Politechnika Warszawska

Wydział Matematyki i Nauk Informacyjnych

# Augmentacja danych obrazowych dla modeli głębokich

Maria Deka, Julia Girtler, Patrycja Piechocka

Zaawansowane Metody Uczenia Maszynowego

2 czerwca 2025

# Spis treści

# 1. Wstęp

## 1.1. Cel projektu

Celem niniejszego projektu jest zbadanie wpływu wybranych metod augmentacji danych obrazowych na jakość działania modeli głębokiego uczenia. Skupiono się na trzech popularnych podejściach: *Mixup*, *CutMix* oraz *Cutout*. Eksperymenty zostały przeprowadzone na zbiorze danych *CIFAR-10* przy użyciu konwolucyjnej sieci neuronowej.

## 1.2. Augmentacja danych

Augmentacja danych jest jedną z najczęściej stosowanych technik poprawy ogólnej zdolności generalizacji modeli uczenia maszynowego, zwłaszcza w kontekście głębokich sieci neuronowych. Poprzez generowanie sztucznych przykładów na podstawie danych treningowych, możliwe jest ograniczenie przeuczenia i poprawa stabilności modelu bez konieczności zbierania dodatkowych danych.

W przypadku danych obrazowych najczęściej stosuje się augmentacje polegające na transformacjach geometrycznych (np. obrót, przeskalowanie, odbicie lustrzane), zmianach koloru lub metodach mieszania obrazów, takich jak *Mixup* czy *CutMix*.

## 1.3. Problematyka klasyfikacji obrazów

Zadanie klasyfikacji obrazów polega na przypisaniu danej próbce obrazu jednej z predefiniowanych klas. W dobie rosnącej liczby zastosowań wizji komputerowej, takich jak rozpoznawanie obiektów, analiza scen, diagnostyka medyczna czy systemy autonomiczne, klasyfikacja obrazów odgrywa kluczową rolę. Modele głębokiego uczenia, w szczególności konwolucyjne sieci neuronowe (CNN), osiągają obecnie znakomite wyniki w zadaniach klasyfikacji obrazów. Jednak ich skuteczność w dużej mierze zależy od ilości i jakości danych treningowych, co czyni metody augmentacji niezwykle istotnym elementem procesu uczenia.

# 2. Zbiór danych i architektura sieci

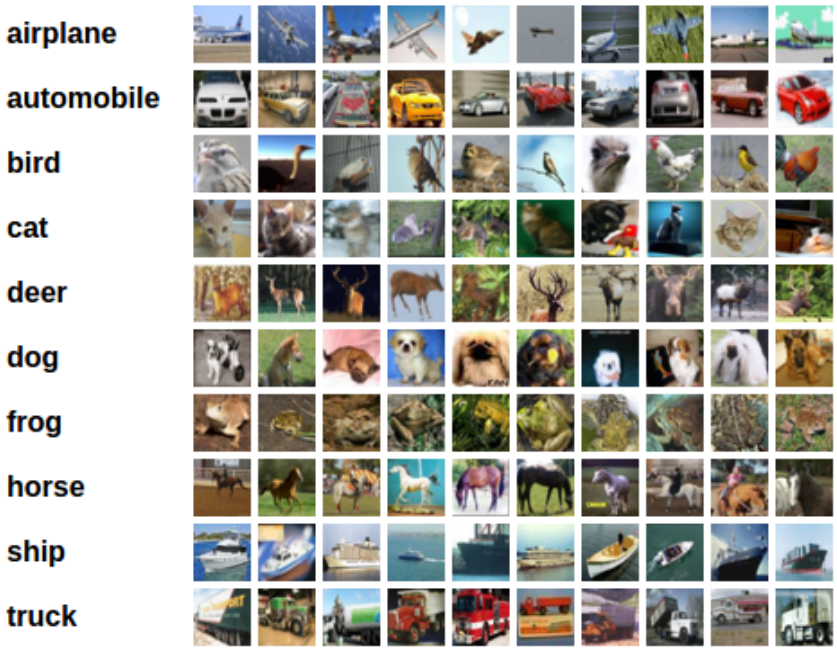
## 2.1. Zbiór danych CIFAR-10

W ramach projektu wykorzystano popularny zbiór danych obrazowych *CIFAR-10*, zawierający łącznie 60 000 kolorowych obrazów w rozdzielczości $32 \times 32$ piksele. Dane są podzielone na 10 klas odpowiadających różnym obiektom, takim jak: samolot, samochód, ptak, kot, jeleń, pies, żaba, koń, statek oraz ciężarówka.

Zbiór danych został rozdzielony na:

- 50 000 obrazów treningowych,
- 10 000 obrazów testowych.

Każda klasa jest reprezentowana równomiernie. Przykładowe obrazy z tego zbioru przedstawiono na rysunku 2.1.




Rysunek 2.1: Przykładowe obrazy ze zbioru CIFAR-10

## 2.2. Architektura sieci neuronowej

Do klasyfikacji obrazów wykorzystano konwolucyjną sieć neuronową (CNN), składającą się z dwóch bloków:

- każdy blok zawiera warstwę konwolucyjną, funkcję aktywacji ReLU oraz warstwę poolingującą,
- po części konwolucyjnej występują dwie w pełni połączone warstwy, z funkcją ReLU i warstwą wyjściową typu softmax.

Sieć przyjmuje obrazy w formacie $32 \times 32$ z 3 kanałami i wyprowadza wektor o 10 elementach odpowiadających klasom. Szczegółowy kod implementacji przedstawiono w rozdziale dotyczącym implementacji.

# 3. Przegląd metod augmentacji danych obrazowych

## 3.1. Klasyczne techniki augmentacji

Klasyczne metody augmentacji polegają na stosowaniu prostych transformacji geometrycznych i kolorystycznych, które nie zmieniają etykiety obrazu. Celem ich stosowania jest zwiększenie różnorodności zbioru treningowego i poprawa zdolności generalizacji modelu. Do najczęściej stosowanych metod należą:

- odbicia lustrzane,
- obrót o losowy kąt,
- przycięcia,
- zmiana jasności lub kontrastu,
- przesunięcia i skalowania.

Techniki te są łatwe do zaimplementowania i skutecznie redukują przeuczenie, jednak mają ograniczoną zdolność do generowania bardziej złożonych wariantów danych.

## 3.2. Augmentacja przez mieszanie obrazów

W ostatnich latach dużą popularność zdobyły metody augmentacji oparte na łączeniu dwóch lub więcej obrazów. Cechą wspólną tych technik jest generowanie nowych przykładów poprzez kombinację pikseli, fragmentów obrazu lub cech semantycznych, a także proporcjonalne mieszanie etykiet. Wśród takich metod wyróżnia się:

### 3.2.1. Mixup

Metoda *Mixup* polega na stworzeniu nowego obrazu $x_{\text{mix}}$ oraz etykiety $y_{\text{mix}}$ jako liniowej kombinacji dwóch przykładów treningowych:

$$x_{\text{mix}} = \lambda x_1 + (1 - \lambda)x_2, \quad y_{\text{mix}} = \lambda y_1 + (1 - \lambda)y_2$$

gdzie $\lambda \in [0, 1]$ to współczynnik losowany z rozkładu beta. Mixup poprawia odporność modelu na zakłócenia i prowadzi do bardziej płynnych granic decyzyjnych.

### 3.2.2. CutMix

W metodzie *CutMix* wycina się fragment z jednego obrazu i wkleja go w losowe miejsce drugiego. Etykieta obrazu jest obliczana proporcjonalnie do powierzchni każdego z fragmentów:

$$x_{\text{mix}} = x_1 \odot M + x_2 \odot (1 - M)$$

gdzie $M$ to maska binarna – czyli obraz tej samej wielkości, w którym piksele mają wartość 1 w miejscu wklejonego fragmentu i 0 poza nim. Dzięki temu część pikseli pochodzi z obrazu $x_1$, a pozostałe z obrazu $x_2$. Etykieta końcowa:

$$y_{\text{mix}} = \lambda y_1 + (1 - \lambda) y_2$$

CutMix łączy zalety Cutout i Mixup, zachowując lokalne cechy obrazu.

### 3.2.3. Cutout

Metoda *Cutout* polega na zamaskowaniu losowego fragmentu obrazu — piksele w wybranym obszarze są zastępowane zerami lub szumem. Nie zmienia się etykieta obrazu, ale zmusza to model do wyciągania informacji z kontekstu i zapobiega przeuczeniu.

## 3.3. Porównanie metod

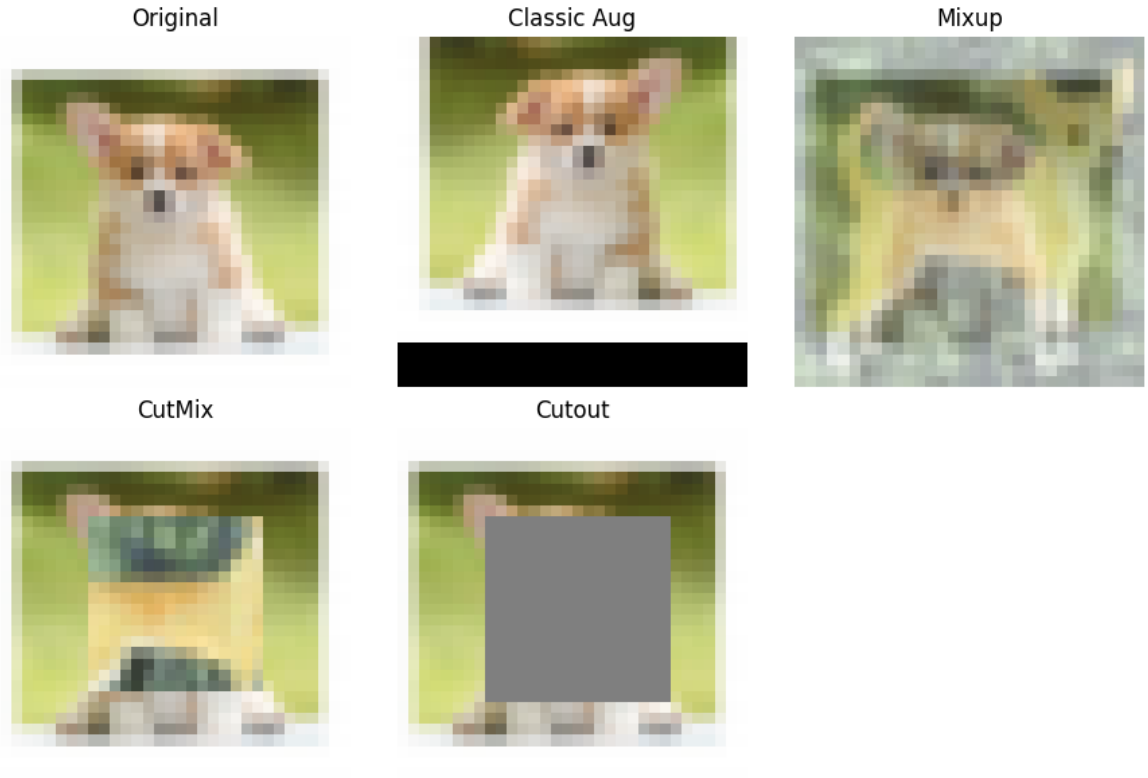
Każda z opisanych metod ma swoje zalety i ograniczenia. W tabeli 3.1 zestawiono ich kluczowe właściwości.

Tabela 3.1: Porównanie metod augmentacji obrazów

| **Metoda** | **Modyfikacja obrazu** | **Zmiana etykiety** | **Zalety** |
|---|---|---|---|
| Mixup | interpolacja pikseli | tak | wygładza granice decyzyjne |
| CutMix | wklejanie fragmentów | tak | lokalność, większy realizm |
| Cutout | maskowanie fragmentów | nie | prostota, zapobiega przeuczeniu |

W kolejnych rozdziałach każda z metod zostanie przetestowana empirycznie na tym samym zbiorze danych i architekturze sieci neuronowej.

Dodatkowo na rysunku 3.1 przedstawiono przykładowe przekształcenia dla obrazu przedstawiającego psa (z poza zbioru CIFAR).




Rysunek 3.1: Przykładowe transformacje obrazu

# 4. Implementacja wybranych metod augmentacji

## 4.1. Środowisko programistyczne

Projekt został zaimplementowany w języku Python z wykorzystaniem biblioteki `PyTorch`. Do obsługi danych obrazowych użyto pakietu `torchvision`, a wykresy generowano przy użyciu biblioteki `matplotlib`.

## 4.2. Architektura sieci

We wszystkich eksperymentach zastosowano jednolitą architekturę konwolucyjnej sieci neuronowej *SimpleCNN*, której opis znajduje się w rozdziale 2.2. Składa się ona z dwóch bloków konwolucyjnych (Conv2D + ReLU + MaxPooling), a następnie dwóch w pełni połączonych warstw zakończonych warstwą softmax.

## 4.3. Model bazowy bez augmentacji

Model bazowy stanowi punkt odniesienia dla oceny skuteczności poszczególnych metod augmentacji. W tym wariancie obrazy były jedynie konwertowane do tensorów oraz normalizowane względem średnich i odchyleń standardowych obliczonych na podstawie całego zbioru treningowego CIFAR-10:

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465),
                         (0.247, 0.243, 0.261))
])
```

Wartości `(0.4914, 0.4822, 0.4465)` oraz `(0.247, 0.243, 0.261)` odpowiadają odpowiednio średnim i odchyleniom standardowym dla kanałów $R$, $G$ i $B$ w zbiorze CIFAR-10 i są powszechnie stosowane w eksperymentach porównawczych w literaturze.

## 4.4. Augmentacja klasyczna

W drugiej wersji modelu zastosowano klasyczne techniki augmentacji danych, powszechnie wykorzystywane w zadaniach klasyfikacji obrazów: losowe przycięcie z marginesem oraz odbicie lustrzane. Transformacja `RandomCrop(32, padding=4)` jest powszechnie stosowana w eksperymentach z CIFAR-10. Obraz zostaje tymczasowo rozszerzony do rozmiaru 40×40 pikseli, a następnie losowo przycięty do oryginalnego rozmiaru 32×32. Pozwala to zwiększyć odporność modelu na przesunięcia oraz poprawiać jego uogólnianie.

```
transform_train = transforms.Compose([
    transforms.RandomCrop(32, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465),
                         (0.247, 0.243, 0.261))
])
```

## 4.5. Augmentacja metodą Mixup

W metodzie *Mixup* dane wejściowe i etykiety są liniowo łączone. W każdej iteracji treningowej losowano pary przykładów oraz współczynnik $\lambda \sim \text{Beta}(\alpha, \alpha)$, a następnie tworzono sztuczne próbki wejściowe i etykiety:

```
mixed_x = lam * x + (1 - lam) * x[index]
y_a, y_b = y, y[index]
loss = lam * criterion(output, y_a) + (1 - lam) * criterion(output, y_b)
```

Parametr $\alpha$ ustawiono na 0.4. Augmentacja miała charakter globalny — wpływała na cały obraz.

## 4.6. Augmentacja metodą CutMix

*CutMix* polega na wklejeniu wycinka jednego obrazu do drugiego. Współczynnik $\lambda$ określa powierzchnię zamienionego fragmentu. Etykieta końcowa jest również mieszana proporcjonalnie:

```
inputs[:, :, bbx1:bbx2, bby1:bby2] =
    inputs[rand_index, :, bbx1:bbx2, bby1:bby2]
lam = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (H * W))
```

Wycinek był losowany dla każdego batcha osobno. Metoda ta zachowuje lokalne cechy semantyczne i jest odporna na przeuczenie.

## 4.7. Augmentacja metodą Cutout

*Cutout* polega na losowym zamaskowaniu fragmentu obrazu poprzez nadpisanie go zerami. Maski miały rozmiar $8 \times 8$ pikseli:

```
y1 = np.clip(y - self.size // 2, 0, h)
        y2 = np.clip(y + self.size // 2, 0, h)
        x1 = np.clip(x - self.size // 2, 0, w)
        x2 = np.clip(x + self.size // 2, 0, w)

        img[:, y1:y2, x1:x2] = 0.0
```

Augmentacja ta nie zmienia etykiet klas i działa lokalnie. Pomaga modelowi uczyć się bardziej ogólnych reprezentacji cech.

## 4.8. Podsumowanie implementacji

Każda metoda została zaimplementowana jako oddzielny plik kodu:

- `model_bazowy.py`
- `model_aug_classic.py`
- `model_mixup.py`
- `model_cutmix.py`
- `model_cutout.py`

Wszystkie modele trenowano na identycznej architekturze i porównano w dalszym rozdziale.

# 5. Eksperymenty i wyniki

## 5.1. Ustawienia treningu

Wszystkie modele trenowano na zbiorze CIFAR-10 z wykorzystaniem tej samej architektury sieci konwolucyjnej opisanej w Rozdziale 2.2. Parametry treningu były stałe dla każdego eksperymentu:

- Liczba epok: 20
- Batch size: 128
- Optymalizator: Adam
- Współczynnik uczenia: 0.001
- Funkcja straty: `CrossEntropyLoss`

## 5.2. Porównanie wyników

Poniżej przedstawiono porównanie skuteczności pięciu wersji modelu: bez augmentacji, z klasyczną augmentacją, Mixup, CutMix i Cutout. Podano najlepszą uzyskaną dokładność testową, odpowiadający jej test loss oraz numer epoki, w której została osiągnięta.

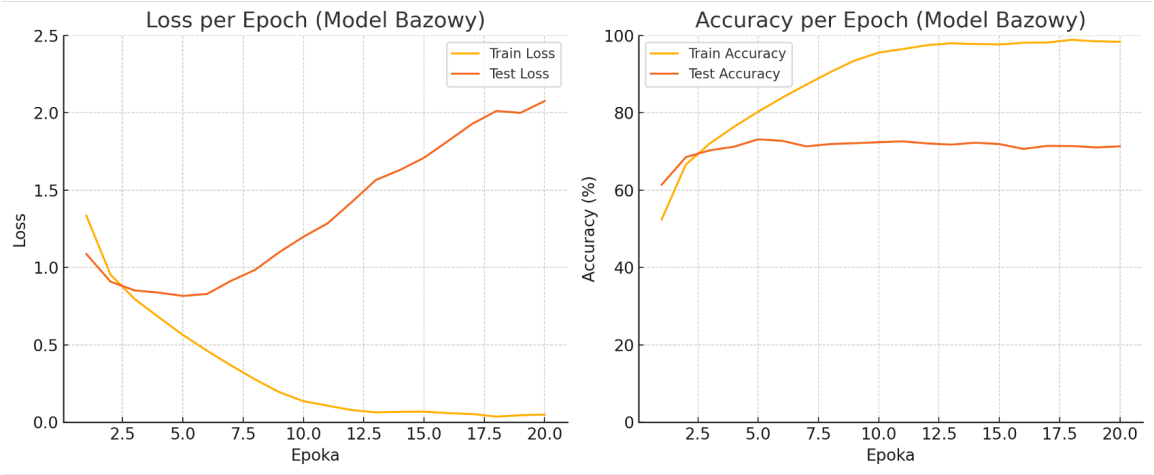
Tabela 5.1: Porównanie skuteczności modeli

| **Model** | **Test Accuracy [%]** | **Test Loss** | **Epoka** |
|---|---|---|---|
| Model bazowy | 73.13 | 0.8162 | 5 |
| Klasyczna augmentacja | **78.53** | **0.6355** | **20** |
| Mixup | 74.22 | 0.7805 | 8 |
| CutMix | 76.04 | 0.7488 | 20 |
| Cutout | 73.26 | 0.8951 | 10 |

Najlepszy wynik osiągnięto w modelu z augumentacją Cutout, który poprawił skuteczność o około 8 punktów procentowych względem modelu bazowego. Mixup i CutMix oraz klasyczna augumentacja również przyczyniły się do wzrostu skuteczności.

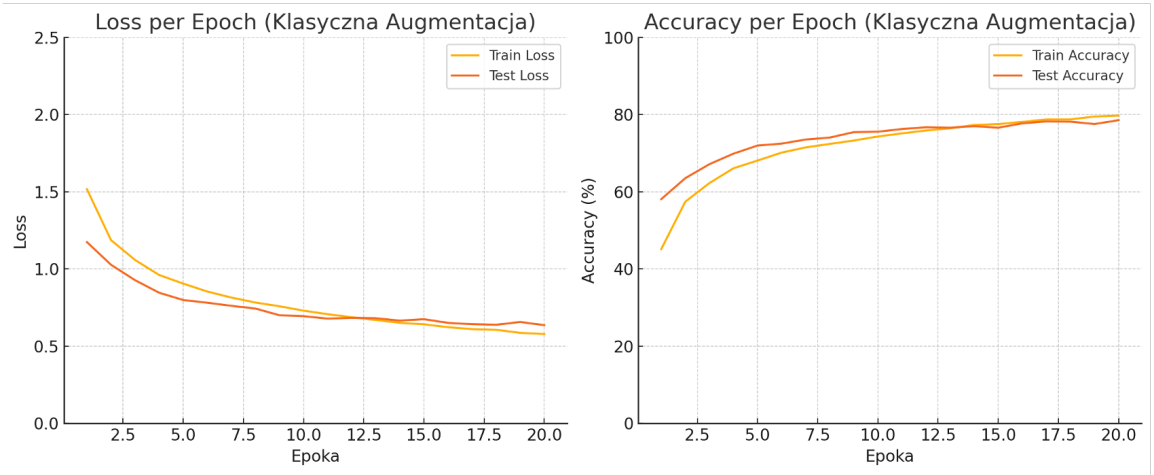
## 5.3. Wizualizacja przebiegów

W celu lepszej interpretacji dynamiki uczenia się poszczególnych modeli przygotowano wykresy zmian funkcji straty i dokładności w trakcie kolejnych epok treningu.



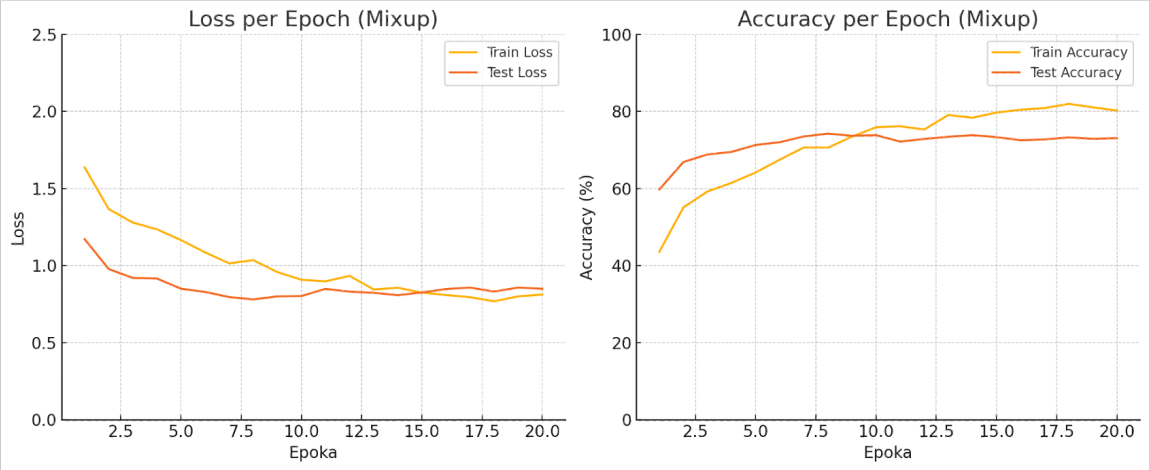

Rysunek 5.1: Model bazowy — przebieg strat i dokładności

Wykres strat i dokładności dla modelu bez augmentacji. Po 5. epoce obserwujemy wzrost test loss przy rosnącym train accuracy — objaw przeuczenia.



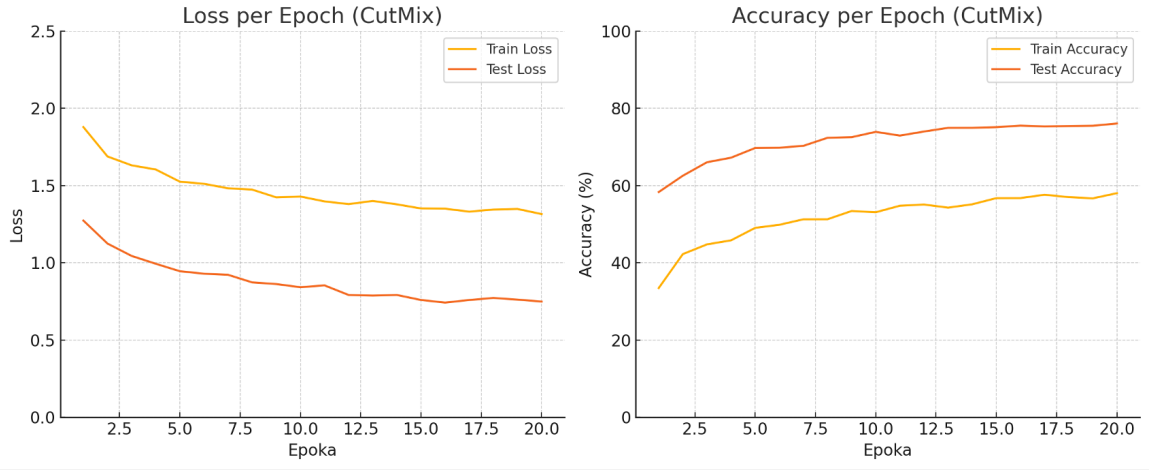

Rysunek 5.2: Klasyczna augmentacja — przebieg strat i dokładności

Klasyczna augmentacja (losowe przycięcie i odbicie). Model osiąga najwyższą skuteczność i zachowuje stabilność strat testowych.



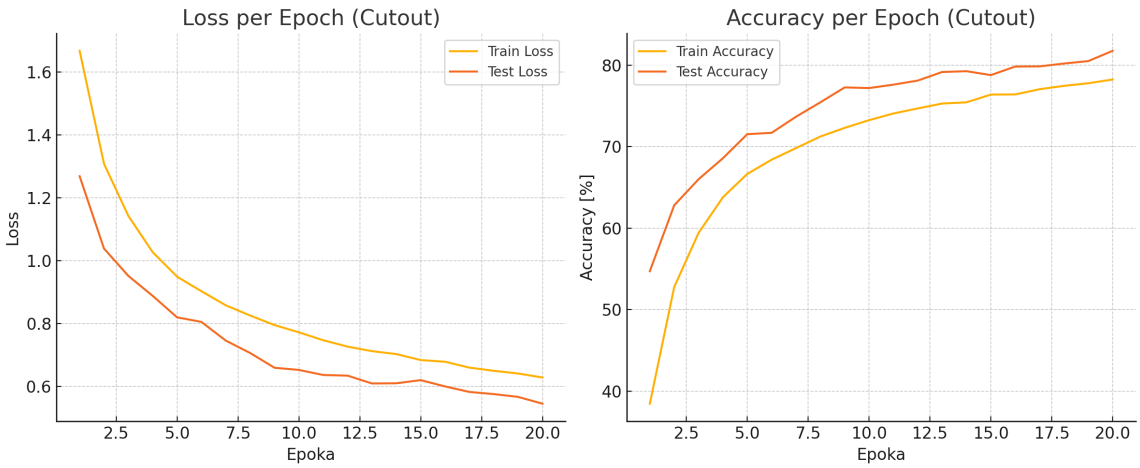

Rysunek 5.3: Mixup — przebieg strat i dokładności

Augmentacja Mixup: liniowe łączenie obrazów i etykiet. Test accuracy osiąga stabilne 74% do końca treningu.




Rysunek 5.4: CutMix — przebieg strat i dokładności

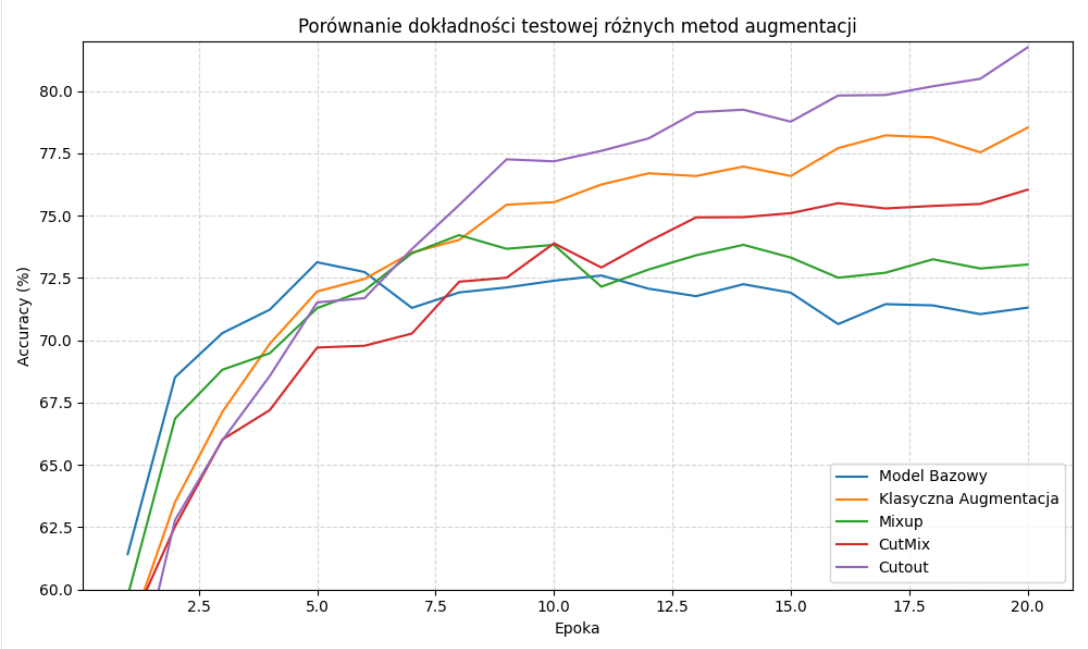
Augmentacja CutMix: fragment jednego obrazu wklejany w drugi. Skuteczność testowa stopniowo rośnie aż do 76%.




Rysunek 5.5: Cutout — przebieg strat i dokładności

Augmentacja Cutout: maskowanie fragmentu obrazu. Metoda daje bardzo dobre wyniki, dokładność testowa stopniowo rośnie, aż do poziomu 81,75%.

## 5.4. Wnioski




Rysunek 5.6: Porównanie test accuracy dla wszystkich metod

Przeprowadzona analiza jednoznacznie pokazuje, że zastosowanie metod augmentacji znacząco poprawia jakość uogólniania modeli uczących się na zbiorze CIFAR-10. Spośród testowanych podejść najlepsze rezultaty osiągnęła metoda Cutout, uzyskując końcową dokładność na

poziomie 81,75%, co stanowi wyraźny wzrost względem modelu bazowego (ok. 71,6%). Model bez augmentacji wykazuje silną tendencję do przeuczenia — mimo że początkowo osiąga wysokie wartości dokładności (nawet powyżej wszystkich metod w epokach 2–4), to jego skuteczność testowa szybko stabilizuje się, a nawet nieznacznie spada po 10. epoce. W przeciwieństwie do tego, wszystkie modele z augmentacją zachowują stabilność i stopniowo zwiększają dokładność wraz z liczbą epok. Metody Mixup i CutMix również poprawiają wyniki względem bazowego modelu, osiągając kolejno ok. 73,4% i 75,9% testowej dokładności. Mimo że początkowo uczą się nieco wolniej (zwłaszcza Cutout), to ostatecznie przewyższają model bazowy o kilka punktów procentowych. Klasyczna augmentacja (losowe kadrowanie, obrót poziomy) przynosi stały przyrost, kończąc na 78,8%. Najważniejszy wniosek: augmentacja nie tylko zwiększa dokładność, ale czyni model bardziej odpornym na przeuczenie. Szczególnie metoda Cutout, która przez ukrywanie fragmentów obrazu, zmusza model do uczenia się bardziej ogólnych i rozproszonych reprezentacji cech.

Pomimo że metody Mixup i CutMix są często skuteczniejsze na dużych i złożonych zbiorach danych, w naszym przypadku (CIFAR-10) nie osiągnęły najwyższych wyników. Jest to najprawdopodobniej spowodowane małą rozdzielczością obrazów (32×32 piksele), gdzie zarówno mieszanie pikseli (Mixup), jak i wklejanie prostokątów (CutMix) prowadzi do utraty istotnych informacji wizualnych. W takich przypadkach prostsze podejścia, takie jak Cutout, które częściowo maskują dane bez zmiany etykiety, okazują się bardziej skuteczne. Dodatkowo, względnie mała pojemność modelu (SimpleCNN) mogła nie pozwolić na pełne wykorzystanie potencjału zaawansowanych technik mieszania danych.

## Bibliografia

[1] WolframAlphaV1.0, *Easy Way to Improve Image Classifier Performance: Preprocessing Techniques*, Medium, 15 lipca 2020. https://medium.com/@wolframalphav1.0/easy-way-to-improve-image-classifier-performance-p

[2] Politechnika Warszawska, *Analiza wydajności i optymalizacji systemów ML*, Repozytorium Politechniki Warszawskiej, 2019.https://repo.pw.edu.pl/info/article/WUTb3002690e731469db98e22d006f73e35/