UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

Acreditación Institucional Internacional

OTORGADA POR EL IAC CINDA ACUERDO 55 DEL 9 DE MAYO-VIGENCIA 5 AÑOS

ACREDITACIÓN INSTITUCIONAL DE ALTA CALIDAD MULTICAMPUS

Res. MEN No. 01456 del 29 de enero de 2016

Vigencia por seis años

QS STARS RATED FOR EXCELLENCE

ISO 9001 icontec SC4289-I

CERTIFIED IQNet MANAGEMENT SYSTEM

UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

Faculty: systems engineer
Course: Deep Learning
Topic: CNN-visión por computadora (Dataset's train/test)

Professor: Luis Fernando Castellanos Guarin
Email: Luis.castellanosg@usantoto.edu.co
Phone: 3214582098

CONTENIDO

1. **Teoría e historia.**
2. **Crear Dataset de entrenamiento y pruebas**
3. Preparando el ambiente para entrenamiento
4. Realizar entrenamiento (usando tensorflow con redes CNN)
5. Congelar el modelo (model.ckpt)
6. Convertir el modelo para uso en dispositivos de bajo rendimiento (celulares/Raspberry)
7. Usando OPENCV y el modelo entrenado:
    1. Crear una app para Raspberry para visión por computadora
    2. Crear una app para android

¡Siempre hacia lo alto!

1. Teoría e historia.

¿Qué son las Redes Neuronales Convolucionales (CNN-Convolutional Neural Networks)?

Las redes neuronales convolucionales (CNN) son muy similares a las redes neuronales ordinarias como el perceptron multicapa **MLP** (MultiLayer Perceptron) que vimos una semana atrás; se componen de neuronas que tienen pesos y sesgos que pueden aprender. Cada neurona recibe algunas entradas, realiza un producto escalar y luego aplica una función de activación.

Lo que diferencia a las CNN es que suponen explícitamente que las entradas son **imágenes**, lo que nos permite codificar ciertas propiedades en la arquitectura; permitiendo ganar en eficiencia y reducir la cantidad de parámetros en la red.

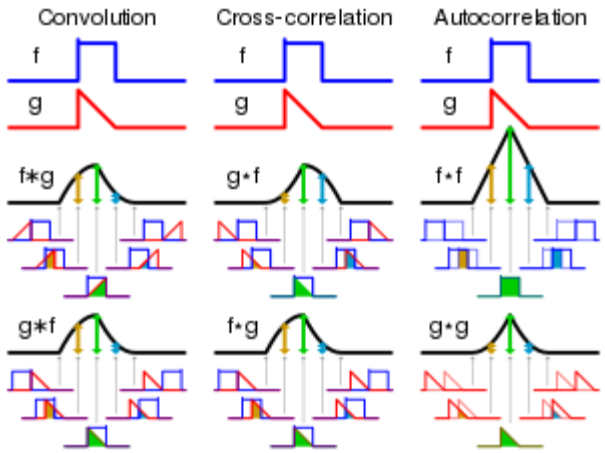
Las CNN vienen a solucionar el problema de que las MLP ordinarias **no escalan bien para imágenes de mucha definición**. En el ejemplo de **MNIST**, las imágenes son de **28x28**; por lo que una sola neurona plenamente conectado en una primera capa oculta de una red neuronal ordinaria tendría **28 x 28 = 784 pesos (se demora pero es manejable)**. Pero con una imagen de mayor tamaño, por ejemplo de 200x200 con colores RGB, daría lugar a neuronas que tienen **200 x 200 x 3 = 120.000** pesos (lo que haría demasiado costoso el entrenamiento). Por otra parte, el contar con tantos parámetros, también sería un desperdicio de recursos y conduciría rápidamente a **sobreajuste**.

1. Teoría e historia.

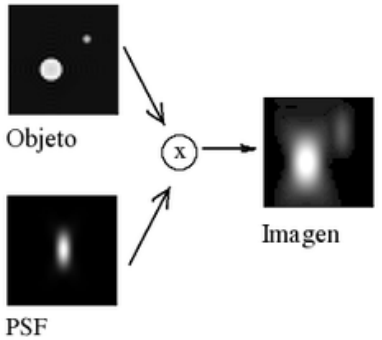
Estructura de las Redes Neuronales Convolucionales

En general, las CNN van a estar construidas con una estructura que contendrá 3 tipos distintos de capas:

1. Una capa **convolucional**, que es la que le da le nombre a la red.
2. Una capa de reducción o de pooling, la cual va a reducir la cantidad de parámetros al quedarse con las características más comunes.
3. Una capa clasificadora totalmente conectada, la cual nos va dar el resultado final de la red.



Convolución y correlación. La simetría de la función f hace que g*f Y f*g sean operaciones idénticas con resultados idénticos.



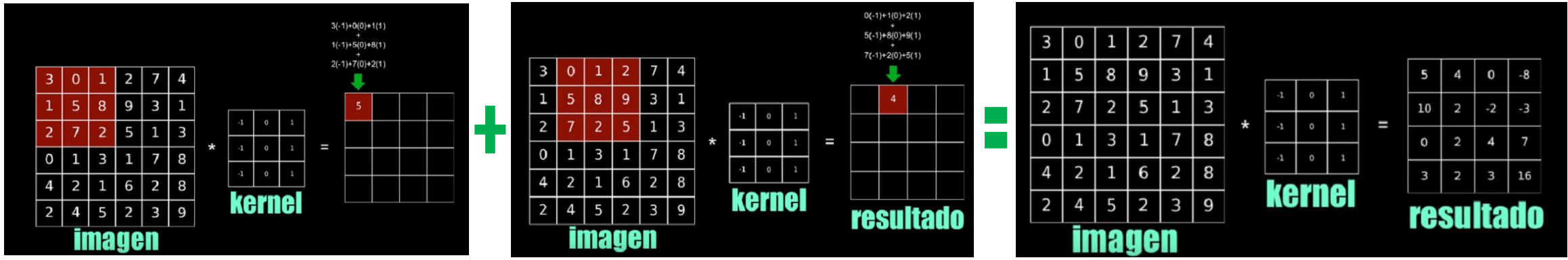
Convolución en un dispositivo óptico (microscopio de fluorescencia, corte longitudinal de una imagen 3D).

Convolución: *es un operador matemático que transforma dos funciones **f** y **g** en una tercera función que en cierto sentido representa la magnitud en la que se superponen **f** y una versión trasladada e invertida de **g**.*

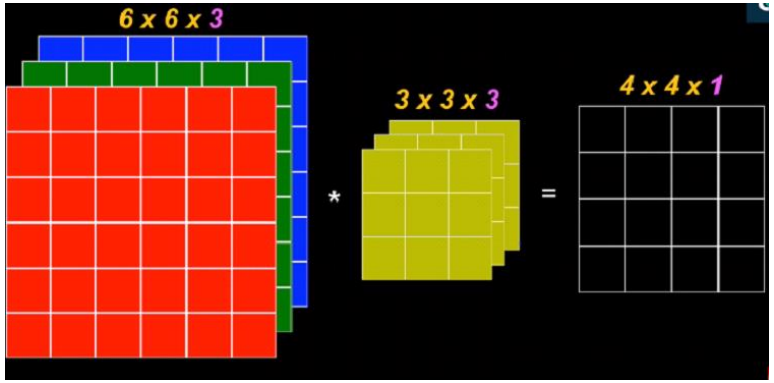
1. Teoría e historia.

CNN - Capa convolucional

La operación de convolución recibe como entrada(***input***) la imagen y luego aplica sobre ella un filtro(***kernel***, por ejemplo el filtro **canny** que me genera los bordes de contornos). Que generan un mapa de las características de la imagen, y con ello reducir el tamaño de los parámetros.



Convolución en RGB



¡Siempre hacia lo alto!

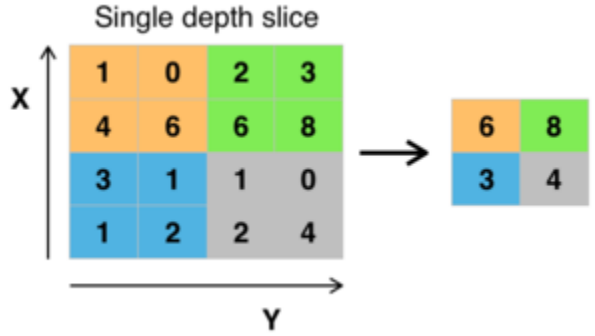
1. Teoría e historia.

CNN - Capa reducción o pooling, Se coloca generalmente después de la capa convolucional.

Permite reducir las dimensiones espaciales (ancho x alto) del volumen de entrada para la siguiente capa convolucional. No afecta a la dimensión de profundidad del volumen. La operación realizada por esta capa también se llama reducción de muestreo, ya que la reducción de tamaño conduce también a la pérdida de información. Sin embargo, una pérdida de este tipo puede ser beneficioso para la red por dos razones:

- La disminución en el tamaño conduce a una menor sobrecarga de cálculo para las próximas capas de la red
- También trabaja para reducir el sobreajuste.

La operación que se suele utilizar en esta capa es max-pooling, que divide a la imagen de entrada en un conjunto de rectángulos y, respecto de cada subregión, se va quedando con el máximo valor.

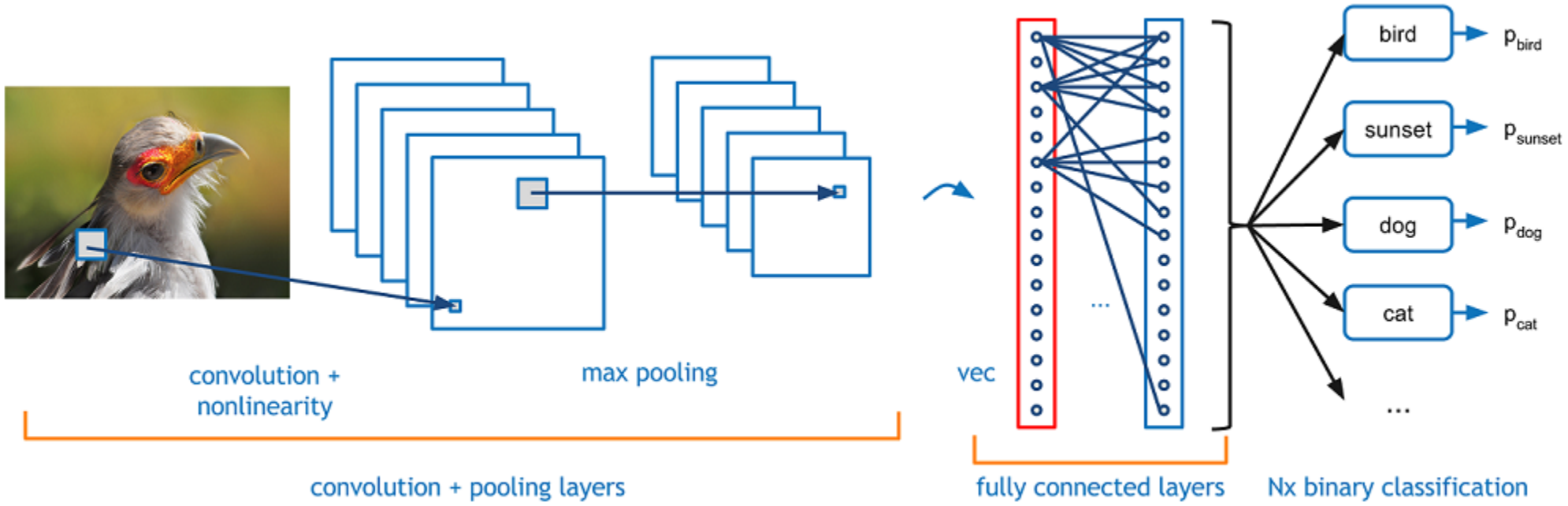


1. Teoría e historia.

CNN - Capa clasificadora totalmente conectada

Al final de las capas convolucional y de pooling, las redes utilizan generalmente capas completamente conectados en la que cada pixel se considera como una neurona separada al igual que en una red neuronal regular. Esta última capa clasificadora tendrá tantas neuronas como el número de clases que se debe predecir.

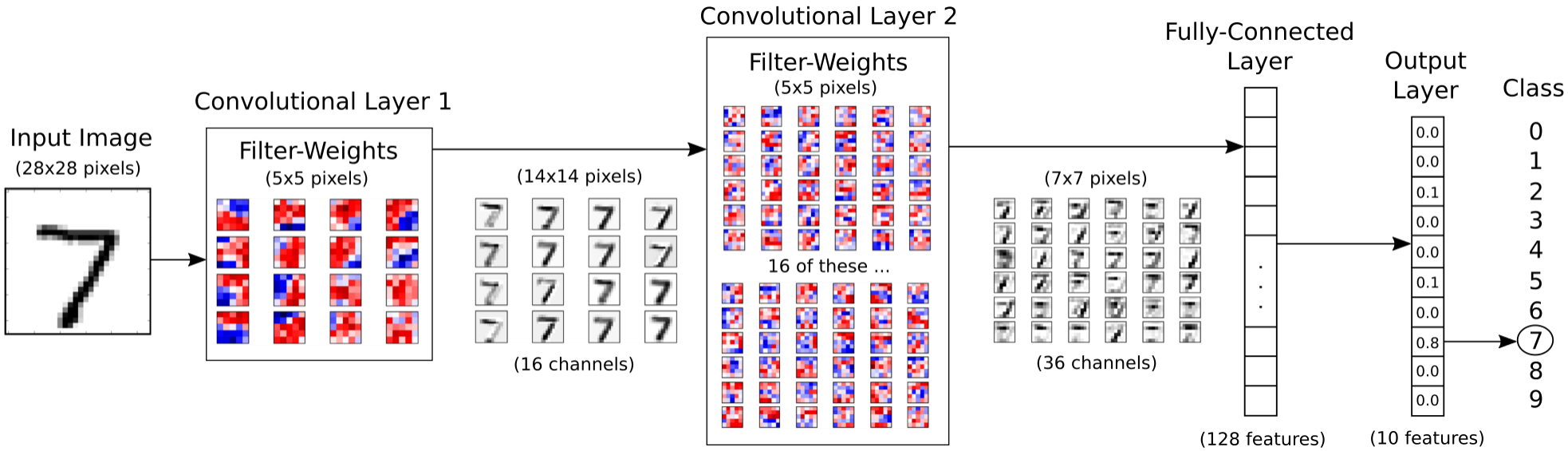
Ejemplo de un CNN



1. Teoría e historia.

¿Que es Tensorflow?

Es una interfaz de programación multiplataforma y escalable para implementar y ejecutar algoritmos de aprendizaje automático, incluyendo ***wrappers*** preparados para aprendizaje profundo (Deep learning).



Wrapper:Una función de envoltura es una subrutina en una biblioteca de software o un programa de computadora cuyo propósito principal es llamar a una segunda subrutina o una llamada al sistema con poco o ningún cálculo adicional.

1. Teoría e historia.

Historia de Tensorflow

Fue desarrollado por investigadores e ingenieros del equipo de Google Brain. Mientras 1que el desarrollo principal esta liderado por un equipo de investigadores e ingenieros de software de Google, su desarrollo también se debe a muchas contribuciones por parte de comunidades de código abierto.

Inicialmente Tensorflow fue creado solo para uso interno de Google, pero posteriormente (noviembre de 2015) fue lanzado bajo la licencia de código abierto permisiva (*minimalista o liberal ósea sin copyleft… la máxima libertad en uso de software*).

Tensorflow soporta varias interfaces front-end para muchos lenguajes (c++, Java, Swift, Haskell, noje.js, javascript) y Python cuyo lenguaje es para el cual esta más optimizado y completo.

Tensorflow es una API de **bajo nivel** y requiere (no es obligatorio pero ayuda) de una API de **alto nivel**, las más populares son LAYERS y KERAS.

Para este proyecto necesitaremos usar los siguientes software/api/librería:

1. Python 3.7 o superior
2. Tensorflow 1.1 o superior
3. Numpy
4. Pandas
5. Matplotlib
6. Protobuf
7. Pillow
8. Opencv

Por lo tanto seguiremos usando **Google colaboratory** para entrenamiento (***con la limitación de máximo 12 horas continuas de entrenamiento***), para ello crearemos una carpeta en GOOGLE DRIVE que llamaremos **deteccion_objetos**, en donde guardaremos todo el proyecto.

2. Crear Dataset de entrenamiento y pruebas

Para crear un dataset especifico para uso en CNN debemos seguir los siguientes pasos:

1. Definir las clases (objetos) que vamos a querer que la IA nos reconozca mediante visión por computador
2. Crear Dataset de imágenes para entrenamiento usando (**labelImg)**
3. Crear CVS con los mapeos de las clases asociadas a todas las imagenes.
4. Crear dataframe (CVS+XML+IMAGENES)

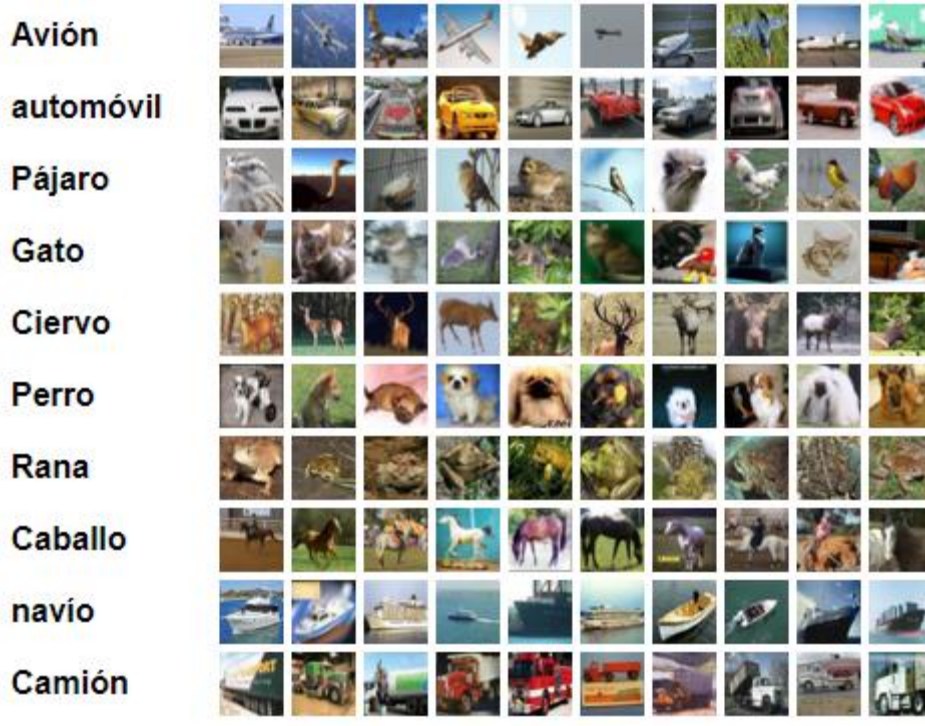
2. Crear Dataset de entrenamiento y pruebas

2.1 Definir las clases (objetos) que vamos a querer que la IA nos reconozca mediante visión por computador:

Debemos determinar que queremos que la IA nos reconozca en las imágenes (fotos o en video) o usar un dataset existente.

Existen grandes Dataset's con clasificaciones de obje

- **CIFAR-10**: consta de 60000 imágenes en color de 32x32 en 10 clases, con 6000 imágenes por clase. Hay 50000 imágenes de entrenamiento y 10000 imágenes de prueba.
- **CIFAR-100:** es igual que el CIFAR-10, excepto que tiene 100 clases que contienen 600 imágenes cada una. Hay 500 imágenes de entrenamiento y 100 imágenes de prueba por clase



https://www.cs.toronto.edu/~kriz/cifar.html

2. Crear Dataset de entrenamiento y pruebas

2.2. Descargar de internet

Descargaremos 30 imágenes (**preferiblemente JPG o JPEG pero todas con la misma extensión**) donde salga el/los **elementos** que queremos que la IA identifique por visión artificial.

entre más diversas sean **las imágenes (diferentes escenas, posiciones/ubicación/ángulos/iluminación/rotación/espacio/color/textura, tamaño dentro de la imagen y en caricaturas)** …mejora la generalización (que es lo que buscamos) reducirá los sobreajustes (que es lo que soñamos).

Recomendaciones:

- Definirles nombres cortos a cada imagen, ejemplo: leon1.jpg, leon2.jpg…
- Tamaño máximo **0.5 MB** en cada imagen (si se supera ese limite, el algoritmo va demorar mucho realizando la **convoluciones** y el **pooling**)…recomiendo usar Herramientas como la web (https://bulkresizephotos.com/es) para redimensionar imágenes de forma masiva (***300 x 300 es lo máximo que deben tener las imágenes***).

*Ancho **X** alto **X** canales*

- Para un trabajo profesional (final del semestre) se requieren mínimo **200 imágenes (200x200x3)** por cada uno de los objetos que queremos detectar.

2. Crear Dataset de entrenamiento y pruebas

2.2. Descargar de internet

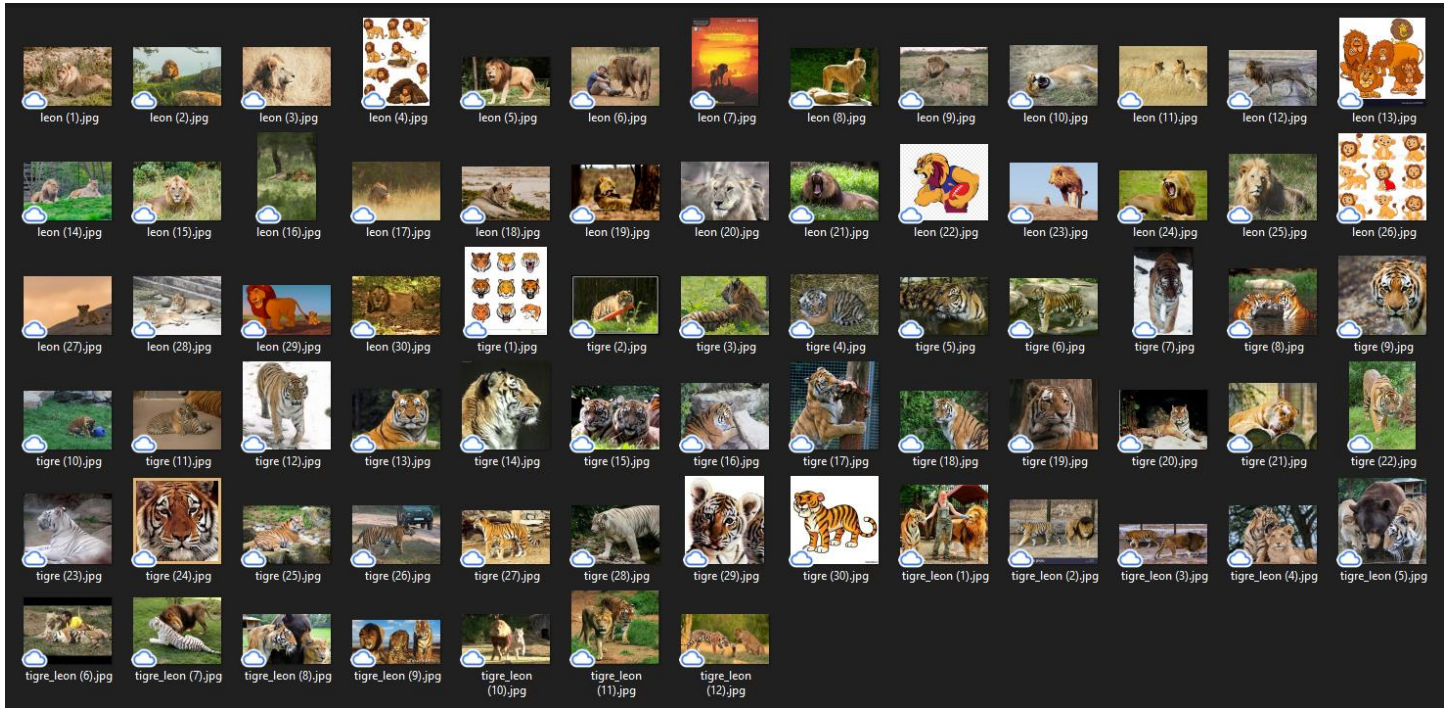
Para el ejercicio se descargaron imágenes:

- 20 Imágenes de leones.
- 20 imágenes de tigres.
- 20 imágenes de panteras
- 20 imágenes donde salen tigres, leones y panteras.

TALLER:

Cada estudiante debe descargar de internet la misma cantidad de imágenes para mínimos tres elementos/animales/cosas diferentes.

NO SE DEBEN REPETIR.



2. Crear Dataset de entrenamiento y pruebas

2.3 Etiquetar imágenes (labels).

usaremos un programa llamado **labelImg** el cual nos facilitara el etiquetado de nuestras imágenes, para descargarlo e instalarlo entrar en su Github:

https://github.com/tzutalin/labelImg

También pueden descargarlos para Windows o Linux desde:

https://tzutalin.github.io/labelImg

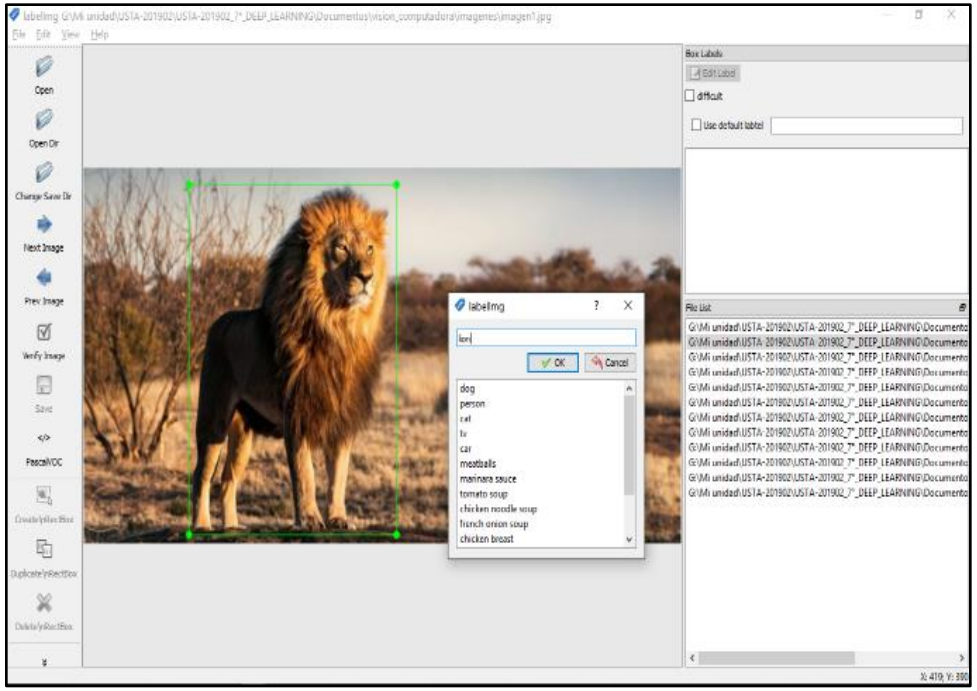
https://github.com/tzutalin/labelImg/releases

Nota:

La carpeta labelImg es recomendable dejarla en la raíz (c:\labelImg)

2. Crear Dataset de entrenamiento y pruebas

2.3 Etiquetar imágenes (labels).



1. Ingrese en la carpeta y en la subcarpeta **data,** abrir el archivo "**predefined_classes.txt**" y agregarle las clases que usted va a trabajar en su proyecto, ejemplo (***lion, tiger***)
2. Una vez inicialice la aplicación dar clic en "**Open Dir**" y seleccionar el directorio donde están las imágenes.
3. Realice los recuadros en cada una de las imágenes y asígnele la etiqueta (label) del objeto/elemento que desea que la IA reconozca por visión.
4. Salve la anotación (**SAVE**), con eso se creara un archivo xml con el mismo nombre de la imagen y siga con la siguiente imagen.

Notas:

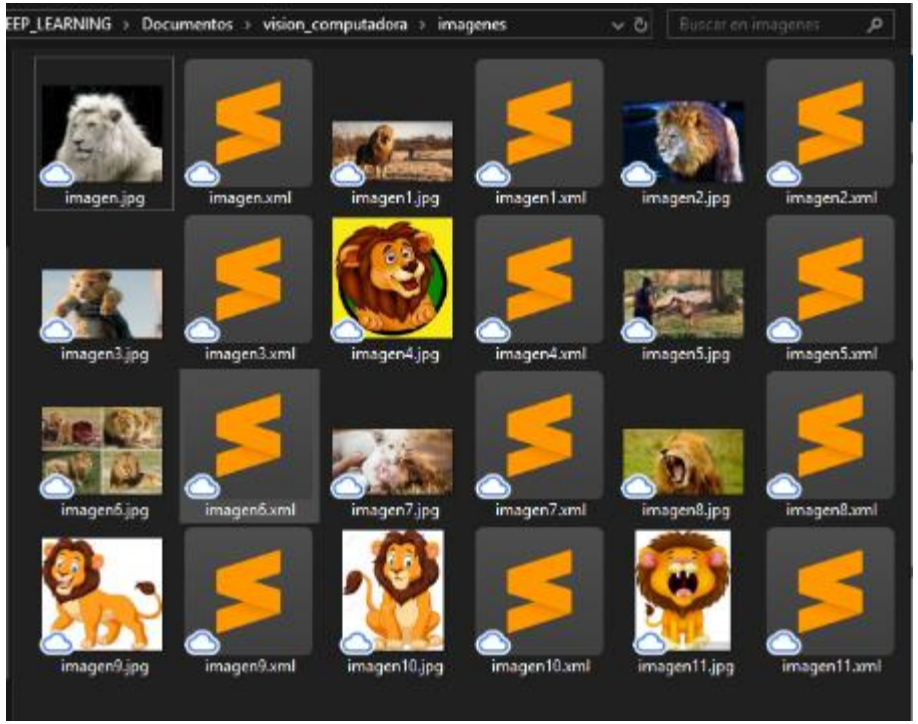
- Cada recuadro debe tener un **label**
- El **label** sea siempre el mismo en todas las imágenes (misma ortografía y todas en minuscula).
- Antes de cambiar de imagen salvar los cambios (**SAVE**)

¡Siempre hacia lo alto!

2. Crear Dataset de entrenamiento y pruebas

2.3 Etiquetar imágenes (labels).

Verifique que en la carpeta donde están las **imágenes** este cada una de las imágenes y su respectivo archivo **XML** con el mismo nombre (estos archivos son las coordenadas donde se dibujaron los recuadros en cada imagen)




```xml
<annotation>
    <folder>imagenes</folder>
    <filename>imagen.jpg</filename>
    <path>G:\Mi unidad\USTA-201902\USTA-201902_7°_DEEP
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>251</width>
        <height>201</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>lion</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>32</xmin>
            <ymin>7</ymin>
            <xmax>212</xmax>
            <ymax>199</ymax>
        </bndbox>
    </object>
</annotation>
```

2. Crear Dataset de entrenamiento y pruebas

2.2 Descargar de internet:

Separar las imágenes en:

- Entrenamiento**(train 80%)**
- Testeo **(test 20%)**

Para ello es necesario crear dos subcarpetas dentro de **deteccion_objetos**

- **img_entrenamiento**: donde colocaremos el 80% de las imágenes
- **img_test**: donde colocaremos el 20% de las imágenes.

Copiar las imágenes con sus respectivos archivos **xml** que están en la carpeta de **deteccion_objetos\images.**

Nota:
Tratar de dejar en test imágenes que tengan relación con las de entrenamiento (ejemplo dejar al menos una imagen en criatura).

En la siguiente presentación esta todo lo referente a la configuración del entorno en:

- Google colaboratory.
- Windows
- Linux/Mac

UNIVERSIDAD SANTO TOMÁS

PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

¡Siempre hacia lo alto!

USTATUNJA.EDU.CO

@santotomastunja