

Ingeniería de Prompts

Gonzalo Cossio Escobar

November 10, 2023

1 Introducción

El deporte siempre ha sido un medio de entretenimiento tanto para quienes juegan como para quienes lo observan, como jugadores y entrenadores existe la necesidad de poder cuantificar el número de acciones realizadas por cada uno de los jugadores, incluso poder llegar al punto de calificar la calidad de cada acción realizada.

El propósito de este trabajo es dar un inicio a la creación de una aplicación de procesamiento de imágenes la cual pueda tomar un vídeo de un partido de rugby, identificar los jugadores de la cancha y poder generar las estadísticas necesarias que pueda necesitar un entrenador para medir el rendimiento del equipo por partido. Claramente el alcance de este proyecto es mucho mas amplio de lo que comprende el curso, por lo que para la entrega de este trabajo se espera poder desarrollar un algoritmo el cual este en la capacidad de procesar un vídeo corto de rugby e identificar al menos la pelota de rugby.

Se tomó como referencia los últimos minutos del partido entre Sudafricana contra Irlanda del mundial de rugby del 2023, el propósito es poder identificar el balón de rugby y seguirlo a lo largo del vídeo.

2 Algoritmos y Herramientas

Como punto de partida se tomo el trabajo de Carion et al. (2023) un modelo de detección de objetos el cual utiliza las llamadas consultas de objetos para detectar objetos en una imagen. Cada consulta de objeto busca un objeto particular en la imagen, el ejemplo tratado en Hugging Face hace uso del benchmark COCO el cual permite identificar un catalogo amplio de objetos en una imagen, a continuación, se muestra el resultado obtenido a partir de una imagen extraída de Google de Faf de Klerk.



De las imágenes anteriores se observa que el algoritmo logra catalogar correctamente el contenido de la imagen, identificando dos personas y un balón.

Continuando con el ejercicio, se tomo un fragmento del partido entre Sudafrica contra Irlanda del mundial de rugby del 2023, mas específicamente el fragmento que consta desde el minuto 72:23 hasta el minuto 73:25. Ya que el algoritmo utilizado en el anterior ejemplo identifica objetos en una imagen, es necesario dividir el fragmento de vídeo en fotogramas, por lo que se toma como referencia el trabajo de Santos (2023) en donde utiliza la librería OpenCV para dividir un vídeo en fotogramas de las cuales para el ejercicio sobre el que se basa este trabajo se obtuvieron 1600 fotografías. Esta es la razón por la cual se eligió un pequeño fragmento de poco mas de un minuto.

En este momento se tienen todos los implementos para poder hacerle seguimiento al balón de rugby a lo largo del vídeo de poco mas de un minuto seleccionado, con ayuda de Chat GPT, se solicita el código el cual va a permitir abrir la carpeta donde se encuentran los fotogramas, procesar cada uno de estos e identificar el balón y por último guardar una nueva imagen en otra carpeta con un recuadro al rededor del balón.

G

Una vez ejecuto el modelo del que estabamos hablando, que código debo utilizar para que el algoritmo me encierre en un recuadro el objeto del tipo que quiero y lo guarde como imagen



Para utilizar el modelo DETR y realizar la detección de objetos en una imagen, puedes seguir el siguiente código como punto de partida. Asegúrate de tener instaladas las bibliotecas necesarias, incluyendo "transformers", "torch", y "PIL" (Python Imaging Library). Puedes instalarlas con:

```
bash
pip install transformers torch Pillow
```

Aquí hay un ejemplo básico de cómo cargar un modelo preentrenado DETR y utilizarlo para realizar la detección de objetos en una imagen:

```
python
from transformers import DetrImageProcessor, DetrForObjectDetection
from PIL import Image, ImageDraw
import torch

# Cargar el modelo preentrenado y el procesador de imágenes
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")
processor = DetrImageProcessor.from_pretrained("facebook/detr-resnet-50")

# Cargar la imagen a la que quieres realizar la detección
image_path = "path/to/your/image.jpg"
image = Image.open(image_path)

# Procesar la imagen y realizar la detección
inputs = processor(images=image, return_tensors="pt")
outputs = model(**inputs)

# Recuperar las predicciones
pred_boxes = outputs.pred_boxes[0]
pred_scores = outputs.scores[0]
pred_classes = outputs.pred_classes[0]

# Crear una nueva imagen con las cajas delimitadoras dibujadas
draw = ImageDraw.Draw(image)

# Umbral para las detecciones
threshold = 0.7

# Dibujar las cajas delimitadoras en la imagen original
for score, box, label in zip(pred_scores, pred_boxes, pred_classes):
    if score > threshold:
        box = [round(coord.item(), 2) for coord in box.tolist()]
        draw.rectangle(box, outline="red", width=3)
        draw.text((box[0], box[1]), f"Clase {label.item()} (score: {score:.2F})", fill="red")

# Guardar la nueva imagen con las cajas delimitadoras
image_with_boxes_path = "path/to/your/image_with_boxes.jpg"
image.save(image_with_boxes_path)

# Mostrar la imagen con cajas delimitadoras
image.show()
```

Asegúrate de reemplazar "path/to/your/image.jpg" con la ruta de tu imagen de entrada y "path/to/your/image_with_boxes.jpg" con la ruta donde deseas guardar la imagen resultante con las cajas delimitadoras.

Este código utiliza el modelo entrenado para realizar detecciones en la imagen y dibuja cajas delimitadoras alrededor de los objetos detectados. Ajusta el umbral ("threshold") según tus necesidades para controlar la confianza de las detecciones que deseas visualizar.

Regenerate



Por favor aplica el ejemplo anterior para aplicárselo a una serie de imágenes guardadas en una carpeta en mi computador



Claro, puedes utilizar un bucle para aplicar el ejemplo anterior a una serie de imágenes guardadas en una carpeta. Aquí hay un ejemplo de cómo hacerlo en Python:

```
python Copy code

import os

# Ruta de la carpeta que contiene las imágenes
folder_path = "path/to/your/image_folder"

# Ruta donde guardar las imágenes con cajas delimitadoras
output_folder = "path/to/your/output_folder"
os.makedirs(output_folder, exist_ok=True)

# Listar todas las imágenes en la carpeta
image_files = [f for f in os.listdir(folder_path) if f.endswith(('jpg', 'png'))]

# Cargar el modelo preentrenado y el procesador de imágenes
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")
processor = DetrImageProcessor.from_pretrained("facebook/detr-resnet-50")

# Umbral para las detecciones
threshold = 0.7

# Iterar sobre las imágenes
for image_file in image_files:
    # Ruta completa de la imagen
    image_path = os.path.join(folder_path, image_file)

    # Cargar la imagen
    image = Image.open(image_path)

    # Procesar la imagen y realizar la detección
    inputs = processor(images=image, return_tensors="pt")
    outputs = model(**inputs)

    # Recuperar las predicciones
    pred_boxes = outputs.pred_boxes[0]
    pred_scores = outputs.scores[0]
    pred_classes = outputs.pred_classes[0]

    # Crear una nueva imagen con las cajas delimitadoras dibujadas
    draw = ImageDraw.Draw(image)

    # Dibujar las cajas delimitadoras en la imagen original
    for score, box, label in zip(pred_scores, pred_boxes, pred_classes):
        if score > threshold:
            box = [round(coord.item(), 2) for coord in box.tolist()]
            draw.rectangle(box, outline="red", width=3)
            draw.text((box[0], box[1]), f"Clase {label.item()} ({score:.2f})")

    # Guardar la nueva imagen con las cajas delimitadoras
    output_path = os.path.join(output_folder, f"boxed_{image_file}")
    image.save(output_path)

print("Proceso completado.")
```

Asegúrate de reemplazar "path/to/your/image_folder" con la ruta de la carpeta que contiene tus imágenes y "path/to/your/output_folder" con la ruta donde deseas guardar las imágenes resultantes con las cajas delimitadoras. Este código creará una imagen con cajas delimitadoras para cada imagen en la carpeta de entrada y las guardará en la carpeta de salida.

Para terminar, se utiliza una vez mas la ayuda de Chat GPT para poder obtener un código el cual permita unir todas las fotos con los recuadros del balón de rugby en un solo vídeo.



En mi código utilicé la librería de python opencv/CV2 para dividir un video en fotogramas, con esta misma librería puedo unir las fotos creadas con el código que me acabas de dar en un solo video?



Sí, puedes utilizar la biblioteca OpenCV ('cv2') para unir las imágenes generadas en un solo video. Aquí hay un ejemplo básico de cómo hacerlo:

```
python
import cv2
import os

# Ruta de la carpeta que contiene las imágenes con cajas delimitadoras
input_folder = "path/to/your/output_folder"

# Ruta donde guardar el video resultante
output_video_path = "path/to/your/output_video.mp4"

# Obtener la lista de archivos de imágenes en la carpeta
image_files = [f for f in os.listdir(input_folder) if f.endswith(('.jpg', '.png'))]

# Ordenar los archivos para garantizar el orden correcto
image_files.sort()

# Leer la primera imagen para obtener la información de tamaño
first_image = cv2.imread(os.path.join(input_folder, image_files[0]))
height, width, _ = first_image.shape

# Configurar el objeto VideoWriter
fourcc = cv2.VideoWriter_fourcc('mp4v') # Puedes cambiar el códec según tu necesidad
video_writer = cv2.VideoWriter(output_video_path, fourcc, 1, (width, height))

# Iterar sobre las imágenes y agregarlas al video
for image_file in image_files:
    image_path = os.path.join(input_folder, image_file)
    frame = cv2.imread(image_path)
    video_writer.write(frame)

# Liberar el objeto VideoWriter
video_writer.release()

print(f"El video ha sido creado y guardado en: {output_video_path}")
```

Este código:

1. Lee las imágenes en la carpeta 'input_folder'.
2. Ordena las imágenes para asegurarse de que estén en el orden correcto.
3. Crea un objeto 'VideoWriter' de OpenCV para escribir el video en la ruta especificada en 'output_video_path'.
4. Itera sobre las imágenes y agrega cada imagen al video.
5. Guarda el video y libera el objeto 'VideoWriter'.

Asegúrate de reemplazar "path/to/your/output_folder" con la ruta de la carpeta que contiene las imágenes con cajas delimitadoras y "path/to/your/output_video.mp4" con la ruta donde deseas guardar el video resultante.

Ten en cuenta que este código asume que todas las imágenes tienen el mismo tamaño. Ajusta el código según tus necesidades específicas y verifica que las imágenes tengan el mismo tamaño antes de usar este enfoque.

Hasta este punto se logra cumplir con el objetivo principal del proyecto planteado, aún así con la expectativa de continuar trabajando en este proyecto, se realizan una serie de preguntas adicionales a Chat GPT con el fin de conocer cual es el proceso de entrenamiento que se debe realizar para poder crear un algoritmo el cual pueda empezar a identificar jugadores y poderlos seguir a lo largo de un vídeo.



Hola, estoy utilizando la siguiente librería de python para poder crear un algoritmo de detección de objetos en imágenes "from transformers import DetrImageProcessor, DetrForObjectDetection" me puedes ayudar a indicarme que código debo utilizar para poder entrenar el algoritmo con el objeto que quiero que identifique



Entrenar un modelo como el DETR (DEtection TRansformer) para la detección de objetos requiere varios pasos y la disponibilidad de datos de entrenamiento adecuados. Además, la librería "transformers" de Hugging Face proporciona una interfaz para utilizar modelos preentrenados, pero no proporciona una funcionalidad completa para entrenar modelos desde cero. Esto se debe a que el entrenamiento desde cero generalmente requiere recursos de cómputo intensivos y grandes conjuntos de datos.

Sin embargo, puedo proporcionarte un ejemplo de cómo puedes ajustar (fine-tune) un modelo preentrenado DETR para la detección de objetos en un conjunto de datos personalizado. Asegúrate de tener instaladas las bibliotecas necesarias, incluyendo "transformers" y "torch". Puedes instalarlas con:

```
bash
pip install transformers torch
```

A continuación, te doy un ejemplo general utilizando la librería "transformers". Ten en cuenta que este ejemplo asume que tienes un conjunto de datos etiquetado en el formato adecuado.

```
python
from transformers import DetrImageProcessor, DetrForObjectDetection
from PIL import Image
import torch

# Cargar el modelo preentrenado y el procesador de imágenes
model = DetrForObjectDetection.from_pretrained("facebook/detr-resnet-50")
processor = DetrImageProcessor.from_pretrained("facebook/detr-resnet-50")

# Configurar parámetros de entrenamiento
num_epochs = 5
learning_rate = 1e-5

# Tu conjunto de datos debe contener imágenes y sus respectivas etiquetas
# Aquí asumimos que tienes una lista de rutas de imagen y una lista de etiquetas
# Asegúrate de tener un formato adecuado para tu conjunto de datos

image_paths = ["path/to/image1.jpg", "path/to/image2.jpg", ...]
labels = [
    [{"label": "objeto1", "coordinates": [xmin, ymin, xmax, ymax]}, ...], #
    [{"label": "objeto2", "coordinates": [xmin, ymin, xmax, ymax]}, ...], #
    # ...
]

# Procesar los datos y prepararlos para el modelo
inputs = processor(images=image_paths, labels=labels, return_tensors="pt")

# Ajustar el modelo (fine-tuning)
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

for epoch in range(num_epochs):
    outputs = model(**inputs)
    loss = outputs.loss

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    print(f"Epoch {epoch + 1}/{num_epochs}, Loss: {loss.item()}")

# Guardar el modelo entrenado
```


Este es un ejemplo básico y puedes necesitar ajustar muchos detalles según tus necesidades específicas. Además, ten en cuenta que entrenar modelos de detección de objetos con éxito generalmente requiere un conjunto de datos grande y diverso.

Recuerda revisar la documentación de la librería `"transformers"` y el modelo específico que estás utilizando para obtener información más detallada sobre cómo ajustar el modelo y los requisitos de los datos de entrenamiento.

G

Cuando dices "A continuación, te doy un ejemplo general utilizando la librería transformers. Ten en cuenta que este ejemplo asume que tienes un conjunto de datos etiquetado en el formato adecuado." como debo tener el conjunto de datos con el etiquetado en el formato adecuado?



Para entrenar un modelo de detección de objetos como DETR, generalmente necesitas tener un conjunto de datos etiquetado en un formato específico. La forma más común de organizar las anotaciones es mediante el uso de archivos COCO (Common Objects in Context), que es un formato estándar para conjuntos de datos de detección de objetos. Este formato incluye información sobre las imágenes, las clases de objetos y las coordenadas de las cajas delimitadoras.

Aquí hay un ejemplo simplificado de cómo podrías organizar tus datos en el formato COCO:

1. Estructura del conjunto de datos:

```
bash
dataset/
├── annotations/
│   └── instances_train.json # Anotaciones de entrenamiento
├── images/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
└── ...
```

2. Archivo de anotaciones ('instances_train.json'):

El archivo 'instances_train.json' contiene la información de anotaciones para el conjunto de entrenamiento. Aquí hay un ejemplo simplificado:

```
json
{
  "info": {},
  "licenses": [],
  "images": [
    { "id": 1, "file_name": "image1.jpg", "height": 800, "width": 1200 },
    { "id": 2, "file_name": "image2.jpg", "height": 600, "width": 900 },
    // ... más imágenes ...
  ],
  "annotations": [
    { "id": 1, "image_id": 1, "category_id": 1, "bbox": [x, y, width, height] },
    { "id": 2, "image_id": 1, "category_id": 2, "bbox": [x, y, width, height] },
    { "id": 3, "image_id": 2, "category_id": 1, "bbox": [x, y, width, height] },
    // ... más anotaciones ...
  ],
  "categories": [
    { "id": 1, "name": "objeto1", "supercategory": "none" },
    { "id": 2, "name": "objeto2", "supercategory": "none" },
    // ... más categorías ...
  ]
}
```

- * `"images"`: Contiene información sobre las imágenes, incluidos sus identificadores, nombres de archivo, alturas y anchuras.
- * `"annotations"`: Contiene información sobre las anotaciones, incluidos sus identificadores, identificadores de imagen, identificadores de categoría y coordenadas de caja delimitadora (`"bbox"`).
- * `"categories"`: Contiene información sobre las categorías de objetos, incluidos sus identificadores y nombres.

3. Proceso de Anotación:

10

Puedes utilizar herramientas de anotación como VGG Image Annotator (VIA), Labelbox, COCO Annotator, etc., para etiquetar tus imágenes y generar archivos de anotaciones en formato COCO.

4. Proceso de Entrenamiento:

Después de organizar tus datos en este formato, puedes utilizar el código que te proporcioné anteriormente para cargar y entrenar el modelo DETR.

Regenerate

A partir de las respuestas suministradas por Chat GPT se comprende mejor el proceso de entrenamiento del algoritmo empleado en este proyecto, es evidente que es necesario comenzar por conseguir una base de buena calidad de imágenes para catalogar los elementos en las fotografías, la herramienta VIA Image Annotator permite realizar ese proceso.

3 Resultados y Conclusiones

Para concluir, gracias a este proyecto se pudo profundizar en el concepto de ingeniería de prompts, conociendo a mayor profundidad el funcionamiento de algoritmos de procesamiento de imágenes, incluyendo su fase de entrenamiento y adicionalmente se hizo uso de la herramienta Chat GPT así como la búsqueda en Google para poder mejorar el código que se estaba construyendo.

Este trabajo será continuado a modo de proyecto personal, con el fin de poder tener una herramienta que pueda generar estadísticas a partir de un vídeo de un partido de rugby.

References

- Carion, Nicolas et al. (2023). *End-to-End Object Detection with Transformers*. URL: <https://huggingface.co/facebook/detr-resnet-50>.
- Santos, Nuno (2023). *Python OpenCV: Splitting video into frames*. URL: <https://techtutorialsx.com/2021/04/29/python-opencv-splitting-video-frames/>.