# Wirtualna szminka

ALEKSANDRA WALCZYBOK

Grudzień 2024

# Problem

Problem polegał na segmentacji ust w czasie rzeczywistym, a następnie nakładaniu wybranego koloru szminki przez użytkownika.

START Research podobnych projektów i technologii.

1 Wybór zbioru danych uczących oraz preprocessing danych.

2 Implementacja oraz uczenie modelu segmentacyjnego.

3 Testowanie modelu w czasie rzeczywistym, wstępny skrypt z kamerką.

4 Przygotowanie interfejsu graficznego.

# Projekty produkcyjne

MAYBELLINE NEW YORK

**Virtual Makeup Try On - Makeup Tool - Makeup Tips**

Get a virtual makeover with the virtual makeup tool by Maybelline. Instantly try on eye, face & lip...

Maybelline New York

**Virtual Makeup Try On – L'Oréal Paris**

Wypróbuj symulator makijażu od L'Oréal Paris, przetestuj najnowsze trendy w makijażu ust, oczu i...

lorealparis.pl

# ŹRÓDŁA “STARTOWE”

**Lip Segmentation with Attention UNet**

Explore and run machine learning code with Kaggle Notebooks | Using data from multiple data sources

Kaggle / Sep 20, 2022

Priyansi/**eyes-lips-segmentation**

PyTorch code for binary segmentation on CelebAMask-HQ dataset via both a UNet written from scratch and a pretrained DeepLabv3 model.

1 Contributor · 0 Issues · 6 Stars · 0 Forks

**Priyansi/eyes-lips-segmentation: PyTorch code for binary segmentation on CelebAMask-HQ dataset via both a UN...**

PyTorch code for binary segmentation on CelebAMask-HQ dataset via both a UNet written from scratch and a pretrained DeepLabv3 model. - Priyansi/eyes-lips-segmentation

GitHub

raahatg21/**Eyes-Lips-Segmentation**

1 Contributor · 1 Issue · 18 Stars · 6 Forks

**raahatg21/Eyes-Lips-Segmentation**

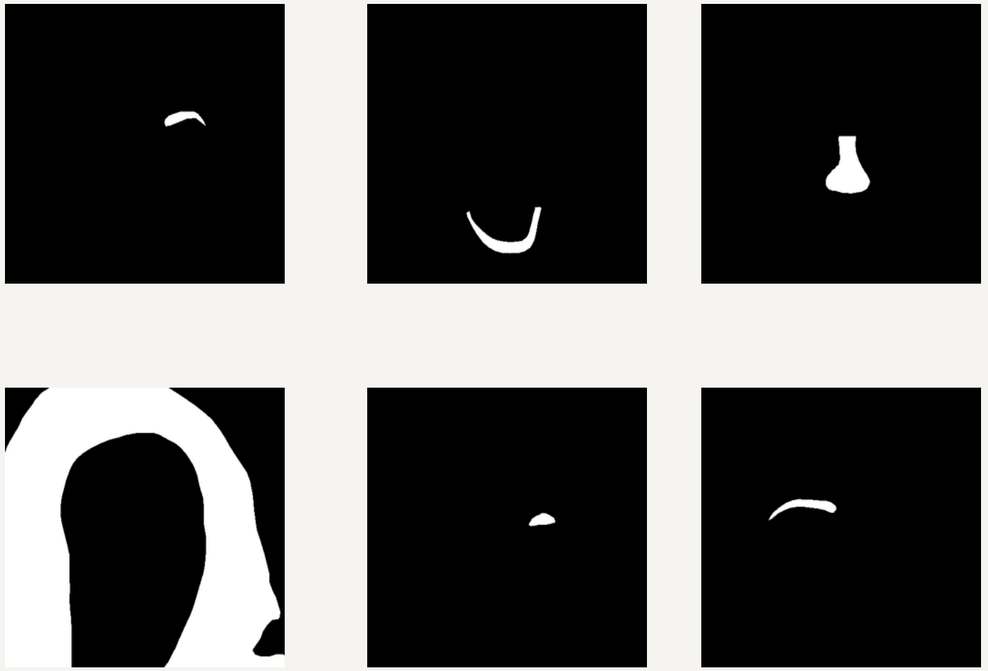
Contribute to raahatg21/Eyes-Lips-Segmentation development by creating an account on GitHub.

GitHub

1

## Zbiór danych

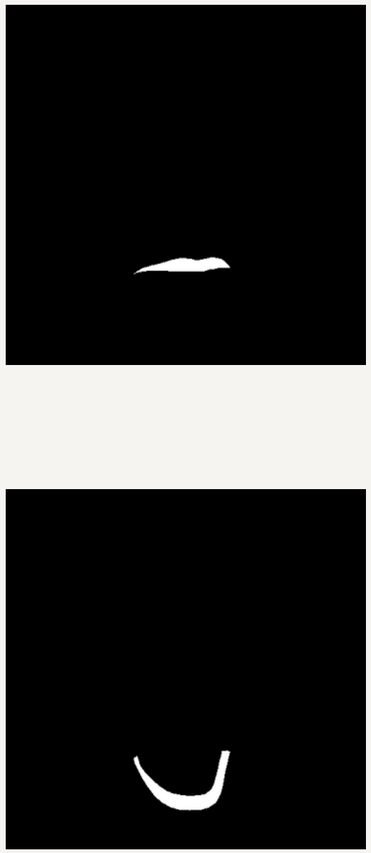
CelebAMask-HQ Dataset

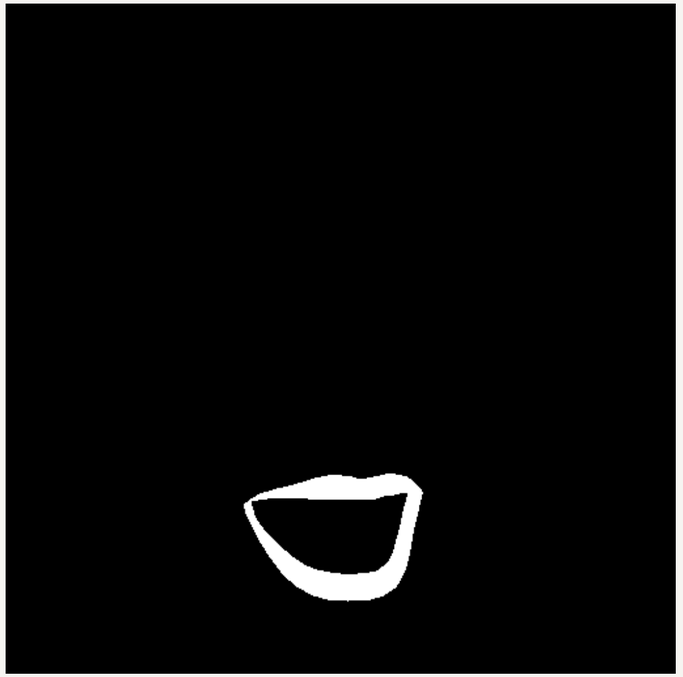


30 000 ⟶ 3000

# 1 Zbiór danych

CelebAMask-HQ Dataset



TRAIN 80%

VAL 10%

TEST 10%

2

# Model

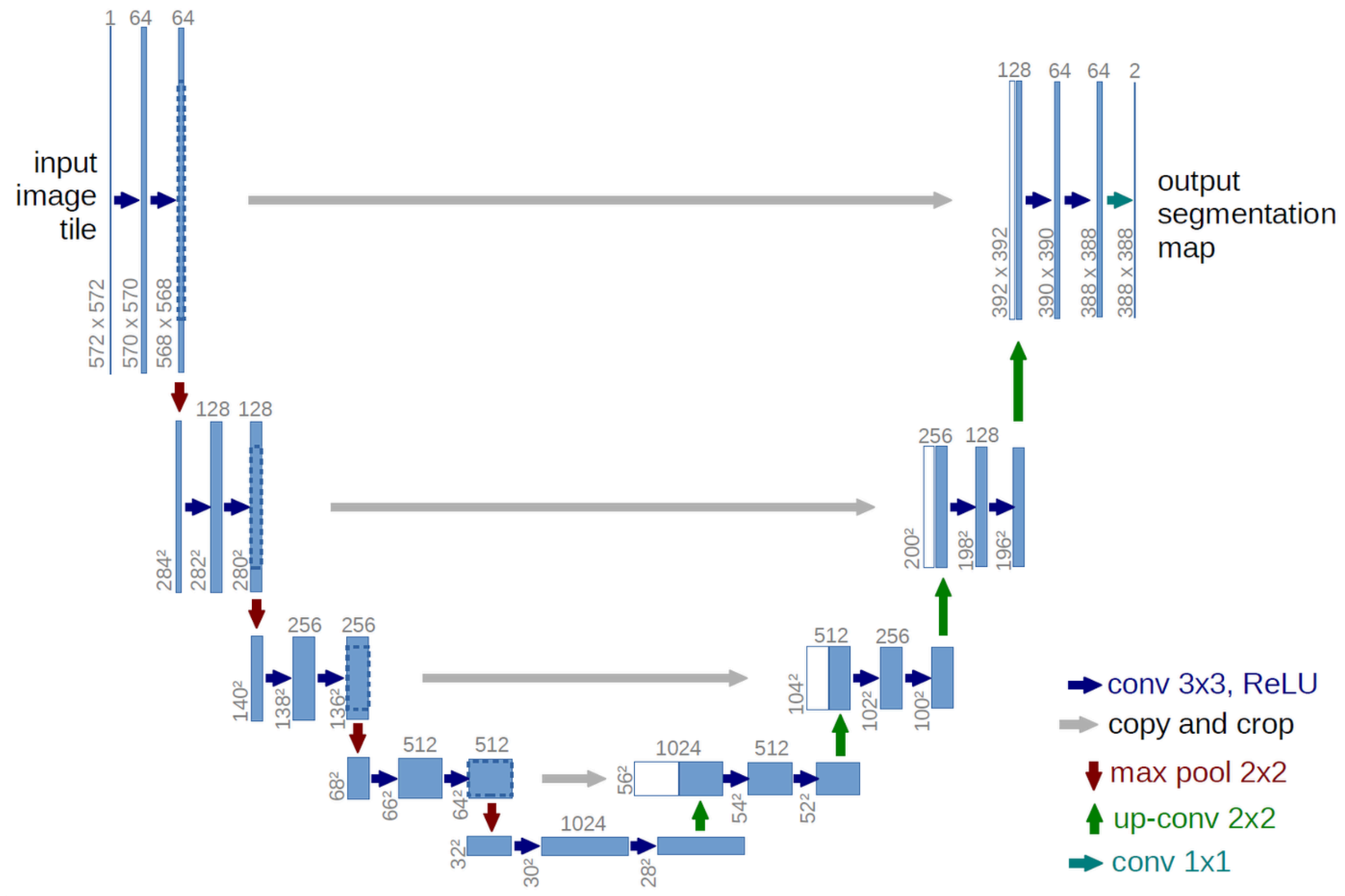
PyTorch

## UNet

segmentation models pytorch + vgg
(Fine Tuning)

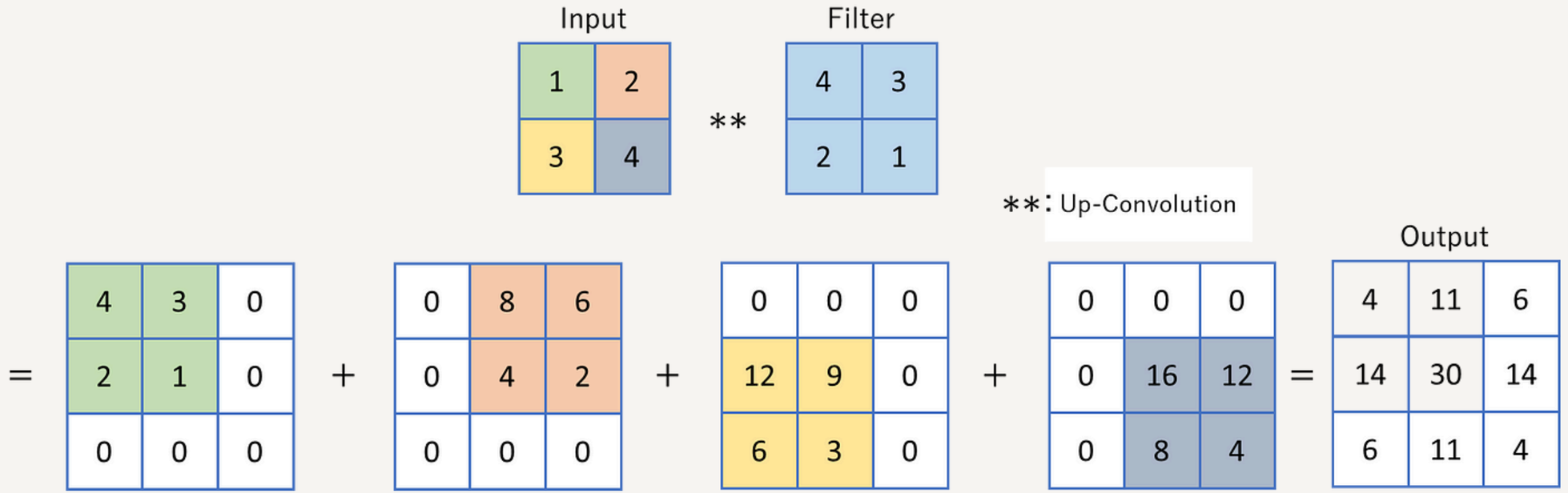
UNet “from scratch”

# UNet

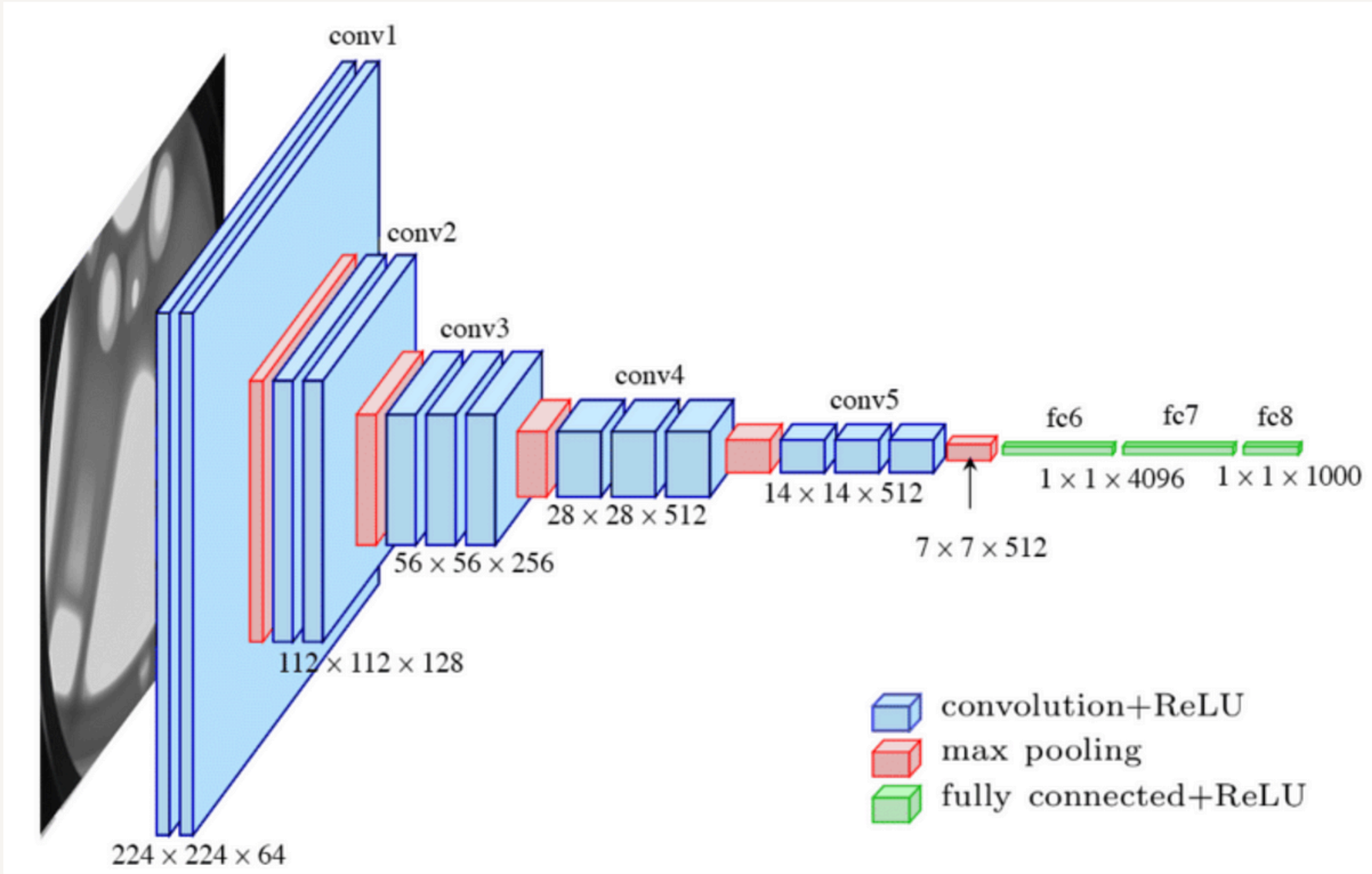



*Źródło: GeeksforGeeks*

# upconv


Input
Filter
1 2
3 4
**
4 3
2 1
**: Up-Convolution
Output
=
4 3 0
2 1 0
0 0 0
+
0 8 6
0 4 2
0 0 0
+
0 0 0
12 9 0
6 3 0
+
0 0 0
0 16 12
0 8 4
=
4 11 6
14 30 14
6 11 4

# VGG

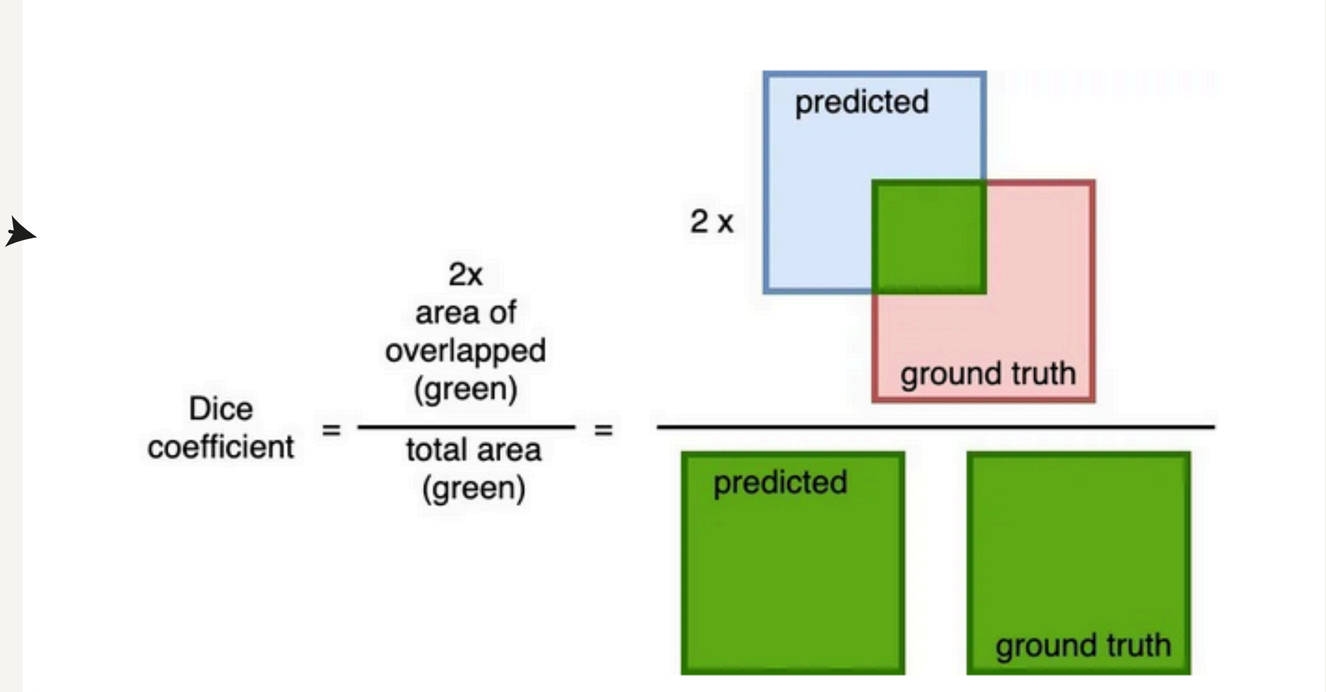



*Źródło: paperswithcode.com*

# Funkcje kosztu

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot \log(\sigma(p_i)) + (1 - y_i) \cdot \log(1 - \sigma(p_i))\right]$$

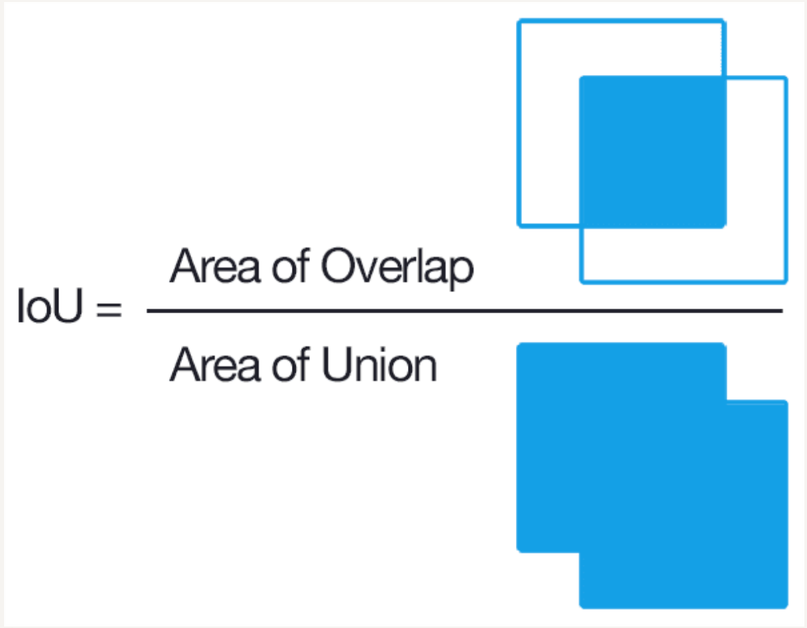
$$\text{DSC} = \frac{2 \cdot \sum_{i=1}^{N} p_i y_i}{\sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} y_i^2}$$




*Źródło: Medium*

- BCE skupia się na pikselach indywidualnie,
- Dice Loss patrzy na całkowite dopasowanie obrazów.

# Testowanie i porównanie




*Źródło: Medium*

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

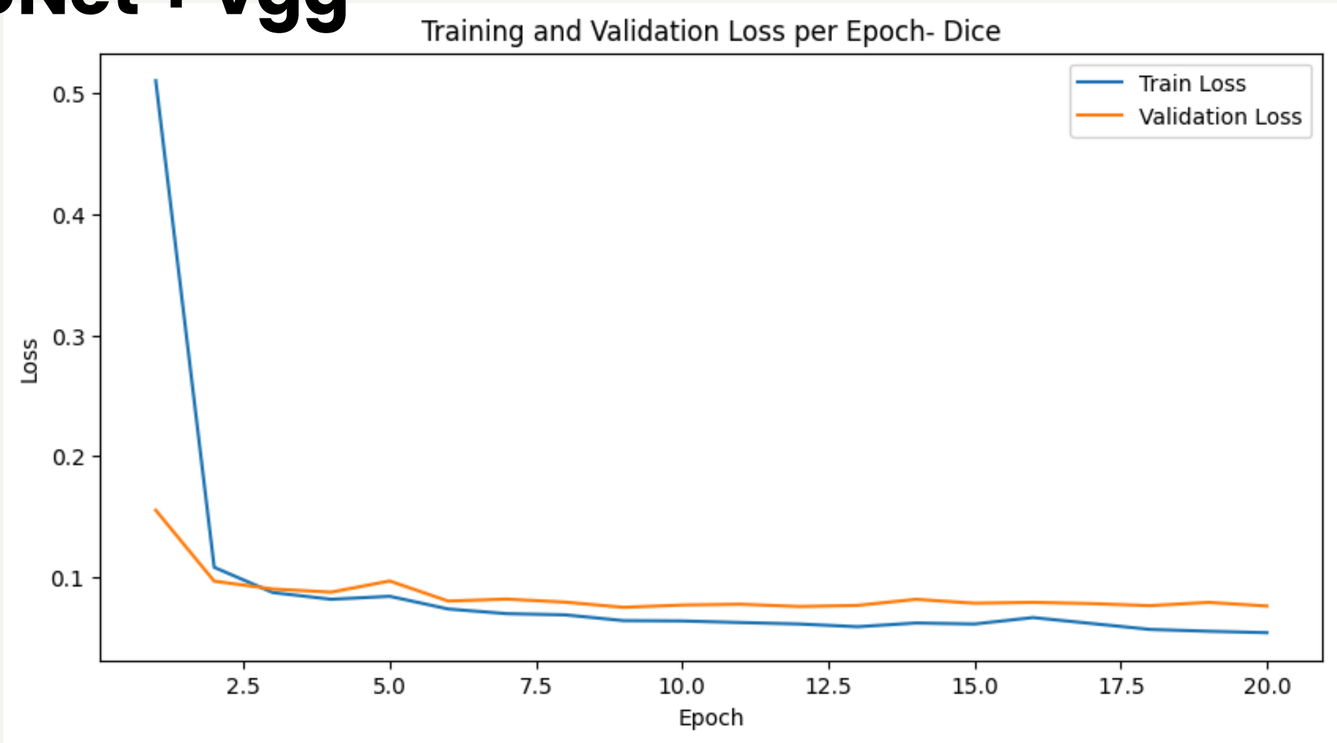
*Źródło: evidentlyai.com*

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

*Źródło: evidentlyai.com*

# Testowanie i porównanie

## UNet + vgg

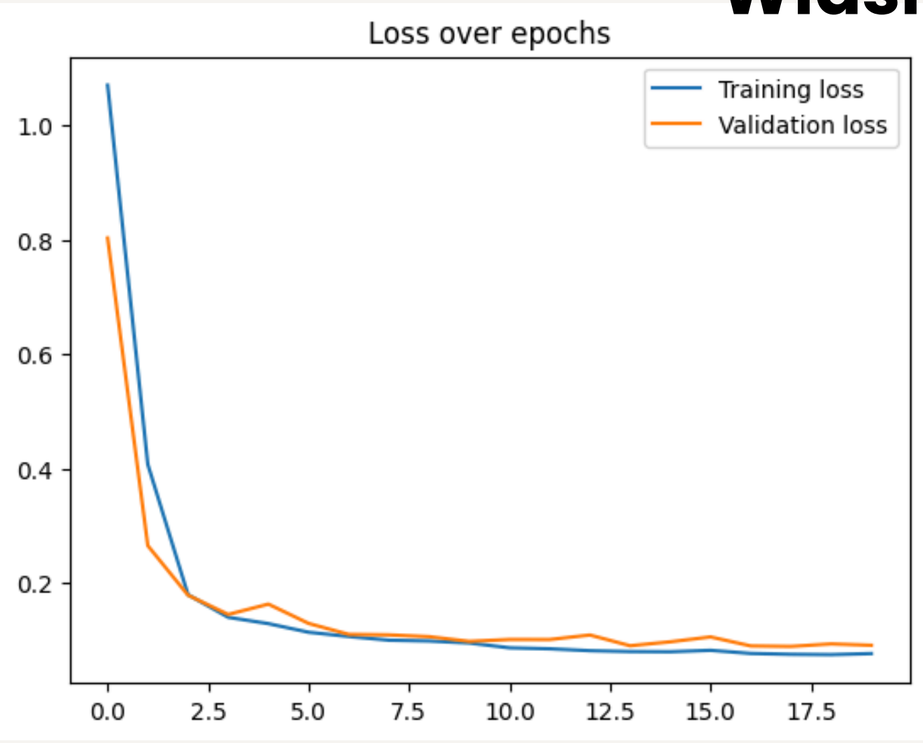



Mean IoU: 0.8596

Mean Recall: 0.9150

Mean Accuracy: 0.9982

## Własny



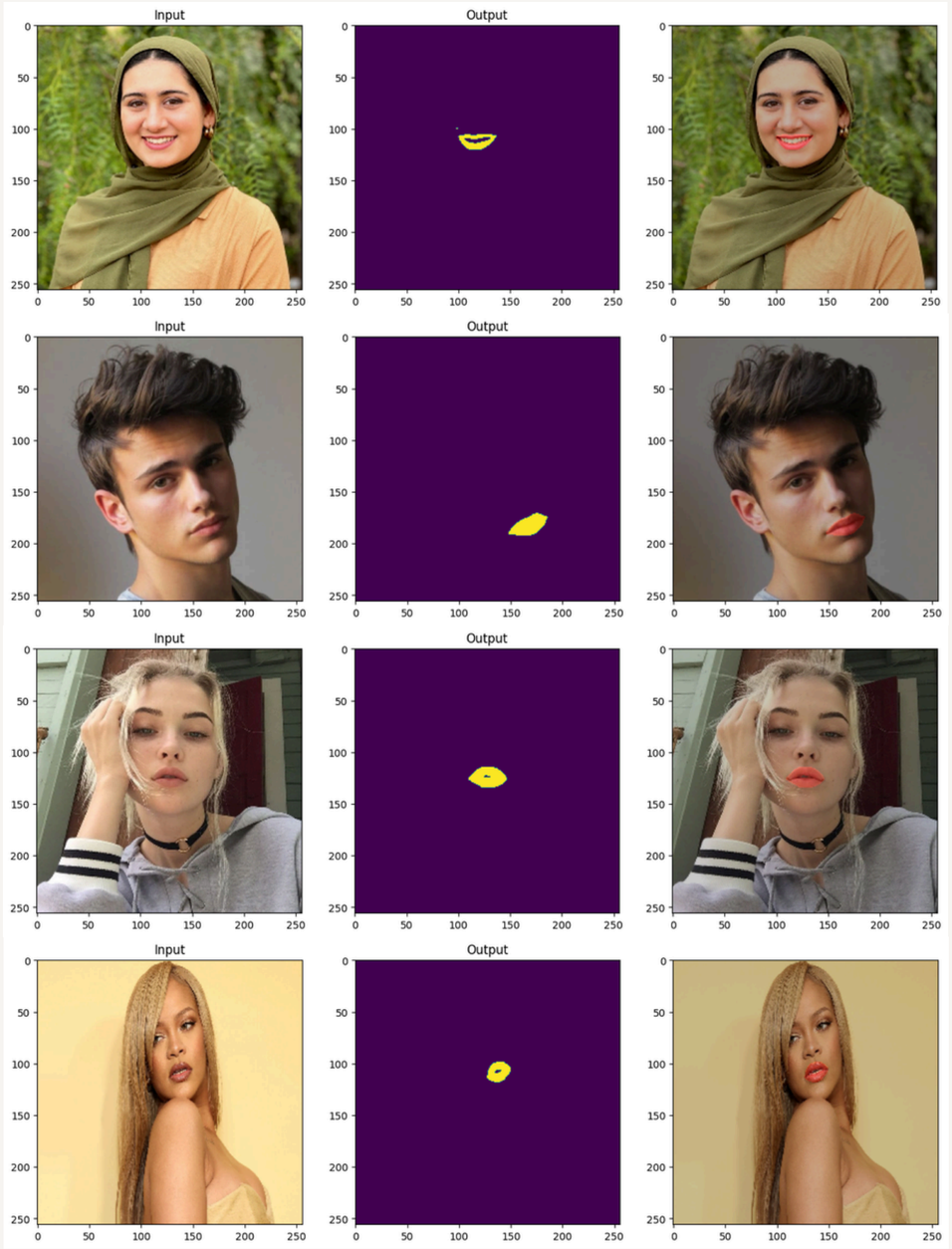

Mean IOU: 0.8221

Mean Recall: 0.8925

Mean Accuracy: 0.9976

# Testowanie i porównanie

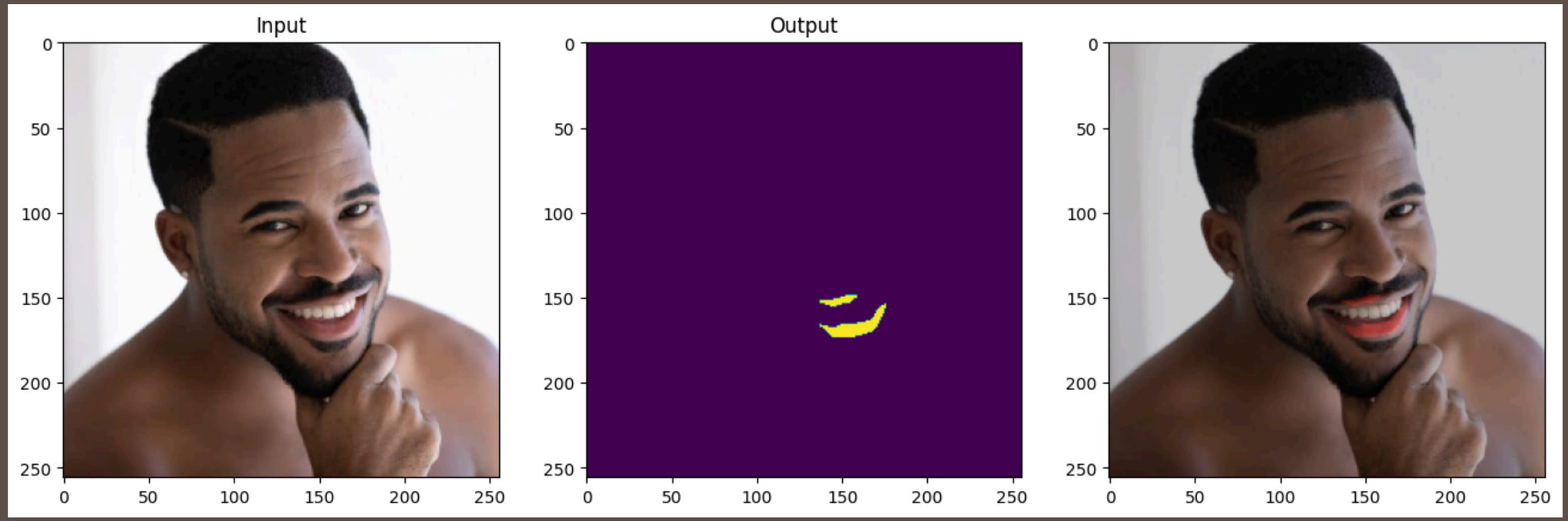
UNet + vgg



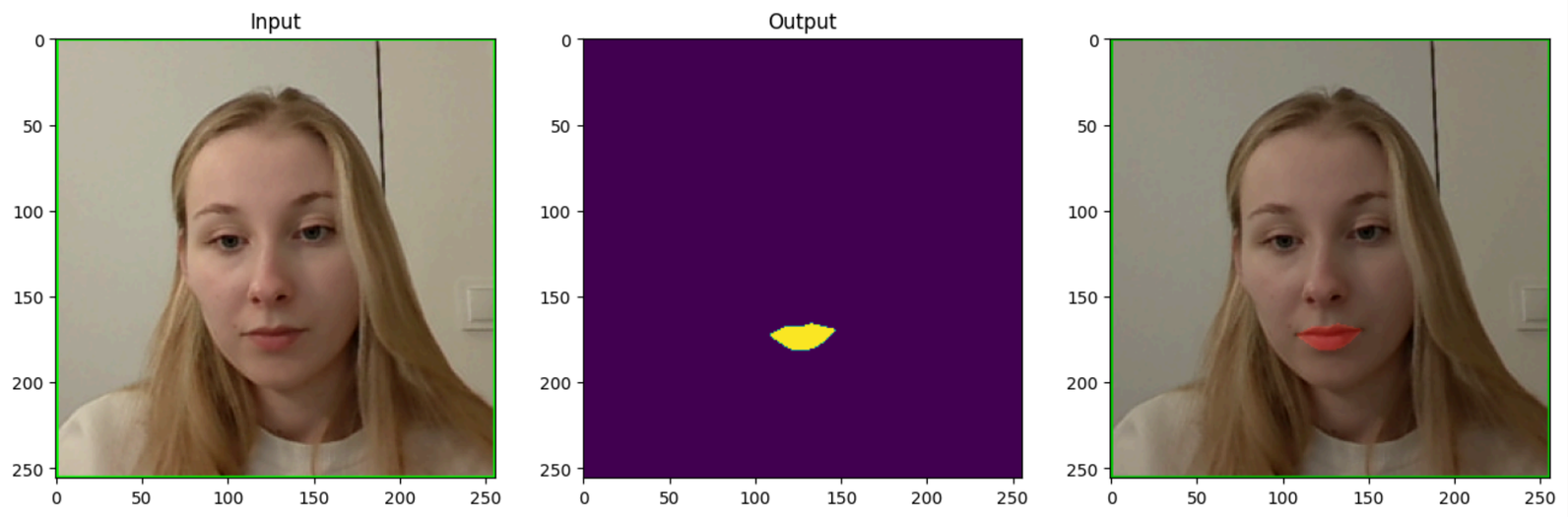


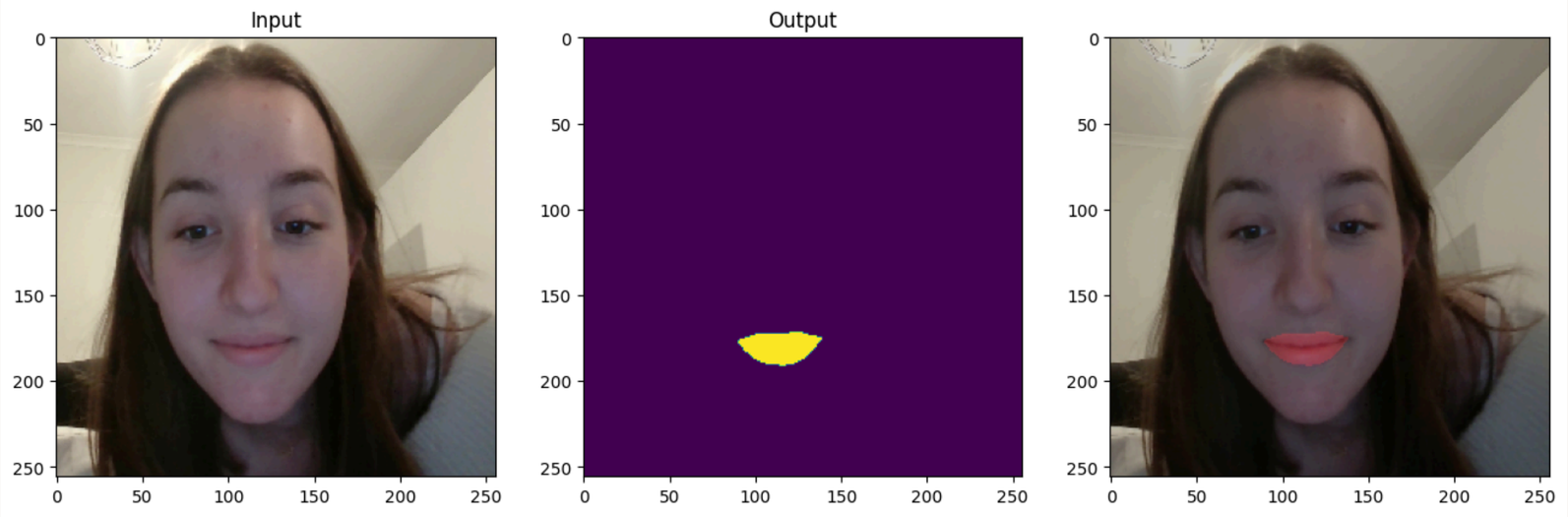
własny UNet

# Testowanie i porównanie



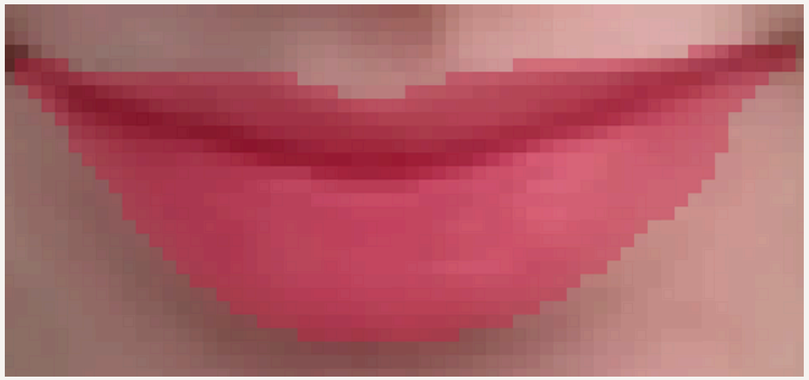
# Testowanie i porównanie



3

# Nakładanie koloru



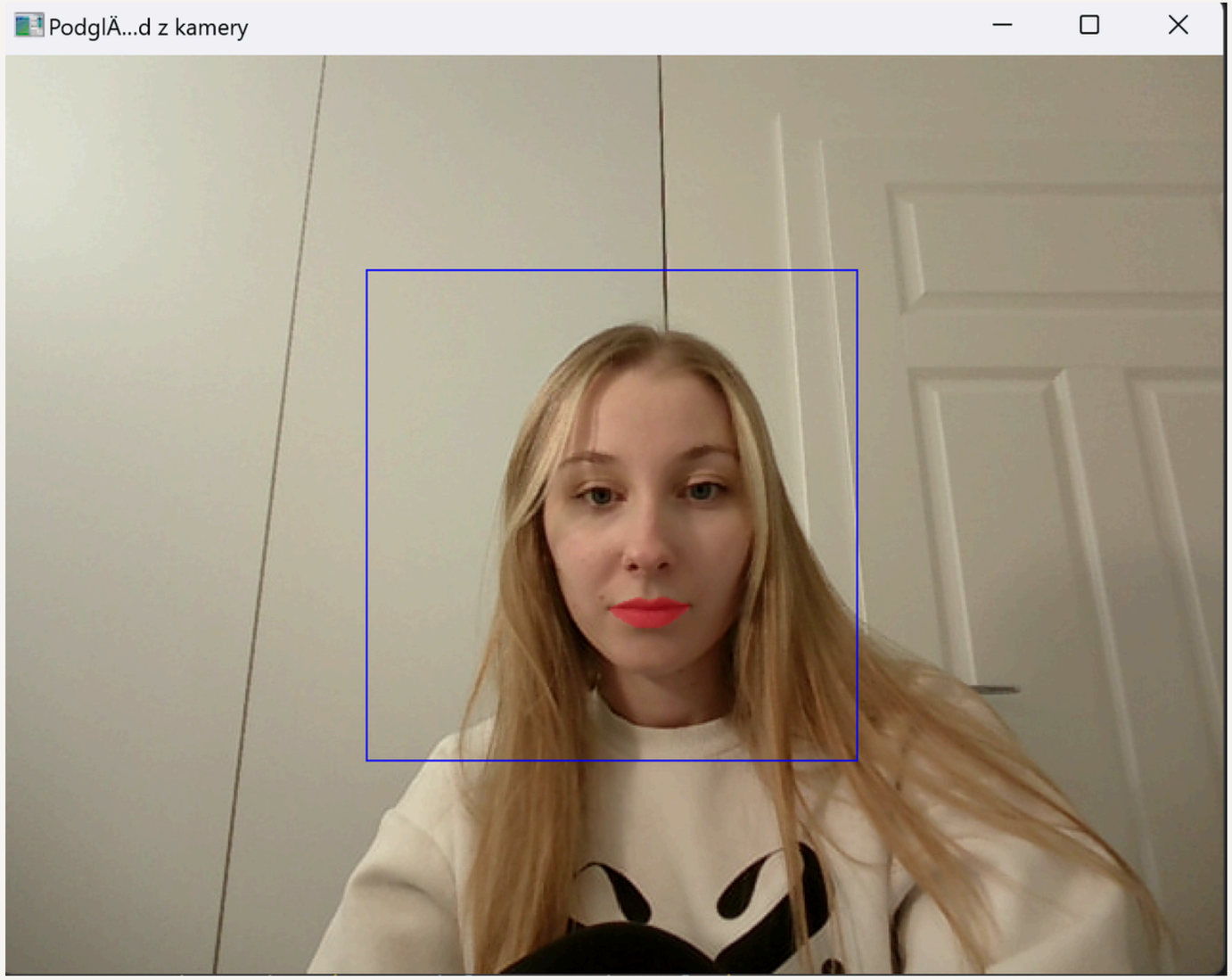
```
def get_mask_on_photo(img, pred, color, alpha=0.3):
    mask = np.zeros_like(img)
    mask[:, :, 0] = pred[:, :, 0] * color[0]    #B
    mask[:, :, 1] = pred[:, :, 0] * color[1]    #G
    mask[:, :, 2] = pred[:, :, 0] * color[2]    #R

    masked_image = img.copy()
    mask_indices = pred[:, :, 0] > 0

    if np.any(mask_indices):
        masked_image[mask_indices] = cv2.addWeighted(
            img[mask_indices].astype(np.float32), 1 - alpha,
            mask[mask_indices].astype(np.float32), alpha, 0
        ).astype(np.uint8)

    return masked_image
```

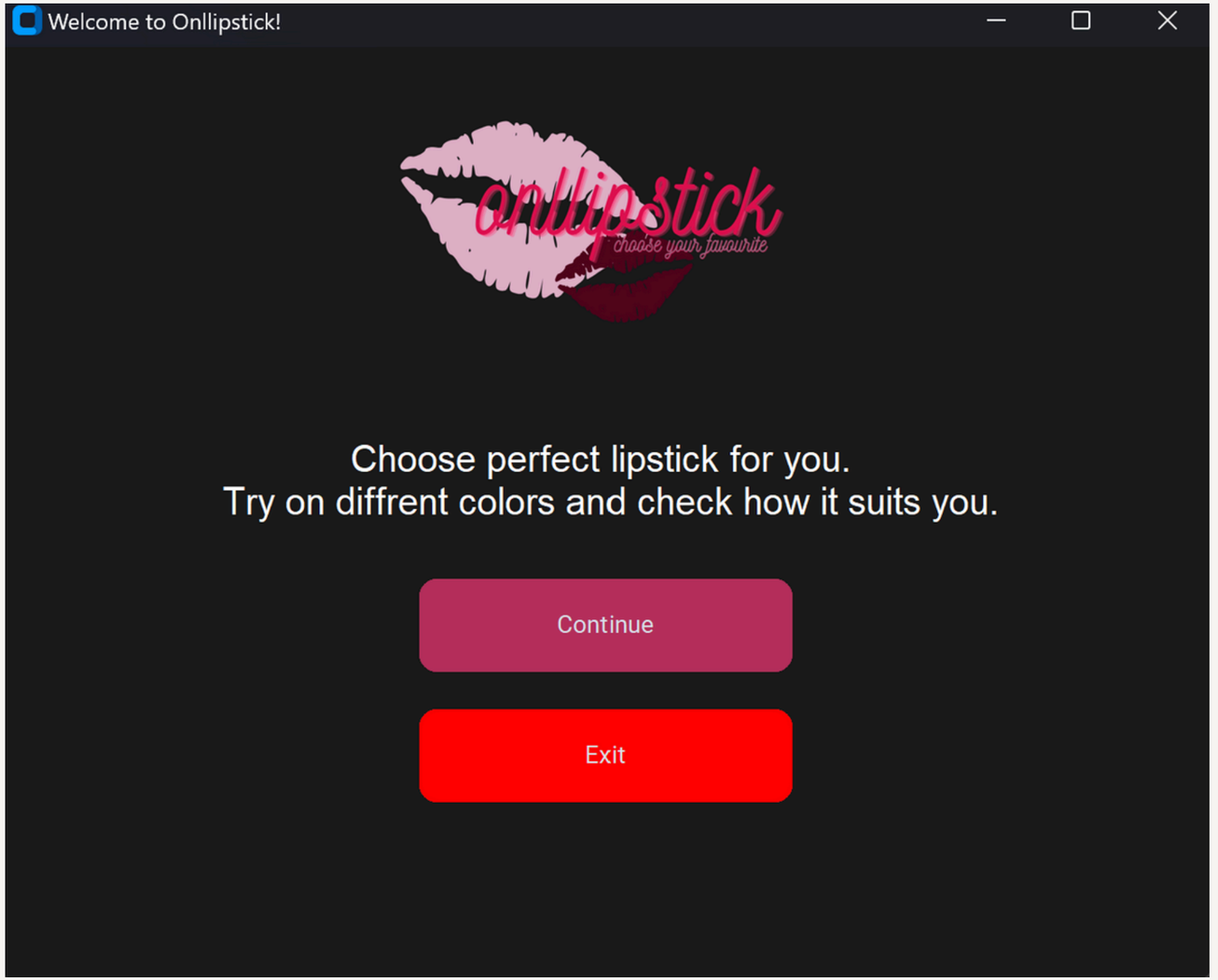
# Kamerka - czas rzeczywisty



4

# Interfejs graficzny

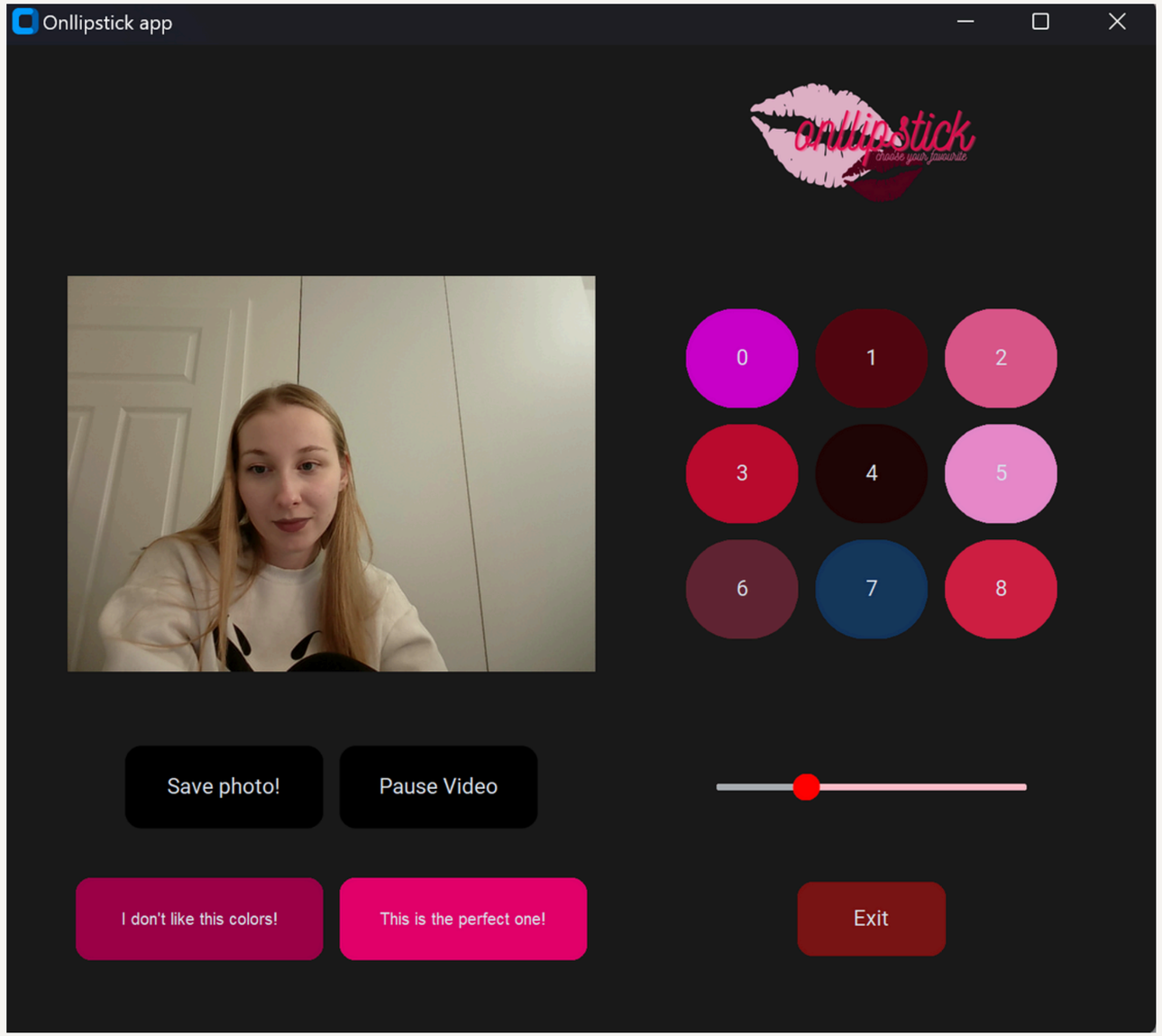
CustomTkinter



4

# Interfejs graficzny

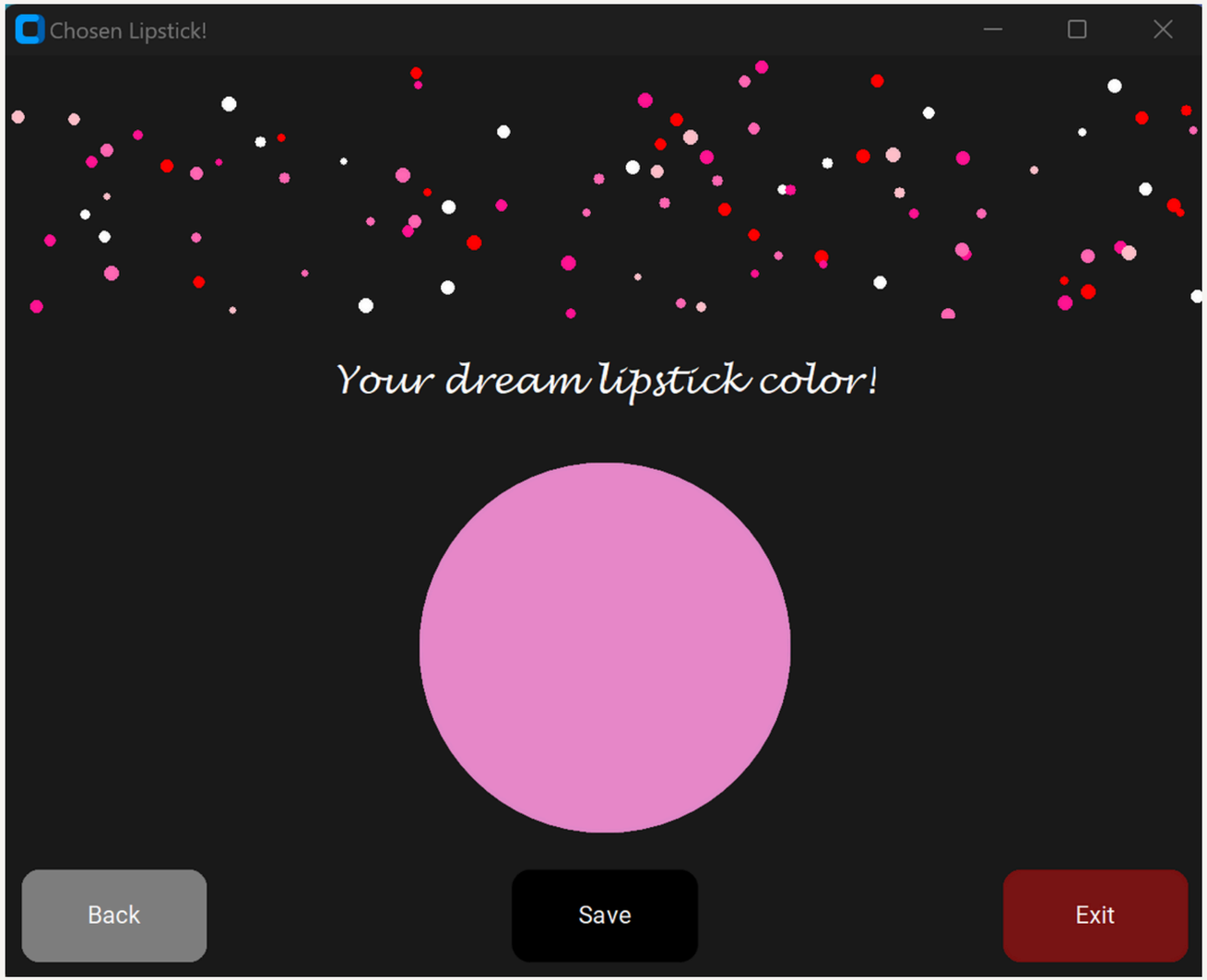
CustomTkinter



4

# Interfejs graficzny

CustomTkinter



# Bibliografia


1. Yu, Y., Wang, C., Fu, Q., Kou, R., Huang, F., Yang, B., Yang, T., & Gao, M. (2023). Techniques and challenges of image segmentation: A review. Electronics, 12(5), 1199. https://doi.org/10.3390/electronics12051199

2. Krithika alias AnbuDevi, M., & Suganthi, K. (2022). Review of semantic segmentation of medical images using modified architectures of UNET. Diagnostics, 12(12), 3064. https://doi.org/10.3390/diagnostics12123064

3. Khan, K., Khan, R. U., Ahmad, K., Ali, F., & Kwak, K. S. (2020). Face segmentation: A journey from classical to deep learning paradigm, approaches, trends, and directions. IEEE Access, 8, 58683–58696.

4. Palomino Cobo, F. (2020, December 10). Mastering U-Net: A step-by-step guide to segmentation from scratch with PyTorch. Medium. medium.com/@fernandopalominocobo/mastering-u-net-a-step-by-step-guide-to-segmentation-from-scratch-with-pytorch-6a17c5916114

5. AssemblyAI. (2022, November 2). How does U-Net work for image segmentation? [Video]. YouTube. Retrieved from https://www.youtube.com/watch?v=GAYJ81M58y8

6. Ito Aramendia, A. (2023, January 10). Decoding the U-Net: A complete guide. Medium. Retrieved from https://medium.com/@alejandro.itoaramendia/decoding-the-u-net-a-complete-guide-810b1c6d56d8

Dziękuję za uwagę!