Kolorowanie szarych obrazków

Jacek Strzałkowski

Warszawa, styczeń 2025

Zasób	Adres
Repozytorium projektu	github.com/jacekstrzalkowski/picturecolorizing
Kaggle notebook	kaggle.com/code/jacekstrzakowski/colorizing/
Dataset	kaggle.com/datasets/jacekstrzakowski/autocolorization

Wprowadzenie

Sztuczne kolorowanie obrazów jest rozumiane jako klasyfikacja obrazu w skali szarości $\mathbf{X} \in \mathbb{R}^{H\times W\times 1}$ do obrazu kolorowego $\tilde{\mathbf{X}}$: tablicy 2D (H, W) trójek $(\tilde{\mathbf{X}}_{\mathbf{R}}, \tilde{\mathbf{X}}_{\mathbf{G}}, \tilde{\mathbf{X}}_{\mathbf{B}})$, $\tilde{\mathbf{X}} \in \mathbb{R}^{H\times W\times 3}$.

Postępując podobnie jak w (Zhang, Isola, and Efros 2016), przechodzimy z reprezentacji `RGB` obrazu do reprezentacji kolorów przestrzeni barw `LAB`. W przestrzeni LAB składowa jasności L jest oddzielona od składowych: a - barwy czerwono - zielone, b - niebiesko żółte,

$$\tilde{\mathbf{X}}_{Lab} = \{L, \tilde{a}, \tilde{b}\}, \quad \tilde{\mathbf{X}}_{Lab} \in \mathbb{R}^{H\times W\times 3}.$$

Sieć neuronowa M w działaniu na obraz $\mathbf{X}$ zwraca $\tilde{\mathbf{X}}_{Lab}$, który następnie jest poddawany odwrotnej transformacji z przestrzeni `Lab` do `RGB`. Zauważmy, że pozwala to ująć problem kolorowania obrazów jako klasyfikacji jedynie dwóch kanałów (a, b), podczas gdy jasność pozostaje dostępna jako wejście.

Figure 1: Rezultaty kolorowania. Środkowy obraz przedstawia wzorzec a prawy - predykcję modelu. Poniżej zostanie zaproponowane wyjaśnienie, ale warto zauważyć, że średnia wartość kanałów barw a, b jest podobna dla obu wyników a z kolei sam "rozstrzał" jest znacząco większy dla wzorca.

Szczegóły implementacji

Użyta architektura sieci neuronowej została zaczerpnięta z (Zhang, Isola, and Efros 2016) natomiast implementacja została poprawiona i unowocześniona. Sieć można przedstawić schematycznie w następujący sposób

Do trenowania sieci wykorzystano `PyTorchLightning`. Do wyliczenia błędu wykorzystano `MSE`, do optymalizacji wag sieci wykorzystano optymalizator `Adam`. Użyto `MSE` do wyznaczenia błędów. Sposób trenowania był autorski i niezależny od (Zhang, Isola, and Efros 2016).

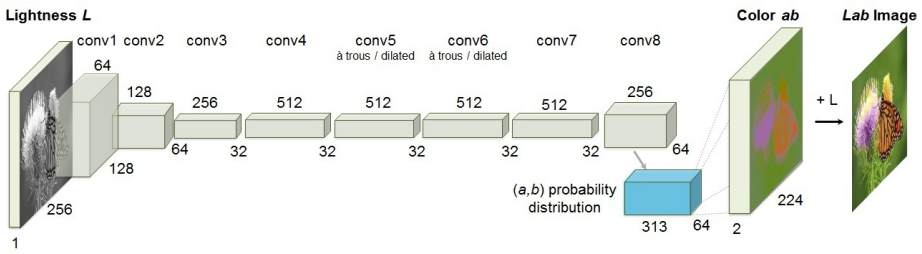

Figure 2: Użyta sieć. Architektura została opisana w (Zhang, Isola, and Efros 2016)

Wyniki

Przeprowadzono trenowanie na próbie 1000 i 50 tys. obrazków. Uzyskano błąd sprawdzenia ≈ 228. Podobne wyniki dla tak różnych prób oraz mierne rezultaty "organoleptyczne" sugerują, że obecna metoda trenowania sieci nie pozwala na satysfakcjonującą metodą "kolorowania".

Ciekawe wnioski nasuwa analiza rozkładu kolorów a, b dla wartości przewidywanych w kontrze do wartości uczonych. O ile dla dobrze wytrenowanej sieci wartość średnia kanałów a, b obrazu przewidywanego jest podobna do tej dla wzorca, to wartość odchylenia standardowego jest z reguły dwa razy mniejsza.

Obrazuje to poniższy przykład, który pokazuje również, że sieć nie ma problemów z mapowaniem różnych wartości kolorów tzn. że nie powstaje efekt podobny do sepii.

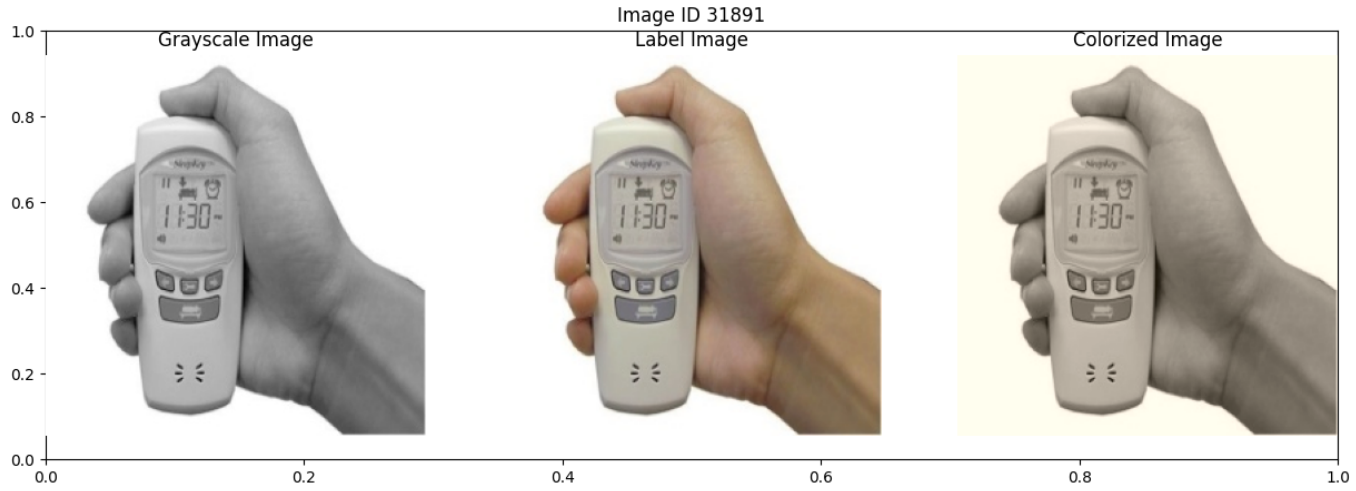

Figure 3: Warto zauważyć, że model dobrze oddaje "średnie ubarwienie" obrazu. Można to dostrzec na przykładzie obrazów, dla których średnie ubarwienie jest nietypowe. W tym wypadku widać, że duża ilość "żółtego" koloru z telefonu została oddana, odpowiednio przeskalowaną, jako tło.

```
TARGET MEAN, STD: 132.95562744140625, 6.902090072631836
PREDICTED MEAN, STD: 133.1239013671875, 3.2288103103637695
```

Podsumowując, przewidywanie (Zhang, Isola, and Efros 2016), że standardowe techniki uczenia oparte na SGD nie pozwolą na satysfakcjonujący model, sprawdziły się. Warto zauważyć, że przykłady o najniższych uzyskanych wartościach błędu, nie sprawiają wrażenia dobrze ubarwionych. Ponadto wartość odchylenia standardowego jest w zasadzie taka sama dla każdego obrazka!

Zbiór danych

Struktura zbioru danych

```
dataset
    train
        img
        label
    val
        img
        label
```

Obrazki zostały zaczerpnięte z https://www.kaggle.com/datasets/lijiyu/imagenet. Następnie został opracowany w sposób przedstawiony w `prepare_data.ipynb`. W zbiorze treningowym znajduje się 50 tys. obrazków `JPEG`.

Wprowadzoną modyfikacją w stosunku do (Zhang, Isola, and Efros 2016) jest przeskalowanie wszystkich obrazków do tego samego wymiaru $(256, 256)$, co w (Zhang, Isola, and Efros 2016) było realizowane w procedurze przygotowawczej dla każdego obrazka.

Dodatkowym wnioskiem po `P4` jest to, że dzielenie zbioru obrazów na `img` i `label` w tej formie nie miało zbyt dużo sensu. Należy poprawić to, budując jeden folder ze zdjęciami `RGB`, który z kolei nie zawierałby obrazków problematycznych dla problemu kolorowania np. dość szarych.

Alternatywnie należy przeprowadzić konwersję z `RGB` do `LAB` poza procedurą uczenia oraz znaleźć sposób zapisu obrazków w `LAB` i z nich sporządzić bazę danych treningowych.

Problemy

- Sugerowane przez (Zhang, Isola, and Efros 2016) rozwiązanie dot. konwersji z przestrzeni `RGB` do `LAB` sprawiało dużo kłopotów, toteż zostało zaimplementowane nowe rozwiązanie bazujące na `opencv`.
- Trudno uzyskać obrazy o dużym urozmaiceniu barw. Proste "skalowanie" wokół średniej wartości kolorów ab.

$$ab' = E_{ab} + 2 \cdot (ab - E_{ab})$$

tak, żeby odchylenie było większe, nie działa.

Plany na usprawnienia

Do projektu w fazie *baseline* dostarczono model, który koloryzuje szare obrazki. Był on trenowany za pomocą klasycznego SGD. Ta funkcja kosztu może okazać się nie optymalną do problemu, bowiem powoduje uśrednianie kanałów a i b, co objawia się szarymi i *zdesaturowanymi* obrazami.

W fazie końcowej projektu zostanie zaimplementowana alternatywna funkcja kosztu przedstawiona w (Zhang, Isola, and Efros 2016) oparta na Softmaxie.

Przestrzeń ab zostanie podzielone na przedziały o rozmiarze $d = 10$. Przestrzeń "robocza" ab będzie określona dyskretnie: ilość kombinacji (a, b) będzie wynosiła $Q = 313$. Następnie model będzie uczony mapowania $X \mapsto [0, 1]^{H \times W \times Q}$. Może być to rozumiane jako klasyfikacja wieloklasowa, gdzie ilosć klas to Q.

Dodatkowo, zostanie zaimplementowany mechanizmu *rebalancingu* klas. Warto zauważyć, że ten mechanizm nie polega na zmianie architektury sieci, lecz modifykacji optymalizacji: błąd na jednym pikselu będzie powodował różną zmianę wag, ze względu na rzadkość występowania koloru w pikselu. Zapobiegnie to postrzeganej *desaturacji* obrazu.

Postępując podobnie jak (Zhang, Isola, and Efros 2016) zostanie zaimplementowany sposób wyznaczania finalnego wyznaczania rozkładu barw. Chodzi tutaj o operacje przeprowadzaną po trenowaniu sieci, ale przed konwersją na `RGB`. Doprowadzić ma to znowu do kompensacji zjawiska desaturacji.

Bibliografia

Zhang, Richard, Phillip Isola, and Alexei A. Efros. 2016. "Colorful Image Colorization." October 5, 2016. http://arxiv.org/abs/1603.08511.