INFOTEC

Programa de Maestría en Ciencia de Datos

Aplicación de YOLOV5 para identificar obstrucciones en espacios comunes de hogares.

Trabajo de titulación

que para optar por el grado de

Maestro en Ciencia de Datos

PRESENTA:

Héctor Ariel Aragón Oliva

Director de tesis:

México, CDMX. (Marzo) 2023

Capítulo 1

Introducción

En el presente trabajo se brinda la investigación realizada sobre la aplicación de redees neuronales convolucionales para hacer la detección de espacios donde no hay obstrucciones dentro de una casa. La idea de este desarrollo es brindar una solución que permita identificar en ciertos lugares de una casa si hay obstrucciones o no, y la funcionalidad de esto puede ser en diferentes lineas, sin embargo, esto podría ser de ayuda para brindar una base para una solución que permita a las personas con alguna discapacidad visual prevenir algún accidente al estar en algún espacio nuevo para ellos.

Es importante entender que, dada la aplicación o la propuesta del presente trabajo, es pensando en las situaciones que pueden enfrentar las personas que tienen alguna discapacidad visual, lo anterior debido a que si, por ejemplo, una persona invidente va a una casa de algún conocido con quien no ha interactuado o que simplemente no conoce algún sitio a donde va, este tipo de situaciones pueden resultar sumamente complicada para las personas que tienen alguna discapacidad visual, por lo que contar con alguna herramienta o tecnología que les permita acceder a lugares de una forma más factible y cómoda permitirá que las personas puedan tener una mejor calidad de vida y brindar seguridad a conocer lugares nuevos. Esto motiva el presente trabajo para poder brindar un enfoque de aplicación de un framework muy popular dentro del campo de visión por computadora, el cual es YOLO, en particular el trabajo se enfoca en la quinta versión, es decir, yolov5, y bajo este framework se desarrollará la solución final. Con respecto

a esta última solución, es que se va a aplicar a un set de datos, donde se encuentran etiquetados los espacios que pueden ser una obstrucción, espacios que no representan o tienen una obstrucción y objetos que pueden representar una semi obstrucción.

El trabajo busca abordar el concepto de redes neuronales para después introducir el concepto de redes neuronales convolucionales (CNN, por sus siglas en inglés) y eventualmente explicar el estado del arte de la solución YOLO (You Only Look Once), en particular, la versión 5, ya que existen diferentes versiones de este framework de object detection.

Una vez desarrollado el estado del arte y marco teorico se busca aplicar el framework que se ha mencionado anteriormente (yolov5), lo que se hizo en este aspecto es buscar bajo un proceso de hiper-parametrización los parámetros que permitan aproximar los mejores resultados, para después entrenar la arquitectura de YOLO y así poder obtener y analizar los resultados de la detección de objetos.

Capítulo 2

Estado del arte

2.0.1. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son un tipo de red neuronal diseñada específicamente para el procesamiento de datos de imágenes. A diferencia de una red neuronal estándar, donde cada neurona está conectada a todas las neuronas de la capa siguiente, una CNN utiliza capas convolucionales que aplican filtros a partes de la imagen. Esta técnica permite que la red neuronal identifique patrones y características en las imágenes a medida que se desplaza por la capa convolucional. Las capas convolucionales están seguidas por capas de agrupamiento, que reducen la dimensión de la información mientras conservan las características importantes. Finalmente, las capas de clasificación utilizan las características extraídas para clasificar la imagen en una categoría.

Cada capa convolucional utiliza filtros para extraer características de la imagen. Un filtro es una pequeña matriz de números que se desliza sobre la imagen y se multiplica elemento por elemento con los píxeles de la imagen que se superponen con el filtro. El resultado de esta multiplicación se suma y se coloca en una nueva matriz llamada "mapa de características". Los filtros se aprenden durante el entrenamiento de la red neuronal y pueden detectar bordes, texturas, formas y otros patrones en la imagen. Cuanto más profunda sea la capa convolucional, más complejas serán las característi-

cas que se pueden extraer.

Las capas de agrupamiento reducen la dimensión de los mapas de características mediante la combinación de píxeles en regiones más grandes. El agrupamiento se puede hacer de diferentes maneras, pero la técnica más común es el "max pooling", donde se selecciona el valor máximo de un área de píxeles. El agrupamiento reduce la cantidad de parámetros en la red neuronal y hace que la red sea más resistente a las pequeñas variaciones en la posición de las características en la imagen.

Finalmente, las capas de clasificación utilizan las características extraídas para determinar la clase a la que pertenece la imagen. Las capas de clasificación están formadas por neuronas que se conectan a todas las neuronas de la capa anterior y calculan una puntuación para cada clase posible. La puntuación se puede transformar en una probabilidad utilizando la función softmax, que normaliza los valores y los convierte en una distribución de probabilidad.

De tras de las redes de tipo CNN son fundamentales las operaciones convolucionales para su funcionamiento. La operación de convolución se puede expresar matemáticamente como una multiplicación de matrices entre el filtro y un parche de la imagen. La operación de agrupamiento se puede ver como una selección del valor máximo dentro de una región de la imagen. La operación de las capas completamente conectadas se puede describir matemáticamente como una multiplicación de matrices entre las activaciones de la capa anterior y una matriz de pesos. Durante el entrenamiento de la red neuronal, se utilizan técnicas de optimización, como el descenso del gradiente, para ajustar los valores de los filtros y las matrices de pesos de las capas completamente conectadas.

La operación de convolución es uno de los bloques fundamentales en las redes neuronales convolucionales (CNNs) utilizadas en el procesamiento de imágenes. La operación de convolución es esencialmente una multiplicación punto a punto entre dos matrices, donde una matriz es la imagen de entrada y la otra es una matriz de filtro conocida

como kernel. La operación de convolución se utiliza para extraer características importantes de la imagen de entrada, lo que la hace muy útil en tareas de clasificación y detección de objetos.

La operación de convolución se puede definir matemáticamente como sigue:

$$C_{i,j} = \sum_{m=1}^{k} I_{i-m+p,j-n+p} \cdot K_{m,n} \tag{2.1}$$

Donde I es la imagen de entrada, K es el kernel (también conocido como filtro), p es la cantidad de ceros que se agregan alrededor de la imagen de entrada (conocido como "padding"), k es el tamaño del kernel y C es la salida de la operación de convolución en la posición (i, j).

La operación de convolución puede ser computacionalmente costosa ya que implica realizar múltiples operaciones de multiplicación y suma para cada píxel de la imagen de entrada. Sin embargo, gracias a los avances en la GPU y la optimización de los algoritmos, las CNNs modernas pueden realizar operaciones de convolución a gran velocidad, lo que las hace muy útiles para una variedad de aplicaciones de visión por computadora.

Existen otras formas equivalentes de definirla. Una de ellas es la siguiente:

$$C_{i,j} = \sum_{m=1}^{k} \sum_{N=1}^{k} I_{i-m,j-n} \cdot K_{m+p,n+p} \tag{2.2}$$

Además existe una forma continua de definir la operación de convolución. En lugar de considerar las funciones discretas f y g, consideramos sus contrapartes continuas $f(x)$ y $g(x)$ respectivamente. La convolución continua se define como:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\tau) g(x - \tau) \partial\tau \tag{2.3}$$

Esta fórmula es similar a la anterior, pero en lugar de mover el kernel alrededor de la imagen de entrada, se mueve la imagen de entrada debajo del kernel. Esto es

equivalente a hacer la operación original, pero con un kernel girado 180 grados.

donde f y g son funciones continuas y $*$ representa la operación de convolución. Aquí, en lugar de una suma ponderada de los valores de f y g en ubicaciones discretas, se realiza una integración sobre todo el rango de f y g.

De manera similar a la definición discreta de convolución, la convolución continua también tiene propiedades importantes como la propiedad conmutativa, asociativa y distributiva. Además, la convolución continua es una operación lineal, lo que significa que se puede distribuir sobre la suma y la multiplicación por escalares.

La definición continua de convolución es útil en muchos campos de las matemáticas y la física, como en el análisis de señales y sistemas continuos. La convolución continua se utiliza para analizar la respuesta de un sistema a una entrada continua, como una señal eléctrica o una onda sonora.

Tomando en cuenta las definiciones y lo que se ha visto al rededor de la operación convolucional, la operación discreta es la que se estara ocupando, y la que es usada por lo regular en la aplicación de object detection.

Una capa de grupo (group layer) es una técnica utilizada en redes neuronales convolucionales para reducir la cantidad de parámetros y mejorar la eficiencia computacional. Según Krizhevsky et al. (2012), esta técnica consiste en dividir los canales de entrada en grupos y realizar convoluciones separadas en cada grupo.

En otras palabras, como explica Zhang et al. (2018), una capa de grupo divide la entrada en G grupos y aplica C/G filtros en cada grupo, donde C es el número de canales de entrada. Luego, las salidas de cada grupo se concatenan para formar la salida final de la capa. Esto puede reducir el número de parámetros de la red y aumentar la eficiencia computacional.

Matemáticamente, la operación de convolución en una capa de grupo se puede expresar como una multiplicación de matrices entre el filtro y una porción de la entrada correspondiente al grupo. Según Krizhevsky et al. (2012), esta operación se puede describir como:

$$y_i = \sigma(\sum_{j=1}^{C/G} W_{i,j} * x_{g(j)}) \tag{2.4}$$

donde y_i es la i-ésima característica de salida, $W_{i,j}$ es el filtro de convolución de la i-ésima característica de salida y el j-ésimo grupo de entrada, $x_{g(j)}$ es la entrada del j-ésimo grupo, C es el número total de canales de entrada, G es el número de grupos y σ es la función de activación.

Como se acaba de explicar, una capa de grupo es una técnica utilizada en redes neuronales convolucionales para reducir la cantidad de parámetros y mejorar la eficiencia computacional al realizar convoluciones separadas en grupos de canales de entrada y concatenar las salidas de cada grupo. La operación matemática de convolución en una capa de grupo se puede expresar como una multiplicación de matrices y una función de activación.

Una capa completamente conectada (fully connected layer) es una capa en una red neuronal en la que todas las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente. Según Goodfellow et al. (2016), estas capas son comúnmente utilizadas al final de una red convolucional para clasificación o regresión.

Matemáticamente, una capa completamente conectada se puede expresar como una multiplicación de matrices entre la entrada y los pesos de la capa, seguida de una función de activación. Como explica Nielsen (2015), si la entrada a la capa es un vector x de tamaño n, y la capa tiene m neuronas, entonces los pesos de la capa son una matriz W de tamaño $m \times n$, y la salida y de la capa se puede expresar como:

$$y = \sigma(Wx + b) \tag{2.5}$$

donde σ es la función de activación, b es el vector de sesgos de tamaño m, y la suma y la multiplicación se realizan elemento a elemento.

La capa completamente conectada se utiliza comúnmente en redes neuronales para clasificación. Según Goodfellow et al. (2016), después de pasar la entrada a través de varias capas convolucionales, se puede aplanar la salida de la última capa convolucional en un vector y pasar este vector a través de una o varias capas completamente conectadas. La salida final de la red es la predicción de la red para la entrada.

Por lo tanto una capa completamente conectada es una capa en una red neuronal en la que todas las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente. Matemáticamente, la salida de una capa completamente conectada se puede expresar como una multiplicación de matrices entre la entrada y los pesos de la capa, seguida de una función de activación. Esta capa se utiliza comúnmente en redes neuronales para clasificación o regresión.

Una capa convolucional (convolutional layer) es una capa en una red neuronal que utiliza la operación de convolución para extraer características de la entrada. Según Goodfellow et al. (2016), una capa convolucional consiste en un conjunto de filtros, donde cada filtro es una pequeña matriz que se desliza sobre la entrada para producir una característica en la salida.

Matemáticamente, una capa convolucional se puede expresar como la convolución de la entrada x con un conjunto de filtros W, seguida de una función de activación no lineal σ y una operación de sesgo b. Como explica Nielsen (2015), si la entrada a la capa es un tensor de tamaño $n_h \times n_w \times n_c$, donde n_h es la altura, n_w es el ancho y n_c es el número de canales de la entrada, y la capa tiene n_f filtros de tamaño $f_h \times f_w \times n_c$, entonces la salida z de la capa se puede expresar como:

$$z_{i,j,k} = \sigma(\sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{c=1}^{n_c} W_{u,v,c,k} x_{i+u-1,j+v-1,c} + b_k) \tag{2.6}$$

donde $z_{i,j,k}$ es el valor de la salida en la posición (i, j, k), $W_{u,v,c,k}$ es el valor del filtro en la posición (u, v, c, k), $x_{i+u-1,j+v-1,c}$ es el valor de la entrada en la posición $(i + u - 1, j + v - 1, c)$, y b_k es el sesgo para el k-ésimo filtro.

La capa convolucional se utiliza comúnmente en redes neuronales para procesamiento de imágenes. Según Goodfellow et al. (2016), una capa convolucional es capaz de detectar características locales en una imagen, como bordes, texturas y patrones simples, y las capas convolucionales posteriores combinan estas características para detectar características más complejas.

En pocas palabras, una capa convolucional es una capa en una red neuronal que utiliza la operación de convolución para extraer características de la entrada. Matemáticamente, la salida de una capa convolucional se puede expresar como la convolución de la entrada con un conjunto de filtros, seguida de una función de activación no lineal y una operación de sesgo. Esta capa se utiliza comúnmente en redes neuronales para procesamiento de imágenes.

Dentro del proceso de optimización de la red neuronal convolucional se hace uso tradicionalmente del algoritmo de descenso de gradiente estocástico (SGD por sus siglas en inglés), el cual se puede resumir en los siguientes pasos:

- 1. Inicialización de parámetros: Antes de comenzar el entrenamiento, se inicializan los parámetros de la red neuronal, incluyendo los pesos de los filtros y la matriz de pesos. La inicialización puede realizarse de varias maneras, pero una común es la inicialización aleatoria con una distribución gaussiana centrada en cero.
- 2. Propagación hacia adelante (Forward propagation): Durante la propagación hacia adelante, se alimenta la red neuronal con una entrada (en este caso, una imagen), y se calcula la salida de la red. La salida es una predicción de lo que

la red cree que es la respuesta correcta para la entrada dada. La salida de la red se calcula utilizando las fórmulas matemáticas para la convolución, la función de activación y la operación de pooling:

- a. Convolución: la convolución se realiza entre los pesos de los filtros y la entrada de la imagen, y se calcula utilizando la fórmula:

$$z_{i,j} = \sum_{l=1}^{n} \sum_{m=1}^{n} w_{l,m} x_{i+l-1,j+m-1} \tag{2.7}$$

  donde $z_{i,j}$ es el elemento de la salida de la convolución en la posición (i, j), $w_{l,m}$ es el peso del filtro en la posición (l, m), y $x_{i+l-1,j+m-1}$ es el píxel de la entrada de la imagen en la posición $(i + l - 1, j + m - 1)$.

- b. Función de activación: la función de activación introduce una no linealidad en la red neuronal y se aplica a la salida de la convolución. Una función común de activación es la función ReLU (unidad lineal rectificada), que se define como:

$$ReLu(z) = max(0, z) \tag{2.8}$$

- c. Operación de pooling: la operación de pooling reduce el tamaño de la salida de la convolución y se utiliza para disminuir la dimensionalidad de la imagen. Una operación de pooling común es la operación de max pooling, que toma el valor máximo de un conjunto de valores en una región de la imagen. La operación de pooling en una capa de una CNN busca reducir el tamaño de la entrada mediante el muestreo de regiones de la misma y tomando el valor máximo o promedio de cada región. El pooling se realiza en una ventana deslizante de tamaño fxF, donde f es el tamaño del filtro.

  Para la operación de pooling, se toma un filtro de tamaño f y se desliza sobre la imagen de entrada, dividiendo la imagen en regiones no superpuestas de tamaño fxF. Luego, para cada región, se calcula el valor máximo o promedio.

  En el caso del max-pooling, la operación consiste en tomar el valor máximo de cada región de la imagen de entrada. Formalmente, para una región R, la operación de max-pooling se define como:

$$maxpool(R) = \max_{i,j \in R} x_{i,j} \tag{2.9}$$

donde x es la imagen de entrada y R es la región de tamaño fxF.

En el caso del average-pooling, se toma el valor promedio de cada región de la imagen de entrada. Formalmente, para una región R, la operación de average-pooling se define como:

$$avgpool(R) = \frac{1}{f^2} \sum_{i,j \in R} x_{i,j} \tag{2.10}$$

donde x es la imagen de entrada y R es la región de tamaño fxF.

- 3. Cálculo de la función de pérdida: La función de pérdida se utiliza para medir cuán bien la red neuronal está realizando la tarea que se le ha asignado. En el caso de la clasificación, una función de pérdida común es la entropía cruzada, que se define como:

$$L(y, \hat{y}) = -\sum_{i} y_i log(\hat{y}_i) \tag{2.11}$$

donde y es el vector de etiquetas verdaderas, $\hat{y}$ es el vector de etiquetas predichas y i es el índice de las etiquetas. El objetivo del proceso de optimización es minimizar la función de pérdida.

- Propagación hacia atrás (Backward propagation): Durante la propagación hacia atrás, se calcula el gradiente de la función de pérdida con respecto a los parámetros de la red. Este gradiente se calcula utilizando la regla de la cadena, que permite calcular el gradiente de una función compuesta de varias capas aplicando sucesivamente la regla del producto a cada una de las capas. En el caso de una red neuronal convolucional, el cálculo del gradiente se realiza mediante la retropropagación de los errores (backpropagation), que consiste en calcular los errores en cada una de las capas de la red y propagarlos hacia atrás a través de la red, ajustando los parámetros en cada capa para minimizar la función de pérdida.

  La propagación hacia atrás (backward propagation) en las redes convolucionales (CNN) es un proceso clave para ajustar los parámetros de la red y reducir la

función de pérdida durante el entrenamiento. En esta etapa, se calcula el gradiente de la función de pérdida con respecto a los parámetros de la red, utilizando la regla de la cadena.

Para entender el proceso de propagación hacia atrás en las CNN, es necesario definir la función de pérdida, que es la medida de cuánto difiere la salida de la red de la salida deseada. La función de pérdida se calcula a partir de la salida de la red y la etiqueta correspondiente de cada imagen de entrenamiento. En las CNN, la función de pérdida más comúnmente utilizada es la función de entropía cruzada (cross-entropy loss), que se define como:

$$L(y, \hat{y}) = \sum_i y_i log(\hat{y}_i) \tag{2.12}$$

donde y es la etiqueta real de la imagen, $\hat{y}$ es la salida de la red para esa imagen y i es el índice de clase.

Para actualizar los pesos y sesgos de la red durante el entrenamiento, se necesita calcular la derivada parcial de la función de pérdida con respecto a cada peso y sesgo en la red. Este cálculo se realiza mediante la regla de la cadena, que implica una propagación hacia atrás del error a través de la red.

Para ilustrar este proceso, consideremos una capa convolucional de la red con una entrada x, una salida y y una función de activación no lineal $f(\cdot)$. La operación de convolución se define como:

$$y_{i,j,k} = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{l=1}^{L} w_{m,n,l,k} x_{i+m-1,j+n-1,l} + b_k \tag{2.13}$$

donde M, N y L son las dimensiones de los filtros, $w_{m,n,l,k}$ son los pesos de los filtros y b_k es el sesgo para la k-ésima salida. La función de activación se aplica a cada elemento de la salida y para producir la salida final $a = f(y)$.

Durante la propagación hacia atrás, se necesita calcular la derivada parcial de la función de pérdida con respecto a cada peso y sesgo en la red. La derivada parcial de la función de pérdida con respecto a la salida de la capa, $\frac{\partial L}{\partial y}$, se calcula como:

$$\frac{\partial L}{\partial y_{i,j,k}} = \frac{\partial L}{\partial a_{i,j,k}} \frac{\partial a_{i,j,k}}{\partial y_{i,j,k}} \tag{2.14}$$

donde $\frac{\partial L}{\partial a_{i,j,k}}$ es la derivada parcial de la función de pérdida con respecto a la salida de la función de activación, y $\frac{\partial a_{i,j,k}}{\partial y_{i,j,k}}$ es la derivada parcial de la salida de la función de activación con respecto a la entrada de la función.

Para calcular la derivada parcial de la función de pérdida con respecto a los pesos y sesgos en una capa específica, se utiliza la regla de la cadena para propagar hacia atrás el gradiente del error a través de la red. Para una capa de convolución, la derivada parcial de la función de pérdida con respecto a los pesos se calcula mediante la convolución del tensor de entrada con el tensor de error. La derivada parcial con respecto a los sesgos se obtiene simplemente sumando el tensor de error a lo largo de los ejes de la altura y la anchura.

Para una capa de pooling, la propagación hacia atrás se realiza mediante la operación de upsample del tensor de error utilizando la máscara de la operación de pooling anterior. Luego, se realiza la propagación hacia atrás de la misma manera que en una capa de convolución.

La propagación hacia atrás se realiza en todas las capas de la red, comenzando desde la última capa hasta la primera. Durante este proceso, se actualizan los pesos y sesgos utilizando un algoritmo de optimización, como el descenso de gradiente estocástico (SGD) o Adam.

El cálculo del gradiente de la función de pérdida con respecto a los parámetros se realiza mediante el algoritmo de descenso de gradiente estocástico (Stochastic Gradient Descent, SGD), que ajusta los valores de los parámetros en la dirección opuesta del gradiente, de manera que la función de pérdida disminuya. El ajuste de los parámetros se realiza iterativamente, actualizando los valores de los parámetros en cada iteración en función del valor del gradiente.

Finalmente, la propagación hacia atrás es el proceso mediante el cual se calcula el

gradiente de la función de pérdida con respecto a los parámetros de la red, utilizando la regla de la cadena y la retropropagación de los errores. El algoritmo de descenso de gradiente estocástico se utiliza para ajustar los valores de los parámetros en la dirección del gradiente, minimizando la función de pérdida.

2.0.2. YOLOv5

La arquitectura YOLOv5 es una red neuronal convolucional utilizada para la detección de objetos en tiempo real. Fue desarrollada por Ultralytics en 2020 y se basa en la arquitectura YOLO (You Only Look Once) original creada por Joseph Redmon y colaboradores en 2015. La arquitectura YOLOv5 consta de varios módulos que se encargan de diferentes tareas. A continuación se describe cada uno de ellos:

Backbone: la columna vertebral de la red neuronal está formada por una serie de capas convolucionales que procesan la imagen de entrada para extraer características de alto nivel.

El componente backbone de YOLOv5 se refiere a la red neuronal convolucional que se encarga de extraer características de las imágenes de entrada. En YOLOv5, se utiliza una versión modificada de la arquitectura EfficientNet como backbone. La arquitectura EfficientNet se basa en un enfoque de optimización conjunta que busca mejorar la precisión de la red y la eficiencia computacional.

El backbone de YOLOv5 se compone de varias capas convolucionales, incluyendo capas convolucionales 3x3, capas convolucionales 1x1 y capas de agrupamiento. Estas capas trabajan en conjunto para extraer características de diferentes escalas de las imágenes de entrada. A medida que las características se propagan a través de la red, su tamaño se reduce, pero su número de canales se incrementa.

La arquitectura EfficientNet utiliza una técnica llamada çompound scaling"para optimizar conjuntamente el tamaño y la profundidad de la red. En lugar de simplemente

aumentar el tamaño de la red, se ajustan simultáneamente el ancho, la profundidad y la resolución de entrada de la red. Esto permite que la red sea más precisa y eficiente computacionalmente.

La extracción de características en el backbone se realiza a través de varias capas convolucionales que aplican operaciones matemáticas sobre los datos de entrada. En particular, cada capa convolucional aplica una operación de convolución en la que se convoluciona un filtro de tamaño determinado con los datos de entrada.

Las operaciones de agrupamiento, como el MaxPooling, también se utilizan en el backbone para reducir el tamaño de las características y ayudar a mantener la invariancia a la traslación.

En resumen, el backbone de YOLOv5 es una red neuronal convolucional basada en la arquitectura EfficientNet que se encarga de extraer características de diferentes escalas de las imágenes de entrada. Esto se logra a través de la aplicación de varias capas convolucionales y capas de agrupamiento que reducen el tamaño de las características a medida que se propagan a través de la red. La técnica de çompound scaling"se utiliza para optimizar conjuntamente el tamaño y la profundidad de la red.

La arquitectura del YOLOv5 utiliza una versión mejorada de la arquitectura CSPNet como su columna vertebral (backbone). La arquitectura CSPNet se enfoca en resolver el problema de la excesiva cantidad de cálculos de la red, y para ello utiliza el concepto de la separación del canal de características (feature channel splitting). La idea es dividir el conjunto de características de entrada en dos grupos, y aplicar diferentes operaciones a cada grupo de características, reduciendo así el número de cálculos necesarios.

El Backbone de YOLOv5 se basa en la arquitectura CSPNet (Cross Stage Partial Network), que utiliza módulos de tipo residual"para mejorar la eficiencia del cálculo.

Cada módulo CSP consta de tres partes: una convolución inicial, bloques residuales y una convolución final. El esquema de CSP se usa para permitir que la información fluya a través de múltiples canales en cada etapa del modelo.

La estructura del Backbone de YOLOv5 consta de varias etapas de CSP, con un número creciente de filtros en cada etapa. Cada etapa se compone de dos bloques CSP, seguidos de una capa de reducción espacial (convolución con stride 2), excepto la última etapa que tiene solo un bloque CSP.

La ecuación detrás de cada bloque CSP es:

$$y = x + F_{csp}(x) \tag{2.15}$$

donde x es la entrada y F_{csp} representa la función de activación residual. Esta función de activación consiste en una serie de capas convolucionales, con una estructura de ramificación que permite el flujo de información a través de múltiples canales.

La capa de reducción espacial en cada etapa reduce la resolución espacial de la entrada por un factor de 2 mediante una convolución con stride 2. Esto se logra mediante la siguiente ecuación:

$$y_{i,j,k} = \sum_{r=0}^{R-1} \sum_{s=0}^{S-1} \sum_{c=0}^{C_{in}-1} w_{r,s,c,k} x_{2i+r,2j+s,c} \tag{2.16}$$

donde x es la entrada, y es la salida, w son los pesos de la capa y R, S y C_{in} son el tamaño del kernel de la convolución, el tamaño del stride y el número de canales de entrada, respectivamente.

Neck: el cuello de la red es un conjunto de capas convolucionales que se encargan de fusionar características de diferentes escalas y resoluciones para mejorar la precisión de la detección.

El componente del çuello"(neck) en YOLOv5 tiene la tarea de fusionar las caracterísi-

cas de varias escalas para mejorar la precisión en la detección de objetos. El çuello.está compuesto por tres bloques principales: el bloque CSP, el bloque SPP y el bloque PAN.

El bloque CSP (Cross-Stage Partial) utiliza un enfoque de división de canal para reducir el costo computacional y mejorar la eficiencia. La entrada se divide en dos ramas, una de las cuales pasa a través de varias capas convolucionales y la otra se envía directamente a la salida. Luego, las características de ambas ramas se concatenan y se procesan en varias capas convolucionales adicionales.

El bloque SPP (Spatial Pyramid Pooling) utiliza una pirámide de agrupación espacial para capturar características de diferentes tamaños. La entrada se procesa a través de una serie de agrupaciones máximas a diferentes tamaños de kernel, y las características resultantes se concatenan y se pasan a través de una capa convolucional.

Finalmente, el bloque PAN (Path Aggregation Network) fusiona las características de diferentes escalas para mejorar la precisión de detección. La entrada se procesa a través de una serie de capas convolucionales para ajustar el tamaño y la dimensión de las características de diferentes escalas. Luego, las características se fusionan por medio de un enfoque de suma ponderada para producir características de detección precisas.

El bloque CSP es un módulo que ayuda a reducir la cantidad de parámetros y mejorar la eficiencia computacional en una red neuronal. La ecuación para el bloque CSP se puede escribir como:

$$\begin{aligned} x &= Conv_B N_L eakyReLU(x) \\ y_1, y_2 &= Split(x) \\ y_1 &= Conv_B N_L eakyReLU(y_1) \\ y_1 &= Conv_B N(y_1) \\ x &= Concatenate([y_1, y_2]) \\ x &= Conv_B N_L eakyReLU(x) \end{aligned} \tag{2.17}$$

Donde x es la entrada y y_1 y y_2 son las dos ramas resultantes de la división de canal. $Conv_B N_L eakyReLU$ se refiere a una secuencia de convolución, normalización de lotes y activación LeakyReLU. *Concatenate* se refiere a la concatenación de las dos ramas.

El bloque SPP es un módulo que realiza un muestreo piramidal espacial para capturar diferentes escalas de información en la red neuronal. La ecuación para el bloque SPP se puede escribir como:

$$\begin{aligned} x = Concatenate([&GlobalAveragePooling2D()(x), \\ &GlobalMaxPooling2D()(x), \\ &*[Conv2D(filters, 1, \\ &padding =' same', \\ &activation =' relu')(x) \\ &\text{for filters in } [512, 256, 128]]]) \end{aligned} \tag{2.18}$$

Donde x es la entrada y $Conv2D$ se refiere a una capa convolucional con un kernel de 1x1 y el número de filtros especificado. $GlobalAveragePooling2D$ y $GlobalMaxPooling2D$ se refieren a agrupaciones globales de promedio y máximo, respectivamente.

El bloque PAN es un módulo que fusiona características de diferentes niveles de la pirámide para mejorar la precisión de la detección de objetos en diferentes escalas. La ecuación para el bloque PAN se puede escribir como:

$$\begin{aligned} x_1, x_2, x_3 &= FPN_features \\ x_3 &= Conv_BN_LeakyReLU(x_3, 256, 1) \\ x_2 &= Add()([UpSampling2D(2)(x_3), x_2]) \\ x_2 &= Conv_BN_LeakyReLU(x_2, 256, 1) \\ x_1 &= Add()([UpSampling2D(2)(x_2), x_1]) \\ x_1 &= Conv_BN_LeakyReLU(x_1, 256) \end{aligned} \tag{2.19}$$

Donde x_1, x_2 y x_3 son las características de diferentes niveles de la pirámide de características (FPN). $Conv_BN_LeakyReLU$ se refiere a una secuencia de convolución, normalización de lotes y activación LeakyReLU. $UpSampling2D$ se refiere a una operación de interpolación que aumenta la resolución de la imagen. *Add* se refiere a una operación de suma que fusiona características de diferentes niveles de la pirámide.

Head: la cabeza de la red está formada por varias capas convolucionales que se encargan de generar las predicciones de detección de objetos.

Post-processing: por último, se aplican una serie de técnicas para mejorar la precisión de las predicciones, como la eliminación de falsos positivos y la corrección de la ubicación de los objetos detectados.

La etapa head en YOLOv5 es responsable de generar las predicciones de objetos a partir de las características extraídas por el backbone. Aquí está una descripción detallada de los pasos involucrados en el proceso del head en YOLOv5:

Predicciones de rejilla (grid predictions): En esta etapa, la salida del backbone se divide en una cuadrícula de celdas, donde cada celda se encarga de predecir los objetos que caen dentro de ella.

Para cada celda en la cuadrícula, se generan múltiples anclas que representan distintos tamaños y relaciones de aspecto posibles de los objetos.

Cada ancla se representa mediante un vector de predicción que contiene información sobre la posición del objeto, la confianza en la detección y las probabilidades de las clases.

Predicciones de caja (box predictions): Cada celda en la cuadrícula genera predicciones de cajas que representan la posición y el tamaño del objeto detectado. Estas predicciones se realizan mediante la regresión de valores relativos a la caja de anclaje, que se calculan en función de la posición de la celda en la cuadrícula.

Predicciones de objeto (object predictions): Para cada celda en la cuadrícula, se realiza una predicción de objeto que determina la confianza de que un objeto esté presente en esa celda. Esta predicción se obtiene calculando la probabilidad de objeto como la multiplicación de la confianza de la detección y las probabilidades de las clases.

Predicciones de clase (class predictions): Cada celda en la cuadrícula también realiza predicciones de clase, que representan las probabilidades de las diferentes clases de objetos. Estas predicciones se calculan utilizando una función de activación softmax para obtener probabilidades normalizadas.

Filtros de confianza (confidence filters): Después de obtener las predicciones de cada celda en la cuadrícula, se aplican filtros de confianza para seleccionar las detecciones más relevantes.

Los filtros de confianza se basan en un umbral predefinido que determina qué detecciones se consideran válidas y cuáles se descartan.

Supresión de no máximos (non-maximum suppression): Para evitar la duplicación de detecciones para el mismo objeto, se aplica una supresión de no máximos.

Este proceso implica seleccionar las detecciones con mayor confianza y eliminar las detecciones redundantes que tienen una superposición significativa con las detecciones seleccionadas.

El paso de la cabeza (head) en YOLOv5 es el último paso en la red neuronal que se encarga de producir las predicciones de objetos. El head toma las características de las capas anteriores y las procesa para obtener las predicciones finales.

La ecuación para la capa head en YOLOv5 se puede escribir de la siguiente manera:

$$
\begin{aligned}
x &= Conv\_BN\_LeakyReLU(x, 512, 3, strides = 2) \\
x &= Conv\_BN\_LeakyReLU(x, 1024, 3) \\
x &= Conv\_BN\_LeakyReLU(x, 512, 1) \\
x &= Conv\_BN\_LeakyReLU(x, 1024, 3) \\
x &= Conv\_BN\_LeakyReLU(x, 512, 1) \\
x &= Conv\_BN\_LeakyReLU(x, 1024, 3)
\end{aligned}
\tag{2.20}
$$

En esta ecuación, x es la entrada que se ha procesado a través de las capas anteriores de la red neuronal. $Conv\_BN_LeakyReLU$ se refiere a una secuencia de convolución, normalización de lotes y activación $LeakyReLU$. Cada capa convolucional se especifica con un tamaño de kernel de $3x3$ y un número de filtros que se especifica en los argumentos de la función.

Después de estas capas, se realiza la detección de objetos utilizando el algoritmo YOLOv5. Este algoritmo utiliza una malla de anclajes (anchor grid) para detectar objetos de diferentes tamaños y formas. Cada celda en la malla de anclajes se encarga de detectar un objeto en particular.

Para cada celda en la malla de anclajes, se generan predicciones para la clase de objeto, la confianza en la detección y las coordenadas de la caja delimitadora. Las predicciones se realizan mediante capas convolucionales y se procesan para producir las predicciones finales.

La ecuación para la detección de objetos en YOLOv5 se puede escribir de la siguiente manera:

$$
\begin{aligned}
output = Conv2D(num\_anchors * (num\_classes + 5), 1, strides = 1, padding =' same')(x) \\
output = Reshape((grid\_size, grid\_size, num\_anchors, num\_classes + 5))(output) \\
grid = CoordGrid(grid\_size) \\
output = Lambda(lambdaargs : args[0] + grid)([output, grid]) \\
box\_xy, box\_wh, objectness, class\_probs = YoloLayer(num\_classes, anchors)(output)
\end{aligned} \tag{2.21}
$$

En esta ecuación, output es la salida de las capas anteriores en la red neuronal. $num\_anchors$ se refiere al número de anclajes en la malla de anclajes y $num\_classes$ se refiere al número de clases de objetos que se están detectando. La función $Conv2D$ se utiliza para generar predicciones para cada celda en la malla de anclajes. La salida de la capa $Conv2D$ se remodela en una forma que se puede procesar para producir las predicciones finales.

CoordGrid se refiere a una función que genera una malla de coordenadas para cada celda en la malla de anclajes. Esta malla se utiliza para generar las predicciones finales para las coordenadas de la caja delimitadora. YoloLayer se refiere a una función que procesa las predicciones y produce las coorden.

2.0.3. Estado del arte, AI para invidentes

Una vez que se ha abordado el contexto de las redes neuronales convolucionales y de YOLOv5, ahora se abordan algunos trabajos relacionados con la detección de objetos para las personas con alguna discapacidad visual. Es importante mencionar que los trabajos pueden ser muy variados con respecto a la aplicación, es decir, hay trabajos enfocados en gnerar lectura de textos para personas invidentes, otros que detectan diferentes objetos, etc. Esto nos permite comprender que la aplicación de soluciones de detección de objetos es muy diversa, lo cual nos permitirá entender dichas aplicaciones y como incluso entre estas como a futuro se puede generar una solución que permita a

las personas con alguna discapacidad visual, tener una gran calidad de vida.

Uno de los principales trabajos que resulta interesante es sobre la detección de objetos para personas con discapacidad visual utilizando el algoritmo SSD, este trabajo propone un método de detección de objetos utilizando el algoritmo SSD. El algoritmo se entrena con un conjunto de datos de imágenes de objetos comúnmente encontrados por personas con discapacidad visual. El sistema es capaz de detectar objetos en tiempo real y proporcionar retroalimentación de audio al usuario.

El trabajo toma una discución sobre el problema de la detección de objetos para personas con discapacidad visual. Se señala que las personas con discapacidad visual a menudo tienen dificultades para moverse en su entorno e identificar objetos. Esto puede llevar a varios riesgos para la seguridad, así como a una disminución en la calidad de vida.

Luego, el artículo introduce el algoritmo SSD. SSD es un algoritmo de aprendizaje profundo que puede detectar objetos en imágenes. El algoritmo se entrena con un conjunto de datos de imágenes de objetos comúnmente encontrados por personas con discapacidad visual. Esto incluye objetos como sillas, mesas, escaleras y puertas.

El artículo describe el sistema de detección de objetos. El sistema consta de una cámara, una Raspberry Pi y un altavoz. La cámara captura imágenes del entorno y la Raspberry Pi utiliza el algoritmo SSD para detectar objetos en las imágenes. Luego, el altavoz proporciona retroalimentación de audio al usuario sobre los objetos detectados.

Posteriormente, el artículo presenta los resultados de un estudio de usuarios. El estudio mostró que el sistema fue capaz de detectar objetos de manera precisa en tiempo real. Los usuarios también informaron que el sistema era fácil de usar y que mejoraba su capacidad para moverse en su entorno.

El artículo concluye discutiendo el potencial del sistema para mejorar la vida de las personas con discapacidad visual. El sistema tiene el potencial de hacer que sea más seguro y más fácil para las personas con discapacidad visual moverse en su entorno. También podría ayudar a mejorar su calidad de vida al brindarles más independencia y libertad.

Por otro lado, se encuentra el trabajo sobre el reconocimiento automatizado de objetos habilitado por IoT para las personas con discapacidad visual, el cual presenta un sistema de reconocimiento de objetos que utiliza dispositivos IoT. El sistema consta de una cámara, una Raspberry Pi y un altavoz. La cámara captura imágenes del entorno y la Raspberry Pi utiliza el algoritmo SSD para detectar objetos en las imágenes. Luego, el altavoz proporciona retroalimentación de audio al usuario sobre los objetos detectados.

El artículo comienza discutiendo el problema del reconocimiento de objetos para personas con discapacidad visual. Se señala que las personas con discapacidad visual a menudo tienen dificultades para identificar objetos en su entorno. Esto puede llevar a varios riesgos para la seguridad, así como a una disminución en la calidad de vida.

Luego, el artículo presenta los dispositivos IoT que se utilizan en el sistema. La cámara es una cámara de bajo costo que está conectada a la Raspberry Pi. La Raspberry Pi es una pequeña computadora de placa única que se utiliza para ejecutar el algoritmo SSD. El altavoz es un pequeño altavoz portátil que se utiliza para proporcionar retroalimentación de audio al usuario.

A continuación, el artículo describe el sistema de reconocimiento de objetos. El sistema se activa cuando el usuario lleva un reloj inteligente. El reloj inteligente envía una señal a la cámara, que captura una imagen del entorno. La imagen se envía luego a la Raspberry Pi, que utiliza el algoritmo SSD para detectar objetos en la imagen. El altavoz proporciona entonces retroalimentación de audio al usuario sobre los objetos

detectados.

Posteriormente, el artículo presenta los resultados de un estudio de usuarios. El estudio mostró que el sistema fue capaz de detectar objetos de manera precisa en tiempo real. Los usuarios también informaron que el sistema era fácil de usar y que mejoraba su capacidad para moverse en su entorno.

El artículo concluye discutiendo el potencial del sistema para mejorar la vida de las personas con discapacidad visual. El sistema tiene el potencial de hacer que sea más seguro y más fácil para las personas con discapacidad visual moverse en su entorno. También podría ayudar a mejorar su calidad de vida al brindarles más independencia y libertad.

Adicional, hay un trabajo que nos permite entender como funcionan los sistemas de reconocimiento de objetos para personas con discapacidad visual, en este trabajo se presenta un sistema de reconocimiento de objetos que utiliza un teléfono inteligente. El sistema utiliza el algoritmo Mobilenet SSD para detectar objetos en imágenes capturadas por la cámara del teléfono inteligente. Los objetos detectados se muestran en la pantalla del teléfono inteligente y se proporciona retroalimentación de audio al usuario.

El artículo comienza discutiendo el problema del reconocimiento de objetos para personas con discapacidad visual. Se señala que las personas con discapacidad visual a menudo tienen dificultades para identificar objetos en su entorno. Esto puede dar lugar a varios riesgos para la seguridad, así como a una disminución en la calidad de vida.

Luego, el artículo presenta el algoritmo Mobilenet SSD. Mobilenet SSD es un algoritmo de aprendizaje profundo que es capaz de detectar objetos en imágenes. El algoritmo se entrena en un conjunto de datos de imágenes de objetos que son comúnmente encontrados por personas con discapacidad visual. Esto incluye objetos como sillas, mesas, escaleras y puertas.

A continuación, el artículo describe el sistema de reconocimiento de objetos. El sistema consta de un teléfono inteligente y un par de auriculares. El teléfono inteligente captura imágenes del entorno utilizando su cámara. Luego, el algoritmo Mobilenet SSD detecta objetos en las imágenes. Los objetos detectados se muestran en la pantalla del teléfono inteligente y se proporciona retroalimentación de audio al usuario a través de los auriculares.

Bibliografía

[1] Goodfellow, Ian, et al. "Deep Learning."MIT Press, 2016.

[2] LeCun, Yann, et al. Çonvolutional neural networks.Çommunications of the ACM 61.6 (2018): 514-529.

[3] Nielsen, Michael. "Neural Networks and Deep Learning."Determination Press, 2015.

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. Ïmagenet classification with deep convolutional neural networks..Advances in neural information processing systems. 2012.

[5] Zhang, Xiangyu, et al. "ShuffleNet: An extremely efficient convolutional neural network for mobile devices."Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[6] Bracewell, R. N. (1986). The Fourier transform and its applications (2nd ed., p. 159). McGraw-Hill.

[7] Richard Haberman, .Applied Partial Differential Equations with Fourier Series and Boundary Value Problems,"Pearson Education, 2003.

[8] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).