Politechnika Wrocławska

Podstawy Sieci Neuronowych

Projekt 2

Sieci głębokie do nauki rozbudowanych problemów

Rozpoznawanie obrazów

13 stycznia 2025r.

Autorzy:

Kateryna Fedoruk 272609

Aleksandra Wencel 272557

Grupa 7, Poniedziałki 9:15

Kierunek: ISA

Prowadzący: Mgr inż. Michał Zmonarski

Wariant na >4.0

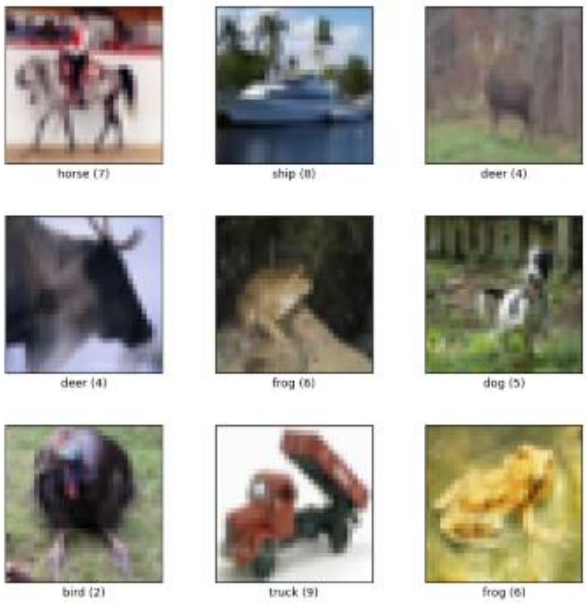
1. Wprowadzenie

W projekcie zajmujemy się zadaniem klasyfikacji obrazów pochodzących z popularnego zestawu danych CIFAR-10. W celu realizacji tego zadania skonstruowano i przetestowano konwolucyjną sieć neuronową (CNN), której podstawowym celem jest poprawne przypisanie każdego obrazu do jednej z dziesięciu możliwych klas (na przykład: samolot, samochód, ptak, kot, pies czy żaba). Projekt obejmuje nie tylko trening i ewaluację sieci, ale również głębszą analizę procesu uczenia poprzez monitorowanie różnych miar błędu (zarówno na zbiorze treningowym, jak i testowym), wgląd w rozkłady wag oraz zastosowanie różnych strategii optymalizacji, w tym adaptacyjnych zmian współczynnika uczenia czy użycie momentum.

2. Opis problemu

Problem, z którym mierzy się projekt, polega na automatycznym przypisywaniu obrazów do właściwej klasy zdefiniowanej w zbiorze danych CIFAR-10. Każdy obraz w tym zbiorze ma rozmiar 32×32 piksele i posiada trzy kanały kolorystyczne (RGB), co w efekcie daje niewielkie, ale wciąż zróżnicowane reprezentacje wizualne. Zbiór CIFAR-10 zawiera łącznie 10 różnych klas przedmiotów i obiektów, które w znacznym stopniu różnią się między sobą wyglądem (np. kot kontra samolot czy żaba kontra samochód).

Ze względu na niewielką rozdzielczość obrazów i stosunkowo mały rozmiar każdego przykładu, CIFAR-10 stanowi dobry punkt wyjścia do testowania różnorodnych architektur sieci neuronowych, w szczególności sieci konwolucyjnych. Sieci konwolucyjne są wiodącym podejściem w dziedzinie rozpoznawania obrazów dzięki zdolności do automatycznego wyodrębniania coraz bardziej złożonych cech w kolejnych warstwach. Mimo że CIFAR-10 jest relatywnie małym zestawem, nadal wymaga skutecznej strategii uczenia i odpowiedniego doboru hiperparametrów.

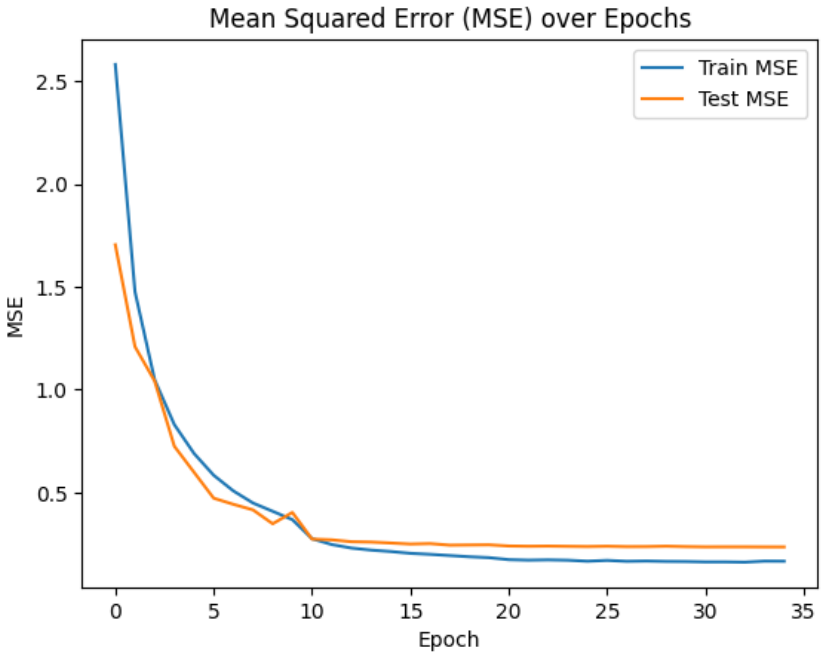


3. Wyniki i analiza

3.1. MSE

Na poniższym wykresie przedstawiono zmianę błędu średniokwadratowego (MSE) w trakcie treningu sieci neuronowej. Pokazano wyniki dla zbioru treningowego, umożliwiając ocenę procesu uczenia.

Wykres pokazuje, że MSE

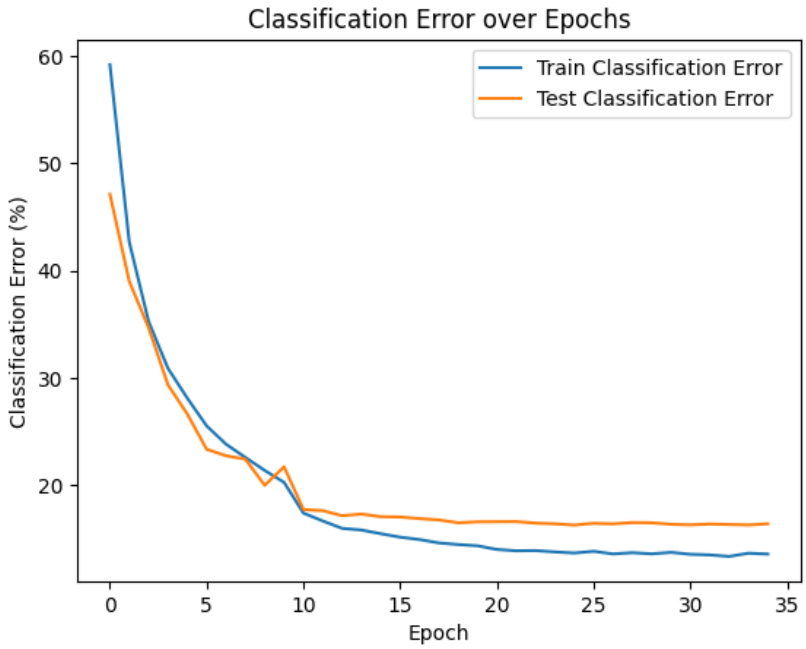


początkowo jest wysokie, ale gwałtownie spada w ciągu pierwszych 10 epok, co oznacza, że model skutecznie uczy się rozpoznawać kluczowe wzorce danych. Stabilizacja obu krzywych po ~10 epokach oraz ich bliskie wartości świadczą o dobrym dopasowaniu modelu i braku oznak przeuczenia.

3.2. Błąd klasyfikacji

Na poniższym wykresie przedstawiono zmianę błędu klasyfikacji (wyrażonego w procentach) w trakcie treningu modelu. Wykresy prezentują błąd dla zbioru treningowego (niebieska linia) oraz testowego (pomarańczowa linia).

Przyjęty próg 0.5 dla klasyfikacji 0-1 oznacza, że model przypisuje obraz do określonej klasy, jeśli przewidywane prawdopodobieństwo dla tej klasy jest równe lub większe niż 50%. W ten sposób błąd klasyfikacji przedstawiony na wykresie pokazuje procent przypadków, w których przewidywania modelu były niezgodne z rzeczywistymi etykietami.



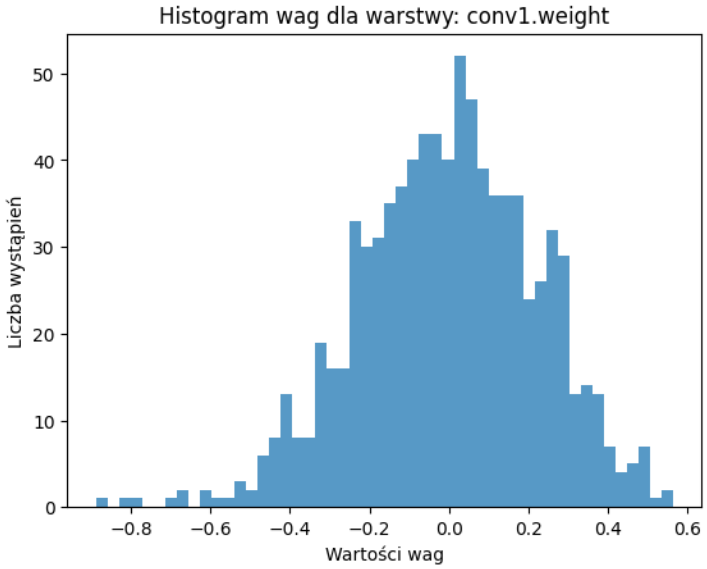
Na początku treningu błąd klasyfikacji jest bardzo wysoki – około 60% na zbiorze treningowym i testowym. Wskazuje to, że model losowo przypisuje klasy do obrazów, ponieważ jego wagi są zainicjalizowane w sposób losowy i nie rozpoznaje jeszcze żadnych wzorców.

Już przy 10 epoce następuje stabilizacja. Obie krzywe są do siebie bardzo zbliżone, co świadczy o dobrej zdolności modelu do generalizacji. Nie widać znacznej różnicy między błędem na zbiorze treningowym a testowym, co wskazuje na brak przeuczenia.

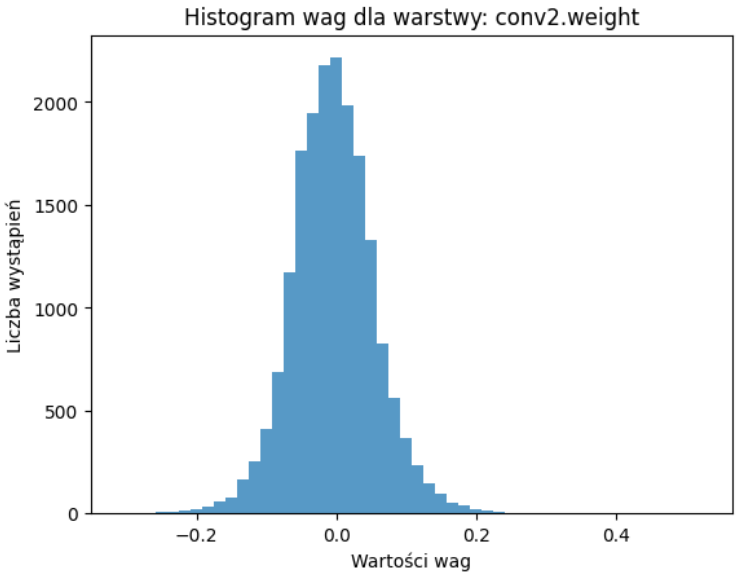
3.3. Wykresy wag

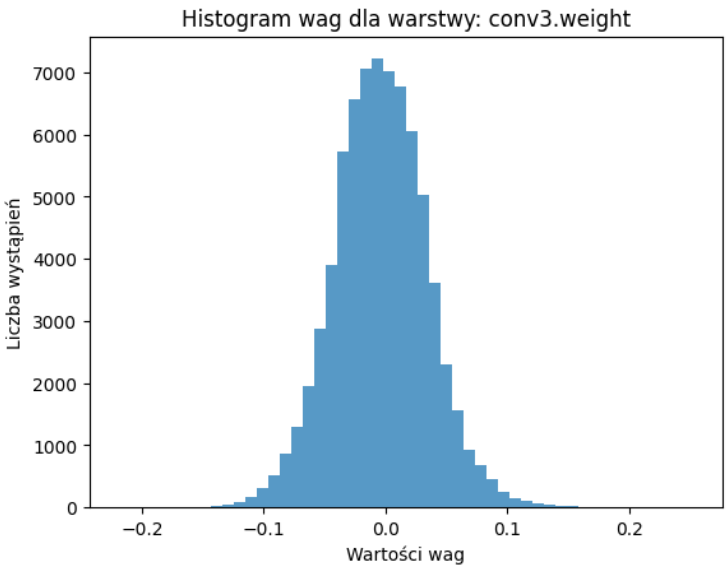
3.3.1. Warstwy konwolucyjne

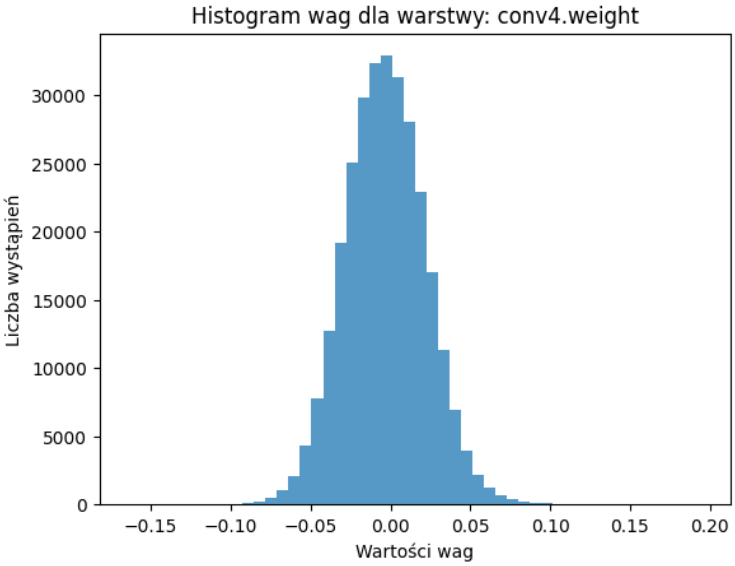
Poniżej przedstawiono histogramy wag dla pięciu warstw konwolucyjnych modelu. Histogramy pokazują rozkład wartości wag, co pozwala na ocenę ich stabilności oraz potencjalnych problemów, takich jak eksplodujące czy zanikające wagi.

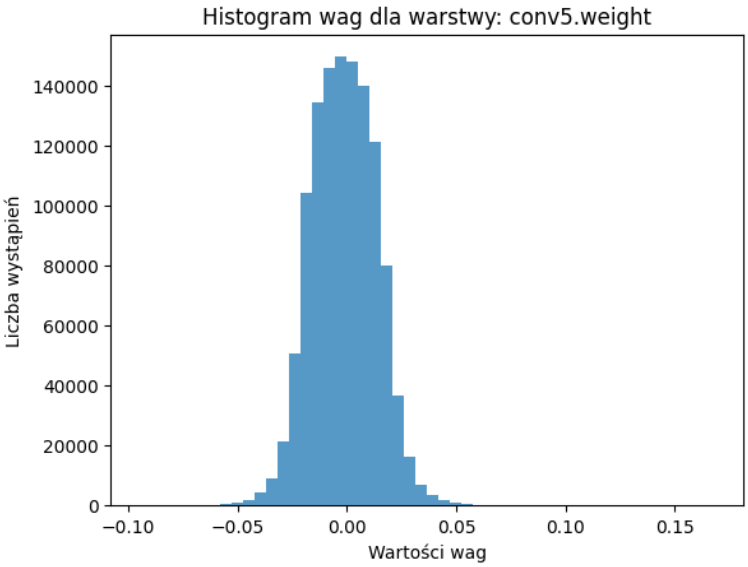


Histogram przedstawia rozkład wartości wag w pierwszej warstwie konwolucyjnej. Widać, że wagi są skoncentrowane wokół zera, a ich wartości mieszczą się w przedziale od około -0.8 do 0.6. Rozkład ten przypomina normalny (gaussowski), co jest pożądane, ponieważ inicjalizacja wag zazwyczaj opiera się na losowym doborze z rozkładu normalnego lub jednostajnego



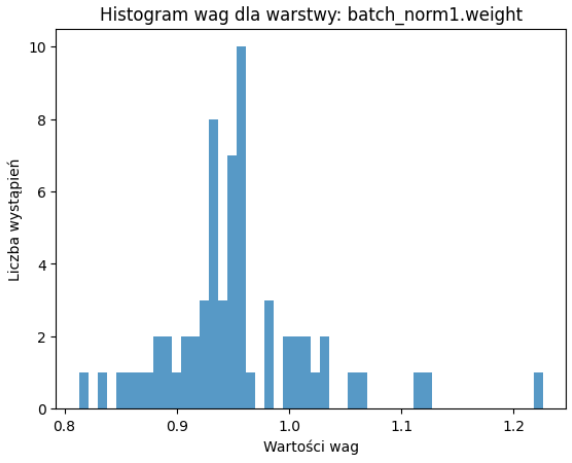


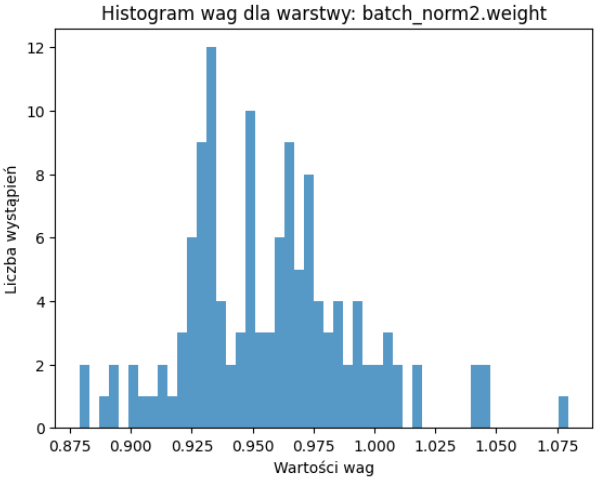


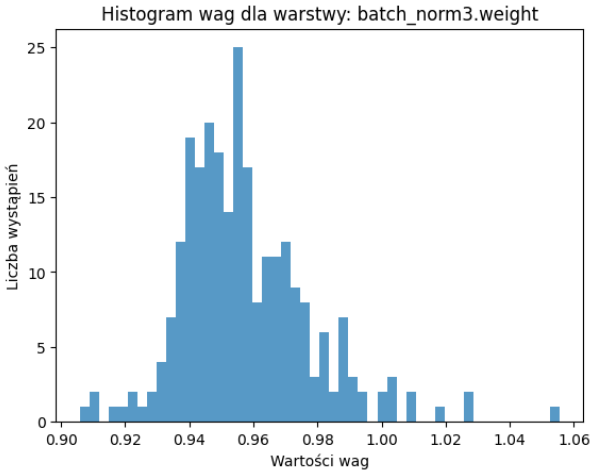


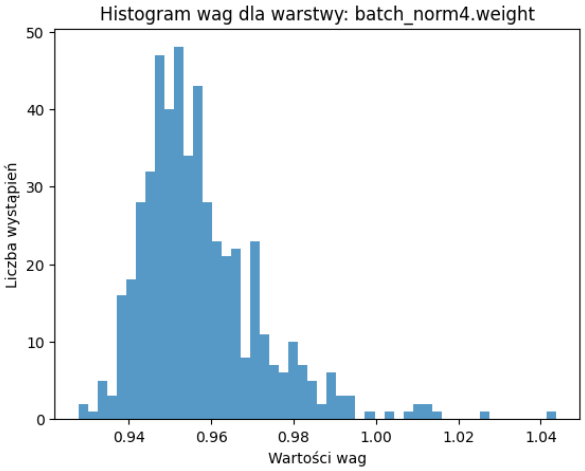
Rozkłady są bardziej „zbite” niż w przypadku pierwszej warstwy, co może wynikać z procesu uczenia się. Wartości wag stają się coraz mniejsze w miarę przechodzenia do kolejnych warstw konwolucyjnych.

3.3.2. Warstwy Batch Norm



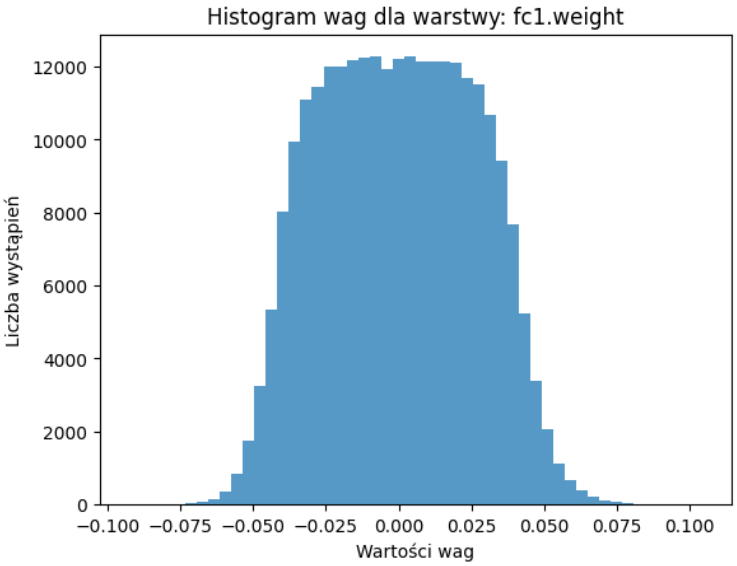


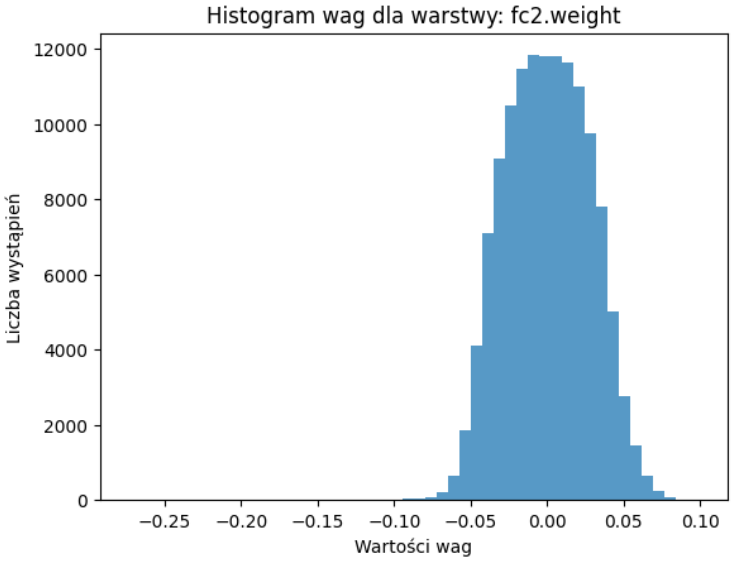


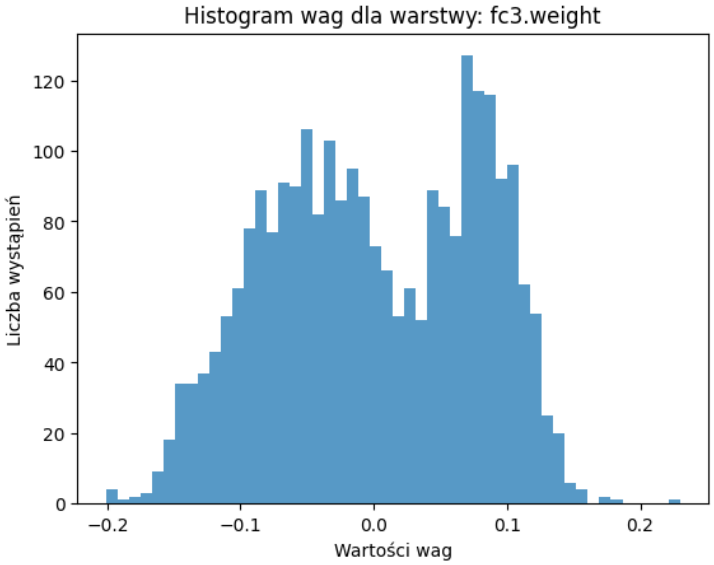


3.3.3. Warstwy w pełni połączone (fully connected)

Warstwy w pełni połączone (FC) przekształcają cechy wyodrębnione z warstwy konwolucyjnej w końcową decyzję klasyfikacyjną. W przypadku CIFAR-10 ostatnia warstwa (fc3) przewiduje prawdopodobieństwa przynależności obrazu do jednej z 10 klas.

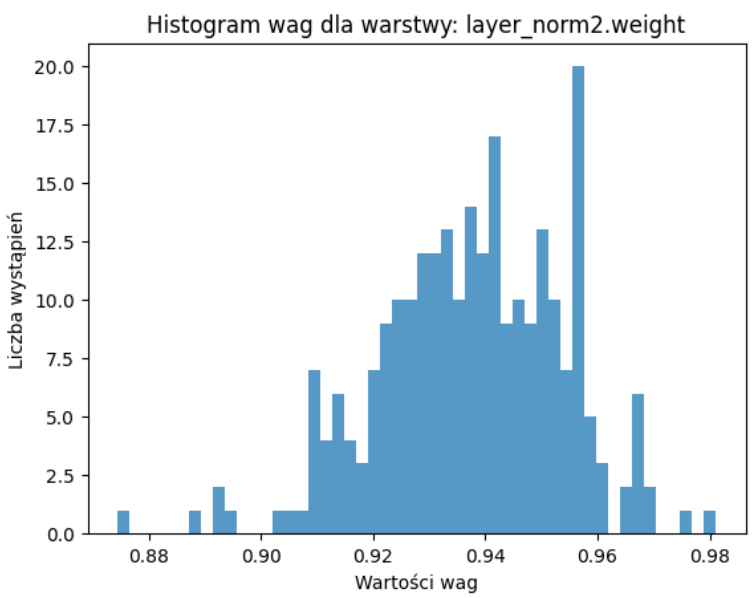


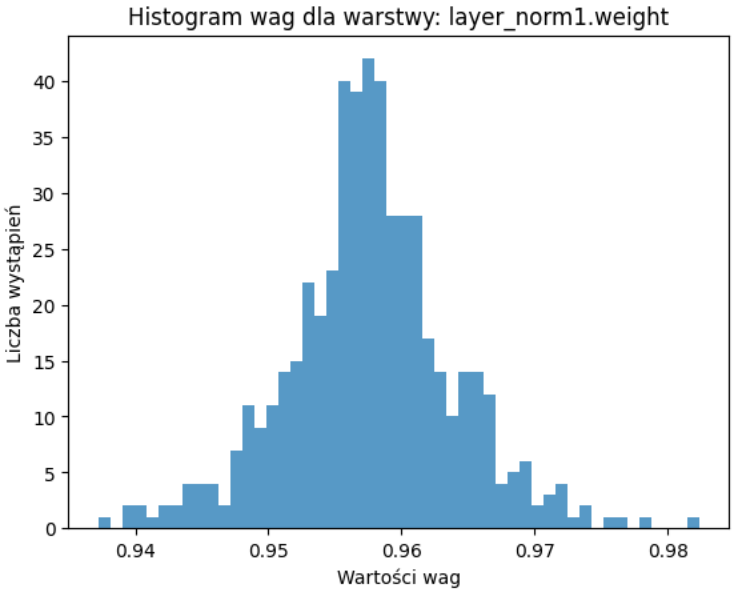




3.3.4. Warstwy normalizujące

Odgrywają kluczową rolę w stabilizowaniu i normalizacji danych w pełni połączonych części modelu.





4. Opcja szybszego kończenia uczenia

Na początku w kodzie zastosowano opcję wcześniejszego zakończenia uczenia się, gdy błąd na zbiorze testowym osiągnął ustalony próg. Jednak w trakcie eksperymentów zauważono, że sieć neuronowa maksymalnie osiąga dokładność na poziomie około 80–90%. W związku z tym wcześniejsze zakończenie treningu mogło prowadzić do pominięcia dalszych optymalizacji w końcowych epokach, które pozwalały na pełne wykorzystanie potencjału modelu.

Ostatecznie ta funkcjonalność została uznana za zbędną i usunięta z kodu, aby umożliwić pełne przeprowadzenie treningu przez wszystkie zaplanowane epoki.

```
Epoch: 60/60
Train Loss: 0.3906, Train Accuracy: 87.11%
Test Loss: 0.4822, Test Accuracy: 84.33%
```

5. Momentum

W kodzie momentum wynosi 0.9. Przyspiesza zbieżność modelu, uwzględniając 90% wpływu gradientów z poprzednich iteracji. Pomaga to eliminować szumy i stabilizować proces uczenia. Dzięki temu model szybciej osiąga minimum funkcji celu i skuteczniej pokonuje płaskie obszary oraz lokalne minima.

6. Adaptacyjny współczynnik uczenia

W kodzie mechanizm adaptacyjnego współczynnika uczenia został zaimplementowany przy użyciu schedulera StepLR, który zmniejsza współczynnik uczenia co 10 epok, dziesięciokrotnie redukując jego wartość. Pozwala to na szybkie uczenie w początkowej fazie i bardziej precyzyjne dostrajanie wag w późniejszych etapach.

7. Zmienny rozmiar kroku uczenia (mini-batch)

W projekcie początkowo stosowano dynamiczny rozmiar mini-batch, który zwiększał się lub zmniejszał w zależności od wyników modelu:

- Rozmiar batcha zwiększano (np. z 128 do 256), aby przyspieszyć trening, gdy wyniki były stabilne.
- Rozmiar batcha zmniejszano (np. do 64 lub 32), gdy wyniki zaczynały się pogarszać.

Jednak ta strategia prowadziła do niestabilności modelu – Test Loss często się pogarszał, a Test Accuracy nie poprawiała się znacząco. Przykładowo, po zmniejszeniu batcha do 64 (Epoch 32), wyniki były gorsze niż przy ustalonym batch size.

Finalnie ustalono stały rozmiar batcha na **128**, co zapewniło lepszą stabilność gradientów i bardziej przewidywalne wyniki treningu oraz testów.

Poniżej znajduje się screen przedstawiający pogorszenie wyników po dynamicznej zmianie rozmiaru batcha.

```
Epoch: 38/60
Train Loss: 0.4215, Train Accuracy: 85.95%
Test Loss: 0.4803, Test Accuracy: 84.25%
Model zapisano w model_epoch_38.pth
Epoch: 39/60
Train Loss: 0.4259, Train Accuracy: 85.78%
Test Loss: 0.4824, Test Accuracy: 84.15%
Model zapisano w model_epoch_39.pth
Epoch: 40/60
Train Loss: 0.4252, Train Accuracy: 85.95%
Test Loss: 0.4831, Test Accuracy: 84.25%
Files already downloaded and verified
Files already downloaded and verified
Stable worsening. Decreasing batch size to 32
Model zapisano w model_epoch_40.pth
Epoch: 41/60
Train Loss: 0.4638, Train Accuracy: 84.59%
Test Loss: 0.4812, Test Accuracy: 84.22%
Model zapisano w model_epoch_41.pth
Epoch: 42/60
Train Loss: 0.4683, Train Accuracy: 84.49%
Test Loss: 0.4824, Test Accuracy: 84.15%
Model zapisano w model_epoch_42.pth
Epoch: 43/60
Train Loss: 0.4587, Train Accuracy: 84.64%
Test Loss: 0.4808, Test Accuracy: 84.12%
Model zapisano w model_epoch_43.pth
Epoch: 44/60
Train Loss: 0.4643, Train Accuracy: 84.56%
Test Loss: 0.4843, Test Accuracy: 84.19%
Model zapisano w model_epoch_44.pth
```

8. Wnioski

- **Przeuczanie i generalizacja:** Początkowy model (dwie warstwy konwolucyjne, dwie w pełni połączone) osiągnął dokładność **96% na treningowym** i **73% na testowym**, co wskazuje na przeuczanie. Wprowadzenie regularyzacji L2 i augmentacji danych (losowe odbicia, przycięcia) poprawiło generalizację, podnosząc dokładność testową do **76.51%** i stabilizując stratę testową.

- **Wpływ Dropout i BatchNorm:** Dodanie Dropout w warstwach w pełni połączonych oraz BatchNorm w warstwach konwolucyjnych poprawiło stabilność i zdolność modelu do generalizacji. Model z 5 warstwami konwolucyjnymi i normalizacją batchów osiągnął stabilne wyniki: **Train Accuracy ~84.6%** i **Test Accuracy ~83.1%**. Wyniki testowe przestały poprawiać się po około 30 epokach, co wskazuje na osiągnięcie plateaus.

- **Liczba warstw konwolucyjnych:** Testowano modele z liczbą warstw konwolucyjnych w przedziale od 2 do 7. Najlepsze wyniki osiągnięto dla **5 warstw konwolucyjnych**, gdzie

udało się znaleźć balans między ilością informacji a możliwością generalizacji. **Wniosek:** Większa liczba warstw nie zawsze oznacza lepsze wyniki – nadmiar warstw może prowadzić do przeuczania lub problemów z uczeniem.

- **Zmienne mini-batche:** Początkowo eksperymentowałyśmy ze zmiennym rozmiarem mini-batch (zwiększając go w przypadku stabilnych wyników i zmniejszając przy pogorszeniu). Okazało się, że dynamiczna zmienność wprowadzała niestabilność gradientów, co skutkowało gorszymi wynikami. Stały rozmiar batcha (128) zapewnił lepszą stabilność i przewidywalność. **Wniosek:** Zmienność mini-batch nie zawsze jest korzystna i wymaga dokładnej analizy w każdym przypadku.

- **Szybsze uczenie:** Wprowadzono techniki przyspieszające trening (np. zmienny learning rate, momentum, i wykorzystanie GPU). Przyspieszyły one trening, ale przy porównywalnych wynikach osiąganych w podobnej liczbie epok i wynikach na poziomie ~84%, pełne wykorzystanie wszystkich technik nie było konieczne. **Wniosek:** Szybsze uczenie jest przydatne w dużych projektach, ale w średniej skali może nie przynosić kluczowych korzyści.

- Model konwolucyjnej sieci neuronowej zdefiniowany w projekcie osiągnął dokładność na poziomie **87%** na zbiorze treningowym i około **84%** na zbiorze testowym. Wyniki te są zgodne z typowymi osiągami dla tego rodzaju architektury przy pracy na zbiorze CIFAR-10.