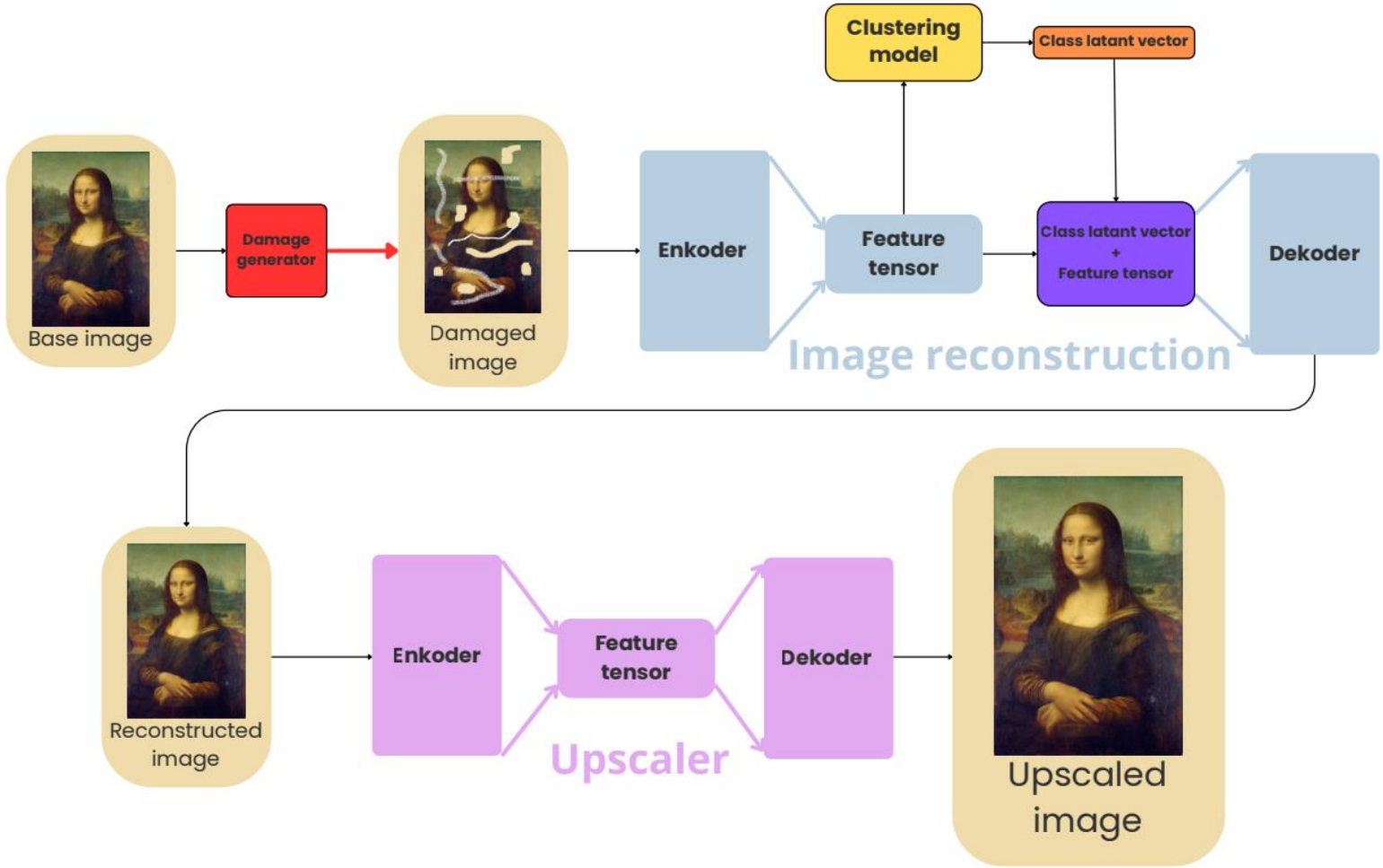
UCZENIE NIENADZOROWANE PROJEKT

Sebastian Borukało (259188)
Jakub Bukała (259189)
Jakub Smakowski (259209)

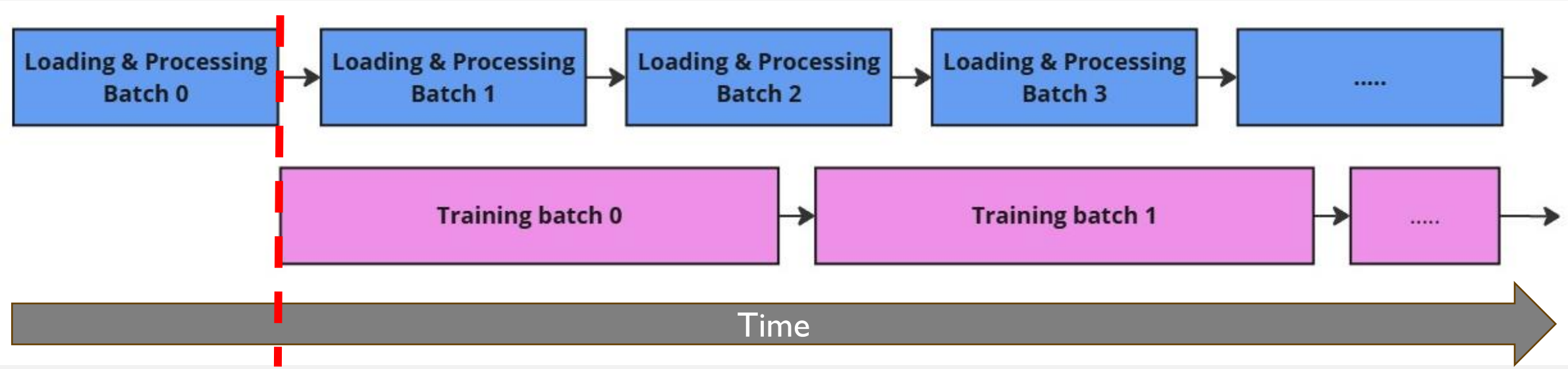
AGENDA

- Schemat projektu
- Dane
- Autoenkoder
- Klasteryzator
- Upscaling
- Prezentacja GUI

SCHEMAT PROJEKTU



ROZDZIAŁ: DANE



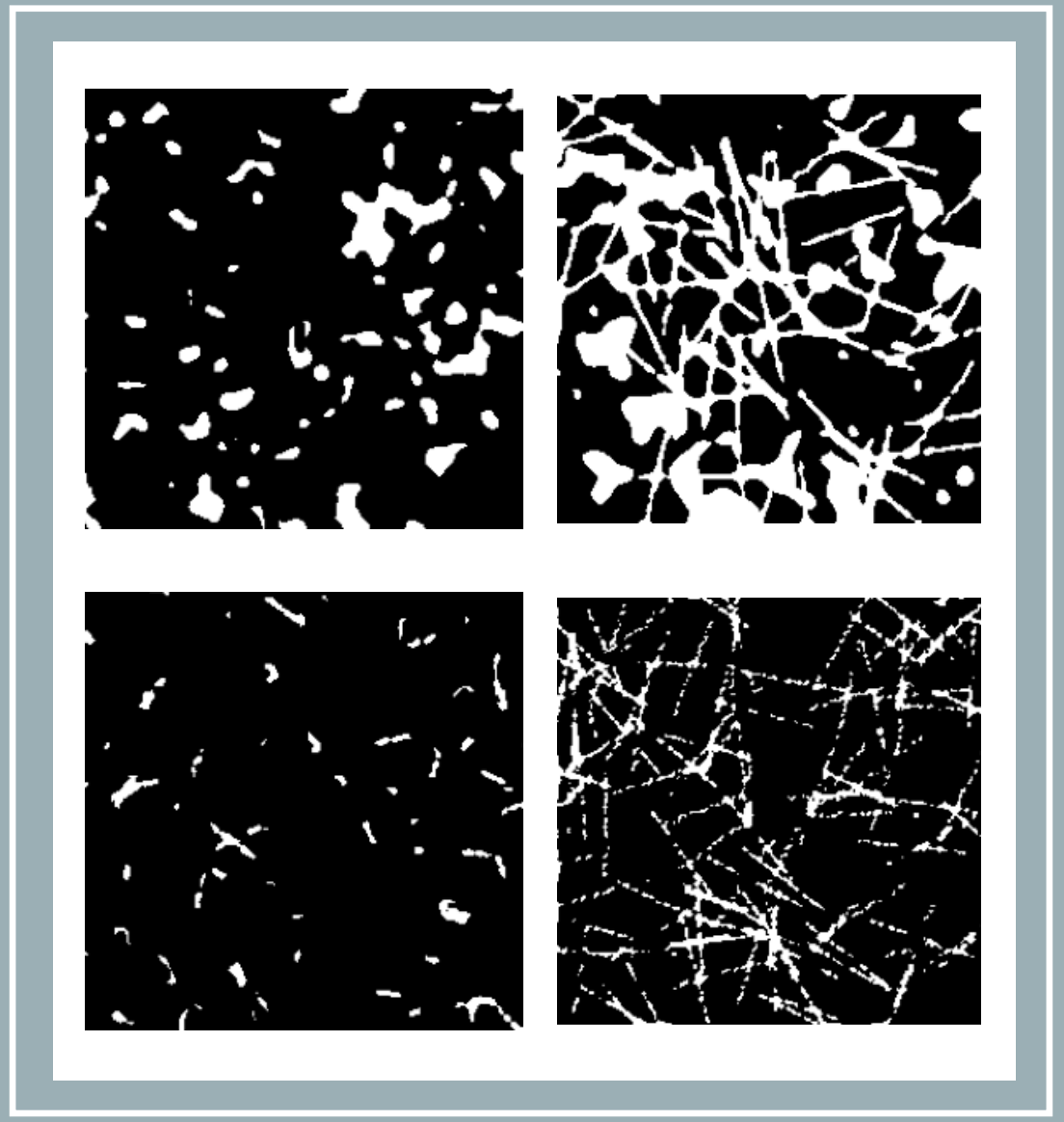
ZARZĄDZANIE DANYMI ; ŁADOWANIE DANYCH

- Dane ładujemy podczas treningu bezpośrednio z bazy danych pobranej w formacie .arrow
- Robimy to w sposób asynchroniczny tzn. równolegle do treningu ładujemy batche potrzebne w następnych krokach.
  Dzięki temu przyspieszamy trening oraz oszczędzamy pamięć.
- Główny preprocessing i przygotowanie próbki jest dokonywane podczas ładowania batcha co pozwala na jeszcze większą optymalizację

METODA GENEROWANIA USZKODZEŃ

- Uszkodzenia generujemy podczas ładowania batchy:
  - Pełni to funkcję regularyzacji (uszkodzenia zawsze są inne)
  - Zapewnia wydajność (integracja z dataloaderem)

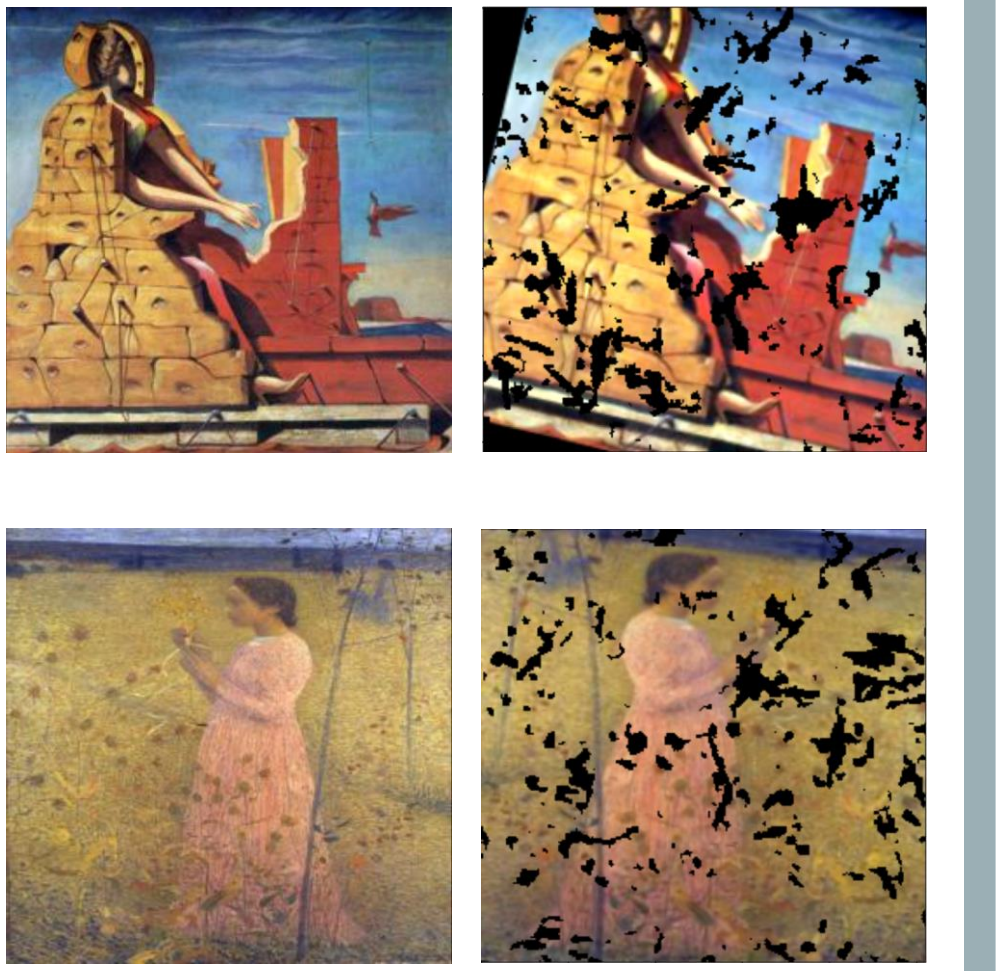
- Uszkodzenia generowane są przez łączenie różnych rodzajów szumów.



AUGMENTACJA

- Dodatkowo augmentujemy obraz poprzez losową:
  - jasność
  - rotację
  - kontrast
  - saturację
  - ucięcie (crop)
  - odbicie lustrzane

- **Zarówno generacja uszkodzeń jak i augmentacja są zoptymalizowane aby działały płynnie na GPU**



ROZDZIAŁ: AUTOENKODER

EKSPERYMENTY ; PODEJŚCIE

- Nasz Inpainter jest pojedynczym modelem przyjmującym każdy obraz. (Conditional input)

- Naprawia obraz w odpowiedni sposób za pomocą własnej analizy oraz wejścia w postaci klasy (wektor cech jest produkowany przez enkoder, ale sama klasteryzacja jest poza autoenkoderem)

- Próbowaliśmy różnych wariacji architektur.
  Zdecydowaliśmy się na wariant V5
  Wszystkie z nich były oparte na tym samym schemacie, jednak różniły się takimi rzeczami jak funkcje aktywacji, ilość filtrów, głębokość sieci itp.

Parametry tych sieci oraz ich treningu są dostępne w dokumentacji projektu (github, comet)

*Na model V5 zdecydowaliśmy się z powodu dobrych wyników w porównaniu do innych modeli przy najlepszej w naszej ocenie separacji klas

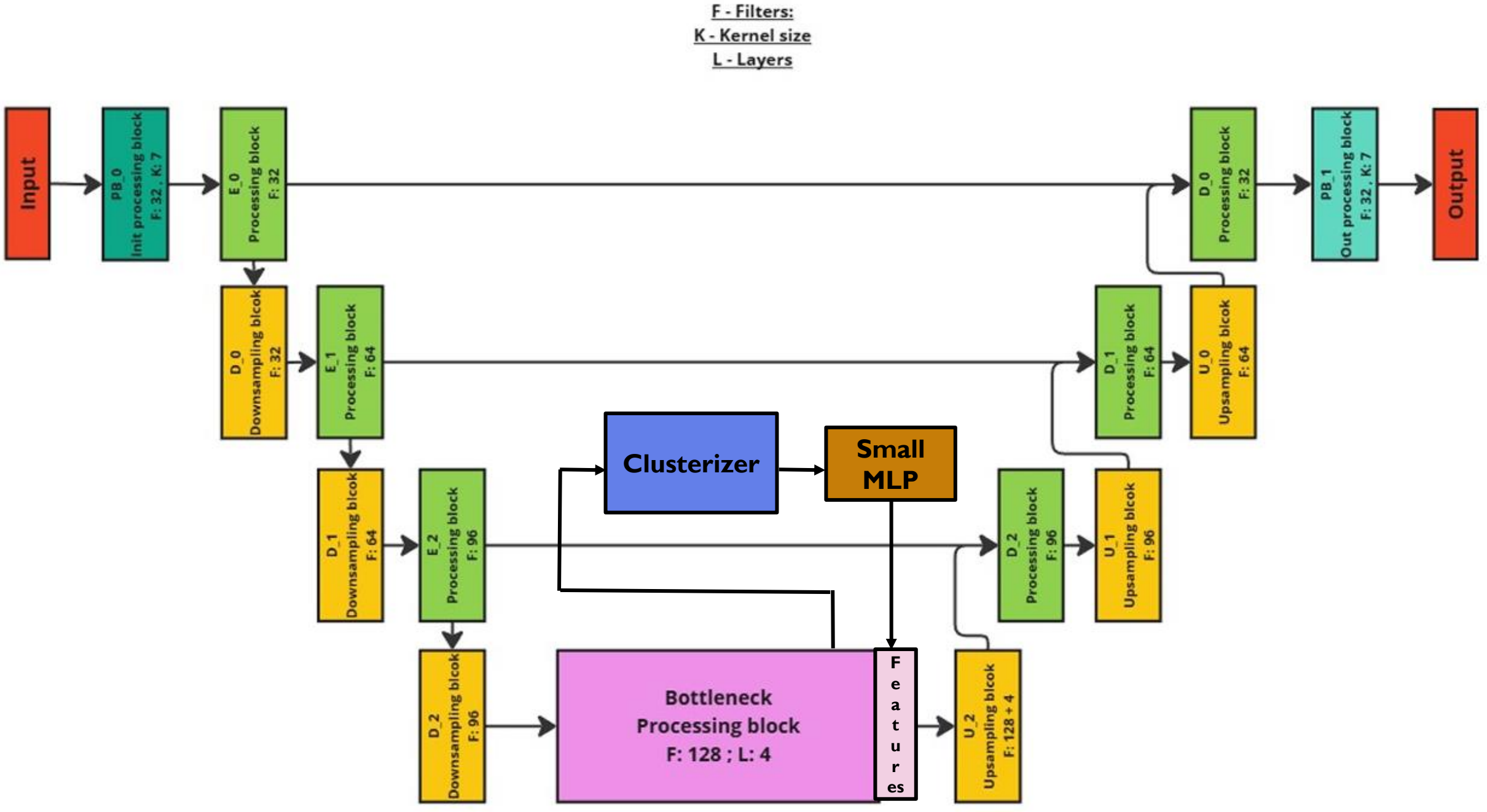
EKSPERYMENTY ; PODEJŚCIE

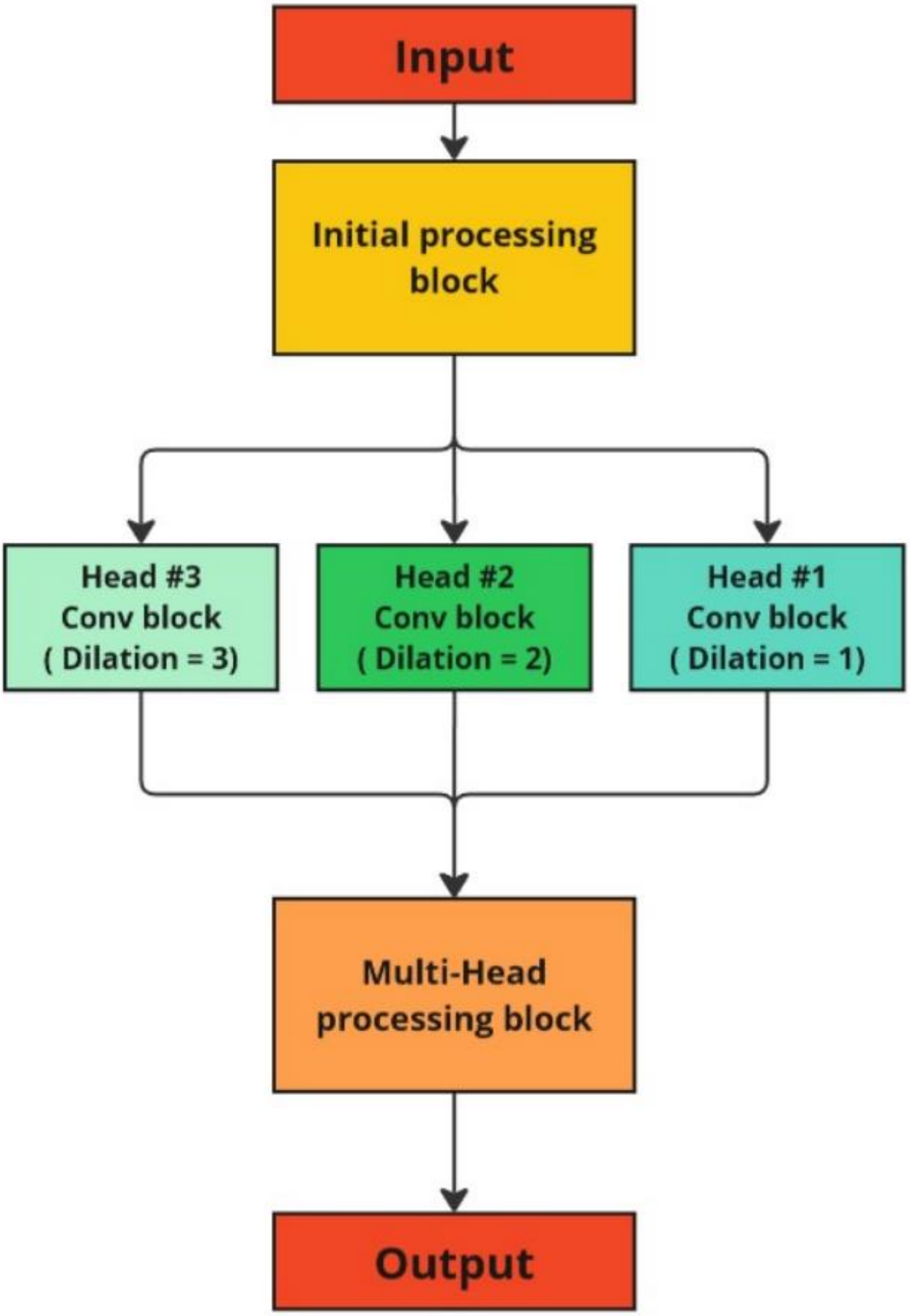
- Po wybraniu modelu V5, rozszerzyliśmy go o nowe sub-modele

  Ostatecznie otrzymaliśmy modele wytrenowane z uwzględnieniem różnych strategii dotyczących klasy obrazu

- **Sposób treningu:**
  - BASELINE: wejście klasy było zawsze 0. Brak istotnej informacji
  - REAL: wejście klasy pochodzące z naszego klasteryzatora
  - MAX_CLASS: wejście klasy było podane sztucznie. 100% dokładność

U-Net based InPainter major scheme



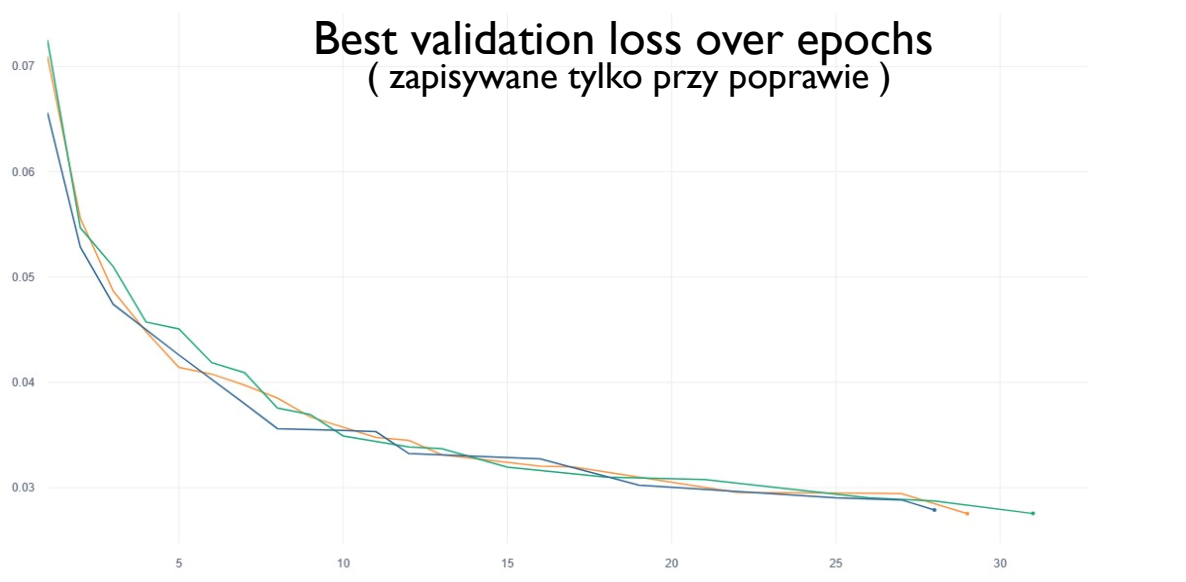


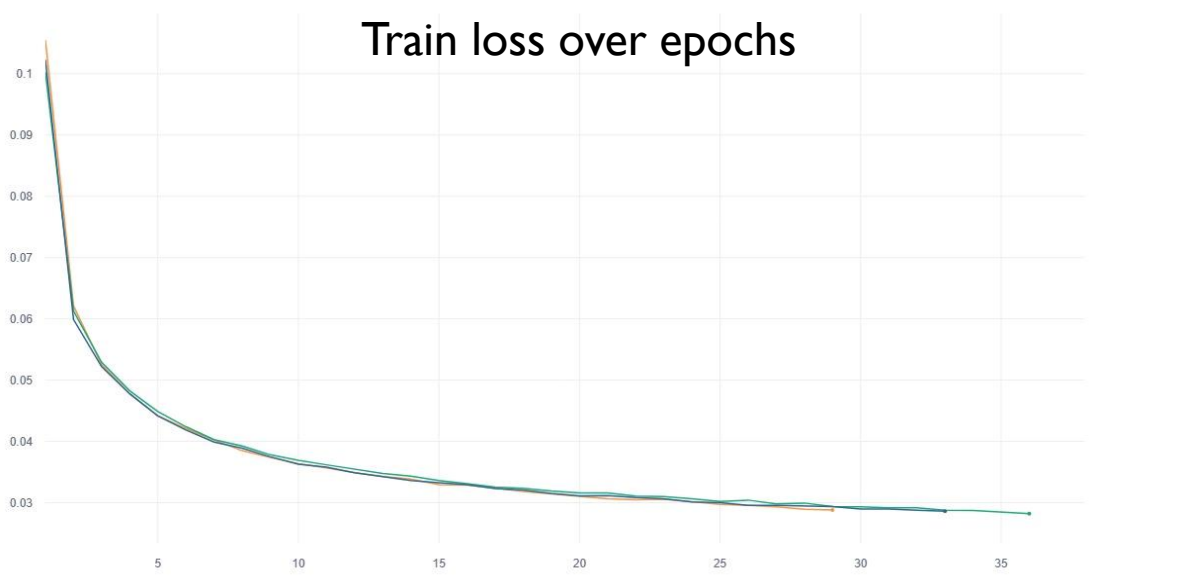
Processing block scheme

WYNIKI

BASELINE
REAL
MAX_CLASS

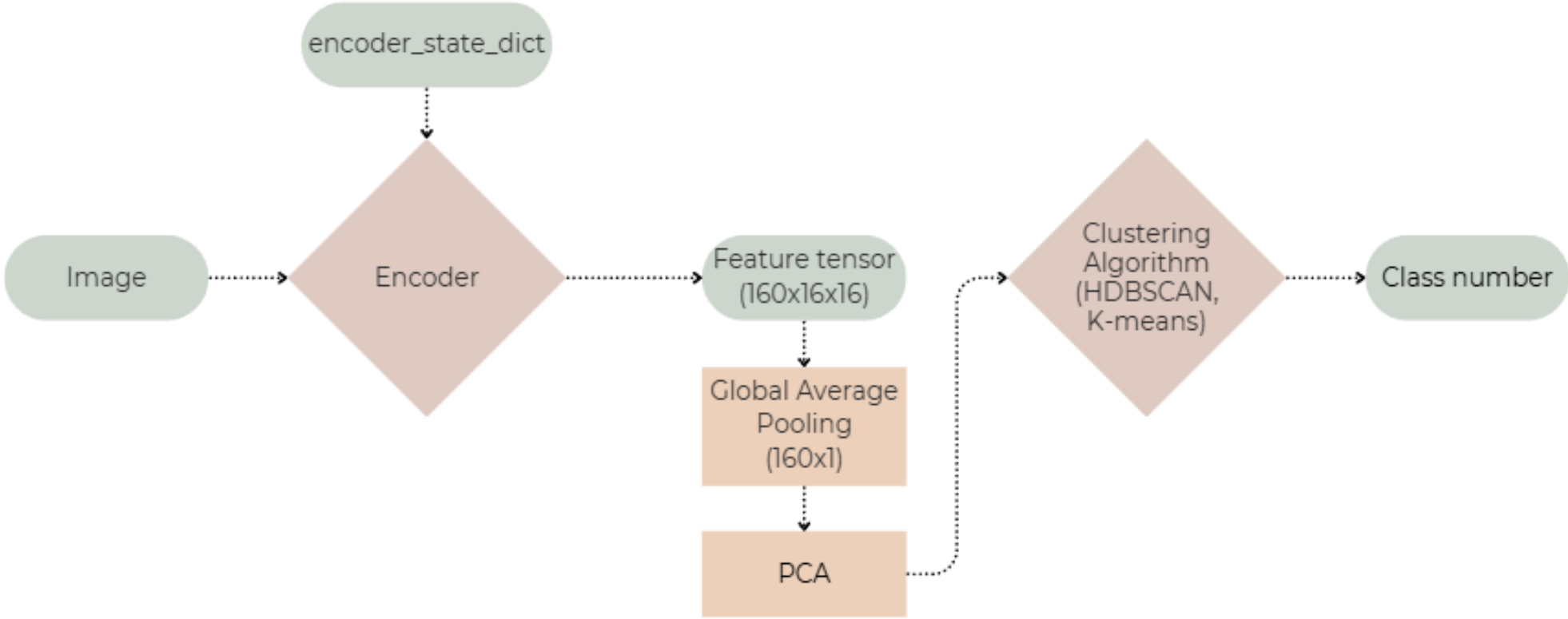
Model	Average loss	PSNR (dB)	LPIPS	Epochs	% difference(to baseline) More is better
DAMAGED	0.1217	16.89	0.30001	---	----
BASELINE	0.0302	30.94	0.0817	31	100 %
REAL	0.0307	31.02	0.0810	28	99.8 %
MAX_CLASS	0.0301	30.97	0.0817	29	100,1 %



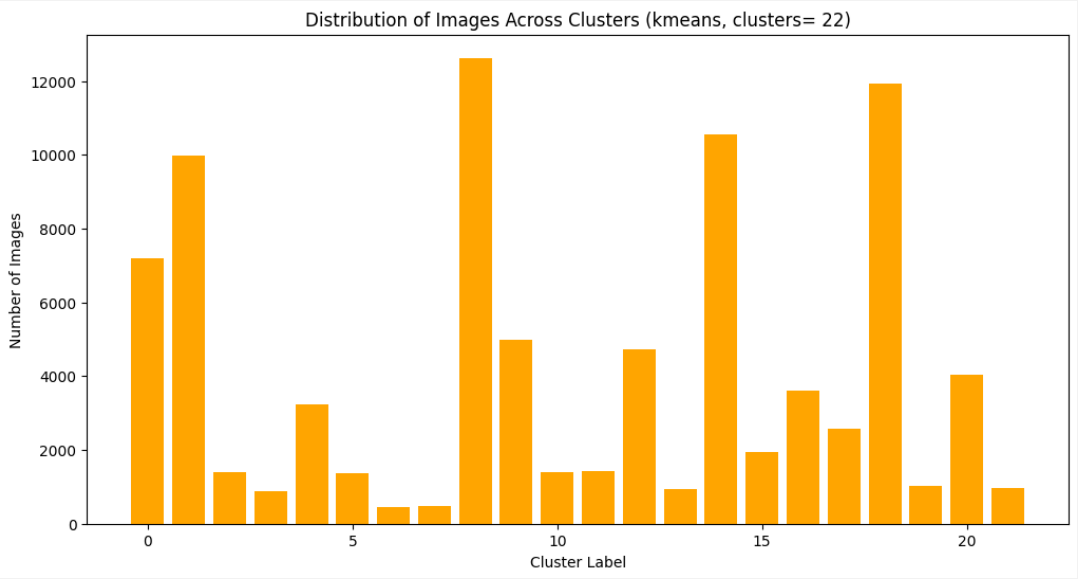


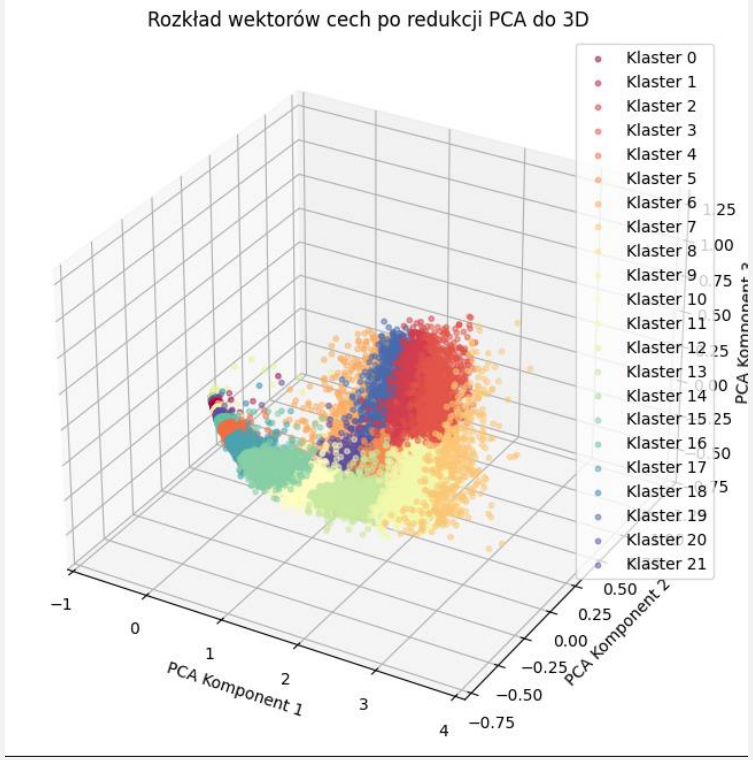
*LPIPS: Learned Perceptual Image Patch Similarity

ROZDZIAŁ: KLASTERYZATOR



SCHEMAT KLASTERYZATORA





HDBSCAN + KMEANS

Czym jest?
Algorytm klasteryzacji oparty na gęstości, rozwinięcie DBSCAN, automatycznie wykrywa klastry o różnej gęstości i usuwa punkty odstające.

Parametry

HDBSCAN(min_samples=2, min_cluster_size=40)

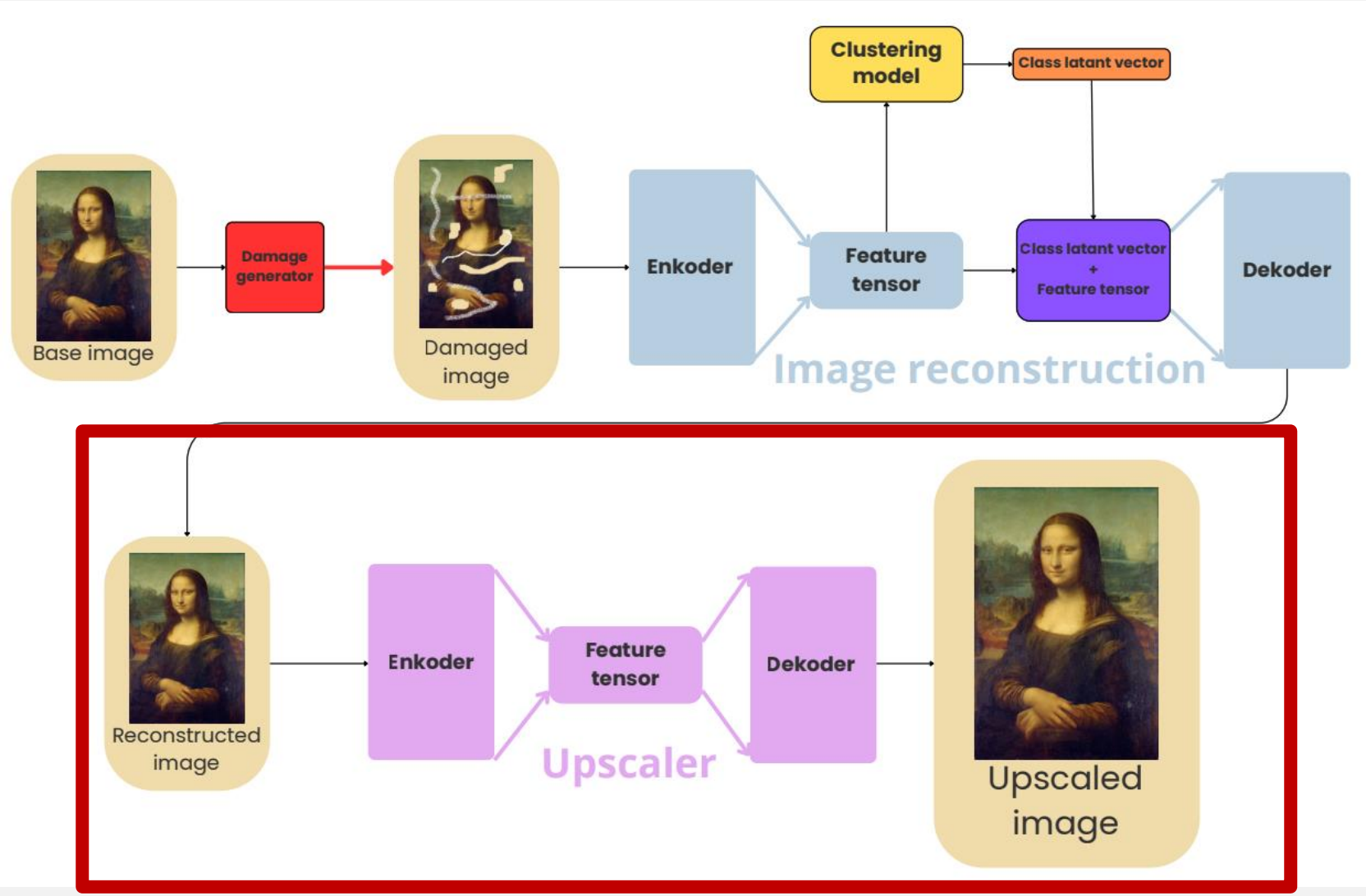
PCA(n_components=10)

Zastosowanie

HDBSCAN został użyty do określenia liczby klastrów jakie, są możliwe do wykrycia. Wyznaczona liczba była następnie przekazywana do algorytmu Kmeans w celu właściwej klasteryzacji zbioru

ROZDZIAŁ: UPSCALING

SUPER RESOLUTION/UPSCALING



1. PRZYGOTOWANIE DANYCH

Źródło danych:
81 444 obrazów ze zbioru huggan/wikiart
jako obrazy referencyjne wysokiej rozdzielczości
(High-Resolution - HR).

Przygotowanie HR:
Obrazy źródłowe zostały wycentrowane i przycięte
(center crop) do jednolitej rozdzielczości 512x512.

Cel:
Model nauczony zwiększania rozdzielczości
z 256x256 do 512x512
(współczynnik skalowania ×2).

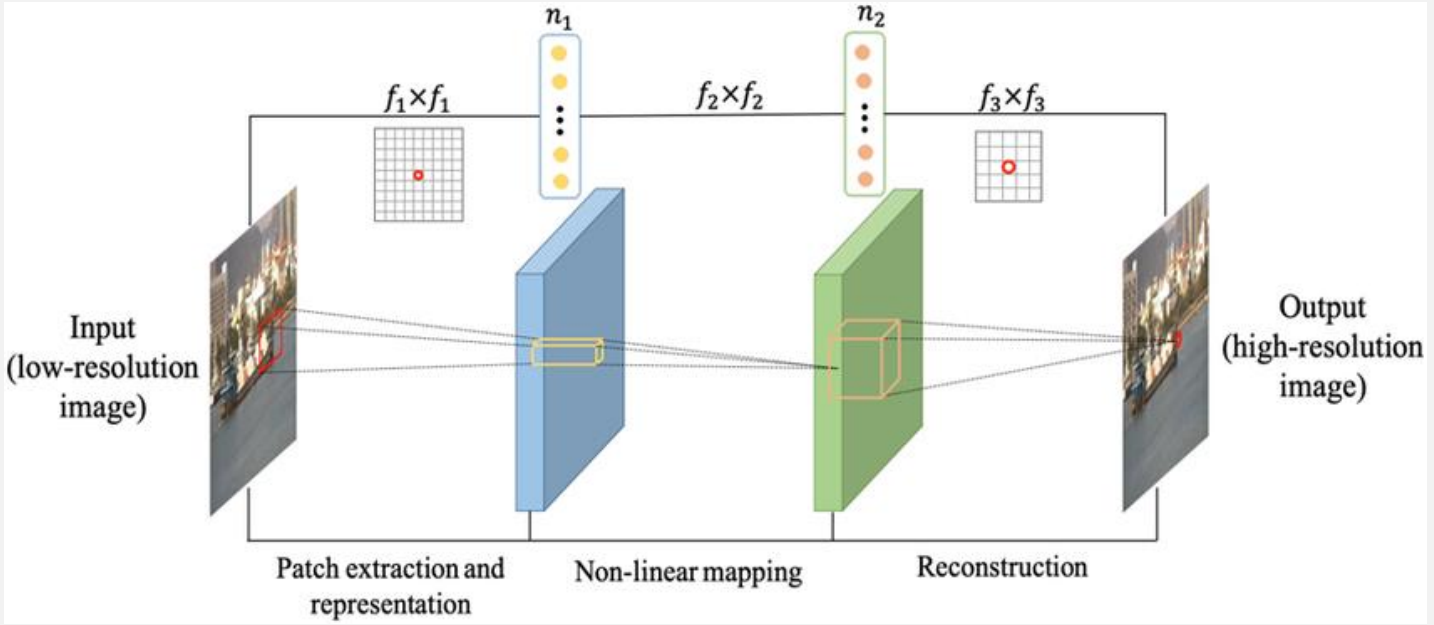
2. GENEROWANIE DANYCH TRENINGOWYCH (UCZENIE SAMONADZOROWANE)

Pipeline degradacji (dla każdego batcha):

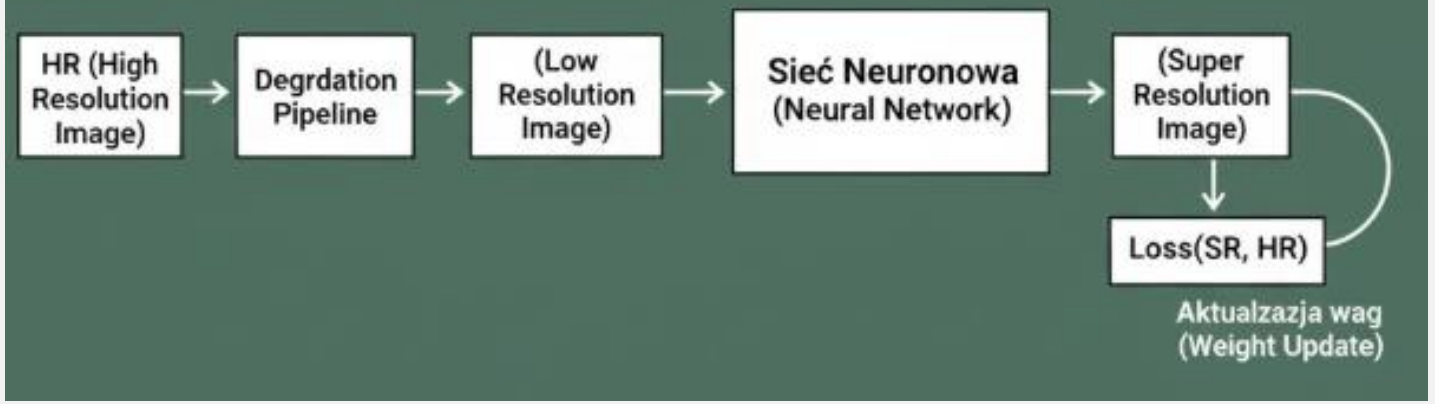
1. Pobierany jest obraz referencyjny HR 512x512.
2. Stosowana jest degradacja:
- Downscaling z użyciem metody interpolacji Bicubic.
3. Rezultatem jest obraz LR o rozdzielczości 256x256.

3. PROCES TRENINGOWY

- **Architektura**:
  Super-Resolution CNN (skala x2)
- **Funkcja Straty:**
  L1 Loss (zamiast MSE) – dla zachowania ostrości krawędzi.
- **Optymalizacja:**
  Adam (Learning Rate: 1e-4).
  Scheduler: ReduceLROnPlateau (Redukcja LR o 50% po 3 epokach braku poprawy).
- **Wydajność:**
  Mixed Precision (FP16) – przyspieszenie treningu i mniejsze zużycie VRAM.
- **Ewaluacja:**
  PSNR obliczany z pominięciem krawędzi (Border Shaving).



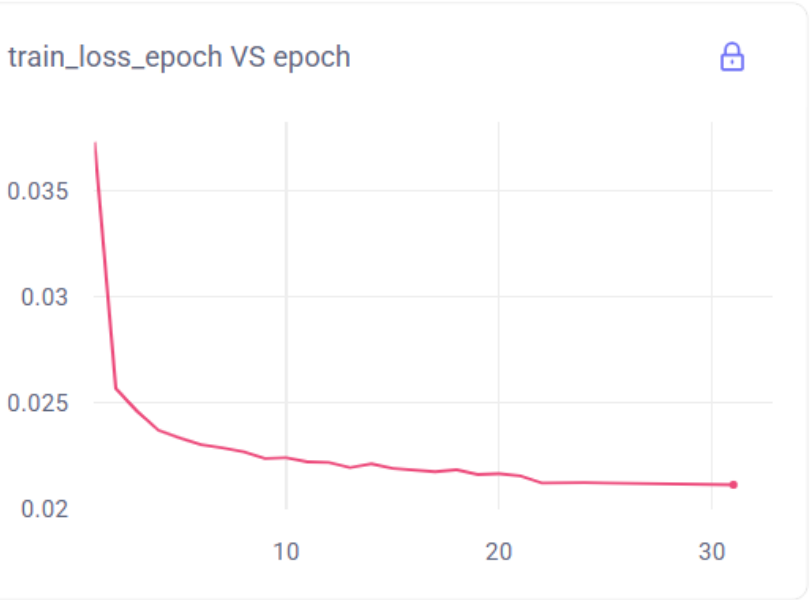
Pętla treningowa - dla każdej partii (batcha) danych:

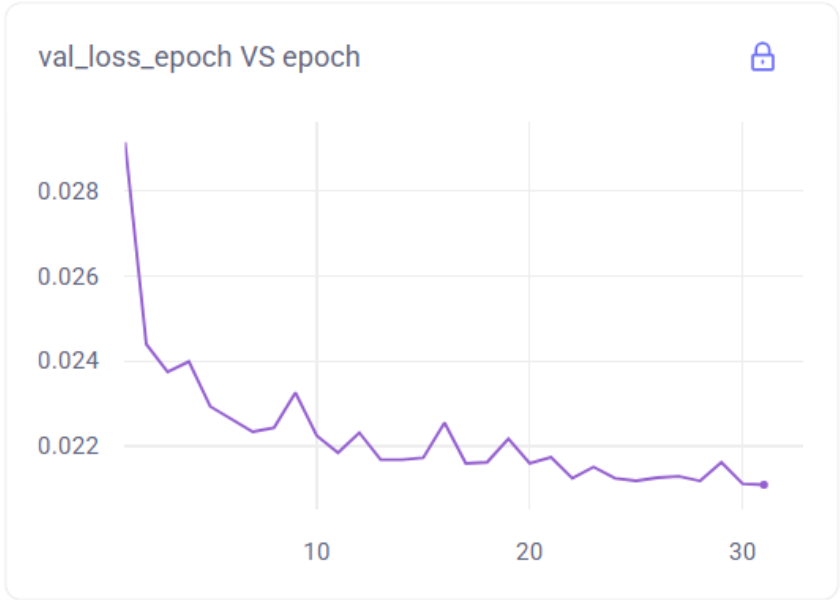


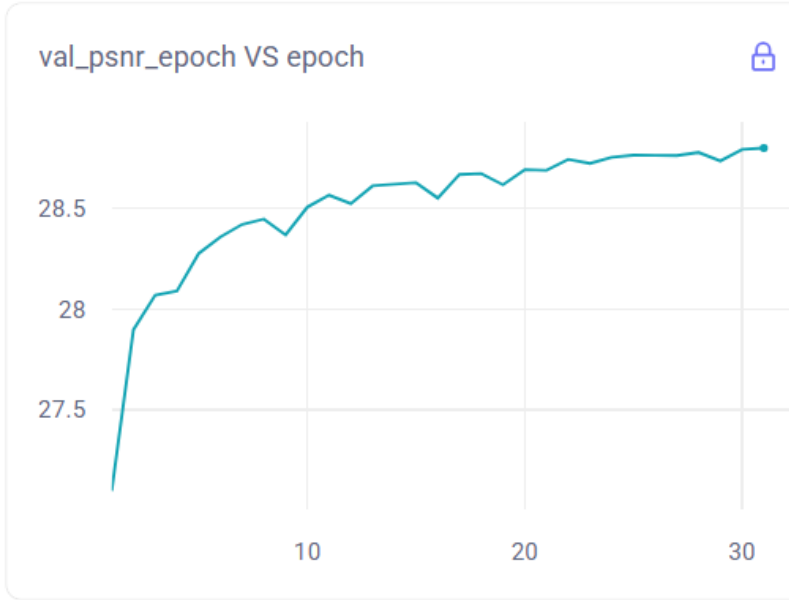
```python
# --- KLASA SRCNN ---
class SRCNN(nn.Module):
    def __init__(self):
        super(SRCNN, self).__init__()
        # Etap 0: Powiększenie wstępne (Bicubic) do rozmiaru docelowego
        self.upsample = nn.Upsample(scale_factor=2, mode='bicubic', align_corners=Fal

        # Etap 1: Ekstrakcja cech (Patch extraction and representation)
        self.conv1 = nn.Conv2d(3, 64, kernel_size=9, padding=4)
        self.relu1 = nn.ReLU(inplace=True)

        # Etap 2: Mapowanie nieliniowe (Non-linear mapping)
        self.conv2 = nn.Conv2d(64, 32, kernel_size=5, padding=2)
        self.relu2 = nn.ReLU(inplace=True)

        # Etap 3: Rekonstrukcja (Reconstruction)
        self.conv3 = nn.Conv2d(32, 3, kernel_size=5, padding=2)

    def forward(self, x):
        x = self.upsample(x)
        out = self.relu1(self.conv1(x))
        out = self.relu2(self.conv2(out))
        out = self.conv3(out)
        return out
```

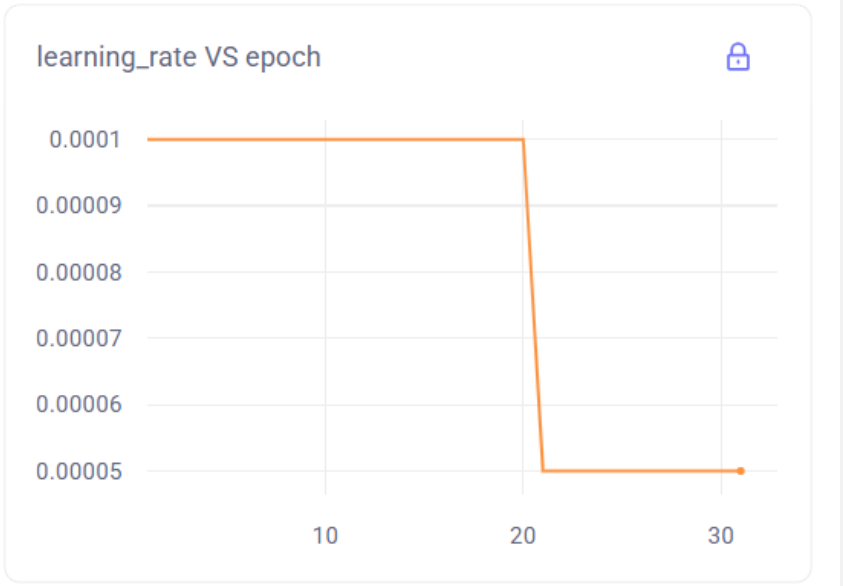
SUPER-RESOLUTION CNN

COMET ML









PSNR ≈29dB.

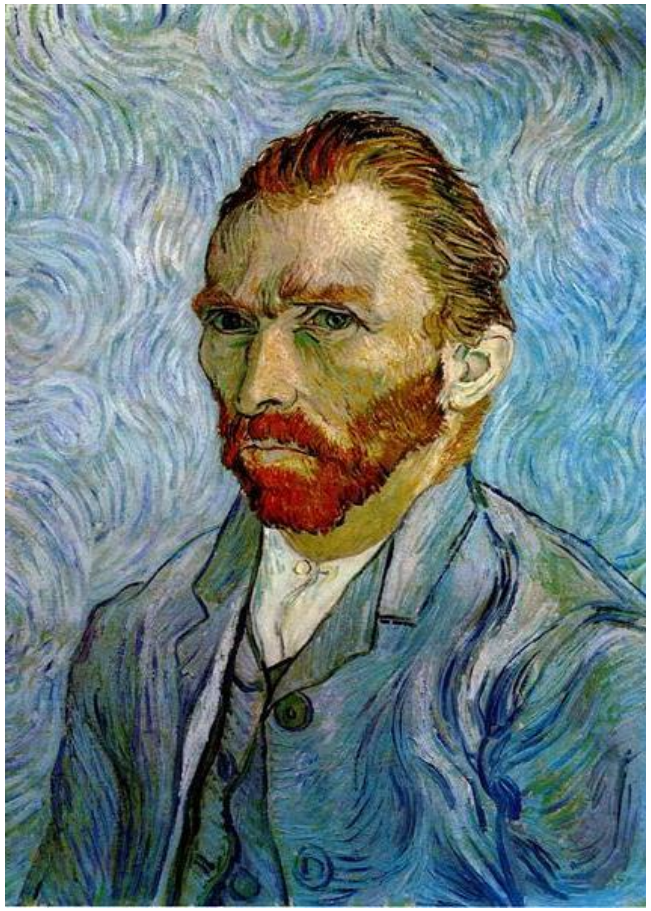
COMET ML

LR
powiększone metodą
BICUBIC

SR
wynik modelu SRCNN

HR
Oryginał

PREZENTACJA GUI



DZIĘKUJEMY ZA UWAGĘ :)

FALLBACK

- Zdjęcia na wszelki wypadek