

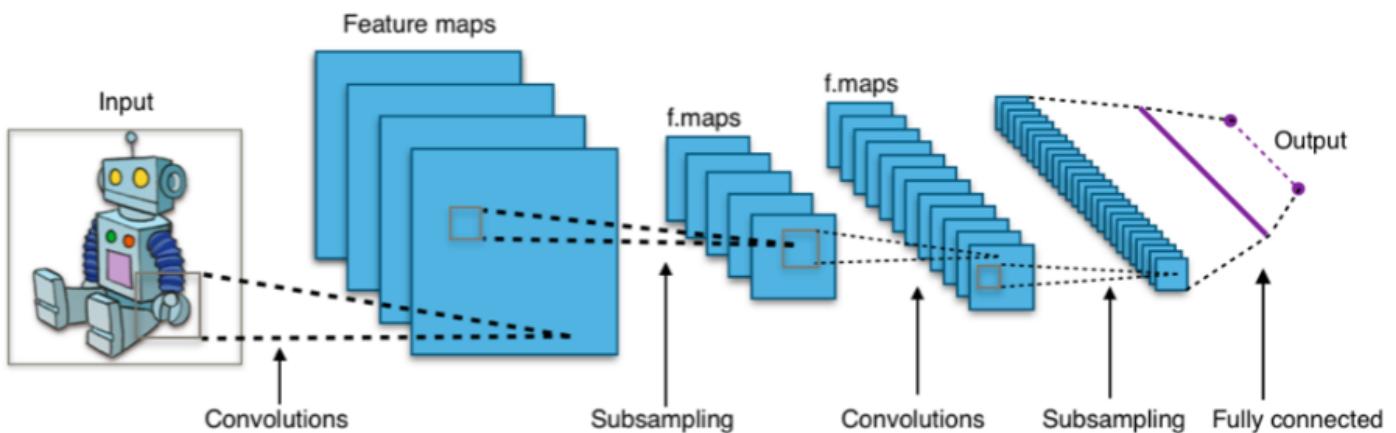
Kacper Puda 188625  
Cyryl Tokarczyk 188624  
Karol Sieniewicz 188623

# Sprawozdanie z projektu z przedmiotu Sztuczna Inteligencja

## Wstęp

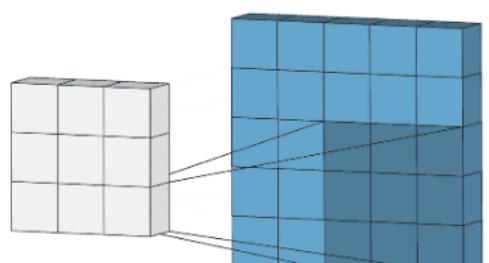
Celem naszego projektu jest utworzenie modelu, który rozpoznaje styl artystyczny danego obrazu wejściowego. Następnie, korzystając z tego modelu, utworzenie modelu transferu stylu artystycznego. Do implementacji modelu skorzystaliśmy z języka **Python** i bibliotek: **tensorflow**, **numpy**, **matplotlib**, **os**, **opencv** oraz **imghdr**.

## Działanie sieci CNN

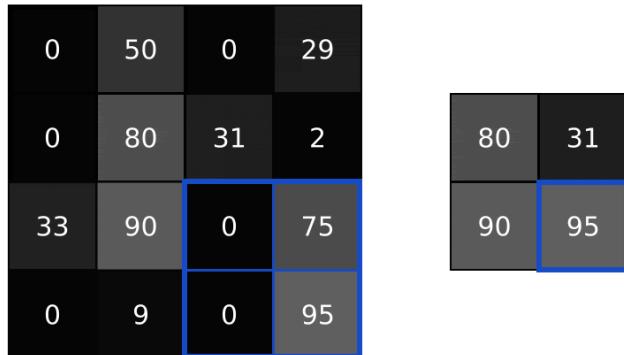


Podstawowym składnikiem sieci CNN są **warstwy konwolucyjne**, które wykonują operację konwolucji na danych wejściowych. Konwolucja polega na przesuwaniu filtru konwolucyjnego (znane również jako jądro) po obrazie, aby wyodrębnić lokalne cechy. Filtrowy ten zawiera wagi, które są dostosowywane w procesie uczenia się sieci.

Ważną cechą sieci CNN jest zdolność do **automatycznego ekstrahowania** istotnych cech z danych wejściowych. Warstwy konwolucyjne wykorzystują różne filtry, aby wykrywać różne **wzorce**, takie jak **krawędzie**, **tekstury** czy **kształty**.

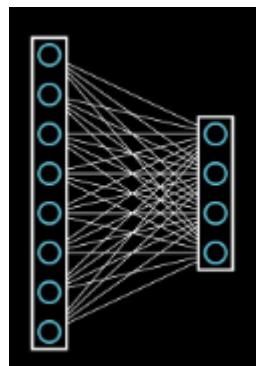


Po warstwach konwolucyjnych sieci CNN często zawierają **warstwy łączące (pooling)**, które zmniejszają rozmiar mapy cech i pomagają w redukcji liczby parametrów do uczenia. Warstwy łączące łączą sąsiednie wartości cech i zwracają najważniejsze wartości, bądź średnią wartość. W ten sposób sieci CNN stają się bardziej odporne na przesunięcia i zmiany w danych wejściowych.



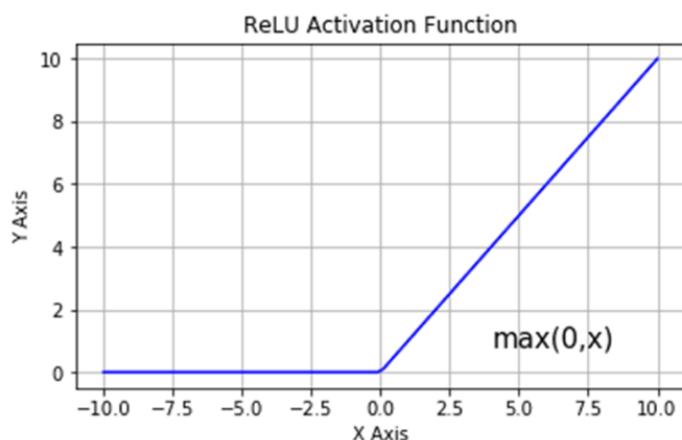
Po przeprowadzeniu operacji konwolucji i łączenia, otrzymujemy tensorowe mapy cech. Warstwa **Flatten** przekształca te mapy cech w tensorze na jednowymiarowy wektor. Dzięki temu, dane są "spłaszczone" i mogą być przekazane do warstw w pełni połączonych. Warstwa Flatten umożliwia pełne połączenie każdej cechy z wagami w pełni połączonymi warstwami, co jest niezbędne do dalszego przetwarzania danych i uzyskania ostatecznych predykcji. Najczęściej używamy funkcji aktywacji **softmax** do klasyfikowania tych cech, co wymaga jednowymiarowych danych wejściowych. Dlatego konieczna jest **spłaszczona warstwa**.

Następnie, po warstwach konwolucyjnych i łączących, zazwyczaj występują **warstwy w pełni połączone (fully connected)**, podobne do tradycyjnych sieci neuronowych. Warstwy te służą do przetwarzania cech wyekstrahowanych przez warstwy konwolucyjne i podejmowania decyzji na podstawie tych cech. Ostatnia warstwa w pełni połączona jest zwykle warstwą wyjściową, która generuje odpowiedzi sieci, na przykład przewidziane etykiety klasy dla danych wejściowych

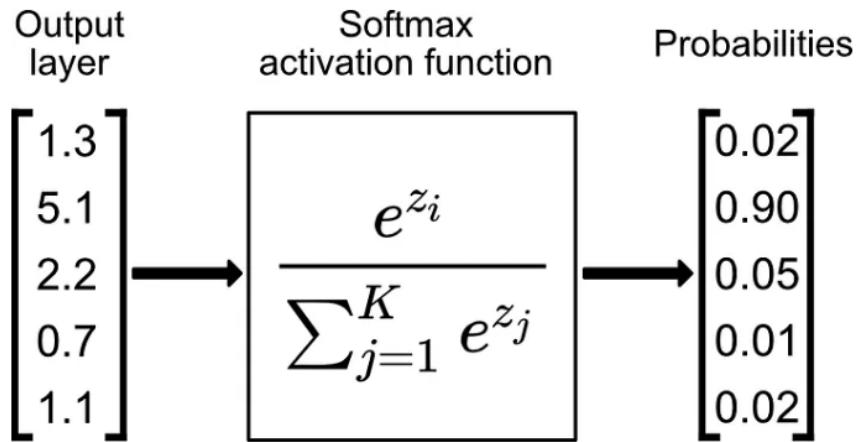


## Funkcje aktywacji

**ReLU (Rectified Linear Activation)** - stosuje do modelu potrzebną nieliniowość. Dodatkowo została opracowana nie przez matematyków, a informatyków. Dzięki temu jest naprawdę szybka w trenowaniu (bo implementacja jest prosta).



**Softmax** - służy kluczowemu celowi: upewnieniu się, że wyniki **CNN** sumują się do 1. Z tego powodu operacje Softmax są przydatne do skalowania wyników modelu na prawdopodobieństwa.



## Nasza implementacja

W naszym modelu rozpoznawanie ograniczyliśmy do 5 różnych stylów: **ekspresjonizm, styl japoński, realizm, rokoko i symbolizm**.

Nasz model **sieci CNN** składa się z:

1. Warstwa z 32 filtrami 3x3, przyjmująca zdjęcia rgb 200 x 200i, funkcja aktywacji - relu
2. Max pooling
3. Dropout 15% neuronów
4. Warstwa z 64 filtrami 3x3i, funkcja aktywacji - relu
5. Max pooling
6. Dropout 15% neuronów
7. Warstwa z 32 filtrami 3x3i, funkcja aktywacji - relu
8. Max pooling
9. Dropout 15% neuronów
10. Warstwa gęsta z 128 neuronami, funkcja aktywacji - relu
11. Warstwa gęsta z 5 neuronami na wyjście dla każdego ze stylów, funkcja aktywacji - softmax

Prezentacja modelu z biblioteki **tensorflow**:

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 32)	896
max_pooling2d (MaxPooling2D)	(None, 99, 99, 32)	0
dropout (Dropout)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 64)	0
dropout_1 (Dropout)	(None, 48, 48, 64)	0
conv2d_2 (Conv2D)	(None, 46, 46, 32)	18464
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 32)	0
dropout_2 (Dropout)	(None, 23, 23, 32)	0
flatten (Flatten)	(None, 16928)	0
dense (Dense)	(None, 128)	2166912
dense_1 (Dense)	(None, 5)	645
<hr/>		
Total params: 2,205,413		
Trainable params: 2,205,413		
Non-trainable params: 0		
<hr/>		

Do uczenia wykorzystaliśmy część danych z **bazy danych**:

<https://www.kaggle.com/datasets/sivarazadi/wikiart-art-movementsstyles>.

Dla każdego stylu wykorzystaliśmy około **800 obrazów**:

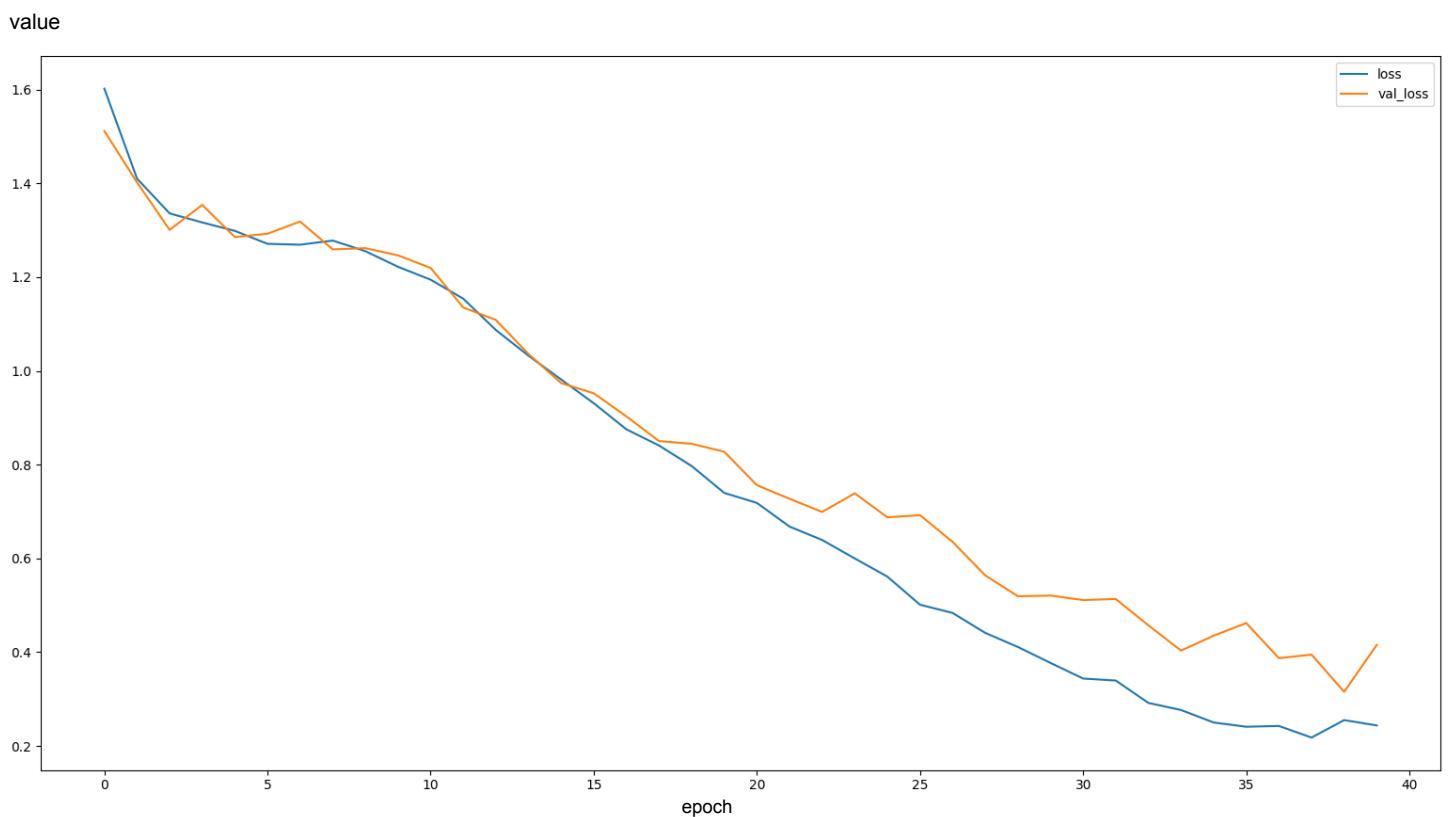
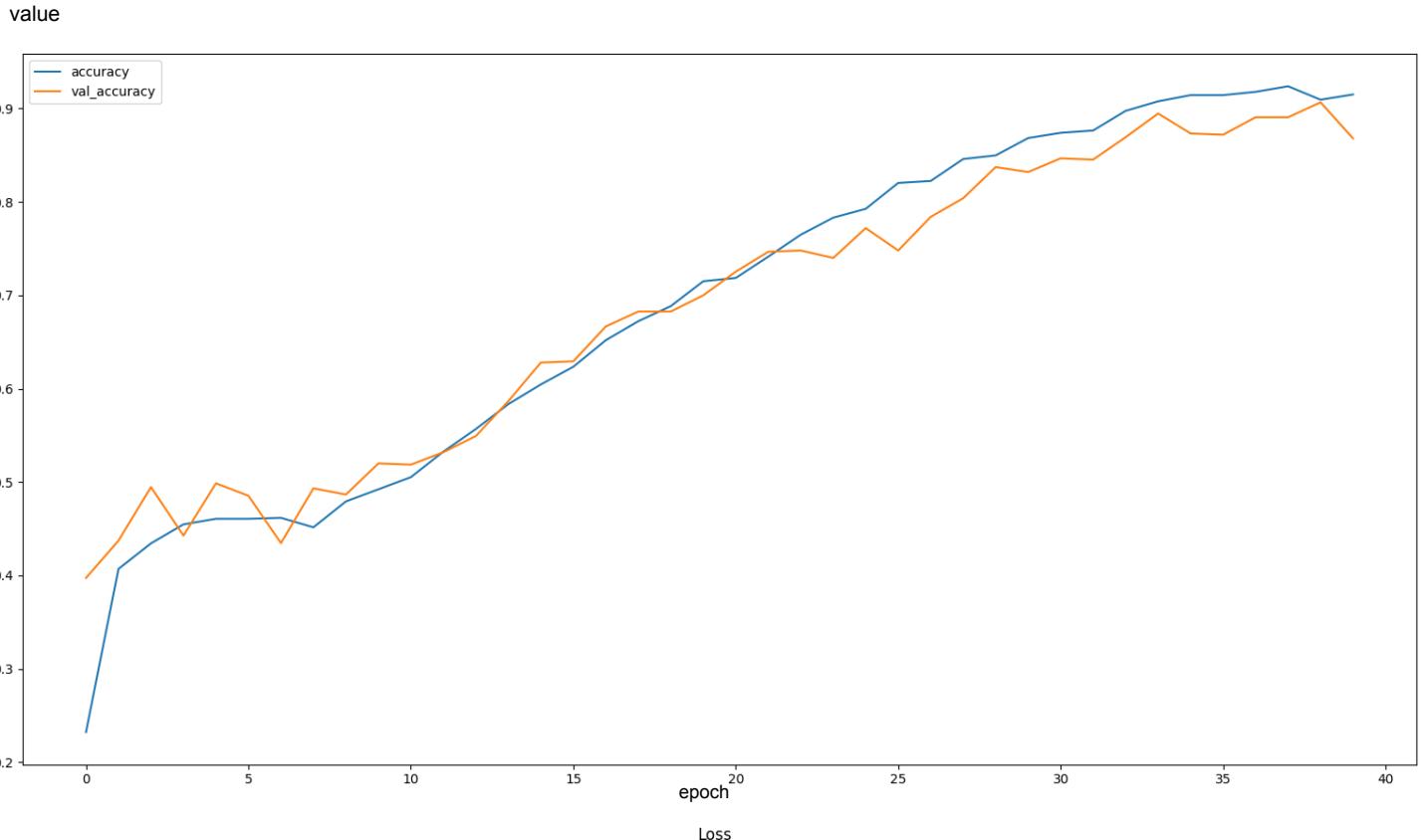
Found 4075 files belonging to 5 classes.

Wszystkie dane podzieliliśmy na **3 zbiory**:

- 70% - **dane treningowe** - wykorzystywane w celu uczenia modelu,
- 20% - **dane walidacyjne** - wykorzystywane w celu walidacji w trakcie uczenia modelu,
- 10% - **dane testowe** - wykorzystywane w celu ostatecznej walidacji po fazie uczenia

# Wyniki

Wykresy uczenia się modelu (**40 epok po 150 obrazów w 1 zbiorze (batch)**):  
Accuracy



Wyniki ostatecznej ewaluacji modelu po fazie uczenia:

```
2/2 [=====] - 33s 663ms/step - loss: 0.3161 - accuracy: 0.8933
```

Po wyuczeniu naszej sieci możemy przeprowadzić testy naszej sieci na przykładowym obrazie (styl realizm):



LA BRETONNERIE IN THE DEPARTMENT OF INDRE, Gustave Courbet, 1856

**Predykcja** dla obrazu:

```
model = load_model("Cypuka3.h5")
img = procesImage(cv2.imread('newData//realism3.jpg'))
prediction = model.predict(img)[0]
prob = [round(x, 3) for x in prediction]
print(prob)
```

**Wynik** programu:

```
1/1 [=====] - 0s 328ms/step
[0.0, 0.186, 0.813, 0.0, 0.0]
```

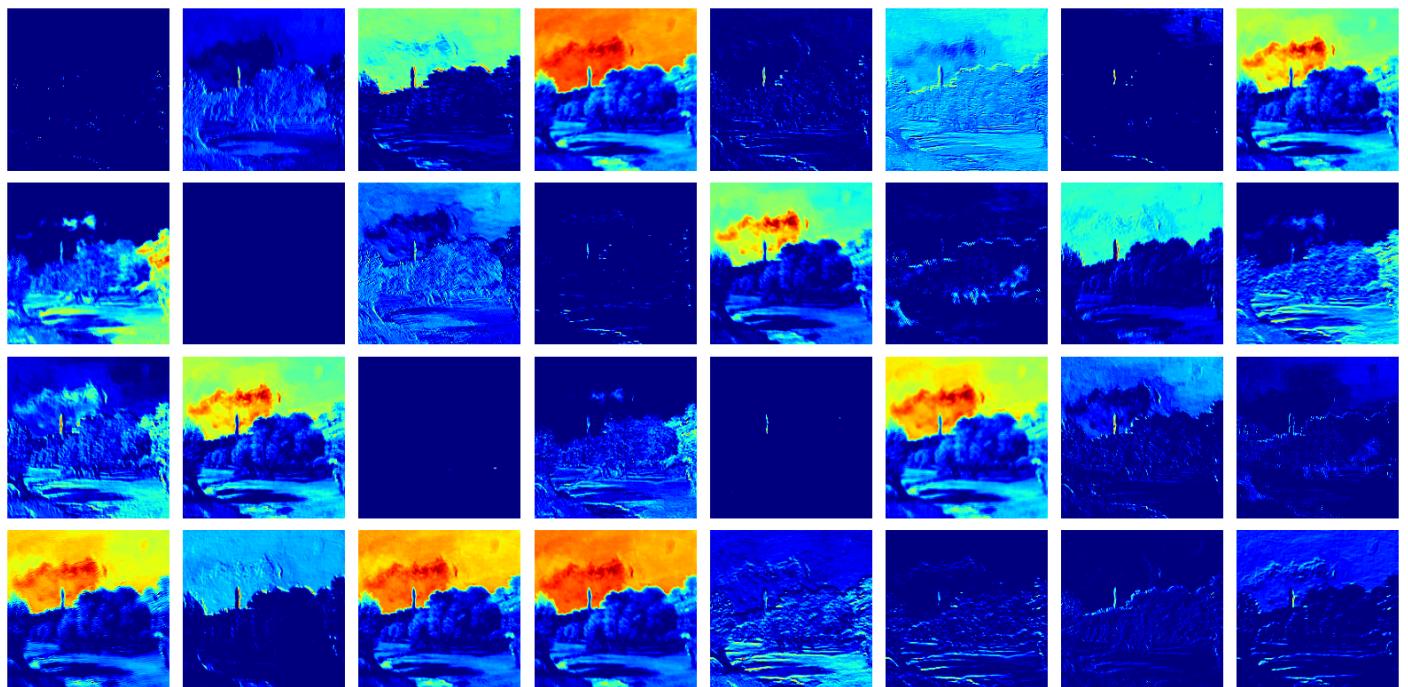
Jak widać model wyliczył największe prawdopodobieństwo na wyjściu o **indeksie 2**, porównując ten indeks z indeksami naszych stylów:

```
['Expressionism', 'Japanese_Art', 'Realism', 'Rococo', 'Symbolism']
```

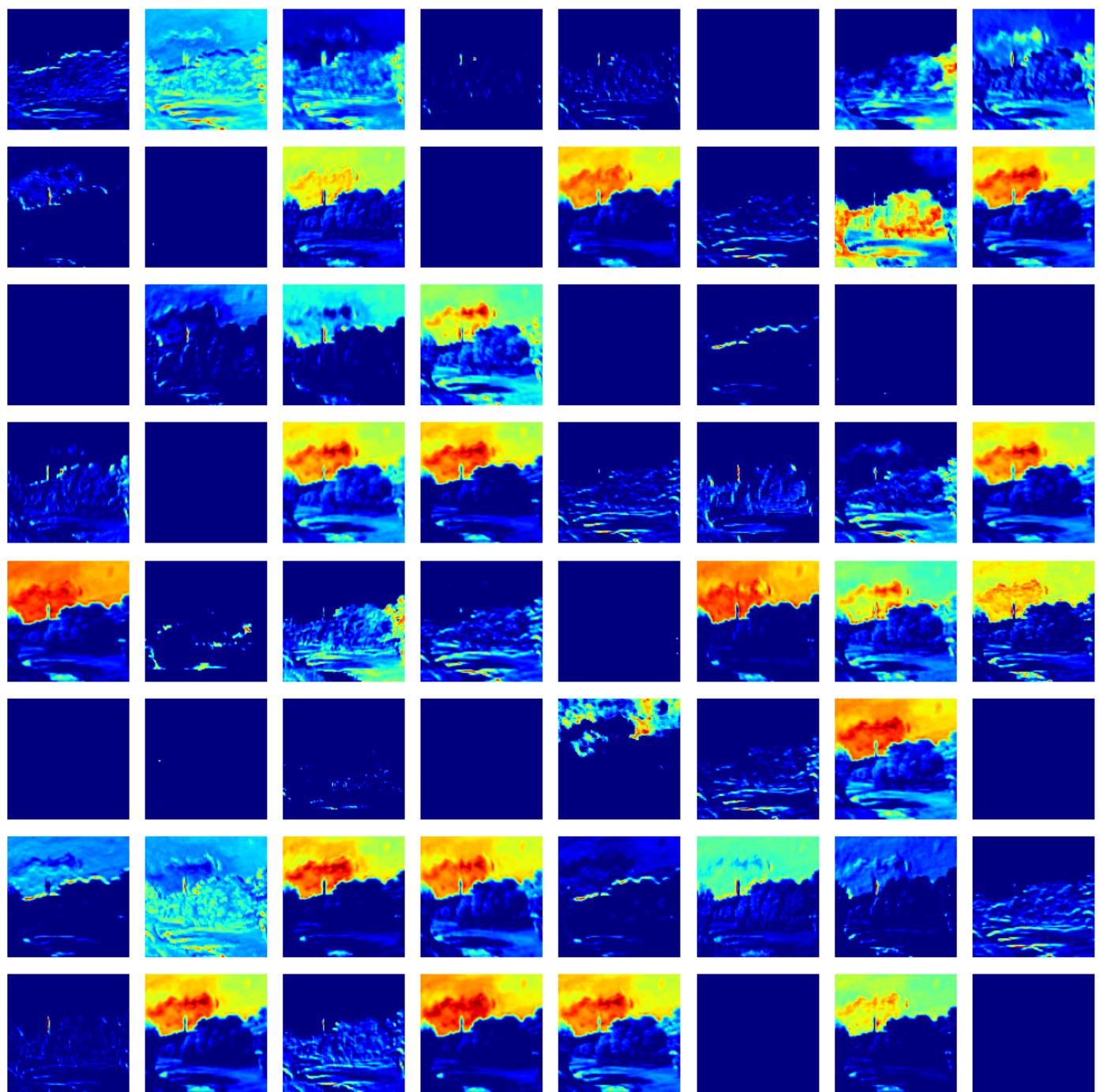
Możemy zauważyć, że został przewidziany styl '**Realism**' co zgadza się z naszym obrazem wejściowym. Widać więc, że model dokonał **poprawnej predykcji**.

Korzystając z naszego modelu możemy wyświetlić **mapy cech** poszczególnych **warstw konwolucyjnych**:

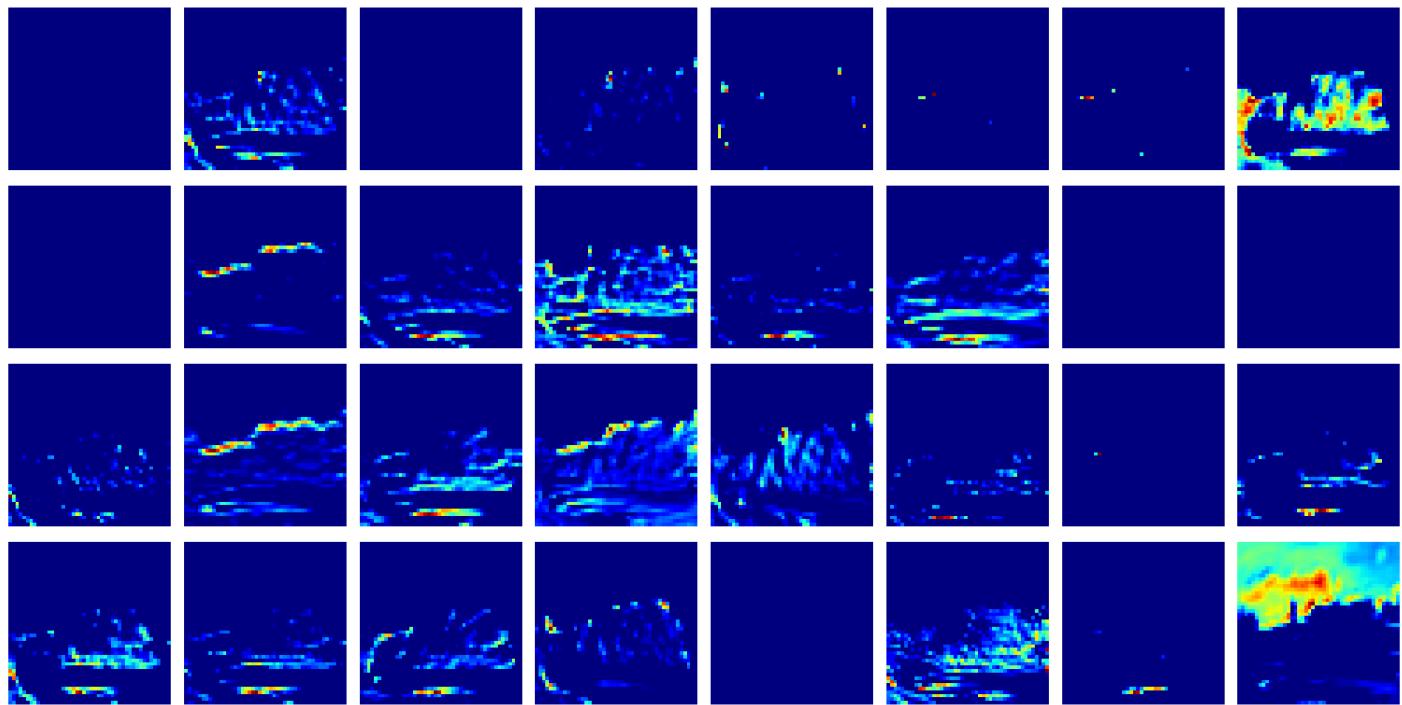
**Konwolucyjna warstwa 1 ('conv2d'):**



**Konwolucyjna warstwa 2 ('conv2d\_1'):**



**Konwolucyjna warstwa 3 ('conv2d\_2'):**



Można zauważyc, że w pierwszych dwóch warstwach model wykrywa cechy **bardziej ogólne**, w szczególności niebo i chmury **w tle**. W trzeciej warstwie skupia się bardziej na **szczegółach**, widać wykrywane **krawędzie** np. linia drzew. Należy pamiętać, że ludzka interpretacja daje tylko **złudzenie zrozumienia**, jakie cechy obrazu wykrywa sieć.

## Guided Grad CAM

(**Guided Gradient Class Activation Mapping**) to technika interpretacji **modeli konwolucyjnych (CNN)**, która pomaga zrozumieć, jakie obszary obrazu są istotne dla decyzji podjętych przez model. Jest to rozszerzenie dwóch innych technik: **Grad CAM** i **Guided Backpropagation**.

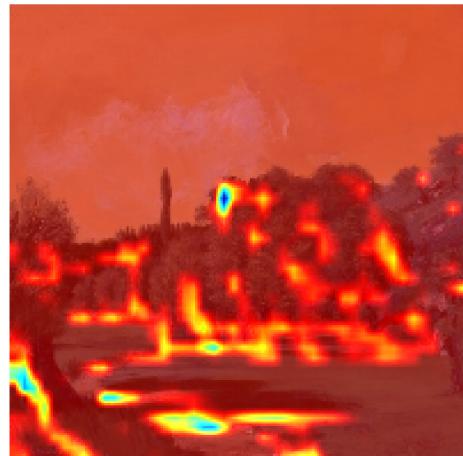
W przypadku modeli CNN, Grad CAM pozwala na wygenerowanie **mapy aktywacji klasowej**, która wskazuje, które obszary obrazu są **istotne dla klasyfikacji**. Ta mapa jest tworzona poprzez **propagację gradientu wstecz** związanych z daną klasą do **warstw konwolucyjnych modelu**. Wynikowy gradient jest **zagregowany i skalowany** w celu wygenerowania mapy aktywacji klasowej, która podkreśla **istotne cechy w obrazie**.

Jednak w przypadku standardowego Grad CAM, propagacja gradientu może prowadzić do **utraty informacji** o lokalizacji w warstwach **ReLU (Rectified Linear Unit)**, które są powszechnie stosowane w modelach CNN. Guided Backpropagation jest techniką, która pozwala na **zachowanie informacji o lokalizacji**, ale **nie generuje mapy aktywacji**.

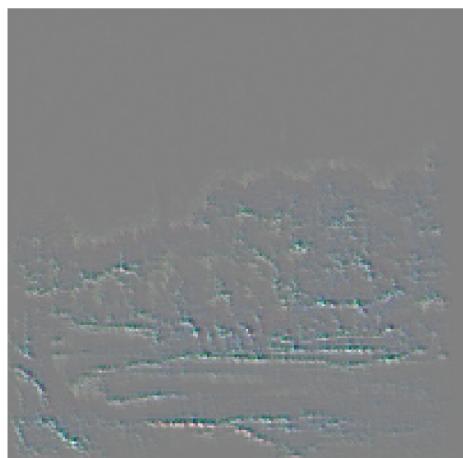
Guided Grad CAM łączy te dwie techniki, łącząc **mapę aktywacji** z Grad CAM z **informacjami o lokalizacji** z Guided Backpropagation. Działa to poprzez **mnożenie element-wise mapy** Grad CAM z mapą wynikową Guided Backpropagation. Ten proces pozwala na utrzymanie informacji o lokalizacji, jednocześnie podkreślając **istotne obszary na mapie aktywacji**.

W rezultacie Guided Grad CAM dostarcza **bardziej szczegółowej interpretacji modelu CNN**, pokazując, które cechy w obrazie przyczyniają się do **określonej klasyfikacji**. To narzędzie może być używane do weryfikacji, czy model koncentruje się na odpowiednich obszarach obrazu i pomaga lepiej zrozumieć, dlaczego dany model **podejmuje określone decyzje**.

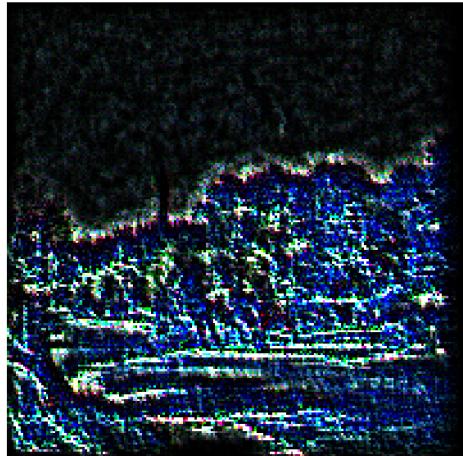
Grad CAM dla naszego zdjęcia:



Guided Backpropagation dla naszego zdjęcia:



Guided Grad CAM dla naszego zdjęcia:



Pokazane wyżej **Guided Grad CAM** możemy próbować interpretować następująco: wizualizacja podkreśla krajobraz, poszarpane kontury roślin, który nasz model może kojarzyć ze stylem realistycznym.

## Wnioski

Guided Grad CAM jest **klasą dyskryminacyjną**, lokalizuje odpowiednie **regiony obrazu**, jednocześnie podkreślając **drobnoziarniste szczegóły pikseli**. Wizualizacja za pomocą Guided Grad CAM pomaga wyjaśnić, na co patrzy model CNN, aby dokonać prognozy, wyjaśniając w ten sposób, dlaczego model zinterpretował to, co zinterpretował.

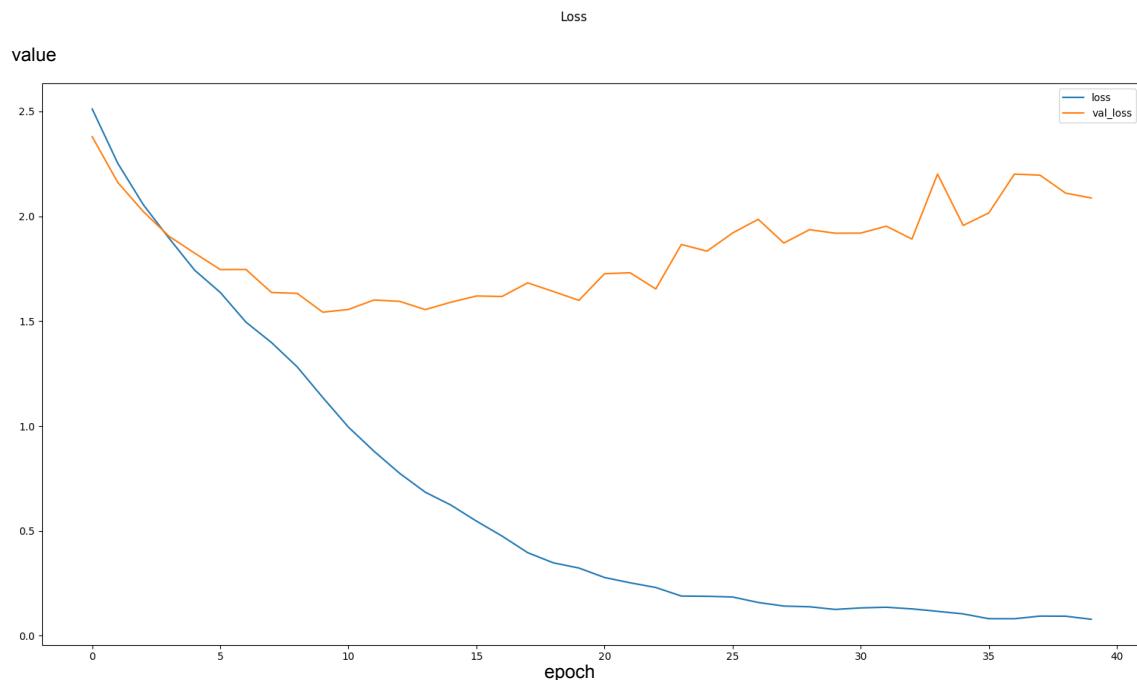
## Drugi trening modelu – dla trzynastu stylów

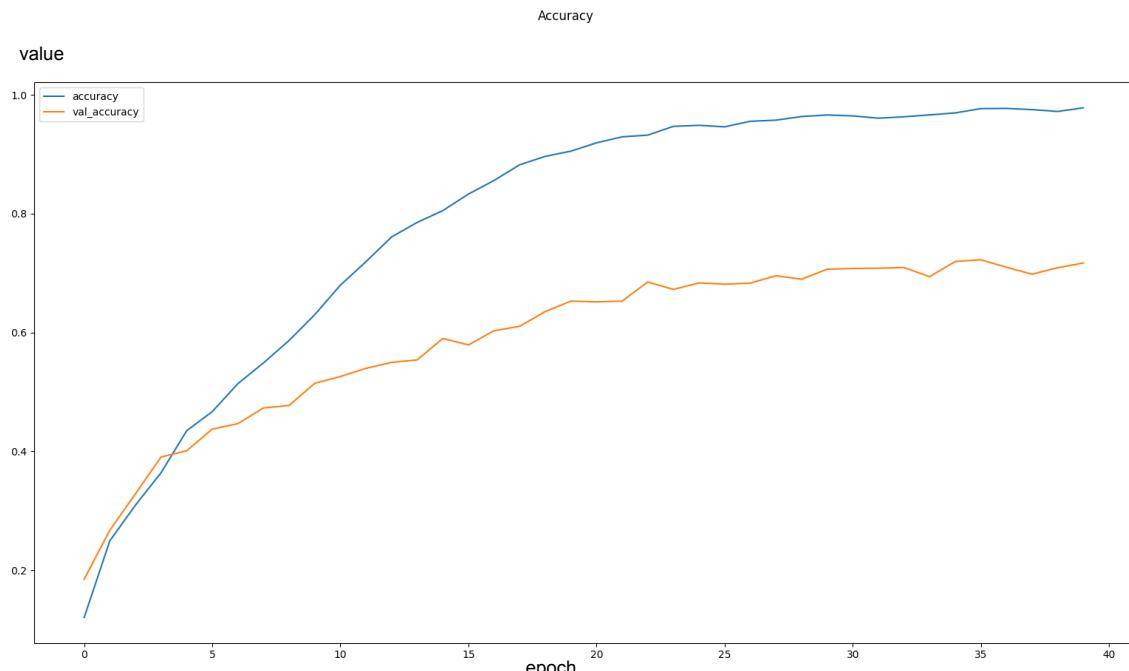
Korzystając z naszego modelu sieci spróbowaliśmy wyuczyć ją na większej ilości danych dla wszystkich stylów z użytej przez nas bazy danych. Ograniczyliśmy jednak dane do **1000 obrazów** na styl.

Zbiór podzieliliśmy tak jak poprzednio:

- 70% - **dane treningowe** - wykorzystywane w celu uczenia modelu,
- 20% - **dane walidacyjne** - wykorzystywane w celu walidacji w trakcie uczenia modelu,
- 10% - **dane testowe** - wykorzystywane w celu ostatecznej walidacji po fazie uczenia

Otrzymaliśmy następujące wyniki:





Wynik na danych testowych:

```
4/4 [=====] - 99s 1s/step - loss: 3.2366 - accuracy: 0.6047
```

Jak da się zauważyć nasza obecna architektura sieci nie poradziła sobie tak dobrze na większej liczbie stylów. Otrzymaliśmy skuteczność na poziomie **ok. 60%**, co w porównaniu do **89%** dla pięciu stylów wypada nie najlepiej, oraz wartość funkcji straty na poziomie **ok. 3.24**, co jest względnie wysoką wartością. Warto pamiętać, że przy większej ilości stylów skuteczność zazwyczaj będzie niższa, gdyż sieć ma więcej możliwości do wyboru. Dodatkowo nasz model był dostosowany do pięciu stylów, więc dla większej ilości należałoby zoptymalizować parametry warstw.

Nasz wynik porównaliśmy do artykułu "[Art Classifier using WikiArts](#)" użytkownika *ROMAN KAHARLYTSKYI*, wykorzystującego tą samą bazę danych:

loss	accuracy
[ 0.9325546622276306 , 0.6928235292434692 ]	

Jak widać wyniki tego użytkownika są lepsze. Korzystał on jednak z gotowych modeli utworzonych przez **profesjonalistów**, posiadających więcej środków do zbudowania odpowiedniego modelu.

# Transfer stylu

Zaimplementowaliśmy gotowe modele służące transferowi stylu z jednego obrazu na drugi. Pierwszy (**NST**) z artykułu: [https://www.tensorflow.org/tutorials/generative/style\\_transfer](https://www.tensorflow.org/tutorials/generative/style_transfer), a drugi (**Magenta**) znajdujący się w repozytorium: <https://tfhub.dev/google/magenta>. Oba modele oczekują na wejściu obrazu do przerobienia oraz obrazu, z którego **wyekstrahują styl**. Następnie wyekstrahowany styl nakładają na pierwszy obraz.

**NST** korzysta z warstw konwolucyjnych gotowego wyuczonego modelu **VGG19** służącego do **klasyfikacji obrazów**. Za ich pomocą uczy się cech obrazu ze stylem. Najpierw ekstrahuje zawartość neuronów tych warstw, następnie oblicza ich gradient i na koniec nakłada go na obraz docelowy. Przeprowadzając tę operację wiele razy, uzyskujemy przerobione zdjęcie.

**Magenta** jest kilka etapów dalej, jeśli chodzi o rozwój techniki transferu stylu. Działa ona dla dowolnego stylu, jednocześnie działa nieporównywalnie szybciej niż **NST** (NST – około 10 minut na obraz; Magenta – około kilkunastu sekund na obraz). Dodatkowo jej wyniki są dużo lepsze.

## Porównanie wyników obu modeli

	Realizm	Rokoko	Symbolizm	Ekspresjonizm
Obraz wejściowy				
NST				
Magenta				

W ramach testu spytaliśmy nasz model o **predykcje** dotyczące tych obrazów:

	<b>Realizm</b>	<b>Rokoko</b>	<b>Symbolizm</b>	<b>Ekspresjonizm</b>
<b>NST</b>	Styl japoński	Ekspresjonizm	Styl japoński	Ekspresjonizm
<b>Magenta</b>	Styl japoński	Ekspresjonizm	Styl japoński	Ekspresjonizm

Oba modele generują bardzo podobne obrazy dla **realizmu, rokoko i symbolizmu**. Z tego powodu nasz model nie zauważa różnic dla **realizmu i symbolizmu**. **NST** przekształca grafiki w dosyć podobny sposób, niezależnie od stylu. **Magenta** radzi sobie pod tym względem dużo lepiej i wygenerowane przez nią obrazy są dużo bardziej przyjazne dla oka, w szczególności obraz wygenerowany dla ekspresjonizmu.