UNIVERSIDAD GALILEO
SP1

Nombres: Erinson Borrayo Carné: 16004336_ Fecha: 25/02/2021

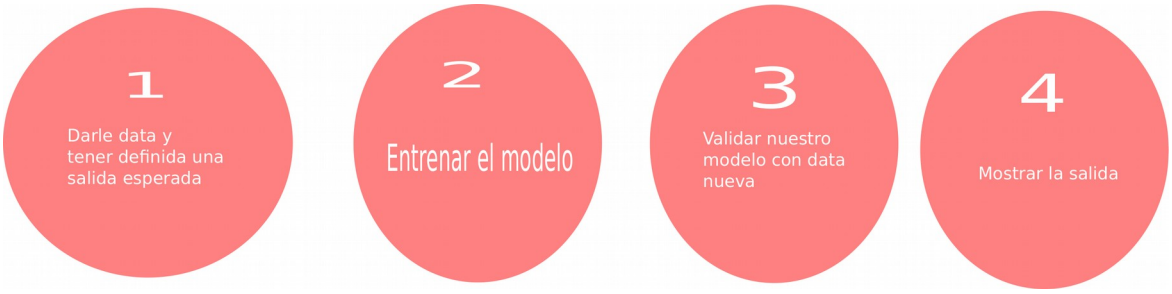
SERIE 1

1. **Explicar en sus propias palabras ¿qué es inferencia?**
   Es un algoritmo que nos sirve para realizar predicciones en el cual vamos a entrenar nuestro modelo pero con data que no esta clasificada o que no tiene etiquetas

2. **¿Cual es la diferencia entre relu, tanh y sigmoid?**
   Sigmoid: Este nos ayuda a retornar una probabilidad entre 0 y 1 lo que la hace mas precsia ya que nuestros rangos no varian mucho y es una funcion no lineal que no centra los resultados en 0
   relu: Es una función que es más rapida para aprender, tiene la ventaja que aprende mucho más rapido y se activa con la entrada y sus valores dejan de ser 0, mientras no se haga una entrada de datos sus valores siempre seran 0.
   tanh: Tiene mayor rango entre -1 y 1 es una funcion de activación que centra sus valores en 0.

3. **Explique los problemas de vanishing y exploding gradient y como puede solucionarse.**
   Vanishing: Este problema se debe cuando el vector gradiente disminuye muy rapido y no se obtienen los valores deseados debido a que toma mucho más tiempo el aprendizaje de la misma.
   Exploding gradient: Es el lado contrario del vanishing ya que este aumenta muy rapido y al igual que el anterior el aprendizaje toma más tiempo debido a que no llegamos al valor que estamos esperando, por lo cual la solución a las mismas es primero, ver por en que punto nuestros datos estan aumentando demasiado (detectar el problema) para que asi podamos realizar cambios en la misma, y poder llegar al valor que hemos definido como deseado.

4. **Realice un diagrama mostrando un pipeline tradicional de Machine Learning y explique cada fase.**



- Definir un problema, ya que no todo tiene que resolverse con ML jajjajaja
- Definir un valor de salida esperado, para asi saber en que momento nuestra red esta bien, y saber en que momento parar, ya que le estaremos dando toda la data para que nuestro algoritmo aprenda todo ya que no se le definen instrucciones sino que el conforme a la marcha va aprendiendo.
- Training: Aqui vamos a definir que esta bien y que no, para que nuestro modelo sepa que hacer.
- New Data: Aca lo que se hace es que se le da nueva data a nuestro modelo, esto para evitar que solo aprenda con los mismos valores y tenga un problema de sobreaprendizaje (tiene un nombre pero se me olvido) y que al darle nueva data nuestro modelo sea malisimo, por lo cual se le entrega nueva data para ver y decir aaaa ok si esta bien mi modelo.
- Output en el cual vamos a realizar las predicciones de nuestro modelo ya sabiendo que el mismo esta bien con nuestros valores esperados.

- **Mencione los tres tipos de aprendizaje en Machine Learning y sus diferencias**
  - Supervised learning, cuando la data que tenemos esta etiquedada y que nos ayuda a resolver problemas de regesión cuando tenemos dos casos o de clasificación cuando tenemos muchos casos.
  - Unsupervised learning, que es cuando la data ya no esta etiqueda, ya no nos sirve para nada, pero que se puede utilizar para resolver problemas de analisis.
  - Aprendizaje reforzado: Es cuando se tienen problemas que tienen multiples estados y donde son problemas más complejos, como detección de imagenes, generar subtitulos en los videos.

- **¿cuándo y por qué se usa binary cross entropy?**
  Se utilizara para resolver problemas de clasificación binario , donde se sabe que solo puede ser dos respuestas al final, ej: cuando el correo es o no spam.

Serie #2

Problema 1

```
model = Sequential([
  Flatten(input_shape=image_size+(3,)),
  Dense(2048, activation='relu'),
  Dense(1024, activation='relu'),
  Dropout(0.2),
  Dense(1, activation='sigmoid')
])
```

- **¿Qué función cumple la capa Flatten?**
  Lo que hace es que convierete nuestra imagen de dos dimensiones a 1
- Si tenemos imágenes de 150x150 RGB, ¿cuál es el valor que recibe input_shape?
  Input_shape=150*150*3
- ¿Qué función cumple la capa Dropout y que significa el parámetro 0.2?
  evitar el overfiting en el entrenamiento y es valor es el learning rate que seria 20%
- ¿por qué la última capa lleva una sola neurona? ¿qué sucedería con el modelo si se usaran 2, 4 o 9?
  Por que es un problema de clasificación binaria, no funcionaria el modelo por el tipo de funcion sigmoid que se esta utilizando

2. Dado el siguiente bloque de código

```
1 model.compile(optimizer='sgd',
2               loss='categorical_crossentropy',
3               metrics=['accuracy'])
4
5 history = model.fit(train_images, train_labels, validation_split=0.3,epochs=10)
6
7 test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=3)
8
9 img_index = np.random.randint(test_images.shape[0])
10 img = test_images[img_index]
11
12 predictions = model.predict(tf.expand_dims(img, 0))
```

- **¿Es un problema de clasificación binaria o un problema de clasificación multiclase? Fundamente su respuesta**
  Como se esta utiliznado la funcion loss='categorical_crossentropy' podemos decir que es un problema de clasificación binario

- **¿Cuál es la diferencia entre las funciones de la línea 7 y 12?**
  .evaluate() evaluando el modelo para asi ver la perdida que se tiene
  .predict() lo que hace es predecir la probabilidad de que una instancia pertenezca a la clase

- Si el modelo presenta overfitting, ¿cómo podría corregirlo? si **presentara underfitting, ¿lo resolvería de la misma forma?**
  Overfitting se presenta cuando entrenamos lo suficiente al modelo pero solo con la misma información, lo que se debe hacer es darle al modelo más información pero variada no de la misma.
  Underfitting: cuando el modelo no aprende nada, se le debe dar más data, para que aprenda más cosas.
  Se podria solucionar para ambos casos dandole más data a diferencia que debe ser data más variada para que aprenda mejor y no solo lo mismo.

- **¿cuántos datos de validación están disponibles para el modelo? Suponiendo que no tengo dataset de validación ¿qué impacto tendría en el entrenamiento del modelo?**
  Una probabilidad de un 0.3, sino tengo algo contra que validar mi modelo no sabra que estoy llegando a algo bueno, ya que podriamos decir que el valor que obtenemos es "correcto" pero no podriamos validar de si en realidad esta bien o no.

```
[ ]  1 train_image_generator = ImageDataGenerator(rescale=1./255)
```

```
     1 train_data_gen = train_image_generator.flow_from_directory(train_dir,
     2                                                            shuffle=True,
     3                                                            batch_size=32,
     4                                                            target_size=IMAGE_SIZE,
     5                                                            class_mode='binary')
```

```
[ ]  1 feature_extractor_url = "https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector/4"
     2
     3 feature_extractor_layer = hub.KerasLayer(feature_extractor_url,
     4                                          input_shape=IMAGE_SIZE+(3,))
```

```
[ ]  1 feature_extractor_layer.trainable = False
```

```
[ ]  1 model = Sequential([
     2   feature_extractor_layer,
     3   Dense(1024, activation='relu'),
     4   Dense(1, activation='sigmoid')
     5 ])
```

```
[ ]  1 model.compile(optimizer='adam',
     2               loss='binary_crossentropy',
     3               metrics=['accuracy'])
```

```
[ ]  1 history = model.fit(
     2     train_data_gen,
     3     epochs=5,
     4     steps_per_epoch=5
     5 )
```

- **¿Por qué es necesario asignar a False la propiedad trainable del feature extractor layer? ¿Qué sucede si se asigna True?**
  Aca lo que hace es que todos los datos que tenemos no cambien, y true cuando no importa que mis valores cambien, creo que esto es más cuando son cosas que van cambiando y que al volver a ejecutar el entrenamiento vamos a obtener diferentes valores.

- **En la celda #5 se define el modelo, la última capa utiliza la función sigmoid ¿por qué? ¿qué sucedería con el modelo si se utiliza ReLu?**
  Por que el modelo no necesita muchas neuronas para poder obtener resultados correctos.

  Si lo cambiamos con relu no vamos tener el problema que nuestro gradiente desaparezca, lo que pasara es que nuestas represetnaciones van a quedar más.

- **¿Por qué se utiliza la variable train_data_gen cómo parámetro de la función model.fit?**
  Por que lo que se va a obtener es un registro de lo que contiene ese variable a lo largo del entrenamiento.

- De una manera global y general explique el código de estas siete celdas. Su respuesta debe incluir al menos detalle de las siguientes interrogantes
  - ¿qué técnica de Deep Learning está siendo aplicada?
  - ¿qué tipo de problema es (regresión, clasificación) ?
  - ¿por qué cada uno de los bloques está en ese orden y porque es necesario cada uno de ellos?
  - ¿qué problema aplicado podría estar revolviéndose?

Lo que se esta tratando de resolver es como se comporta a lo largo del tiempo el entrenamiento, donde vamos a obtener un set de imagenes y lo vamos a clasificar entre dos tipos, para eso estamos uitlizando transferlearning, el orden que se esta utilizando es por que primero necesito entrenar mi modelo para asi ver con el .fit como se comporta a lo largo del entrenamiento ya que no podria yo validar esto sin antes entrenarlo, y tampoco podria hacerlo sin antes obtener el set de imagenes que voy a utilizar.