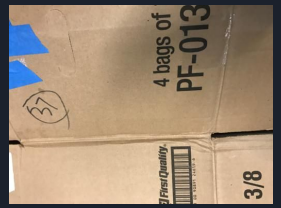
Porównanie metod klasyfikacji obrazów w rozpoznawaniu typu odpadu komunalnego

Sieć konwolucyjna, SVM (Support Vector Machine) z transfer-learning oraz Naive Bayes i k-NN za pomocą histogramów koloru.

Baza zdjęć - TrashNet (~2500 zdjęć).

6 rodzajów odpadów:

Cardboard (403)



Metal (410)

Plastic (482)

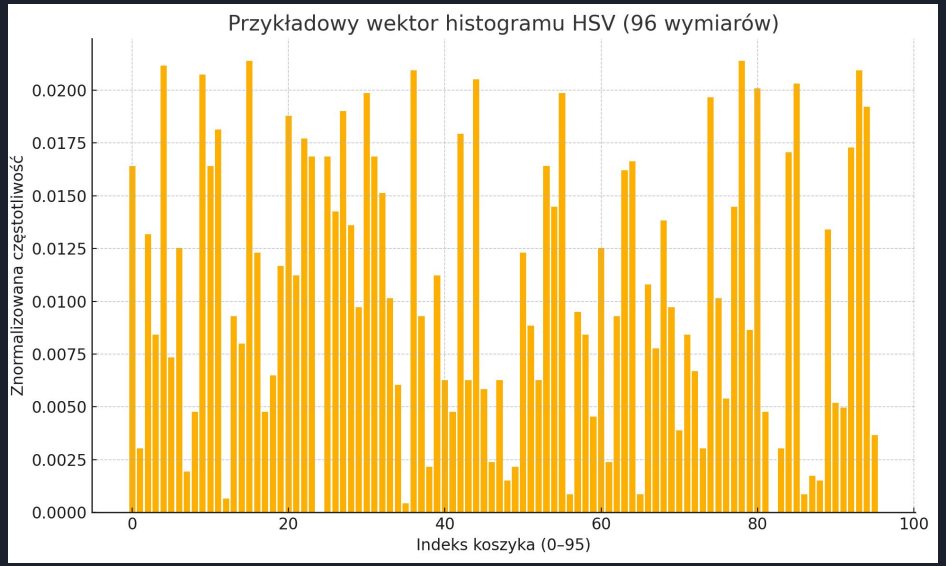
Glass (501)

Paper (594)

Trash (137)

Preprocessing dla k-NN i Naive Bayes

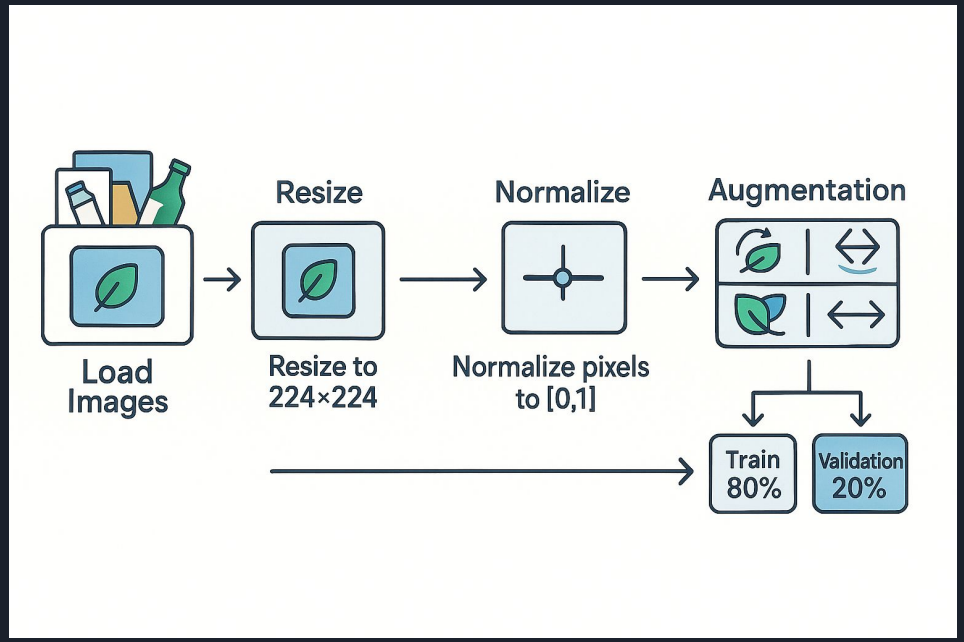
- Wczytanie obrazów z dysku i skalowanie do 224x224px.
- Konwersja z RGB na HSV. H (Hue) – odcień (0–180), S (Saturation) – nasycenie (0–255), V (Value) – jasność (0–255).
- Tworzymy histogram dla każdego kanału (wektor 32 wartości, ilość pikseli w danych przedziale.
- Łączymy wektory trzech kanałów w jeden wektor 96 wartości.
- Normalizujemy - dzielimy każdą wartość przez sumę wartości wektora (sumują się do 1).
- Dzielimy zbiór na 80% danych treningowych i 20% testowych.



Osie 0–31 → histogram dla odcienia (H), 32–63 → nasycenie (S), 64–95 → wartość/jasność (V).

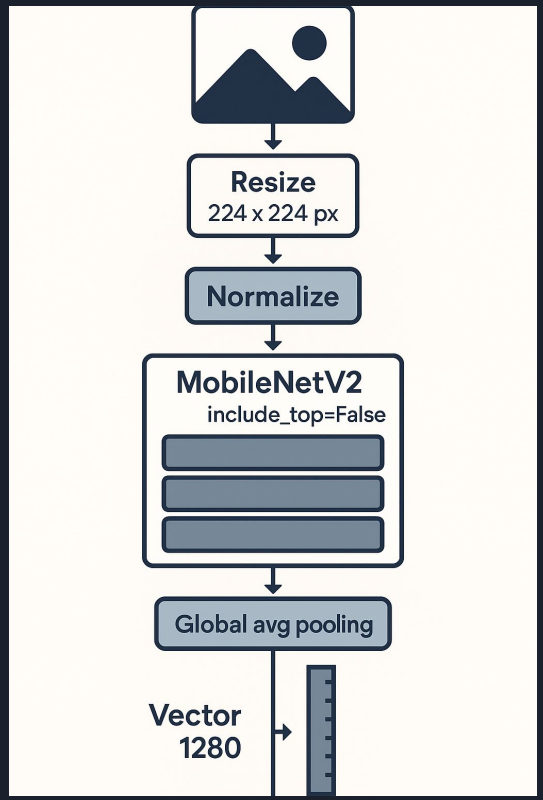
Preprocessing w CNN od zera

- Wszystkie obrazy z katalogu klas ładowane są i przeskalowane do 224x224 px.
- Piksele dzielone przez 255 → wartości w zakresie [0.0, 1.0].
-
- Augmentacja on-the-fly:
  - Obrót losowy w zakresie ±15°.
  - Przesunięcie w poziomie/pionie do ±10% rozmiaru obrazu.
  - Odbicie poziome z prawdopodobieństwem 50%.
- Dzielimy zbiór na 80% danych treningowych i 20% testowych.



Preprocessing dla SVM, używając sieci MobileNetV2 (transfer-learning)

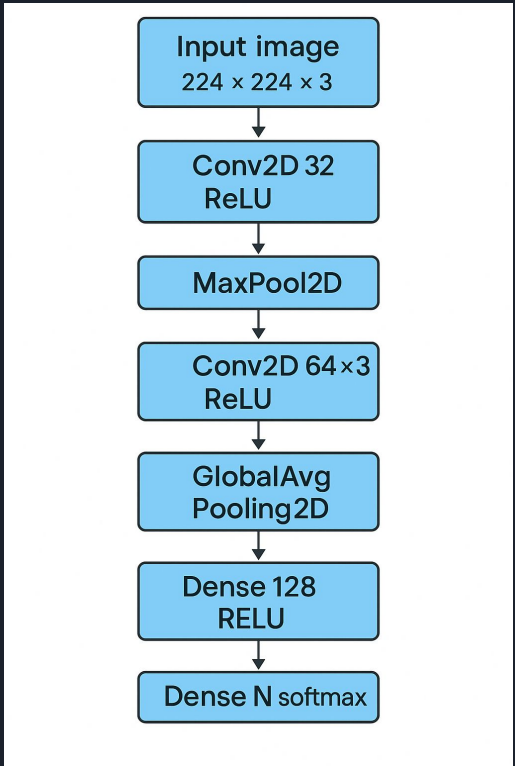
- Skalujemy obrazy do rozmiaru 224x224px.
- Używamy Keras-owego preprocess_input (ImageNet):
  - centrowanie po kanałach (od każdej wartości piksela w kanale R, G i B odejmujemy średnią wartość tego kanału).
  - skalowanie tak, by wartości odpowiadały oryginalnemu treningowi MobileNetV2
  - (przeskalowania pikseli z zakresu [0–255] do [−1…+1], dzielenie przez 127.5 i przesunięcie o 1).
- Przepuszczamy obraz przez MobileNetV2 (include_top=False, pooling='avg'). Usuwamy 1000-klasowy head (softmax) sieci pretrenowanej na ImageNet.
- Zamrożone wszystkie wagi (nie uczymy ich dalej).
- Otrzymujemy wektor cech 1×1280 opisujący treść obrazu.



Budowa mojej sieci CNN.

Architektura:

- Input:
  - obrazy 224×224×3 (RGB).
- Blok 1:
  - Conv2D(32, 3×3) + ReLU
  - MaxPool 2x2
- Blok 2:
  - Conv2D(64, 3x3) + ReLU
  - MaxPool 2x2
- Blok 3:
  - Conv2D(128, 3×3) + ReLU
  - GlobalAvgPool → wektor 128D
- Głowa klasyfikatora:
  - Dense(128) + ReLU
  - Dense(num_classes) + Softmax → rozkład prawdopodobieństw



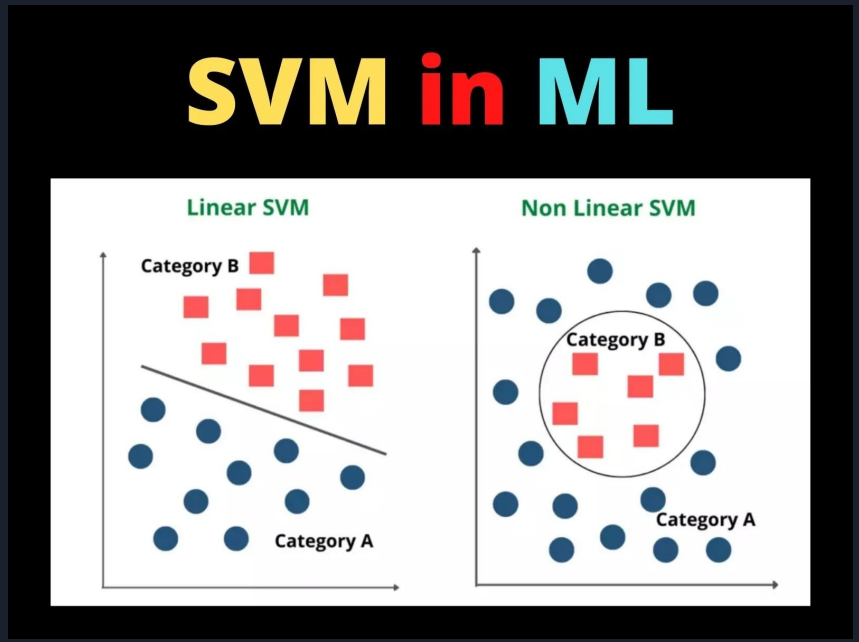
Trening i zapisywanie wag:

- Optymalizator
  - Adam
- Strata
  - Categorical Crossentropy
- Batch size:
  - 32
- Epochs:
  - 100

Jak działa SVM (Support Vector Machine)?

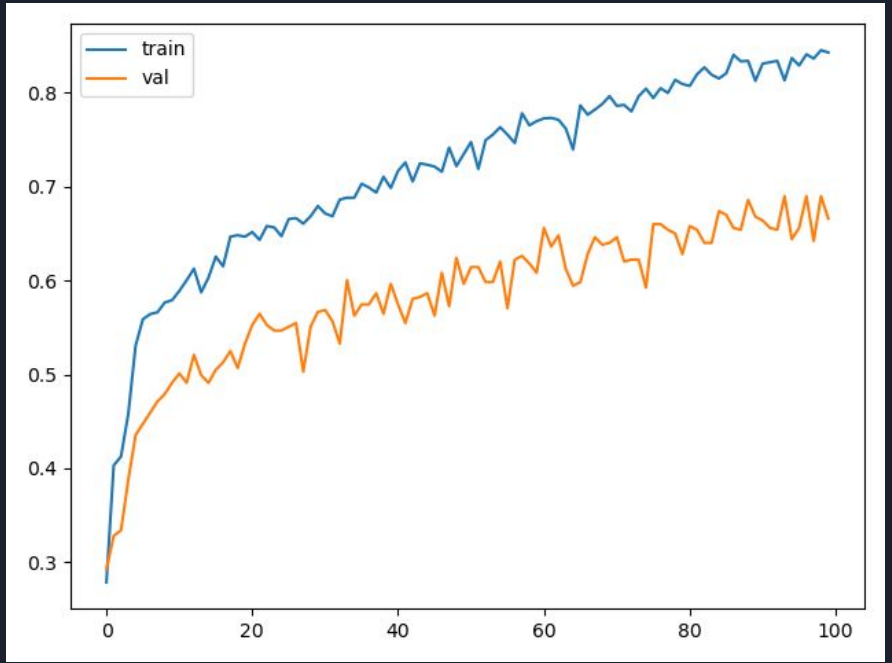
Celem SVM jest narysować taką „linię” (w wielowymiarowej przestrzeni – tzw. hiperpłaszczyznę), która najlepiej oddziela punkty z dwóch klas. Najlepiej” oznacza: linia ta jest odsunięta jak najdalej od najbliższych punktów obu klas – tworzymy więc największy możliwy „margines” między klasami.

- Soft Margin:
  - Dopuszcza niewielkie naruszenia marginesu, karane parametrem C:
    - Małe C → większy margines, więcej naruszeń
    - Duże C → ścisłe dopasowanie, mniejszy margines
- Support Vectors:
  - Podczas uczenia SVM szuka tej idealnej granicy, jednocześnie dopuszczając, by kilka punktów znalazło się w marginesie lub po niewłaściwej stronie granicy.
  - leżą najbliżej granicy (na samym brzegu marginesu) lub nawet naruszają margines (są po „złej” stronie).
- Predykcja:
  - Dla nowego punktu SVM sprawdza, z którymi support vectors jest najbardziej „podobny” (przez funkcję jądra, np. RBF),
  - Następnie na podstawie tych najbliższych support vectors decyduje, po której stronie granicy ten punkt leży → etykieta klasy.



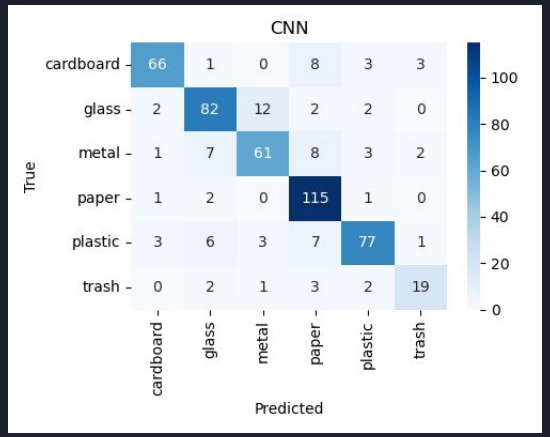
Krzywa uczenia mojego CNN

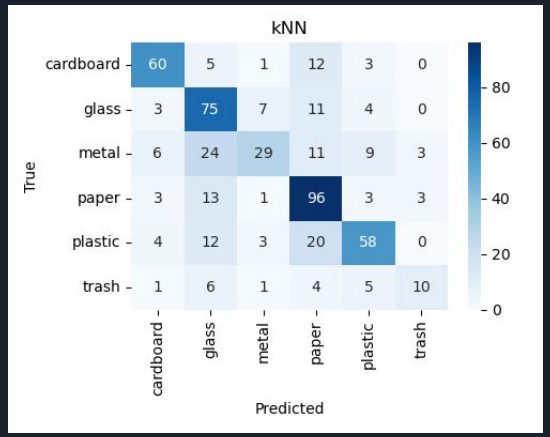
- Szybki wzrost na początku:
  - W 5 epokach train_acc: ~0.28→0.55, val_acc: ~0.28→0.50.
- Divergencja po ~10 epokach:
  - train_acc ≈0.65 vs. val_acc ≈0.50 – zaczyna się zbytnie dopasowanie do treningu.
- Wyraźny overfitting:
  - Końcowo train_acc≈0.85, val_acc≈0.65 (20–25 p.p. różnicy).
- Fluktuacje walidacji:
  - Nieregularne skoki val_acc →potencjalnie zbyt mały zbiór walidacyjny lub zbyt agresywna augmentacja

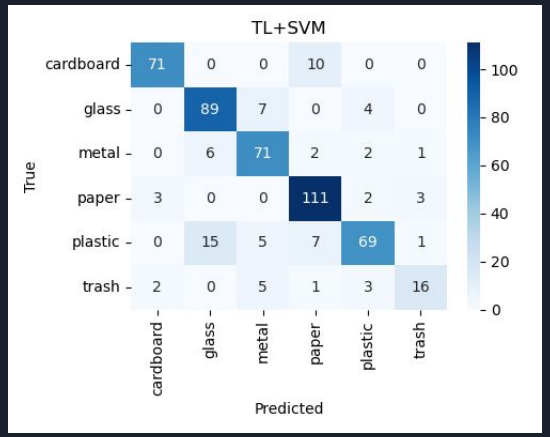


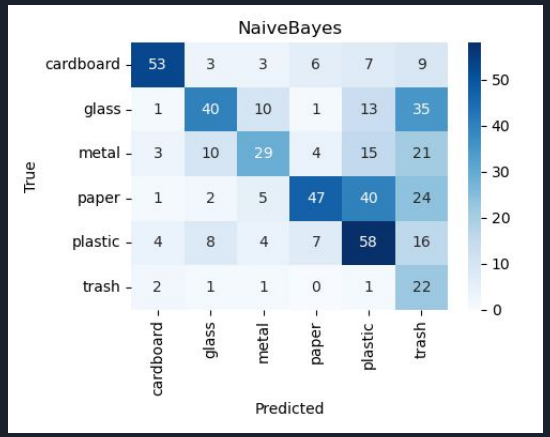
Podsumowanie:
Szybkie uczenie, ale rosnąca różnica między train/val to sygnał overfittingu – wprowadzenie early stoppingu i regularizacji może poprawić generalizację.

Macierze błędów klasyfikatorów









Porównanie wyników

Model	Średnia trafień na diagonalnej	Najlepsza klasa	Najgorsza klasa	Typowe pomyłki
TL+SVM	~81 %	paper (111 / 119)	trash (16 / 27)	plastic→glass, metal→glass
CNN	~78 %	paper (115 / 119)	trash (19 / 27)	metal→glass, plastic→glass
k-NN	~71 %	paper (96 / 119)	metal (29 / 52)	cardboard→paper, glass→paper
NaiveBayes	~55 %	plastic (58 / 97)	glass (40 / 80)	glass→trash, paper→plastic

Najlepszy wynik

TL+SVM:
Silna diagonalna dominacja: duża część próbek każdej klasy trafiona poprawnie.

Główne błędy:
plastic ↔ glass (np. 15 plastików uznanych za szkło),
metal ↔ glass (6 metal → glass).

Niewiele gorszy

CNN:
Paper niemal doskonale (115/119),

Nadal myli metal ze szkłem (7 prób) i plastik ze szkłem (6),

Trash lepiej niż k-NN/NB, ale ciągle trudna klasa.

Umiarkowanie dobry

k-NN:
Spore problemy z metal (29/52) i glass (75/100).

Cardboard i paper najstabilniejsze, ale znacznie gorzej niż TL+SVM.

Najsłabszy

Naive Bayes:
Dobre only dla plastic (58/97),

Dramatyczne pomyłki glass→trash (35/80) i paper→plastic (40/119).

Zakres kolorów w histogramach nie rozdziela dobrze wszystkich klas.

Końcowe wnioski

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Model	Accuracy	Macro-F1	Train time (s)
TL+SVM	0.8439	0.8203	4.3400
CNN (from scratch)	0.8300	0.8143	1402.6700
k-NN (HSV)	0.6482	0.6137	0.0011
Naive Bayes (HSV)	0.4921	0.4917	0.0016

Najlepszy kompromis – TL+SVM:

-Accuracy 84 %, macro-F1 0.82 przy kilku sekundach trenowania.

-Ekstrakcja 1280-wymiarowych cech z MobileNetV2 + szybki SVM daje świetne rezultaty bez kosztu długiego treningu sieci.

CNN od zera:

-Bardzo dobra jakość (83 % acc, F1 0.81), ale ponad 1 400 s (23 min) trenowania na GPU.

-Umożliwia samodzielne wyciąganie cech, ale kosztem czasu i mocy obliczeniowej.

k-NN na histogramach HSV:

-Średnia dokładność (65 %), bardzo szybki „trening” (pamięta tylko dane).

-Nadaje się jako szybki baseline, ale nie wyłapuje złożonych wzorców kolorystycznych.

Naive Bayes na histogramach HSV:

-Najsłabsze wyniki (<50 % accuracy), pomimo błyskawicznego „trenowania”.

-Założenie niezależności koszyków histogramu okazuje się zbyt uproszczone.