**Tecnológico de Monterrey**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Proyecto Final**

Tecnologías de información emergentes

**Equipo 3**

A01039364 Fernando Romero

A01113373 Alan Velasco

A01282298 Rafael Serna

A01038086 Eduardo Mancilla

A01282769 Daniel Hernandez

A01176380 Katia Tarin

## Recolección de Datos

Para este proyecto nos fue asignada la tarea de entrenar una red neuronal para que pudiera identificar qué tipo de animal le era presentado. Nos dimos a la tarea de seleccionar cinco tipos de animales, los cuales fueron: tortuga, caballo, gato, perro y mono. Una vez definidos, nos dimos a la tarea de recolectar imágenes que correspondieren a los respectivos animales previamente listados. Recolectamos alrededor de 1000 imágenes de cada animal e hicimos uso de la técnica de Data Augment para obtener una mayor variedad de fotos que le fueron proporcionadas a la red como entrenamiento.

En el caso del animal tortuga, las fotos fueron recolectadas del sitio [seaturtle.org](http://www.seaturtle.org/imagelib/), cuyos contribuyentes son miembros de la sociedad quienes suben imágenes de las diferentes especies de tortugas con las que se encuentren. Haciendo uso de la técnica de webscrapping, alrededor de 1600 imágenes pudieron ser extraídas de este sitio.

Las fotos de caballos se obtuvieron del sitio <https://pixabay.com> con el uso de la extension FoxyImage. Pixabay es un sitio en donde la comunidad sube fotos, arte y vectores para uso libre.

Las fotos de los monos también se obtuvieron de pixabay.com, y se descargaron con la extensión FoxyImage, el cambio al tamaño correcto fué con python.

Las fotos de los perros y gatos se obtuvieron del sitio <https://kaggle.com> que facilita varios datasets de diferentes tipos (imágenes, datos, audio) normalmente ya distribuidos en sets de entrenamiento, pruebas y validación.

## 

## Modo de operar del código

Para realizar la red neuronal optamos por hacer uso de la técnica de transfer learning, en la cual, se busca almacenar conocimiento proveniente de la resolución de un problema y hacer uso de este conocimiento para resolver otro problema similar al original. Se optó por hacer uso de la librería fast.ai, tomando como modelo base “resnet50”, incluido en la librería previamente mencionada, la cual cuenta con 50 capas.

La biblioteca fastai estructura su proceso de capacitación en torno a la clase Learner, cuyo objeto une un modelo PyTorch, un conjunto de datos, un optimizador y una función de pérdida; todo el objeto de aprendizaje nos permitirá iniciar la capacitación.

basic\_train define esta clase Learner, junto con el contenedor alrededor del optimizador PyTorch que usa la biblioteca. Define el ciclo de entrenamiento básico que se usa cada vez que se llama al método de ajuste (o una de sus variantes) en fastai.

El método fit ahora se puede llamar especificando las eras que se desean generar. Cuando se llama esto podemos generar un nivel de confianza, o accuracy.

Utilizando las imagenes que estan en la validación, si la red decide por ejemplo, que una tortuga es un caballo con alto confianza, separamos estos casos con la ayuda del método plot\_top\_losses para ajustar la red. Fastai tiene otras herramientas para ajustar como los métodos most\_confused que consigue las imágenes en las que más se equivocó y se puede hacer fit otra vez.

Para comenzar el entrenamiento, se tienen que importar las imágenes base como un objeto de clase ImageDataBunch. Es en este mismo paso en donde se realiza la técnica de Data Augment. Al pasarle como parámetro ds\_tfms la función de get\_transforms(), se asegura que al momento de solicitar batches de las imágenes para entrenar a la red, estas sufran transformaciones lo cual permite obtener mejores resultados que si solo se usaran las imágenes base que se lograron recolectar.

Posteriormente, se crea un instancia del tipo cnn\_learner a la cual se le pasan los parámetros de las imágenes, el modelo a usar y la métrica base con la cual se evaluará la red neuronal, que en este caso se optó por error\_rate. Se hace uso del parámetro recorder para poder conocer el learning rate que mejores resultados da a las capas de la red (Fig. 1) donde se puede apreciar que justo antes de 1e^-01 se dispara el Loss, de modo que se seleccionó el rango de 1e-6,1e-2 para poder entrenar a la red neuronal.

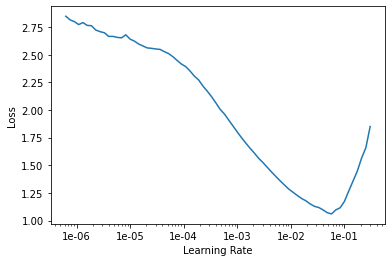


Fig 1: Learning Rate vs. Loss

Tras concluir el entrenamiento de la red neuronal, se obtuvo la siguiente matriz de confusión (Fig. 2), la cual almacena los resultados obtenidos por la red al tratar de categorizar las imágenes proporcionadas en la carpeta de valid.

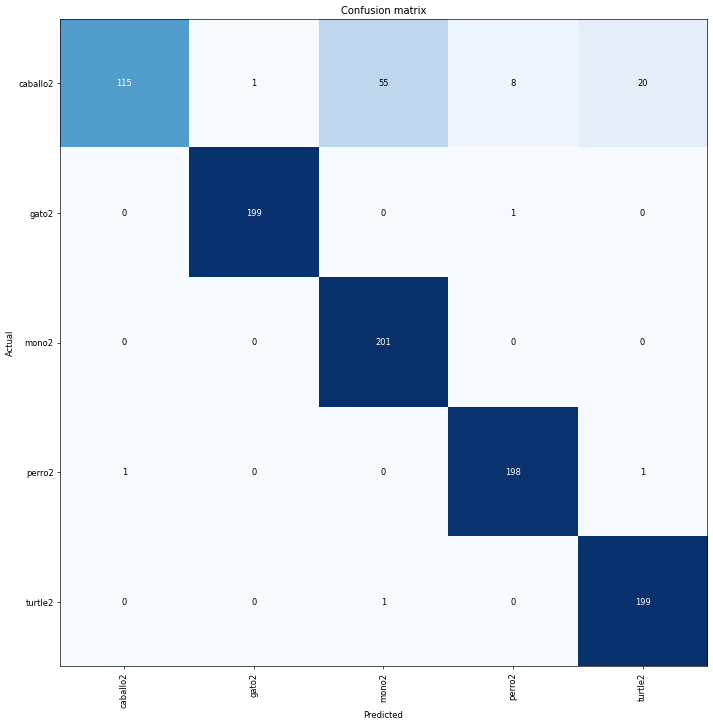


Fig. 2: Matriz de Confusión de la red neuronal

Tal y como se logra apreciar en la Fig. 2, la red no tiene problemas en identificar imágenes de “gatos2”, “monos2”, “perros2” y “turlte2”, siendo la clase “caballo2” la que más se le complica. Tras investigar acerca de este comportamiento, se encontró que era común el que la red tuviera problemas en identificar ciertas categorías. El comportamiento esperado sería que la red neuronal tuviera complicaciones para poder categorizar efectivamente la misma clase, situación que ocurre en este trabajo con la clase “caballo2”, mientras que el resto de las clases son correctamente categorizadas en la mayoría de las ocasiones.