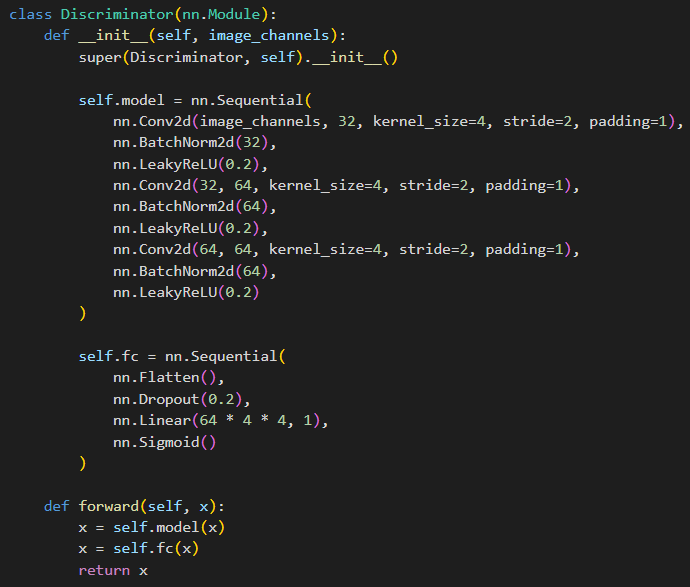
**Czy sieci neuronowe śnią o ciastach marchewkowych?**

**Bartosz Dudek**

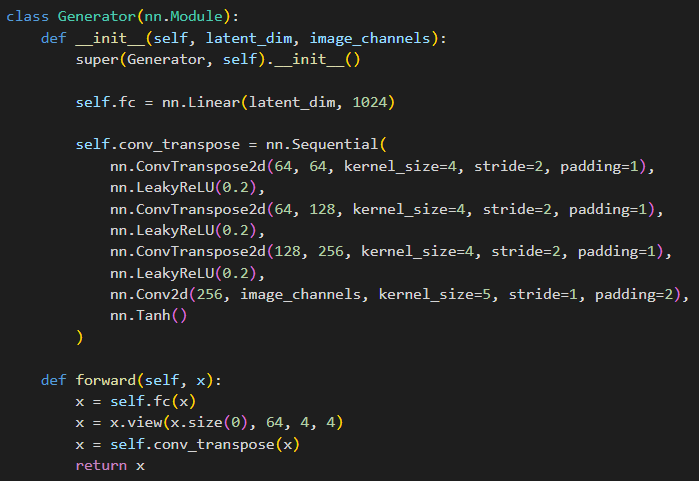
Do obliczeń wykorzystałem PyTorch’a.

1. Przygotowanie modelu dyskryminatora

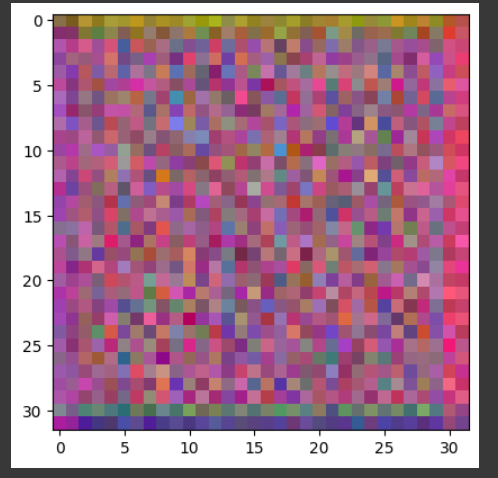


Ze względu na brak padding “same” w PyTorchu został dobrany ręcznie.

1. Przygotowanie modelu generatora

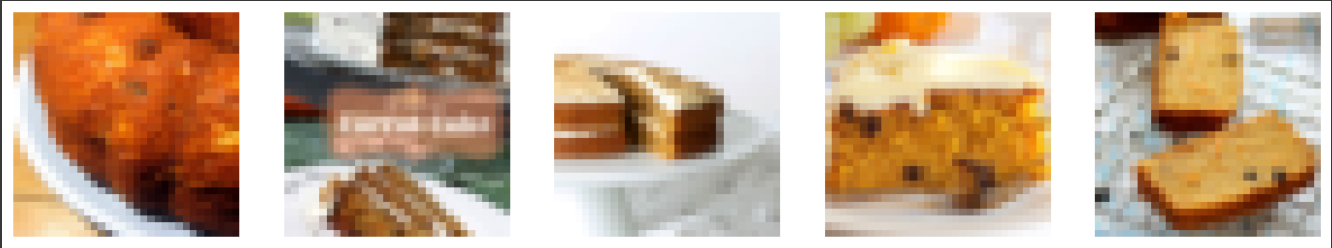


1. Przykładowe zdjęcie wygenerowanie przez generator

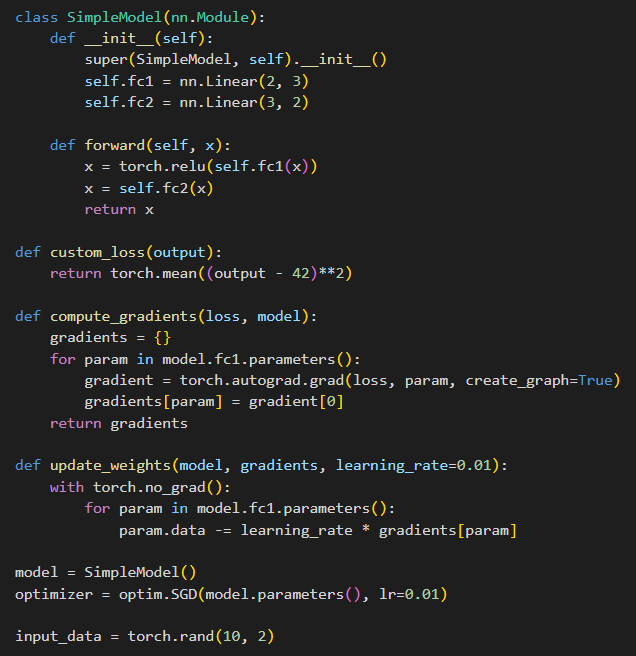


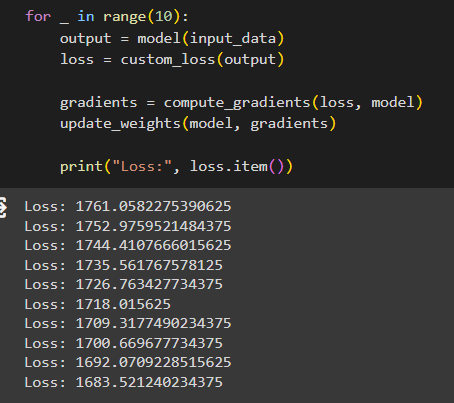
Wartości pikseli przeskalowane do przedziału [-1, 1]

1. Przygotowanie zbioru startowego - kilka wyplotowanych zdjęć



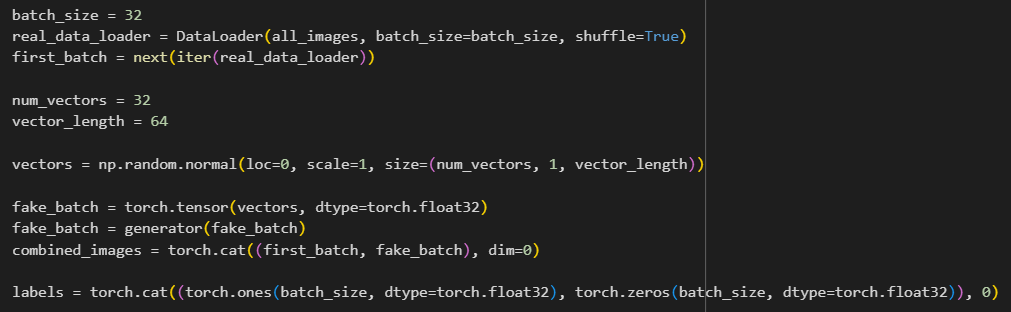
1. Przygotowanie prostego, costomowego modelu



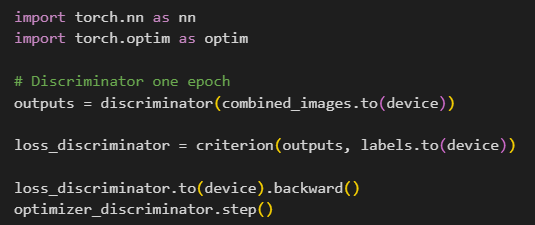
Pętla ucząca i spadający loss

1. Iteracja uczenia dyskryminatora

* przygotowanie danych - wektor prawdziwych danych połączony z wektorem wygenerowanych (prawdziwe mają etykiety 1, sztuczne 0)

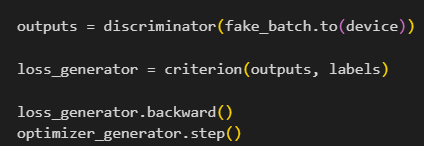


* trening

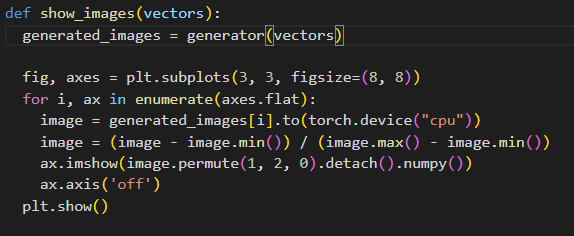


1. Iteracja nauki generatora

* dane możemy wygenerować nowe lub skorzystać z tych podczas treningu dyskryminatora. Tym razem wyniki dyskryminatora uważamy za prawdziwe.



1. Funkcja do generacji obrazów w danym momencie treningu



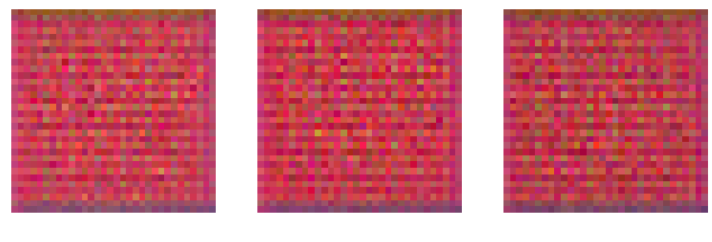
* funkcja do zapisywania danych o modelach w danym momencie treningu

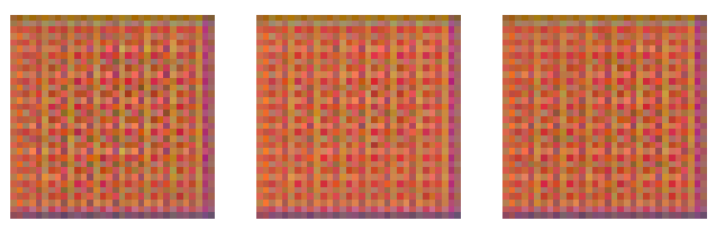


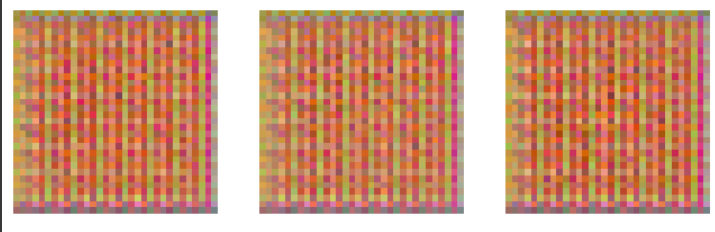
Zapisuje numer epoki, dane modelu, dane optymalizatora, wartość straty oraz czas.

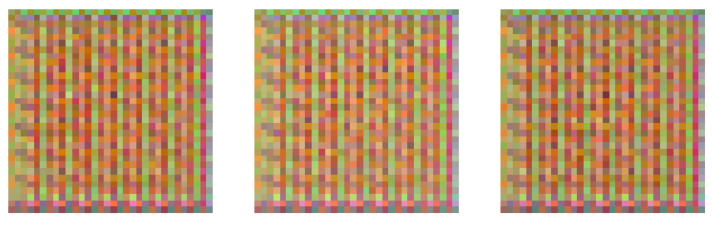
Osobno dyskryminator i generator.

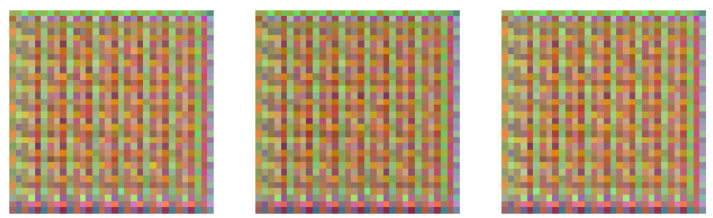
1. Rezultaty na przestrzeni czasu











**Podsumowanie**

Pomimo tego że widoczna była zmiana wyglądu obrazów i zmniejszający się los na przestrzeni epok nie mogłem znaleźć przyczyny niepoprawnych wyników.

Największym problemem w laboratorium jest zdecydowanie brak świadomości czy otrzymany wynik jest mniej więcej ok i czy idzie w dobrym kierunku.

Najwięcej czasu zajmuje znalezienie drobnych błędów w strukturze czy to modeli czy też w zbiorach danych.