Reto Whale & Jaguar:

Clasificación de imagenes por sentimientos

Integrantes:

Samir José de La Cruz Palmera

sj.delacruz@uniandes.edu.co

Departamento de Física,

Universidad de Los Andes.

Angélica Herrera Alba

a.herrera1@uniandes.edu.co

Departamento de Física,

Universidad de Los Andes.

1. Definición del proyecto

1.1 Definición del problema

Whale & Jaguar es una compañía colombiana compuesta por un equipo de cientificos de datos, ingenieros de machine learning, desarrolladores web, especialistas en consultoría, entre otros. Nacieron con el objetivo de maximizar el alcance y potencial del cliente en cuestión (empresas), al analizar datos tomados de los entornos digitales por medio de algoritmos de inteligencia artificial (IA). Han desarrollado varios proyectos, entre los que se encuentran 1. el sismógrafo electoral, el cual fue una herramienta creada por la compañía para analizar la acogida de diferentes temas y opiniones en las elecciones presidenciales de 2018, y 2. la Mai Fina, la cual es un bot creado con IA que crea canciones de reggaeton, aprendiendo de una base de datos de 7000 canciones.

El reto que fue propuesto consistía en clasificar las imagenes del siguiente dataset: https://data.world/crowdflower/image-sentiment-polarity de acuerdo al sentimiento, existiendo la clasificación de: Highly Positive, Positive, Neutral, Negative and Highly Negative. La empresa ya ha analizado el sentimiento y reacción de las personas en el ámbito digital, pero por medio del análisis de texto plano encontrado en diversas redes sociales, y por medio de este reto buscan ampliar su alcance incorporando algoritmos que

permitan la clasificación de imagenes de acuerdo a la reacción emocional que generan o que presentan.

1.2 Descripción de la solución y los algoritmos

Para la solución de este reto, se decidió usar transfer learning, y reentrenar dos modelos distintos de redes neuronales convolucionales. Los modelos implementados fueron Redes Residuales (ResNet), del módulo torchvision.models. Se usaron las versions 152 y 18 de ResNet. Se empezó implementando esta solución con la versión 152 de ResNet y con los siguientes números de datos de entrada:

Positive	1710
Neutral	845
Negative	423
Highly positive	86
Highly negative	56

Para ResNet 18 se usaron 158 datos de entrada por cada clase. Una vez terminado el entrenamiento de la red 152 y al observar sus resultados, se decidió probar con la 18, y al hacer esto se encontraron estrategias útiles. Primeramente, al hacer uniforme estos datos de entrada (158 por cada clase), se logra que conseguir uniformidad en función de loss del modelo, lo cuál evita que los batch estén sesgados.

Para ambos modelos de ResNet, se deben redimensionar los datos de entrada, tanto para el valid como para el train sets, pero para ResNet 152 adicionalmente se deben pasar las imagenes a escala de grises, lo cuál le quita muchas características que el modelo podría identificar para la clasificación por clases. Por esta razón también se decidió probar con otro modelo.

Los datos fueron redimensinados así para el ResNet 18:

```
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
    ]),
}
```

Y para el ResNet 152 así:

```
DATA_TRANSFORM = transforms.Compose([
                 transforms.Resize((224, 224)),
                 transforms.Grayscale(3),
                 transforms.ToTensor(),
                 transforms.Normalize(mean=[0.485, 0.456, 0.406],std=[0.229, 0.224, 0.225])
                 ])
```

Los modelos fueron reentrenados añadiendoles una última capa con los datos del dataset cargado, de la siguiente manera para ResNet 18:

```
out_dim = 5
model.fc = torch.nn.Sequential(
    torch.nn.Linear(model.fc.in_features, int(model.fc.in_features/2)),
    torch.nn.Tanh(),
    torch.nn.Linear(int(model.fc.in_features/2), int(model.fc.in_features/4)),
    torch.nn.Tanh(),
    torch.nn.Linear(int(model.fc.in_features/4), out_dim)
    )
model.load_state_dict(model.state_dict())
```

Y para ResNet 152:

```
out_dim = 5
model.fc = torch.nn.Sequential(torch.nn.Linear(model.fc.in_features, out_dim))
model.load_state_dict(model.state_dict())
```

Para ResNet 18, se usaron 300 épocas guardando 2 datos por época (2 de loss, de accuracy y de F1 score), mientras que para ResNet 152 se usaron 40 épocas guardando 1 dato por época. Los parámetros ópticos encontrados fueron: EPOCHS=300, LEARNING_RATE = 1e-3, WEIGHT_DECAY = 1e-8.

Para ResNet 152 los parámetros óptimos encontrados fueron EPOCHS=20, LEARNING_RATE = 0.01, WEIGHT_DECAY = 0.00.

Los scripts de código se pueden encontrar en el siguiente repositorio: https://github.com/sj0delacruz/Reto_WJ

1.3 Pré Requisitos

Se necesita el dataset mencionado al inicio, pero por medio del .ipynb creado, la descarga se realiza por medio de la primera casilla de código, y se observa la cantidad de imagenes que están clasificadas por los valores mecionados. Código de descarga y muestra de cantidad de datos a usar para ResNet 152:

```
import pandas as pd
df = pd.read_csv('https://query.data.world/s/j7zymbjelxlf4v5meujq32s27bbirf')
labels=['Positive', 'Neutral', 'Negative', 'Highly positive', 'Highly negative']
percent=0.2
sentiment_counts=df['which_of_these_sentiment_scores_does_the_above_image_fit_into_best'].value_counts()
for i in range(len(labels)):
  df[df['which_of_these_sentiment_scores_does_the_above_image_fit_into_best']==labels[i]]=df[df['which_of_these_sen
labels=['Highly positive', 'Positive', 'Neutral', 'Negative', 'Highly negative'][::-1]
df['which_of_these_sentiment_scores_does_the_above_image_fit_into_best'].value_counts()
```

Python

```
Positive          1710
Neutral            845
Negative           423
Highly positive     86
Highly negative     56
Name: which_of_these_sentiment_scores_does_the_above_image_fit_into_best, dtype: int64
```

Ejemplo de algunas de las imagenes del dataset:

2. Evaluación de los experimentos

2.1 Descripción de los datos de entrada

El dataset contenía columnas relacionadas con la clasificación de sentimientos, otra con los links de las imagenes, otra columna con el numero de clasificaciones correctas, entre otras. Primeramente, se crearon los directorios donde se guardarían las imagenes, y seguidamente se descargaron las imagenes de la columna 'imageurl' usando threds por eficiencia. Todo esto se muestra a continuación:

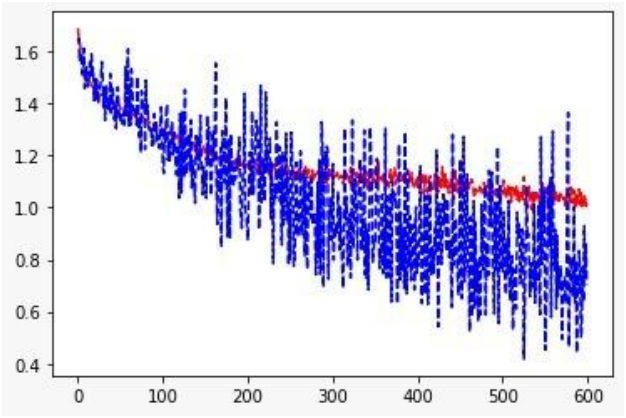
```python
import threading
import wget

def download_img(url,directory,label,i):
  try:
    wget.download(url, '/content/images/'+directory+'/'+label+'/img'+i+'.jpg', )
  except:
    pass
for i, label in enumerate(labels):
  urls = df[df['which_of_these_sentiment_scores_does_the_above_image_fit_into_best']==label]['imageurl']
  n1_split=int(len(urls)*0.8)
  n2_split=int(len(urls)*0.95)
  n=0
  for j, url in enumerate(urls):
    if n<n1_split:
      directory=directories[0]
    elif n<n2_split:
      directory=directories[1]
    else:
      directory=directories[2]
    threading.Thread(target=download_img, args= (url,directory,str(i),str(j),)).start()
    n+=1
```

Como se ve en la imagen, fue divido el dataset entre las imagenes que tuvieran el mismo sentimiento reportado en este, y luego clasificadas entre training, valid y test sets. Esta clasificación se realizó con porcentajes de 80%, 15% y 5% respectivamente, y de manera uniforme entre los 5 sentimientos.

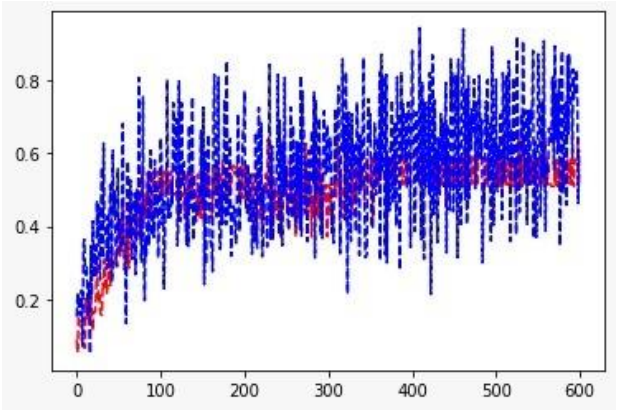
2.2 Resultados

Usando ResNet 18: curva azul: train set curva roja: valid set

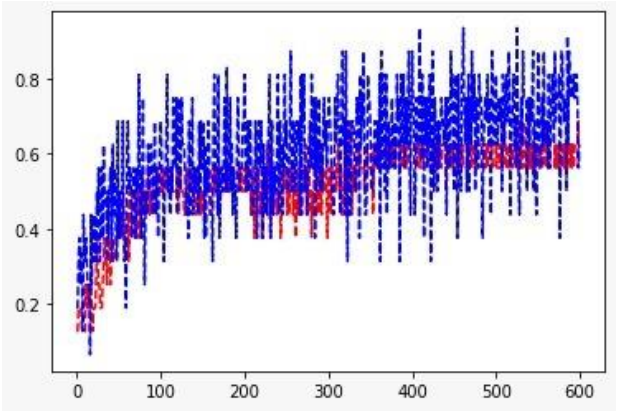
Grafica del loss:



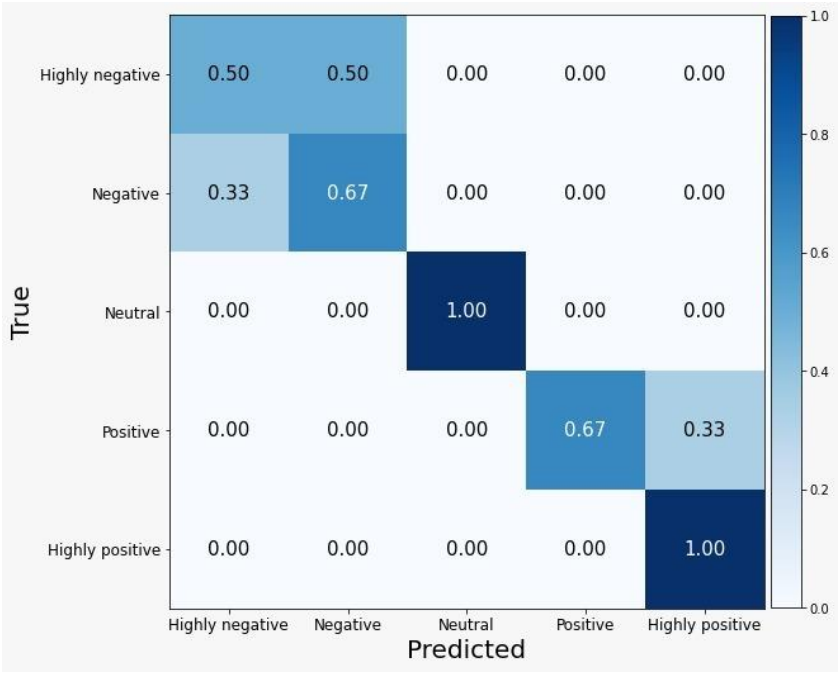
Grafica del F1:



Grafica del Accuracy:



Matriz de confusión:

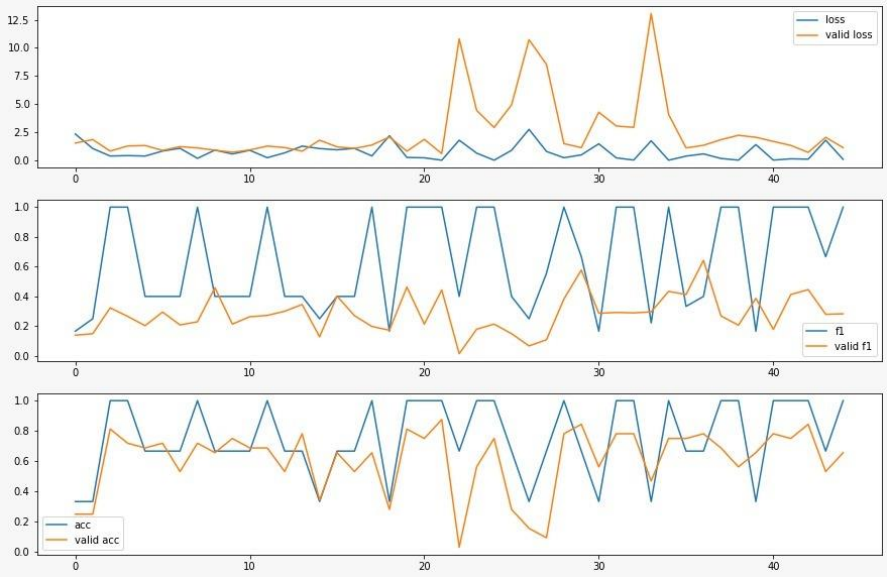


Los resultados finales fueron: Loss:0.72, F1:0.75, Acc:0.75, usando los siguientes parámetros: EPOCHS=300, LEARNING_RATE = 1e-3, WEIGHT_DECAY = 1e-8, usando estos valores de entrada:

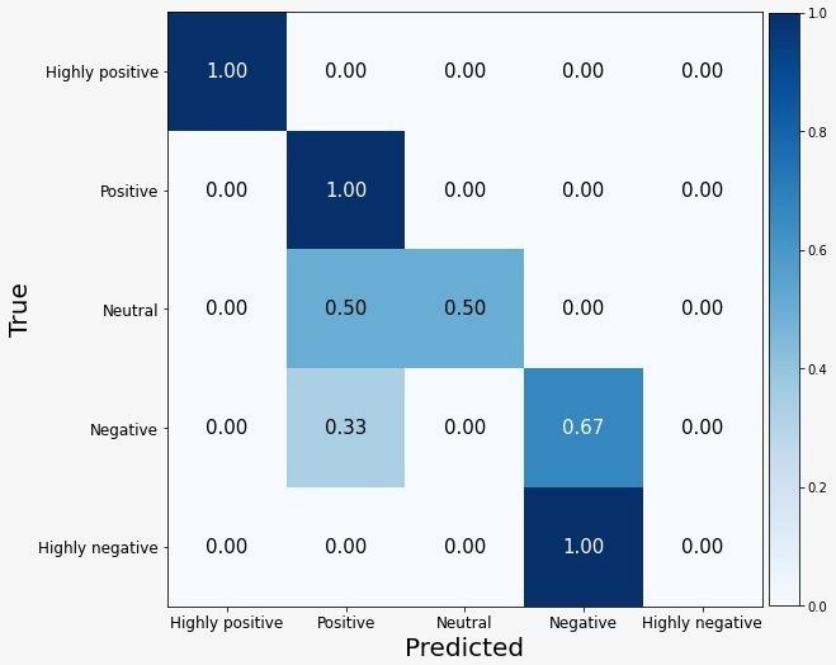
```
Positive            283
Highly positive     283
Negative            283
Highly negative     283
```

Usando ResNet 152:

Loss, F1 y accuracy:



Matriz de confusión:



Los resultados finales fueron: Loss:0.63, F1:0.67, Acc:0.88 usando los siguientes parametros: EPOCHS=20, LEARNING_RATE = 0.01, WEIGHT_DECAY = 0.005

Bibliografía

https://whaleandjaguar.co/

Zhang, A., Lipton, Z. C., Li, M., Smola, A. J. (2021). Dive into deep learning. arXiv preprint arXiv: 2106.11342.