

# Proyecto Final de Carrera

Desarrollo de una aplicación Android, asistida por visión artificial, para administrar imágenes almacenadas en un dispositivo móvil

Francisco Yackel

Director: Dr. Leandro D. Vignolo  
Co-Director: Ing. Leandro Ferrado

Facultad de Ingeniería y Ciencias Hídricas,  
Universidad Nacional del Litoral

24 de Mayo del 2019



# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



# Motivación

---

- La transmisión de información multimedia es un aspecto común y diario.



- Las tareas llevadas a cabo para organizar las imágenes consumen mucho tiempo y esfuerzo.



# Objetivos

---

- Desarrollar una aplicación móvil que, a partir de imágenes almacenadas en el dispositivo, sugiera una estructura de directorios para organizar el conjunto.
- Proveer al sistema funcionalidades que le permitan al usuario la interacción con el resultado, para desempeñar distintas tareas de administración.



# Alcance

---

- Desarrollar de manera **nativa** la aplicación para el sistema operativo Android.
- Implementar el algoritmo de procesamiento y agrupación de imágenes **on premise**.
- Asistir al usuario, brindando **herramientas de administración** para que pueda interactuar con el resultado.
- Procesar un total de **400 imágenes** por ejecución.

# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

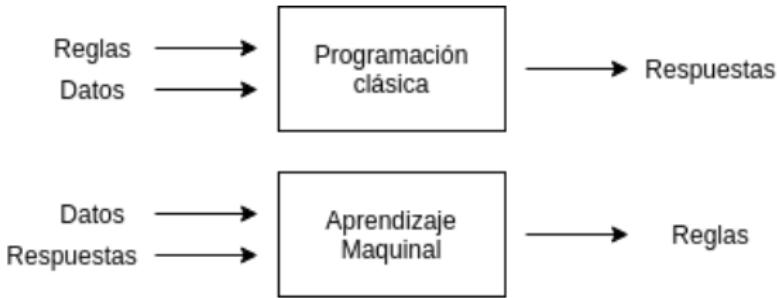
- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



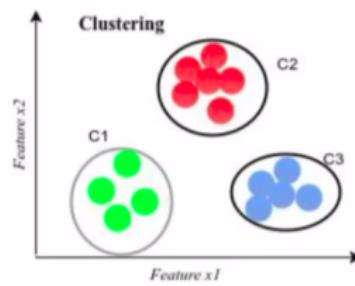
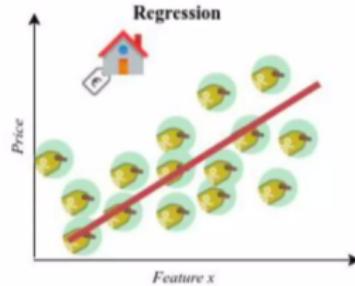
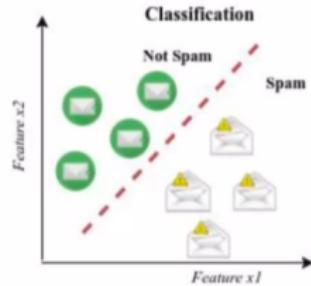
# Machine learning

*Abarca algoritmos y técnicas para construir sistemas que son capaces de aprender a partir de datos y a realizar diversos tipos de tareas, sin requerir que se les indique **cómo** hacerlas.*



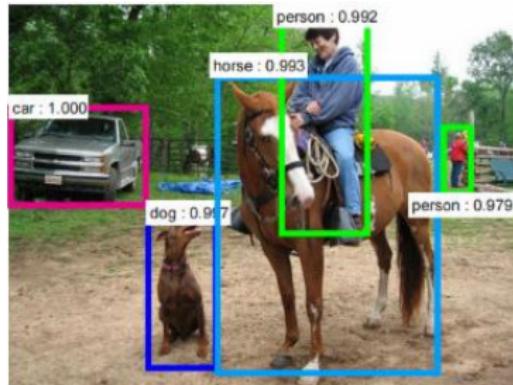
# Machine learning

- Los dos tipos de aprendizaje mas usuales son: **supervisado** y **no supervisado**.
- Las tareas a realizar suelen clasificarse en estas categorías:
  - ▶ Clasificación
  - ▶ Regresión
  - ▶ Clustering

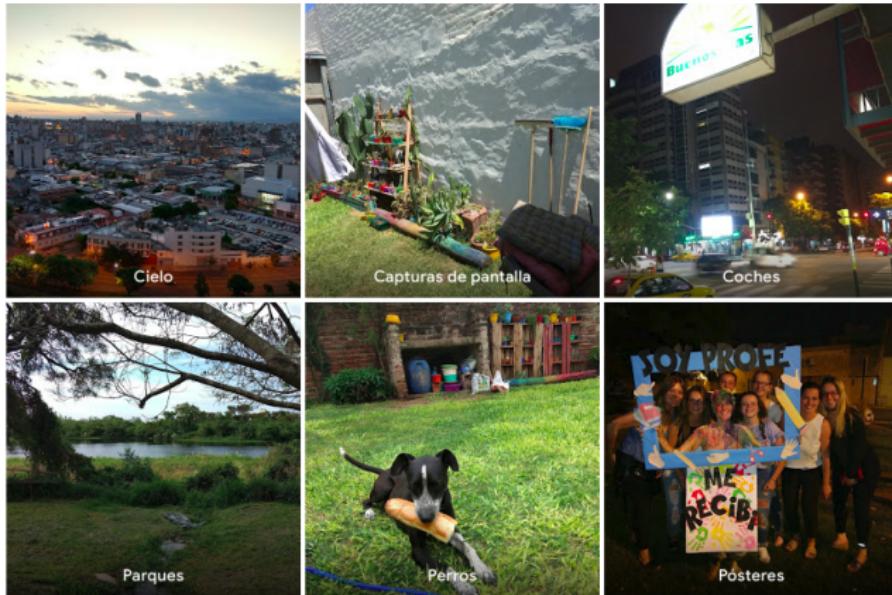


# Visión artificial

*Disciplina que pertenece al aprendizaje maquinal, la cual incluye métodos para **adquirir, procesar, analizar y comprender** las imágenes del mundo real.*

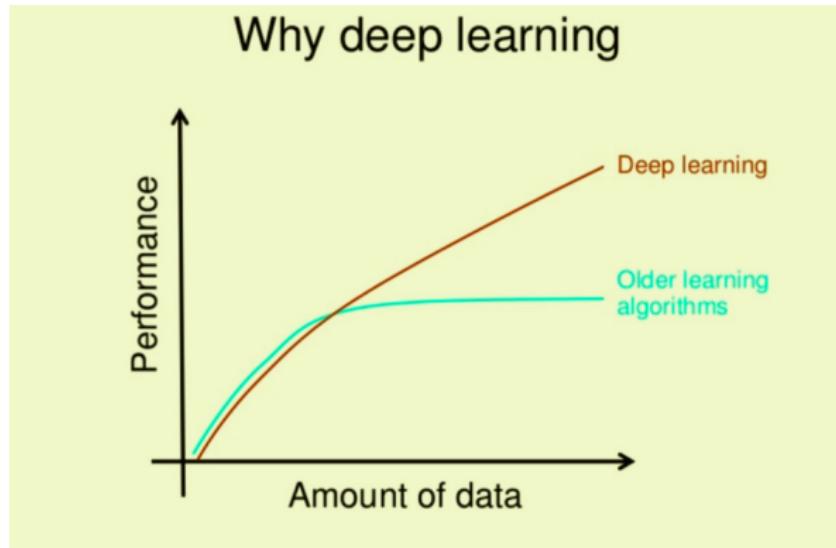


# Antecedentes - Google Photos Organize



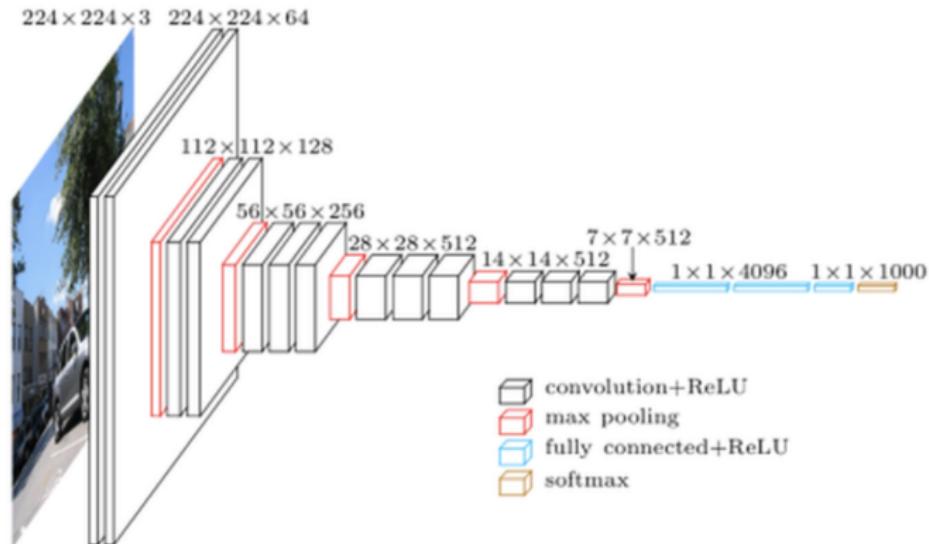
# Deep Learning

*Subcampo específico del aprendizaje maquinal: una manera de aprender desde los datos, poniendo énfasis en sucesivas capas desde las cuales se van obteniendo representaciones cada vez más significativas.*



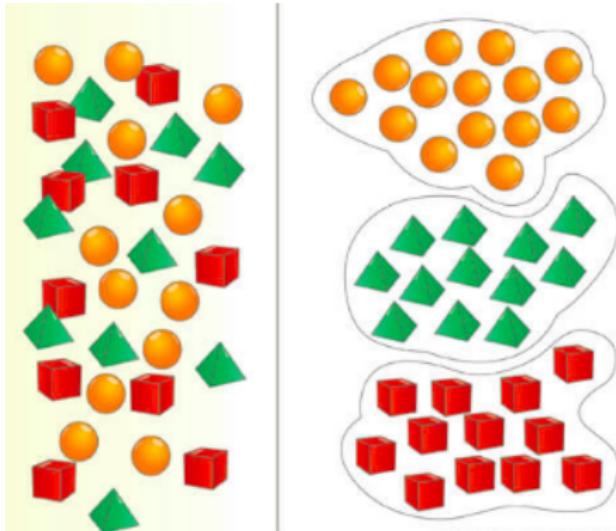
# Deep Learning - Visión Computacional

Red Neuronal Convolucional (**CNN**):



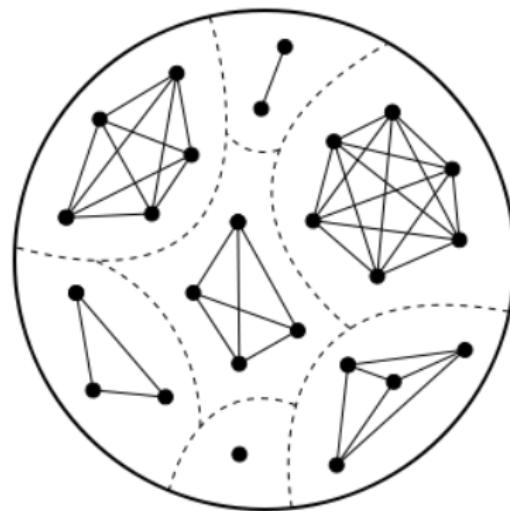
# Agrupamiento o *Clustering*

Tarea que consiste en agrupar un conjunto de objetos tal que los miembros del mismo grupo (llamado **cluster**) sean más similares, en algún sentido o bajo algún criterio, y a la vez menos similares a elementos de otros clusters.



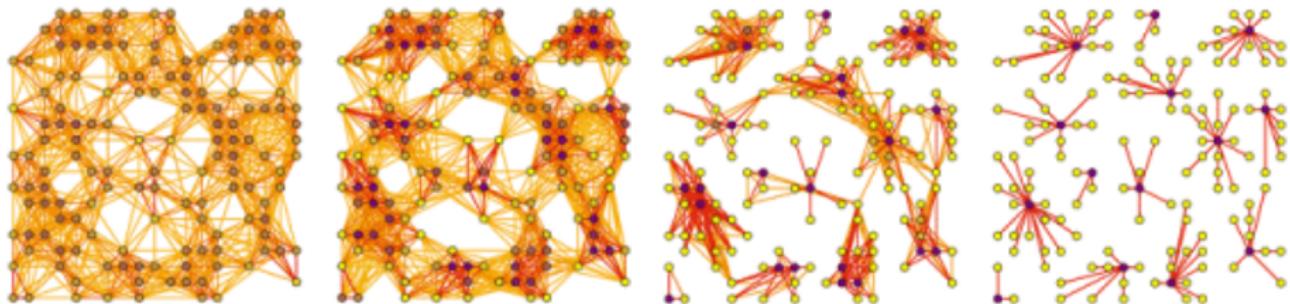
# Agrupamiento basado en grafos

*Estructuras de datos en las cuales los nodos representan entidades, y las aristas representan relaciones.*



# Algoritmo de agrupamiento elegido

Para realizar el agrupamiento de las imágenes se utilizó el algoritmo de Markov (**MCL**).



# Markov clustering

---

- Algoritmo basado en grafos: **simple, rápido y escalable.**
- No requiere instrucciones para ensamblar, unir o dividir grupos.
- No se necesita especificar la cantidad de grupos a obtener.
- Utiliza dos operaciones algebraicas simples en matrices:

① **Expansión:** Multiplicación normal de la matriz.

$$M_i = M_i^e$$

② **Inflación:** Potencia de Hadamard, seguida de una normalización.

$$M_{ij} = M_{ij}^P$$

# Markov clustering

---

Algoritmo:

- Se llena la diagonal con valores 1.
- Se normaliza la matriz.
- Mientras que no converja o llegue al máximo de iteraciones:
  - ① Etapa de expansión.
  - ② Etapa de inflación
  - ③ Normalización de la matriz
  - ④ Se compara matriz actual contra matriz en la iteración anterior.
  - ⑤ Se repite hasta que haya *convergido* o supere la *cantidad máxima de iteraciones*.

# Programación en dispositivos móviles

- Ámbito multimedia con mayor crecimiento en los últimos años.
- Existen lenguajes de programación nativos o *frameworks* multidispositivo que permiten la creación de aplicaciones.
- Lenguajes de programación nativo para desarrollo Android: Java o Kotlin.
- Patrones de arquitectura utilizados: **MVP, MVC, MVVM**.



# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



# Descripción del sistema

- IMachineApp es una **aplicación nativa Android**.
- No necesita establecer una conexión a Internet.
- A partir de un conjunto de imágenes, sugiere una estructura de organización según el grado de similitud o contexto que exista entre ellas.
- Es *open source*.



# Diseño del sistema

---

**La aplicación fue diseñada para que todo el proceso ocurra en 4 etapas:**

- ① Elección del/los directorio/s a procesar.
- ② Procesamiento de las imágenes a través del motor.
- ③ Agrupamiento de las imágenes procesadas.
- ④ Administración del resultado.

# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- **Arquitectura del sistema**
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión

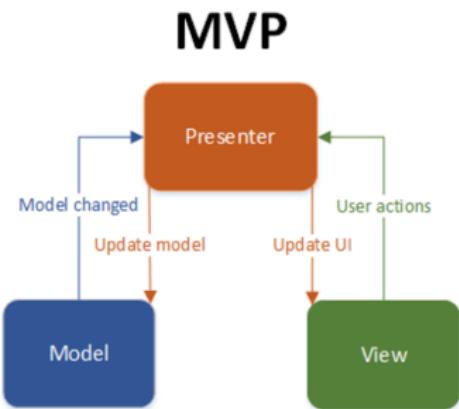


# Arquitectura del sistema

La aplicación se desarrolló siguiendo el patrón de programación MVP:

## Modelo-Vista-Presentador

- **Modelo:** responsable de la lógica de negocio.
- **Vista:** presenta las vistas al usuario e interactúa con el mismo.
- **Presentador:** puente que conecta al modelo con la vista. Además, actúa como instructor para la vista.



# MVP: Justificación

---

- Android no recomienda ninguna forma específica de diseñar u organizar la implementación de aplicaciones.
  - No es una buena práctica cargar de lógica a las actividades que se encargan de interactuar con el usuario.
  - Deberían ser clases por las cuales simplemente la información que interactúa entre el sistema y el usuario fluye.
- ⇒ **Enseñanza:** tomar un tiempo para elegir la arquitectura antes de comenzar con el desarrollo.

# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- **Motor de procesamiento de las imágenes**
- Aplicación Android

## 3 Evaluación de desempeño

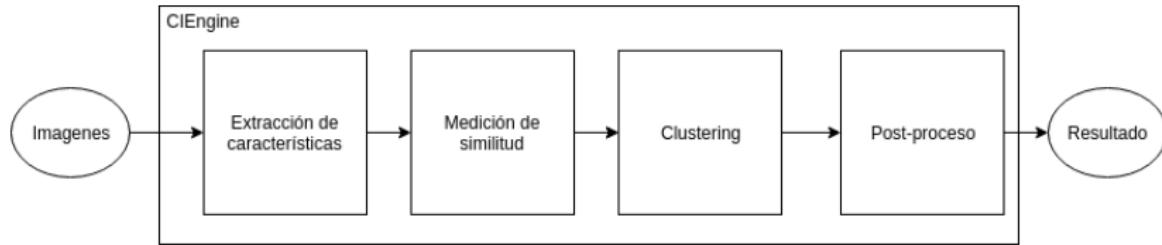
- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



# Motor de procesamiento de las imágenes

- Se denominó **CIEngine**.
- Fue pensado como un módulo independiente.
- Posible adaptación a otros ambientes:
  - ▶ Programa escritorio.
  - ▶ Servicio alojado en la nube.
  - ▶ Aplicación nativa en otros sistemas operativos utilizados en dispositivos móviles.
- Se dividió en 4 etapas:



# Extracción de características

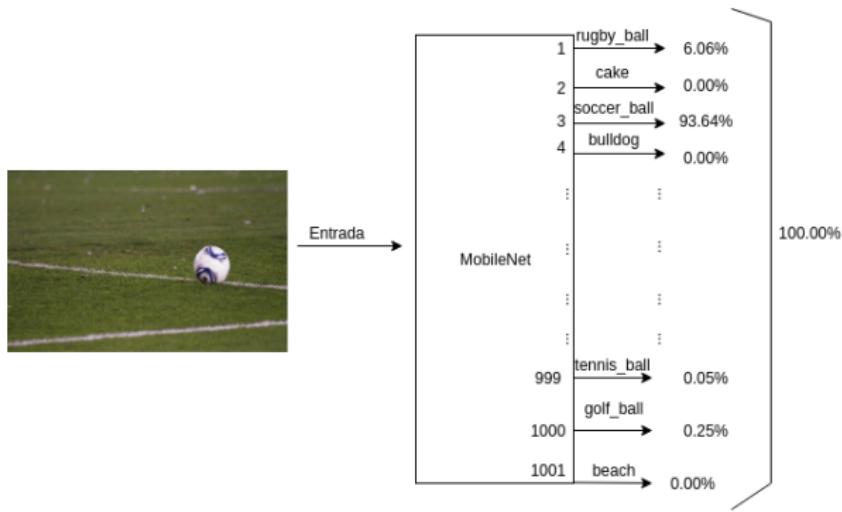
---

- Se utilizó un modelo de **CNN** denominado MobileNet:
  - ▶ Se encuentra preparado y optimizado para ser utilizado en dispositivos móviles.
  - ▶ Puede descargarse una versión pre-entrenada.
  - ▶ Puede ser adaptado según la necesidad del proyecto.
  - ▶ El tamaño final de la red utilizada es de 16.9MB.



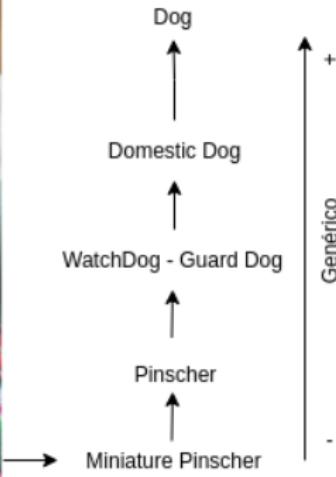
# MobileNet

- Fue entrenada con 1001 grupos pertenecientes a la base de datos **ImageNet**.
- A partir de una imagen de entrada, devuelve la probabilidad de que la imagen pertenezca a alguna de las 1001 clases conocidas.

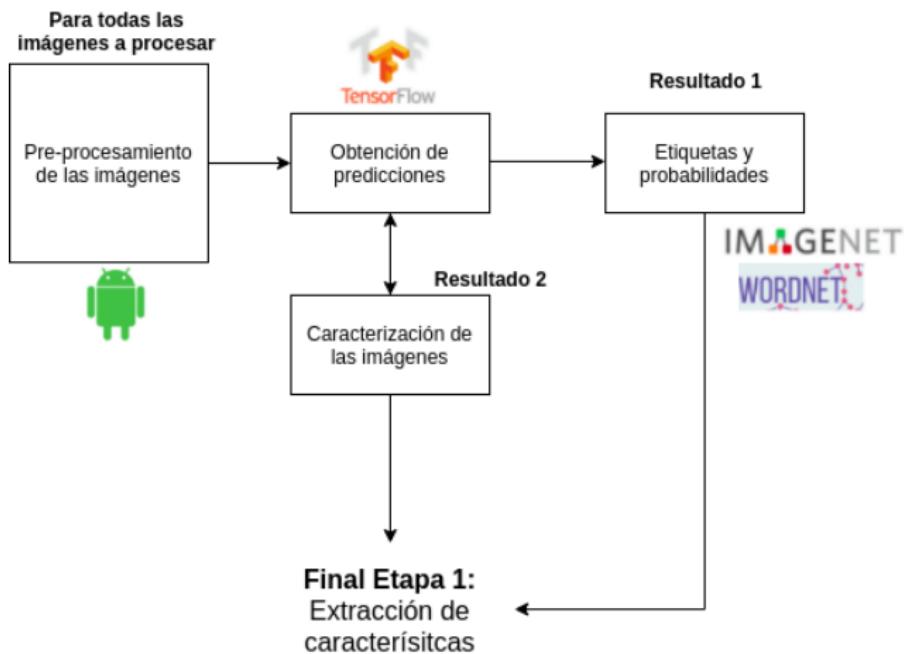


# MobileNet

- Dichos grupos están organizados acorde a una jerarquía de etiquetas, provenientes de una base de datos llamada **WordNet**.
- Jerarquía: parten desde una genérica, hacia el detalle en concreto.



# Etapa 1: pre-procesamiento y extracción de características



# Pre-procesamiento de las imágenes

Para todas las imágenes:

- ① Carga en memoria (bajando su resolución).
- ② Se hace un re-escalado.
- ③ Convierte en un arreglo de bytes.



# Resultado 1: etiquetas y probabilidades

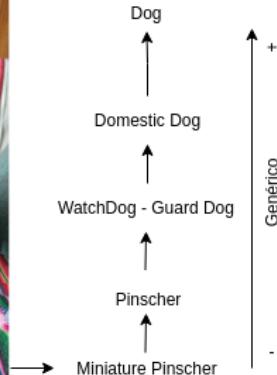
Por cada imagen:

- ① Se obtiene una lista de hasta 5 etiquetas cuyas probabilidades fueron las más altas.
- ② Por cada una de ellas se obtienen hasta 4 etiquetas “padres” según **WordNet**, asignándole la misma probabilidad que la “hija”.



Entrada

MobileNet	
1	rugby_ball → 6.06%
2	cake → 0.00%
3	soccer_ball → 93.64%
4	bulldog → 0.00%
⋮	⋮
999	tennis_ball → 0.05%
1000	golf_ball → 0.25%
1001	beach → 0.00%

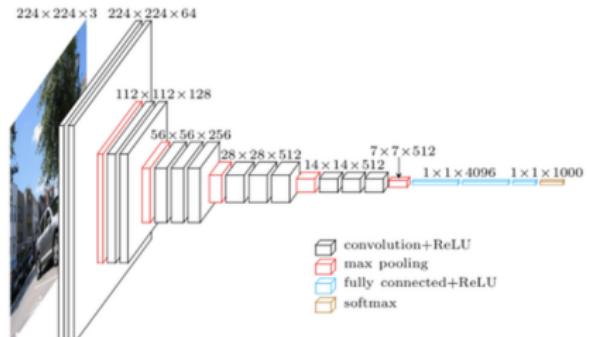
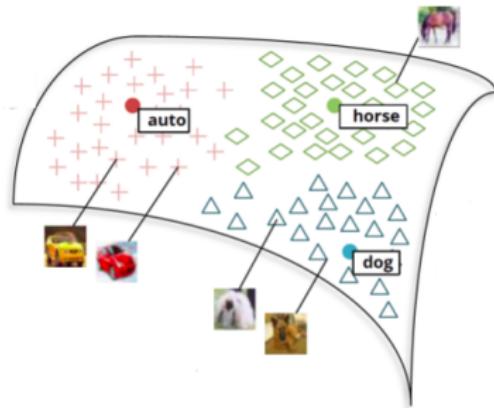


## Resultado 2: caracterización de las imágenes

Por cada imagen:

- Se obtiene el vector de *embeddings*.

Los *embeddings* representan como se activó la última capa antes de realizar la clasificación.



# Resultado final de la etapa 1

---

Por cada imagen se obtiene:

- Un conjunto de etiquetas y probabilidades.
- Un vector caracterizador.

## Etapa 2: Medición de similitud

---

- Se mide el grado de similitud por cada par de imágenes basándose en:
  - ① Las etiquetas y probabilidades de cada una. → **Afinidad gramatical**

## Etapa 2: Medición de similitud

- Se mide el grado de similitud por cada par de imágenes basándose en:
  - Las etiquetas y probabilidades de cada una. → **Afinidad gramatical**

soccer_ball
ball
goldfish
tiger shark
electric ray
beacon
...
beer bottle
barbershop
CD player
joystick
laptop
oxygen mask
pillow
throne
ice cream

soccer_ball	0.733
ball	0.733
ice cream	0.087
joystick	0.0544
electric ray	0.06
beacon	0.051
throne	0.12



soccer_ball	0.733
ball	0.733
goldfish	0
tiger shark	0
electric ray	0.06
beacon	0.051
...	...
beer bottle	0
barbershop	0
CD player	0
joystick	0.0544
laptop	0
oxygen mask	0
pillow	0
throne	0.12
ice cream	0.087

## Etapa 2: Medición de similitud

---

- Se mide el grado de similitud por cada par de imágenes basándose en:
  - ① Las etiquetas y probabilidades de cada una. → **Afinidad gramatical**
  - ② El vector caracterizador de cada una. → **Afinidad según embeddings**

## Etapa 2: Medición de similitud

---

- Se mide el grado de similitud por cada par de imágenes basándose en:
  - ① Las etiquetas y probabilidades de cada una. → **Afinidad gramatical**
  - ② El vector caracterizador de cada una. → **Afinidad según embeddings**
- Se registra en una matriz el grado de similitud entre cada par de imágenes **promediando** las 2 anteriores.

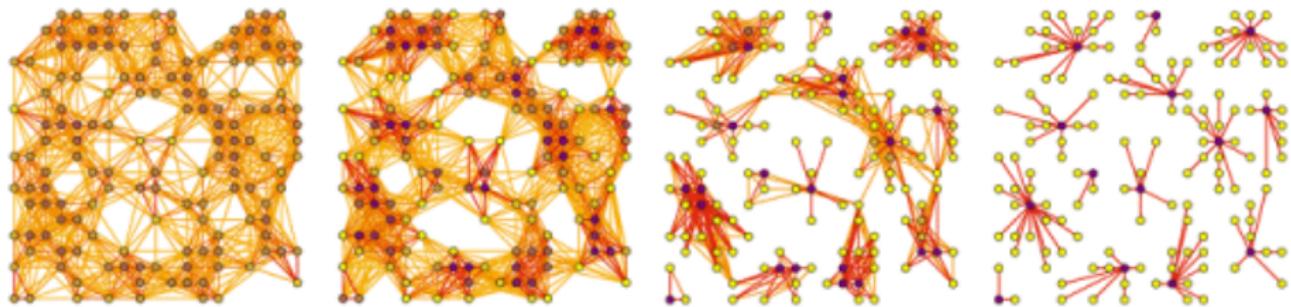
## Etapa 2: Medición de similitud

---

- Se mide el grado de similitud por cada par de imágenes basándose en:
  - ① Las etiquetas y probabilidades de cada una. → **Afinidad gramatical**
  - ② El vector caracterizador de cada una. → **Afinidad según embeddings**
- Se registra en una matriz el grado de similitud entre cada par de imágenes **promediando** las 2 anteriores.

⇒ Resultado: **Matriz de afinidad entre imágenes**

## Etapa 3: Ejecución algoritmo de clustering



⇒ Resultado: **Matriz final de clustering**

## Etapa 4: Post-proceso

---

Debido al origen de muchos grupos con 1 sola imagen, por cada una de ellas:

- Se busca en la matriz de afinidad (previo a la realización del *clustering*) aquella imagen con la que más afinidad comparte, y se la agrega al grupo al cual pertenece la segunda.
- En caso de no existir ninguna imagen con la cual tenga afinidad, se la junta con aquellas imágenes que cumplen la misma condición.

# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



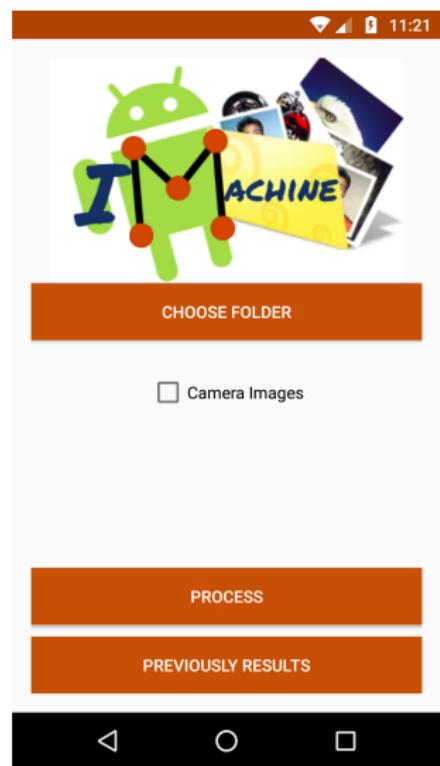
# Demo - Ejemplo de uso

---



# Otras características

- Manejo de excepciones:
  - ▶ Si se presiona el botón procesar sin elegir directorio.
  - ▶ Si se elige ver resultados anteriores y no existen.
  - ▶ Se realiza un control de espacio de almacenamiento disponible.
- Multi-lenguaje: Español e Inglés



# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

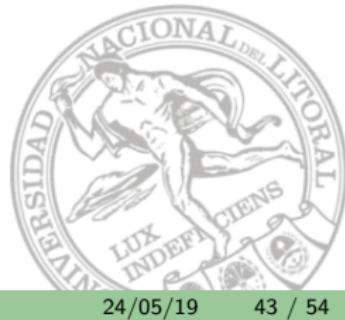
## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

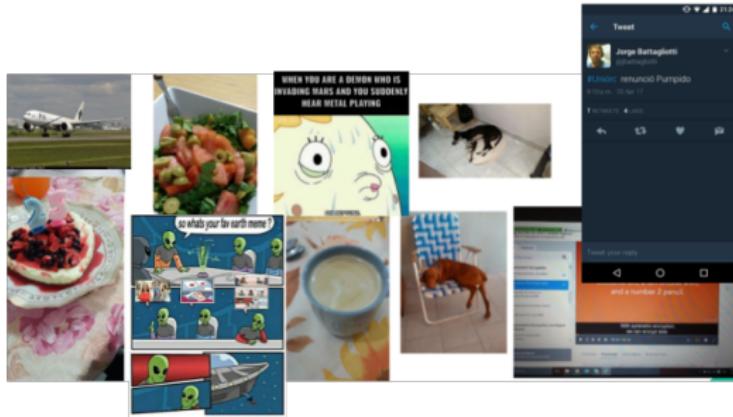
- **Datos usados para la experimentación**
- Protocolo de experimentación

## 4 Conclusiones y discusión



# Datos usados para la experimentación

- Imágenes de WhatsApp del ejecutor del proyecto: 1.600 imágenes.
- Imágenes del portal de Internet 9GAG: 2.000 imágenes.
- Base de datos VOC: 17.000 imágenes



# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

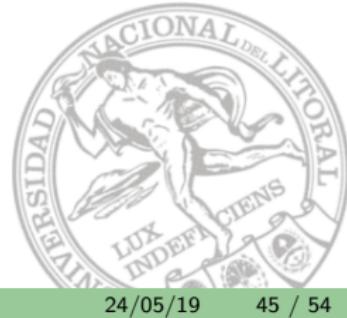
## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



# Protocolo de experimentación

---

- Métricas:
  - ▶ No se espera tener un desempeño específico en términos de calidad de *clustering*, por lo que no fue necesario realizar una comparación con el estado del arte en dicha tarea.
  - ▶ Métricas utilizadas - Clasificación y Clustering:
    - ① Precisión, Recall, F1 Score, Accuracy
    - ② Sorense-Dice, Silhouette
- Desempeño computacional: se midió el tiempo utilizado por cada prueba en diferentes dispositivos móviles:
  - ▶ Motorola Moto G4 PLUS (Gama media/baja)
  - ▶ Xiamoi Redmi Note 4 (Gama media/alta)
  - ▶ Samsung Galaxy S9 (Gama alta)

# Prueba N° 1

Se utilizaron 184 imágenes distribuidas en 6 clases diferentes:

- **Accuracy:** 83.69 %

Categoría	Cant. Ejemplos	Precision[ % ]	Recall[ % ]	F1-Score[ % ]
Avión	25	95.83	95.83	95.83
Computadora	35	82.85	85.29	84.06
Documento	50	86.53	91.83	89.11
Bebida	30	66.67	62.07	64.29
Fiestas	33	88.89	1	94.12
Estadio	11	70	70	70
Macro-averaging		<b>81.79</b>	<b>84.17</b>	<b>82.90</b>

- **Silhouette:** 0.25
- **Sorensen-Dice:** 0.83

# Prueba N° 1

---

<b>Dispositivo</b>	<b>Tiempo de proceso (segundos)</b>
Moto G4 Plus	142.75
Xiaomi Redmi 4	88.31
Samsung Galaxy S9	35.86

⇒ Cuanto mayor es la gama del dispositivo, menor es el tiempo de proceso

# Prueba N° 2

- **Accuracy:** 63 %
- **Silhouette:** 0.16 %
- **Sorensen-Dice:** 0.63 %

Categoría	Cant. Ejemplos	Precision[%]	Recall[%]	F1-Score[%]
Avión	25	57.69	62.50	59.99
Bicicleta	50	56.25	73.46	63.71
Pájaro	35	71.79	82.35	76.71
Bus	30	65.63	72.41	68.85
Automóvil	70	57.40	44.93	50.40
Gato	15	81.25	92.85	86.66
Computadora	51	54.90	56	55.44
Perro	101	76.57	85	80.57
Memes	90	55.55	44.94	49.68
Fiestas	33	51.42	56.25	53.73
Macro-averaging		<b>62.84</b>	<b>67.07</b>	<b>64.59</b>

Dispositivo	Tiempo de proceso (segundos)
Moto G4 Plus	515.38
Xiaomi Redmi 4	335.91
Samsung S9	170.25

# Validación de la aplicación

---

- Se implementó un esquema automatizado de testeo realizando *unit testing*.



- Se realizó un protocolo de validación de todos los módulos que componen la aplicación, con el objetivo de validar que se comporta y actúa según lo esperado.

# Validación de la aplicación

- Se implementó un esquema automatizado de testeo realizando *unit testing*.



- Se realizó un protocolo de validación de todos los módulos que componen la aplicación, con el objetivo de validar que se comporta y actúa según lo esperado.

- Prueba Nº 3

- **Tarea:** Verificar estado de procesamiento de las imágenes durante la ejecución.
    - **Criterio de aprobación:** El sistema muestra en todo momento el grado de avance del procesamiento, calculado en base a la cantidad de imágenes procesadas y la cantidad total de imágenes a procesar.
    - **Estado:** APROBADO
    - **Comentarios:** Durante el procesamiento de las imágenes se puede visualizar el porcentaje de procesamiento realizado en cada momento.

# Agenda

---

## 1 Conceptos básicos

- Introducción
- Fundamentos teóricos

## 2 IMachineApp

- Diseño del sistema
- Arquitectura del sistema
- Motor de procesamiento de las imágenes
- Aplicación Android

## 3 Evaluación de desempeño

- Datos usados para la experimentación
- Protocolo de experimentación

## 4 Conclusiones y discusión



# Conclusiones

---

- Producto de software sobre la plataforma Android que, sin la necesidad de una conexión a Internet y a partir de imágenes almacenadas en el dispositivo, sugiera una estructura de directorios para organizar el conjunto.
- Provee funcionalidades que le permiten al usuario la interacción con el resultado anterior, adecuándolo a sus preferencias.
- Aprendizaje de nuevas tecnologías:
  - ▶ Herramientas para construir la aplicación (e.g. Java, SDK Android)
  - ▶ Patrón de diseño de software: MVP
  - ▶ Algoritmos de *computer vision* y *clustering* implementados (e.g APIs de TensorFlow, MobileNet, ImageNet, MCL).

# Trabajos futuros

---

Se consideran las siguientes cuestiones a mejorar en un futuro:

- Incorporar nuevas características para describir mejor las imágenes.
- Estudiar metodologías para aumentar el número de imágenes a procesar a la vez.
- Reducir el consumo de almacenamiento y memoria.
- Dar un nombre representativo a cada carpeta.
- Extender el uso del motor de procesamiento hacia otras plataformas (e.g Desktop, iOS, Cloud)

# Preguntas

---

¡Muchas gracias por su atención!

