Klasyfikacja gatunków ptaków

z wykorzystaniem Transfer Learningu

pracę wykonał Bartłomiej Muranowicz

Spis treści

Spis treści 1

1. Czym jest transfer learning. 2

Dwufazowa strategia treningu 2

Dlaczego EfficientNet-B0? 2

2. Struktura projektu 2

Struktura danych 3

3. Jak zostały przebadane próbki 4

3.1 Dane wejściowe 4

3.1. Augmentacja danych 4

3.2. Faza 1 — Feature Extraction 4

3.3. Faza 2 — Fine-Tuning 5

3.4. Przebieg treningu 6

4. Rezultaty 7

4.1. Ogólne metryki 7

4.2. Training History 7

4.3. F1-Score per gatunek 8

4.4. Macierz pomyłek (Confusion Matrix) 9

4.5. Przykłady błędnych predykcji 10

4.6. Przykłady poprawnych predykcji 11

5. Jak można bardziej usprawnić model? 12

5.1. Większy i lepszy zbiór danych 12

5.2. Większy model bazowy 12

5.3. Zaawansowana augmentacja 12

5.4. Test Time Augmentation (TTA) 12

5.5. Ensemble modeli 12

5.6. Trening na GPU 12

1. Czym jest transfer learning.

Transfer learning to technika uczenia maszynowego, w której model wytrenowany na dużym zbiorze danych jest adaptowany do nowego, mniejszego zadania. Zamiast trenować sieć neuronową od zera, wykorzystujemy wiedzę, którą model już zdobył.

W tym projekcie wykorzystano model EfficientNet-B0 pretrenowany na zbiorze ImageNet (1.2 miliona obrazów, 1000 klas). Model ten nauczył się rozpoznawać podstawowe cechy wizualne — krawędzie, tekstury, kształty — które są uniwersalne i przydatne również w rozpoznawaniu ptaków.

Dwufazowa strategia treningu

Trening przebiegał w dwóch fazach:

Faza 1 — Feature Extraction: Warstwy bazowe modelu zostały zamrożone (ich wagi się nie zmieniały). Trenowana była wyłącznie nowa warstwa klasyfikacyjna, która uczyła się mapować cechy z EfficientNet na nasze gatunki ptaków.

Faza 2 — Fine-Tuning: Odmrożono ostatnie 3 bloki modelu i całość była delikatnie dostrajana z niższym learning rate, aby nie zniszczyć wcześniej nauczonych cech.

Dlaczego EfficientNet-B0?

EfficientNet-B0 został wybrany ze względu na najlepszy stosunek jakości do rozmiaru:

Model	Parametry	Top-1 Acc (ImageNet)	Rozmiar
ResNet-50	25.6M	76.1%	98 MB
EfficientNet-B0	5.3M	77.1%	21 MB
VGG-16	138M	71.6%	528 MB

Mniejszy model oznacza szybszy trening i mniejsze wymagania sprzętowe, przy jednocześnie lepszych wynikach niż ResNet-50.

2. Struktura projektu

Projekt składa się z następujących elementów:

Plik	Funkcja
split_dataset.py	Podział datasetu CUB-200-2011 na zbiory train/val/test
select_species.py	Ograniczenie datasetu do wybranych 30 gatunków
train.py	Trening modelu — dwie fazy: Feature Extraction i Fine-Tuning
predict.py	Ewaluacja wytrenowanego modelu na zbiorze testowym
requirements.txt	Lista zależności Python

Struktura danych

Dane zorganizowane są w katalogi — każdy folder odpowiada jednemu gatunkowi. ImageFolder z PyTorch automatycznie przypisuje etykiety na podstawie nazw folderów.

Zbiór	Przeznaczenie	Udział
train/	Dane treningowe — model się na nich uczy	69.8%
val/	Dane walidacyjne — kontrola jakości w trakcie treningu	14.7%
test/	Dane testowe — końcowa ocena modelu (nigdy nie widziane)	15.5%

3. Jak zostały przebadane próbki

3.1 Dane wejściowe

Liczba załadowanych danych wejściowych:

- train : 1238 obrazów
- val : 259 obrazów
- test : 276 obrazów

Dane pochodzą z datasetu CUB-200-2011, ograniczonego do 30 wybranych gatunków ptaków. Każde zdjęcie zostało przeskalowane do rozmiaru 224×224 pikseli i znormalizowane wartościami ImageNet.

3.1. Augmentacja danych

Na danych treningowych zastosowano augmentację — losowe transformacje zwiększające różnorodność danych i zapobiegające przeuczeniu modelu:

Transformacja	Opis
RandomResizedCrop (224px)	Losowe przycięcie fragmentu obrazu (80–100% powierzchni)
RandomHorizontalFlip (50%)	Losowe odbicie poziome zdjęcia
RandomRotation (±15°)	Losowy obrót obrazu
ColorJitter	Losowa zmiana jasności, kontrastu, nasycenia i odcienia
Normalize (ImageNet)	Normalizacja do wartości oczekiwanych przez pretrenowany model

Augmentacja NIE była stosowana na danych walidacyjnych i testowych — te zostały jedynie przeskalowane i znormalizowane.

3.2. Faza 1 — Feature Extraction

W pierwszej fazie zamrożono wszystkie warstwy bazowe EfficientNet-B0. Trenowana była wyłącznie nowa warstwa klasyfikacyjna. Trening trwał 5 epok z learning rate 0.001 i batch size 16.

Epoka	Train Loss	Train Acc	Val Loss	Val Acc	Czas
1	2.7629	28.76%	1.6234	66.02%	57.7s
2	1.4655	60.10%	0.9898	76.06%	54.0s
3	1.0933	68.50%	0.8165	79.54%	50.7s
4	0.9627	71.24%	0.7734	80.31%	51.9s
5	0.8763	72.13%	0.7323	80.31%	51.9s

Najlepszy wynik Fazy 1: Val Accuracy = 80.31% (epoka 4). Model osiągnął 80% dokładności trenując jedynie nową warstwę klasyfikacyjną — pretrenowane cechy z ImageNet okazały się bardzo przydatne.

3.3. Faza 2 — Fine-Tuning

W drugiej fazie odmrożono ostatnie 3 bloki EfficientNet-B0, zwiększając liczbę trenowalnych parametrów z około 670 tysięcy do 3 827 002 (z 4 678 810 łącznie). Zastosowano niższy learning rate (0.0001) z różnicowaniem — warstwy bazowe otrzymały 10x niższy LR niż klasyfikator.

Epoka	Train Loss	Train Acc	Val Loss	Val Acc	Czas
1	0.7187	79.56%	0.6790	83.01%	67.4s
2	0.6548	80.05%	0.5858	84.56%	67.9s
3	0.6512	80.61%	0.6065	83.78%	69.9s
4	0.5668	84.41%	0.5403	85.71%	66.2s
5	0.5398	84.25%	0.5274	86.49%	63.7s
6	0.5493	84.41%	0.4923	87.64%	68.8s
7	0.5294	84.98%	0.4995	87.26%	66.8s
8	0.4743	86.19%	0.4648	86.49%	63.7s
9	0.4462	87.32%	0.4947	86.87%	70.1s

10	0.4361	87.40%	0.4514	88.80%	61.8s
11	0.4672	85.70%	0.4545	88.80%	68.4s
12	0.4038	88.05%	0.4421	89.19%	60.1s
13	0.3910	89.10%	0.4273	90.35%	59.0s
14	0.4326	87.64%	0.4206	89.96%	58.8s
15	0.3983	89.34%	0.4307	87.64%	69.0s
16	0.3734	88.93%	0.4521	88.03%	74.7s
17	0.3744	89.58%	0.4201	89.19%	75.8s
18	0.3983	88.85%	0.4269	89.19%	76.4s

Najlepszy wynik Fazy 2: Val Accuracy = 90.35% (epoka 13/18). Early Stopping zatrzymał trening po 5 epokach bez poprawy. Fine-tuning poprawił wynik o **+10.04 punktów procentowych** względem Fazy 1.

3.4. Przebieg treningu

Parametr	Faza 1 (Feature Extraction)	Faza 2 (Fine-Tuning)
Epoki (wykonane)	5	18 (z 20, Early Stopping)
Learning Rate	0.001	0.0001 (bazowe: 0.00001)
Batch size	16	16
Trenowane parametry	~670 tys. (klasyfikator)	3 827 002 (81.8% modelu)
Najlepsza Val Acc	80.31%	90.35%
Czas na epokę	~52s	~66s
Łączny czas	~4 min 25s	~19 min 50s
Urządzenie	CPU	CPU

Łączny czas treningu wyniósł około 24 minuty na CPU

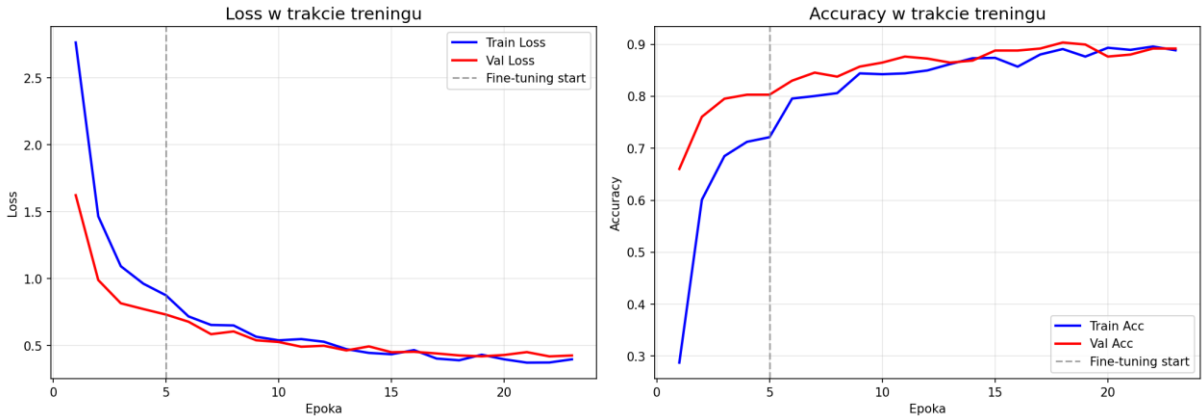
4. Rezultaty

4.1. Ogólne metryki

Metryka	Wartość
Top-1 Accuracy	0.9130434782608695
Top-5 Accuracy	0.9855072463768116
Średni Precision	0.9210858585858586
Średni Recall	0.9148148148148149
Średni F1-Score	0.9151787147607582
Liczba obrazów testowych	276
Liczba błędów	24

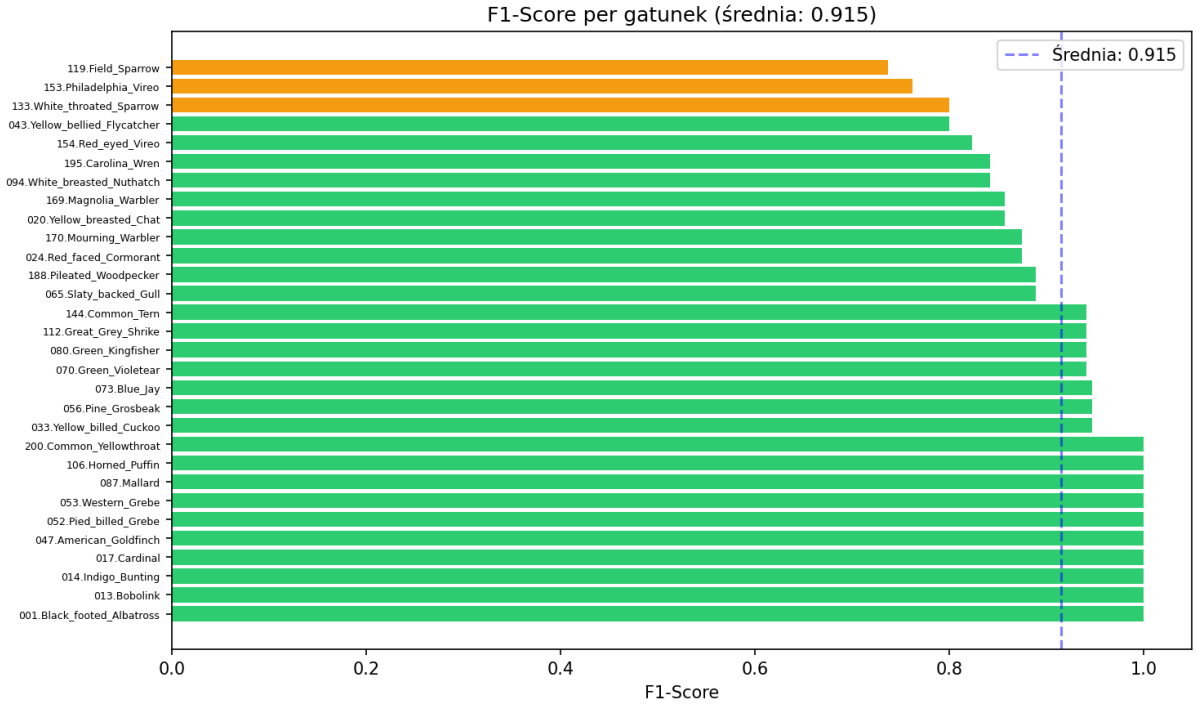
4.2. Training History

Poniższy wykres przedstawia przebieg treningu. Pionowa linia oznacza przejście z Feature Extraction (epoki 1–5) do Fine-Tuning (epoki 6–23). Widać wyraźny spadek loss i wzrost accuracy po odmrożeniu warstw.



4.3. F1-Score per gatunek

Wykres F1-Score pozwala zidentyfikować, które gatunki model rozpoznaje najlepiej, a z którymi ma trudności.



Top 5 najlepiej rozpoznanych gatunków

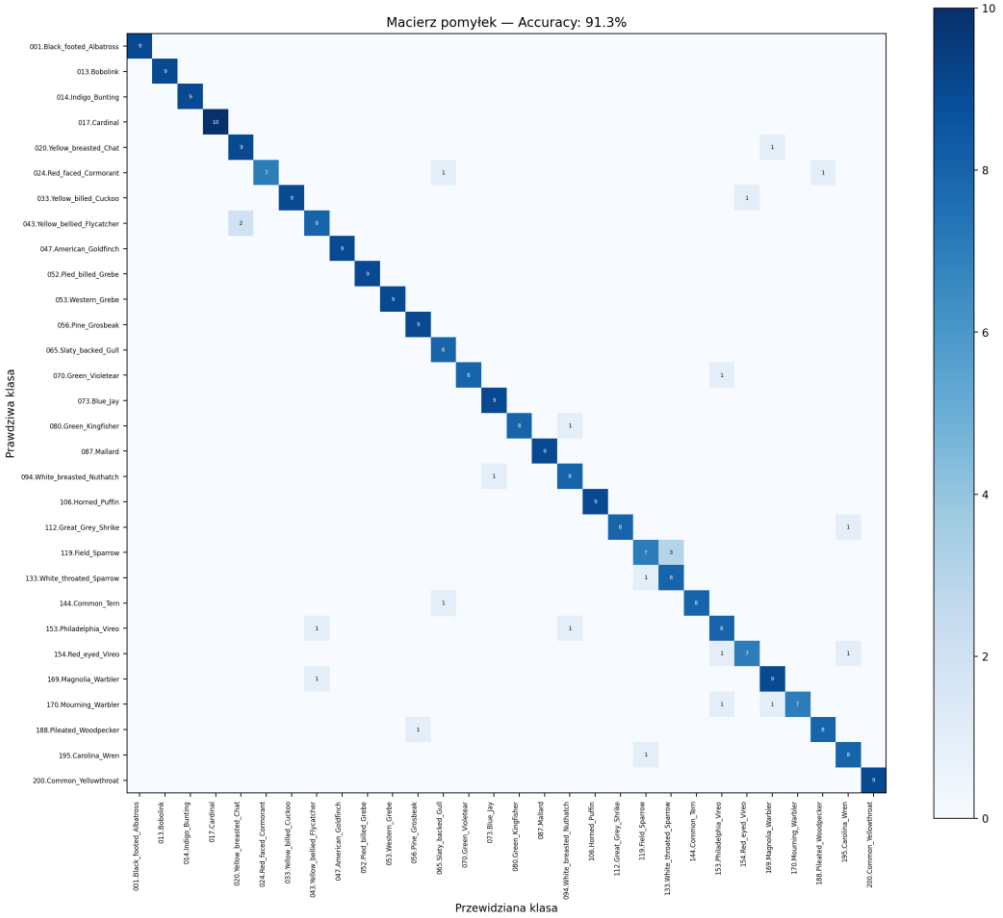
- 001.Black_footed_Albatross F1: 1.000 (9/9)
- 013.Bobolink F1: 1.000 (9/9)
- 014.Indigo_Bunting F1: 1.000 (9/9)
- 017.Cardinal F1: 1.000 (10/10)
- 047.American_Goldfinch F1: 1.000 (9/9)

Top 5 najgorzej rozpoznanych gatunków

- 154.Red_eyed_Vireo F1: 0.824 (7/9)
- 043.Yellow_bellied_Flycatcher F1: 0.800 (8/10)
- 133.White_throated_Sparrow F1: 0.800 (8/9)
- 153.Philadelphia_Vireo F1: 0.762 (8/10)
- 119.Field_Sparrow F1: 0.737 (7/10)

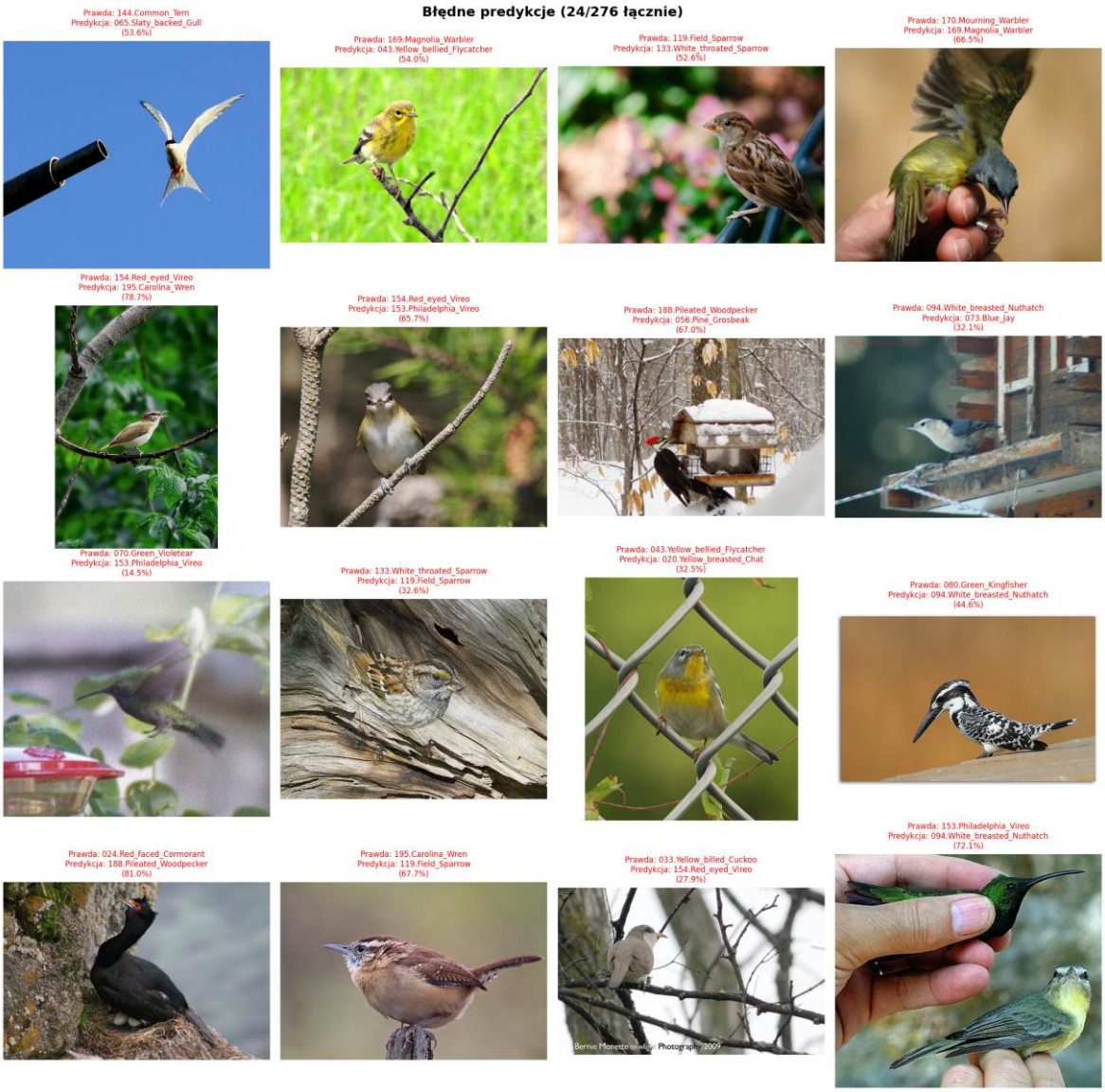
4.4. Macierz pomyłek (Confusion Matrix)

Macierz pomyłek pokazuje, jak model klasyfikował każdy gatunek. Wartości na przekątnej to poprawne predykcje, poza przekątną — błędy.



4.5. Przykłady błędnych predykcji

Poniżej przykłady zdjęć sklasyfikowanych niepoprawnie. Każde zdjęcie zawiera prawdziwą klasę, przewidzianą klasę oraz pewność modelu.



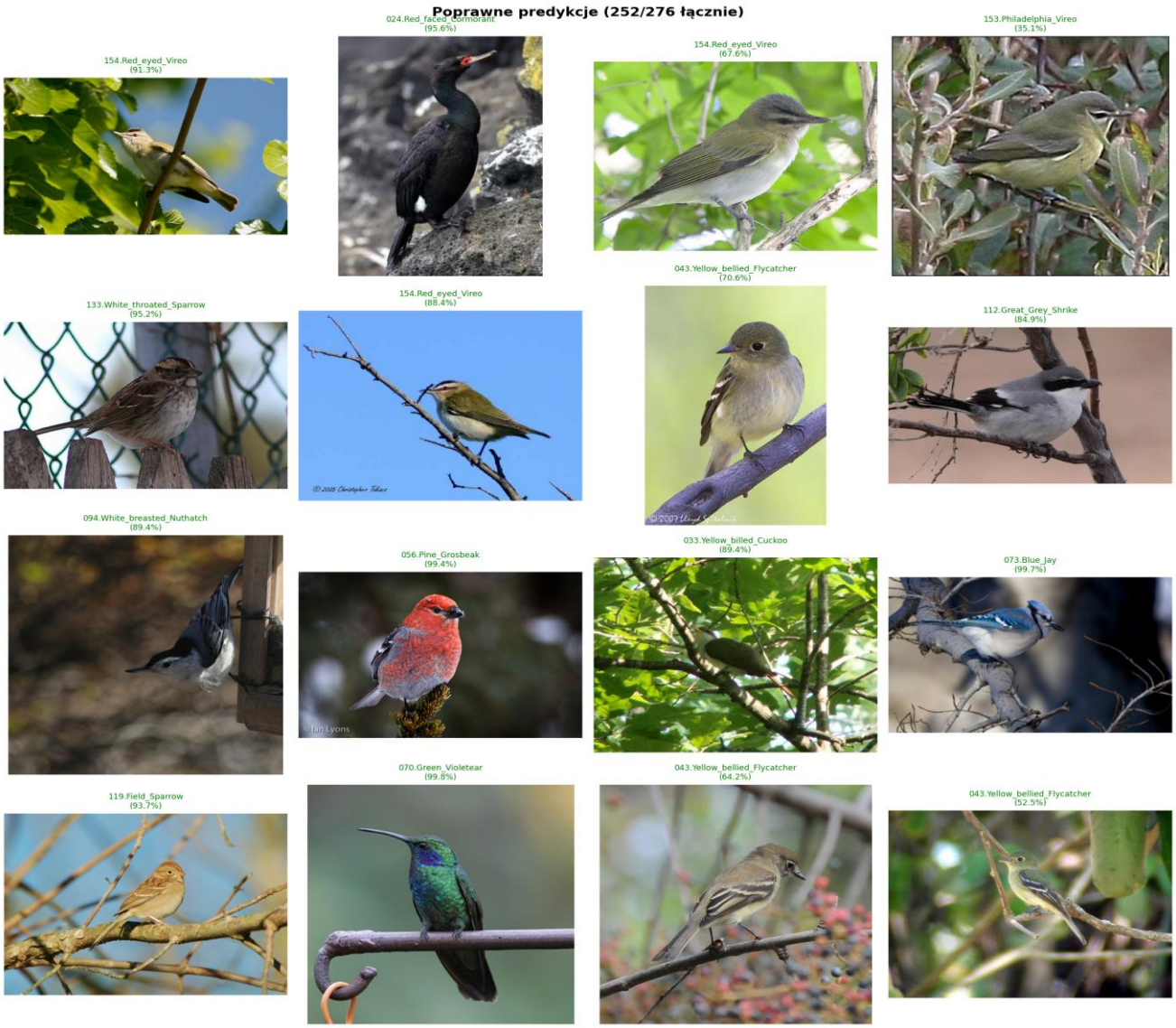
Model popełnił 24 błędy na 276 zdjęć testowych. Analiza błędnych predykcji ujawnia kilka wzorców:

Najczęstszą przyczyną pomyłek było podobieństwo wizualne między gatunkami — szczególnie wśród małych ptaków o zbliżonym ubarwieniu. Model mylił ze sobą gatunki vireo (Red-eyed Vireo, Philadelphia Vireo), wróblowate (Field Sparrow, White-throated Sparrow) oraz pokrzewki (Magnolia Warbler, Mourning Warbler). Są to gatunki, które nawet doświadczonym ornitologom sprawiają trudność w identyfikacji na podstawie samego zdjęcia.

Część błędów wynikała z trudnych warunków zdjęcia — ptak w locie (Common Tern), częściowo zasłonięty (Yellow-bellied Flycatcher przy karmniku), nietypowa poza lub odległy kadr. W kilku przypadkach model wykazywał niską pewność predykcji (14.5% dla Green Violetear, 27.9% dla Yellow-billed Cuckoo), co sugeruje, że sam „nie był pewny” swojej odpowiedzi.

Warto zauważyć, że niektóre błędy miały wysoką pewność (81% dla Red-faced Cormorant → Pileated Woodpecker), co wskazuje na przypadki, gdzie model był przekonany o błędnej klasyfikacji — prawdopodobnie z powodu nietypowego ujęcia ptaka.

4.6. Przykłady poprawnych predykcji



5. Jak można bardziej usprawnić model?

Osiągnięty wynik 91.3% dokładności na zbiorze testowym jest solidny, jednak istnieje kilka sposobów na dalszą poprawę modelu.

5.1. Większy i lepszy zbiór danych

CUB-200-2011 zawiera stosunkowo niewiele zdjęć na gatunek. Wykorzystanie większego datasetu, takiego jak Birds 525 Species z Kaggle (~90 000 obrazów), dostarczyłoby modelowi więcej przykładów do nauki, co bezpośrednio przełożyłoby się na lepszą generalizację.

5.2. Większy model bazowy

EfficientNet-B0 to najmniejszy model z rodziny EfficientNet. Zastosowanie większych wariantów, takich jak EfficientNet-B3 lub B4, zwiększyłoby liczbę parametrów i zdolność modelu do wychwytywania subtelnych różnic między podobnymi gatunkami (np. vireo czy wróblowate). Wymaga to jednak więcej pamięci i mocy obliczeniowej — najlepiej trenować na GPU.

5.3. Zaawansowana augmentacja

Obecna augmentacja obejmuje podstawowe transformacje (obrót, odbicie, zmiana kolorów). Bardziej zaawansowane techniki mogłyby poprawić odporność modelu: Mixup (mieszanie dwóch obrazów i ich etykiet), CutMix (wklejanie fragmentów jednego obrazu w drugi) oraz RandomErasing (losowe zakrywanie części obrazu). Techniki te zmuszają model do skupienia się na większej liczbie cech, a nie tylko na jednym dominującym elemencie zdjęcia.

5.4. Test Time Augmentation (TTA)

Podczas predykcji można zastosować augmentację również na etapie testowania. Zamiast klasyfikować jedno zdjęcie, tworzymy kilka jego wariantów (obrócony, odbity, przycięty) i uśredniamy predykcje. Dzięki temu model podejmuje decyzję na podstawie wielu perspektyw tego samego obrazu, co zwykle podnosi dokładność o 1–3 punkty procentowe.

5.5. Ensemble modeli

Połączenie predykcji z kilku niezależnie wytrenowanych modeli (np. EfficientNet-B0 + ResNet-50 + MobileNet) pozwala kompensować słabości poszczególnych architektur. Każdy model może „widzieć” inne cechy, a ich wspólna decyzja jest zazwyczaj trafniejsza niż decyzja pojedynczego modelu.

5.6. Trening na GPU

Obecny trening odbywał się na CPU, co ograniczało możliwości eksperymentowania. Wykorzystanie GPU (np. przez Google Colab lub kartę

NVIDIA) pozwoliłoby na trening większych modeli, większych batch sizeów i większą liczbę epok w znacznie krótszym czasie — co umożliwiłoby szybsze testowanie różnych konfiguracji i hiperparametrów.