteligencję, otwierając nowe możliwości dla rozwoju intelligentnych systemów na platformach mobilnych.

**Słowa kluczowe:**

Sztuczna Inteligencja, Uczenie Maszynowe, Transfer Wiedzy, Głębokie Dopasowywanie Wzorów, Konwolucyjne Sieci Neuronowe, Rozpoznawanie Obrazów.

**Thesis title:**

”Application of convolutional neural networks in image recognition for mobile devices”

**Abstract:**

This thesis is closely related to the domain of artificial intelligence (AI) and machine learning (ML). There has been a particular focus on the application of convolutional neural networks (CNNs) on mobile devices. Due to the computational limitations of the hardware, it is often necessary to implement specialized techniques to utilize and efficiently perform CNNs. An example of such techniques involves using feature extractors from models originally trained on large databases, such as ImageNet, as a ground for other tasks. Potential techniques in this area include the transfer learning and deep template matching. The aim of these techniques is to enable devices with limited resources, such as mobile devices, to perform advanced image classification tasks while keeping reasonable speed and accuracy. The results presented in the thesis show that these methods allow the development of mobile applications using AI, thus unlocking new opportunities for the development of intelligent systems on mobile platforms.

**Keywords:**

Artificial Intelligence, Machine Learning, Transfer Learning, Deep Template Matching, Convolutional Neural Networks, Image Recognition.

**Klasyfikacja tematyczna (2010 Mathematics Subject Classification):**

68T05 Learning and adaptive systems in artificial intelligence

68T45 Machine vision and scene understanding

90C90 Applications of mathematical programming

68T20 Problem solving in the context of artificial intelligence (heuristics, search strategies, etc.)

# **Spis treści**

<b>Wstęp</b>	<b>7</b>
<b>1. Zagadnienia teoretyczne</b>	<b>13</b>
1.1. Sztuczna inteligencja . . . . .	13
1.2. Uczenie maszynowe . . . . .	15
1.3. Głębokie sieci neuronowe . . . . .	17
1.4. Matematyczne podstawy . . . . .	21
1.5. Przetwarzanie obrazów z wykorzystaniem sieci neuronowych . . . . .	30
1.5.1. Przypisywanie etykiet . . . . .	33
1.5.2. Funkcja całkowitego pobudzenia układu . . . . .	33
1.6. Trenowanie i testowanie modelu . . . . .	36
<b>2. Zastosowanie konwolucyjnych sieci neuronowych w klasyfikacji obrazów na urządzeniach mobilnych</b>	<b>51</b>
2.1. Bazy danych . . . . .	52
2.2. Charakterystyka modeli wstępnie wytrenowanych z Keras . . . . .	55
2.3. Opis wybranych techniki rozpoznawania obrazów . . . . .	57
2.3.1. Głębokie dopasowywanie wzorców . . . . .	57
2.3.2. Transfer wiedzy . . . . .	59
<b>3. Wyniki</b>	<b>63</b>
3.1. Transfer wiedzy . . . . .	63
3.2. Głębokie dopasowywanie wzorców . . . . .	72
<b>4. Podsumowanie</b>	<b>91</b>
<b>Bibliografia</b>	<b>93</b>



# Wstęp

Sztuczna inteligencja (ang. Artificial Intelligence, AI) to rodzaj technologii, która nadaje maszynie pewną inteligencję, dzięki czemu maszyna ma taką samą zdolność do rozwiązywania zadań jak ludzie [58]. Sztuczna inteligencja jest bardzo szeroką i interdyscyplinarną dziedziną nauki, ze względu na różnorodność technik i podejść do tworzenia systemów komputerowych [40, 6]. Jedną z możliwości wykorzystywania AI jest domena uczenia maszynowego (ang. Machine Learning, ML). Fundamentem działania ML jest rozwijanie umiejętności poznawczych algorytmów uczących się automatycznie poprzez doświadczenie [8]. Ten aspekt programowania pozwala na wykorzystywanie sieci neuronowych w celu identyfikowania wzorców i zależności w danych, które pozwalają na skuteczne podejmowanie decyzji i prognozowanie przyszłych zdarzeń. Zatem uczenie maszynowe pozwala na dokładną analizę i wykorzystaniem danych, szczególnie w kwestii odkrycia ukrytych reguł, zależności, anomalii i ułatwia zrozumienie oraz interpretację danych [14, 52]. Niniejsza praca magisterska dotyczy zadania klasyfikacji obrazów, który jest jednym z istotnych obszarów AI. Ponieważ identyfikacja obiektów jest niezwykle ważnym zagadnieniem w dziedzinie wizji komputerowej, a jej zastosowania są rozległe - od medycyny po przemysł [3], to celem tej pracy jest przeprowadzenie starannej oceny technik przetwarzania wizualnego, opartych na konwolucyjnych sieciach neuronowych (ang. Convolutional Neural Networks, CNN) [56, 4], które mogą być efektywnie stosowane na urządzeniach mobilnych.

W pracy szczególna uwaga została skierowana na takie techniki rozpoznawania obrazów, które umożliwiają skuteczną klasyfikację obiektów przy ograniczonej dostępności danych, co znacząco redukuje zapotrzebowanie na obszerne zasoby obliczeniowe. Do metod tych zalicza się podejścia takie jak **transfer wiedzy** (ang. Transfer Learning, TL) oraz **głębokie dopasowywanie wzorców** (ang. Deep Matching

Templates, DMT). Wykorzystanie tych technik jest istotne w kontekście koncepcji uczenia się metodą niewielu strzałów (ang. few-shot learning) [49, 41, 36]. Minimizacja ilości danych w zbiorze treningowym odgrywa kluczową rolę w implementacji rozwiązań uczenia maszynowego na urządzeniach mobilnych, które często dysponują ograniczoną mocą obliczeniową. Jest to również istotną kwestią w scenariuszach wymagających szybkiego i dokładnego rozpoznawania obrazów przy ograniczonej ilości danych, na przykład w diagnostyce medycznej [34, 16] czy analizie obrazów hiper-spektralnych czy satelitarnych [29, 22, 2].

Praca ta została podzielona na trzy główne rozdziały. Pierwszy z nich stanowi wprowadzenie w tematykę sztucznej inteligencji, która jest głównym obszarem tematycznym niniejszej pracy. Zawarte w tym rozdziale opisy zagadnień stanowią bazę do dalszych rozważań i głębszego zrozumienia działania metod uczenia maszynowego. Rozdział następny dotyczy zastosowania konwolucyjnych sieci neuronowych w klasyfikacji obrazów dla urządzeń mobilnych, co jest tematem niniejszej pracy. Rozdział ostatni zawiera wyniki uzyskane dla testów badanych w tej pracy technik transferu wiedzy i głębokiego dopasowywania wzorców. Poniżej opisano bardziej szczegółowo zawartość poszczególnych rozdziałów. W podrozdziałach od 1.1 do 1.3 scharakteryzowane zostały szczegółowo kluczowe koncepcje, takie jak sztuczna inteligencja, uczenie maszynowe i uczenie głębokie, podkreślając różnice między tymi obszarami. Ta część pracy rozpoczyna się od omówienia pierwotnego podejścia do sztucznej inteligencji, znanego jako symboliczna sztuczna inteligencja, która opierała się na wierze ekspertów w to, że osiągnięcie sztucznej inteligencji na poziomie ludzkim jest możliwe poprzez utworzenie przez programistów wystarczająco dużego zbioru reguł służących do manipulowania zbiorami danych. Tak pojmowana sztuczna inteligencja była skuteczna w rozwiązywaniu dobrze zdefiniowanych logicznych problemów typu gra w szachy, natomiast okazała się bezużyteczna w przypadku bardziej złożonych, niejasnych logicznie problemów typu np. klasyfikacja obrazów. To stało się bodźcem do zmiany i odejścia od tradycyjnych paradigmatów programowania, w kierunku coraz intensywniej rozwijającej się dziedziny uczenia maszynowego. Dalej

znajduje się opis trzech głównych typów uczenia maszynowego: nadzorowane, nie-nadzorowane oraz uczenie przez wzmacnianie. Zostały również podane konkretne przykłady zastosowań każdego z tych typów, co ułatwia zrozumienie ich praktycznej użyteczności. Podrozdział 1.3 poświęcony jest zagadnieniu głębokiego uczenia się. Przedstawiono w nim budowę i zasady działania perceptronu, który jest elementarną jednostką sieci neuronowej. Zaprezentowano strukturę sieci neuronowej wraz z omówieniem jej kluczowych komponentów i mechanizmów działania. Podrozdział zatytułowany **matematyczne podstawy** precyzyjnie przedstawia matematyczne aspekty związane z konstrukcją i działaniem CNN, które szeroko wykorzystywane są w technikach rozpoznawania obrazów ukazanych w tej pracy. W tej części opisane zostały elementarne koncepcje i metodologia niezbędna do zrozumienia mechanizmów funkcjonowania tych sieci. Podrozdział 1.6 został podzielony na dwie główne części: etap trenowania oraz etap testowania modelu. W pierwszej części opisano, czym są epoki w kontekście trenowania, wraz ze znaczeniem wyboru odpowiedniej liczby iteracji w kontekście jej wpływu na efektywności i dokładności modelu. Dodatkowo opisany został problem niedopasowania i nadmiernego dopasowania modelu. Następnie omówiono różne funkcje aktywacji używane w sieciach neuronowych i ich wpływ na zdolność modelu do nauki i generalizacji. Na zakończenie zaprezentowano różne algorytmy optymalizacyjne stosowane w celu minimalizacji funkcji straty. W drugiej części zdefiniowano macierz podobieństwa oraz metryki służące do oceny stopnia podobieństwa. Na koniec przedstawiono różne metody oceny wydajności modelu klasyfikacyjnego, w tym takie miary jak dokładność, precyzja, czułość i specyficzność, oraz ich znaczenie w kontekście testowania i walidacji modelu. Ta część pracy pozwala zdobyć wiedzę i umiejętności związanych z implementacją, trenowaniem i testowaniem modeli uczenia maszynowego.

Rozdział drugi opisuje kluczowy aspekt tej pracy, którym jest klasyfikacja obrazów przy użyciu CNN. Omówione w nim metodologie stanowi fundament dla badań i wniosków zawartych w tej pracy, co czyni ten rozdział niezwykle istotnym. Rozdział trzeci, poświęcony problemowi klasyfikacji obrazów, przedstawia opis technik

klasyfikacji obrazów takich jak transfer wiedzy i głębokie dopasowywanie wzorców. W szczególności, punkt zwrotny w badaniach nad sztucznymi sieciami neuronowymi, szkolonymi na podstawie danych obrazowych, nastąpił po wprowadzeniu konkursu ImageNet Large Scale Visual Recognition Challenge (ILSVRC) w 2010 roku. To wydarzenie miało kluczowe znaczenie dla rozwoju dziedziny, zwłaszcza po osiągnięciu przełomowego sukcesu w 2015 roku, kiedy to udało się rozwiązać problem klasyfikacji obrazów z niespotykaną dotąd dokładnością. Następnie przedstawione zostały bazy danych wykorzystywane do trenowania i testowania modeli opisanych technik, a są to **ImageNet**, **ImageNette** i autorska baza danych. W kolejnej części scharakteryzowane zostały trzy wybrane wstępnie wytrenowane modele z kolekcji **Keras**.

Rozdział ostatni przedstawia wyniki uzyskane podczas testowania technik transferu wiedzy i głębokiego dopasowywania wzorców, będące głównym celem tej pracy. Wnioski w nim zamieszczone pozwalają na głębsze zrozumienie dokładności i efektywności zastosowanych metod, poznanie ich potencjalnych zalet, ograniczeń oraz możliwości dalszego rozwoju. W rozdziale czwartym przedstawiono wyniki klasyfikacji obrazów, uzyskanych dzięki zastosowaniu technik szczegółowo opisanych w poprzednim rozdziale. Dokładność klasyfikacji została przedstawiona w formie tabel i za pomocą graficznych reprezentacji macierzy pomyłek, jak również poprzez wykresy liniowe i słupkowe, co pozwala na głębsze porównanie działania poszczególnych modeli.

Autorka tej pracy szczegółowo opisała technikę głębokiego dopasowywania wzorców (DTM) w artykule konferencyjnym pt. „Robust category recognition based on deep templates for educational mobile applications”. W badaniach zawartych w tym artykule zbadano wpływ liczby próbek na dokładność budowy reprezentacji klasy oraz wybór miar podobieństwa i odległości dla dopasowywania. Konferencja Big Data, na której autorka osobiście zaprezentowała ten artykuł, jest klasyfikowana jako „Core B” na australijskiej liście Core, będącej podstawą ministerialnej listy konferencji punktowanych (konferencja Big Data została oceniona na 70 punktów).

# Notacja

W niniejszej pracy przyjęta została następująca konwencja notacyjna.

1. Obiekty numeryczne i teoria zbiorów:

$x \in \mathbb{R}$  - liczba rzeczywista,

$x_i$  -  $i$ -ty element  $n$ -wymiarowego wektora,

$x_{ij}$  element macierzy  $A$  w  $i$ -tym wierszu i  $j$ -tej kolumnie.

$\mathbb{R}$  - zbiór liczb rzeczywistych,

$\mathbb{R}^n$  - zbiór  $n$ -wymiarowych wektorów liczb rzeczywistych,

$\mathbb{R}^{n \times m}$  - zbiór  $n \times m$ -wymiarowych macierzy liczb rzeczywistych,

$M_{m,n}$  - zbiór wszystkich macierzy rozmiaru  $m \times n$ ,

$\forall$  - symbol kwantyfikatora ogólnego, który oznacza "dla każdego",

$\iff$  symbol równoważności,

$\in$  - symbol przynależności.

2. Funkcje i operatory:

$f(\cdot)$  - funkcja ogólna,

$[.]^T$  - transpozycja wektora lub macierzy,

$\sum$  - sumowanie nad zbiorem elementów,

$A$  - macierz,

$I$  - macierz identyczności,

$e$  - funkcja wykładnicza,

$\sigma$  - funkcja aktywacyjna,

$\nabla$  - gradient funkcji,

$G_{t,ii}$  - suma kwadratów gradientów do czasu  $t$ ,

$E[g^2]_t$  - średni ruchomy kwadrat gradientów,  
 $|\cdot|$  - symbol wartości bezwzględnej,  
 $\circ$  - iloczyn skalarny,  
 $*$  - splot,  
 $\frac{df}{dx}$  pochodna funkcji  $f$  względem zmiennej  $x$ ,  
 $\frac{df}{dy}$  - pochodna funkcji  $f$  względem zmiennej  $y$ ,  
 $\frac{\partial f}{\partial x}$  - pochodna cząstkowa funkcji  $f$  względem zmiennej  $x$ ,  
 $\frac{\partial f}{\partial y}$  - pochodna cząstkowa funkcji  $f$  względem zmiennej  $y$ ,  
 $\lim_{x \rightarrow x_0}$  - granica właściwa funkcji przy  $x$  dążącym do  $x_0$ .

3. Ogólne oznaczenia obiektów w uczeniu maszynowym:

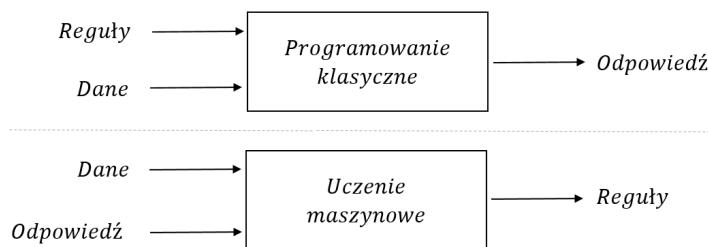
$x$  - dane wejściowe (ang. input),  
 $x^{(i)}$  -  $i$ -ta próbka,  
 $y^{(i)}$  -  $i$ -ta rzeczywista etykieta,  
 $\hat{y}^{(i)}$  -  $i$ -ta etykieta przewidywana,  
 $L(w)$  - funkcja straty (ang. loss function),  
 $w^{(i)}$  - waga  $i$ -tej warstwy,  
 $z$  - suma ważona wejść neuronu,  
 $F(z)$  - funkcja całkowitego pobudzenia układu,  
 $\theta$  - wartość progowa, parametr decydujący o klasyfikacji obiektów,  
 $\varphi(z)$  - funkcja skoku jednostkowego (Heaviside'a) z wartością progową  $\theta$ .

# 1. Zagadnienia teoretyczne

Rozdział pierwszy stanowi teoretyczny opis zagadnień związanych z tematyką niniejszej pracy magisterskiej. Począwszy od wprowadzenia do dziedziny sztucznej inteligencji poprzez wprowadzenie matematycznych definicji i twierzeń, aż po szczegółowy opis mechanizmu działania konwolucyjnych sieci neuronowych.

## 1.1. Sztuczna inteligencja

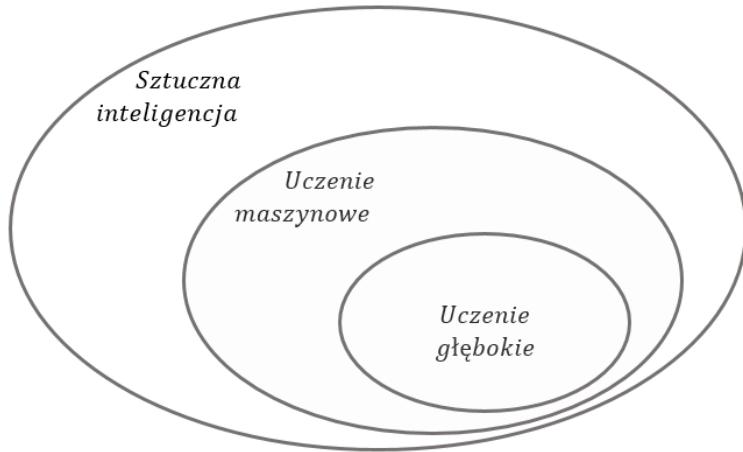
Współczesna koncepcja sztucznej inteligencji znacznie ewoluowała od pierwotnych założeń, które dominowały w latach 50. do końca lat 80. XX wieku. W tamtym okresie AI była głównie rozumiana jako **symboliczna sztuczna inteligencja**, operująca się na precyzyjnie zdefiniowanych zasadach ustalanych przez programistów. To podejście znalazło szerokie zastosowanie w rozwiązywaniu problemów logicznych, jak te występujące w grach takich jak szachy. Jednak, gdy nadeszła konieczność rozwiązywania bardziej skomplikowanych problemów — takich jak klasyfikacja obrazów, rozpoznawanie mowy czy tłumaczenie języków naturalnych, gdzie definiowanie ścisłych reguł okazało się niewystarczające — symboliczna sztuczna inteligencja zaczęła ustępować miejsca **uczeniu maszynowemu**. Ta zmiana paradymatu zilustrowana została na odpowiednim schemacie (patrz rys. 1).



Rysunek 1: Schemat uczenia maszynowego - stary i nowy model programowania.

Przełom nastąpił w latach 90. XX wieku, kiedy to uczenie maszynowe zaczęło dynamicznie się rozwijać, stając się jednym z najszybciej rozwijających się obszarów AI. Wspierane przez postępy w dziedzinie sprzętu komputerowego oraz rosnącą dostępność obszernych zbiorów danych, uczenie maszynowe zapoczątkowało nową erę w rozwoju sztucznej inteligencji.

W celu przedstawienia zależności między sztuczną inteligencją, uczeniem maszynowym oraz uczeniem głębokim wykorzystany został poniższy schemat (patrz rys. 2). Podział ten został zaproponowany przez *François Chollet*, francuskiego informatyka, znanego przede wszystkim jako twórca **Keras**, biblioteki do głębokiego uczenia się, a także współtwórcę pakietu uczenia maszynowego **TensorFlow**. Podkreśla on, że



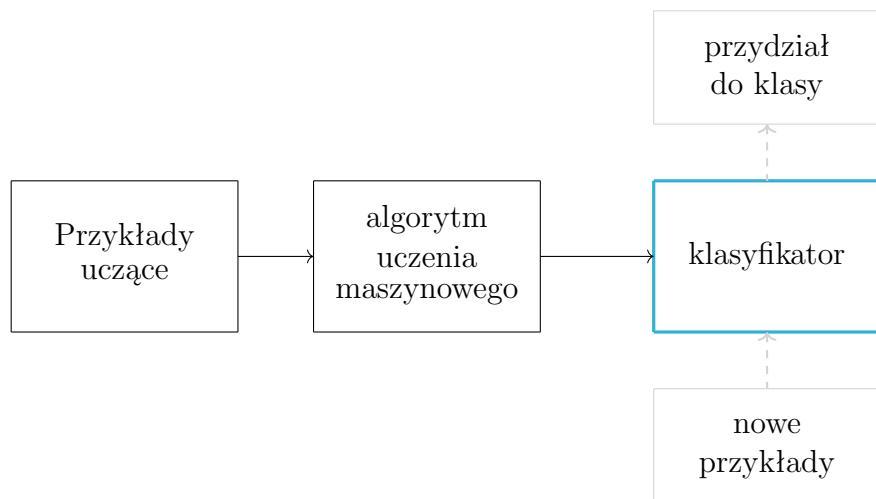
Rysunek 2: Sztuczna inteligencja, uczenie maszynowe i uczenie głębokie.

AI obejmuje szerszy zakres podejść niż tylko uczenie maszynowe i głębokie uczenie, a także inne metody, które nie wymagają uczenia się. Z kolei uczenie maszynowe jest węższym działem, to nowy paradymat programowania, w którym zamiast ręcznego definiowania reguł, komputer automatycznie uczy się, analizując dane treningowe. Głębokie uczenie jest konkretnym poddziałem uczenia maszynowego, w którym termin **głębokie** odnosi się do wielu warstw reprezentacji w modelu, które są uczone automatycznie na podstawie danych treningowych [7].

## 1.2. Uczenie maszynowe

Uczenie maszynowe to dziedzina sztucznej inteligencji, która koncentruje się na opracowywaniu algorytmów wykorzystujących dane w celu poprawy ich dokładności. Technicznie rzecz biorąc, uczenie maszynowe wyszukuje użyteczne reprezentacje niektórych danych wejściowych w ramach wcześniej zdefiniowanej przestrzeni możliwości, wykorzystując wskazówki z sygnału zwrotnego. Ta idea pozwala na rozwiązywanie niezwykle szerokiego zakresu zadań intelektualnych, od rozpoznawania obrazów po autonomiczną jazdę samochodem [7]. Istnieją trzy obszerne kategorie uczenia maszynowego [19]:

1. uczenie nadzorowane (ang. supervised learning) - w tym rodzaju uczenia najistotniejsze jest odpowiednie oznaczenie danych, czyli przypisanie etykiet (ang. labels) do każdego zestawu cech (zmiennych). Algorytmy starają się wypracować regułę (funkcję), która jest wykorzystywana do klasyfikowania nowych przykładów. Proces wyboru najlepszej reguły to szukanie takiego rozwiązania, które będzie dawało jak najmniej błędów na danych treningowych, przy czym nie zostanie utracona zdolność do generalizacji. Najbardziej znane algorytmy uczenie nadzorowane to m.in. regresja liniowa, regresja logistyczna, naiwny klasyfikator bayesowski oraz algorytm  $k$  najbliższych sąsiadów,
2. uczenie nienadzorowane (ang. unsupervised learning) - w przeciwieństwie do poprzedniej metody, algorytmy uczą się z nieoznaczonych danych. Metody takie jak  $k$ -średnich i grupowanie hierarchiczne są często stosowane w uczeniu nienadzorowanym do organizowania danych w grupy, bazując na ich wzajemnym podobieństwie. Jednakże ten rodzaj uczenia również często jest wykorzystywany do klasyfikacji, np. przez sztuczne sieci neuronowe. Głównym zadaniem algorytmów klasyfikacyjnych jest odkrywanie w zbiorze danych wzorców bez wcześniej istniejących etykiet i przy minimalnej ingerencji człowieka,
3. uczenie przez wzmacnianie (ang. reinforcement learning) - idea tego typu uczenia się jest zupełnie inna od poprzednich i opiera się na roli **agenta**, który



Rysunek 3: Schemat ilustrujący przebieg procesu uczenia się dla zadania klasycznego.

próbuje wykonywać pewne akcje zgodne z zadanimi regułami w pewnym środowisku, tak aby maksymalizować zysk - **nagrodę** za dobrze wykonane działania. Uczenie przez wzmacnianie nie opiera się na gotowych danych do nauki, ale na eksperymentalnym odkrywaniu, które działania przynoszą najlepsze efekty. Agent musi rozważać różne opcje i uczyć się z konsekwencji swoich wyborów, co w rezultacie pozwala na opracowanie optymalnej strategii działania w danym środowisku. Jest to dynamiczny proces uczenia się metodą prób i błędów, mający na celu osiągnięcie jak największego zysku z działania.

Pomimo tego, że poszczególne metody uczenia maszynowego różnią się mechanizmem działania, to ogólny schemat przebiegu procesu uczenia się (z wyjątkiem uczenia się przez wzmacnianie) jest taki sam (zobacz rys. 3). ML pozwala na ciągłe doskonalenie rozwiązywania zadań na podstawie zdobytego doświadczenia.

### **1.3. Głębokie sieci neuronowe**

Wraz z rozwojem sztucznej inteligencji, techniki eksploracji danych oparte na uczeniu maszynowym zainspirowały wielu badaczy z różnych dziedzin naukowych. Głębokie sieci neuronowe (ang. Deep Neural Network, DNN) wykorzystywane są głównie do zadań związanych z wizją komputerową [51], rozpoznawaniem mowy [12] czy przetwarzaniem języka naturalnego [31, 54, 20]. Jednakże ich zastosowanie jest znacznie szersze, obejmując takie obszary jak zarządzenie relacjami z klientami [46], marketing [9], internet rzeczy, bezpieczeństwo, statystyka, automatyzacja przemysłowa, robotyka oraz rozrywka.

Istnieją różne rodzaje sztucznych sieci neuronowych, które są wykorzystywane w zależności od typów danych. W pracy skupiono się przede wszystkim na konwolucyjnych sieciach neuronowych, które w przypadku klasyfikacji obrazów są powszechnie używane [25, 21, 37]. Działanie tych sieci opiera się na podstawowych operacjach algebraicznych, takich jak mnożenie macierzy, co pozwala na wykrywanie i identyfikację wzorców w obrazach. Warstwy konwolucyjne skanują obrazy za pomocą filtrów, wyodrębniając istotne cechy, a następnie przekazując je do kolejnych warstw, co umożliwia sieci bardziej zaawansowane i hierarchiczne rozpoznawanie obiektów.

Działanie CNN wymaga dużych zasobów obliczeniowych i ogromnych ilości danych treningowych, a ich dokładność zależy od odpowiedniego doboru architektury sieci i hiperparametrów. Optymalizować działanie sieci można za pomocą **transfery wiedzy i głębokiego dopasowywania wzorców**, co umożliwia wykorzystanie wcześniejszych reprezentacji i cech, znacznie poprawiając efektywność uczenia przy ograniczonych danych i redukując zapotrzebowanie na moc obliczeniową.

Zainteresowanie tymi technikami wynika przede wszystkim z ich zdolności do efektywnego wykorzystania w kontekście uczenia się **metodą niewielu strzałów**. Ta metoda uczenia jest szeroko stosowana w aplikacjach mobilnych, gdzie ograniczenia sprzętowe wymuszają stosowanie małych i lekkich modeli sieciowych. Dzięki swojej dokładności i efektywności, techniki te mają potencjał do znaczącego wpływu na rozwój inteligentnych systemów mobilnych. Zdolność do szybkiego i precyzyjnego

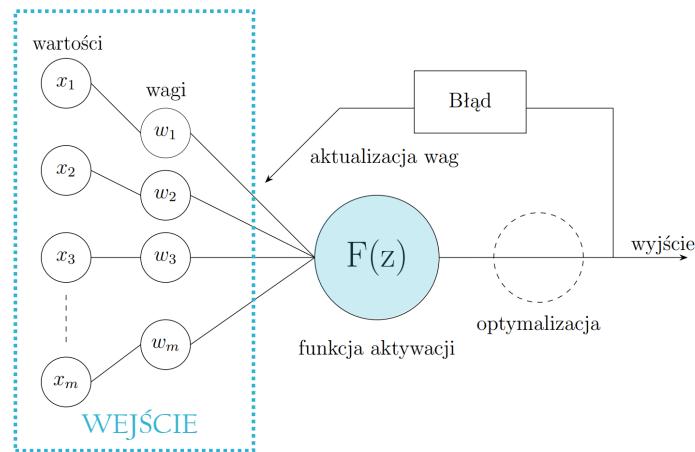
rozpoznawania obrazów może mieć znaczący wpływ na szybkość i dokładność diagnost, co przekłada się na poprawę jakości życia i zdrowia. Dzięki wykorzystaniu AI możliwa jest identyfikacja różnorodnych przedmiotów, osób, wykrywanie scen i działań zagrażających bezpieczeństwu ludzi, ale również klasyfikowanie, lokalizowanie w przestrzeni i czasie [55, 26]. Pozwala to reagować na niepożądane sytuacje szybko, co przyczynia się do poprawy bezpieczeństwa i optymalizacji procesów decyzyjnych w różnych dziedzinach, od monitoringu miejskiego po zarządzanie kryzysowe. CNN są fundamentem dla algorytmów wizyjnych wykorzystywanych w pojazdach autonomicznych. Dzięki nim, pojazdy te są w stanie precyzyjnie rozpoznawać i klasyfikować różne obiekty na drodze, takie jak inni uczestnicy ruchu, piesi, znaki drogowe, czy przeszkody [1]. Co więcej, zaawansowane modele sieciowe potrafią przewidywać potencjalne zagrożenia i reagować na nie w czasie rzeczywistym, co jest niezbędne dla bezpieczeństwa i dokładności tych pojazdów [17]. W ramach tej pracy magisterskiej dąży się do głębokiego zrozumienia technik konwolucyjnych sieci neuronowych i ich aplikacji w praktycznych scenariuszach rozpoznawania obrazów. Celem jest nie tylko teoretyczne zgłębianie tematu uwzględniające matematyczne aspekty, ale także praktyczne zastosowanie i ewaluacja wybranych technik, tak aby przeprowadzone badania przyczyniły się do postępu w tej dynamicznie rozwijającej się dziedzinie.

### **Budowa prostej sieci neuronowej**

**Perceptron** jest rodzajem prostego sztucznego neuronu (patrz rys. 4), który działa na zasadzie klasyfikacji danych na podstawie ich wejścia i przypisywania odpowiednich wartości na wyjściu. Perceptron podlega procesowi trenowania (proces ten dokładnie opisany będzie w kolejnym rozdziale), co pozwala mu na samodzielną klasyfikację danych. Ten pojedynczy neuron składa się z trzech głównych elementów: wejście, wagi oraz funkcji aktywacji. Dodatkowo w budowie perceptronu uwzględniona została optymalizacja, wyznaczanie błędu i aktualizacja wag, co zostało opisane w dalszej części pracy.

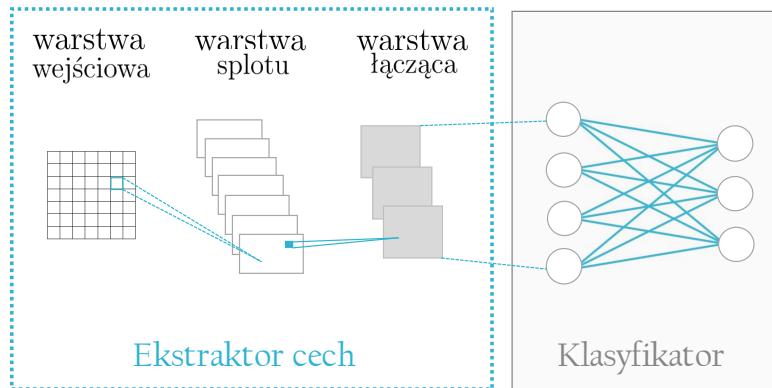
Perceptron przyjmuje na wejściu zestaw wartości, które reprezentują cechy  $x = [x_1, \dots, x_n]$  lub atrybuty danych. Następnie każdemu z atrybutów przypisywa-

ne są ich wagi  $w = [w_1, \dots, w_n]$ , która decyduje o wpływie danej cechy na wynik końcowy. Wagi te są aktualizowane w procesie trenowania perceptronu, aby dostosować jego zachowanie do konkretnego zadania.



Rysunek 4: Schemat budowy prostej sieci neuronowej.

Sieci neuronowe składają się z wielu perceptronów, które współpracują, tworząc bardziej złożone architektury (patrz rys. 5).



Rysunek 5: Schemat architektury konwolucyjnej sieci neuronowej, która zbudowana jest z jednej warstwy splotu i jednej warstwy łączącej.

Budowa CNN składa się z dwóch głównych części:

1. ekstraktor cech (ang. feature extractor) - odpowiada za wydobywanie istotnych informacji z danych wejściowych, składa się z następujących warstw:
  - (a) warstwa wejściowa - w przypadku obrazów warstwa ta przyjmuje surowe piksele jako dane wejściowe,
  - (b) warstwy splotu (konwolucyjne) - warstwa, w której stosuje się filtry do danych wejściowych. Na tym etapie model uczy się wykrywania cech. Złożona architektura modelu zwykle składa się w wielu warstw splotu połączonych ze sobą warstwą łączącą. Wczesne warstwy konwolucyjne wyodrębniają ogólne lub niskopoziomowe cechy, takie jak linie i krawędzie, podczas gdy późniejsze warstwy uczą się drobniejszych szczegółów lub cech wysokiego poziomu abstrakcji,
  - (c) warstwa łącząca - warstwa ta jest odpowiedzialna za zmniejszenie wymiarowości map cech wygenerowanych przez warstwy konwolucyjne w celu zaoszczędzenia kosztów obliczeniowych. Najpopularniejsze rodzaje operacji w warstwie łączącej to max pooling, average pooling czy sum pooling,
2. klasyfikator

- (a) warstwa pełnych połączeń - warstwa wykorzystując cechy wyodrębnione przez poprzednie warstwy konwolucyjne i łączące przewiduje klasy obrazu. W warstwach pełnych połączeń każdy neuron jest połączony ze wszystkimi aktywacjami z poprzedniej warstwy, co umożliwia modelowi uczenie się złożonych wzorców i relacji między cechami.

Rozbudowane architektury sieci neuronowych często nazywa się ogólniej jako **modele**. Pojęcie modelu odnosi się do algorytmu lub techniki, która jest używana do rozwiązania określonego problemu, uwzględniając nie tylko budowę architektury sieci, ale także aspekty takie jak dostrajanie hiperparametrów.

Opisując model jako całą architekturę sieci neuronowej wraz z jej działaniem pod uwagę bierze się kilka kluczowych elementów:

1. etap konstrukcji sieci neuronowej - budując sieć neuronową wykorzystuje się warstwy ukryte (ang. hidden) lub wyjściowe (ang. output). Warstwy ukryte przetwarzają informacje między warstwami wejściowymi a wyjściowymi, ucząc się reprezentacji danych. Warstwy wyjściowe generują wynik końcowy sieci. Opcjonalnie możliwa jest także kontynuacja uczenia już istniejącego modelu, więcej o tym podejściu zostało napisane w rozdziale dotyczącym zaawansowanej klasyfikacji obrazów,
2. etap trenowania - to etap, w którym model jest uczyony na danych treningowych. W tym procesie modele dostosowują swoje wagi (parametry) w celu minimalizacji błędu predykcji i poprawienia jakości przewidywań,
3. etap testowania/walidacji - po trenowaniu modelu, jest on testowany na danych walidacyjnych lub testowych, aby ocenić jego dokładność i zdolność do generalizacji na nowe dane.

Dokładny opis etapu trenowania oraz testowania został przedstawiony na końcu tego rozdziału. Wcześniej zdefiniowane zostały najistotniejsze matematyczne pojęcia oraz niezbędne informacje dotyczące przetwarzania obrazów w kontekście CNN.

## **1.4. Matematyczne podstawy**

W tym podrozdziale przedstawiono teoretyczne podstawy najważniejszych zagadnień z zakresu algebry [33] i analizy matematycznej [27], które stanowią fundament dla zrozumienia mechanizmów klasyfikacji obrazów. W powyższych pozycjach książkowych można znaleźć również opisy wielu nieomówionych tutaj pojęć oraz dowody twierdzeń.

**Definicja 1.** Macierz  $A \in \mathbb{R}^{m \times n}$  nazywany jest układem  $m \times n$  liczb rzeczywistych, gdzie  $m$  to liczba wierszy, a  $n$  liczba kolumn zgrupowanych w następującej postaci:

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}.$$

Jeśli  $n = 1$ , wówczas macierz nazywa się **wektorem kolumnowym**  $v$ , natomiast jeśli  $m = 1$  wówczas macierz taką nazywa się **wektorem** co można zapisać jako  $v^T$ , gdzie  $T$  oznacza **transpozycję**.

**Definicja 2.** Niech  $v, u \in \mathbb{R}^n$  będą dwoma wektorami  $v = [v_1, \dots, v_n], u = [u_1, \dots, u_n]$ , wówczas **iloczynem skalarnym** oznaczanym jako  $\circ$ , definiuje się liczbę rzeczywistą  $v^T u$  w postaci:

$$v \circ u = v^T u = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} [u_1, \dots, u_n] = \sum_{i=1}^n v_i u_i. \quad (1)$$

**Przykład 1.1.** Dane są dwa wektory  $v = [1, 2, 3], u = [4, 5, 6]$ , wówczas ich iloczyn skalarny obliczyć można w następujący sposób:

$$v \circ u = v^T u = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} [4, 5, 6] = 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 = 4 + 10 + 18 = 32.$$

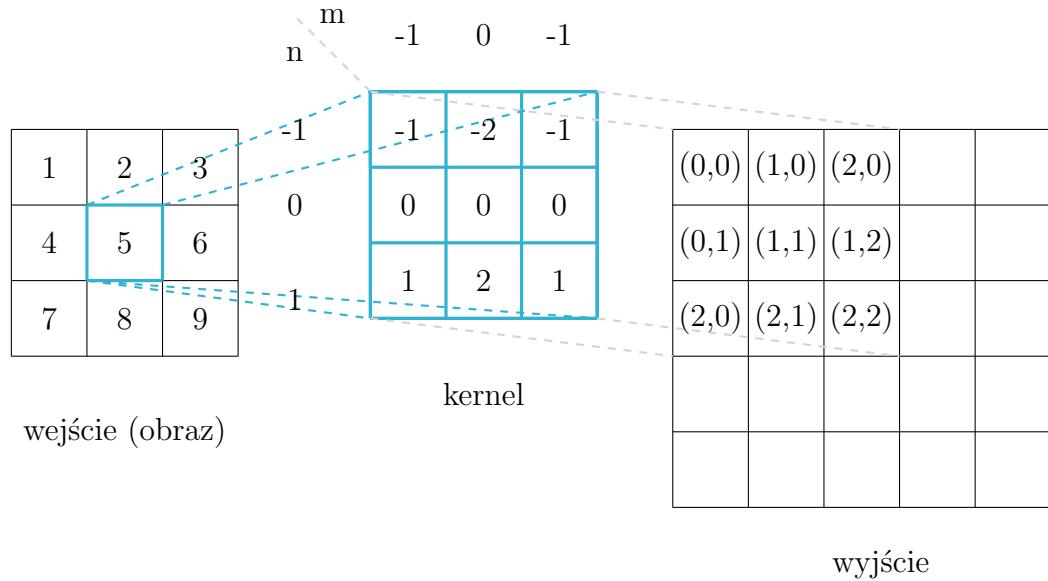
Konwolucyjne sieci neuronowe różnią się od innych sieci neuronowych doskonałą wydajnością w przetwarzaniu obrazów. Termin konwolucja oznacza „splot” i odnosi się do funkcji splotu  $h$ , która jest szczególnym rodzajem operacji liniowej dwóch funkcji.

**Definicja 3.** Niech  $g$  będzie funkcją filtrem/kernel (niewielką macierzą o wymiarach np.  $3 \times 3px$ ) wzduż sygnału  $f$  (dwuwymiarowa macierz zawierająca wartości pikseli obrazu), wówczas funkcję **splotu**  $h$  wyrazić można w następujący sposób:

$$h(m, n) = (f * g)(m, n) = \sum_j \sum_k f(j, k) \cdot g(m - j, n - k). \quad (2)$$

**Przykład 1.2.** Funkcja splotu dla 2-wymiarowego fragmentu obrazu o wymiarach  $3 \times 3$  i odpowiadającemu mu krenelowi (filtrowi) oblicza się poprzez odpowiednią operację na macierzy. Rozważa się następującą macierz wejściową  $A$ :

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$



Rysunek 6: Wizualizacja działania funkcji splotu dla wyjścia  $h(1, 1)$ . Tutaj w przypadku kernela początek  $(0,0)$  znajduje się w jego środku. Należy pamiętać, że podczas obliczania filtr ten ulega odbiciu w poziomie i w pionie.

Wyjście w  $(1, 1)$  dla tego przykładu będzie następujące:

$$\begin{aligned}
 h(1, 1) = & f(0, 0) \cdot g(1, 1) + f(1, 0) \cdot g(0, 1) + f(2, 0) \cdot g(-1, 1) + f(0, 1) \cdot g(1, 0) + f(1, 1) \cdot g(0, 0) + \\
 & f(2, 1) \cdot g(-1, 0) + f(0, 2) \cdot g(1, -1) + f(1, 2) \cdot g(0, -1) + f(2, 2) \cdot g(-1, -1) = \\
 & 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1 + 4 \cdot 0 + 5 \cdot 0 + 6 \cdot 0 + 7 \cdot (-1) + 8 \cdot (-2) + 9 \cdot (-1) = -24.
 \end{aligned}$$

Zgodnie z powyższym schematem, oblicza się kolejne wartości funkcji splotu  $h$ :

$$h(0,0) = 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1) = -13,$$

$$h(1,0) = 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 + 4 \cdot (-1) + 5 \cdot (-2) + 6 \cdot (-1) = -20,$$

$$h(1,2) = 4 \cdot 1 + 5 \cdot 2 + 6 \cdot 1 + 8 \cdot 0 + 9 \cdot 0 + 0 \cdot 0 + 0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1) = 20,$$

$$h(2,0) = 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 + 5 \cdot (-1) + 6 \cdot (-2) + 0 \cdot (-1) = -17,$$

$$h(2,1) = 2 \cdot 1 + 3 \cdot 2 + 0 \cdot 1 + 5 \cdot 0 + 6 \cdot 0 + 0 \cdot 0 + 8 \cdot (-1) + 9 \cdot (-2) + 0 \cdot (-1) = -18,$$

$$h(2,2) = 5 \cdot 1 + 6 \cdot 2 + 0 \cdot 1 + 8 \cdot 0 + 9 \cdot 0 + 0 \cdot 0 + 0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1) = 17,$$

$$h(0,2) = 0 \cdot 1 + 4 \cdot 2 + 5 \cdot 1 + 0 \cdot 0 + 7 \cdot 0 + 8 \cdot 0 + 0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1) = 13.$$

Co ostatecznie daje następujące wartości na wyjściu:

-13	-20	-17
-18	-24	-18
13	20	17

**Twierdzenie 1.** Niech funkcja  $h(m,n)$  będzie splotem sygnałów  $f$  i  $g$  wówczas zachodzą następujące własności:

1. przemienność:  $(f * g)(m,n) = (g * f)(m,n)$ ,
2. łączność:  $((f * g) * i)(m,n) = (f * (g * i))(m,n)$ ,
3. rozdzielność względem dodawania:  $(f * (g + i))(m,n) = ((f * g) + (f * i))(m,n)$ .

Funkcja straty, oznaczana jako  $L(w)$  pozwala zmierzyć dokładność modelu predykcyjnego. Definiowana jest jako suma błędów dla poszczególnych obserwacji w zbiorze danych.

**Definicja 4.** Niech  $L(w)$  będzie funkcją straty zdefiniowaną następująco:

$$L(w) = \sum_{i=1}^n l_i(w), \quad (3)$$

gdzie  $l_i$  to błąd  $i$ -tej obserwacji, a  $n$  to liczba obserwacji w zbiorze danych.

**Przykład 1.3.** Dane są trzy obserwacje (czyli  $n = 3$ ) z następującymi błędami obserwacji:  $l_1(w) = 2, l_2(w) = 4$  oraz  $l_3(w) = 3$ . Zgodnie z definicją funkcji straty  $L(w)$ :

$$L(w) = \sum_{i=1}^3 l_i(w) = l_1(w) + l_2(w) + l_3(w) = 2 + 4 + 3 = 9.$$

**Twierdzenie 2.** (Warunek konieczny i wystarczający istnienia pochodnej funkcji w punkcie).

Niech funkcja  $f$  będzie określona w otoczeniu punktu  $x_0 \in \mathbb{R}$ . Funkcja  $f$  posiada pochodną w punkcie  $x_0$  wtedy i tylko wtedy gdy prawostronna i lewostronna granica w tym punkcie są równe.

Dla funkcji posiadającej wiele zmiennych, zaczynając od dwóch argumentów, pochodna cząstkowa jest definiowana podobnie jak pochodna w punkcie.

**Definicja 5.** Niech  $(x_0, y_0) \in \mathbb{R}^2$  będzie otoczeniem otartym punktu  $(x, y)$  to **pochodna cząstkowa funkcji**  $f$ :

1. względem  $x$  w punkcie  $(x_0, y_0)$  oznaczana  $\frac{\partial f(x_0, y_0)}{\partial x}$  nazywa się granicę:

$$\lim_{x \rightarrow x_0} \frac{f(x, y_0) - f(x_0, y_0)}{x - x_0}, \quad (4)$$

2. względem  $y$  w punkcie  $(x_0, y_0)$  oznaczana  $\frac{\partial f(x_0, y_0)}{\partial y}$  nazywa się granicę:

$$\lim_{y \rightarrow y_0} \frac{f(x_0, y) - f(x_0, y_0)}{y - y_0}. \quad (5)$$

**Przykład 1.4.** Aby wyznaczyć pochodne cząstkowe funkcji  $f(x, y) = xy^2 - yx$ , należy podstawić  $h = x - x_0$  do wzoru (4):

$$\frac{\partial f(x_0, y_0)}{\partial x} = \lim_{h \rightarrow 0} \frac{[(x_0 + h) \cdot y_0^2 - y_0 \cdot (x_0 + h)] - [x_0 y_0^2 - y_0 x_0]}{h} = \lim_{h \rightarrow 0} \frac{hy_0^2 - y_0 h}{h} =$$

$$\lim_{h \rightarrow 0} \frac{h(y_0^2 - y_0)}{h} = y_0^2 - y_0.$$

Następnie podstawić należy  $g = y - y_0$  do wzoru (5):

$$\frac{\partial f(x_0, y_0)}{\partial y} = \lim_{g \rightarrow 0} \frac{[(y_0^2 + g) \cdot x_0 - x_0 \cdot (y_0 + g)] - [x_0 y_0^2 - y_0 x_0]}{g} = 2x_0 y_0.$$

Analogicznie do pochodnej cząstkowej, zdefiniować można pochodną kierunkową.

**Definicja 6.** Niech  $v = [v_x, v_y]$  będzie wektorem niezerowym wówczas **pochodną kierunkową** funkcji  $f(x, y)$  w punkcie  $(x_0, y_0) \in \mathbb{R}^2$  oznaczaną jako  $\frac{\partial f(x_0, y_0)}{\partial v}$  nazywa się granicę:

$$\lim_{t \rightarrow 0^+} \frac{f(x_0 + tv_x, y_0 + tv_y) - f(x_0, y_0)}{t}, \quad (6)$$

gdzie  $t$  dąży do 0 z prawej strony ponieważ określa tempo zmian funkcji w danym kierunku.

**Przykład 1.5.** Pochodną kierunkową funkcji  $f(x, y) = ye^{xy}$  w kierunku wektora  $v = [4, -3]$  w punkcie  $(0, 0)$ , można wyznaczyć w następujący sposób:

$$\lim_{t \rightarrow 0^+} \frac{f(0 + 4t, 0 - 3t) - f(0, 0)}{t} = \lim_{t \rightarrow 0^+} \frac{-3t \cdot e^{-12t^2}}{t} = \lim_{t \rightarrow 0^+} -3 \cdot e^{-12t^2} = -3.$$

**Definicja 7.** Gradientem, oznaczanym symbolem  $\nabla f$  funkcji  $f(x, y)$  w punkcie  $(x_0, y_0) \in \mathbb{R}^2$  nazywany jest wektor określony wzorem:

$$\nabla f(x_0, y_0) = \begin{bmatrix} \frac{\partial f(x_0, y_0)}{\partial x} \\ \frac{\partial f(x_0, y_0)}{\partial y} \end{bmatrix}. \quad (7)$$

**Przykład 1.6.** Gradient funkcji  $f(x, y) = 2x + 3y^2$  w punkcie  $(1, -1)$  korzystając ze wzoru (7) obliczyć można następująco:

$$\frac{\partial f(1, -1)}{\partial x} = 2, \quad \frac{\partial f(1, -1)}{\partial y} = -6,$$

$$\nabla f(1, -1) = \begin{bmatrix} 2 \\ -6 \end{bmatrix}.$$

Podobnie dla funkcji 3 lub więcej zmiennych, powyższą definicję gradientu można zapisać z wykorzystaniem macierzy  $A$ .

**Definicja 8.** Niech  $f : \mathbf{R}^{m \times n} \rightarrow \mathbf{R}$  wówczas gradientem funkcji  $f$  jest macierz  $A \in M_{m,n}\mathbf{R}^{m \times n}$  zdefiniowana następująco:

$$\nabla_A f(A) \in \mathbf{R}^{m \times n} = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{12}} & \cdots & \frac{\partial f(A)}{\partial A_{1n}} \\ \frac{\partial f(A)}{\partial A_{21}} & \frac{\partial f(A)}{\partial A_{22}} & \cdots & \frac{\partial f(A)}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(A)}{\partial A_{m1}} & \frac{\partial f(A)}{\partial A_{m2}} & \cdots & \frac{\partial f(A)}{\partial A_{mn}} \end{bmatrix}, \quad (8)$$

gdzie

$$(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}. \quad (9)$$

Funkcja  $f(x, y)$  określona w otoczeniu punktu  $(x_0, y_0) \in \mathbb{R}^2$  jest w tym punkcie różniczkowana, jeśli istnieją stałe  $H, K \in \mathbb{R}$ , takie, że:

$$\lim_{(h,k) \rightarrow (0,0)} \frac{f(x_0 + h, y_0 + k) - f(x_0, y_0) - Hh - Kk}{\sqrt{h^2 + k^2}} = 0. \quad (10)$$

Uwaga. W punktach, w których funkcja jest różniczkowalna jest ona również ciągła.

**Twierdzenie 3.** Jeśli funkcja  $f(x, y)$  posiada ciągłe pochodne cząstkowe w punkcie  $(x_0, y_0)$  to pochodna kierunkowa dla każdego  $v \in \mathbb{R}^n$  może być wyrażona wzorem:

$$\frac{\partial f(x_0, y_0)}{\partial v} = \nabla f(x_0, y_0) \circ v, \quad (11)$$

gdzie  $\circ$  oznacza iloczyn skarany wektorów.

**Przykład 1.7.** Korzystając z **Twierdzenia 2.** wyznaczamy pochodną kierunkową dla funkcji  $f = (x^2 + 2y^3)$  w punkcie  $(1, 1)$  w kierunku wektora  $v = [3, 3]$ .

Gradientu w punkcie  $(1, 1)$  jest równy:

$$\nabla f = \begin{bmatrix} 2 \\ 6 \end{bmatrix}.$$

Teraz obliczona zostanie pochodna kierunkowa:

$$\frac{\partial f(1, 1)}{\partial v} = \begin{bmatrix} 2 \\ 6 \end{bmatrix} \circ [3, 3] = 24.$$

Dla porównania można obliczyć pochodną kierunkową z tego przykładu, tym razem wykorzystując definicję:

$$\lim_{h \rightarrow 0} \frac{f(1 + 3h, 1 + 3h) - f(1, 1)}{h}.$$

Wartość pochodnej kierunkowej jest równa  $f(x^2 + 2y^3)$  dla  $x = 1 + 3h$  i  $y = 1 + 3h$ :

$$f(1 + 3h)^2 + 2(1 + 3h)^3 = (1 + 3h)^2 + 2(1 + 3h)^3 = \\ (1 + 6h + 9h^2) + 2(1 + 9h + 27h^2) = 1 + 6h + 9h^2 + 2 + 18h + 54h^2 = 3 + 24h + 63h^2.$$

Podstawiając tę wartość do definicji otrzyma się:

$$\lim_{h \rightarrow 0} \frac{3 + 24h + 63h^2 - (1 + 2)}{h} = \lim_{h \rightarrow 0} \frac{24h + 63h^2}{h} = \lim_{h \rightarrow 0} (24 + 63h) = 24.$$

Ostatecznie pochodna kierunkowa obliczona z definicji jak również z wykorzystaniem twierdzenia w kierunku wektora  $v = [3, 3]$  w punkcie  $(1, 1)$  wynosi 24.

**Twierdzenie 4.** Niech  $f$  i  $g$  będą funkcjami różniczkowalnymi w pewnym przedziale. Wówczas następujące własności są prawdziwe:

1. pochodna sumy dwóch funkcji jest równa sumie ich pochodnych:

$$\frac{d}{dx}(f(x) + g(x)) = f'(x) + g'(x), \quad (12)$$

2. pochodna iloczynu dwóch funkcji jest równa pochodnej pierwszej funkcji pomnożonej przez drugą funkcję plus pierwsza funkcja pomnożona przez pochodną drugiej funkcji:

$$\frac{d}{dx}(f(x) \cdot g(x)) = f'(x) \cdot g(x) + f(x) \cdot g'(x), \quad (13)$$

3. pochodna ilorazu dwóch funkcji jest równa różnicy ilorazu pochodnej licznika przez mianownik i iloczynu licznika przez pochodną mianownika, podzielonej przez kwadrat mianownika.

$$\frac{d}{dx} \left( \frac{f(x)}{g(x)} \right) = \frac{f'(x) \cdot g(x) - f(x) \cdot g'(x)}{[g(x)]^2}, \quad (14)$$

4. pochodna złożenia dwóch funkcji jest równa pochodnej funkcji zewnętrznej obliczonej w punkcie będącym wartością funkcji wewnętrznej, pomnożonej przez pochodną funkcji wewnętrznej.

$$\frac{d}{dx} f(g(x)) = f'(g(x)) \cdot g'(x). \quad (15)$$

**Przykład 1.8.** Dane są funkcje  $f(x) = x^2$  i  $g(x) = \sin(x)$ . Korzystając z powyższego twierdzenia obliczone zostaną pochodne dla sumy, iloczynu, ilorazu oraz złożenia tych funkcji.

Pochodna sumy tych funkcji wynosi:

$$\frac{d}{dx}(f(x) + g(x)) = \frac{d}{dx}(x^2 + \sin(x)) = \frac{d}{dx}x^2 + \frac{d}{dx}\sin(x) = 2x + \cos(x).$$

Pochodna iloczynu tych funkcji wynosi:

$$\frac{d}{dx}(f(x) \cdot g(x)) = \frac{d}{dx}(x^2 \cdot \sin(x)) = \frac{d}{dx}x^2 \cdot \sin(x) + x^2 \cdot \frac{d}{dx}\sin(x) = 2x\sin(x) + x^2\cos(x).$$

Pochodna ilorazu tych funkcji wynosi:

$$\frac{d}{dx} \left( \frac{f(x)}{g(x)} \right) = \frac{d}{dx} \left( \frac{x^2}{\sin(x)} \right) = \frac{2x\sin(x) - x^2\cos(x)}{\sin^2(x)}.$$

Pochodna złożenia tych funkcji wynosi:

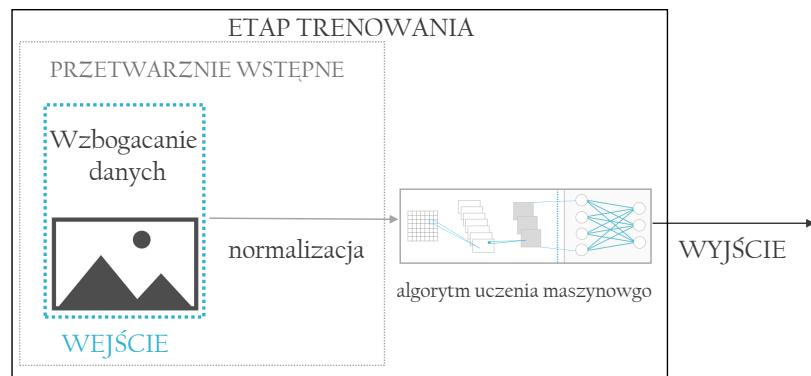
$$\frac{d}{dx} f(g(x)) = \frac{d}{dx}(\sin(x))^2 = 2\sin(x)\cos(x).$$

## 1.5. Przetwarzanie obrazów z wykorzystaniem sieci neuronowych

W rozpoznawaniu obrazów, dane wejściowe, czyli obrazy, pełnią kluczową rolę na której opiera się cały proces rozpoznawania i klasyfikacji. Zatem obrazy są reprezentacją danych, na których model uczy się rozpoznawać wzorce i cechy charakterystyczne dla różnych klas obiektów. Ogólnie ze względu na charakter typy danych wejściowych możemy podzielić na cztery kategorie:

1. dane wektorowe - dane jednowymiarowe,
2. dane szeregu czasowego lub dane sekwencyjne,
3. obrazy - dane dwuwymiarowe czyli macierze,
4. materiały wideo - dane trójwymiarowe.

W przypadku obrazów mamy do czynienia z danymi dwuwymiarowymi, co oznacza, że każdy obraz jest reprezentowany jako zbiór pikseli w tabeli o określonej szerokości i wysokości. Ze względu na specyfikę danych, odpowiednie ich przygotowanie jest kluczowe jako pierwszy etap trenowania sieci. Wymaga to przetwarzania wstępnego, które obejmuje wzbogacanie danych i normalizację (patrz rys. 7).



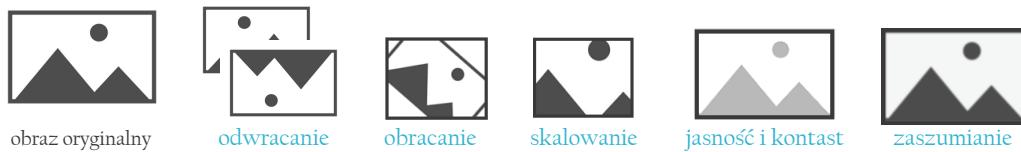
Rysunek 7: Schemat przedstawiający etapy procesu uczenia się, uwzględniający przetwarzanie wstępne, w tym wzbogacanie danych oraz normalizację.

## Przetwarzanie wstępne

**Przetwarzanie wstępne** (ang. preprocessing) jest to proces wykonywania szeregu operacji na danych wejściowych przed ich przekazaniem do algorytmu rozpoznawania obrazów. Głównym celem przetwarzania wstępnego jest zoptymalizowanie jakości danych i przygotowanie ich do bardziej efektywnego i skutecznego uczenia się przez model.

1. **Wzbogacanie danych** (ang. data augmentation) to zbór technik stosowanych w celu sztucznego zwiększenia różnorodności i rozmiaru szkoleniowego zbioru danych, mogą one obejmować następujące sposoby przekształcania obrazu (patrz rys. 8):

- odwracanie (ang. flipping) - odwrócenie obrazu w poziomie lub w pionie,
- obracanie (ang. rotation) - obracanie obrazu o określony kąt,
- skalowanie (ang. scaling) - zmiana rozmiaru przy zachowaniu proporcji,
- regulacja jasności i kontrastu (ang. brightness and contrast),
- zaszumianie (ang. noise) - dodawanie losowego szumu do obrazu.



Rysunek 8: Graficzna interpretacja wybranych technik wzbogacania danych.

Główną zaletą wynikającą z zastosowanie tej techniki jest uzyskanie znacznie lepszej generalizacji [30]. Model staje się bardziej odporny i skuteczniejszy w radzeniu sobie z rzeczywistymi zmianami w danych wejściowych. W przypadku obrazów zmiany takie są naturalne ponieważ kontrast i oświetlenie fotografii, a także poziom ostrość może być zróżnicowany. Dodatkowo zwiększenie różnorodności wśród obrazów zmniejsza ryzyko przeuczenia się modelu, czyli nadmiernego dopasowania modelu do treningowych danych. Model ma możliwość

nauczenia się istotnych wzorców i cech charakterystycznych, zamiast zapamiętywać konkretne przykłady.

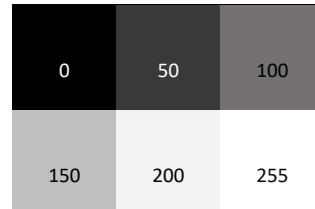
W sytuacjach, gdy pozyskanie większej ilości oryginalnych danych wiąże się z wysokimi kosztami lub długim czasem zbierania, technika rozszerzania danych pozwala na maksymalne wykorzystanie dostępnych zasobów [60]. Przez tworzenie syntetycznych przykładów z istniejących danych, model może lepiej uchwycić różnorodność możliwych scenariuszy i zwiększyć swoje zdolności predykcyjne na danych nowych i niewidzianych wcześniej.

2. **Normalizacja** ma na celu przeskalowanie wartości danych, aby znajdowały się w określonym przedziale lub miały określony rozkład. W odniesieniu do rozpoznawania obrazów, wartości pikseli będące danymi wejściowymi algorytmu powinny przyjmować wartość z przedziału  $[0, 1]$  w celu uniknięcia dużych zmian gradientu co utrudniłoby w znacznym stopniu znalezienie optymalnego rozwiązania. Normalizacja pikseli do zakresu  $[0, 1]$  może być wykonana za pomocą wzoru:

$$N_{value} = \frac{original_{value}}{255}, \quad (16)$$

gdzie  $original_{value}$  to wartość piksela przed normalizacją, a  $N_{value}$  to wartość piksela po normalizacji.

**Przykład 1.9.** Dany jest czarno-biały obraz o wymiarach  $2 \times 3$ , gdzie piksele o określonej jasności reprezentują wartości od 0 (czarny) do 255 (biały).



Wykorzystując wzór (1) otrzymane zostaną następujące wartości znormalizowane:

$$N_{value} = \frac{0}{255} = 0.0000, \quad N_{value} = \frac{50}{255} = 0.1961, \quad N_{value} = \frac{100}{255} = 0.3922,$$

$$N_{value} = \frac{150}{255} = 0.5882, \quad N_{value} = \frac{200}{255} = 0.7843, \quad N_{value} = \frac{255}{255} = 1.0000.$$

Ostatecznie otrzymana została macierz znormalizowanych danych wejściowych:

$$\begin{bmatrix} 0.0000 & 0.1961 & 0.3922 \\ 0.5882 & 0.7843 & 1.0000 \end{bmatrix}.$$

### 1.5.1. Przypisywanie etykiet

Niech  $\hat{y}$  będzie etykieta klasy przewidziana przez wcześniej zdefiniowaną funkcję skoku jednostkowego, a równoczesną aktualizację każdej wagi  $w_j$  w wektorze wag w co można zapisać następująco:

$$w_j = w_j + \Delta w_j. \quad (17)$$

Służąca do aktualizowania wagi wartość  $\Delta w_j$  jest wyliczana za pomocą reguły uczenia w następujący sposób:

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)}, \quad (18)$$

gdzie  $\eta$  jest współczynnikiem uczenia (jest to stała przyjmująca wartości w zakresie od 0 do 1),  $y^{(i)}$  to rzeczywista etykieta klasy  $i$ -tej próbki, a  $\hat{y}^{(i)}$  to etykieta przewidywana klasy  $i$ -tej próbki,  $x_j^{(i)}$  to  $j$ -ty element wektora danych wejściowych dla  $i$ -tej próbki.

### 1.5.2. Funkcja całkowitego pobudzenia układu

W poprzedniej części dotyczącej budowy perceptronu wspomniano o całkowitym pobudzeniu układu, które w tym podrozdziale zostało dokładnie umówione. Klasyfikacja binarna odnosi się do zadań klasyfikacyjnych, które mają dwie klasy. W ujęciu matematycznym odnosi się ona do dwóch klas: 1 (klasa pozytywna) oraz  $-1$  (klasa negatywna).

**Definicja 9.** Niech  $z$  będzie liniową kombinacją wartości wejściowych  $x_1, \dots, x_m$  oraz powiązanych z nimi wektorem wag  $w = [w_1, \dots, w_m]$ :

$$z := w_1x_1 + \dots + w_mx_m = \sum_{j=0}^m w_jx_j = w^T x. \quad (19)$$

Wówczas funkcja  $F(z)$  nosi nazwę **całkowitego pobudzenia układu**. Zmienna  $z$  jest to iloczyn skalarny dwóch wektorów, którymi są wektor wag  $w = [w_1, \dots, w_n]$  oraz podanych na wejściu sygnały  $x_1, \dots, x_n$  tworzące wektor  $x$ .

Jeżeli całkowite pobudzenie z danej próbki  $x^{(i)}$  jest wyższe od zdefiniowanej wartości progowej  $\theta$ , to przewiduje się, że dany obiekt przynależy do klasy 1, w przeciwnym wypadku — do klasy przeciwej −1. W algorytmie funkcja aktywacji jest prostą funkcją skoku jednostkowego, zwaną czasem również funkcją skokową Heaviside'a:

$$\varphi(z) = \begin{cases} 1 & z \geq \theta \\ -1 & z < \theta. \end{cases} \quad (20)$$

Algorytm działania perceptronu można opisać następującymi etapami:

1. inicjalizacja wag o wartościach 0 lub niewielkich, losowych wartościach,
2. dla każdej próbki uczącej  $x^{(i)}$  wykonaj poniższe czynności:
  - (a) oblicz wartość wyjściową  $\hat{y}$ ,
  - (b) zaktualizuj wagi.

### Algorytm propagacji wstecz

Propagacja wsteczna (ang. backpropagation) - algorytm ten umożliwia aktualizację wag sieci w celu minimalizacji funkcji straty. Rozważa się sieć neuronową  $f$  związaną z trzema różnymi zestawami wag:  $W_1, W_2, W_3$ . Matematycznie struktura sieci może być przedstawiona jako złożenie trzech funkcji  $y, g, h$ :

$$f(W_1, W_2, W_3) = y\left(W_1g(W_2(h(W_3)))\right).$$

Jeśli  $L(w)$  jest funkcją straty naszego modelu, to należy obliczyć pochodną funkcji straty względem wyjścia funkcji  $y, g, h$ , a następnie pochodne wyjść z tych funkcji względem ich parametrów wagowych. Łącząc wszystkie te elementy zgodnie ze wzorem (15) z **Twierdzeniem 4** otrzymane zostaje:

$$\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial g} \cdot \frac{\partial g}{\partial h} \cdot \frac{\partial h}{\partial W_3}, \quad \frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial g} \cdot \frac{\partial g}{\partial W_2}, \quad \frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial W_1}.$$

W algorytmie propagacji wstecznej, te pochodne są obliczane w odwrotnej kolejności, zaczynając od  $W_3$ , ponieważ  $\frac{\partial L}{\partial y}$  jest najpierw wykorzystywane do obliczenia  $\frac{\partial L}{\partial W_3}$ , a następnie kolejne pochodne są obliczane wstecz do warstw wejściowych.

### **Algorytm propagacji w przód**

Propagacja w przód (ang. forward propagation) jest drugim sposobem przetwarzania danych wejściowych przez sieć neuronową w celu uzyskania wyników predykcji. Podczas tego procesu dane są przekazywane od wejścia sieci, przez wszystkie warstwy, aż do warstwy wyjściowej, gdzie generowane są predykcje. Wartość aktywacji  $\sigma_j$  dla  $j$ -tego neuronu w warstwie ukrytej można obliczyć z poniższego wzoru:

$$\sigma_j^{(i)} = f \left( \sum_{k=1}^m w_{kj} \cdot x_k^{(i)} + b_j \right), \quad (21)$$

gdzie  $w_{ij}$  to waga między  $i$ -tym neuronem wejściowym a  $j$ -tym neuronem w warstwie ukrytej,  $x_k^{(i)}$  to  $k$ -ty element wektora danych wejściowych dla  $i$ -tej próbki,  $b_j$  to bias  $j$ -tego neuronu w warstwie ukrytej oraz  $f$  to funkcja aktywacji, która wprowadza nieliniowość do modelu. Wynik predykcji jest obliczany jako kombinacja liniowa aktywacji neuronów w warstwie ukrytej z odpowiednimi wagami, która jest ponownie poddawana funkcji aktywacji  $g$  w następujący sposób:

$$\hat{y}_j^{(i)} = g \left( \sum_{l=1}^p w_{kl} \cdot \sigma_l^{(i)} + c_k \right), \quad (22)$$

gdzie  $w_{kl}$  to waga między  $k$ -tym neuronem w warstwie ukrytej a  $l$ -tym neuronem w warstwie wyjściowej,  $\sigma_l^{(i)}$  to aktywacja  $l$ -tego neuronu w warstwie ukrytej dla  $i$ -tej próbki,  $c_k$  to bias  $k$ -tego neuronu w warstwie wyjściowej oraz  $g$  to funkcja

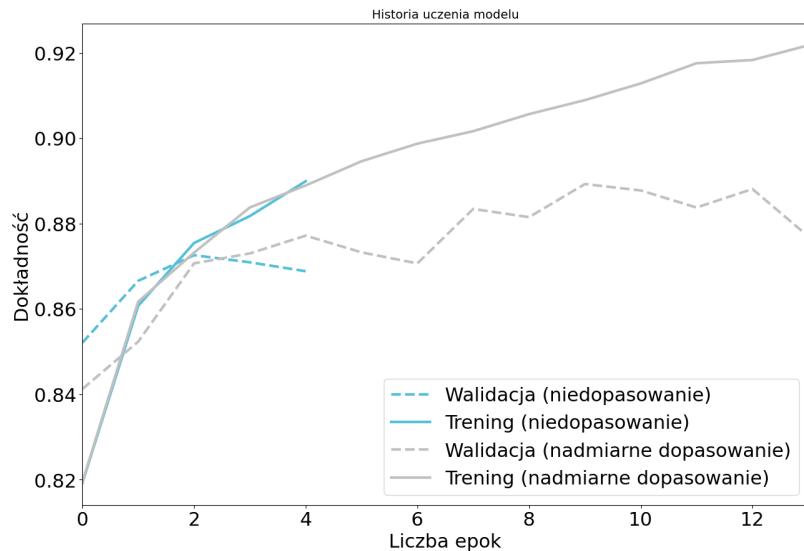
aktywacji na wyjściu, której wybór zależy od problemu, np. funkcja softmax w przypadku klasyfikacji wieloklasowej.

## 1.6. Trenowanie i testowanie modelu

W tym podrozdziale omówiono dwa kluczowe etapy uczenia głębokiego: trenowanie i testowanie modelu. W części dotyczącej trenowania poruszono problemy nadmiernego i niedostatecznego dopasowania oraz kwestie dostrajania hiperparametrów. W części testowania opisano metody oceny wydajności modelu.

### Etap trenowania

Celem etapu trenowania jest maksymalizacja zdolności modelu do przewidywania i generalizacji. Pod tym pojęciem kryje się zdolność modelu do zachowania wysokiej wydajności nie tylko względem danych, na których był trenowany, ale również na nowych, nieznanych przykładach. W związku z tym pojawiają się dwa zasadnicze wyzwania: nadmierne dopasowanie i zbyt słabe dopasowanie (patrz rys. 9).



Rysunek 9: Dokładność dla modelu nadmiernie i niedopasowanego.

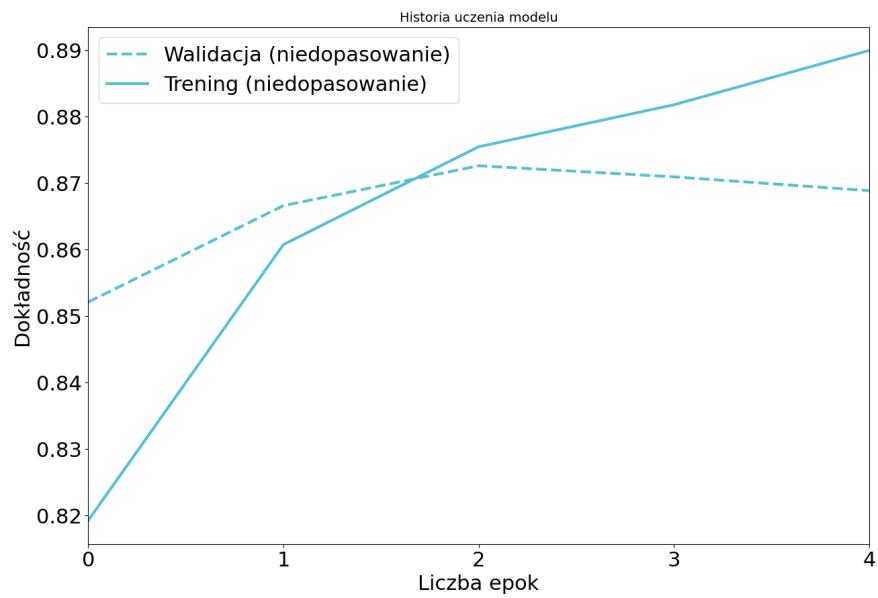
**Nadmierne dopasowanie** (ang. overfitting) często określane mianem przetrenowania oznacza zbyt dokładne dopasowanie modelu uczenia maszynowego do zestawu danych wykorzystanych do trenowania. Efektem jest utarta zdolność do uogólniania i właściwego reagowania na nowe informacje.

**Niedostateczne dopasowanie** (ang. underfitting) oznacza, że model jest zbyt uproszczony, aby uchwycić istotne wzorce i zależności w danych, co prowadzi do słabych wyników zarówno na treningu, jak i w praktycznym zastosowaniu.

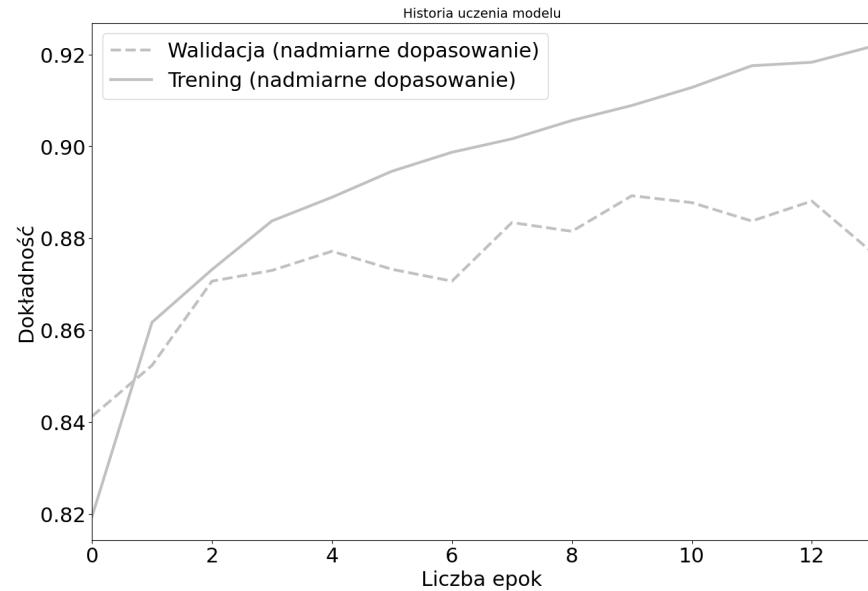
Aby zmniejszyć negatywny wpływ nadmiernego dopasowania i niedostatecznego dopasowania na ogólną dokładność modelu, istotne jest dokładne dostrojenie hiperparametrów. Ważne jest odpowiednie wyważenie procesu optymalizacji i generalizacji, które kontrolują działanie modelu podczas uczenia. Można to osiągnąć poprzez staranną regulację następujące atrybutów modelu:

1. liczba iteracji,
2. funkcja aktywacji,
3. algorytm optymalizujący.

Termin **epoka** oznacza pojedynczą iterację algorytmu uczenia się modelu na zbiorze treningowym. Liczbę epok ustala się eksperymentalnie na podstawie obserwacji wartości błędu obliczanego dla zbioru treningowego i dla zbioru walidacyjnego. Zbyt mała liczba iteracji może skutkować niedostatecznym dopasowaniem (patrz rys. 10). Natomiast zbyt wiele iteracji może prowadzić do nadmiernego dopasowania, szczególnie jeśli błędy na zbiorze treningowym stale maleją, podczas gdy błędy na zbiorze walidacyjnym zaczynają rosnąć (patrz rys. 11).



Rysunek 10: Porównanie dokładności treningu i walidacji dla modelu niedopasowanego.



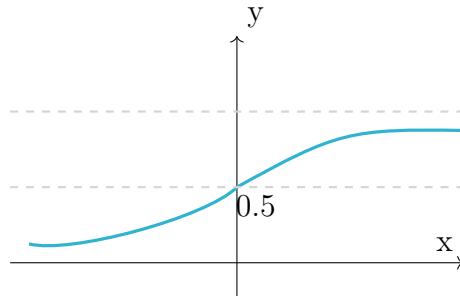
Rysunek 11: Porównanie dokładności treningu i walidacji dla modelu nadmiernie dopasowanego.

**Funkcje aktywacji** decydują o tym, czy wartość całkowitego pobudzenia neuronu jest wystarczająco duża, aby przekazać ją do kolejnych warstw modelu. Wartość tych funkcji zależy zatem od wartości wag na wejściu perceptronu (patrz rys. 4). Dzięki nim model jest w stanie reprezentować nieliniowe zależności, co jest kluczowe dla rozwiązywania złożonych problemów. Funkcje te charakteryzują się monotonicznością, co oznacza, że ich wartość stale rośnie lub maleje oraz mają skończony zakres. Ponadto posiadają one pochodne, co umożliwia stosowanie algorytmów optymalizacyjnych bazujących na gradientach. W kontekście działania konwolucyjnych sieci neuronowych kluczowy jest wybór najbardziej odpowiedniej funkcji aktywacji [43]. Najbardziej powszechnie używane funkcje aktywacji to:

1. Sigmoid - rodzaj funkcji logistycznych czyli funkcja sigmoidalna zdefiniowana jest następującym wzorem:

$$\text{sigmoid}(z) = \frac{1}{1 + e^{-z}}. \quad (23)$$

Wykres tej funkcji jest następujący:



**Przykład 1.10.** Dany jest następujący wektor wejściowy  $z = [-1, 0.5, 3, -2]$ . Korzystając ze wzoru 23, wartości tego wektora dla funkcji sigmoidalnej obliczono w następujący sposób:

$$\text{sigmoid}(-1.0) = \frac{1}{1 + e^{-(-1.0)}} = \frac{1}{1 + 2.71828183^{-1.0}} \approx 0.73105858,$$

$$\text{sigmoid}(0.5) = \frac{1}{1 + e^{-0.5}} = \frac{1}{1 + 0.60653066} \approx 0.62245933,$$

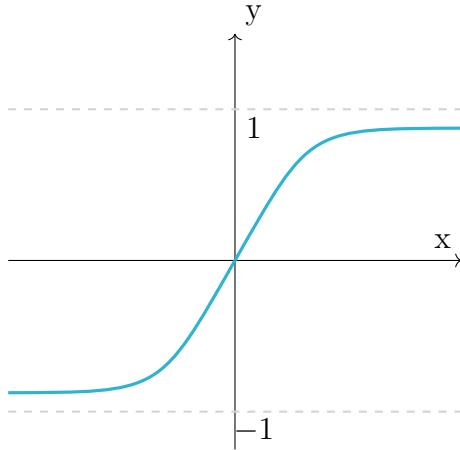
$$\text{sigmoid}(3.0) = \frac{1}{1 + e^{-3.0}} = \frac{1}{1 + 0.04978707} \approx 0.95257413,$$

$$\text{sigmoid}(-2.0) = \frac{1}{1 + e^{-(2.0)}} = \frac{1}{1 + 2.71828183^{-2.0}} \approx 0.88079708.$$

2. Tanh - rodzaj funkcji hiperbolicznej czyli tangens hiperboliczny zdefiniowany jest następującym wzorem:

$$\text{Tanh}(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \quad (24)$$

Wykres tej funkcji jest następujący:



**Przykład 1.11.** Dany jest następujący wektor wejściowy  $z = [-1, 0.5, 3, -2]$ . Korzystając ze wzoru 24, wartości tego wektora dla funkcji tangens hiperboliczny obliczono w następujący sposób:

$$\tanh(-1.0) = \frac{e^{-1.0} - e^{1.0}}{e^{-1.0} + e^{1.0}} = \frac{0.36787944 - 2.71828183}{0.36787944 + 2.71828183} \approx -0.76159416,$$

$$\tanh(0.5) = \frac{e^{0.5} - e^{-0.5}}{e^{0.5} + e^{-0.5}} = \frac{1.64872127 - 0.60653066}{1.64872127 + 0.60653066} \approx 0.46211716,$$

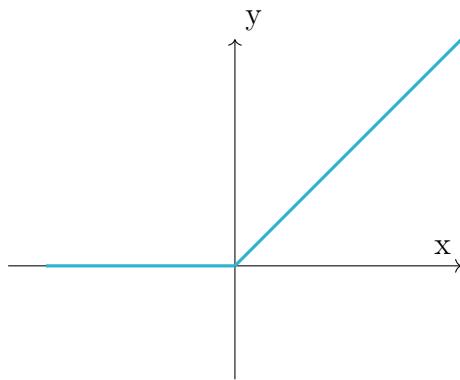
$$\tanh(3.0) = \frac{e^{3.0} - e^{-3.0}}{e^{3.0} + e^{-3.0}} = \frac{20.08553692 - 0.04978707}{20.08553692 + 0.04978707} \approx 0.99505475,$$

$$\tanh(-2.0) = \frac{e^{-2.0} - e^{2.0}}{e^{-2.0} + e^{2.0}} = \frac{0.13533528 - 7.3890561}{0.13533528 + 7.3890561} \approx -0.96402758.$$

3. ReLu function - skorygowana liniowa funkcja aktywacji jest funkcją liniową, która wyprowadza dane wejściowe bezpośrednio, jeśli są one dodatnie, w przeciwnym razie wyprowadza zero:

$$ReLU(z) = \max(0, z) \quad (25)$$

Wykres tej funkcji jest następujący:



**Przykład 1.12.** Dany jest następujący wektor wejściowy  $z = [-1, 0.5, 3, -2]$ . Korzystając ze wzoru 25, wartości tego wektora dla funkcji ReLU obliczono w następujący sposób:

$$ReLU(-1.0) = \max(0, -1.0) = 0,$$

$$ReLU(0.5) = \max(0, 0.5) = 0.5,$$

$$ReLU(3.0) = \max(0, 3.0) = 3.0,$$

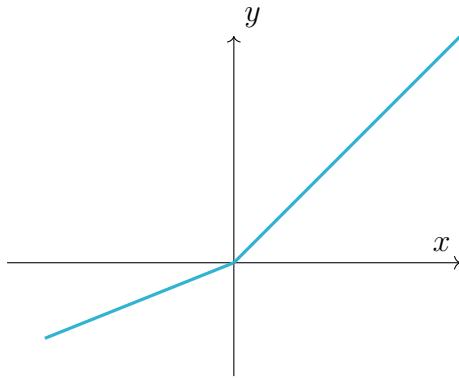
$$ReLU(-2.0) = \max(0, -2.0) = 0.$$

4. Leaky ReLU - to zmodyfikowana wersja klasycznej funkcji aktywacji ReLU. W przeciwieństwie do ReLU, która ma pochodną równą zero dla ujemnych wejść, Leaky ReLU ma małe nachylenie do ujemnego wejścia. To oznacza, że pochodna nigdy nie jest dokładnie równa zeru. Dzięki temu zmniejsza się występowanie *cichych neuronów*, czyli tych, które nie uczą się w trakcie procesu

uczenia [53], co pozwala na zachowanie gradientu i pomaga rozwiązać problem związaną z brakiem uczenia się neuronów po tym, jak funkcja ReLU wchodzi w ujemny przedział. Wzór jest następujący:

$$L_{ReLU} = \max(0.1 \cdot x, x). \quad (26)$$

Wykres tej funkcji jest następujący:



**Przykład 1.13.** Dany jest następujący wektor wejściowy  $z = [-1, 0.5, 3, -2]$ . Korzystając ze wzoru 26, wartości tego wektora dla funkcji Leaky ReLU obliczono w następujący sposób:

$$L_{ReLU}(-1.0) = 0.01 \cdot (-1.0) = -0.01,$$

$$L_{ReLU}(0.5) = 0.5,$$

$$L_{ReLU}(3.0) = 3.0,$$

$$L_{ReLU}(-2.0) = 0.01 \cdot (-2.0) = -0.02.$$

5. SOFTMAX - znormalizowana funkcja wykładnicza zadana następującym wzorem:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}, \quad (27)$$

gdzie  $K$  oznacza liczbę klas.

**Przykład 1.14.** Dany jest następujący wektor wejściowy  $z = [-1, 0.5, 3, -2]$ .

Korzystając z wzoru 27 wartości tego wektora dla funkcji softmax obliczono w następujący sposób:

$$\begin{aligned} \text{softmax}(z_i) &= \left[ \frac{e^{-1.0}}{\sum_{i=1}^5 e^{z_i}}, \frac{e^{0.5}}{\sum_{i=1}^5 e^{z_i}}, \frac{e^{3.0}}{\sum_{i=1}^5 e^{z_i}}, \frac{e^{-2.0}}{\sum_{i=1}^5 e^{z_i}} \right] \\ &= \left[ \frac{0.36787944}{29.62652801}, \frac{1.64872127}{29.62652801}, \frac{20.08553692}{29.62652801}, \frac{0.13533528}{29.62652801} \right] \\ &= [0.01242073, 0.05574386, 0.67896752, 0.00412800]. \end{aligned}$$

Jak przedstawiono w tabeli 1, wyniki uzyskane w powyższych przykładach pokazują, że wybór funkcji aktywacyjnej powinien być świadomy i dostosowany do specyfiki zadania, struktury danych oraz oczekiwaniń względem modelu. Wybór ten powinien uwzględniać równowagę między zdolnością do uczenia się a ryzykiem nadmiernego dopasowania, co ma kluczowe znaczenie dla osiągnięcia optymalnej wydajności modelu.

$z_i$	funkcja aktywacyjna				
	sigmoid	Tanh	ReLU	$L_{ReLU}$	softmax
-1	0.7311	-0.7616	0	-0.01	0.0124
0.5	0.6225	0.4621	0.5	0.5	0.0557
3	0.9526	0.9951	3.0	3.0	0.6790
-2	0.8808	-0.9640	0	-0.02	0.0041

Tabela 1: Porównanie wyników uzyskanych przy zastosowaniu różnych funkcji aktywacyjnych w sieciach neuronowych.

**Optymalizacja** w procesie trenowania sieci neuronowych ma za zadanie minimalizować funkcję kosztu, czyli różnicę między przewidywaniami modelu a rzeczywistymi wartościami. W tej pracy opisane zostały najważniejsze algorytmy optymalizacyjne, takie jak gradient prosty, Adam, RMSprop oraz ich warianty, na podstawie artykułu przeglądowego [38].

**Metoda gradientu prostego** jest iteracyjnym algorytmem wyszukiwania minimum zadanej funkcji straty  $L(w)$ . Algorytm działania tej metody można zapisać następująco:

1. wybranie rozwiązania początkowego  $w_0 = 0$ ,
2. wylosowanie  $i \in \{1, \dots, n\}$  :
  - (a) obliczenie gradientu elementu  $l_i$  w punkcie  $w_{k-1}, \nabla_w L(w_{k-1})$ ,
  - (b) przesunięcie wartości parametrów modelu w kierunku przeciwnym do gradientu funkcji celu:  $w_k = w_{k-1} - \alpha_k \nabla_w L(w_{k-1})$ .

Ogólnie proces minimalizacji funkcji można zapisać następująco:

$$w := w - \alpha \nabla_w L(w), \quad (28)$$

gdzie  $\alpha$  oznacza współczynnik uczenia (ang. learning rate), a  $\nabla_w L(w)$  to gradient funkcji celu względem parametrów  $w$ .

**Przykład 1.15.** W celu znalezienia minimum funkcji, obliczony zostanie gradient względem  $w$  i wykonana aktualizacja  $w$  za pomocą metody gradientu prostego. Założono, że początkowa wartość  $w$  wynosi 0, a współczynnik uczenia  $\alpha$  jest równy 0.01. Funkcja kosztu  $L(w)$  wyraża się wzorem:

$$L(w) = \sum_{i=1}^2 (wx_i - y_i)^2.$$

Zatem, gradient funkcji kosztu wynosi:

$$\begin{aligned} \nabla_w L(w) &= \frac{dL}{dw} [(w \cdot 1 - 2)^2 + (w \cdot 2 - 3)^2] = \\ &= 2(w \cdot 1 - 2) \cdot 1 + 2(w \cdot 2 - 3) \cdot 2 = 2w - 4 + 8w - 12 = 10w - 16. \end{aligned}$$

Aktualizacja wektora wag  $w$  wygląda następująco:

$$w - \alpha \nabla_w L(w) = 0 - 0.01 \cdot (10 \cdot 0 - 16) = 0 + 0.16 = 0.16.$$

Po wykonaniu jednej iteracji metody gradientu prostego, nowa wartość parametru  $w$  jest równa 0.16.

Poza metodą gradientu prostego, która jest najczęściej stosowana, istnieją również różne jego modyfikacje oraz inne algorytmy optymalizacji mające na celu poprawę działania sieci neuronowych. Do najważniejszych z nich należą:

1. stochastyczny spadek gradientowy (ang. Stochastic Gradient Descent, SGD)  
- w tym algorytmie aktualizacja wag jest dokonywana dla każdej próbki (lub małego zbioru próbek, znanego jako mini-batch):

$$w := w - \eta \cdot \nabla_w L(w; x^{(i)}; y^{(i)})$$

gdzie  $x^{(i)}$  i  $y^{(i)}$  to pojedyncza próbka danych wejściowych i jej etykieta,

2. Momentum - jest to modyfikacja SGD poprzez dodanie terminu "pędu"  $v$ :

$$v := \gamma v - \eta \nabla_w L(w), \quad w := w + v$$

gdzie  $\gamma$  to współczynnik pędu, zazwyczaj ustalany na wartość około 0.9,

3. Adagrad - modyfikuje szybkość uczenia się dla każdego parametru:

$$w := w - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot \nabla_w L(w),$$

gdzie  $G_{t,ii}$  to suma kwadratów gradientów do czasu  $t$  dla parametru  $i$ , a  $\epsilon$  to mała stała zapobiegająca dzieleniu przez zero,

4. RMSprop jest modyfikacją Adagrad, algorytm ten wykorzystuje średnie ruchome kwadratów gradientów:

$$E[g^2]_t := \beta E[g^2]_{t-1} + (1 - \beta) g_t^2, \quad w := w - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot \nabla_w L(w),$$

5. Adam - algorytm ten łączy pomysły Momentum i RMSprop:

$$\begin{aligned} m_t &:= \beta_1 m_{t-1} + (1 - \beta_1) \nabla_w L(w), \quad v_t := \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_w L(w))^2, \\ \hat{m}_t &:= \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t := \frac{v_t}{1 - \beta_2^t} \quad w := w - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t. \end{aligned}$$

Gdzie wartości  $\eta$ ,  $\beta$ ,  $\beta_1$ ,  $\beta_2$ , i  $\epsilon$  są hiperparametrami, które należy dostosować do konkretnego problemu.

## Etap testowanie

Etap testowania w uczeniu maszynowym ma na celu nie tylko sprawdzenie jak dobrze działa metoda, ale przede wszystkim ustalenie, które cechy mają istotny wpływ na wyniki modelu, a które mogą być pominięte lub zmodyfikowane. W procesie weryfikacji i oceny wykorzystanych zostanie kilka ważnych metryk, które odzwierciedlają jakość modelu i pozwalają uzyskać pełniejszy obraz wydajności modelu.

Istnieją różne miary podobieństwa, które pozwalają na dokładne określenie, jak bardzo dwa obiekty są do siebie podobne. W celu oceny podobieństwa między obiektami często wykorzystuje się odległości między nimi, które są obliczane za pomocą różnych miar dystansu, ale także wiele innych miar podobieństw nie będących metrykami (odległościami).

**Definicja 10.** Funkcję  $D : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$  przyporządkowującą parze punktów wartość rzeczywistą zwaną odlegością tych punktów, nazywa się **metryką** (odlegością), jeśli spełnia następujące warunki:

1.  $\forall_{x,y \in \mathbb{R}^n} : D(x, y) = 0 \iff x = y,$
2.  $\forall_{x,y \in \mathbb{R}^n} : D(x, y) = D(y, x),$
3.  $\forall_{x,y,z \in \mathbb{R}^n} : D(x, y) + D(y, z) \leq D(x, z).$

Metryka, czyli miara odległości lub po prostu odległość, daje możliwość skonstruowania reprezentacji miar podobieństw między obiektami zwanej **macierzą podobieństw**. Niech  $X_i, Y_j \in \mathbb{R}^n, i, j = 1, \dots, n$  wówczas następujące odległości są dane wzorami:

- odległość miejska (ang. Manhattan, city block):

$$D(X_i, X_j) = \sum_{k=1}^n |X_{i,k} - X_{j,k}|, \quad (29)$$

- odległość Minkowskiego:

$$D(X_i, X_j) = \sqrt{\left( \sum_{k=1}^n |X_{i,k} - X_{j,k}| \right)^r}, \quad (30)$$

- odległość Euklidesowa (ang. Euclidean):

$$D(X_i, X_j) = \sqrt{\sum_{k=1}^n (X_{i,k} - X_{j,k})^2}, \quad (31)$$

- odległość Czebyszewa:

$$D(X_i, X_j) = \max_{1 \leq k \leq n} |X_{i,k} - X_{j,k}|, \quad (32)$$

- podobieństwo Jaccarda:

$$S(X_i, X_j) = \frac{\sum_{k=1}^n X_{i,k} \cdot X_{j,k}}{\sqrt{\sum_{k=1}^n X_{i,k}^2} + \sqrt{\sum_{k=1}^n X_{j,k}^2} - \sum_{k=1}^n X_{i,k} \cdot X_{j,k}}, \quad (33)$$

- podobieństwo cosinusów:

$$S(X_i, X_j) = \frac{\sum_{k=1}^n X_{i,k} \cdot X_{j,k}}{\sqrt{\sum_{k=1}^n X_{i,k}^2} \sqrt{\sum_{k=1}^n X_{j,k}^2}}. \quad (34)$$

**Przykład 1.16.** Dany jest zbiór punktów  $A(1, 2), B(3, 4), C(5, 6)$ . Najpierw została obliczona odległość między poszczególnymi obiektami dla kolejnych metryk, a następnie skonstruowano macierz podobieństw dla uzyskanych wyników (patrz tab.2).

1. odległość miejska:

$$D(A, B) = |x_A - x_B| + |y_A - y_B| = |1 - 3| + |2 - 4| = 2 + 2 = 4,$$

$$D(A, C) = |x_A - x_C| + |y_A - y_C| = |1 - 5| + |2 - 6| = 4 + 4 = 8,$$

$$D(B, C) = |x_B - x_C| + |y_B - y_C| = |3 - 5| + |4 - 6| = 2 + 2 = 4,$$

2. odległość Minkowskiego dla  $r = 3$ :

$$D(A, B) = \sqrt[3]{(x_A - x_B)^3 + (y_A - y_B)^3} = \sqrt[3]{(1 - 3)^3 + (2 - 4)^3} = \sqrt[3]{-16},$$

$$D(A, C) = \sqrt[3]{(x_A - x_C)^3 + (y_A - y_C)^3} = \sqrt[3]{(1 - 5)^3 + (2 - 6)^3} = \sqrt[3]{-128},$$

$$D(B, C) = \sqrt[3]{(x_B - x_C)^3 + (y_B - y_C)^3} = \sqrt[3]{(3 - 5)^3 + (4 - 6)^3} = \sqrt[3]{-16},$$

3. odległość Euklidesowa:

$$D(A, B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} = \sqrt{(1 - 3)^2 + (2 - 4)^2} = \sqrt{8},$$

$$D(A, C) = \sqrt{(x_A - x_C)^2 + (y_A - y_C)^2} = \sqrt{(1 - 5)^2 + (2 - 6)^2} = \sqrt{32},$$

$$D(B, C) = \sqrt{(x_B - x_C)^2 + (y_B - y_C)^2} = \sqrt{(3 - 5)^2 + (4 - 6)^2} = \sqrt{8},$$

4. Odległość Czebyszewa:

$$D(A, B) = \max\{|x_A - x_B|, |y_A - y_B|\} = \max\{2, 2\} = 2,$$

$$D(A, C) = \max\{|x_A - x_C|, |y_A - y_C|\} = \max\{4, 4\} = 4,$$

$$D(B, C) = \max\{|x_B - x_C|, |y_B - y_C|\} = \max\{2, 2\} = 2.$$

Porównanie macierzy podobieństw dla poszczególnych miar odległości											
miejiska			Minkowskiego			Euklidesowa			Czebyszewa		
A	B	C	A	B	C	A	B	C	A	B	C
A	0	4	8	A	0	$\sqrt[3]{-16}$	$\sqrt[3]{-128}$	A	0	$\sqrt{8}$	$\sqrt{32}$
B	4	0	4	B	$\sqrt[3]{-16}$	0	$\sqrt[3]{-16}$	B	$\sqrt{8}$	0	$\sqrt{8}$
C	8	4	0	C	$\sqrt[3]{-128}$	$\sqrt[3]{-16}$	0	C	$\sqrt{32}$	$\sqrt{8}$	0

Tabela 2: Macierze podobieństw zawierające odległości obliczone za pomocą różnych miar: miejskiej, Minkowskiego, Euklidesowej i Czebyszewa.

### Opis metody oceny klasyfikatora

Ocena wydajności klasyfikatora to proces analizy jakości działania klasyfikatora w rozpoznawaniu i przypisywaniu danych do odpowiednich klas.

Istnieje wiele metod oceny wydajności klasyfikatora, oto kilka z najczęściej używanych:

1. macierz pomyłek (ang. confusion matrix) - przedstawia liczbę poprawnie sklasyfikowanych przykładów oraz liczby błędów w każdej klasie, a przedstawić to można przy pomocy wzoru:

$$A_{conf} = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}, \quad (35)$$

gdzie:

- $TP$  Prawdziwie Pozytywne (ang. True Positive) to liczba poprawnie sklasyfikowanych pozytywnych przykładów,
- $FP$  Fałszywie Pozytywne (ang. False Positive) to liczba błędnie sklasyfikowanych pozytywnych przykładów,
- $FN$  Fałszywie Negatywne (ang. False Negative) to liczba błędnie sklasyfikowanych negatywnych przykładów,
- $TN$  Prawdziwie Negatywne (ang. True Negative) to liczba poprawnie sklasyfikowanych negatywnych przykładów.

2. dokładność (ang. accuracy): Jest to procent poprawnie sklasyfikowanych przykładów w stosunku do całkowitej liczby przykładów, wyrażana wzorem:

$$\text{Dokładność} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (36)$$

3. precyza (ang. precision) czyli jak wiele z pozytywnie sklasyfikowanych przykładów faktycznie należy do danej klasy, obliczana za pomocą wzoru:

$$\text{Precyzja} = \frac{TP}{TP + FP}. \quad (37)$$

4. czułość (ang. recall) liczba jak wiele z rzeczywiście pozytywnych przykładów zostało poprawnie sklasyfikowanych jako pozytywne, można ją obliczyć ze wzoru:

$$\text{Czułość} = \frac{TP}{TP + FN}. \quad (38)$$

**Przykład 1.17.** Macierz pomyłek przedstawia wyniki modelu binarnego, który przewiduje dwie klasy: 1 pozytywna i 0 negatywna.

$$A_{conf} = \begin{bmatrix} 30 & 10 \\ 5 & 55 \end{bmatrix},$$

Na podstawie tych wyników obliczona została dokładność, precyza i czułość:

$$\text{Dokładność} = \frac{TP + TN}{TP + FP + FN + TN} = \frac{30 + 55}{30 + 5 + 10 + 55} = 0.85,$$

$$\text{Precyza} = \frac{TP}{TP + FP} = \frac{30}{30 + 5} = 0.857, \quad \text{Czułość} = \frac{TP}{TP + FN} = \frac{30}{40} = 0.75.$$

Ogólna dokładność klasyfikacji wynosi 85% oraz 87% przypadków zaklasyfikowanych przez model jako pozytywne jest rzeczywiście pozytywnych. Model poprawnie identyfikuje 75% rzeczywistych pozytywnych przypadków.

Wizualnie działanie macierzy pomyłek może być zilustrowany za pomocą schematu (patrz rys. 12).

		Fałsz	Prawda
		wejście	
Fałsz	wejście		
	predykcja	nie torus	torus
Prawda	wejście		
	predykcja	nie torus	torus

Rysunek 12: Przykład macierzy pomyłek prezentująca porównanie rzeczywistych kategorii z predykcjami modelu klasyfikacyjni binarnej.

## **2. Zastosowanie konwolucyjnych sieci neuronowych w klasyfikacji obrazów na urządzeniach mobilnych**

Rozdział drugi dotyczy zastosowania konwolucyjnych sieci neuronowych w klasyfikacji obrazów na urządzeniach mobilnych. Zawiera on opis baz danych, charakterystykę modeli z Keras oraz omówienie technik głębokiego dopasowywania wzorców i transferu wiedzy.

Konwolucyjne sieci neuronowe posiadają imponujące zdolności w zakresie rozpoznawania obrazów. Wykorzystanie tych sieci do ekstrakcji cech staje się szczególnie skuteczne, gdy są one trenowane na dużych zbiorach danych, takich jak ImageNet. Jednakże, architektury CNN ze względu na swoją wielkość wymagają znaczących zasobów obliczeniowych, co sprawia, że ich zastosowanie na urządzeniach mobilnych jest często nieefektywne ze względu na ograniczone moce przetwarzania tych urządzeń. W odpowiedzi na te ograniczenia niniejsza praca proponuje rozwiązania dostosowane do urządzeń mobilnych, które zachowują wysoką dokładność klasyfikacji obrazów, a są to głębokie dopasowywanie wzorców i transfer wiedzy.

Rozkwit badań nad sztucznymi sieciami neuronowymi trenowanymi na obrazach rozpoczął się w 2010 roku kiedy to ogłoszono konkurs - **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** (<https://www.image-net.org/challenges/LSVRC>). Był to projekt badawczy skierowany do naukowców i inżynierów z całego świata, a celem tego przedsięwzięcia było zbudowanie systemu, który może automatycznie rozpoznawać i klasyfikować obrazy w różne kategorie, na poziomie zbliżonym do człowieka. Konkurs ten składał się z kilku zadań, takich jak klasyfikacja obrazów, detekcja obiektów i lokalizacja obiektów. Jednym z ważniejszych przełomów była edycja konkursu z 2012 roku, w której zespół z Uniwersytetu

Toronto, kierowany przez *Alexa Krizhevsky'ego*, zaprezentował model głębszej sieci neuronowej o nazwie AlexNet, który uzyskał dokładność na poziomie 83,6%. Natomiast zwycięzca konkursu w 2015 roku uzyskał dokładność na poziomie 96,4%, a klasyfikacja obrazów konkursu ILSVRC została uznana za problem, który został całkowicie rozwiązany.

## 2.1. Bazy danych

W tym podrozdziale znajduje się opis baz danych, które zostały wykorzystywane do testowania modeli wybranych technik rozpoznawania obrazów. Badanie tych technik na różnorodnych zbiorach danych stanowi ważny aspekt w procesie oceny ich efektywności w różnych kontekstach. Dlatego też wybrano dwie bazy danych do testowania: **ImageNette** oraz zbiór autorskich zdjęć reprezentujących 5 wybranych klas obiektów.

Wstępne ustalenie wag wektora  $w$  oparte zostały na rozległej ontologii obrazów **ImageNet**, która korzysta z hierarchicznej struktury **WordNet**. Głównym atutem WordNet jest jego ontologia pojęć [15], gdzie każde znaczące pojęcie opisane jest przez wiele słów lub fraz słownych (tak zwanych synsetów). Ta szczególna baza danych, znana jako ImageNet, została wprowadzona do użytku w 2009 roku, z ambitnym celem odwzorowania większości spośród 80 000 synsetów WordNet, każdy poprzez średnio 500-1000 wyselekcyjowanych obrazów o wysokiej rozdzielczości. Wówczas ImageNet wyróżniał się na tle innych zbiorów danych obrazów (Caltech101/256, PASCAL) swoją skalą, różnorodnością oraz precyzją, ustanawiając nowy standard w dziedzinie przetwarzania obrazów [10]. W 2010 roku opublikowano artykuł, który zaprezentował wyniki badań nad kategoryzacją obrazów na szeroką skalę, włączając w to serię zaawansowanych eksperymentów klasyfikacyjnych obejmujących ponad 10 000 kategorii obrazów. W publikacji podkreślono znaczenie wyzwań obliczeniowych w procesie tworzenia algorytmów. Dodatkowo, autorzy artykułu zwrócili uwagę na nieoczekiwane mocną korelację między strukturą WordNet, pierwotnie stworzoną do analizy językowej, a problemami napotykanyimi przy wizualnej kategoryzacji ob-

razów. Wyniki badań sugerowały, że wykorzystanie hierarchii semantycznej może znacząco przyczynić się do usprawnienia procesu klasyfikacji [11].

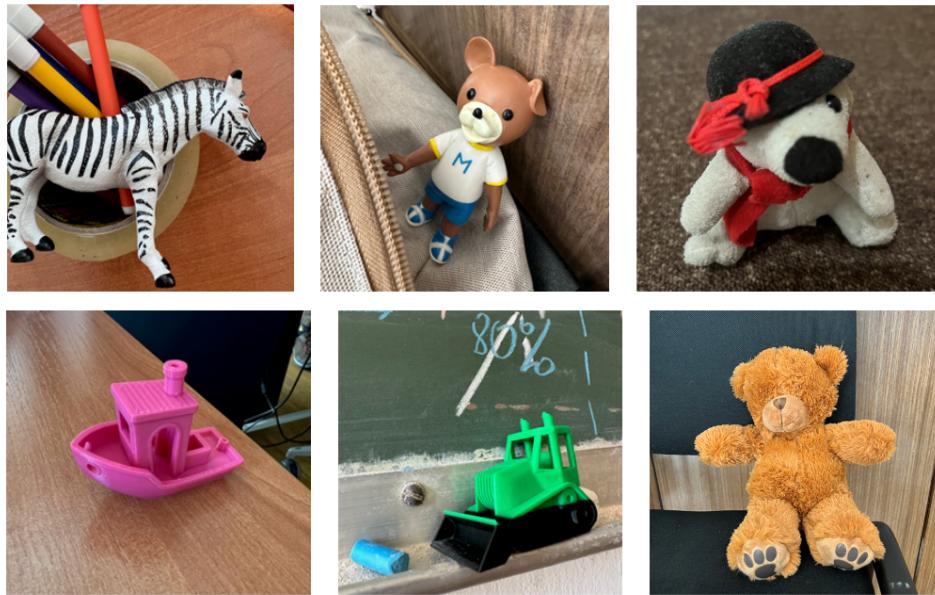
**ImageNette** to uproszczona wersja opisanego powyżej zbioru danych ImageNet. Dziedzicząc kluczowe elementy, które czynią ImageNet cennym narzędziem w dziedzinie wizji komputerowej podzióbór ten umożliwia szybsze i łatwiejsze przeprowadzanie badań oraz eksperymentów. Składa się on z 10 kategorii obiektów, wśród których znajdują się następujące klasy: liny, springer spaniel angielski, odtwarzacz kasetowy, piła łańcuchowa, kościół, waltornia, śmieciarka, dystrybutor paliwa, piłka golfowa, spadochron, co zapewnia wystarczającą różnorodność i poziom trudności niezbędny do testowania i ulepszania metod klasyfikacji obrazów. W kontekście opracowywania rozwiązań do rozpoznawania kategorii przeznaczonych dla aplikacji mobilnych, zdecydowano się rozważać wyłącznie obrazy przedstawiające przedmioty. W związku z tym wykluczono kategorie nienależące do przedmiotów, na przykład budynki, co pozwoli na dokładniejszą ocenę wydajności modeli w bardziej kontrolowanych warunkach. Poniżej (patrz rys. 13) przedstawiono pięć klas wybranych do dalszych badań jako zbiór treningowy i testowy, a są to: dystrybutor paliwa, piłka golfowa, odtwarzacz kasetowy, piła łańcuchowa i waltornia.



Rysunek 13: Graficzna reprezentacja obiektów zawartych w bazie danych ImageNette.

**Autorski zbiór danych** pozwala na zrozumienie, jak wybrane techniki rozpoznawania obrazów sprawdzają się w praktyce, w warunkach zbliżonych do tych, z jakimi mamy do czynienia w scenariuszach aplikacyjnych. Dzięki temu, badania mogą dostarczyć cennych wskazówek dotyczących optymalnego wykorzystania tech-

nik TL oraz DTM w praktycznych zastosowaniach. Zbiór danych zawiera kolekcję 300 obrazów, należących do jednej z sześciu klas: zebra, Uszatek, miś polarny, łódeczka, buldożer i miś. Poniżej zaprezentowano przykładowe obrazy reprezentujące każdą z klas (patrz rys. 14). Zbiór danych został udostępniony publicznie poprzez repozytorium na GitHubie (<https://github.com/iitis/EduToyz>). Autorski zbiór danych zawiera przedmioty przeznaczone do wsparcia edukacji najmłodszych dzieci, jednakże są na tyle ogólne, że mogą być stosowane w szerszym kontekście edukacyjnym.



Rysunek 14: Graficzna reprezentacja obiektów zawartych w autorskiej bazie danych.

### **Zbiór treningowy, testowy i walidacyjny**

Zbiór danych wejściowych zazwyczaj dzielony jest na trzy podzbiory: zbiór treningowy, zbiór testowy oraz zbiór walidacyjny.

**Zbiór treningowy** to zbiór danych, który jest używany do trenowania modelu ML. Zawiera przykłady wejściowe wraz z odpowiadającymi im wynikami (lub etykietami).

tami), które model uczy się przewidywać. Podczas procesu treningu, model analizuje te dane, ucząc się wzorców i zależności między danymi wejściowymi.

**Zbiór testowy** to zbiór danych, który nie jest używany podczas trenowania modelu, ale służy do oceny jego wydajności po zakończeniu treningu. Pozwala to na sprawdzenie, jak model radzi sobie z nowymi, nieznanymi danymi.

**Zbiór walidacyjny** to dodatkowy podzbiór danych, który jest używany podczas procesu trenowania do regularnego sprawdzania wydajności modelu i dostrajania hiperparametrów. Pomaga to w zapobieganiu nadmierнемu dopasowaniu modelu do danych treningowych, umożliwiając jednocześnie jego optymalizację.

Wybór liczby elementów w zbiorze treningowym bezpośrednio oddziałuje na dokładność modelu. W niniejszej pracy zgodnie z ideą uczenia metodą niewielu strzałów, rozpatrywane były zbiory treningowe o ograniczonej liczby przykładów. Aby zbadać wpływ różnej liczby obrazów w zbiorach treningowych na dokładność wybranych technik, przetestowano modele na zbiorach zawierających odpowiednio 1, 2, 3, 4, 5, 10 oraz 20 elementów. Wyniki eksperymentu zaprezentowane w następującym rozdziale wskazują na zależność pomiędzy liczbą dostępnych przykładów, a dokładnością modelu, podkreślając znaczenie odpowiedniego doboru próbek do efektywnego uczenia. Możliwość takiej oceny zapewnił dobrze skonstruowany zbiór testowy, który jest wykorzystywany do ewaluacji wydajności modelu.

## 2.2. Charakterystyka modeli wstępnie wytrenowanych z Keras

W tej pracy wykorzystano trzy popularne modele konwolucyjnych sieci neuronowych, charakteryzujące się kompaktowymi architekturami. Wybór modeli **MobileNetV2**, **ResNetV2** oraz **EfficientNetV2B0** (patrz tab. 3) jako **ekstraktorów cech** w wybranych technikach został podyktowany potrzebą znalezienia efektywnych rozwiązań dla urządzeń mobilnych. Urządzenia te często dysponują ograniczonymi zasobami obliczeniowymi, co wymusza poszukiwanie modeli, które są nie tylko skuteczne, ale również lekkie i ekonomiczne. Modele te są przykładem kompromisu między wydajnością a wielkością, co jest kluczowe w kontekście aplikacji mobilnych.

MobileNetV2 to mała i lekka sieć, która została zaprojektowana z myślą o minimalizacji wymagań obliczeniowych. Charakteryzuje ją użycie konwolucji separowalnych, które upraszczają proces filtrowania poprzez jego podział na dwie niezależne operacje. Dodatkowo, zastosowanie w modelu wąskich warstw prowadzi do zmniejszenia liczby kanałów w poszczególnych warstwach, co skutkuje ograniczeniem liczby parametrów i upraszczaniem struktury sieci [39]. ResNetV2 wprowadza innowacyjny pomysł bloków rezydualnych, które pozwalają na budowanie bardzo głębokich sieci neuronowych, jednocześnie rozwiązuając problem zanikającego gradientu, który utrudnia trenowanie bardzo głębokich modeli [24]. EfficientNetV2B0 wykorzystuje kombinację wyszukiwania architektury neuronowej i skalowania z treningiem, aby zoptymalizować szybkość szkolenia i wydajność parametrów. Ponadto, model opiera się na adaptacyjnie dostosowanej regularyzacji [45].

Charakterystyka modeli z aplikacji Keras			
Nazwa	Rozmiar (MB)	Parametry	Szybkość obliczeniowa (ms)
MobileNetV2	16	4.3M	22.6
ResNetV2	232	25.6M	58.2
EfficientNetV2B0	29	7.2M	4.9

Tabela 3: Porównanie różnych modeli konwolucyjnych sieci neuronowych uwzględniające ich wielkość, liczbę parametrów i szybkość.

Ze względu na te właściwości, modele opisane powyżej szczegółowo dobrze nadają się do zastosowań na urządzeniach mobilnych. Charakteryzują się one kompaktową architekturą dzięki czemu idealnie nadają się do wykorzystania na platformach z ograniczonymi zasobami obliczeniowymi. Ponadto wydajność energetyczna i zmniejszone zużycie pamięci są dodatkowym atutem używania tej techniki.

## **2.3. Opis wybranych techniki rozpoznawania obrazów**

W tej części pracy przedstawiono techniki stosowane w rozpoznawaniu obrazów, takie jak głębokie dopasowywanie wzorców oraz transfer wiedzy. Autorka tej pracy, technikę DTM szczegółowo opisała w artykule konferencyjnym pt. *"Robust category recognition based on deep templates for educational mobile applications"* [23]. Artykuł ten został zaprezentowany na międzynarodowej konferencji 2023 IEEE International Conference on Big Data, odbywającej się w Sorrento.

### **2.3.1. Głębokie dopasowywanie wzorców**

**Głębokie dopasowywanie wzorców** to technika, która pozwala na wykrywanie i dopasowywanie wzorców w obrazach za pomocą konwolucyjnych sieci neuronowych. Sieci te wykorzystywane do ekstrakcji cech, dostarczają nam abstrakcyjnych reprezentacji danych wizualnych, zwykle pozyskiwanych z przedostatniej warstwy sieci. Technika ta jest wykorzystywana głównie w obszarach rozpoznawania twarzy [44, 5, 42], gdzie proponuje się użycie uśrednionych reprezentacji dla różnych twarzy w celu uzyskania głębokich szablonów (ang. templates). Algorytm działania techniki DTM dzieli się na dwa główne etapy: generowania głębokich szablonów (ang. Deep Template Generation) i dopasowywania głębokich wzorców (ang. Deep Template Matching).

#### **Etap generowania szablonów**

Niech  $N$  oznacza liczbę obrazów wykorzystanych do generowania szablonu  $T_C$  danej klasy  $C$ . Szablon klasy  $T_C$  jest to wartość średnia reprezentacji klasy, obliczona na podstawie reprezentacji poszczególnych obrazów ( $T_{C_1}, \dots, T_{C_N}$ ), zgodnie z wzorem:

$$T_C = \frac{1}{N} \sum_{i=1}^N T_{C_i}. \quad (39)$$

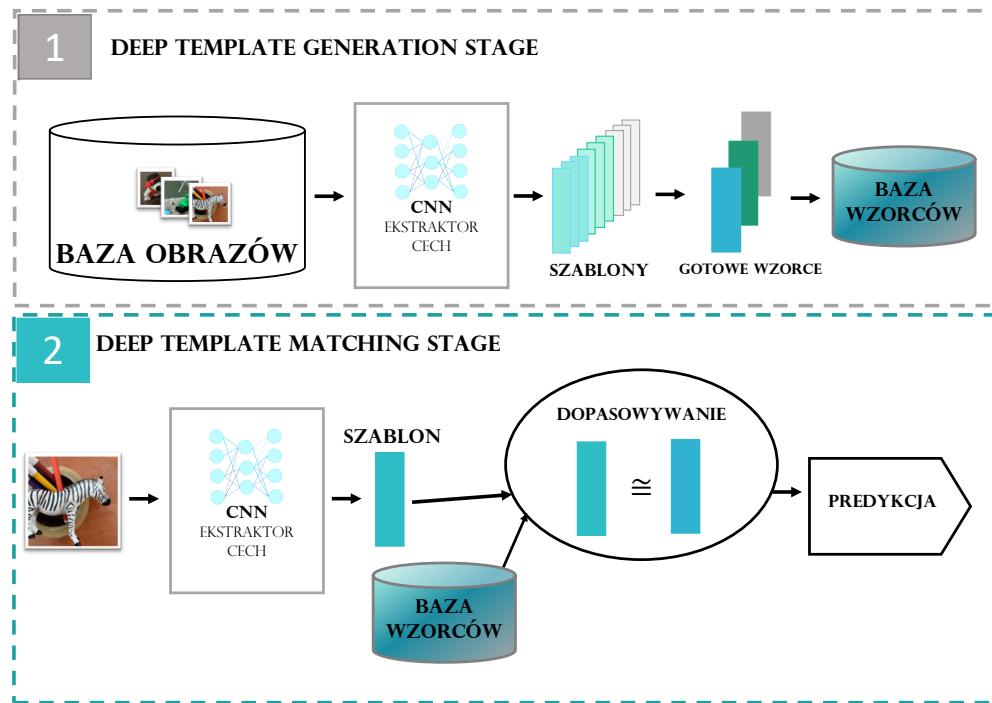
W efekcie otrzymuje się zbiór optymalnych szablonów dla różnych kategorii. Te szablony stanowią bazę danych wzorców, która będzie wykorzystywana do dopasowywania szablonów w drugim etapie.

## Etap dopasowywania głębokich szablonów

Niech  $T_i$  będzie szablonem dla  $i$ -tego obrazu. Należy dokonać porównania  $T_i$  z końcowymi szablonami  $T_C$ . Do porównania wykorzystuje się różne miary odległości i podobieństwa, wybierając klasę  $C$  na podstawie najmniejszej wartości odległości (dla miar odległości) lub największej wartości podobieństwa (dla miar podobieństwa), co można zapisać jako:

$$\hat{y}^{(i)} = \arg \min_C D(T_i, T_C), \quad \hat{y}^{(i)} = \arg \max_C S(T_i, T_C). \quad (40)$$

Ostatecznie zwracana jest predykcja  $\hat{y}^{(i)}$  dla  $i$ -tego obrazu, która odpowiada klasie  $C$  dla której szablon  $T_i$  nowego obrazu najlepiej dopasowuje się do szablonu klasy  $T_C$ . Ogólny proces działania algorytmu głębokiego dopasowywania wzorców można zilustrować, jak na schemacie poniżej (patrz rys. 15).



Rysunek 15: Schemat działania techniki głębokiego dopasowywania wzorców.

W celu osiągnięcia najwyższej dokładności klasyfikacji, rozważono wpływ na działanie modelu pod kątem:

1. oszacowanie optymalnej liczby próbek do reprezentacji klasy,
2. wybór następujących miary podobieństwa/odległości dla dopasowania:
  - (a) odległość Euklidesowa,
  - (b) odległość miejska,
  - (c) odległość Czebyszewa,
  - (d) odległość Minkowski (z parametrem  $r = 3$ ),
  - (e) podobieństwo Jaccarda,
  - (f) podobieństwo cosinusów.

W niniejszej pracy zastosowano technikę DTM ze względu na jej wysoką dokładność, odporność na zniekształcenia obrazu i niskie wymagania dotyczące wielkości bazy danych. Jest to obiecująca technika o szerokim zastosowaniu w rozpoznawaniu i analizie obrazu, szczególnie w kontekście urządzeń mobilnych, gdzie ograniczona moc obliczeniowa i niewielkie rozmiary ekranów wymagają stosowania wydajnych rozwiązań.

### **2.3.2. Transfer wiedzy**

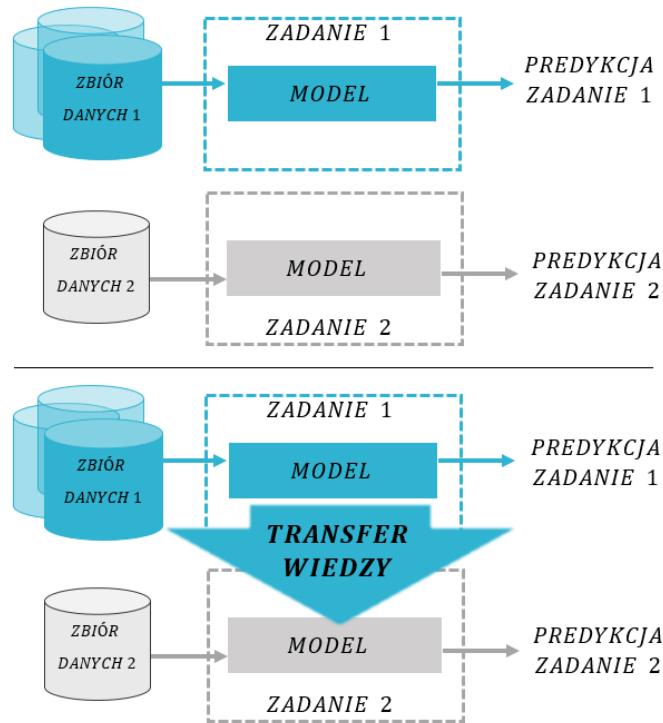
W tym podrozdziale opisując technikę TL korzystano z wiedzy i informacji zgromadzonych w następujących pozycjach [47, 50, 59].

Transfer wiedzy to technika, która pozwala modelowi wykorzystać nabycą wiedzę, zdobywaną podczas treningu na dużym zbiorze danych, do rozwiązywania nowych zadań. Istnieje kilka kryteriów kategoryzacji transferu wiedzy. Ze względu na dostępności etykiet danych wyróżnia trzy następujące kategorie: induktywne (ang. *inductive*), transduktywne (ang. *transductive*), nienadzorowane (ang. *unsupervised*).

Pełna definicja każdej z tych trzech kategorii została przedstawiona w [32]. W niniejszej pracy został opisany transfer induktywny, który najczęściej stosowany jest w konwolucyjnych sieci neuronowych.

Istnieje wiele obszarów z zakresu uczenia maszynowego takich jak klasyfikacja obrazów [13, 18, 28], klasyfikacja nastrojów w tekście [48] oraz wielojęzyczna klasyfikacja tekstu [35, 57], w których ta technika znalazła swoje zastosowanie.

Transfer wiedzy jest szczególnie przydatny, gdy dysponujemy ograniczonym zestawem danych wejściowych. W przypadkach, gdy brakuje odpowiedniej liczby próbek do samodzielnego wytrenowania modelu od podstaw, technika TL pozwala wykorzystać model już wytrenowany na obszernym i zróżnicowanym zbiorze danych, a następnie dostosowywać go do specyficznych potrzeb przy użyciu mniejszej ilości danych (patrz rys. 16).



Rysunek 16: Porównanie klasycznego uczenia maszynowego i transferu wiedzy.

W przypadku transferu wiedzy induktywnego, czyli kiedy etykiety zadania docelowego są dostępne, wyróżnia się dwa algorytmy działania:

1. bazujący na wstępnie wytrenowanej sieci jako ekstraktorze cech,
2. dostrojenie wstępnie wytrenowanej sieci (ang. Fine-tuning).

W omawianej pracy wykorzystano jeden ze wyżej wymienionych sposobów stosowania transferu wiedzy, bazujący na zastosowaniu wstępnie wytrenowanego modelu konwolucyjnych sieci neuronowych jako ekstraktora cech. Model ten, wcześniej był uczyony na ogromnej bazie danych, co pozwoliło na wydobycie bogatych i zróżnicowanych cech. Te cechy będące zaawansowaną reprezentacją przekształconych danych wejściowych, pełnią kluczową rolę w procesie klasyfikacji, służąc jako wkład do nowo skonstruowanego klasyfikatora. Nowy klasyfikator jest następnie odpowiedzialny za dokładne rozróżnienie klas na podstawie wydobytych cech. Dzięki temu, nowy model opierając się na solidnych fundamentach wstępnie wytrenowanego modelu, wykazuje wysoką dokładność w rozwiązywaniu nawet złożonych problemów klasyfikacyjnych.

Niech  $M$  oznacza model bazowy,  $l$  to liczba warstw w modelu  $M$ ,  $w_i$  to wagi  $i$ -tej warstwy, a  $w_C$  to wagi nowego klasyfikatora. Wówczas algorytm transferu wiedzy jest następujący:

1. wybór modelu bazowego  $M$  np. z kolekcji `Keras applications`,
2. zamrożenie warstwy modelu  $M$ :

$$w_i.\text{trainable} := \text{False}, \quad \forall i \in \{1, 2, \dots, l - 1\},$$

3. dodanie nowej warstwy klasyfikatora  $C$  do warstwy wyjściowej modelu  $M$ :

$$C(x) = F(w_C \cdot x),$$

gdzie  $x$  jest wyjściem z przedostatniej warstwy modelu bazowego, a  $F$  jest funkcją aktywacji,

4. trenowanie modelu  $M$  z nową warstwą klasyfikatora  $C$  na nowej bazie danych  $D$ :

$$\min_{W_C} L(M_C(D), Y),$$

gdzie  $M_C$  to model bazowy z dodaną warstwą klasyfikatora  $C$ ,  $D$  to dane treningowe, a  $Y$  to etykiety prawdziwe dla danych  $D$ .

Sposób, w jaki wiedza jest przekazywana z wstępnie wytrenowanego modelu, znacząco zależy od stopnia pokrycia się dziedzin zadań. W przypadku, gdy nowe zadanie jest silnie powiązane z dziedziną, na której model był pierwotnie trenowany, możliwe jest wykorzystanie większej części modelu, w tym bardziej zaawansowanych warstw. Jeśli jednak dziedziny te się różnią, konieczne może być ograniczenie się do wykorzystania tylko podstawowych warstw konwolucyjnych, które generalnie wydobywają bardziej ogólne cechy. Zatem to w jakim stopniu poszczególne elementy modelu są przydatne w nowym zadaniu, zależy bezpośrednio od zbieżności pomiędzy oryginalną i nową dziedziną zastosowania modelu. Szczegółowy opis tego tematu znajduje się w [59]. W tym artykule przeglądowym autorzy dokładnie przeanalizowali mechanizmy i strategie transferu wiedzy z perspektywy danych i modelu.

W tej pracy technikę TL wykorzystano do obniżenia kosztów obliczeniowych na urządzeniach mobilnych. Transfer wiedzy pozwala na użycie rozbudowanych architektur CNN bez konieczności ich trenowania od zera, co zmniejsza obciążenie urządzenia i przyspiesza przetwarzanie danych. Dodatkowo, dzięki zbieżności dziedzin danych zbioru docelowego i źródłowego, istnieje możliwość skutecznego wykorzystania głębszych warstw sieci konwolucyjnych. Jest to istotne ponieważ sieci CNN uczą się różnych poziomów abstrakcji na różnych etapach architektury: początkowe warstwy mogą reprezentować bardziej ogólne cechy, takie jak krawędzie czy tekstury, natomiast głębsze warstwy, uczone na specyficznych danych, wychwytują znacznie bardziej złożone i specyficzne dla danej dziedziny elementy. Dzięki temu, że w pracy klasyfikacja opiera się na danych o podobnej naturze, można efektywnie wykorzystać cechy ekstrahowane przez głębsze warstwy modelu.

# 3. Wyniki

W tej części przedstawiono dokładność klasyfikacji z wykorzystaniem technik szczegółowo opisanych w poprzednim rozdziale. Wyniki zostały zaprezentowane w formie tabel oraz za pomocą graficznych reprezentacji macierzy pomyłek, a także wykresów liniowych i słupkowych, co pozwala na dokładne zapoznanie się z dokładnością uzyskaną przez poszczególne modele wykorzystywane w testowanych technikach TL oraz DTM.

## 3.1. Transfer wiedzy

W ramach tego podrozdziału zaprezentowano wyniki klasyfikacji wieloklasowej z zastosowaniem techniki transferu wiedzy. W tabelach 4, 5, 6 przedstawiono rezultaty oceny trzech wstępnie wytrenowanych modeli, szczegółowo opisanych w podrozdziale 2.2. Aby przetestować ich dokładność w rozpoznawaniu obiektów, zastosowano metryki takie jak dokładność, precyza i czułość. Podczas trenowania modeli ustalono liczbę epok na 3, a jako funkcję optymalizacyjną zastosowano optymalizator *Adam*. Do obliczania funkcji straty użyto *categorical\_crossentropy* dostępnej w bibliotece **Keras**. Standaryzacja parametrów pozwoliła na zbadanie wpływu liczby próbek w zbiorze treningowym na dokładność modeli, eliminując zmiany dokładności wynikające z dostrajania hiperparametrów. W badaniu rozważano różne scenariusze testowe, gdzie zbiór treningowy zawierał 1, 2, 3, 4, 5, 10 lub 20 obrazów, co pozwoliło na lepsze zrozumienie wpływu rozmiaru danych na efektywność klasyfikacji. Ponadto testy przeprowadzono na dwóch bazach testowych: Imagenette oraz autorskiej bazie danych, aby wykazać, że technika jest skalowalna i efektywna w różnych scenariuszach. Kod programu, wykorzystany do testowania modeli, został udostępniony za pośrednictwem repozytorium na GitHubie i jest dostępny pod adresem: [https://github.com/iitis/Transfer\\_wiedzy\\_program](https://github.com/iitis/Transfer_wiedzy_program).

Model **MobileNetV2** w przypadku testowym wykorzystującym obrazy z bazy ImageNette uzyskał poziom ogólnej dokładności wynoszący 93% dla  $N = 3$  próbek w zbiorze treningowym. Dla autorskiej bazy danych osiągnięto ten sam poziom dokładności dla  $N \leq 10$  (patrz tab. 4), co wskazuje na zdolność modelu do skutecznej klasyfikacji nowych obiektów już przy użyciu niewielkiej ilości danych.

MobileNetV2							
baza danych		ImageNette			autorska baza danych		
N	Epoka	Dokładność	Precyzja	Czułość	Dokładność	Precyzja	Czułość
1	1	0.6758	0.8136	0.3797	0.4192	0.5248	0.2124
	2	0.7117	0.8136	0.6075	0.6394	0.8634	0.4264
	3	0.7235	0.7923	0.6608	0.6806	0.8121	0.5251
2	1	0.5035	0.5623	0.3882	0.5582	0.8448	0.3378
	2	0.7975	0.8847	0.6976	0.7869	0.9351	0.6046
	3	0.8446	0.8875	0.8002	0.8339	0.9437	0.7131
3	1	0.5758	0.6043	0.4064	0.4074	0.4585	0.3042
	2	0.9133	0.9645	0.8328	0.7682	0.8981	0.6235
	3	0.9300	0.9727	0.8765	0.7262	0.7779	0.6684
4	1	0.7349	0.8470	0.5514	0.4645	0.6321	0.4122
	2	0.9034	0.9549	0.8501	0.6188	0.6949	0.5130
	3	0.9161	0.9532	0.8829	0.8086	0.8857	0.7370
5	1	0.7259	0.8768	0.5870	0.4186	0.5030	0.3311
	2	0.9471	0.9809	0.8932	0.7424	0.8635	0.6576
	3	0.9132	0.9339	0.8902	0.7994	0.8880	0.7164
10	1	0.8023	0.9178	0.7444	0.3943	0.5031	0.3224
	2	0.9401	0.9565	0.9268	0.6592	0.7174	0.6040
	3	0.8948	0.8999	0.8904	0.9132	0.9597	0.8753
20	1	0.7870	0.8334	0.7424	0.5750	0.6601	0.5423
	2	0.9163	0.9347	0.9033	0.7393	0.7730	0.7095
	3	0.9284	0.9345	0.9235	0.9089	0.9308	0.8881

Tabela 4: Porównanie dokładności, precyzji i czułości modelu MobileNetV2 na zestawie danych ImageNette oraz autorskiej bazy danych w zadaniu klasyfikacji obrazów z uwzględnieniem różnej liczby obrazów (kolumna  $N$ ) w zbiorze treningowym.

Model **EfficientNetV2B0** uzyskał ogólną dokładność ponad 90% przy wykorzystaniu  $N = 5$  obrazów z bazy IamgeNette (patrz tab. 5). Dla obrazów z autorskiej bazy danych model ten dokładność ponad 90% uzyskał dla 10 próbek wykorzystanym do trenowania modelu.

EfficientNetV2B0							
baza danych		ImageNette			autorska baza danych		
N	Epoka	Dokładność	Precyzja	Czułość	Dokładność	Precyzja	Czułość
1	1	0.8479	0.9989	0.1384	0.6293	1.0000	0.0792
	2	0.9205	0.9944	0.6726	0.7207	0.9593	0.3283
	3	0.9380	0.9919	0.8066	0.7458	0.9001	0.5022
2	1	0.9241	0.9947	0.2263	0.7869	0.9940	0.0928
	2	0.9605	0.9966	0.8486	0.8585	0.9670	0.5240
	3	0.9679	0.9946	0.9231	0.8809	0.9566	0.7030
3	1	0.8812	0.9940	0.1736	0.7800	0.9570	0.1375
	2	0.9662	0.9948	0.8067	0.8260	0.9282	0.5735
	3	0.9772	0.9929	0.9335	0.8356	0.9072	0.7026
4	1	0.9506	0.9995	0.3227	0.7342	1.0000	0.1346
	2	0.9801	0.9990	0.8690	0.8446	0.9778	0.5445
	3	0.9843	0.9979	0.9458	0.8784	0.9577	0.7010
5	1	0.9101	0.9782	0.2783	0.7350	1.0000	0.2119
	2	0.9509	0.9839	0.8872	0.8023	0.9531	0.6079
	3	0.9624	0.9810	0.9378	0.8616	0.9404	0.7305
10	1	0.9674	0.9882	0.9146	0.9052	0.9959	0.5598
	2	0.9806	0.9857	0.9715	0.9115	0.9577	0.8333
	3	0.9848	0.9876	0.9794	0.9247	0.9486	0.9017
20	1	0.9812	0.9879	0.9879	0.9506	0.9794	0.8786
	2	0.9869	0.9897	0.9843	0.9649	0.9821	0.9464
	3	0.9896	0.9907	0.9875	0.9673	0.9762	0.9518

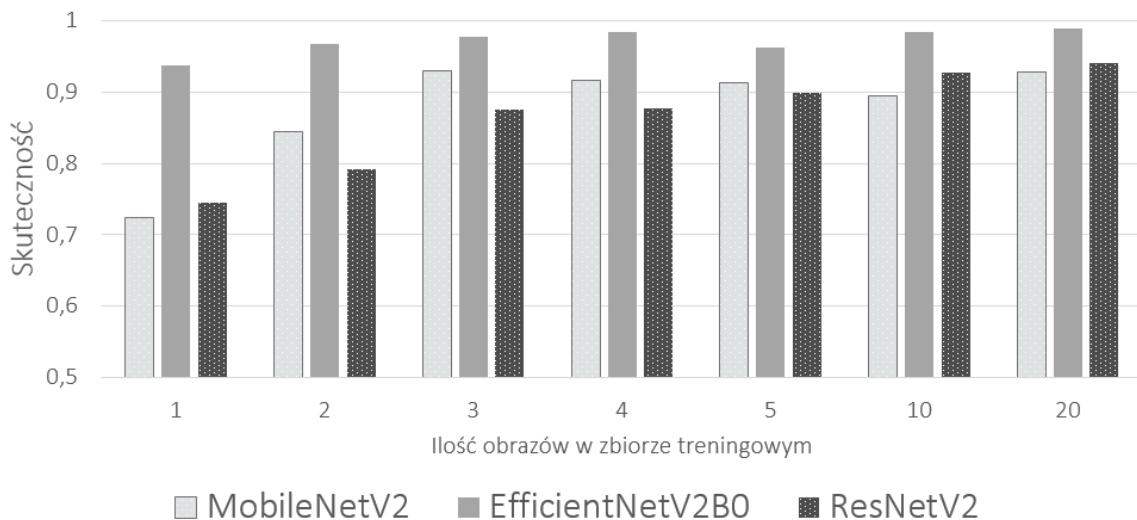
Tabela 5: Porównanie dokładności, precyzji i czułości modelu EfficientNetV2B0 na zestawie danych ImageNette oraz autorskiej bazy danych w zadaniu klasyfikacji obrazów z uwzględnieniem różnej liczby obrazów (kolumna N) w zbiorze treningowym.

Wyniki dla modelu **ResNetV2** wskazują na ponad 90% dokładność wyłącznie w przypadku klasyfikacji obiektów z bazy danych ImageNette (patrz tab. 6). Najwyższy poziom dokładności, wynoszący 0.8970, model uzyskał przy użyciu  $N = 20$  obrazów z autorskiej bazy danych, czyli największej liczby testowanych próbek.

ResNetV2							
baza danych		ImageNette			autorska baza danych		
N	Epoka	Dokładność	Precyzja	Czułość	Dokładność	Precyzja	Czułość
1	1	0.6191	0.7198	0.5602	0.5808	0.6817	0.4955
	2	0.7074	0.7634	0.6705	0.6706	0.7245	0.6098
	3	0.7441	0.7847	0.7244	0.7062	0.7453	0.6639
2	1	0.7197	0.7714	0.6555	0.6012	0.6455	0.5285
	2	0.7783	0.8031	0.7627	0.8255	0.8774	0.7724
	3	0.7916	0.8100	0.7824	0.8384	0.8771	0.8059
3	1	0.6173	0.7120	0.5553	0.5864	0.6137	0.5483
	2	0.8136	0.8738	0.7730	0.7155	0.7704	0.6891
	3	0.8748	0.9099	0.8518	0.7071	0.7327	0.6846
4	1	0.7221	0.7860	0.6975	0.6588	0.8842	0.5203
	2	0.8238	0.8545	0.7995	0.7528	0.8523	0.7247
	3	0.8776	0.8962	0.8640	0.7517	0.8095	0.7393
5	1	0.5660	0.5716	0.5606	0.5972	0.6309	0.5486
	2	0.8177	0.8664	0.7827	0.8062	0.8401	0.7808
	3	0.8996	0.9226	0.8854	0.7672	0.7806	0.7576
10	1	0.9159	0.9338	0.9039	0.7218	0.7760	0.7029
	2	0.9048	0.9093	0.9024	0.8764	0.8956	0.8678
	3	0.9272	0.9318	0.9254	0.8678	0.8804	0.8546
20	1	0.9198	0.9295	0.9142	0.7929	0.8123	0.7756
	2	0.9322	0.9371	0.9307	0.9024	0.9112	0.8923
	3	0.9412	0.9437	0.9392	0.8917	0.8970	0.8869

Tabela 6: Porównanie dokładności, precyzji i czułości modelu ResNetV2 na zestawie danych ImageNette oraz autorskiej bazy danych w zadaniu klasyfikacji obrazów z uwzględnieniem różnej liczby obrazów (kolumna  $N$ ) w zbiorze treningowym.

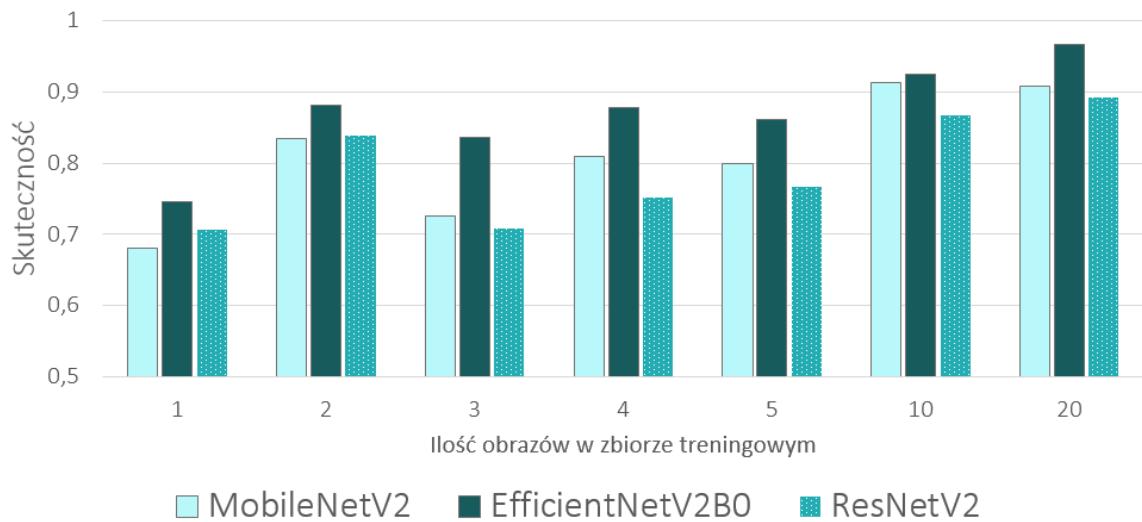
Wykres umieszczony poniżej (patrz rys. 17) ilustruje zależności pomiędzy liczbą obrazów pochodzących ze zbioru ImageNette wykorzystanych do trenowania sieci, a otrzymaną ogólną dokładnością dla rozważanych modeli. Zauważać można, że przy  $N = 1$  każdy model wykazał się ponad 70% dokładnością. W szczególności, model EfficientNetV2B0 wyróżnia się na tle pozostałych modeli, uzyskując ponad 90% dokładności niezależnie od liczby użytych próbek do trenowania modelu. W porównaniu z innymi testowanymi modelami, EfficientNetV2B0 jest najbardziej efektywny w kontekście zadań klasyfikacji obrazów, nawet gdy dostępne są jedynie minimalne dane do trenowania. Model ResNetV2 pozostaje nieco w tyle, co jest szczególnie widoczne przy małej liczbie próbek treningowych. Jego dokładność poprawia się wraz ze wzrostem liczby obrazów w zbiorze treningowym, jednak nie osiąga poziomów skuteczności widocznych w przypadku dwóch pozostałych modeli.



Rysunek 17: Dokładność techniki transferu wiedzy dla bazy danych ImageNette w zależności od ilości elementów w zbiorze treningowym.

Na wykresie dla autorskiej bazy danych (patrz rys. 18) przedstawiono zależność między liczbą obrazów wykorzystanych do trenowania modelu a uzyskaną ogólną dokładnością. Już przy liczbie 2 obrazów, czyli  $N = 2$  każdy z modeli przekroczył

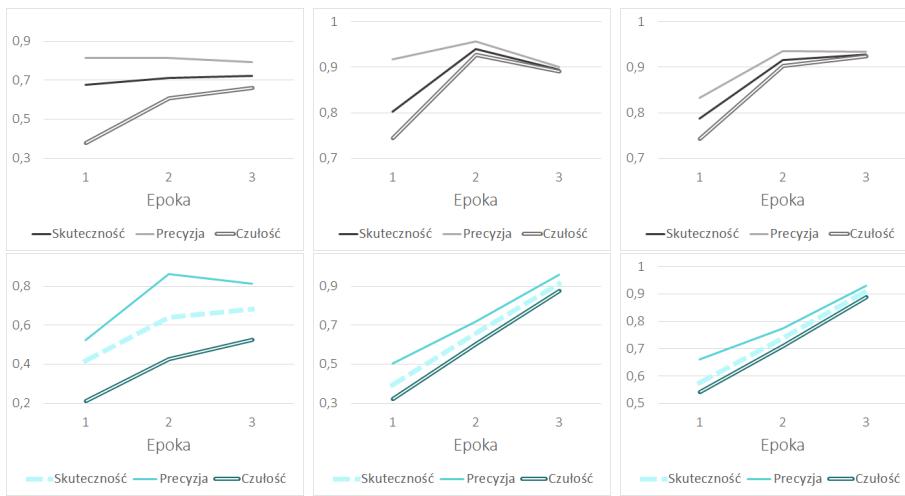
dokładność 80%. W miarę zwiększania liczby próbek, zaobserwowano początkowy spadek dokładności, który następnie stopniowo się poprawiał, uzyskując ponownie poziom ponad 80% przy  $N = 10$ . Model **EfficientNetV2B0** uzyskał najlepsze wyniki w testach wykorzystujących obrazy z autorskiej bazy danych. Wysoka skuteczność tego modelu w różnych konfiguracjach liczby obrazów potwierdza jego zdolność do generalizacji oraz adaptacji do nowych danych.



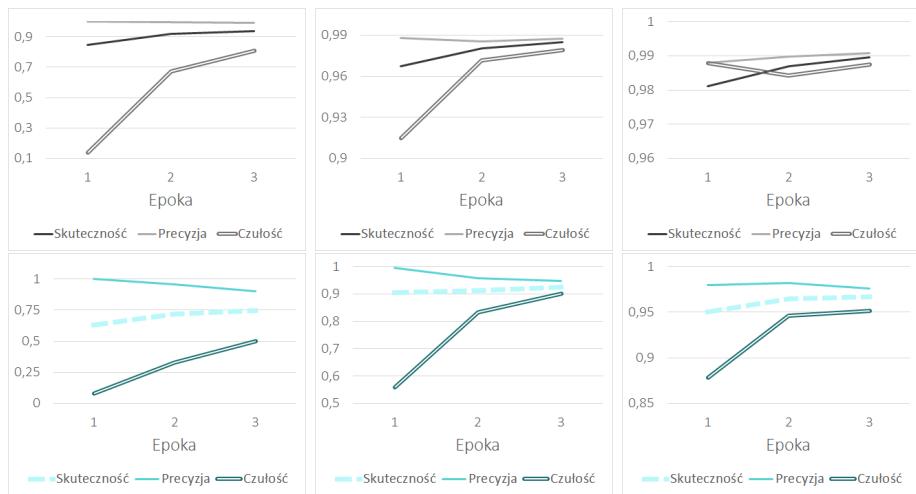
Rysunek 18: Dokładność techniki transferu wiedzy dla autorskiej bazy danych w zależności od ilości elementów w zbiorze treningowym.

Powyższe wyniki jednoznacznie wskazują, że wśród testowanych modeli, dla obu baz danych, model EfficientNetV2B0 uzyskał najwyższą dokładność. Natomiast model ResNetV2, choć osiągnął najsłabsze wyniki, wciąż wykazuje wysoką efektywność. Model MobileNetV2 w obu przypadkach utrzymuje wysoką dokładność, choć w autorskiej bazie danych jego wydajność jest bardziej stabilna w miarę zwiększania liczby próbek. Zauważalnie wysokie wartości dokładności, precyzji i czułości uzyskane przez wszystkie modele pokazują, że technika transferu wiedzy pozwala na skutecną klasyfikację obrazów nawet przy ograniczonej liczbie próbek treningowych.

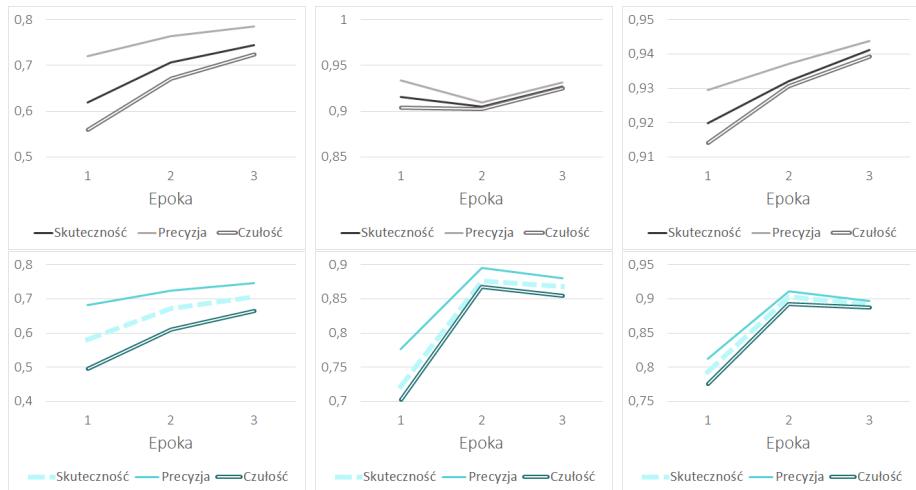
W celu przedstawienia dynamiki procesu trenowania dla trzech wybranych metryk mierzących dokładność modeli, poniżej przedstawiono serię wykresów. Wykresy 19, 20, 21 odpowiednio oznaczone dla poszczególnych modeli, odzwierciedlają poziom dokładności, precyzji i czułości w zależności od liczby iteracji treningowych. Przedstawione wykresy dostarczają informacji o stopniu doskonalenia modeli podczas każdej epoki. Dzięki tym informacjom łatwiej zrozumieć, jak modele uczą się na różnych etapach treningu, jakie metryki są poprawiane w miarę postępu iteracji oraz w którym momencie następuje stabilizacja wyników.



Rysunek 19: Wykres przedstawia zmiany metryk dokładności podczas kolejnych epok dla modelu MobileNetV2. Testy przeprowadzono na obrazach z dwóch baz danych: ImageNette (szary) oraz autorskiej bazy danych (niebieski), przy próbkach treningowych: 1, 10 i 20.



Rysunek 20: Wykres przedstawia zmiany metryk dokładności podczas kolejnych epok dla modelu EfficientNetV2B0. Testy przeprowadzono na obrazach z dwóch baz danych: ImageNette (szary) oraz autorskiej bazy danych (niebieski), przy próbkach treningowych: 1, 10 i 20.



Rysunek 21: Wykres przedstawia zmiany metryk dokładności podczas kolejnych epok dla modelu ResNetV2. Testy przeprowadzono na obrazach z dwóch baz danych: ImageNette (szary) oraz autorskiej bazy danych (niebieski), przy próbkach treningowych: 1, 10 i 20.

Wyniki te wskazują, że technika TL z powodzeniem może być stosowana w rozwiązańach AI dla urządzeń mobilnych, ukazując zalety tej metody w zdolności do znaczącego skracania czasu treningu modeli oraz poprawy ich wydajności w różnych zadaniach klasyfikacyjnych. Wykorzystanie tej techniki pozwala również zmniejszyć zapotrzebowanie na moc obliczeniową, co jest kluczowe w kontekście ograniczonych zasobów sprzętowych.

### **3.2. Głębokie dopasowywanie wzorców**

W tym podrozdziale przedstawiono wyniki klasyfikacji przy użyciu metody głębo-kiego dopasowywania wzorców. Wykorzystane zostały standardowe wskaźniki oceny modelu, takie jak ogólna dokładność, aby zbadać wpływ wyboru miary odległości lub podobieństwa oraz liczby obrazów użytych do tworzenia szablonów na wydaj-ność DTM. Metodę przetestowano dla zestawów zawierających 1, 2, 3, 4, 5, 10 i 20 próbek, oceniając dokładność klasyfikacji dla każdej konfiguracji.

W tabeli poniżej (patrz tab. 7) przedstawiono ogólną dokładność poszczególnych modeli. Pogrubioną czcionką wyróżniono najlepsze wyniki dla każdej liczby próbek treningowych, aby zaznaczyć najwyższą dokładność osiągniętą dla konkretnej miary odległości/podobieństwa w obu bazach danych. Można zauważyć, że nieco lepsze wyniki uzyskały modele trenowane na obrazach pochodzących z ImageNette. W przy-padku tej bazy danych, dla jednej lub większej liczby próbek w zbiorze treningowym, każdy z modeli uzyskał ponad 84% dokładności dla miar Euklidesowej i Minkowskiego (nieco gorsze wyniki uzyskano dla odległości miejskiej). Z kolei miary podobieństwa, takie jak Jaccard i cosinusowa, wykazywały dokładność powyżej 80%. Wzrost licz-by próbek treningowych do  $N = 20$  pozwalał na osiągnięcie niemal maksymalnych wartości dokładności, z wynikami na poziomie 98% dla miary Euklidesowej. W przy-padku obrazów z autorskiej bazy danych wyniki były bardziej zróżnicowane. Model MobileNetV2 wyróżnił się najwyższą dokładnością przy minimalnej liczbie próbek, uzyskując imponujące rezultaty w zakresie podobieństwa cosinusowego. Dla większej liczby próbek inne modele również wykazywały wysoką dokładność, jednak wyniki te były nieco niższe w porównaniu z bazą ImageNette. Poniżej znajduje się dokładny opis wyników dla każdego modelu z osobna, przedstawiający szczegółowe rezultaty dla testowanych scenariuszy.

Model	N	Miara odległości/podobieństwa											
		Euklidesowa		miejiska		Czebyszewa		Minkowski		Jaccard		cosinusów	
		IN	A	IM	A	IN	A	IN	A	IN	A	IN	A
MN	1	0.84	0.66	0.68	0.70	0.57	0.47	0.84	0.66	0.89	0.74	<b>0.90</b>	<b>0.77</b>
	2	0.95	0.89	0.84	0.86	0.62	0.48	0.95	0.89	0.94	0.87	<b>0.96</b>	<b>0.89</b>
	3	<b>0.98</b>	0.89	0.92	0.87	0.64	0.55	0.98	0.89	0.93	0.87	0.96	<b>0.89</b>
	4	0.96	0.90	0.94	0.88	0.65	0.56	0.96	0.90	0.95	0.89	<b>0.97</b>	<b>0.91</b>
	5	<b>0.98</b>	0.91	0.94	0.90	0.66	0.59	0.98	0.91	0.92	0.89	0.96	<b>0.92</b>
	10	<b>0.98</b>	<b>0.96</b>	0.95	0.96	0.71	0.62	0.98	0.96	0.91	0.96	0.97	0.96
	20	<b>0.98</b>	<b>0.98</b>	0.94	0.97	0.73	0.66	0.98	0.98	0.91	0.97	0.97	0.97
EF	1	0.93	0.75	0.93	0.74	0.72	0.52	0.93	0.75	0.98	<b>0.79</b>	<b>0.98</b>	0.79
	2	0.98	0.90	0.98	0.88	0.81	0.66	0.98	0.90	<b>0.99</b>	0.90	0.99	<b>0.90</b>
	3	0.99	<b>0.87</b>	<b>0.99</b>	0.85	0.84	0.66	0.99	0.87	0.99	0.86	0.99	0.86
	4	0.99	<b>0.87</b>	0.99	0.85	0.85	0.68	0.99	0.87	<b>0.99</b>	0.87	0.99	0.87
	5	0.98	<b>0.91</b>	<b>0.99</b>	0.90	0.86	0.67	0.98	0.91	0.98	0.90	0.98	0.91
	10	0.99	0.96	<b>0.99</b>	0.93	0.89	0.71	0.99	0.96	0.99	0.95	0.99	<b>0.96</b>
	20	0.99	0.97	<b>0.99</b>	0.91	0.90	0.75	0.99	0.97	0.98	0.97	0.99	<b>0.98</b>
RN	1	0.84	0.66	0.81	0.70	0.72	0.47	0.84	0.66	0.89	<b>0.78</b>	<b>0.89</b>	0.77
	2	0.92	0.87	0.88	0.85	0.73	0.58	0.92	0.87	0.95	0.89	<b>0.95</b>	<b>0.91</b>
	3	<b>0.96</b>	0.84	0.94	0.85	0.74	0.60	0.96	0.84	0.96	<b>0.87</b>	0.96	0.87
	4	0.92	0.88	0.93	0.88	0.72	0.59	0.92	0.88	0.92	0.89	<b>0.93</b>	<b>0.90</b>
	5	<b>0.97</b>	0.90	0.93	0.90	0.76	0.59	0.97	0.90	0.95	0.90	0.96	0.91
	10	<b>0.98</b>	0.94	0.95	0.95	0.80	0.60	0.98	0.94	0.96	<b>0.95</b>	0.97	0.94
	20	<b>0.97</b>	0.99	0.94	0.98	0.80	0.65	0.97	0.99	0.96	0.98	0.96	<b>0.98</b>

Tabela 7: Dokładność modeli ([%]) dla poszczególnych miar podobieństwa. Pierwsza kolumna zawiera skrócone nazwy modeli: MN - MobileNetV2, EF - EfficientNetV2B0, RN - ResNetV2. Kolumna N przedstawia ilość obrazów użytych do wygenerowania szablonów. Dla baz danych użyto skrótów: IN dla ImageNette i A dla autorskiej bazy danych.

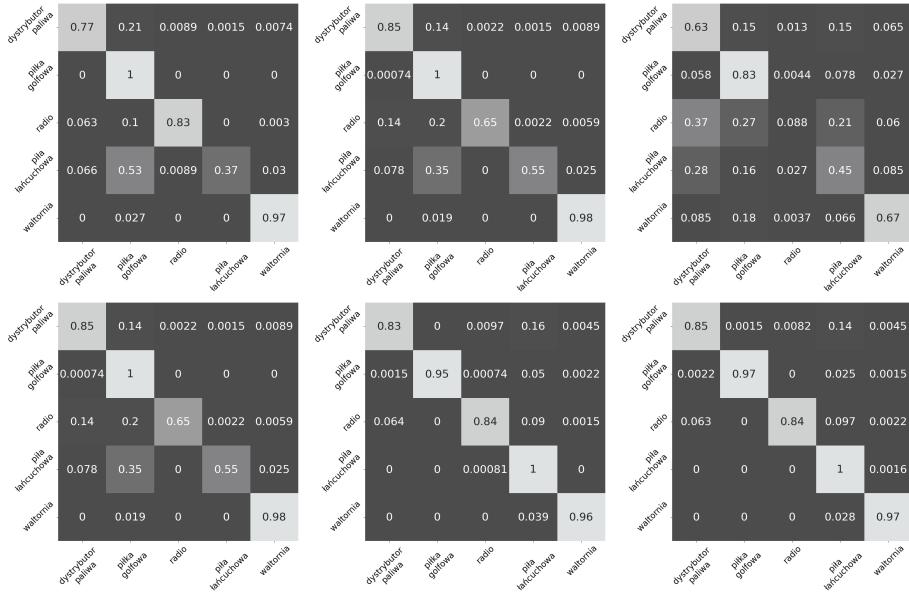
W tej części pracy przedstawiono macierze pomyłek dla modeli kolejno dla każdego z trzech modeli MobileNetV2, ResNetV2 oraz EfficientNetV2B0. Macierze te prezentują wyniki uzyskane przy wykorzystaniu 1, 10 i 20 obrazów z ImageNette (kolor szary) oraz z autorskiej bazy danych (kolor niebieski) do tworzenia szablonów. Klasy obiektów w bazie ImageNette, to odpowiednio: dystrybutor paliwa, piłka golfo-wa, odtwarzacz kasetowy, piła łańcuchowa i waltornia; natomiast klasy w autorskiej bazie danych, to odpowiednio: zebra, Uszatek, miś polarny, łódeczka, buldożer i miś. W rozmieszczeniu macierzy na zaprezentowanych wykresach ustalono następujące ułożenie: pierwszy rząd od lewej - odległość Euklidesowa, odległość miejska, odległość Czebyszewa; drugi rząd od lewej - odległość Minkowski (z parametrem  $r = 3$ ), podobieństwo Jaccarda, podobieństwo cosinusów.

Prezentowane wyniki pozwoliły na ocenę dokładności różnych modeli, uwzględniając różnorodne bazy danych oraz metryki odległości. Testy wykazały, które rozwiązania są najbardziej efektywne w zadaniach klasyfikacji obrazów.

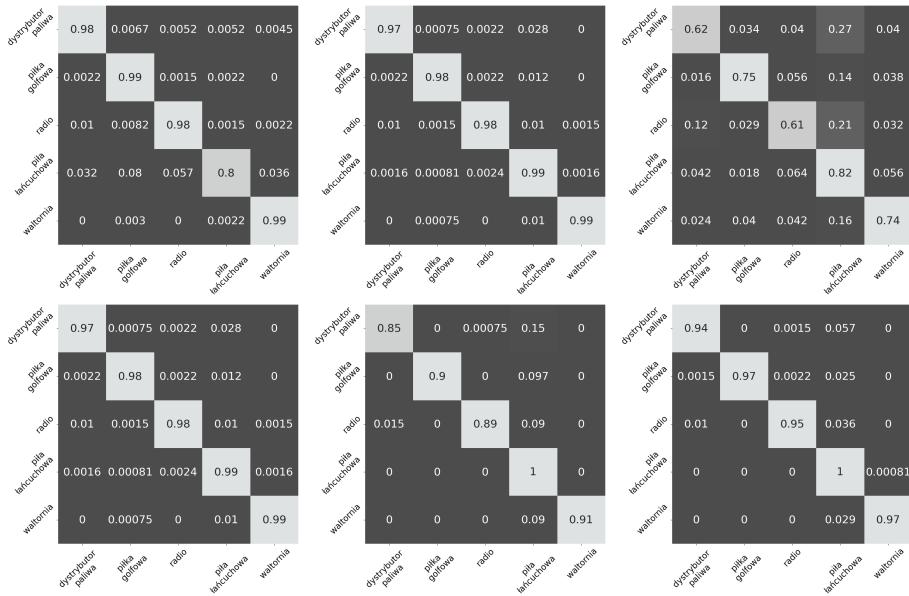
## **MobileNetV2**

Macierz pomyłek umieszczona poniżej (patrz rys. 22) przedstawia wyniki klasyfikacji otrzymane dla pojedynczego obrazu w zbiorze treningowym. Wyniki jednoznacznie pokazują lepszą dokładność przy zastosowaniu miar podobieństwa, takich jak Jaccard i podobieństwo cosinusowe, w przypadku rozpoznawania obiektów należących do ImageNette. Dla autorskiej bazy danych (patrz rys. 25) wyniki wskazują na dwie grupy klas, które model najczęściej mylił: misia polarnego z uszatkim oraz łódeczkę z buldożerem.

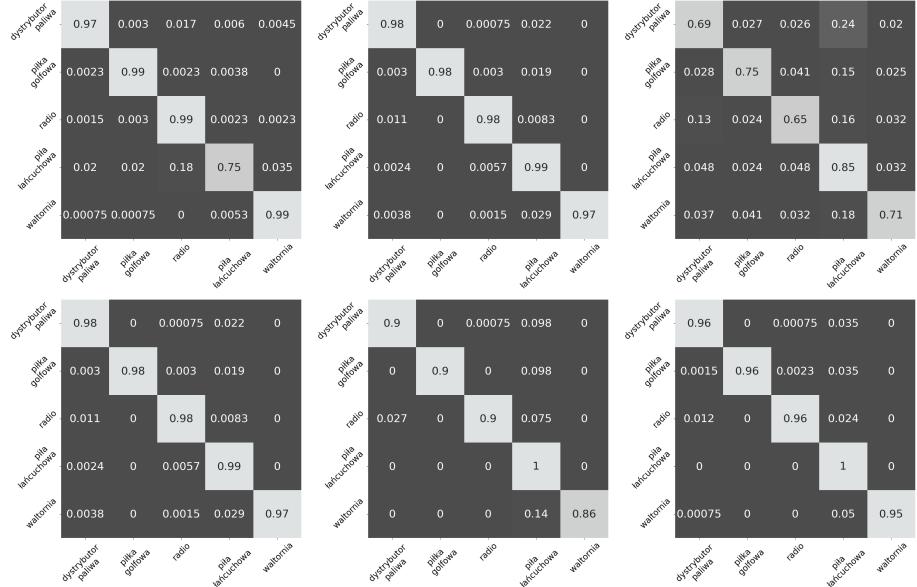
W przypadku, gdy szablony klas zostały wygenerowane przy wykorzystaniu 10 lub 20 obrazów, najczęściej błędy pojawiają się przy zastosowaniu odległości Czebyszewa (patrz rys. 23, 24, 26, 27). Dla pozostałych miar podobieństwa model wykazywał bardzo wysoką pewność decyzyjną.



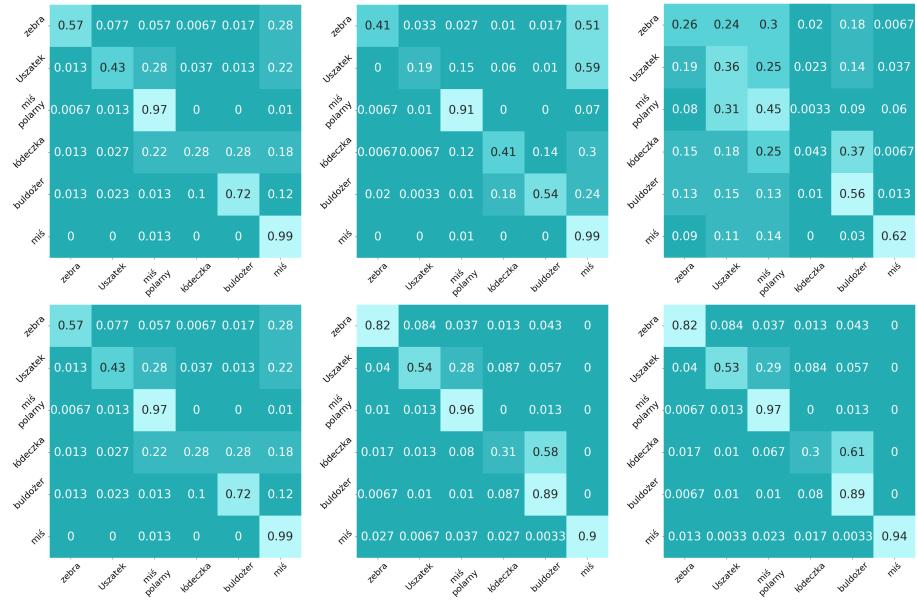
Rysunek 22: Macierze pomyłek modelu MobileNetV2 dla ImageNette, gdzie  $N = 1$ .



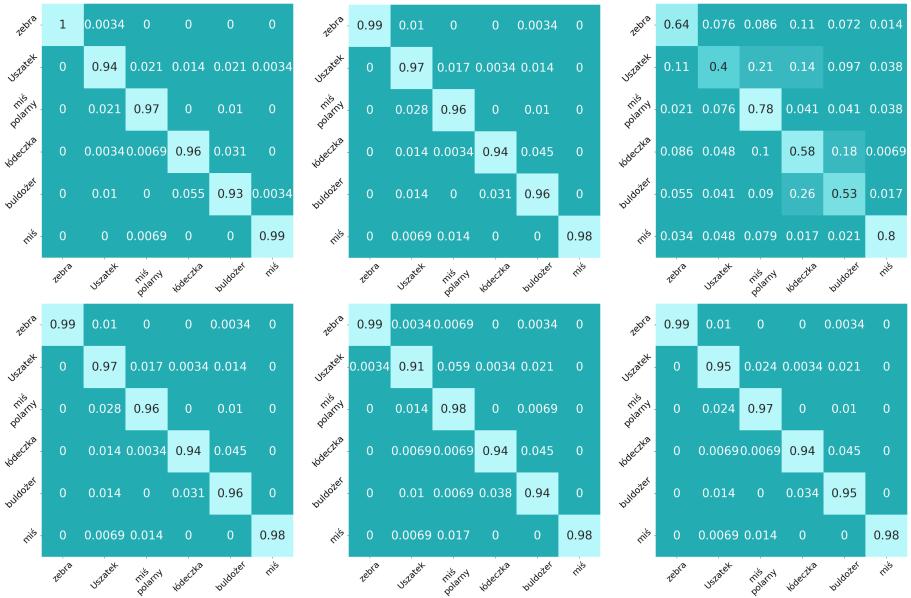
Rysunek 23: Macierze pomyłek modelu MobileNetV2 dla ImageNette, gdzie  $N = 10$ .



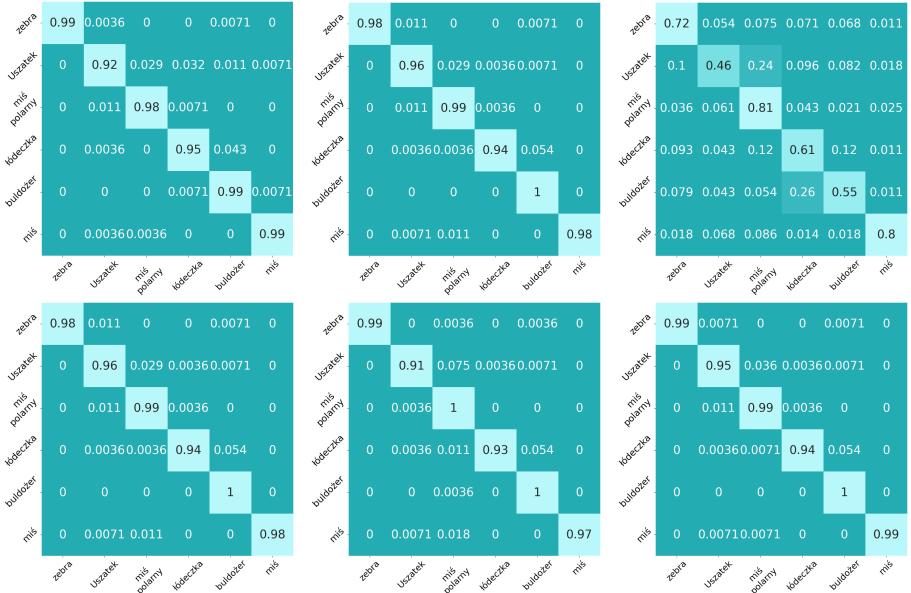
Rysunek 24: Macierze pomyłek modelu MobileNetV2 dla ImageNette, gdzie  $N = 20$ .



Rysunek 25: Macierze pomyłek modelu MobileNetV2 dla autorskiej bazy danych, gdzie  $N = 1$ .



Rysunek 26: Macierze pomyłek modelu MobileNetV2 dla autorskiej bazy danych, gdzie  $N = 10$ .

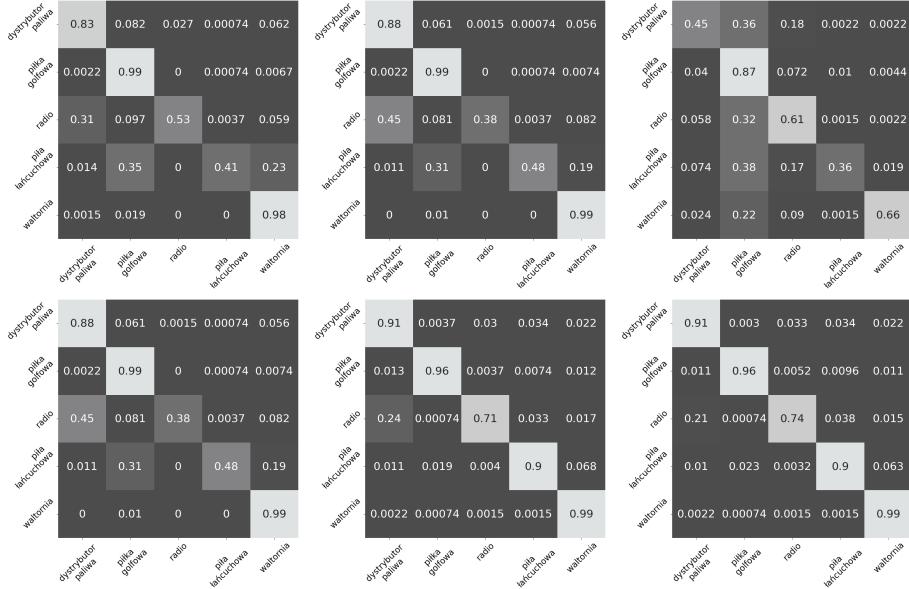


Rysunek 27: Macierze pomyłek modelu MobileNetV2 dla autorskiej bazy danych, gdzie  $N = 20$ .

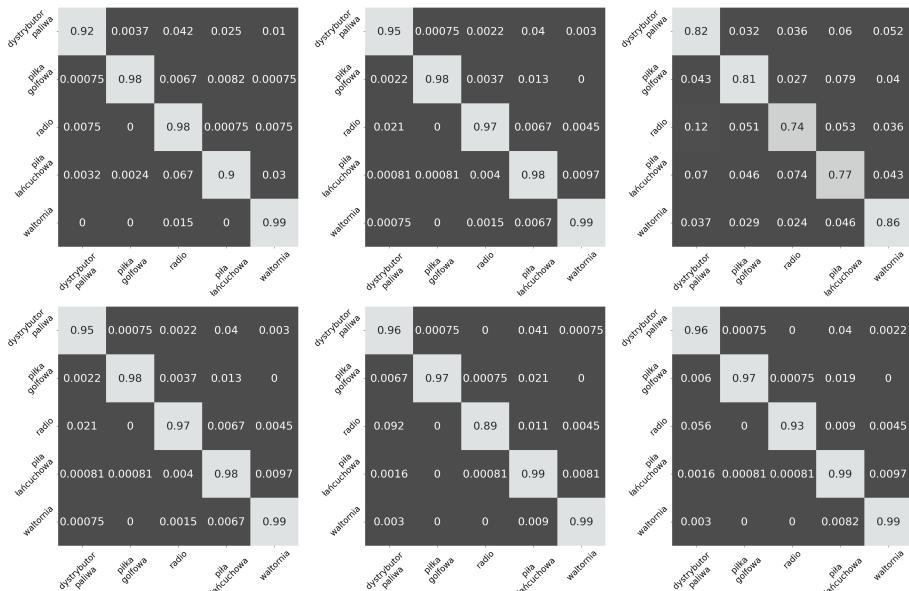
## **ResNet50V2**

Macierz pomyłek przedstawiona poniżej (patrz rys. 28) prezentuje wyniki klasyfikacji uzyskane dla jednego obrazu wykorzystanego do generowania szablonów klas. Dla bazy danych ImageNette najniższą wartość prawdziwie pozytywnych wyników dla miar dystansów uzyskano dla dwóch klas obiektów: radia i piły łańcuchowej. W przypadku dopasowywania szablonów z wykorzystaniem podobieństwa cosinusów obie klasy zostały sklasyfikowane poprawnie na poziomie 90%, a dla miary podobieństwa Jaccarda te same klasy uzyskały wartości 71% dla radia i 74% dla piły łańcuchowej. Zatem podobnie jak w przypadku modelu MobileNetV2, zauważono, że wyniki klasyfikacji uzyskane dla  $N = 1$  próbki w zbiorze treningowym są lepsze przy zastosowaniu miar podobieństwa takich jak Jaccard i podobieństwo cosinusów w odniesieniu do ImageNette. Dla autorskiej bazy danych (patrz rys. 31) różnice te nie są tak wyraźne. Z tej macierzy pomyłek można również odczytać, że łódeczka była obiektem najczęściej niepoprawnie rozpoznawanym. W zależności od zastosowanej miary, jej prawidłowa klasyfikacja wała się od 21% do 44%.

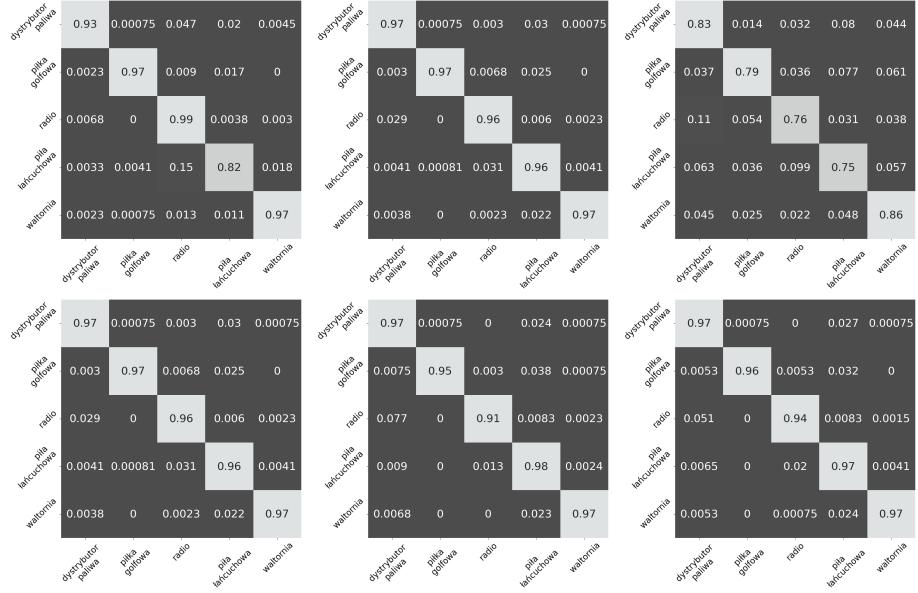
Wraz ze wzrostem liczby obrazów do 10 lub 20, odległość Czebyszewa (patrz rys. 29 30, 32, 33) dawała gorsze wyniki niż pozostałe miary, co wskazuje na jej niższą efektywność na etapie dopasowywania wzorców. Przy użyciu 10 lub więcej obrazów do trenowania modelu, obiekt zebra z autorskiej bazy danych był stabilnie i niemal w 100% poprawnie sklasyfikowany.



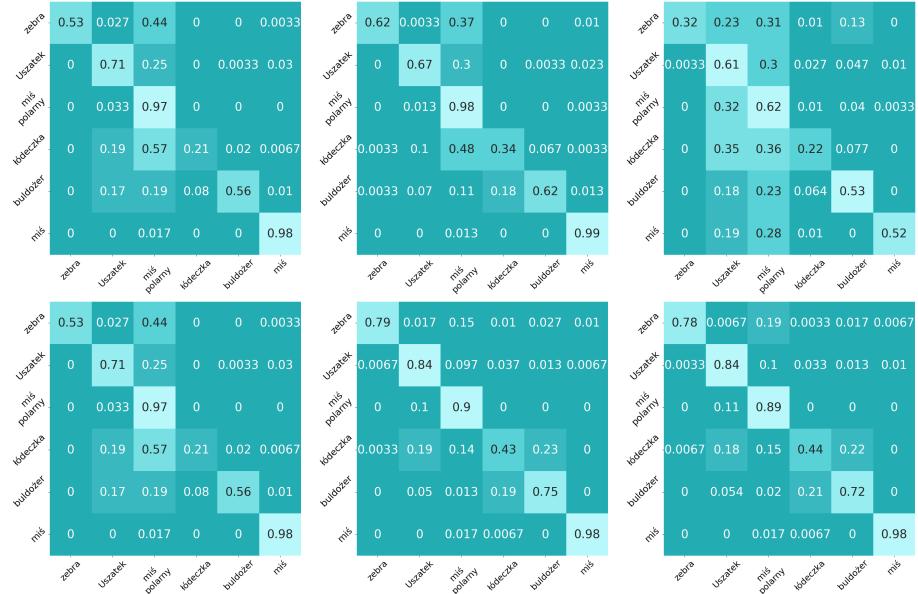
Rysunek 28: Macierze pomyłek modelu ResNet50V2 dla ImageNette, gdzie  $N = 1$ .



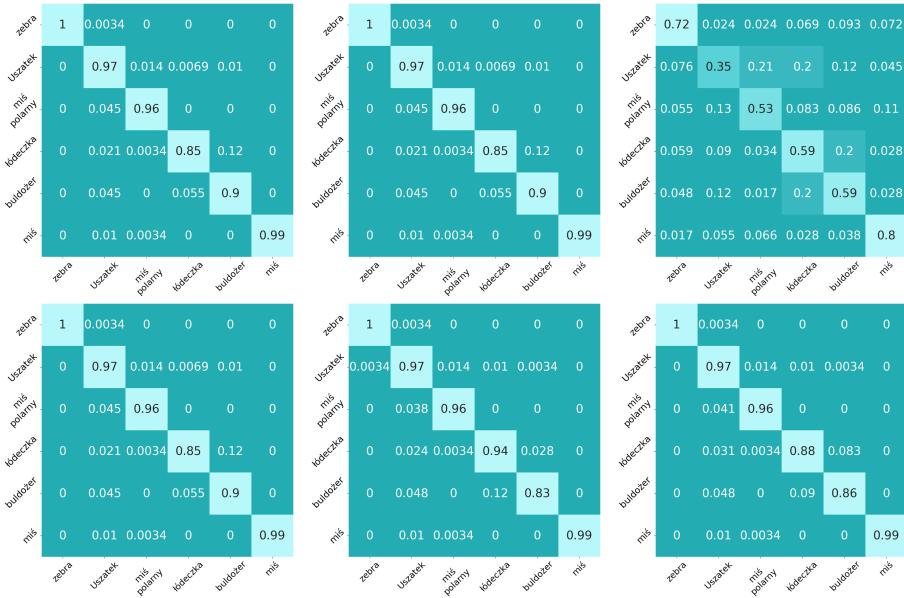
Rysunek 29: Macierze pomyłek modelu ResNet50V2 dla ImageNette, gdzie  $N = 10$ .



Rysunek 30: Macierze pomyłek modelu ResNet50V2 dla ImageNette, gdzie  $N = 20$ .



Rysunek 31: Macierze pomyłek modelu ResNet50V2 dla autorskiej bazy danych, gdzie  $N = 1$ .



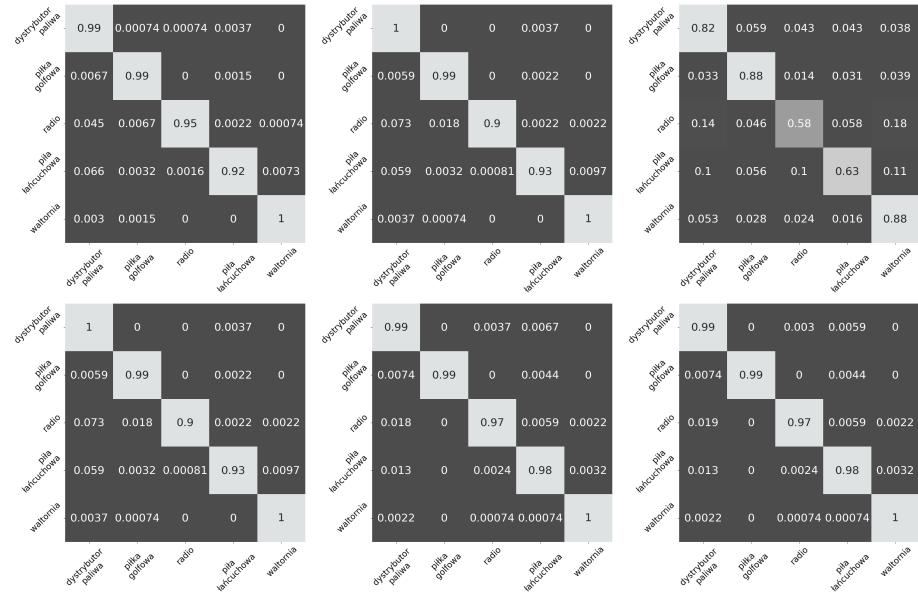
Rysunek 32: Macierze pomyłek modelu ResNet50V2 dla autorskiej bazy danych, gdzie  $N = 10$ .



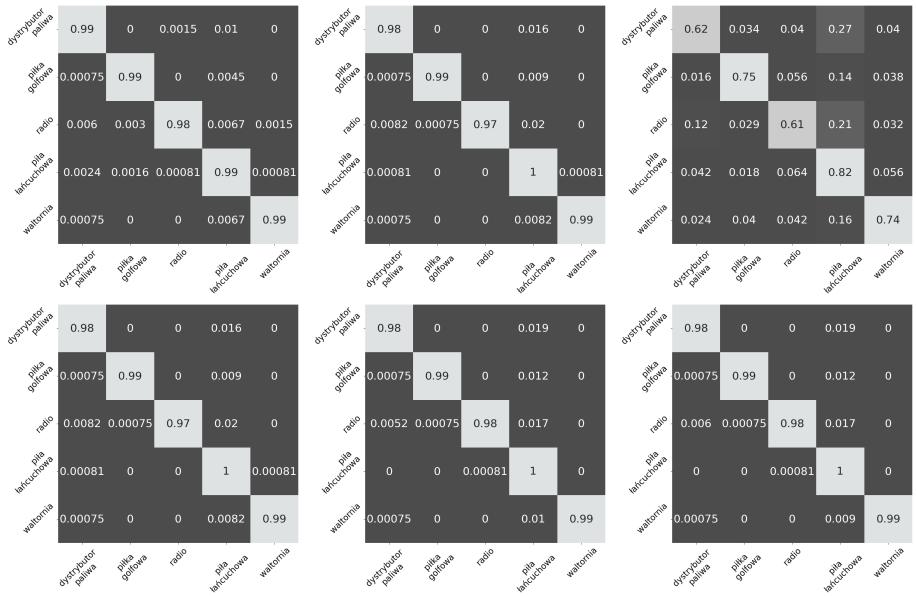
Rysunek 33: Macierze pomyłek modelu ResNet50V2 dla autorskiej bazy danych, gdzie  $N = 20$ .

## EfficientNetV2B0

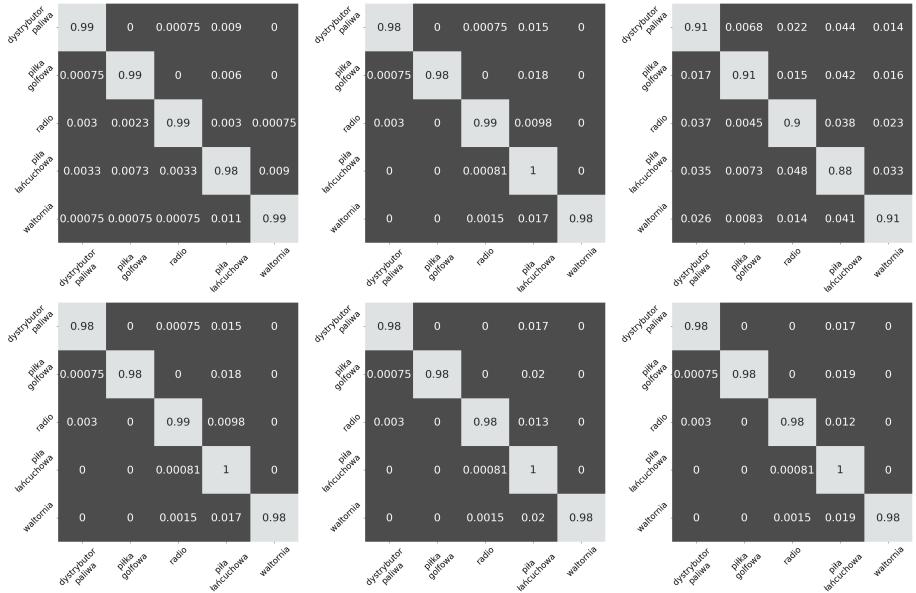
Macierze pomyłek prezentowane poniżej (patrz rys. 34) przedstawiają bardzo wysoką wartość poprawnie sklasyfikowanych wszystkich klas należących do bazy ImageNette, przy użyciu jednej próbki w zbiorze treningowym. Wyjątkiem jest tutaj odległość Czebyszewa, która wykazuje niższą dokładność w porównaniu z pozostałymi miarami. Bardzo wysoka dokładność modelu dla wszystkich miar pozostała stabilnie wysoka przy zwiększeniu liczby obrazów do N=10 i N=20 (patrz rys. 35, 36). W przypadku eksperymentów, w których klasyfikowane były obiekty z autorskiej bazy danych, zauważalna jest duża różnica w dokładności modelu przy korzystaniu z miar odległości w porównaniu do miar podobieństwa (patrz rys. 37). Model ten, podobnie jak MobileNetV2, najczęściej mylił dwie grupy klas: misią polarnego z Uszatkim oraz łódeczkę z buldożerem. Błędy w przypadku tych klas zostały wyeliminowane wraz ze zwiększeniem liczby obrazów wykorzystywanych do generowania szablonów klas (patrz rys. 38, 39).



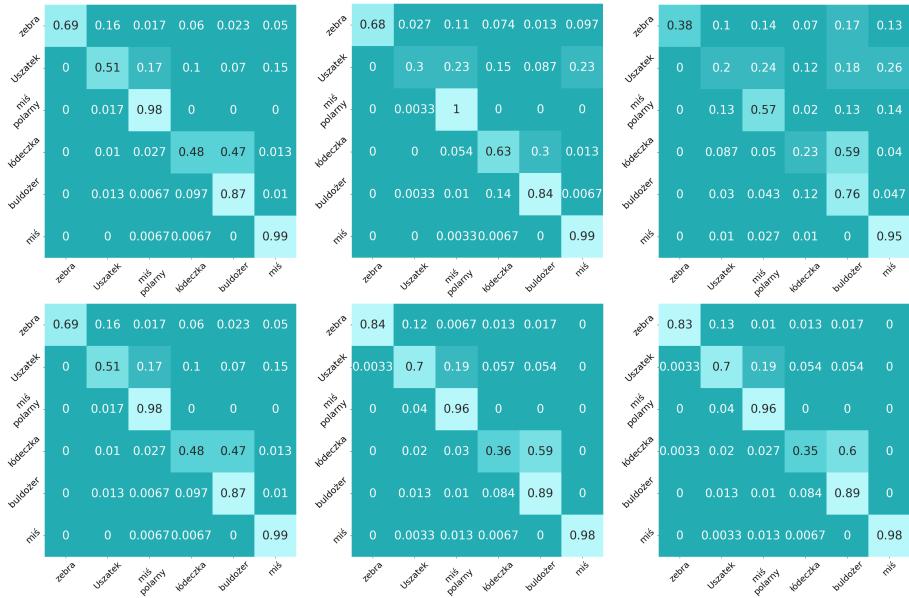
Rysunek 34: Macierze pomyłek modelu MobileNetV2 dla ImageNette, gdzie  $N = 1$ .



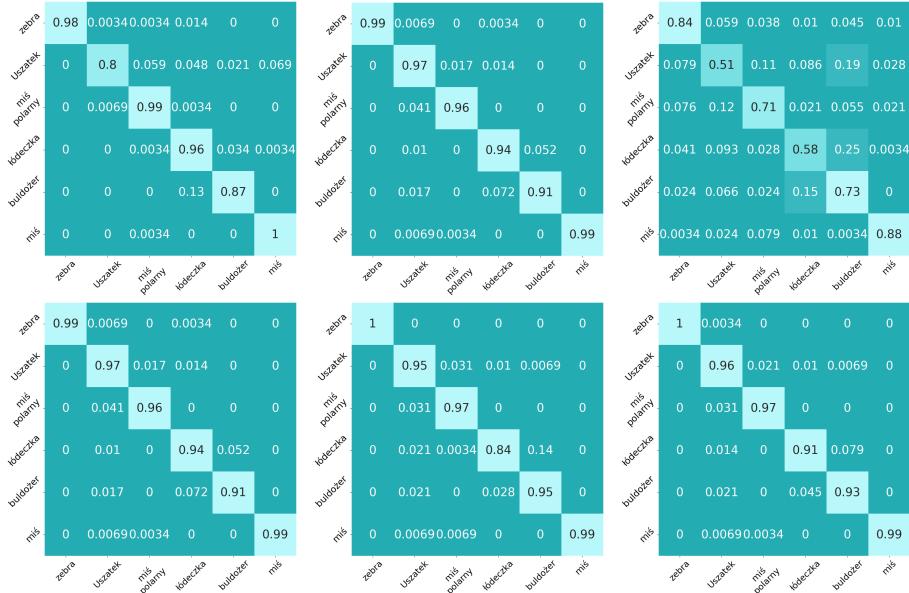
Rysunek 35: Macierze pomyłek modelu EfficientNetV2B0 dla ImageNette, gdzie  $N = 10$ .



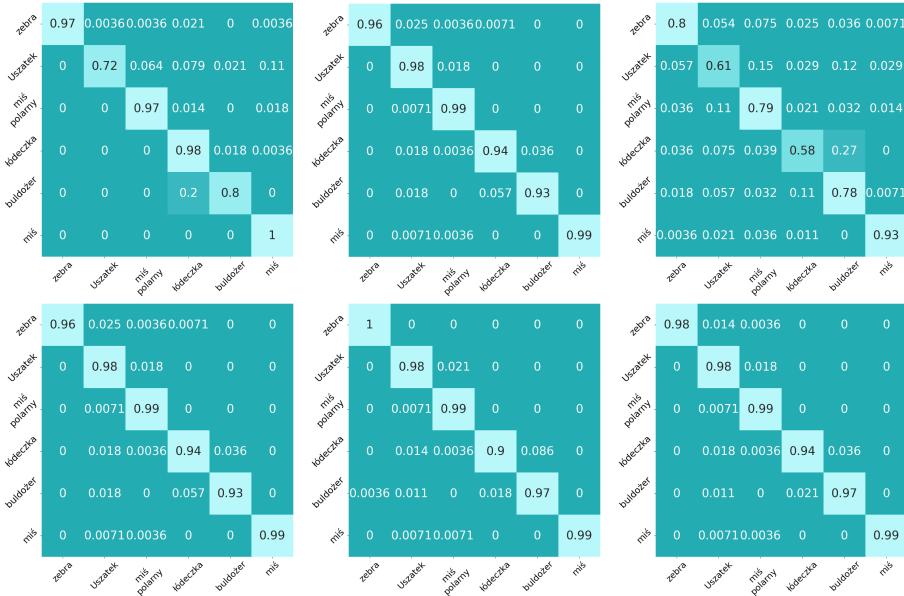
Rysunek 36: Macierze pomyłek modelu EfficientNetV2B0 dla ImageNette, gdzie  $N = 20$ .



Rysunek 37: Macierze pomyłek modelu EfficientNetV2B0 dla autorskiej bazy danych, gdzie  $N = 1$ .



Rysunek 38: Macierze pomyłek modelu EfficientNetV2B0 dla autorskiej bazy danych, gdzie  $N = 10$ .



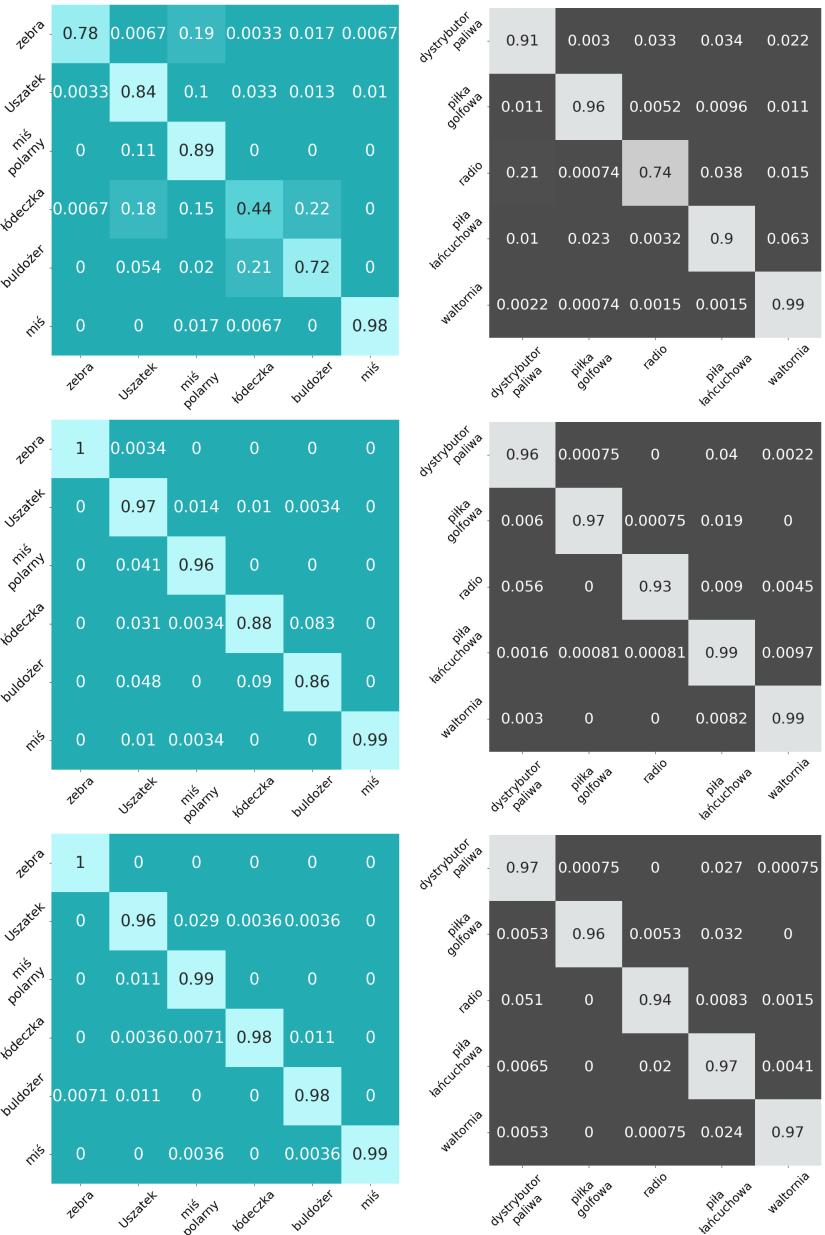
Rysunek 39: Macierze pomyłek modelu EfficientNetV2B0 dla autorskiej bazy danych, gdzie  $N = 20$ .

Wyniki przedstawione w tej części pracy pozwalają na szczegółowe porównanie poprawności klasyfikacji poszczególnych klas dla dwóch różnych baz danych. Ze względu na to, że miara podobieństwa cosinusowego najczęściej osiągała najlepsze wyniki, poniższe ilustracje przedstawiają dokładność uzyskaną przy wykorzystaniu tej miary, kolejno dla każdego z trzech modeli: MobileNetV2 (patrz rys. 40), ResNetV2 (patrz rys. 41) i EfficientNetV2B0 (patrz rys. 42). Macierze te prezentują wyniki uzyskane przy wykorzystaniu 1, 10 i 20 obrazów do tworzenia szablonów.

Na poniższych macierzach pomyłek można zauważyć, jak ze wzrostem liczby elementów w zbiorze treningowym zmieniała się liczba poprawnie sklasyfikowanych obrazów podczas testów modeli. W przypadku bazy ImageNette, wartości na przekątnych były niezwykle wysokie już przy użyciu jednego obrazu do tworzenia szablonu klasy. Natomiast dla autorskiej bazy danych, wartości te były bardziej zróżnicowane, wykazując znaczną poprawę wraz ze zwiększeniem liczby obrazów w zbiorze treningowym.



Rysunek 40: Macierz pomyłek dla modelu **MobileNetV2** po lewej stronie wyniki dla autorskiej bazy danych (kolor niebieski), po prawej stronie wyniki dla ImageNette (kolor szary), kolejno dla 1, 10 oraz 20 obrazów użytych do tworzenia szablonu klasy.



Rysunek 41: Macierz pomyłek dla modelu **ResNet50V2** po lewej stronie wyniki dla autorskiej bazy danych (kolor niebieski), po prawej stronie wyniki dla ImageNette (kolor szary), kolejno dla 1, 10 oraz 20 obrazów użytych do tworzenia szablonu klasy.



Rysunek 42: Macierz pomyłek dla modelu **EfficientNetV2B0** po lewej stronie wyniki dla autorskiej bazy danych (kolor niebieski), po prawej stronie wyniki dla ImageNette (kolor szary), kolejno dla 1, 10 oraz 20 obrazów użytych do tworzenia szablonu klasy.

Wyniki otrzymane dla metody głębokiego dopasowywania wzorców wykazały znaczną skuteczność tej techniki w klasyfikacji obrazów. Już przy użyciu jednej próbki ze zboru ImageNette w zbiorze treningowym, wartości na przekątnych macierzy pomyłek były niezwykle wysokie, co wskazuje na wysoką poprawność klasyfikacji. Również dla autorskiej bazy danych wraz ze zwiększeniem liczby obrazów w zbiorze treningowym, wartości na przekątnych macierzy pomyłek stawały się coraz wyższe, wskazując na poprawę dokładności klasyfikacji. Zatem metoda DTM okazała się efektywnym narzędziem do klasyfikacji obrazów, szczególnie w przypadku stosowania miar podobieństwa. Z powodzeniem można stosować tę technikę już przy bardzo ograniczonej wielkości bazy danych, zachowując jednocześnie wysoką dokładność ogólną modeli w klasyfikacji. Jest to szczególnie istotne dla urządzeń mobilnych, które często dysponują ograniczoną mocą obliczeniową.



## **4. Podsumowanie**

Z badań przeprowadzonych w ramach pracy wynika, że zastosowanie technik takich jak transfer wiedzy i głębokie dopasowywanie wzorców w zadaniach klasyfikacji obrazów opartych na konwolucyjnych sieciach neuronowych stanowi wartościowe narzędzia w dziedzinie przetwarzania i analizy danych wizualnych. Wyniki eksperymentów pokazują, że techniki te nie tylko znacząco poprawiają dokładność modeli, ale również minimalizują obciążenie zasobów obliczeniowych poprzez ograniczenie wielkości wymaganych zbiorów treningowych. Dzięki temu możliwe jest efektywne wykorzystanie konwolucyjnych sieci neuronowych w różnych zastosowaniach, takich jak medycyna, przemysł czy bezpieczeństwo. Te wyniki podkreślają potencjał technologii AI w różnych dziedzinach, w których kluczowa jest szybka i dokładna analiza obrazów.

Transfer wiedzy może być z powodzeniem stosowany do rozwiązania specyficznych problemów przy użyciu mniejszych zestawów danych. W testach wykazano, że aby osiągnąć dokładność klasyfikacji na poziomie 90%, konieczne jest użycie co najmniej 10 próbek w zbiorze treningowym. Ta zdolność TL do uzyskania wysokiej dokładności nawet przy ograniczonej liczbie próbek treningowych jest szczególnie cenna w kontekście ograniczonych zasobów obliczeniowych. Ponadto transfer wiedzy pozwala na wykorzystanie wstępnie wytrenowanych modeli, co znacząco skracą czas treningu. Dzięki temu modele mogą efektywnie działać w różnorodnych środowiskach, zapewniając wysoką dokładność. Jest to szczególnie istotne w dziedzinach wymagających szybkiego i niezawodnego przetwarzania danych, takich jak rozpoznawanie twarzy, analiza obrazów medycznych czy monitorowanie wizyjne.

Technika głębokiego dopasowywania wzorców jest również efektywna w przypadku mniejszych zbiorów danych. Pozwala ona na uzyskanie wysokiej dokładności klasyfikacji przy minimalnym obciążeniu zasobów, co jest kluczowe dla zastosowań

wymagających przetwarzania dużej ilości danych w krótkim czasie. W niniejszej pracy zastosowano technikę DMT ze względu na jej wysoką dokładność, odporność na zniekształcenia obrazu i niskie wymagania dotyczące wielkości bazy danych. DTM jest obiecującą techniką o szerokim zastosowaniu w rozpoznawaniu i analizie obrazów. Może być stosowana do autoryzacji biometrycznej, analizy zdjęć medycznych czy w systemach bezpieczeństwa. Dzięki swojej zdolności do efektywnego przetwarzania obrazów, nawet przy ograniczonej liczbie próbek treningowych, technika ta może znacząco poprawić funkcjonalność i wydajność w wielu różnych dziedzinach.

Podsumowując, praca ta wnosi istotny wkład w dziedzinę sztucznej inteligencji, szczególnie w kontekście praktycznych zastosowań na urządzeniach mobilnych. Przeprowadzone badania i uzyskane wyniki stanowią solidną podstawę do dalszych badań i rozwoju technologii klasyfikacji obrazów, wskazując jednocześnie na potencjalne kierunki przyszłych badań w zakresie optymalizacji i adaptacji tych technik do różnorodnych zastosowań. Aktualność tematyki została potwierdzona akceptacją artykułu opisującego część wyników uzyskanych w ramach tej pracy magisterskiej.

# Bibliografia

- [1] D. A. Alghmham, G. Latif, J. Alghazo, and L. Alzubaidi. *Autonomous traffic sign detection and recognition using deep CNN*. Procedia Computer Science, 2019.
- [2] D. Argüeso, A. Picon, U. Irusta, A. Medela, M. G. San-Emeterio, A. Bereciartua, and A. Alvarez-Gila. *Few-shot learning approach for plant disease classification using images taken in the field*. Computers and Electronics in Agriculture, 2020.
- [3] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, and H. Ghayvat. *CNN variants for computer vision: History, architecture, application, challenges and future scope*. Electronics, 2021.
- [4] R. Chauhan, K. K. Ghanshala, and R. Joshi. *Convolutional Neural Network (CNN) for image detection and recognition*. In 2018 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2018.
- [5] J. C. Chen, V. M. Patel, and R. Chellappa. *Unconstrained face verification using deep CNN features*. In 2016 IEEE Winter Conference on Applications of Computer Vision. IEEE, 2016.
- [6] Y. Chen. *IoT, cloud, big data and AI in interdisciplinary domains*. Simulation Modelling Practice and Theory, 2020.
- [7] F. Chollet. *Deep learning with Python*. Simon and Schuster, 2021.
- [8] P. Cichosz. *Systemy uczące się*. WNT, 2000.
- [9] S. De, A. Maity, V. Goel, S. Shitole, and A. Bhattacharya. *Predicting the popularity of instagram posts for a lifestyle magazine using deep learning*. In 2017

IEEE 2nd International Conference on Communication Systems, Computing and IT Applications. IEEE, 2017.

- [10] J. Deng, et. al. *Imagenet: A large-scale hierarchical image database*. In 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2009.
- [11] J. Deng, et. al. *What does classifying more than 10,000 image categories tell us*. Computer Vision 2010: 11th European Conference on Computer Vision. IEEE, 2010.
- [12] L. Deng, G. Hinton, and B. Kingsbury. *New types of deep neural network learning for speech recognition and related applications: An overview*. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013.
- [13] L. Duan, D. Xu, and I. Tsang. *Learning with augmented features for heterogeneous domain adaptation*. arXiv:1206.4660, 2012.
- [14] S. Fahle, C. Prinz, and B. Kuhlenkötter. *Systematic review on machine learning (ML) methods for manufacturing processes-Identifying artificial intelligence (AI) methods for field application*. Procedia CIRP, 2020.
- [15] E. F. Christiane. *Wordnet: An electronic lexical database*. MIT press, 1998.
- [16] A. R. Feyjie, R. Azad, M. Pedersoli, C. Kauffman, I. B. Ayed, and J. Dolz. *Semi-supervised few-shot learning for medical image segmentation*. arXiv:2003.08462, 2020.
- [17] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li. *Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment*. In 2018 IEEE Transactions on Industrial Informatics. IEEE, 2018.
- [18] Y. Gao and K. M. Mosalam. *Deep transfer learning for image-based structural damage recognition*. Computer-Aided Civil and Infrastructure Engineering, 2018.

- [19] A. Géron. *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow*. Helion, 2018.
- [20] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya. *Multilingual language processing from bytes*. arXiv:1512.00103, 2015.
- [21] T. Guo, J. Dong, H. Li, and Y. Gao. *Simple convolutional neural network on image classification*. In 2017 IEEE 2nd International Conference on Big Data Analysis. IEEE, 2017.
- [22] Y. Guo, N. C. Codella, L. Karlinsky, J. V. Codella, J. R. Smith, K. Saenko, T. Rosing, and R. Feris. *A broader study of cross-domain few-shot learning*. In Computer Vision 2020: 16th European Conference. Springer, 2020.
- [23] M. Halama, K. Filus, and J. Domanska. *Robust category recognition based on deep templates for educational mobile applications*. In 2023 IEEE International Conference on Big Data. IEEE, 2023.
- [24] K. He, X. Zhang, S. Ren, and J. Sun. *Identity mappings in deep residual networks*. In Computer Vision 2016: 14th European Conference. Springer, 2016.
- [25] M. Hussain, J. J. Bird, and D. R. Faria. *A study on CNN transfer learning for image classification*. In Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence. Springer, 2019.
- [26] M. S. Islam, E. H. Moon, M. A. Shaikat, and M. J. Alam. *A novel approach to detect face mask using cnn*. In 2020 IEEE 3rd International Conference on Intelligent Sustainable Systems. IEEE, 2020.
- [27] W. Kołodziej. *Analiza matematyczna*. Wydawnictwo Naukowe PWN, 2009.
- [28] S. Li, W. Song, L. Fang, Y. Chen, P. Ghamisi, and J. A. Benediktsson. *Deep learning for hyperspectral image classification: An overview*. In 2019 IEEE Transactions on Geoscience and Remote Sensing. IEEE, 2019.

- [29] B. Liu, X. Yu, A. Yu, P. Zhang, G. Wan, and R. Wang. *Deep few-shot learning for hyperspectral image classification*. In 2018 IEEE Transactions on Geoscience and Remote Sensing. IEEE, 2018.
- [30] D. M. Montserrat, Q. Lin, J. Allebach, and E. J. Delp. *Training object detection and recognition CNN models using data augmentation*. Electronic Imaging, 2017.
- [31] D. W. Otter, J. R. Medina, and J. K. Kalita. *A survey of the usages of deep learning for natural language processing*. In 2020 IEEE Transactions on Neural Networks and Learning Systems. IEEE, 2020.
- [32] S. J. Pan and Q. Yang. *A survey on transfer learning*. In 2009 IEEE Transactions on Knowledge and Data Engineering. IEEE, 2009.
- [33] B. Pańczyk, E. Łukasik, J. Sikora, and T. Guziak. *Metody numeryczne w przykładach*. Lublin: Politechnika Lubelska, 2012.
- [34] V. Prabhu, A. Kannan, M. Ravuri, M. Chaplain, D. Sontag, and X. Amatriain. *Few-shot learning for dermatological disease diagnosis*. In Machine Learning for Healthcare Conference. PMLR, 2019.
- [35] P. Prettenhofer and B. Stein. *Cross-language text classification using structural correspondence learning*. In 48th annual meeting of the association for computational linguistics, 2010.
- [36] S. Ravi and H. Larochelle. *Optimization as a model for few-shot learning*. In International Conference on Learning Representations, 2016.
- [37] W. Rawat and Z. Wang. *Deep Convolutional Neural Networks for image classification: A comprehensive review*. Neural computation, 2017.
- [38] S. Ruder. *An overview of gradient descent optimization algorithms*. arXiv:1609.04747, 2016.

- [39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. *Mobilenetv2: Inverted residuals and linear bottlenecks*. In 2018 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2018.
- [40] T. Sitek. *Technologie informatyczne wykorzystywane w projektowaniu i implementacji systemów inteligentnych*. Pomorskie Wydawnictwo Naukowo-Techniczne, 2006.
- [41] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. *Meta-transfer learning for few-shot learning*. In 2019 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2019.
- [42] Y. Sun, Y. Chen, X. Wang, and X. Tang. *Deep learning face representation by joint identification-verification*. Advances in neural information processing systems, 2014.
- [43] T. Szandała. *Review and comparison of commonly used activation functions for deep neural networks*. Bio-inspired neurocomputing, 2021.
- [44] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. *Deepface: Closing the gap to human-level performance in face verification*. In 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2014.
- [45] M. Tan and Q. Le. *Efficientnetv2: Smaller models and faster training*. In International Conference on Machine Learning. PMLR, 2021.
- [46] Y. Tkachenko. *Autonomous CRM control via CLV approximation with deep reinforcement learning in discrete and continuous action space*. arXiv:1504.01840, 2015.
- [47] L. Torrey and J. Shavlik. *Transfer learning*. In Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. IGI global, 2010.

- [48] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. *Dense trajectories and motion boundary descriptors for action recognition*. International journal of computer vision, 2013.
- [49] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. *Generalizing from a few examples: A survey on few-shot learning*. ACM computing surveys, 2020.
- [50] K. Weiss, T. M. Khoshgoftaar, and D. Wang. *A survey of transfer learning*. Journal of Big data, 2016.
- [51] M. Wu and L. Chen. *Image recognition based on deep learning*. In 2015 IEEE Chinese Automation Congress. IEEE, 2015.
- [52] X. Wu. *Data mining: An AI perspective*. IEEE Intell. Informatics Bull, 2004.
- [53] J. Xu, Z. Li, B. Du, M. Zhang, and J. Liu. *Reluplex made more practical: Leaky relu*. In 2020 IEEE Symposium on Computers and Communications. IEEE, 2020.
- [54] W. Yin, K. Kann, M. Yu, and H. Schütze. *Comparative study of cnn and rnn for natural language processing*. arXiv:1702.01923, 2017.
- [55] Y. Yu and N. Bian. *An intrusion detection method using few-shot learning*. IEEE Access, 2020.
- [56] K. Zhang, W. Zuo, S. Gu, and L. Zhang. *Learning deep CNN denoiser prior for image restoration*. In 2017 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2017.
- [57] J. Zhou, S. Pan, I. Tsang, and Y. Yan. *Hybrid heterogeneous transfer learning through deep learning*. In AAAI Conference on Artificial Intelligence, 2014.
- [58] P. Zhou, W. Chen, S. Ji, H. Jiang, L. Yu, and D. Wu. *Privacy-preserving online task allocation in edge-computing-enabled massive crowdsensing*. In 2019 IEEE Internet of Things Journal. IEEE, 2019.

- [59] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. *A comprehensive survey on transfer learning*. IEEE, 2020.
- [60] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le. *Learning data augmentation strategies for object detection*. In Computer Vision: 16th European Conference. Springer, 2020.