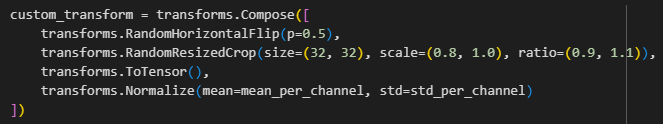
**Visual Transformer**

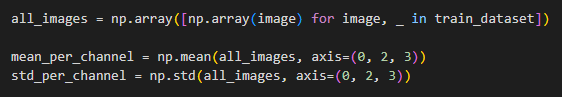
**Bartosz Dudek**

Do obliczeń wykorzystałem PyTorch’a.

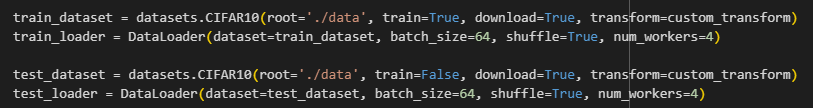
1. Tworzenie sekwencji transformacji



Średnie i odchylenia standardowe policzone w poniższy sposób:

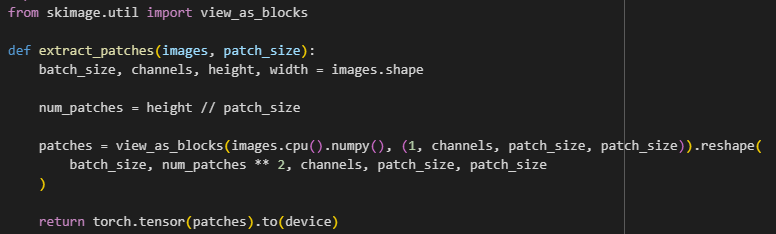


1. Zastosowanie sekwencji transformacji



W Pytorch’u sekwencja jest stosowana przy pobieraniu datasetu, następnie dla każdego zbioru - testowy i treningowy tworzę data loader.

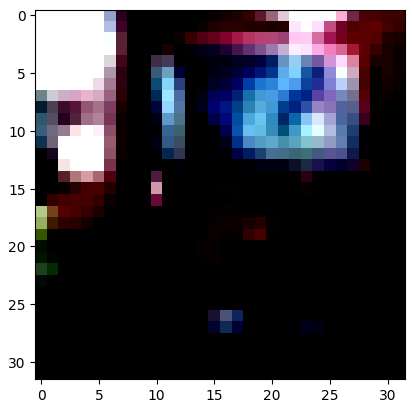
1. Tworzenie batch sekwencji patch’y



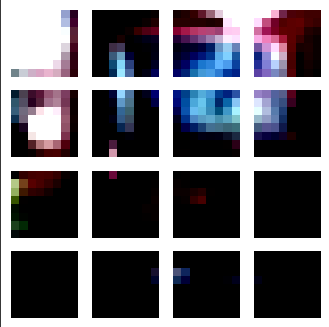
Wykorzystałem metodę view\_as\_block z biblioteki skimage.util.

1. Sprawdzenie podziały na pat patch’e

Obraz oryginalny

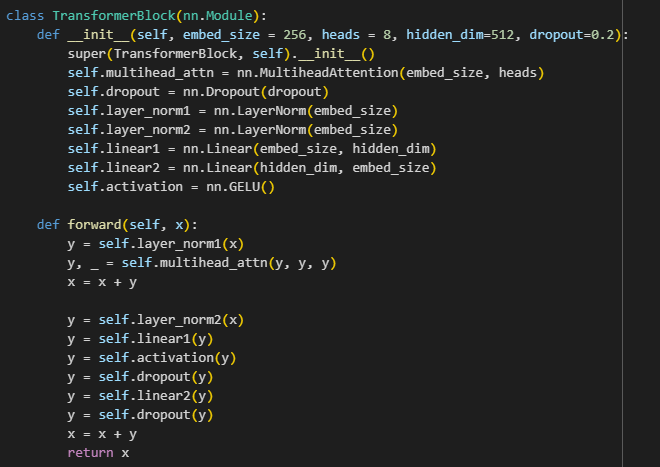


Obraz podzielony na patch’e

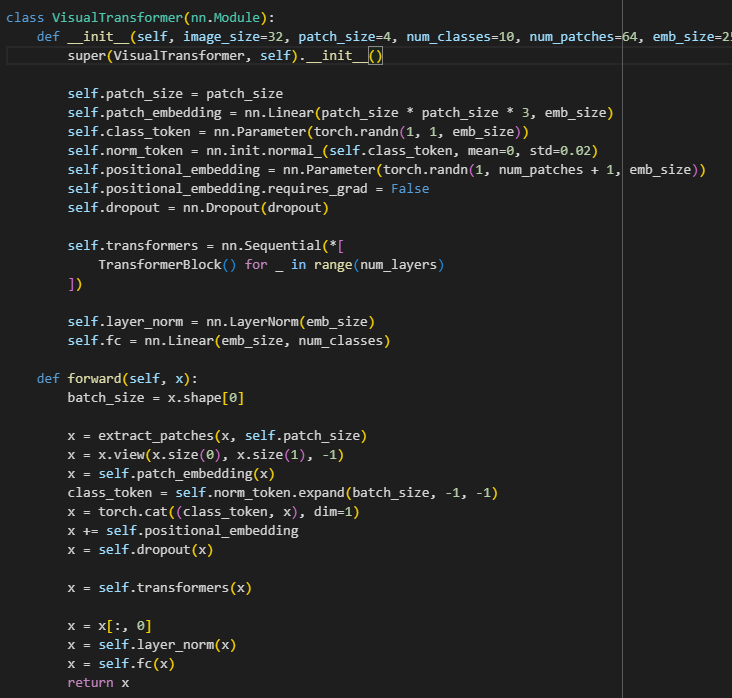


1. Definicja modelu

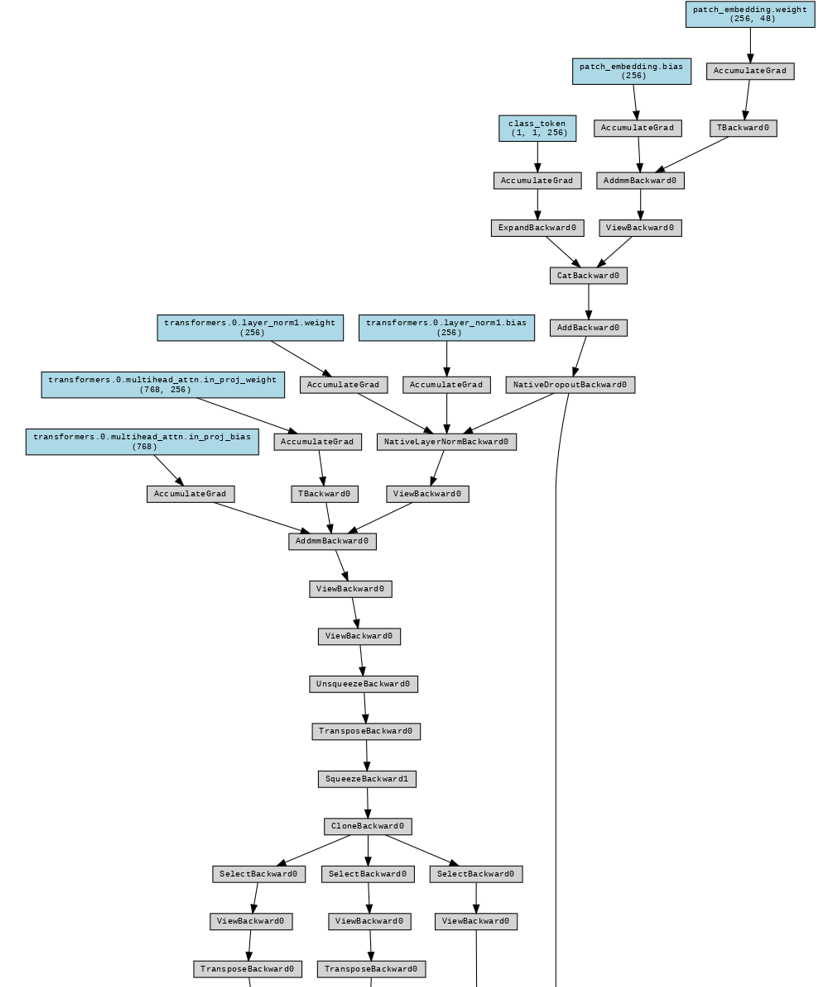
* Transformer Block



* VIT

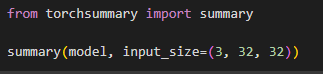


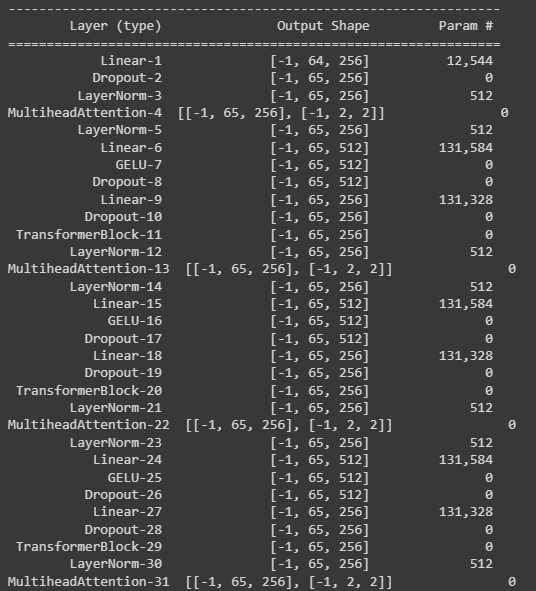
* computation graph (fragment ze względu na duży rozmiar)



Przez 5 warstw transformer block graf jest bardzo duży ale wydaje się być kompletny.

Żeby sprawdzić model użyłem też metody summary z biblioteki torchsummary.





Podobnie jak wcześniej załączam fragment bo bloki transformera powtarzają się

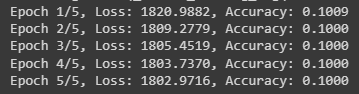
i znacznie zwiększają całe summary.

* kształt wynikowych danych dla batch’a z 64 obrazkami



Wszystko się zgadza 10 klas dla każdego z 64 obrazków.

1. Wyniki wytrenowanej sieci



W trakcie treningu sieć oscylowała z wartością losu w okolicach tej samej wartości i przystosowywała się do zwracania tej samej klasy niezależnie od danych.

Wartości gradientu zmieniały się w trakcie treningu ale ostatecznie nie udało mi się zlokalizować błędu.