# Uniwersytet w Białymstoku

Instytut Informatyki
Kierunek: Informatyka

**Rok III semestr V**

Przedmiot

# Programowanie równoległe i rozproszone

**Sprawozdanie**

**Prowadzący:**
dr Krzysztof Szerszeń

**Autor:**
**Piotr Bołonkowski, nr indeksu : 80206**

## Zadanie

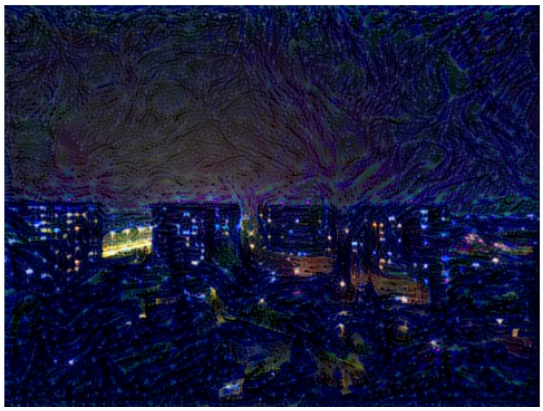
Poeksperymentuj z algorytmem **Neural Style Transfer (NTS)** opisanym w następującej pracy
**Leon A. Gatys, Alexander S. Ecker, Matthias Bethge**
**A Neural Algorithm of Artistic Style**
**https://arxiv.org/abs/1508.06576**
na podstawie następującego kodu
**https://keras.io/examples/generative/neural_style_transfer/**
w szczególności:
- poeksperymentuj z obrazkami wejściowymi (content) i (style)
- wpływem wyboru warstw sieci VGG-19 na wyniki przetwarzania
- opisz wykonaną pracę nad implementacją i rozwinięciem kodów oraz wyniki w postaci sprawozdania umieszczonego np. na github.com

Transfer stylu polega na wygenerowaniu obrazu o takiej samej „treści" jak obraz bazowy, ale w „stylu" innego obrazu (zazwyczaj artystycznego). Osiąga się to poprzez optymalizację funkcji straty, która składa się z 3 elementów: „utrata stylu", „utrata treści" i „całkowita utrata zmienności". Całkowita utrata zmienności narzuca lokalną ciągłość przestrzenną pomiędzy pikselami połączonego obrazu, nadając mu wizualną spójność.
Utrata stylu to miejsce, w którym utrzymuje się głębokie uczenie — jest ono definiowane za pomocą głębokiej, splotowej sieci neuronowej. Dokładniej, składa się z sumy odległości L2 między macierzami Grama reprezentacji obrazu bazowego i obrazu odniesienia stylu, wyodrębnionych z różnych warstw sieci convnet (uczonej na ImageNet). Ogólną ideą jest uchwycenie informacji o kolorze/tekstury w różnych skalach przestrzennych (dość duże skale – definiowane przez głębokość rozważanej warstwy).
Utrata treści to odległość L2 między cechami obrazu bazowego (wyekstrahowanego z głębokiej warstwy) a cechami obrazu złożonego, utrzymująca wygenerowany obraz wystarczająco blisko oryginalnego.

Jednym z eksperymentów, który postanowiłem wykonać była zmiana wartości w funkcji deprocess_image.

```
def deprocess_image(x):
    # Util function to convert a tensor into a valid image
    x = x.reshape((img_nrows, img_ncols, 3))
    # Remove zero-center by mean pixel
    x[:, :, 0] += 123.939
    x[:, :, 1] += 96.779
    x[:, :, 2] += 96.68
    # 'BGR'->'RGB'
    x = x[:, :, ::-1]
    x = np.clip(x, 0, 255).astype("uint8")
    return x
```

Można tutaj pobawić się w ustawianie „nasilanie" kolorów RGB. Są one ustawione od końca tak jak jest to napisane w komentarzu



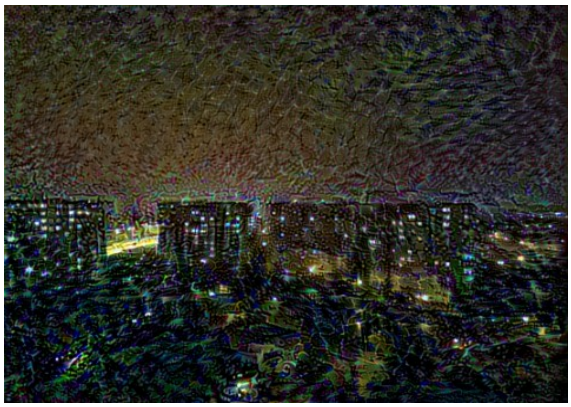
Funkcja style_loss  utrzymuje wygenerowany obraz blisko lokalnych tekstur obrazu odniesienia stylu. Opiera się na macierzach gramów (które rejestrują styl) mapy funkcji z obrazu referencyjnego stylu i z wygenerowanego obrazu

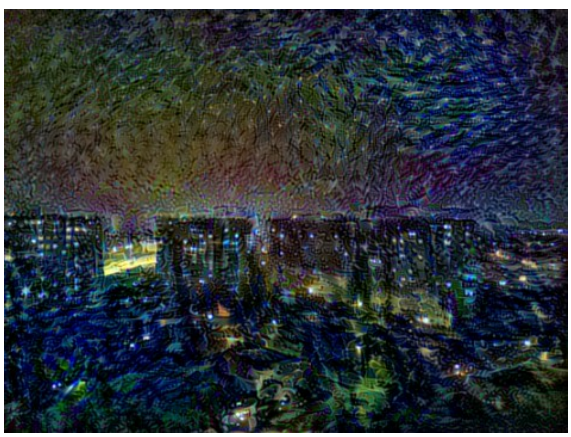
Funkcja content_loss utrzymuje wysoką reprezentację wygenerowanego obrazu zbliżoną do obrazu bazowego.

Funkcja total_variation_loss to  utrata regularyzacji, która utrzymuje lokalnie spójny wygenerowany obraz.

Kolejnym eksperymentem była zabawa ze zmienianiem wartości zmiennej content_weight. Postanowiłem ustawić go na wartość `content_weight = 2.5e-4.`s
Wyniki były bardzo interesujące, więc postanowiłem porównać obrazki po wykonaniu 800 iteracji z wartością domyślną oraz z moja wartością. Obrazy generowane są co 100 iteracji.

Wersja content_weight = 2.5e-4:





Wersja content_weight = 2.5e-8;