Co ja tu widzę...

*Piotr KRZYŻANOWSKI**

* Uniwersytet Warszawski, Wydział Matematyki, Informatyki i Mechaniki

W nowej, dziś inaugurowanej, cokwartalnej serii *Pół szklanki mocnego kodu* w każdym odcinku zajmiemy się konkretnym, czasem niełatwym problemem. Pokażemy, jak go rozwiązać, pisząc program zajmujący mniej więcej pół strony – co będzie możliwe dzięki wykorzystaniu bardzo mocnych bibliotek i technologii. Nie wnikając w ich detale, chcemy zwrócić uwagę Czytelników (także tych niekoniecznie zainteresowanych programowaniem), jak dużo można osiągnąć, pisząc tak niewiele!

Zadziwiające, jak szybko uczenie maszynowe trafiło pod strzechy! Jeszcze nie tak dawno wymagało biegłości w programowaniu, znajomości takich konceptów, jak funkcja aktywacji, rozkład macierzy względem wartości szczególnych, optymalizacja dla funkcji niegładkich itp. Aby wszystko to zadziałało, niezbędny też był dostęp do dostatecznie dobrych danych treningowych i morza czasu obliczeniowego... A dziś?

Teraz możemy, w kilku linijkach mocnego kodu, wykorzystać gotowe, uprzednio wyuczone moduły – na przykład do rozpoznania, jaki *obiekt* przedstawia zdjęcie. Oczywiście, to wszystko jest możliwe pod warunkiem, że mamy dostęp do dostatecznie mądrego pakietu... Wybierzemy więc potężny Tensorflow od samego Google Brain. Sposób jego instalacji na komputerze jest bardzo dokładnie opisany na stronie domowej projektu (ja skorzystałem z instalacji w wirtualnym środowisku Pythona na Ubuntu). Poniższy kod w Pythonie rozpoznaje obiekt ze zdjęcia (u nas: znajdującego się w pliku `image.jpg`):

`www.tensorflow.org`

```
1  from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input, decode_predictions
2  from tensorflow.keras.preprocessing.image import img_to_array, load_img
3
4  def prepare_image_iv3(filename):
5    import numpy as np
6    img = load_img(filename, target_size=(299, 299))
7    return np.expand_dims(img_to_array(img), axis=0)
8
9  model = InceptionV3()
10 x = preprocess_input( prepare_image_iv3('image.jpg') )
11 preds = model.predict(x)
12
13 for name, label, score in decode_predictions(preds)[0]:
14   print("%.2f:␣%s" % (score, label))
```

Na podstawie www.keras.io/applications/

Cabot Tower w miejscowości St. John's, Kanada (plik `image.jpg`).

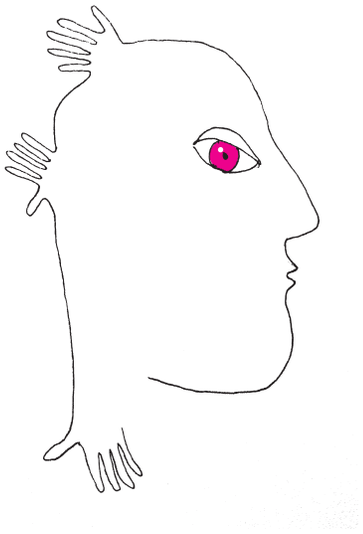
Więcej o sieci Inception: `arxiv.org/abs/1512.00567` oraz `ai.googleblog.com/2016/03/train-your-own-image-classifier-with.html`

Na początku programu (linie 1–2) wybieramy z Tensorflow tylko te funkcje, które będą nam potrzebne. Należą one do modułów pakietu Keras (www.keras.io), który dostarcza interfejs bardzo wysokiego poziomu, m.in. do aplikacji takich, jak InceptionV3. Notabene, inspiracja dla powyższego kodu pochodzi z dokumentacji pakietu. InceptionV3 to wcześniej wytrenowana konwolucyjna sieć neuronowa (Keras daje też dostęp do kilku innych), która przy pierwszym uruchomieniu funkcji zostanie pobrana z internetu – i dalej już cały proces rozpoznawania będzie odbywał się u nas na komputerze, bez korzystania z usług w chmurze obliczeniowej. Sieć neuronowa Inception-v3 została nauczona rozpoznawania obrazów, czyli przypisywania im prawdopodobieństwa przynależności do każdej z 1000 z góry ustalonych klas, w rodzaju: „samochód", „bikini", „pies" itp. Podobno poziom trafności jej wskazań jest podobny do osiąganego przez ludzi!

Najważniejsza część programu – rozpoznanie obrazu – składa się zaledwie z trzech linii 9–11:

```
model = InceptionV3()
x = preprocess_input( prepare_image_iv3('image.jpg') )
preds = model.predict(x)
```

Najpierw inicjalizujemy sieć neuronową (z domyślnymi parametrami). Ponieważ będziemy klasyfikować zdjęcia, musimy przygotować je do postaci odpowiedniej dla Kerasa. Mała funkcja pomocnicza prepare_image_iv3 (linie 4–7) zwraca wstępnie spreparowane zdjęcie z pliku (m.in. nadaje mu odpowiednie rozmiary,



gdyż Inception-v3 oczekuje obrazu o rozmiarach 299×299 pikseli). Następnie zdjęcie poddajemy dalszej obróbce przez `preprocess_input` (normalizując i uśredniając dane). Na końcu – w linii 11 – uruchamiamy sieć, która informuje nas, z jakim prawdopodobieństwem zaliczyłaby rozpoznawane obiekty do każdej z klas.

Wyniki klasyfikacji drukujemy (linie 13–14), przebiegając kolejne elementy listy zwróconej przez funkcję `decode_predictions`; domyślnie zostanie podanych pięć najbardziej prawdopodobnych typowań. Oto wynik dla naszego zdjęcia:

```
0.84: castle
0.02: palace
0.01: monastery
0.01: bell_cote
0.01: church
```

Nieźle, prawda? Program stwierdził, że z prawdopodobieństwem 84% zdjęcie przedstawia zamek, z czym z pewnością zgodziłaby się większość z nas. Jednak nie zawsze będzie tak dobrze, a czasem nawet rezultat będzie całkiem bez sensu. Zachęcam do eksperymentowania: z różnymi zdjęciami i z różnymi klasyfikatorami dostępnymi w Keras!

Ciekawe, że sieć Inception-v3 możemy także douczyć rozpoznawać obrazy według naszych własnych kryteriów (od razu pomyślałem o automatycznym rozpoznawaniu znajomych: coś podobnego potrafi już np. aplikacja `Zdjęcia` w Windows 10). Jak to zrobić po swojemu – można przeczytać na stronach Tensorflow poświęconych *image recognition*. Co prawda tak rozbudowany kod zajmie trochę więcej niż pół strony, ale można się przy tym sporo dowiedzieć!

Zapewne niektórzy Czytelnicy czują pewien niedosyt, widząc opisy kategorii po angielsku („*castle*” zamiast „*zamek*”). Czemu więc nie przetłumaczyć w automatyczny sposób wyników z języka, w którym Czesław Miłosz wykładał, na język, w którym pisał wiersze – korzystając na przykład z Google Translate?

Odpowiednie biblioteki i API znajdziesz na stronie `cloud.google.com/translate/docs/reference/libraries`

PS W rzeczywistości obiekt na zdjęciu… niestety nie jest zamkiem (co łatwo sprawdzić w Wikipedii), ale – nie da się ukryć – bardzo *przypomina* zamek. Ja w każdym razie dałbym się nabrać.

Zadania

Przygotował Andrzej MAJHOFER

F 971. W prostym odcinku miedzianego przewodnika o długości $l = 1\,\mathrm{m}$ płynie prąd $I = 10\,\mathrm{A}$. Ile wynosi pęd przenoszony przez elektrony? Masa elektronu $m_e \approx 9{,}1 \cdot 10^{-31}\,\mathrm{kg}$, a jego ładunek $e \approx 1{,}6 \cdot 10^{-19}\,\mathrm{C}$.
Rozwiązanie na str. 8

F 972. Długi pręt z przewodzącego materiału w kształcie walca zakończonego półkulą ma być ładowany do wysokiego napięcia. Jaki powinien być minimalny promień krzywizny R kulistego końca tego pręta, żeby po naładowaniu go do potencjału $U = 50\,\mathrm{kV}$ nie nastąpiło wyładowanie do atmosfery? Powietrze ulega przebiciu (jonizacji) w polu elektrycznym $E_0 = 3\,\mathrm{kV/mm}$.
Rozwiązanie na str. 8

Przygotował Łukasz BOŻYK

W kolejnych trzech zadaniach przez *n-turniej* będziemy rozumieli układ rozgrywek, w którym każda para spośród n zawodników ($n \geqslant 2$) rozegrała dokładnie jeden mecz i nie było remisów.

Powiemy, że zawodnik A jest *mistrzem*, jeśli dla każdego zawodnika C, z którym A przegrał, istnieje zawodnik B, który przegrał z A i wygrał z C. Innymi słowy, mistrz to zawodnik, który wygrał z każdym innym bezpośrednio lub pośrednio.

M 1594. Wykazać, że w każdym n-turnieju istnieje co najmniej jeden mistrz.
Rozwiązanie na str. 3

M 1595. a) Wykazać, że nie ma takiego n-turnieju, w którym jest dokładnie dwóch mistrzów.
b) Wykazać, że dla każdego $n \geqslant 3$ istnieje n-turniej, w którym jest dokładnie trzech mistrzów.
Rozwiązanie na str. 3

M 1596. Wyznaczyć wszystkie $n \geqslant 3$, dla których istnieje n-turniej, w którym każdy jest mistrzem.
Rozwiązanie na str. 2