COLOMBIA POTENCIA DE LA VIDA

TIC

Actividad 4

Problemas en visión computacional

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Problemas en visión computacional

La visión computacional es una rama de la inteligencia artificial que se enfoca en la interpretación automática de imágenes y videos por parte de los ordenadores. Algunas de las aplicaciones más comunes en visión computacional incluyen la clasificación de objetos, la detección de objetos y la segmentación de imágenes.

Problemas de Clasificación de Objetos

Definición del problema de clasificación de objetos en imágenes

El problema de clasificación de objetos en imágenes es un campo de la visión computacional que se centra en asignar una etiqueta o categoría a una imagen que contiene uno o varios objetos específicos. En este contexto, un objeto puede referirse a cualquier entidad o elemento visual presente en la imagen, como personas, animales, vehículos, edificios, objetos inanimados, etc. El objetivo es desarrollar algoritmos y modelos capaces de reconocer y distinguir diferentes clases de objetos en imágenes de entrada.

Problemas de Clasificación de Objetos

La clasificación de objetos es un problema de aprendizaje supervisado, donde se entrenan modelos utilizando un conjunto de datos etiquetados previamente. Este conjunto de datos consiste en imágenes de entrenamiento, cada una asociada con una etiqueta que indica la clase del objeto presente en la imagen. Durante el entrenamiento, el modelo aprende a asociar características visuales específicas de las imágenes con las clases de objetos correspondientes.

Para abordar el problema de clasificación de objetos, se utilizan diversas técnicas y algoritmos, desde enfoques tradicionales de aprendizaje automático hasta redes neuronales convolucionales (CNN) y otras arquitecturas de redes neuronales profundas. Las CNN, en particular, han demostrado ser altamente efectivas en la clasificación de objetos debido a su capacidad para aprender características visuales jerárquicas y su invariancia a la translación, escala y rotación.

El proceso de clasificación de objetos generalmente implica las siguientes etapas

Preprocesamiento de datos:
Las imágenes de entrenamiento se preprocesan para garantizar la consistencia en términos de tamaño, formato y calidad. Esto puede incluir operaciones como la normalización de píxeles, el redimensionamiento de imágenes y la corrección de la iluminación.

Extracción de características:
Se extraen características relevantes de las imágenes que ayudan al modelo a distinguir entre diferentes clases de objetos. Estas características pueden incluir bordes, texturas, colores y formas.

El proceso de clasificación de objetos generalmente implica las siguientes etapas

Entrenamiento del modelo:
Se entrena un modelo de clasificación utilizando el conjunto de datos etiquetados. Durante el entrenamiento, el modelo ajusta sus parámetros para minimizar la diferencia entre las etiquetas predichas y las etiquetas reales de las imágenes de entrenamiento.

Evaluación del modelo:
Finalmente, el modelo entrenado se evalúa utilizando un conjunto de datos de prueba independiente para determinar su rendimiento en la clasificación de objetos en imágenes nuevas y no vistas. Las métricas comunes de evaluación incluyen la precisión, la precisión, la exhaustividad y la puntuación F1.

El proceso de clasificación de objetos generalmente implica las siguientes etapas

El problema de clasificación de objetos en imágenes es fundamental para numerosas aplicaciones en visión computacional, como reconocimiento de objetos, sistemas de vigilancia, diagnóstico médico asistido por computadora, vehículos autónomos y mucho más. Un rendimiento preciso y robusto en la clasificación de objetos es esencial para el éxito de estas aplicaciones en una amplia variedad de dominios.

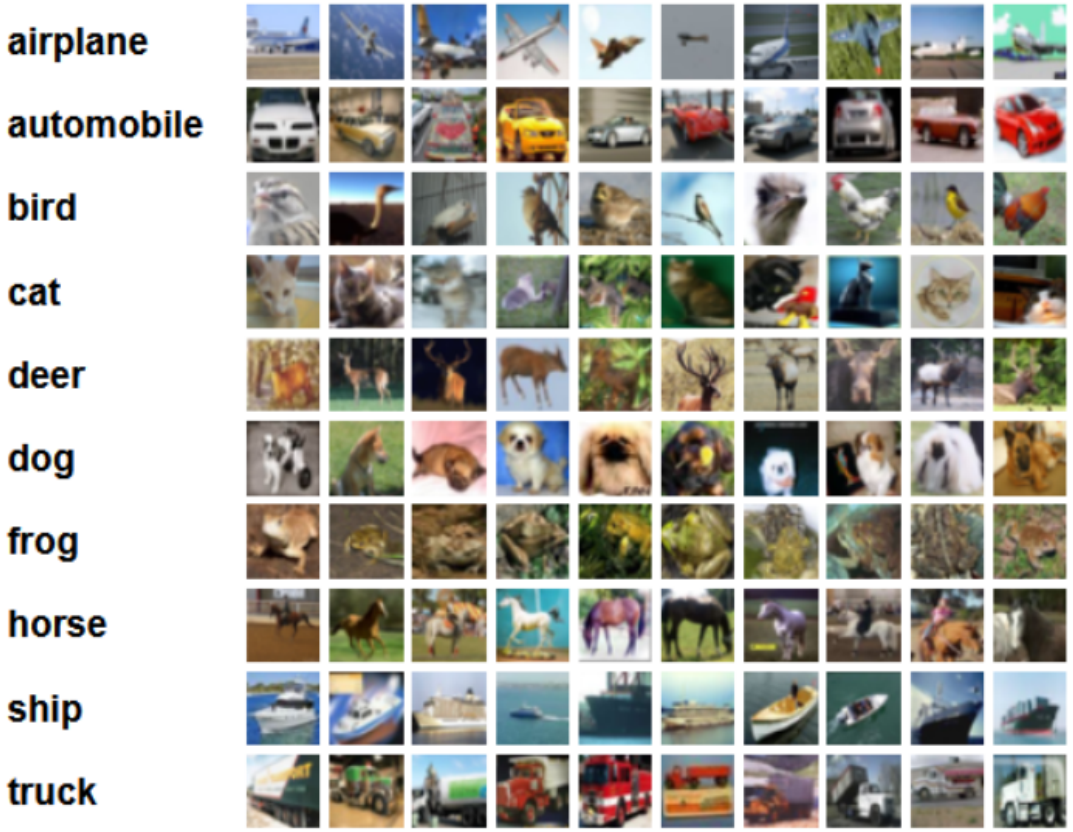
Introducción a los conjuntos de datos

En el campo de la clasificación de objetos en imágenes, los conjuntos de datos juegan un papel fundamental al proporcionar imágenes etiquetadas que permiten entrenar y evaluar modelos de aprendizaje automático. Estos conjuntos de datos varían en tamaño, complejidad y dominio de aplicación, y cada uno tiene sus propias características y desafíos.

Introducción a los conjuntos de datos

Algunos de los conjuntos de datos más populares utilizados en clasificación de objetos:

CIFAR-10 y CIFAR-100:
ver documentación oficial en https://www.cs.toronto.edu/~kriz/cifar.html
Estos son dos conjuntos de datos ampliamente utilizados que contienen imágenes pequeñas en color de 32x32 píxeles. CIFAR-10 consta de 10 clases de objetos diferentes, cada una con 6,000 imágenes de entrenamiento y 1,000 imágenes de prueba.



Introducción a los conjuntos de datos

CIFAR-100 es similar pero contiene 100 clases de objetos agrupadas en 20 superclases. Estos conjuntos de datos son útiles para la evaluación de algoritmos de clasificación de objetos en imágenes de baja resolución.

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Introducción a los conjuntos de datos

MNIST:

ver documentación oficial
http://yann.lecun.com/exdb/mnist/
Aunque MNIST se utiliza principalmente para reconocimiento de dígitos escritos a mano, también se ha utilizado para problemas de clasificación de objetos debido a su simplicidad y facilidad de acceso. Contiene 70,000 imágenes de dígitos escritos a mano en escala de grises de 28x28 píxeles, divididas en conjuntos de entrenamiento y prueba.

Introducción a los conjuntos de datos

ImageNet:

ver documentacion oficial https://www.image-net.org/

ImageNet es uno de los conjuntos de datos más grandes y diversos disponibles para la clasificación de objetos en imágenes.

Contiene millones de imágenes etiquetadas que abarcan miles de clases de objetos diferentes. ImageNet se ha utilizado para entrenar algunos de los modelos de redes neuronales convolucionales (CNN) más avanzados y ha sido un punto de referencia para el desarrollo de técnicas de clasificación de objetos.

Introducción a los conjuntos de datos

Estos son solo algunos ejemplos de los conjuntos de datos utilizados en clasificación de objetos en imágenes. La elección del conjunto de datos adecuado depende de factores como la complejidad del problema, la disponibilidad de datos y el dominio de aplicación específico. Es importante seleccionar un conjunto de datos que sea representativo de las imágenes con las que se encontrará el modelo en el entorno de implementación real y que proporcione una cobertura adecuada de las clases de objetos de interés.

Problema de Detección de Objetos

Descripción del problema de detección de objetos y sus aplicaciones

La detección de objetos es una tarea fundamental en el campo de la visión por computadora que implica identificar y localizar la presencia de objetos específicos dentro de una imagen o video. A diferencia de la clasificación de objetos, que se centra en asignar una etiqueta a una imagen en su conjunto, la detección de objetos se ocupa de encontrar la ubicación precisa de múltiples objetos individuales dentro de una escena.

Componentes de la detección de objetos

Identificación de objetos:
El primer paso en la detección de objetos es identificar las clases de objetos presentes en la imagen. Esto implica reconocer qué tipos de objetos están presentes en la escena, como personas, automóviles, bicicletas, animales, señales de tráfico, entre otros.

Localización de objetos:
Una vez que se han identificado las clases de objetos, se procede a determinar la ubicación espacial precisa de cada objeto dentro de la imagen. Esto generalmente se logra definiendo cajas delimitadoras (bounding boxes) que rodean a cada objeto detectado, especificando sus coordenadas en términos de píxeles o proporciones relativas en la imagen.

Componentes de la detección de objetos

Clasificación y etiquetado:
Además de localizar los objetos, es necesario clasificarlos correctamente asignando una etiqueta o categoría a cada uno. Por ejemplo, si se detecta un automóvil, se debe etiquetar como "automóvil" para identificar su clase.

Técnicas utilizadas en la detección de objetos

Feature-based detection:
Se basa en la extracción de características relevantes de la imagen, como bordes, esquinas o regiones de interés, y la comparación con modelos predefinidos para identificar objetos.

Deep learning approaches:
Las redes neuronales convolucionales (CNN) han revolucionado la detección de objetos al permitir el aprendizaje de características directamente de los datos. Modelos como Faster R-CNN, YOLO (You Only Look Once) y SSD (Single Shot MultiBox Detector) son ampliamente utilizados en este campo.

Aplicaciones de la detección de objetos

Conducción autónoma:
En vehículos autónomos, la detección de objetos es esencial para identificar y evitar obstáculos, reconocer señales de tráfico y peatones, y tomar decisiones seguras durante la conducción.

Seguridad y vigilancia:
En sistemas de vigilancia, la detección de objetos se utiliza para identificar intrusiones, reconocer caras, seguir objetos en movimiento y alertar sobre actividades sospechosas.

Robótica:
Los robots utilizan la detección de objetos para interactuar con su entorno, recoger objetos, navegar por entornos desconocidos y realizar tareas específicas en aplicaciones industriales, de servicio y domésticas.

Aplicaciones de la detección de objetos

Medicina:
En aplicaciones médicas, la detección de objetos se emplea en el análisis de imágenes médicas para identificar estructuras anatómicas, detectar anomalías y ayudar en el diagnóstico y tratamiento de enfermedades.

Realidad aumentada:
En la realidad aumentada, la detección de objetos permite superponer elementos virtuales, como información contextual, gráficos o efectos visuales, sobre el mundo real en aplicaciones de juegos, entretenimiento, navegación y publicidad.

La detección de objetos es un campo en constante evolución que desempeña un papel fundamental en una amplia gama de aplicaciones y sectores industriales. Con el avance de las técnicas de aprendizaje profundo y la disponibilidad de grandes conjuntos de datos, se espera que continúe creciendo en importancia y sofisticación en el futuro.

Problema de Segmentación

Explicación del problema de segmentación de imágenes

La segmentación de imágenes es una tarea fundamental en visión por computadora que consiste en dividir una imagen en regiones o segmentos significativos y homogéneos según ciertas características, como color, textura, intensidad de píxeles o contornos. El objetivo principal de la segmentación es simplificar y/o cambiar la representación de una imagen para que sea más fácil de analizar y comprender por parte de los algoritmos de procesamiento de imágenes y los seres humanos.

Existen diferentes tipos de segmentación de imágenes, entre los cuales se destacan

Segmentación basada en umbrales:
Este enfoque implica la asignación de píxeles a diferentes segmentos según si sus valores de intensidad están por encima o por debajo de ciertos umbrales predefinidos. Es útil cuando la imagen presenta diferencias claras de intensidad entre los objetos y el fondo.

Segmentación por regiones:
En este enfoque, se agrupan los píxeles vecinos que comparten características similares, como color, textura o nivel de gris. Se utilizan algoritmos de agrupamiento, como el algoritmo de crecimiento de regiones, para dividir la imagen en segmentos homogéneos.

Existen diferentes tipos de segmentación de imágenes, entre los cuales se destacan

Segmentación por contornos:
Este enfoque se centra en detectar los límites o contornos de los objetos en la imagen. Se utilizan técnicas de detección de bordes, como el operador de Sobel o el algoritmo de Canny, para identificar las transiciones abruptas en la intensidad de los píxeles que indican los bordes de los objetos.

Segmentación semántica:
Este tipo de segmentación asigna una etiqueta semántica a cada píxel de la imagen, lo que permite identificar y distinguir diferentes clases de objetos. Por ejemplo, en una imagen de calle, la segmentación semántica puede etiquetar los píxeles como "coche", "persona", "edificio", etc.

Existen diferentes tipos de segmentación de imágenes, entre los cuales se destacan

Segmentación basada en contornos activos:
También conocidos como "contornos activos" o "serpientes", estos métodos permiten delinear manual o automáticamente los contornos de los objetos de interés en la imagen mediante curvas deformables que se ajustan a los bordes deseados.

Aplicaciones de la segmentación de imágenes

Medicina:
La segmentación se utiliza en aplicaciones médicas para detectar y delimitar órganos, tejidos y estructuras anatómicas en imágenes médicas, como resonancias magnéticas (RM) o tomografías computarizadas (TC).

Procesamiento de imágenes satelitales:
En la cartografía y la teledetección, la segmentación de imágenes se utiliza para identificar y clasificar diferentes tipos de terreno, vegetación, cuerpos de agua y otros elementos geográficos.

Aplicaciones de la segmentación de imágenes

Automatización industrial:
En la inspección de calidad y el control de procesos industriales, la segmentación se emplea para identificar defectos, clasificar productos y guiar robots en la manipulación de objetos.

Reconocimiento de objetos y seguimiento:
La segmentación de imágenes es esencial para el reconocimiento y seguimiento de objetos en aplicaciones de vigilancia, robótica, vehículos autónomos y realidad aumentada.

Restauración de imágenes:
En el campo del procesamiento de imágenes, la segmentación se utiliza para mejorar la calidad visual de las imágenes eliminando el ruido y las distorsiones.

Aplicaciones de la segmentación de imágenes

La segmentación de imágenes es un proceso fundamental en visión por computadora que desempeña un papel crucial en una variedad de aplicaciones en campos tan diversos como la medicina, la agricultura, la industria, la seguridad y la investigación científica.

Conjuntos de datos

COCO (Common Objects in Context):
COCO es un conjunto de datos ampliamente utilizado que se centra en la detección y segmentación de objetos en contextos naturales. Contiene imágenes de alta resolución con objetos en situaciones cotidianas, como personas, animales, vehículos, muebles, etc. Además de las imágenes, COCO proporciona anotaciones detalladas que incluyen etiquetas de objetos, cajas delimitadoras y máscaras de segmentación.

Open Images Dataset:
Este es otro conjunto de datos grande y diverso que contiene millones de imágenes etiquetadas con una amplia variedad de objetos. Proporciona anotaciones precisas y detalladas que abarcan múltiples clases de objetos, lo que lo hace adecuado para una variedad de aplicaciones de clasificación y detección de objetos.

Ejercicio de Reconocimiento de Rostros de gatos

Este código utiliza la biblioteca OpenCV para detectar rostros de gatos en una imagen
Utilizando el clasificador de cascada Haar para detectar rostros de gatos en una imagen.

1 Importación de bibliotecas:

Se importa la biblioteca OpenCV con el alias cv2.
Se importa la biblioteca NumPy con el alias np.

```
import cv2
import numpy as np
```

Ejercicio de Reconocimiento de Rostros de gatos

2 Carga del clasificador de cascada:

- En este paso primero debes ir al repositorio de opencv de descargar haarcascade_frontalcatface.xml en https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalcatface.xml
lo debe guardar en la misma carpeta donde tenga el codigo.
- Se carga el clasificador de cascada entrenado para detectar rostros de gatos desde el archivo XML 'haarcascade_frontalcatface.xml'. Este archivo contiene las características que el algoritmo utiliza para detectar rostros de gatos.

```
faceClassif = cv2.CascadeClassifier('haarcascade_frontalcatface.xml')
```

Ejercicio de Reconocimiento de Rostros de gatos

3 **Carga de la imagen:**

- Se carga la imagen de los gatos desde el archivo 'gatos.jpg' utilizando la función cv2.imread().
- Se convierte la imagen a escala de grises utilizando la función cv2.cvtColor().

```
# Cargar la imagen
image = cv2.imread('gatos.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Ejercicio de Reconocimiento de Rostros de gatos

4 **Detección de rostros de gatos:**

Se utiliza el método detectMultiScale() del clasificador de cascada para detectar rostros de gatos en la imagen.

Se especifican varios parámetros para el proceso de detección:

scaleFactor: Factor de escala que especifica cuánto se reduce la imagen en cada escala de la pirámide de imagen. Un valor más pequeño da como resultado una detección más precisa, pero aumenta el tiempo de cálculo.

minNeighbors: Número mínimo de vecinos que deben encontrarse para que se considere un objeto. Un valor más alto da como resultado una detección más fiable, pero puede perder objetos pequeños.

Ejercicio de Reconocimiento de Rostros de gatos

4 **Detección de rostros de gatos:**

minSize: Tamaño mínimo de los objetos a considerar.

maxSize: Tamaño máximo de los objetos a considerar.

Los resultados se almacenan en la variable faces, que contiene las coordenadas (x, y, w, h) de los cuadrados delimitadores de los rostros detectados.

```
#Detección de rostros de gatos:
faces = faceClassif.detectMultiScale(gray,
  scaleFactor=1.1,
  minNeighbors=5,
  minSize=(30,30),
  maxSize=(200,200))
print(faces)
```

Ejercicio de Reconocimiento de Rostros de gatos

5 **Dibujar rectángulos alrededor de los rostros detectados:**

Se utiliza un bucle for para recorrer cada conjunto de coordenadas en faces.

Se dibuja un rectángulo alrededor de cada rostro detectado utilizando la función cv2.rectangle(). Los parámetros de la función son:

- La imagen en la que se dibuja el rectángulo.
- Las coordenadas (x, y) de la esquina superior izquierda del rectángulo.
- Las coordenadas (x + w, y + h) de la esquina inferior derecha del rectángulo.
- El color del rectángulo en formato BGR.
- El grosor del rectángulo.

```
# Dibujar rectángulos alrededor de los rostros detectados
for (x,y,w,h) in faces:
  cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),2)
```

Ejercicio de Reconocimiento de Rostros de gatos

6 **Mostrar la imagen resultante:**

Se utiliza la función cv2.imshow() para mostrar la imagen resultante con los rectángulos alrededor de los rostros detectados.

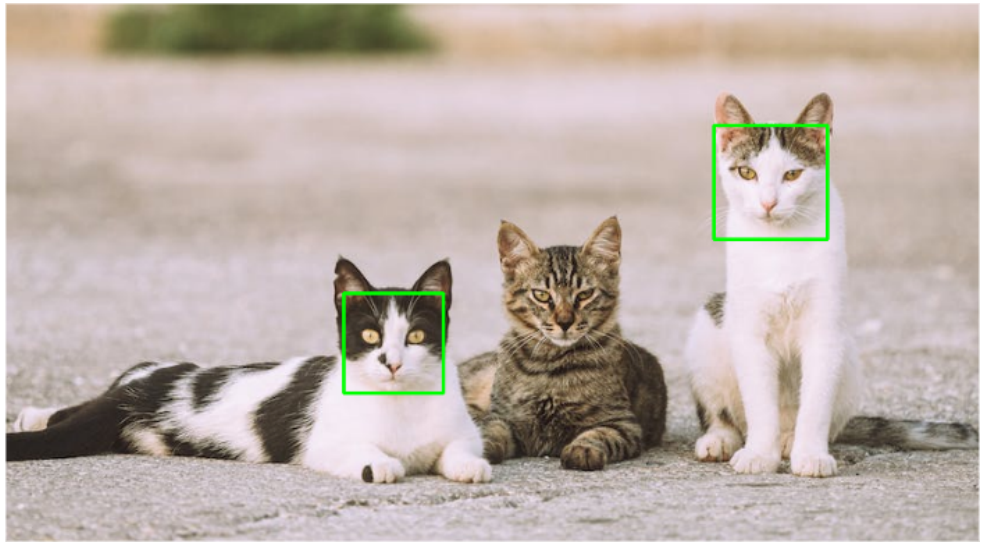
Se utiliza cv2.waitKey(0) para esperar a que el usuario presione una tecla antes de continuar.

Se utiliza cv2.destroyAllWindows() para cerrar todas las ventanas abiertas después de que se presione una tecla.

```
# Mostrar la imagen con los contornos resaltados
cv2.imshow('Objects Detected', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Ejercicio de Reconocimiento de Rostros de gatos

6 Resultado



Este código detecta rostros de gatos en una imagen utilizando un clasificador de cascada preentrenado y muestra la imagen resultante con los rectángulos delimitadores alrededor de los rostros detectados.

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira