

# UMIR

Raport z projektu "zdjęcia dronów"

Zespół 8:

**Kazimierz Roman**

**Artur Romaniuk**

**Karol Duszczyk**

26 stycznia 2025

## Spis treści

<b>1</b>	<b>Dane</b>	<b>2</b>
1.1	Cel projektu i dataset . . . . .	2
1.2	Data augmentation . . . . .	2
<b>2</b>	<b>Zadanie 1: Uczenie klasyfikatora</b>	<b>2</b>
2.1	Opis zadania . . . . .	2
2.2	Wybór architektury modelu . . . . .	2
2.3	Wyniki uczenia klasyfikatora . . . . .	2
2.4	Zastąpienie klasyfikatora sieci SVM . . . . .	3
2.4.1	Wyniki dla różnych jąder . . . . .	3
2.4.2	Macierze pomyłek . . . . .	4
2.5	Podsumowanie . . . . .	4
<b>3</b>	<b>Zadanie 2: Uczenie sieci głębokiej</b>	<b>4</b>
3.1	Przeprowadzić uczenie ostatniej warstwy splotowej wraz z częścią klasyfikującą .	4
3.2	Wytrenować całą sieć dla zadanych danych . . . . .	5
3.3	Uprościć strukturę sieci wytrenowanej w zadaniu 2b (np. poprzez usunięcie jednej lub więcej końcowych warstw splotowych, usunięcie warstw regularyzujących itp.) i ponowić uczenie . . . . .	7
3.4	Zanalizować wyniki 2 abc . . . . .	8
<b>4</b>	<b>Zadanie 3: Wizualizacja</b>	<b>8</b>
<b>5</b>	<b>Podsumowanie</b>	<b>8</b>

# 1 Dane

## 1.1 Cel projektu i dataset

Celem projektu jest stworzenie modelu do wykrywania dronów na zdjęciu, który mógłby znaleźć zastosowanie np. na lotniskach, gdzie drony stanowią zagrożenie dla samolotów. Model musi umieć odróżnić drony od samolotów i ptaków, aby nie dawać fałszywych alarmów.

W projekcie użyliśmy zbioru zdjęć *Drone Detection*. Zbiór ten zawiera 4 klasy: 0 - samolot, 1 - dron, 2 - helikopter, 3 - ptak.

## 1.2 Data augmentation

W każdym zadaniu zastosowano augmentację danych. Zastosowano następujące transformacje:

- Random Rotation - obrót o losowy kąt z zakresu  $(-54^\circ, +54^\circ)$  (factor=0.15)
- Random Translation - przesunięcie o losową wartość z zakresu  $(-10\%, +10\%)$  (height\_factor=0.1, width\_factor=0.1)
- Random Flip - losowe odbicie w poziomie lub pionie
- Random Contrast - losowa zmiana kontrastu z zakresu  $(-10\%, +10\%)$  (factor=0.1)

# 2 Zadanie 1: Uczenie klasyfikatora

## 2.1 Opis zadania

Celem tego zadania było wykorzystanie wstępnie wytrenowanej sieci do uczenia wyłącznie części klasyfikującej (ostatnich warstw o połączeniach kompletnych). Następnie wyniki klasyfikacji zostały przeanalizowane. Kolejnym krokiem było zastąpienie części klasyfikującej sieci klasyfikatorem SVM z różnymi jądrami: liniowym, kwadratowym (poly) oraz RBF. Wyniki klasyfikacji dla każdego jądra zostały porównane, a szczególna uwaga została poświęcona efektowi dopuszczenia błędnych klasyfikacji.

## 2.2 Wybór architektury modelu

Do realizacji zadania wykorzystano sieć EfficientNetB0, wstępnie wytrenowaną na zbiorze ImageNet. Warstwy bazowe modelu zostały zamrożone, aby nie były aktualizowane podczas uczenia. Ostatnie warstwy klasyfikujące zostały zastąpione następującą strukturą:

- Warstwa GlobalAveragePooling2D,
- Warstwa Dropout (z prawdopodobieństwem 0.2),
- Warstwa Dense z liczbą neuronów odpowiadającą liczbie klas i funkcją aktywacji softmax.

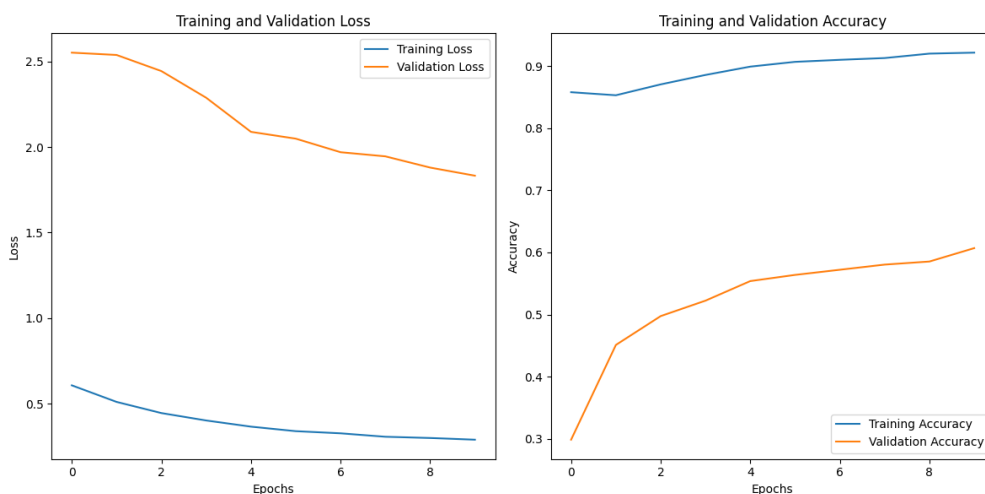
Struktura modelu została przedstawiona w Tabeli 1.

## 2.3 Wyniki uczenia klasyfikatora

Wyniki procesu uczenia, przedstawione na wykresach (Rys. 1), ilustrują zmniejszanie się wartości funkcji straty (loss) oraz wzrost dokładności (accuracy) w zbiorach treningowym i walidacyjnym.

Warstwa	Rozmiar wyjścia	Liczba parametrów
Wejście	(None, 224, 224, 3)	0
EfficientNetB0	(None, 7, 7, 1280)	4,049,571
GlobalAveragePooling2D	(None, 1280)	0
Dropout	(None, 1280)	0
Dense	(None, 4)	5,124

Tabela 1: Podsumowanie modelu klasyfikatora.



Rysunek 1: Wykresy funkcji straty i dokładności dla zbiorów treningowego i walidacyjnego.

	Klasa 0	Klasa 1	Klasa 2	Klasa 3
Klasa 0	41	39	46	2
Klasa 1	26	206	2	0
Klasa 2	4	7	97	3
Klasa 3	29	51	32	11

Tabela 2: Macierz pomyłek dla zbioru testowego.

## 2.4 Zastąpienie klasyfikatora sieci SVM

W celu poprawy klasyfikacji, cechy wyodrębnione z ostatniej warstwy modelu EfficientNetB0 zostały wykorzystane do uczenia klasyfikatora SVM z różnymi jądrami. Wyniki klasyfikacji zostały ocenione za pomocą miar takich jak precision, recall i F1-score.

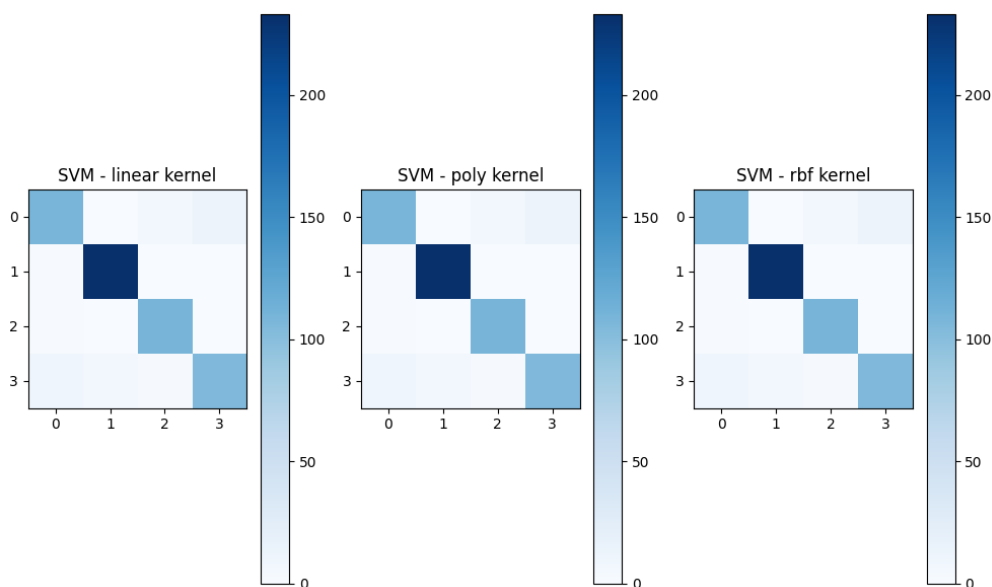
### 2.4.1 Wyniki dla różnych jąder

- **Jądro liniowe:** Dokładność wyniosła 88%, a F1-score dla poszczególnych klas mieściło się w zakresie 0.80–0.95.
- **Jądro kwadratowe (poly):** Najwyższa dokładność na poziomie 93%, przy wysokich wartościach precision i recall.

- **Jądro RBF:** Dokładność również wyniosła 93%, jednak efektywność była różna w zależności od klasy.

#### 2.4.2 Macierze pomyłek

Macierze pomyłek dla każdego jądra pozwoliły na identyfikację klas trudniejszych do rozróżnienia, co przedstawiono na rysunku 2.



Rysunek 2: Macierze pomyłek dla różnych jąder SVM.

### 2.5 Podsumowanie

Porównanie wyników klasyfikacji dla różnych jąder SVM wykazało, że jądra kwadratowe i RBF osiągały wyższą dokładność niż jądro liniowe, co sugeruje lepsze odwzorowanie granic decyzyjnych w bardziej złożonych przestrzeniach. Wyniki te wskazują na możliwość dalszej poprawy wydajności modelu poprzez dostrojenie hiperparametrów klasyfikatora.

## 3 Zadanie 2: Uczenie sieci głębokiej

### 3.1 Przeprowadzić uczenie ostatniej warstwy splotowej wraz z częścią klasyfikującą

Ostatnią warstwą splotową w sieci EfficientNetB0 jest warstwa `top_conv`, która jest trzecią warstwą od góry. Z tego powodu zamrożono wszystkie warstwy sieci poza trzema ostatnimi. Jako wagi początkowe zastosowano wagi `imagenet`. Następnie przeprowadzono uczenie z wykorzystaniem zbioru treningowego. W trakcie uczenia zastosowano optymalizator Adam, funkcję straty `sparse categorical crossentropy` oraz metrykę `accuracy`. Przetestowano różne wartości współczynnika uczenia, ostatecznie wybrano wartość  $1e-5$ . Po 40 epokach uczenia osiągnięto na zbiorze

testowym **accuracy na poziomie 0.74**, **loss na poziomie 0.66** oraz macierz pomyłek przedstawioną w tabeli 3. Wyniki uczenia w czasie przedstawiono na rysunku 3. Dalsze uczenie nie przynosiło poprawy wyników.

Przewidywane:	samolot (0)	dron (1)	helikopter (2)	ptak (3)
samolot (0)	32	18	29	49
dron (1)	0	229	4	1
helikopter (2)	1	4	105	1
ptak (3)	4	19	24	76

Tabela 3: Macierz pomyłek dla zadania 2a



Rysunek 3: Wyniki uczenia dla zadania 2a

Model bardzo dobrze radzi sobie z wykrywaniem dronów, za to ma problem z wykrywaniem samolotów.

### 3.2 Wytrenować całą sieć dla zadanych danych

W tym zadaniu odmrożono wszystkie warstwy EfficientNetB0 i nadano im wagi początkowe imagenet. Następnie przeprowadzono uczenie z wykorzystaniem zbioru treningowego. W trakcie uczenia zastosowano optymalizator Adam, funkcję straty sparse categorical crossentropy oraz metrykę accuracy. Przetestowano różne wartości współczynnika uczenia, ostatecznie wybrano wartość  $1e-5$ . Podczas uczenia zastosowano mechanizm early stopping, który zatrzymywał uczenie jeśli przez 4 epoki nie następowała poprawa wyników. Uczenie zatrzymało się po 22 epokach. Ostatecznie osiągnięto na zbiorze testowym **accuracy na poziomie 0.63**, **loss na poziomie 1.01** oraz macierz pomyłek przedstawioną w tabeli 4. Wyniki uczenia w czasie przedstawiono na rysunku 4.

Przewidywane:	<b>samolot (0)</b>	<b>dron (1)</b>	<b>helikopter (2)</b>	<b>ptak (3)</b>
<b>samolot (0)</b>	56	17	55	0
<b>dron (1)</b>	0	234	0	0
<b>helikopter (2)</b>	22	1	87	1
<b>ptak (3)</b>	9	36	78	0

Tabela 4: Macierz pomyłek dla zadania 2b z wagami początkowymi imagenet



Rysunek 4: Wyniki uczenia dla zadania 2b z wagami imagenet

Model bardzo dobrze radzi sobie z wykrywaniem dronów (100% dronów zostało wykrytych), za to ma problem z wykrywaniem samolotów i ptaków - są one klasyfikowane jako helikoptery. Przykładowe błędne detekcje przedstawiono na rysunku 5.



Rysunek 5: Przykładowe błędne detekcje

Próbowano także wytrenować całą sieć bez zadanych wag początkowych, jednak w tym przypadku wyniki były dużo gorsze (accuracy na poziomie 0.38). Może to wynikać z faktu, że zbiór treningowy jest stosunkowo mały, a wagi początkowe z imagenet są lepsze niż losowe.

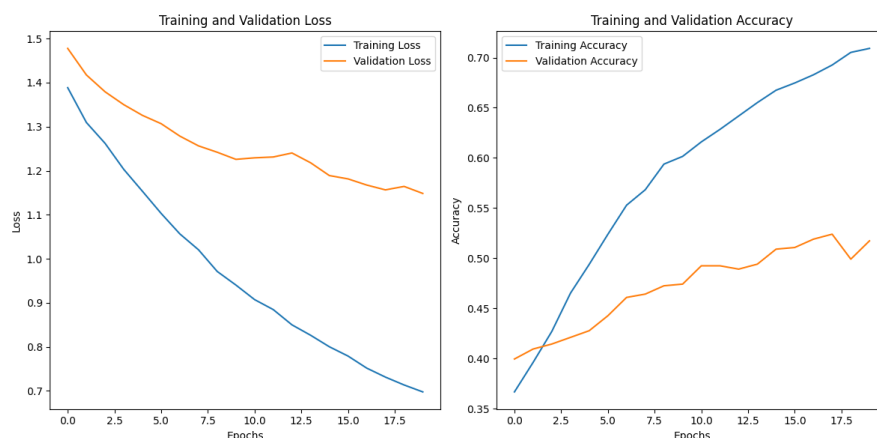
### 3.3 Uprościć strukturę sieci wytrenowanej w zadaniu 2b (np. poprzez usunięcie jednej lub więcej końcowych warstw splotowych, usunięcie warstw regularyzujących itp.) i ponowić uczenie

W celu uproszczenia sieci usunięto wszystkie warstwy EfficientNetB0 powyżej warstwy block5b\_add oraz zmodyfikowano head. Początkowo model miał 4,012,672 (15.31 MB) trainable params oraz 42,023 (164.16 KB) non-trainable params. Po uproszczeniu wielkość zmalała do 703,144 (2.68 MB) trainable params oraz 13,255 (51.78 KB) non-trainable params. Następnie przeprowadzono uczenie z wykorzystaniem zbioru treningowego. W trakcie uczenia zastosowano optymalizator Adam z współczynnikiem uczenia  $1e-5$ , funkcję straty sparse categorical crossentropy oraz metrykę accuracy. Po 20 epokach osiągnięto na zbiorze testowym **accuracy na poziomie 0.52**, **loss na poziomie 1.18** oraz macierz pomyłek przedstawioną w tabeli 5. Wyniki uczenia w czasie przedstawiono na rysunku 6.

Przewidywane:	samolot (0)	dron (1)	helikopter (2)	ptak (3)
samolot (0)	6	25	60	37
dron (1)	0	175	52	7
helikopter (2)	2	5	99	5
ptak (3)	6	7	80	30

Tabela 5: Macierz pomyłek dla zadania 2c





Rysunek 6: Wyniki uczenia dla zadania 2c

### 3.4 Zanalizować wyniki 2 abc

Uczenie ostatniej warstwy splotowej pozwoliło osiągnąć dokładność 74%, co jest średnio zadowalającym wynikiem. Jednak, jeśli wystarczyłaby nam informacja binarna (jest dron / nie ma drona), to taki klasyfikator mógłby być użyteczny, ponieważ 97,96% dronów zostało wykrytych i jedynie 11,32% nie-dronów zostało sklasyfikowanych jako drony. Uczenie całej sieci z wagami imagenet pozwoliło osiągnąć dokładność 63%, co jest wynikiem gorszym niż w przypadku uczenia ostatniej warstwy splotowej, jednak w tym przypadku 100% dronów ze zbioru testowego zostało poprawnie wykrytych. Uproszczenie sieci pozwoliło zmniejszyć wielkość modelu oraz liczbę parametrów ponad 5-krotnie, jednak kosztem dokładności - wynik wyniósł zaledwie 52%. Warto zauważyć, że w przypadku uproszczenia sieci, model nadal całkiem dobrze radzi sobie z wykrywaniem dronów (75% dronów zostało wykrytych), jednak ma problem z wykrywaniem pozostałych klas.

Wyniki nie są dość zadowalające, aby móc wykorzystać model do praktycznych zastosowań. Możliwe przyczyny niskiej dokładności to mała ilość danych treningowych oraz ich jakość. Przykład słabej jakości danych widać na rysunku 5 - nawet człowiek ma problem z klasyfikacją niektórych obrazów.

## 4 Zadanie 3: Wizualizacja

## 5 Podsumowanie