



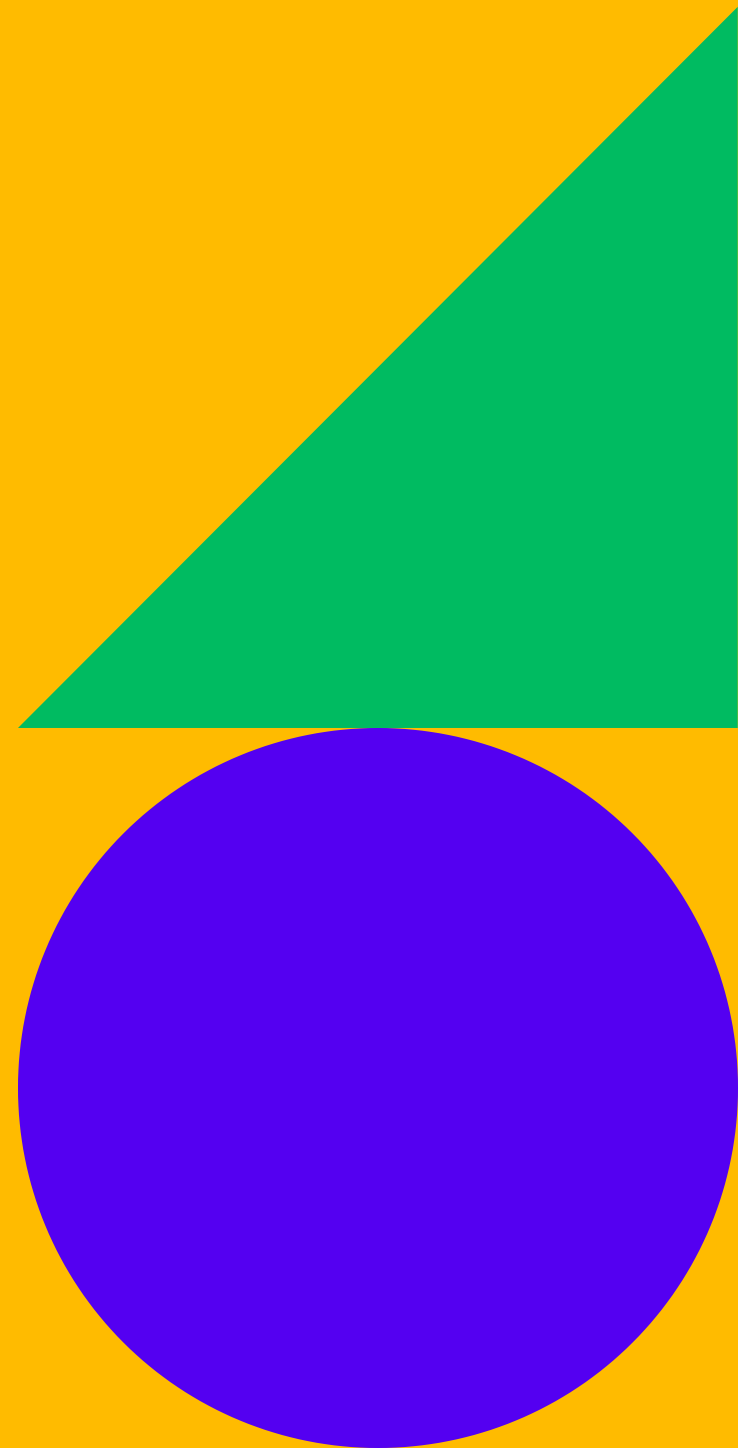
TensorflowJS

Clasificación de Imágenes en la web

Gonzalo Mardones | gonzalo-a@hotmail.com
Ago/2020

TensorflowJS nos permite desarrollar modelos de IA muy rápido, con gran performance y versatilidad en la web

Mobilenet es un modelo pre-entrenado con muchas imágenes, lo utilizaremos como base para detectar elementos de nuestro interés



Por medio de Transferencia de aprendizaje usaremos mobilenet en conjunto con un clasificador KNN

PD: No vendo los anteojos

Pared

Persona

OK

Resultado Modelo Mobilnet

Predicción: neck brace
Probabilidad:0.488629013299942

Resultado Modelo Mobilnet + Algoritmo de clasificación KNN

Inexperto (Sin experiencia)

Webcam



El modelo se carga y la webcam inicia, mobilenet reconoce cualquier elemento, nada cool (ver resultado), aún no le hemos dicho a nuestro clasificador de que es la imagen que tenemos en pantalla

ESTAMOS **CATEGORIZANDO** EL TIPO DE IMAGEN, PARA ESTE EJEMPLO TENEMOS LOS CASOS DE: PARED, PERSONA, □, TAZA O TELÉFONO

Resultado Modelo Mobilnet

Predicción: neck brace
Probabilidad: 0.49278196692466736

Resultado Modelo Mobilnet + Algoritmo de clasificación KNN

Predicción: Persona
Probabilidad: 1

Webcam



El almacenamiento se realiza en LocalStorage

La carga se realiza de LocalStorage

Almacenar configuración

Recuperar configuración

El modelo mobilenet + KNN identifica una persona en la webcam, pero ¿Qué hicimos? Para cada frame reconocido le decimos que tipo de elemento es (para esto existen unos botones con los tipos disponibles a reconocer)

Resultado Modelo Mobilnet

Predicción: sweatshirt
Probabilidad:0.1072123572230339

Resultado Modelo Mobilnet + Algoritmo de clasificación KNN

Predicción: OK
Probabilidad: 1

Webcam



Le hemos dado imágenes al clasificador con diversos contextos (ángulos, cambios de iluminación, distancia a la webcam, etc)

MOBILENET RECONOCE ESTA IMAGEN COMO SWEATSHIRT CON UN 10% DE PRECISIÓN, EN CAMBIO NUESTRA VARIANTE NOS INDICA QUE ☐ POSEE UNA PRECISIÓN DEL 100% (NO PROFUNDIZARÉ PARA ESTE CASO DE LA PRECISIÓN PORQUE ES UN CASO EJEMPLO Y SOLO DEMOSTRAMOS EL FUNCIONAMIENTO DEL MÉTODO, PERO LO REVISAREMOS EN OTRO EJERCICIO)

Application

Manifest

Service Workers

Clear storage

Storage

Local Storage

http://127.0.0.1:8080

Session Storage

IndexedDB

Web SQL

Cookies

Cache

Cache Storage

Application Cache

Background Services

Background Fetch

Background Sync

Notifications

Payment Handler

Periodic Background Sync

Push Messaging

Frames

top

Filter

Key

knnClassifier

Value

[[{"1", [0, 0, 0.0045760...]

▼ [{"1", ...}, ...]

0: [{"1", ...}]

1: [{"2", [0.00027717428747564554, 0.04663221910595894, ...]}]

2: [{"3", [0.028292866423726082, 0.0258171483874321, 0. ...]}]

3: [{"4", [0.03666067123413086, 0.04795932397246361, 0. ...]}]

4: [{"5", [0.014429749920964241, 0.06343376636505127, 0. ...]}]

ALMACENAMOS
NUESTRO MODELO
EN
LOCALSTORAGE

Esto nos permite además de almacenar el modelo,
recuperarlo en otro momento



Muchas gracias

Gonzalo Mardones | gonzalo-a@hotmail.com
Ago/2020