UMIR

Raport z projektu "zdjęcia dronów"

Zespół 8:
Kazimierz Roman
Artur Romaniuk
Karol Duszczyk

3 lutego 2025

Spis treści

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 $(-54°, +54°)$ (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.

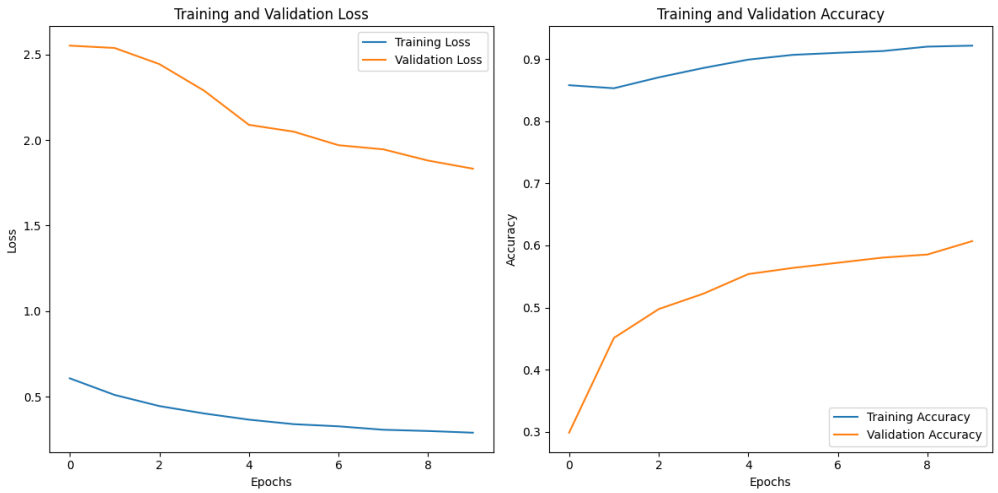
Struktura modelu została przedstawiona w Tabeli 1.

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.

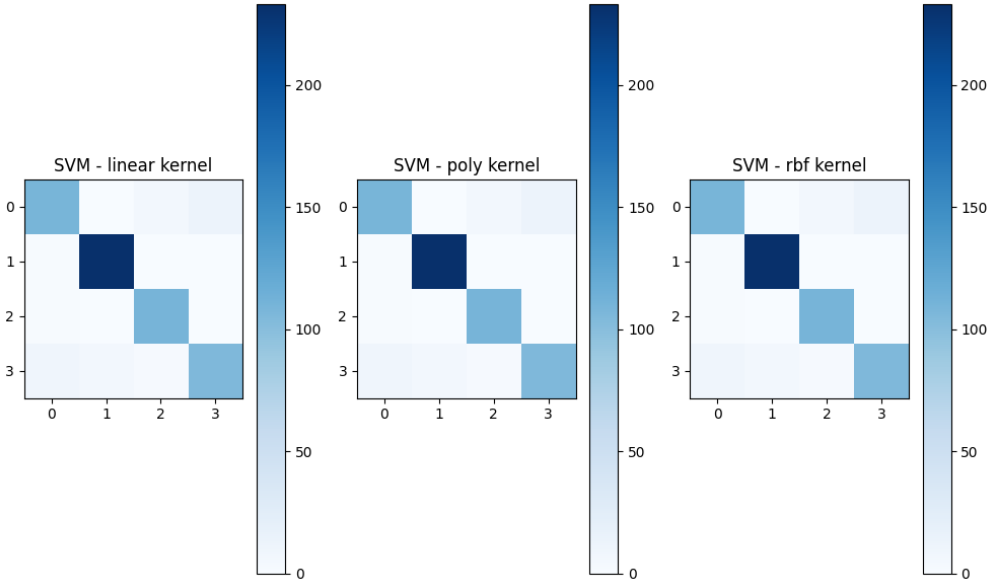
2.4.1 Wyniki dla różnych jąder

- **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 osiągają 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

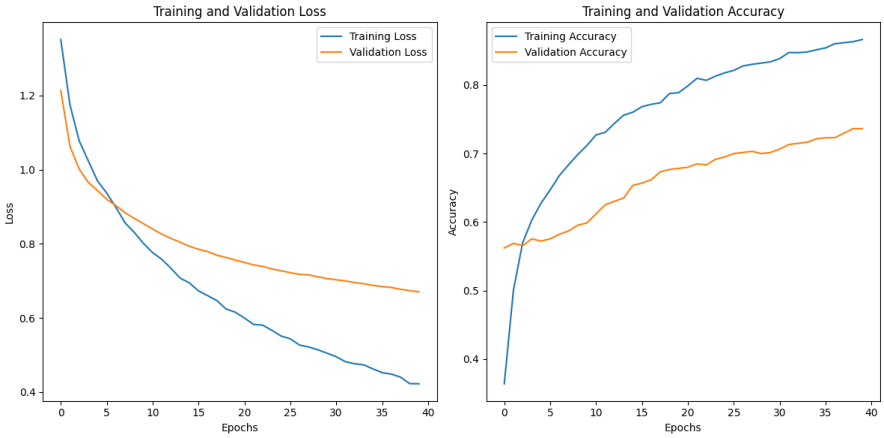
3.1 Przeprowadzić uczenie ostatniej warstwy splotowej wraz z częścią klasyfikującą

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

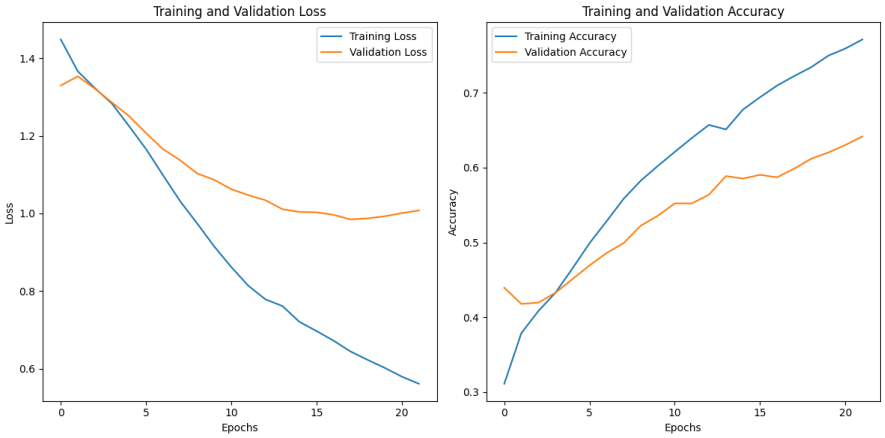
Model bardzo dobrze radzi sobie z wykrywaniem dronów, za to ma problem z wykrywaniem samolotów.

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:	**samolot (0)**	**dron (1)**	**helikopter (2)**	**ptak (3)**
samolot (0)	56	17	55	0
dron (1)	0	234	0	0
helikopter (2)	22	1	87	1
ptak (3)	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 ostało 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

Celem zadania trzeciego było wykonanie wszechstronnej wizualizacji i interpretacji wyników działania sieci głębokich (oraz klasyfikatorów wykorzystujących ich cechy). W szczególności zrealizowano następujące etapy:

1. **Analiza błędnych klasyfikacji (misclassifications)** – identyfikacja i omówienie przypadków, w których model się myli, oraz wskazanie możliwych przyczyn błędów.
2. **Wizualizacja obszarów uwagi (CAM, Class Activation Map)** – sprawdzenie, które fragmenty obrazu mają największy wpływ na odpowiedź modelu.
3. **Wizualizacja wewnętrznych warstw (DeepDream)** – zbadanie i ukazanie sposobu, w jaki wytrenowane filtry sieci reagują na specyficzne wzorce w danych.

4. **Testy z własnym zbiorem danych** – ocena jakości klasyfikatorów na nowym, niestandardowym zbiorze (również generowanym za pomocą **Fooocus v2.5.0**) oraz prezentacja metryk (Loss, Accuracy, macierze pomyłek, raporty klasyfikacji).

4.1 Analiza błędnych klasyfikacji

W pierwszym kroku dokonano predykcji klas dla wszystkich próbek z oryginalnego zbioru testowego (zadania 1 i 2), po czym porównano je z rzeczywistymi etykietami. Obrazy sklasyfikowane błędnie zapisywano do osobnego katalogu w celu ich pogłębionej inspekcji (`misclassified_images.zip`). Przykładowe błędnie sklasyfikowane obrazy dla wersji sieci z klasyfikatorem SVM - jądro poly wraz z etykietami:

Rysunek 7: od lewej: [true - samolot (0), predicted - helikopter (2)], [true - ptak (3), predicted - dron (1)]

Najczęstsze przyczyny błędów klasyfikacji to:

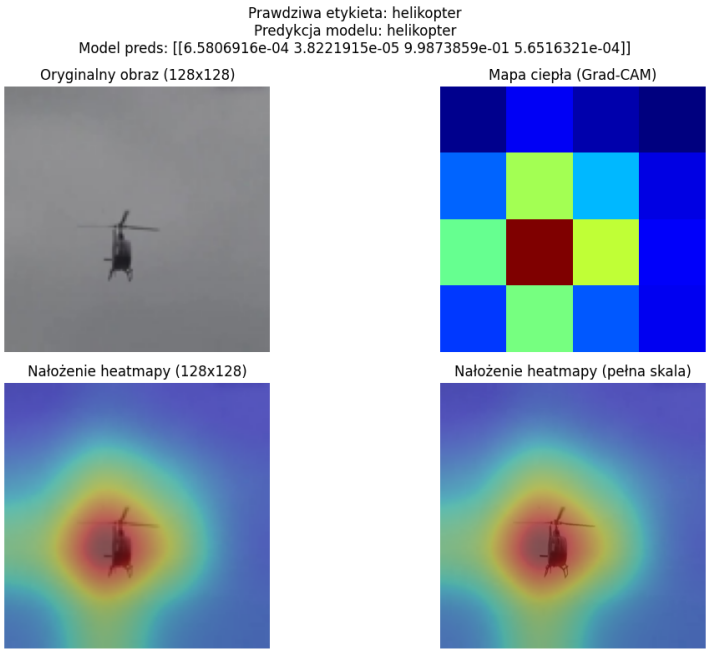
- **Niska jakość obrazu:** rozmyte lub nieostre zdjęcia, trudne do zinterpretowania nawet przez człowieka,
- **Podobieństwo klas:** daleki dron przypominający ptaka, nietypowy kształt helikoptera z zewnątrz wyglądający jak samolot,
- **Błędy etykiet (label noise):** istniejące w zbiorze przykłady mogą być niepoprawnie oznakowane,
- **Kontekst otoczenia:** sieć bywała zwodzona np. przez specyficzne tło (np. chmury, budynki).

4.2 Wizualizacja obszarów uwagi (CAM)

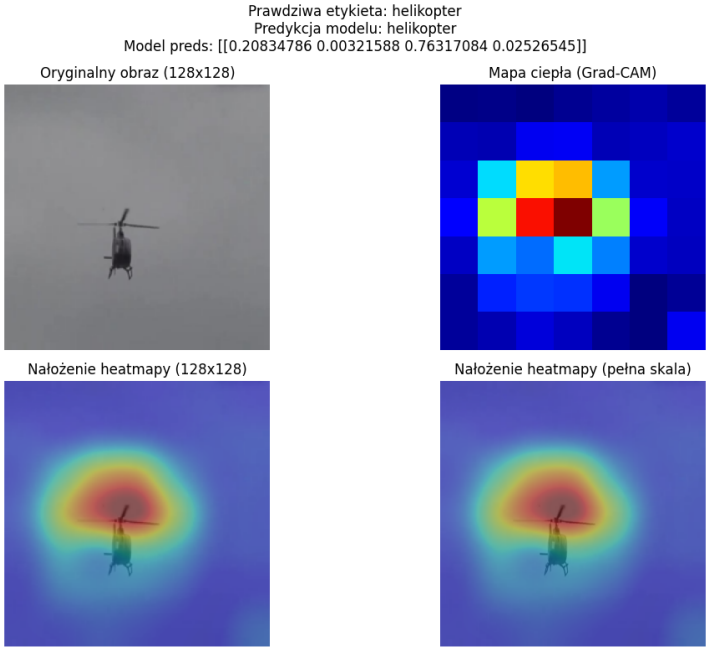
Aby określić, na które regiony obrazu sieć zwraca największą uwagę, wykorzystano technikę *Class Activation Map (CAM)*:

- Z modelu wybrano ostatnią warstwę splotową (`top_conv` w *EfficientNetB0*),
- Wyliczono gradienty od wyniku klasyfikacji do tejże warstwy, a następnie uśredniono (tzw. *global average pooling*),
- Otrzymaną mapę *heatmap* nałożono na oryginalny obraz (także w wersji zeskalowanej do 128×128).

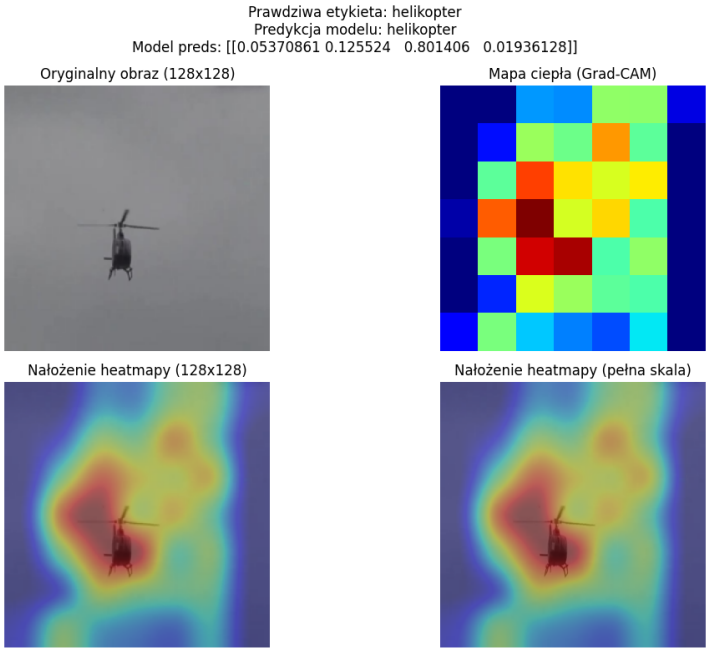
Dla wytrenowanych wersji sieci z zadania 1,2a,2b,2c wykonano wizualizację obszarów uwagi dla wybranego obrazu testowego (nr 521). Przykładowe wyniki przedstawiono poniżej:



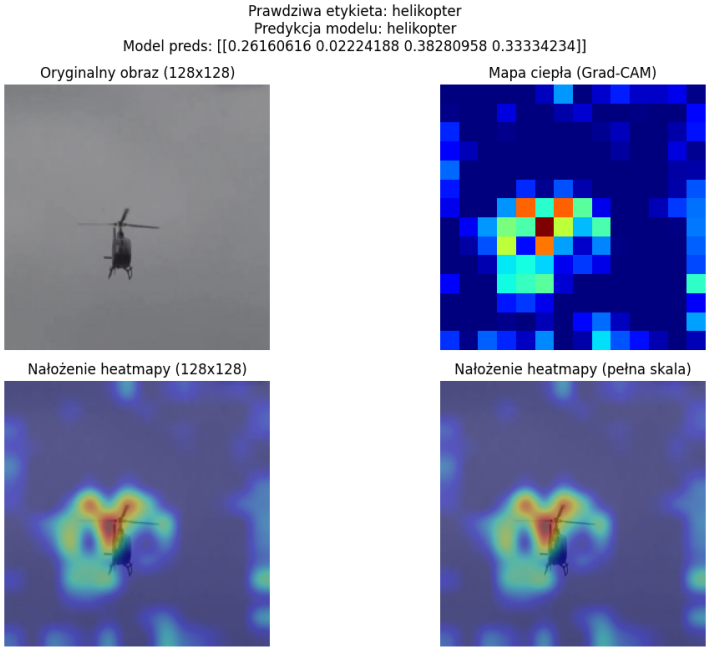
Rysunek 8: Przykładowy wynik CAM. Obraz testowy nr. 521 - model z zadania 1 (bez SVM).



Rysunek 9: Przykładowy wynik CAM. Obraz testowy nr. 521 - model z zadania 2a.



Rysunek 10: Przykładowy wynik CAM. Obraz testowy nr. 521 - model z zadania 2b.



Rysunek 11: Przykładowy wynik CAM. Obraz testowy nr. 521 - model z zadania 2c.

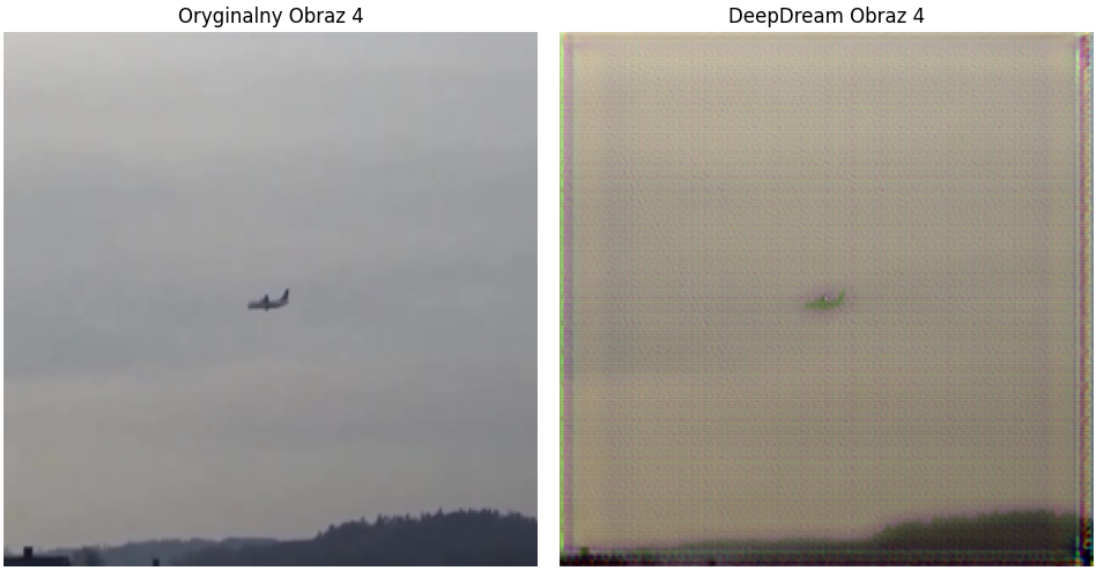
W prawidłowych klasyfikacjach sieć zwykle koncentruje się na obiekcie (np. dronie), natomiast w przypadku błędnych etykiet skupienie może paść na elementy tła i doprowadzić do mylnej decyzji.

4.3 Wizualizacja wewnętrznych warstw (DeepDream)

Technika **DeepDream** umożliwia wizualną eksplorację aktywacji głębszych warstw splotowych:

- Dla wybranych warstw (np. `block3a_activation`, `block5a_activation` i `block7a_activation` w *EfficientNetB0*) wykonuje się *gradient ascent* wzmacniający odpowiadające im filtry,
- Proces dzieli się na tzw. *octave scales*, by uwypuklić różne poziomy szczegółowości (skalowanie w górę obrazu w kolejnych krokach).

Wybrany obraz w wersji oryginalnej i po zastosowaniu DeepDream przedstawiono poniżej:

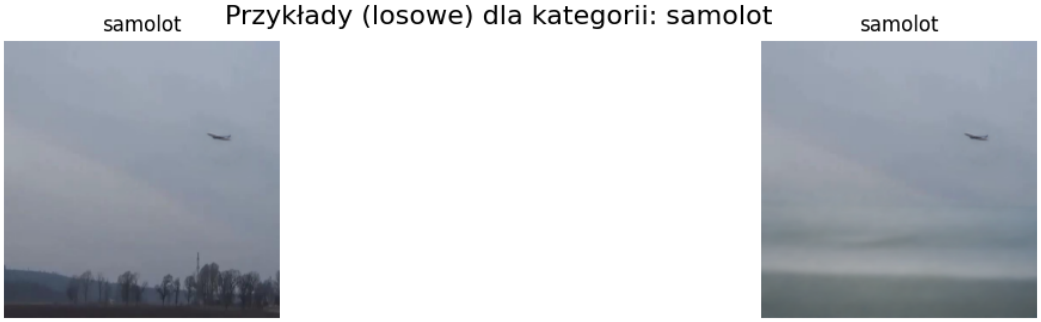


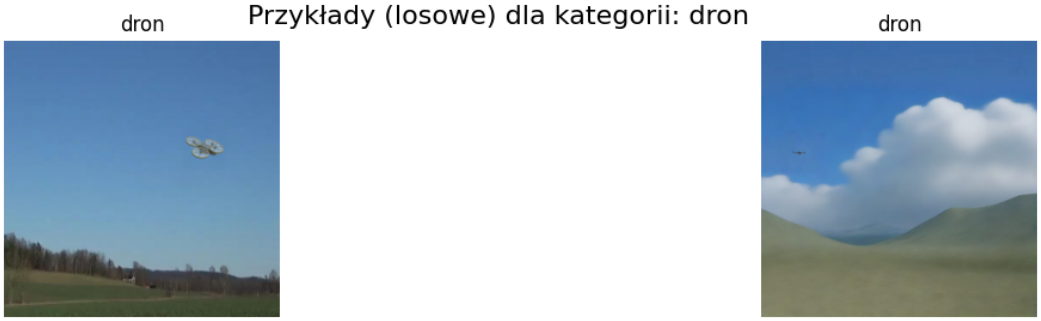
Rysunek 12: Obraz nr. 4 oryginalny (po lewej) i po zastosowaniu DeepDream (po prawej).

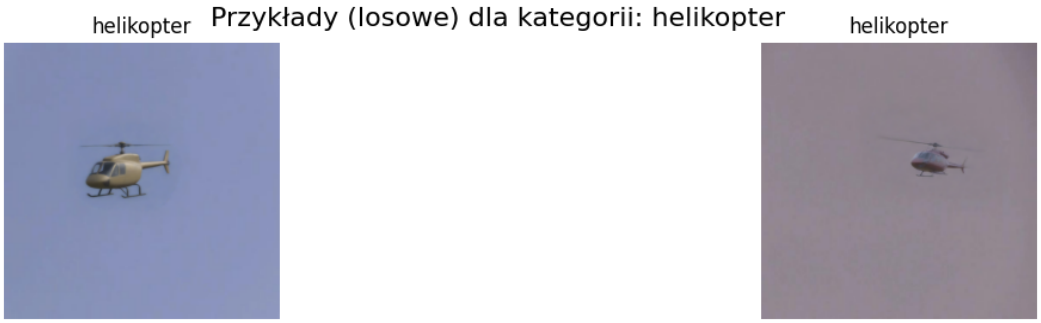
Jak widać na Rys. 12, sieć „uwydatnia" w obrazie wzory przypominające fraktale i abstrakcyjne tekstury, ujawniając tym samym swoje wewnętrzne detektory krawędzi i wzorców.

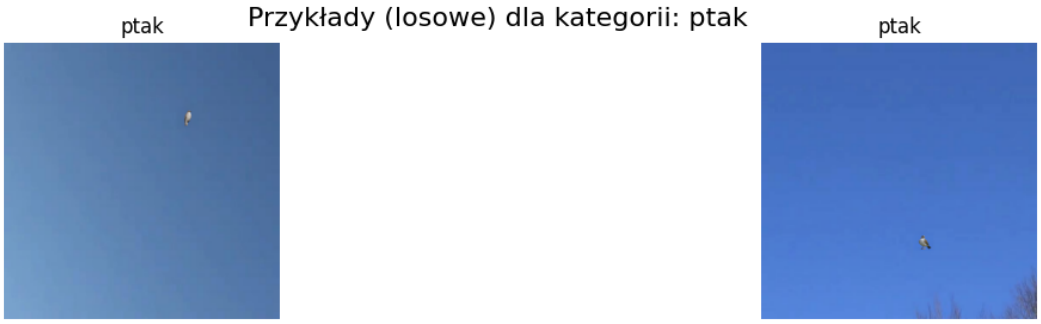
4.4 Testy z własnym zbiorem danych i szczegółowe wyniki

Aby zbadać uogólnianie modeli na dane odbiegające od oryginalnego podziału `train/valid/test`, wygenerowano **160 obrazów** (po 40 dla każdej kategorii: *samolot*, *dron*, *helikopter*, *ptak*) za pomocą programu **Fooocus v2.5.0**, działającego lokalnie na NVIDIA GeForce GTX 1070. Przykładowe obrazy z tego zbioru przedstawiono poniżej:









Przetestowano cztery finalne modele (z zadań 1 i 2). Poniżej przedstawiono zestawienia uzyskanych metryk, macierzy pomyłek i raportów klasyfikacji.

Model `umir_model_zad1.keras`

- **Loss:** 4.3591 **Accuracy:** 0.3125

Tabela 6: Macierz pomyłek (`umir_model_zad1`) na zbiorze 160 obrazów.

	samolot	**dron**	**helikopter**	**ptak**
samolot	0	0	40	0
dron	1	10	29	0
helikopter	0	0	40	0
ptak	0	0	40	0

Model `umir_model_zad2a.keras`

- **Loss:** 1.5077 **Accuracy:** 0.3938

Tabela 7: Macierz pomyłek (`umir_model_zad2a`) na zbiorze 160 obrazów.

	samolot	**dron**	**helikopter**	**ptak**
samolot	20	0	20	0
dron	17	5	18	0
helikopter	2	0	38	0
ptak	0	0	40	0

Model `umir_model_zad2b.keras`

- **Loss:** 2.4320 **Accuracy:** 0.2250

Tabela 8: Macierz pomyłek (`umir_model_zad2b`) na zbiorze 160 obrazów.

	samolot	**dron**	**helikopter**	**ptak**
samolot	0	0	40	0
dron	0	2	38	0
helikopter	0	6	34	0
ptak	0	0	40	0

Model `umir_model_zad2c.keras`

- **Loss:** 1.3587 **Accuracy:** 0.3750

Tabela 9: Macierz pomyłek (`umir_model_zad2c`) na zbiorze 160 obrazów.

	samolot	**dron**	**helikopter**	**ptak**
samolot	1	24	13	2
dron	17	19	3	1
helikopter	0	0	40	0
ptak	0	0	40	0

4.5 Wnioski z zadania 3

- **Analiza błędnych klasyfikacji** (Rys. 7) pomogła wyodrębnić najtrudniejsze przypadki (mała rozdzielczość, nietypowe kąty, stylizowane obrazy).
- **Wizualizacje CAM** (Rys. 11) potwierdziły, że poprawne predykcje wynikają najczęściej z właściwego ukierunkowania uwagi na docelowy obiekt (kadłub samolotu, drona, helikoptera). W przypadku błędów model często skupia się na elementach tła.
- **DeepDream** (Rys. 12) odsłonił charakterystyczne wzorce detektorów (np. krawędzi, faktur), które sieć „wzmacnia" w trakcie procesu *gradient ascent*.
- **Eksperyment z własnym zbiorem danych**, który uwzględniał **160 obrazów** z programu **Fooocus v2.5.0** (po 40 na klasę), pokazał słabości modeli w kontekście wysoce nietypowych, generowanych ujęć.
  - `umir_model_zad1` i `umir_model_zad2b` wypadły najsłabiej (w tym konkretnym eksperymencie) pod względem *Accuracy* (0.31 i 0.23), co wskazuje na potencjalną nieadekwatność treningu do tak różnorodnego stylu obrazów,
  - `umir_model_zad2a` i `umir_model_zad2c` osiągnęły nieco wyższe wyniki (ok. 0.39 i 0.38), choć ogólny poziom jest wciąż daleki od zadowalającego,
  - modele miewały szczególne problemy w rozróżnianiu *ptaków* i *dronów* w stylizowanych scenach — *recall* i *precision* dla tych klas były często niskie.

Wyniki te pokazują, że większe zróżnicowanie (i szersze dostrojenie) bazy treningowej lub bardziej rozbudowane sieci mogłyby poprawić sprawność klasyfikacji na *generowanych* (nietypowych) obrazach.

5 Podsumowanie i ocena narzędzi

W projekcie korzystaliśmy z Google Colab. Narzędzie to pozwala w wygodny sposób współpracować online i trenować modele bez posiadania własnej karty graficznej. Niestety darmowa wersja ma spore ograniczenia i czasami brakowało nam zasobów, przez co musieliśmy ograniczyć rozdzielczość zdjęć i czas uczenia. Google Colab korzysta z Notebooków, które pozwalają na wygodne dzielenie kodu na komórki i wykonywanie ich pojedynczo. Dzięki temu można łatwo testować różne fragmenty kodu i sprawdzać wyniki.

W projekcie korzystaliśmy z biblioteki TensorFlow, która pozwala na łatwe tworzenie modeli uczenia maszynowego. W naszym przypadku użyliśmy gotowego modelu EfficientNetB0, który jest dostępny w bibliotece Keras. Model ten jest bardzo wydajny i pozwala na uzyskanie dobrych wyników przy niewielkiej liczbie parametrów.

Do generowania własnych zdjęć wykorzystaliśmy narzędzie Foocus. Jest to zaawansowane narzędzie open source, które pozwala na generowanie zdjęć z różnymi parametrami na własnej karcie graficznej. Narzędzie to jest wygodne w użyciu i pozwala na generowanie dużej ilości zdjęć w krótkim czasie.