Metody Rozpoznawania Obrazów

Laboratorium 1

Maria Polak

1. Framework

Wybranym przeze mnie frameworkiem do pracy na zajęciach z tego przedmiotu jest Tensorflow. Głównym powodem jest poprzednia styczność z nim na zajęciach z przedmiotu “Elementy Statystycznego Uczenia Maszynowego” prowadzonego na 4 semestrze studiów oraz obszerna baza materiałów pomocniczych (dokumentacja, artykuły, wpisy na forach).

2. Przygotowanie zbioru

Ze zbioru zostały usunięte wszystkie kategorie ubrań poza “sandałami” (5) oraz “sukienkami” (3). Po przefiltorwaniu zbioru otrzymaliśmy tensory trójwymiarowe o wymiarach (12000, 28, 28) i (12000,).

Filtracja zdjęć została wykonana poprzez stworzenie mask binarnych, w których zaznaczone były elementy do zachowania.

```
bin_mask_train = (y_train_data == 3) | (y_train_data == 5)
bin_mask_test = (y_test_data == 3) | (y_test_data == 5)

x_train = x_train_data[bin_mask_train]
y_train = y_train_data[bin_mask_train]
x_test = x_test_data[bin_mask_test]
y_test = y_test_data[bin_mask_test]
```

Rysunek 1. Przykładowe elementy skórczonego zbioru

3. Model decyzyjny - regresja logistyczna

Konstrukcja modelu w bibliotece Tensorflow

W frameworku Tensorflow, jedną z istniejących możliwości konstruowania modelu (odpowiednią dla tego laboratorium) jest tworzenie modelu Sekwencyjnego zaimplementowanego w pakiecie “Keras”. Model sekwencyjny jest odpowiednikiem stosu warst, gdzie każda warstwa ma jeden tensor na wejście i wyjście. Tensorflow oferuje możliwośc inicjalizacji modelu i następnie dodawania do niego kolejnych warst. Przykładowa konstrukcja modelu wygląda następująco:

```
model = keras.Sequential()
model.add(layers.Dense(2, activation="relu"))
model.add(layers.Dense(3, activation="relu"))
model.add(layers.Dense(4))
```

Wstępna obróbka i skrawanie

Celem przystosowania dancyh wejściowych do formatu, na którym pracuje regresja logistyczna, zostały wykonane następujące przekształcenia:

- Spłaszczenie obrazów na wektory zostało obsłużone przez dodanie do modelu warstwy “Flatten” . Wartwa ta przekształca obraz o rozmiarze (28,28) do wektora o rozmiarze (784)

```
model.add(tf.keras.layers.Flatten(input_shape = (28, 28)))
```

- Normalizacja wartości została obsłużona przez dodanie do modelu warsty “Normalization”. Zamienia ona wartości w wektorach z początkowego przedziału na zakres <0,1>

```
model.add(tf.keras.layers.Normalization())
```

- Zamiana etykiet klas na 1 i 0 została obsłużona “ręcznie”.

```
y_train[y_train == 5] = 1
y_train[y_train == 3] = 0
y_test[y_test == 5] = 1
y_test[y_test == 3] = 0
```

Wstępny test modelu

Nie udało mi się podpiąć debuggera, żeby przetestować działanie “na żywo” jednak z materiałów dostępnych w internecie wynika, że jest to wspierane. Przepuszczenie przykładowych danych (pierwszych 20 obrazków) przez przygotowany model zwróciło oczekiwane efekty - zdjęcia zostały przekształcone do znormalizowancyh wektorów.

Implementacja regresji logistycznej

Dodany został mechanizm “Dense” z aktywacją “Sigmoid” do przygotowanego modelu.

```
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

Podsumowanie stworzonego modelu prezentuje się następująco:

```
Model: "sequential_17"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_17 (Flatten)         (None, 784)               0
_________________________________________________________________
normalization_16 (Normalizat (None, 784)               1569
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 785
=================================================================
Total params: 2,354
Trainable params: 785
Non-trainable params: 1,569
```

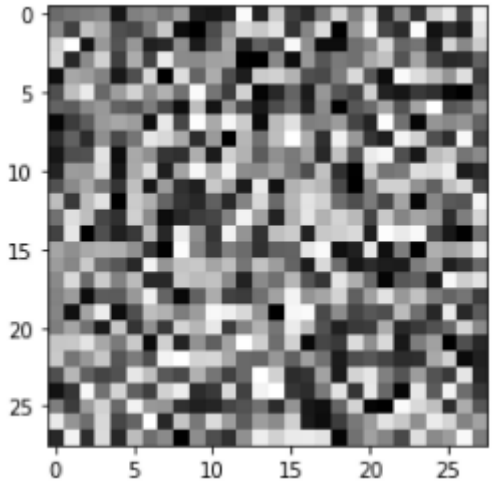
Stworzony przez nas model ma sens ze strony matematycznej. Rozmiary, jak i parametry są zgodne z naszymi przewidywaniami.

Wagi funkcji liniowej

W modelu sekwencyjnym dostęp do warstwy gęstej i jej wag odbywa się następująco:

```
model.layers[id_warstwy].weights
```

Po narysowaniu wartości w warstwie gęstej widzimy, że jest to faktycznie losowy bezsensowny szum.

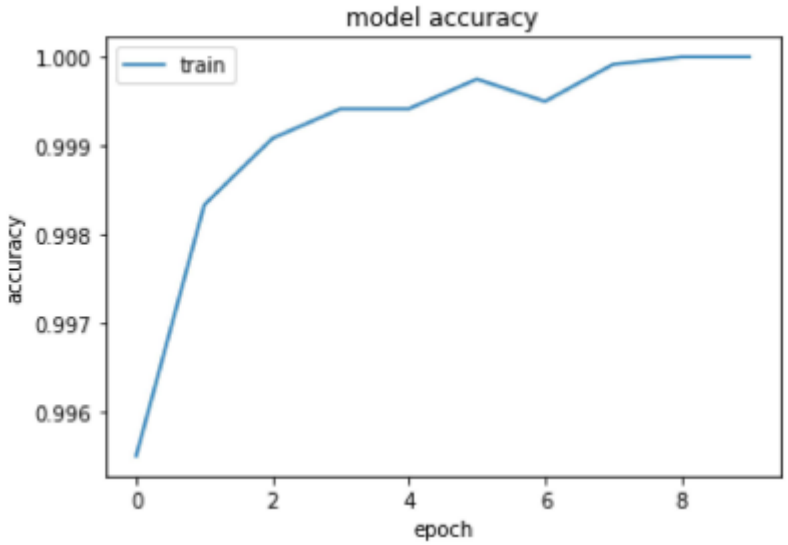


Rysunek 2. Wagi funkcji liniowej zawarte w warstwie gęstej po inicjalizacji

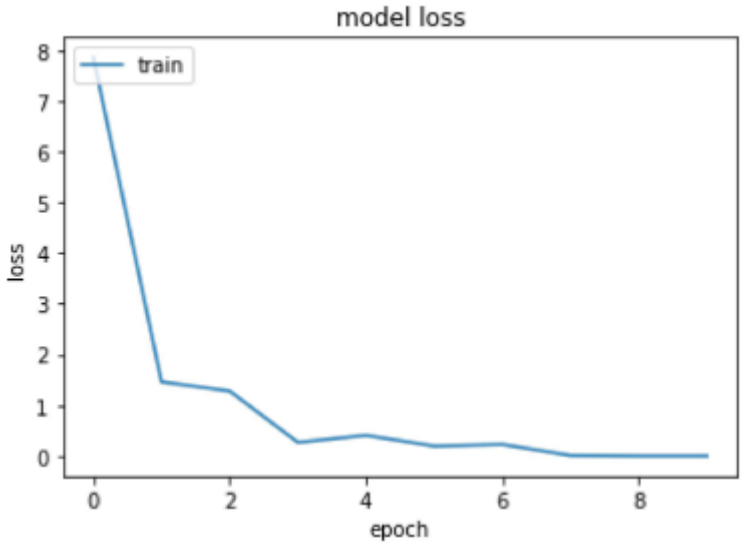
Do modelu dodana została funkcja straty (binarna entropia krzyżowa) oraz mechanizm optymalizujący (stochastyczny spadek po gradiencie). Tensorflow umożliwia dodanie tych elementów modelu w funkcji “model.compile()”.

```
model.compile(optimizer='sgd', loss='binary_crossentropy',
metrics=['binary_accuracy'])
```

Przy pomocy funkcji model.fit() zostało uruchomione strojenie modelu na danych treningowych. Wybrana liczba epok to 10, a rozmiar batcha to 16.



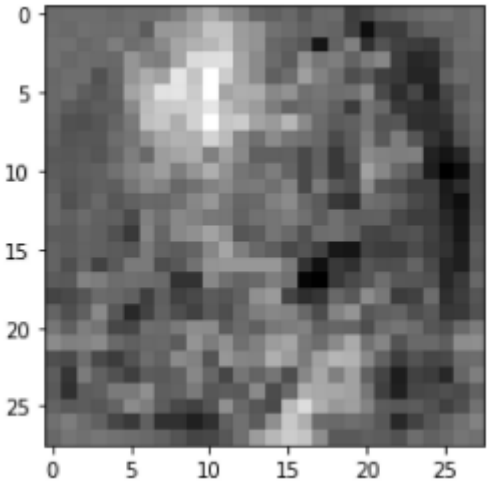
Wykres 1. Celność modelu w kolejnych epokach



Wykres 2. Wartość funkcji straty w kolejnych epokach

Uzyskane wyniki są satysfakcjonujące. Jak widać na wykresach z każdą epoką celność oceny wzrasta, a funkcja straty maleje.

Wygląd filtra uzskanego z wag po strojeniu modelu prezentuje się następująco:



Rysunek 3. Wagi funkcji liniowej zawarte w warstwie gęstej po strojeniu

Nie odzwierciedla on w sposób dokładny żadnej z klas, jednak jest on bardziej zbliżony do ich wyglądu niż po samej inicjalizacji. Szukając “na siłę” można powiedzieć, że na dole obrazka znajduje się lekko widoczny kształt buta, a na górze minimalnie widać ramiączka i dekolty sukienek.

Ewaulacja modelu

Ewaluacja modelu zwraca bardzo wysoką celność rzędu 99.90%. Jest to bardzo dobry wynik jak na tak prostą metodę rozpoznawania. Tak wysokie wyniki są możliwe dzięki świetnemu przygotowaniu danych wejściowych - zdjęcia dostarczone przez zalando zostały zrobione w tym samym świetle oraz z tym samym pozycjonowaniem ubrań na zdjęciach.

Rysunek 4. Przykładowe wyniki decyzji podejmowanych przez model (Pred - klasa przyporządkowana przez model, True - klasa prawdziwa)