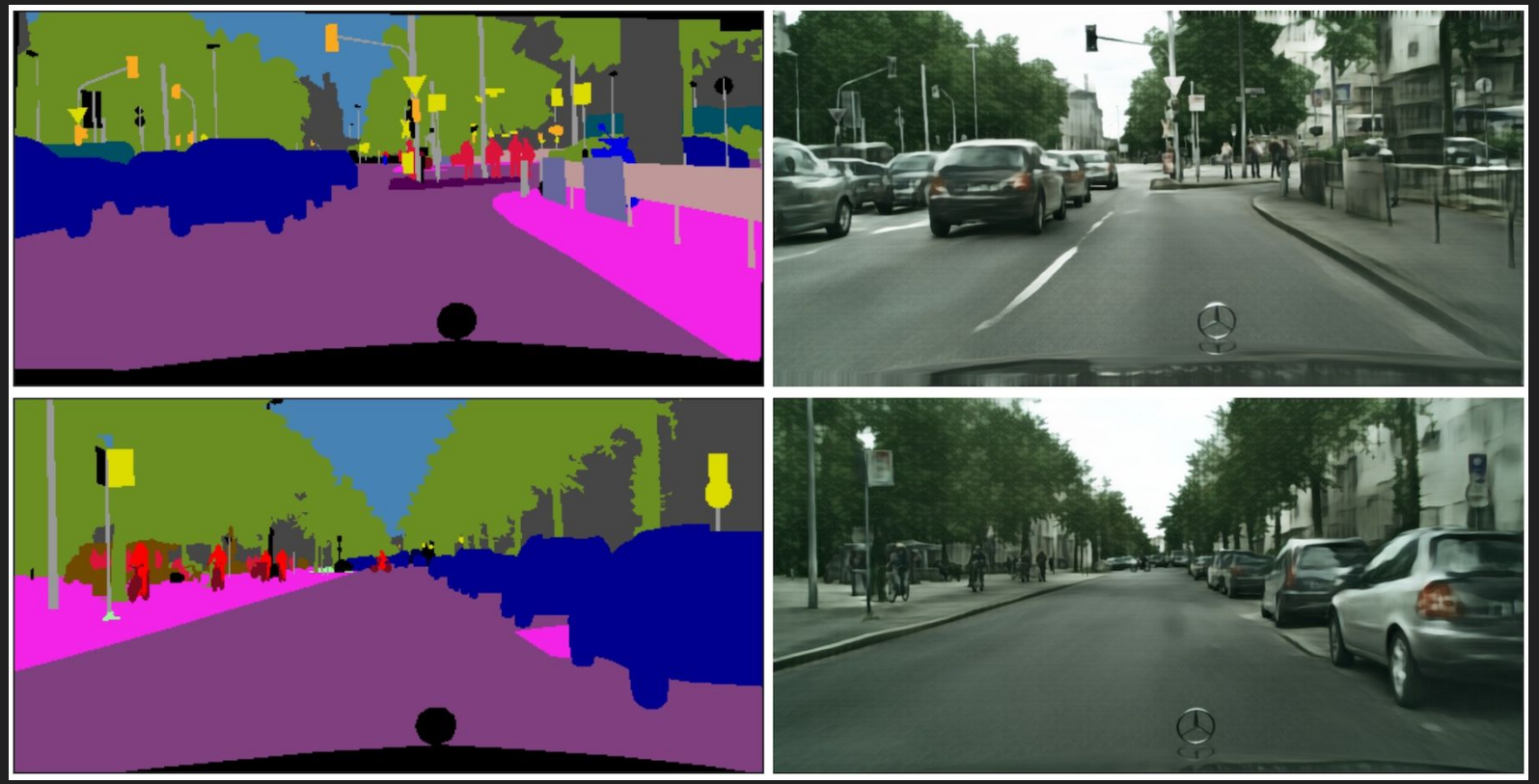
SPADE

Semantic Image Synthesis with **Sp**atially-**A**daptive (**De**)Normalization

Michał Królikowski

Semantic Image Synthesis

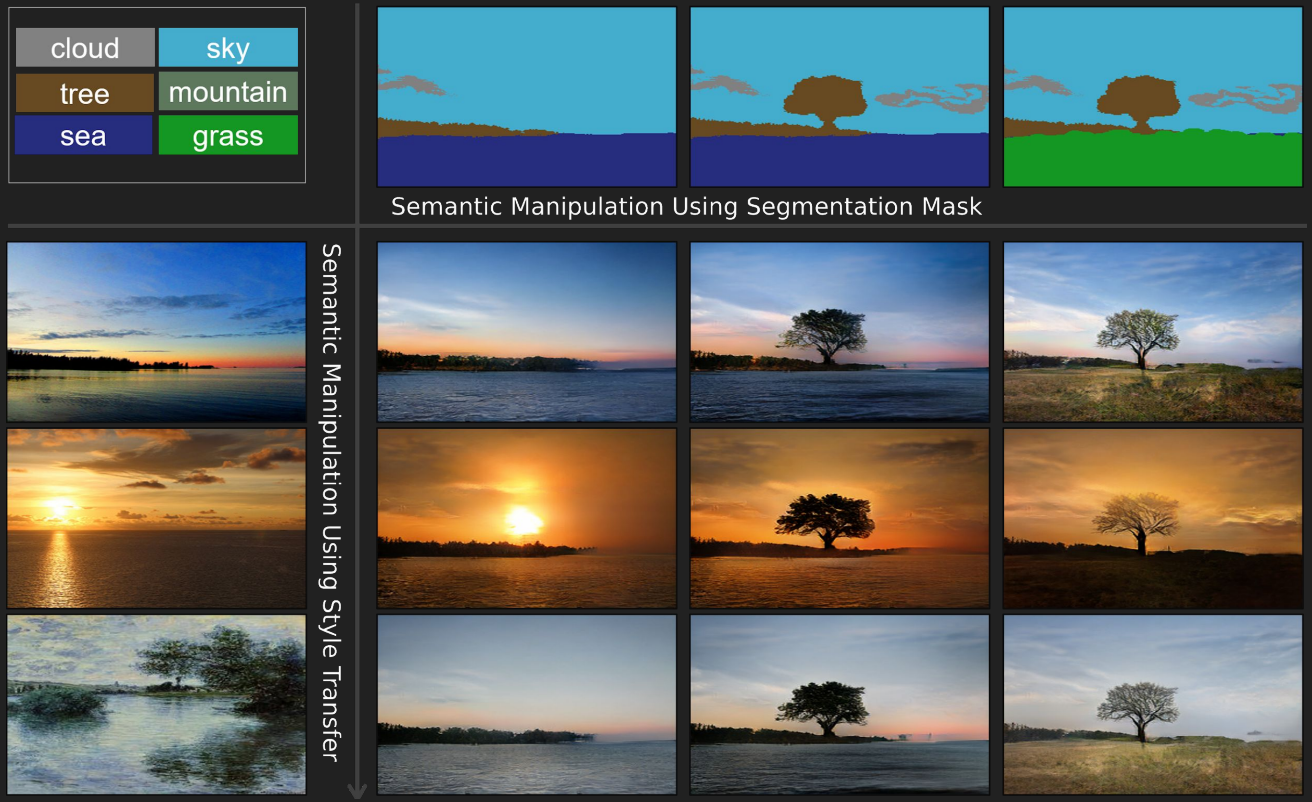


Conditional Image Synthesis

|

=

Więcej przykładów



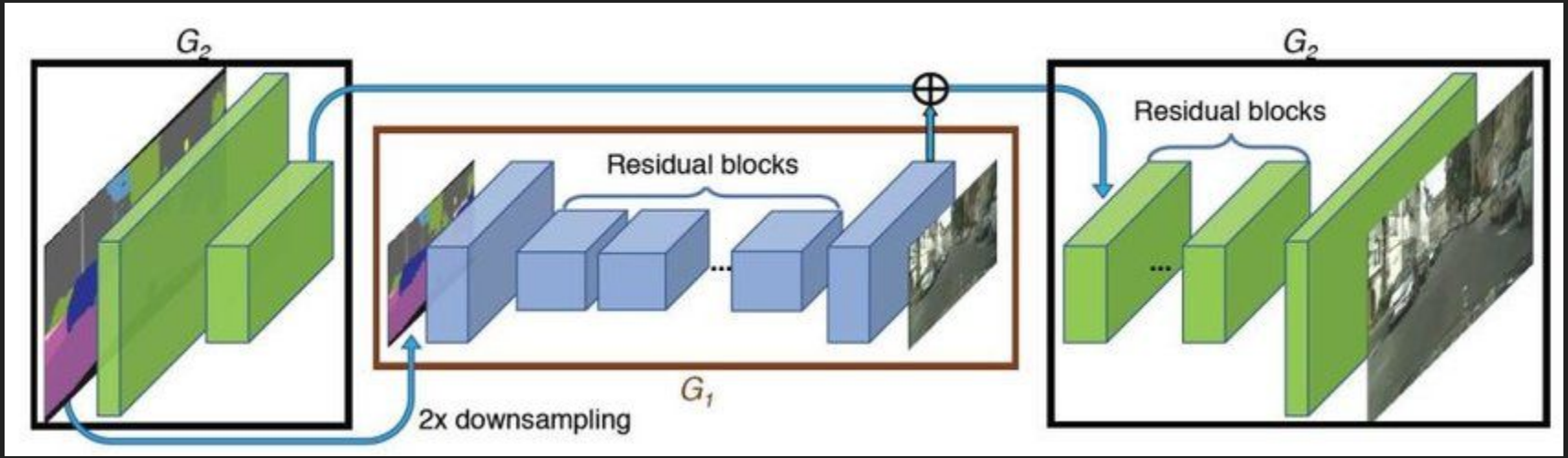
F*ckin’ magnets, how do they work?!

- korzystamy z **GAN**ów
  - *"the coolest idea in machine learning in the last twenty years"* - Yann LeCun, 2016
  - **G**enerative **A**dversarial **N**etwork
  - **2 sieci:** discriminator i generator
  - **intuicja: generator** ma za zadanie stworzyć na tyle realny obraz, żeby **discriminator** nie potrafił odróżnić go od ground truth (prawdziwych obrazów)
  - bardzo wiele zastosowań w transfer learningu, superresolution, super slow-motion etc.
- generator:
  - na wejściu przyjmuje wektor wartości inicjujących obrazek (często losowy)
  - na wyjściu dostajemy wygenerowany obraz
- discriminator:
  - na wejściu przyjmuje obraz (wygenerowany przez generator lub prawdziwy)
  - na wyjściu otrzymujemy klasyfikację (fake vs. true)

GANs in Semantic Image Synthesis

- jak wykorzystać GANy do syntezacji obrazów z semantic maps?
  - do **generatora** dodajemy **encoder**
- **encoder:**
  - na wejściu dostaje semantic map
  - na wyjściu daje wektor cech (wartości inicjujących obraz), który następnie trafia do **generator**a
- **intuicja**:
  - encoder ma za zadanie “*streścić”* wszystkie istotne informacje w latent vector
  - generator ma za zadanie je z niego odtworzyć
  - discriminator ocenia jak dobrze im poszło

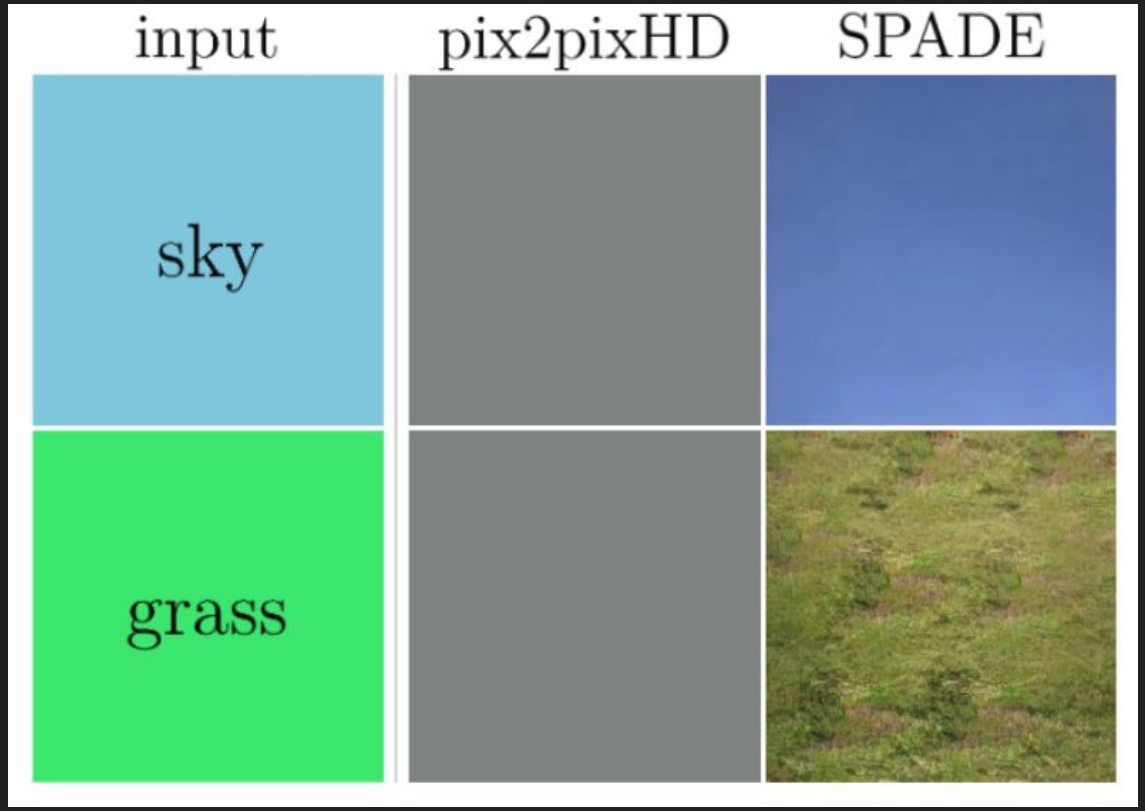
Przykładowa architektura generatora (pix2pixHD)



Batch Normalization

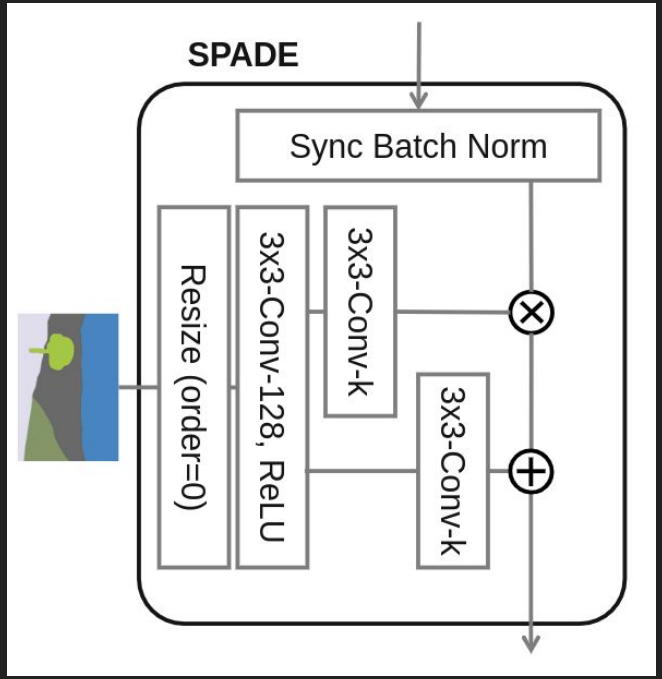
- stabilizuje naukę sieci neuronowych przez minimalizację **covariate shift**
- **intuicja**:
  - algorytmy ML w dużej mierze zakładają stabilność rozkładu danych
  - w sieciach przy każdym kroku (i w każdej warstwie) ten rozkład może się zmieniać, co utrudnia naukę sieci
  - **solution:** wrzucamy warstwy Batch Normalization, które normalizują wyjście warstwy do 0 mean i unit variance
- fajnie? fajnie, ale **niekoniecznie w Semantic Image Synthesis**
  - **problem:** tracimy informacje z Semantic Segmentation Map
  - ***“at best sub-optimal”*** – autorzy SPADE ;)

Przykład problemu



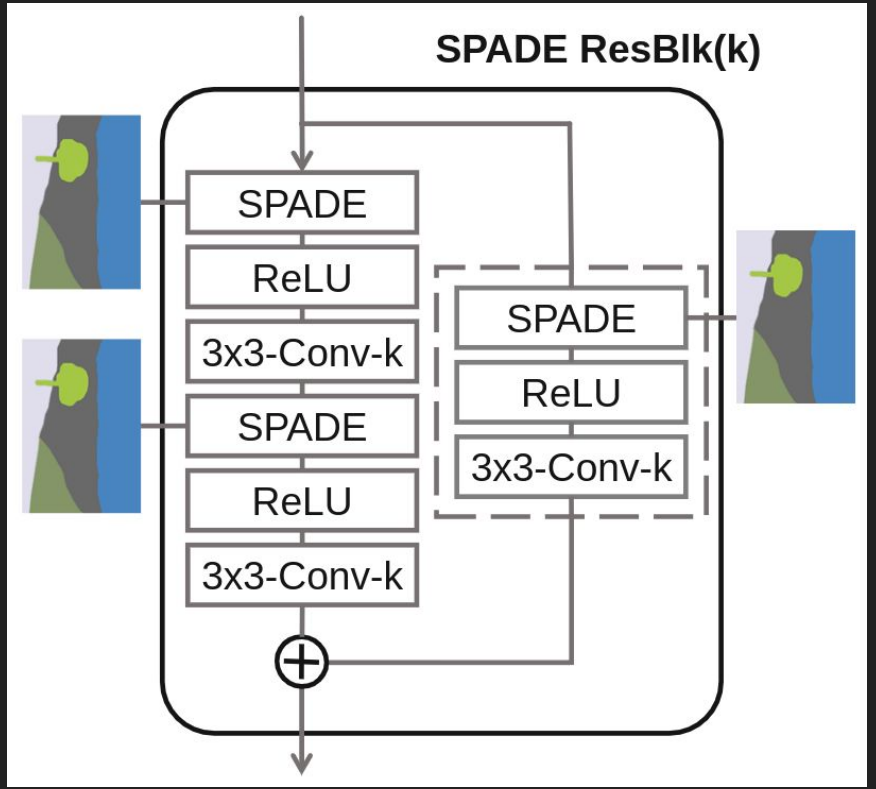
I wtedy wchodzi SPADE, cały na biało

- **Sp**atially **A**daptive (**De**)normalization
- **intuicja:**
  - warstwa normalizacji ma wiedzę o Semantic Segmentation Map (SSM)
  - SSM jest skalowany do obecnego rozmiaru warstwy (nearest, bez interpolacji!)
  - aplikujemy zwykły Batch Normalization…
  - …i **denormalizujemy** warstwę przy użyciu informacji z SSM
  - pomijam matmę, polecam przeczytać w oryginalnym artykule



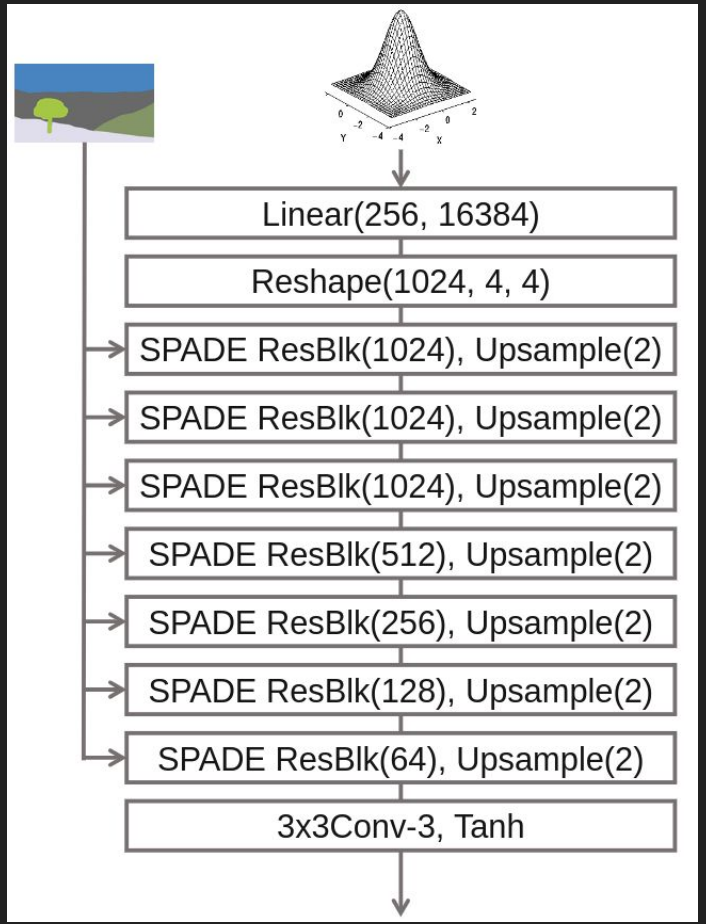
SPADE ResBlock

- korzystając z pomysłu Residual Neural Network (każdy zna?) budujemy ResBlock SPADE
- podstawowy “materiał” do budowy naszego generatora
- **SSM** jako input dla naszej sieci staje się zbędny – informacje te są przekazywane w odpowiednich warstwach



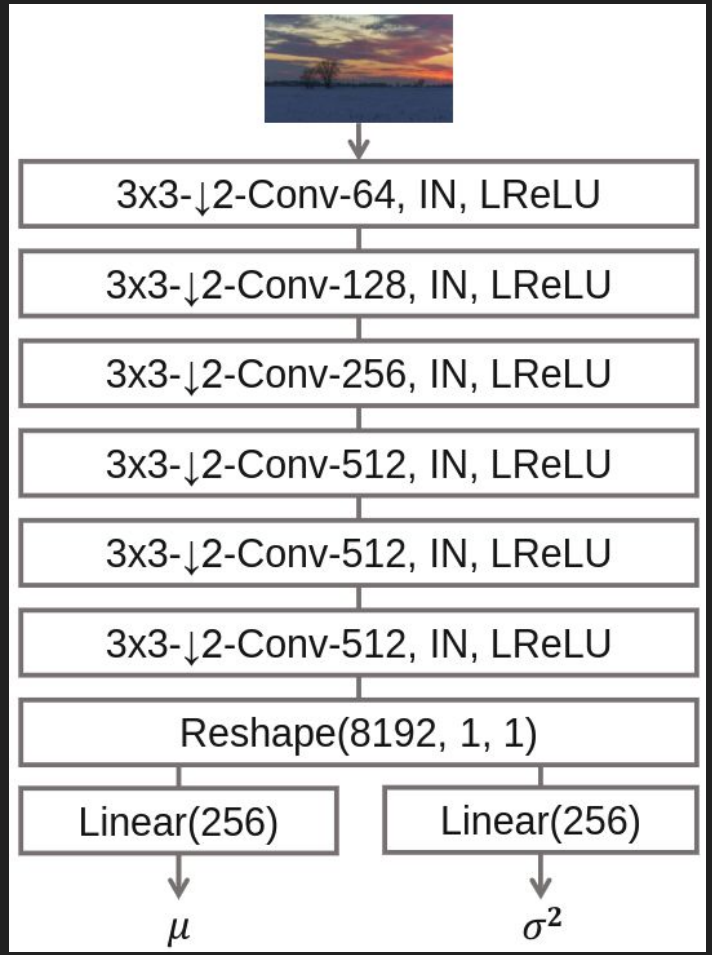
Architektura Generatora

- wejście stanowi wspomniany wcześniej **latent vector**, mówiący o stylu generowanego obrazu
- co zrobić, gdy chcemy skorzystać z stylu jakiegoś rzeczywistego zdjęcia?



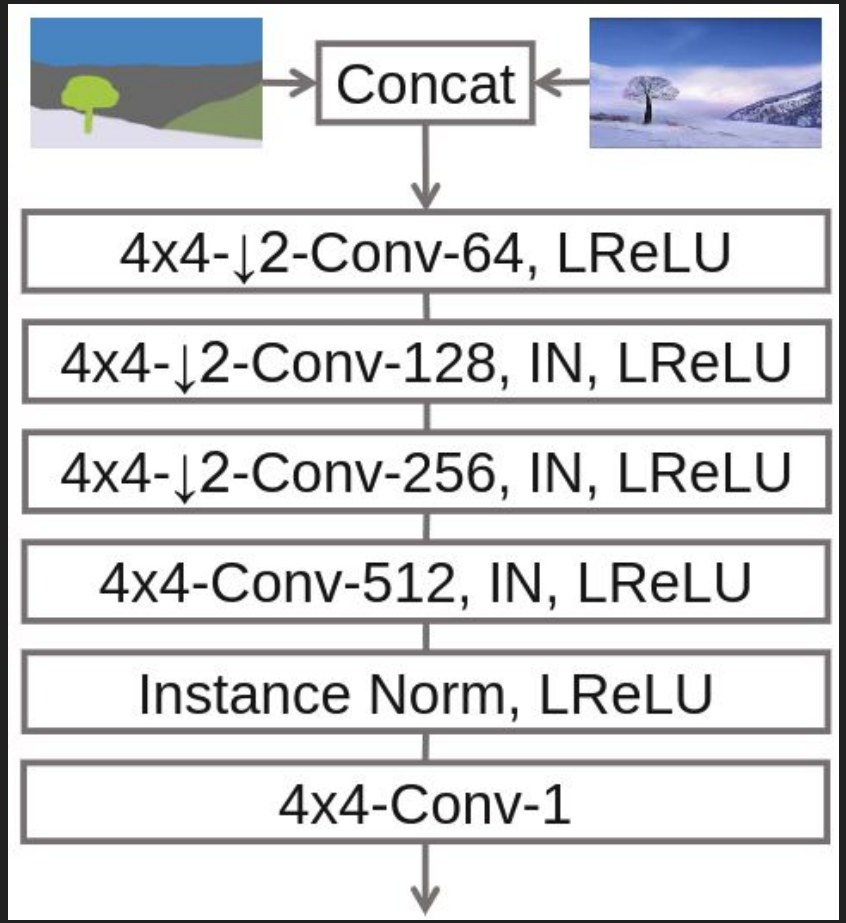
Architektura Encoder

- na wejściu podajemy zdjęcie, którego styl chcemy wyekstraktować
- na wyjściu dostajemy wektory średniej i wariancji – używając dystrybucji z takimi własnościami tworzymy latent vector, który stanowi wejście do generatora



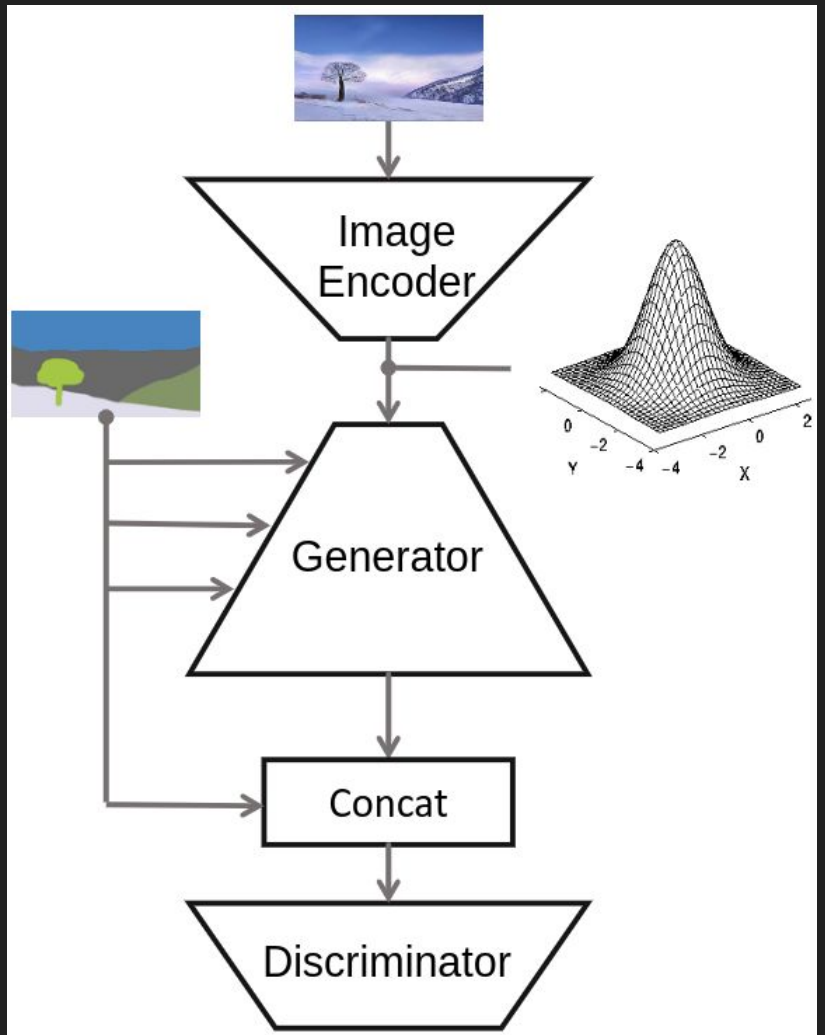
Architektura Discriminator

- na wejściu otrzymuje skonkanetowany SSM i obraz (może prawdziwy, może nie)
- daje wynik klasyfikacji obrazu
- dość standardowa architektura



Final product

- sklejamy wszystko w całość
- zostaje nakupić mocy obliczeniowej tysiące $$$
- i do boju!



The end – dzięki!