Analiza Obrazów - dokumentacja projektu

Michał Kacprzak
Paweł Korytowski
Mikołaj Marchewa

25 stycznia 2021

Spis treści

1 Opis projektu

Projekt zakłada stworzenie aplikacji dzięki, której użytkownik będzie mógł dowiedzieć się czy wybrane przez niego zdjęcie zostało kiedykolwiek zmodyfikowane poprzez wklejenie dodatkowego elementu. Poniższy przykład lepiej obrazuje cel naszego projektu. Na rysunku nr 1 znajduje się przykładowy oryginalny obraz natomiast na rysunku nr 2 i 3 znajdują się jego modyfikacje Obraz z rysunku nr 2 powstał poprzez zmianę parametru gamma. Natomiast obraz z rysunku nr 3 jest rezultatem wklejenia kilku losowych obrazków. Naszym celem było stworzenie programu, który jako modyfikacje oznaczy tylko obraz nr 3, ponieważ powstał on na skutek wklejenia a nie modyfikacji parametru obrazu.

Rysunek 1: Oryginalny obraz

Rysunek 2: Przykładowa modyfikacja obrazu nr 1

Rysunek 3: Przykładowa modyfikacja obrazu nr 2

2 Założenia wstępne przyjęte w realizacji projektu

1. umożliwienie użytkownikowi wczytania zdjęcia w formacie PNG lub JPG o dowolnym rozmiarze. Jednak im bardziej wymiary będą różnić się od 512x512 tym rezultat będzie mniej wiarygodny, dlatego zalecamy korzystanie z zdjęć w dokładnie takim rozmiarze;
2. udostępnienie użytkownikowi możliwości edycji wczytanego zdjęcia;
3. stworzenie datasetu, który będzie potrzebny do treningu sieci neuronowej;
4. znalezienie optymalnej konfiguracji sieci neuronowej;
5. graficzne przedstawienie rezultatów otrzymanych dzięki wytrenowanej sieci neuronowej;

3 Wykorzystane technologie

Projekt został napisany w całości w Pythonie 3 (co sprawia, że posiadanie pythona wersji co najmniej 3.7 jest wymogiem do uruchomienia projektu). Dodatkowo wykorzystaliśmy następujące biblioteki:

- PyQt5 - stworzenie prostego GUI aplikacji;
- tensorflow - w celu pracy z siecią neuronową;
- Pillow - stworzenie datasetu, który był używany do treningu sieci neuronowej.
- cv2 - modyfikacje wczytanego obrazu;

4 Sposób uruchomienia programu

```
$ pip install requirements.txt
$ python3 src/main.py
```

5 Podział pracy

Dokonany został następujący podział zadań:

- Michał Kacprzak - stworzenie graficznej strony GUI za pomocą Qt Designera, implementacja funkcjonalności głównego menu oraz w wstepnego okna, stworzenie datasetu używanego do treningu sieci neuronowej, stworzenie dokumentacj;
- Paweł Korytowski - stworzenie i trening sieci neuronowej, analiza i obróbka danych, stworzenie dokumentacji;
- Mikołaj Marchewa - implementacja funkcjonalności w edytorze wczytanego obrazu oraz w wynikowym oknie, stworzenie dokumentacji

6 Analiza projektu

6.1 Dataset

Realizowanie projektu rozpoczęliśmy od stworzenia datasetu, który będzie użyty do treningu sieci neuronowej. Bazą naszego datasetu został zbiór obrazów dostępny na stronie kaggle. Obrazy z tego zbioru zgrupowaliśmy w jeden folder, ponieważ w treningu sieci miały być sklasyfikowane jako autentyczne. Następnie za pomocą biblioteki Pillow stworzyliśmy przerobioną kopię każdego obrazu. Dla każdego obrazu losowaliśmy miejsce modyfikacji oraz jaki obrazek zostanie wklejony. Przykładowy rezultat modyfikacji obrazu znajduje się poniżej.

Rysunek 4: Oryginalny obraz

Rysunek 5: zmodyfikowany obraz

Rysunek 6: maska użyta do modyfikacji obrazu

Obrazy, które zostały sztucznie zmodyfikowane posłużyły do treningu sieci neuronowej, ponieważ zostały sklasyfikowane jako falsyfikaty.

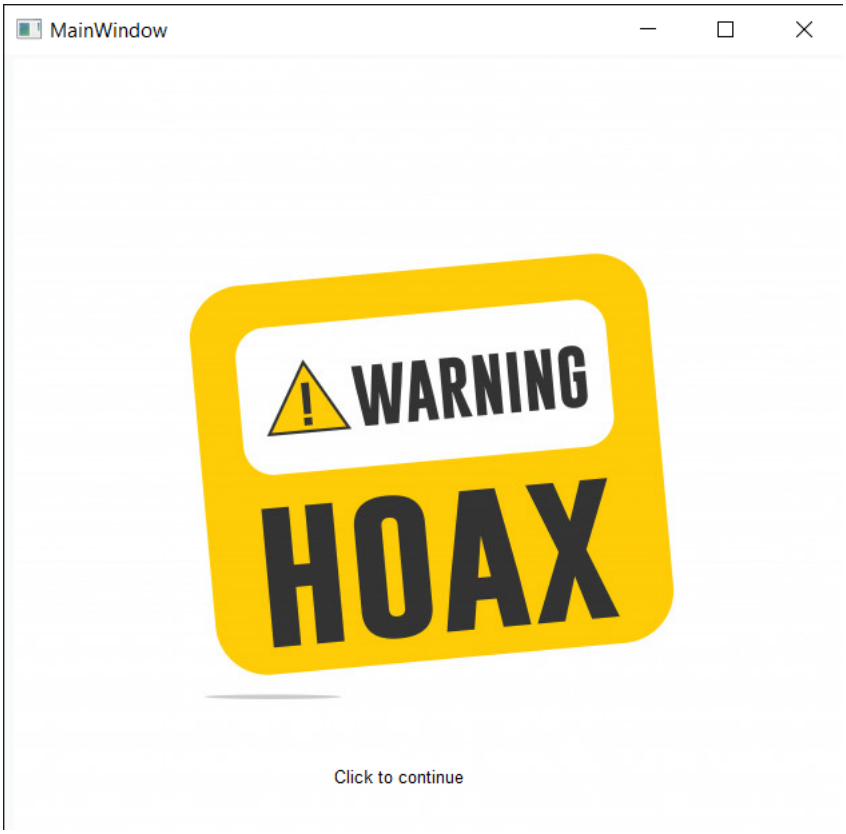
6.2 GUI

GUI naszej aplikacji zostało stworzone za pomocą frameworka Qt, a dokładniej biblioteki PyQt, która umozliwia tworzenie aplikacji w Pythonie za pomocą tego frameworka. Layout poszczególnych okien został stworzony za pomocą Qt Designera. Następnie została zaimplementowana warstwa logiczna aplikacji, czyli np przełączanie pomiędzy poszczególnymi oknami aplikacjami. W naszej aplikacji można wyróżnić następujące okna:

6.2.1 IntroWindow

Poniższy rysunek przedstawia okno, które użytkownik widzi jako pierwsze w momencie gdy uruchamia aplikacje. Zawiera tylko logo aplikacji. W momencie uruchomienia aplikacji rozpoczyna się ładowanie modelu sieci neuronowej. Użytkownik musi zaczekać na poprawne załadowanie się modelu. Poprawne załadowanie zostanie zasygnalizowane poprzez zmiane wyświetlanego napisu z "Loading Neutral Network" na "Click to continue".

Dopiero wtedy, jeśli chce może przejść dalej poprzez kliknięcie na okno. Zostanie wtedy przekierowany do StartWindow.

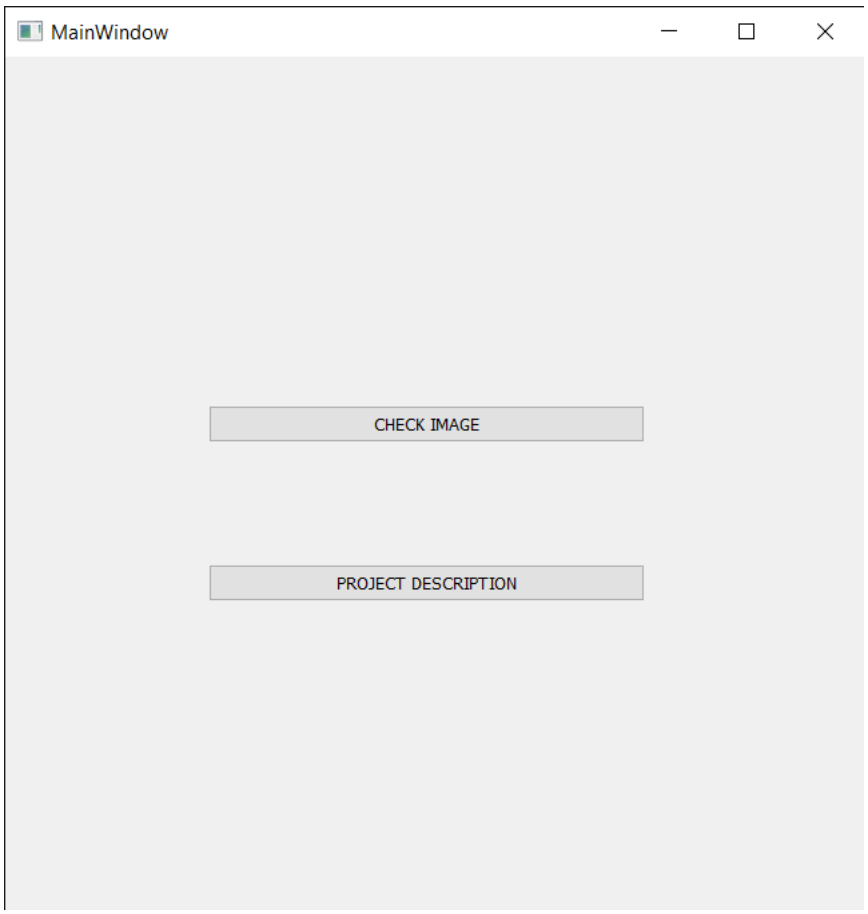


Rysunek 7: Wygląd okna `IntroWindow`

6.2.2 StartWindow

- Poniższy rysunek przedstawia główne okno aplikacji. Udostępnia następujące funkcjonalności użytkownikowi:

1. sprawdzenie zdjęcia - główna funkcjonalność programu. Po kliknięciu na przycisk "CHECK IMAGE" użytkownik zostanie poproszony o wybranie dowolnego zdjęcia które aktualnie znajduje się na komputerze użytkownika a następnie zostanie przeniesiony do ImageEditorWindow.
2. przeczytanie dokumentacji projektu - Po kliknięciu na przycisk "PROJECT DESCRIPTION" - zostanie otwarty plik zawierający dokumentacje projektu

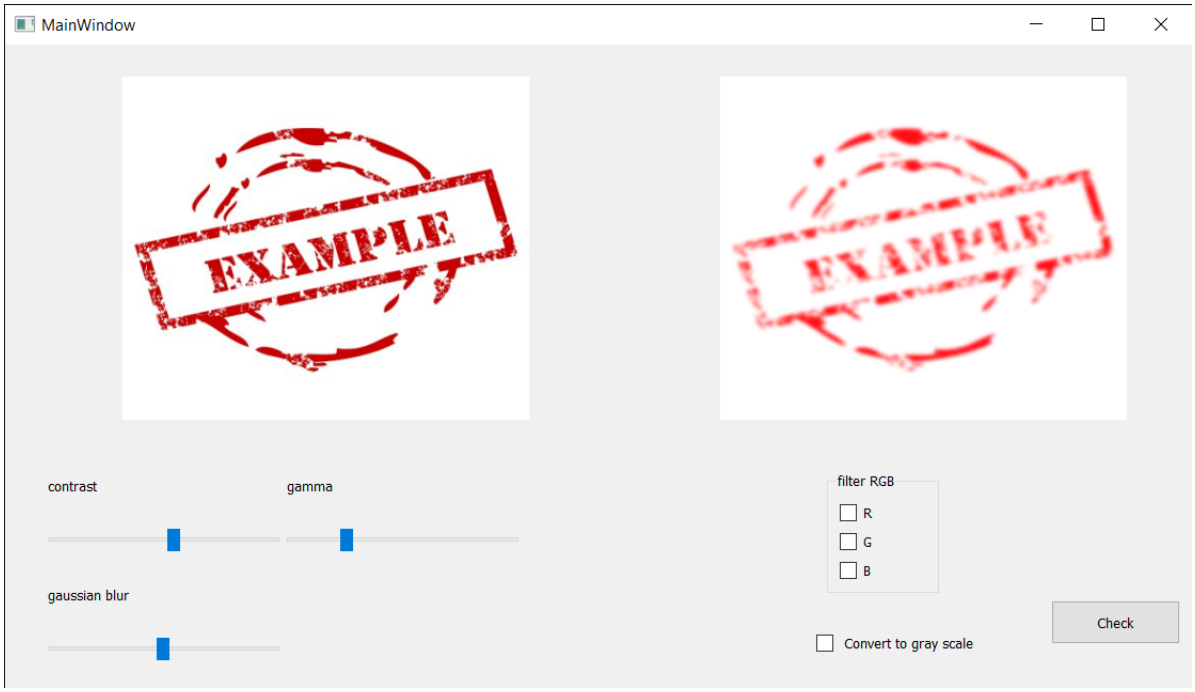


Rysunek 8: Wygląd okna `StartWindow`

6.2.3 ImageEditorWindow

Poniższy rysunek przedstawia okno, w którym użytkownik może modyfikować wybrane wcześniej przez siebie zdjęcie. Celem tego edytora jest umożliwienie użytkownikowi sprawdzenia jak zmiana poszczególnych parametrów wpływa na działanie wytrenowanej sieci. Użytkownikowi może modyfikować zdjęcie w następujący sposób:

1. zmiana kontrastu zdjęcia;
2. zmiana parametru gamma zdjęcia;
3. nałożenie filtru Gaussa na zdjęcie;
4. przekonwertowanie zdjęcia do skali szarości;
5. filtrowanie obrazu poprzez zerowanie kanałów RGB; Powyższe możliwości modyfikacji zostały uzyskane za pomocą wbudowanych funkcji biblioteki cv2

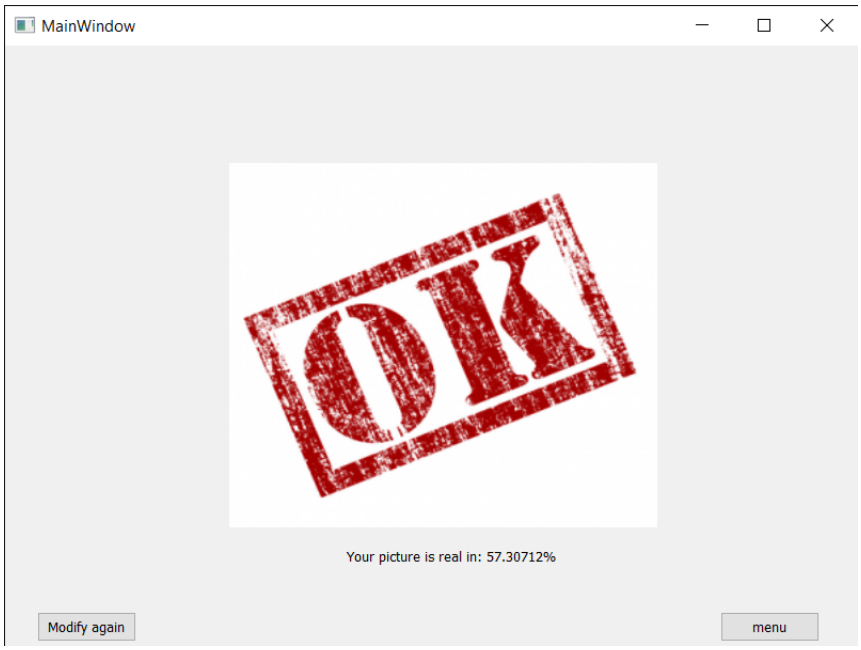


Rysunek 9: Wygląd okna `ImageEditorWindow` wraz z zastosowanym pewnymi modyfikacjami (po lewej oryginalny obraz, po prawej zmodyfikowany)

6.2.4 ResultWindow

Poniższy rysunek przedstawia okno, w którym użytkownikowi zostają przedstawione wyniki działania sieci neuronowej, która sprawdza czy i z jaką pewnością dany obraz był modyfikowany poprzez wklejenie dodatkowego obiektu do zdjęcia. Udostępnia następujące funkcjonalności użytkownikowi:

1. powrót do MenuWindow - poprzez kliknięcie przycisku "menu";
2. powrót do ImageEditorWindow - poprzez kliknięcie przycisku "Modify again";



Rysunek 10: Wygląd okna `ResultWindow`

6.3 Sieć neuronowa

1. Do wykonywania analizy autentyczności wykorzystana została sieć neuronowa Inception V3 autorstwa Google'a. Jest to sieć wyspecjalizowana w wykrywaniu obiektów i kształtów na obrazach. Aby zwiększyć dokładność działania zastosowana została metoda transfer learningu. Oznacza to, że sieć została wstępnie wytrenowana na bazie obrazów (w tym przypadku bazie Imagenet), a dopiero na końcu na przygotowanym przez nas datasecie.

2. Sieć jako parametry wejściowe przyjmuje trzykanałowy obraz. Na wyjściu otrzymujemy tablicę wyników o długości zależnej od ilości klas (w naszym przypadku 2 - prawda, fałsz), która określa procentowe prawdopodobieństwo przynależności obiektu do każdej z nich.

3. Do pracy nad siecią zostało wykorzystane środowisko Google Colab. Jego zaletą jest darmowy dostęp do GPU, które znacząco przyspiesza trening. Dzięki temu możliwe było wykorzystanie większej liczby obrazów dla zwiększenia dokładności (docelowy model został wytrenowany za pomocą 4500 obrazów). Po zakończeniu treningu otrzymujemy wytrenowany model w postaci pliku .hdf5, który zostanie załączony i wykorzystany w kodzie aplikacji.

4. W docelowym programie korzystanie z wytrenowanego modelu jest bardzo proste. Podczas uruchamiania programu model zostaje załadowany do pamięci. Następnie po kliknięciu klawisza "check" wejściowy obraz zostaje przekazany do metody predict() modelu, która zwraca wynik procentowy. Jako przynależność do danej klasy zostanie uznany najwyższy z obu wyników. Do wyniku użytkownik otrzyma wartość prawdopodobieństwa, że obraz jest prawdziwy/fałszywy.

5. Pierwotnie chcieliśmy wspomóc się analizą wyników szybkiej transformaty Fouriera. Po wielu próbach w tym analizy widma fazowego, amplitudowego i obydwu na raz doszliśmy do wniosku, że najlepsze efekty daje trening na kolorowych obrazach, poddanych jedynie normalizacji. To potwierdza fakt, że sieć najlepiej radzi sobie z wykrywaniem kształtów rzeczywistych obiektów. Wykrywanie fałszerstw korzystające z FFT wymaga odmiennego podejścia, które zasiało w nas ziarno ciekawości, natomiast brak czasu uniemożliwił szersze prace nad tym zagadnieniem.

6. Parametry dla których przeprowadzony został trening: wielkość zbioru treningowego 4244, wielkość zbioru testowego 426, wielkość pojedynczej paczki: 16, liczba epok 30. Uzyskane rezultaty 99,9 procent zbioru testowego. Podczas testów z użyciem widma, amplitudy otrzymane rezultaty wahały się w granicach 50-60%.

7. Skrypty użyte podczas treningu, wraz z opisem metod zostały umieszczone w projekcie w folderze notebooks.

7 Testowanie

Testowanie aplikacji przebiegało równoległe wraz z implementacją kolejnych funkcjonalności. Takie postępowanie zagwarantowało nam bieżące wykrywanie i usuwanie błędów co pozwoliło uniknąć nakładania się błędów na siebie. W rezultacie finalna wersja programu jest w naszej ocenie pozbawiona błędów. Działanie sieci było testowane na własnoręcznie stworzonym zbiorze obrazów. W folderze "testing_images/original" znajdują się zdjęcia, które są nie modyfikowane i za każdym razem zostaną uznane przez aplikacje za niemodyfikowane. Natomiast w folderze "testing_images/fake" znajdują się zdjęcia modyfikowane i każdorazowo przy ich sprawdzeniu wynik będzie potwierdzał że te zdjęcia były modyfikowane. Powyższe testowanie odbywało się przy braku modyfikacji testowanych zdjęć w utworzonym przez nas edytorze. Zadbaliśmy także o przetestowanie wpływu modyfikacji możliwych do dokonania w naszym edytorze na rezultat działania sieci. Zaobserwowaliśmy, że wpływ ten może być zarówno gigantyczny jak i minimalny. Największy wpływ ma nałożenie filtru gaussowskiego lub ustawienie pozostałych parametru na skrajne wartości. Wyniku to z faktu, że użyta przez nas sieć jest wyspecjalizowana w rozróżnianiu obiektu. A takie ustawienie parametrów powoduje, że obiekty są gorzej widoczne.

8 Podsumowanie

Efektem końcowym realizacji projektu jest aplikacja, w której użytkownik ma możliwość wczytania wybranego przez siebie zdjęcia, jego modyfikacji oraz sprawdzenia czy obraz był modyfikowany poprzez wklejenie elementu. W ramach dalszego rozwijania projektu należy poprawić całościowy wygląd aplikacji, aby uczynić ją atrakcyjniejszą dla użytkownika. Kolejnym elementem wymagającym poprawy jest umożliwienie użytkownikowi wczytania zdjęcia o dowolnym rozmiarze. Dodatkową funkcjonalnością którą w następnej kolejności należałoby

zaimplementować z pewnością jest wykrywanie obszarów zdjęcia w których doszło do modyfikacji i ich zaznaczanie. Należałoby także zadbać o zminimalizowanie wpływu modyfikacji dostępnych edytorze na wynik, jednakże prawdopodobnie wymagałoby to zupełnie innego podejścia. Reasumując aplikacja spełnia wszystkie konieczne wymogi i implementuje wszystkie wstępne założenia projektu.