Noviembre 2023

Tecnológico de Monterrey

Implementación de un Modelo de Deep Learning

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Querétaro

Inteligencia Artificial Avanzada para la Ciencia de Datos II

TC3007B.501

Presenta:
Ricardo Cáceres | A01706972

Índice

Introducción

Las emociones juegan un papel crucial en la interacción humana y, por ende, el reconocimiento de emociones en imágenes se ha vuelto esencial para diversas aplicaciones.

En este proyecto, se aborda el reconocimiento de emociones en distintas imágenes a través de un modelo de Deep Learning que sea capaz de distinguir entre una imagen que muestra una emoción de enojo, con una que muestra felicidad y con otra que muestra tristeza.

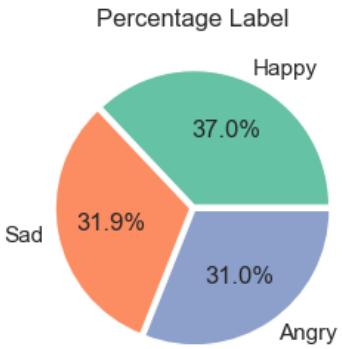
El propósito principal de este proyecto es diseñar y entrenar una red neuronal profunda capaz de diferenciar entre emociones como el enojo, la felicidad y la tristeza. Este enfoque se enmarca en el interés de modelar la expresión emocional humana a través de imágenes, lo que puede tener un impacto relevante en diversas aplicaciones.

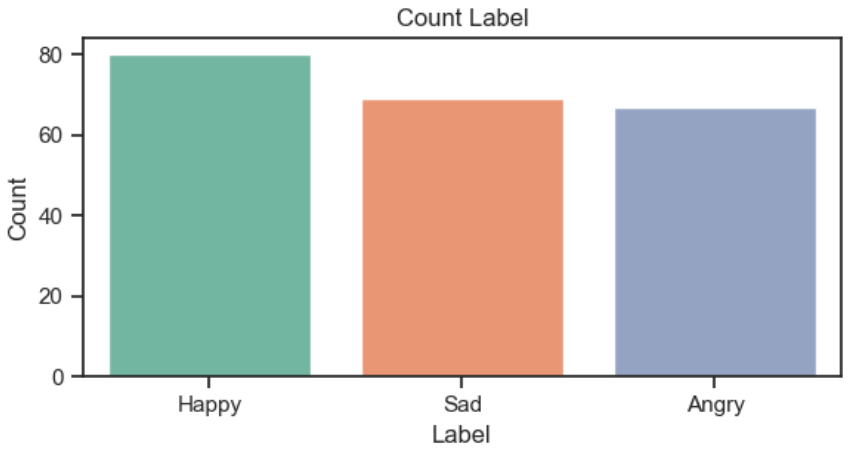
Descripción del Conjunto de Datos

El conjunto de datos utilizado en este proyecto es esencial para la correcta formación y evaluación de nuestro modelo de reconocimiento de emociones. Este conjunto de datos se organiza de manera jerárquica en una carpeta principal denominada “data”, que contiene tres subcarpetas principales, cada una representando una emoción específica: “Happy”, “Sad”, “Angry”.

- Happy:
    - Esta carpeta alberga imágenes que representan expresiones asociadas con la emoción de la felicidad. Estas imágenes capturan momentos felices, con gestos más evidentes de alegría.
- Sad:
    - Esta carpeta contiene imágenes que reflejan expresiones vinculadas a la tristeza. Aquí se encuentran manifestaciones más pronunciadas de pensar, colores grises y oscuros.
- Angry:
    - Esta carpeta tiene imágenes que representan emociones de enojo. Desde colores fuertes, hasta gestos intensos.

Cabe destacar que cada imagen está etiquetada correctamente gracias a su path con su emoción correspondiente. Por ejemplo: “data\Angry\1564075.jpg”.
El conjunto de datos está distribuido de la siguiente manera:





De la misma manera, el tamaño y la resolución de las imágenes se mantiene uniforme para garantizar una entrada coherente al modelo. En este caso las imágenes tienen un formato de 224x224 y 3 canales (RGB).

El código implementado divide los datos en tres conjuntos: train, validation, y test. La primera división asigna el 90% de los datos a entrenamiento y el 10% a prueba. Luego dentro del conjunto de entrenamiento, se realiza una segunda división, asignando 80% para entrenamiento y el 20% para validación. Esto nos permite un buen entrenamiento, ajuste de hiperparametros en el conjunto de validation y testing en datos independientes con el conjunto de test.

Descripción del Modelo

La arquitectura inicial del modelo se basó en la red neuronal pre entrenada de VGG16, (Transfer Learning, ImageNet en este caso), conocida por su capacidad de tareas de clasificación de imágenes. Sin embargo, se añadieron capas adicionales adaptadas a nuestro problemas en específico.

```
# Crear un modelo secuencial para agregar capas adicionales sobre el modelo VGG16
vgg_model = Sequential([
    base_vgg, # Agregar el modelo VGG16 como la primera capa del nuevo modelo
    Flatten(), # Aplanar la salida de VGG16
    Dense(128, activation='relu'), # Agregar una capa densa (fully connected) con 128 neuronas y función de activación ReLU
    Dropout(0.4), # Agregar una capa de dropout con una tasa del 40% para evitar el sobreajuste
    Dense(3, activation='softmax') # Agregar una capa densa de salida con 3 neuronas y función de activación softmax (clasificación multiclase)
])
```

Se creó un modelo secuencial para poder agregar capas adicionales al modelo VGG16.

Luego se aplicó una capa Flatten para convertir la salida tridimensional de VGG 16 en un vector unidimensional, que se utiliza como entrada para las capas densas.

La capa Densa cuenta con 128 neuronas y una función de activación ReLU, la elección de 128 neuronas se basa en la necesidad de tener suficientes parámetros para capturar patrones relevantes sin aumentar excesivamente la complejidad del modelo.

La capa de Dropout del 0.4 (40%) nos ayuda a prevenir el sobreajuste al evitar que el modelo dependa demasiado de ciertas neuronas durante el entrenamiento. Apaga el 40% de las neuronas durante el entrenamiento.

Y finalmente está la capa Densa de salida con 3 neuronas y una función de activación softmax. Esta es la capa que proporciona las predicciones del modelo (las cuales son 3 clases). La función softmax convierte las puntuaciones de salida en probabilidades, facilitando la interpretación de la salida como la probabilidad de pertenecer a alguna clase.

Regularización

De la misma forma, se ocuparon distintas técnicas de regularización durante el entrenamiento. Estas técnicas de regularización consisten en hacer transformaciones aleatorias a las imágenes. En este caso se aplicaron las siguientes transformaciones:

- **width_shift_range = 0.15**: Desplazamiento horizontal aleatorio de hasta el 15% del ancho de la imagen.
- **height_shift_range = 0.15**: Desplazamiento vertical aleatorio de hasta el 15% de la altura de la imagen.
- **vertical_flip** = True: Volteo vertical aleatorio de la imagen.
- **horizontal_flip** = True: Volteo horizontal aleatorio de la imagen.
- **rescale** = 1/255: Normalización de los valores de píxeles dividiendo cada valor por 255.

Modelo Anterior

En el primer modelo que se elaboró, la función de pérdida “categorical cross entropy”, optimizador “adam” y métrica de “accuracy”. Y el conjunto de datos fue entrenado en 5 lotes y por 20 epochs.

```
# Entrenar el modelo con el conjunto de entrenamiento y validación
history_vgg = vgg_model.fit(train_set,
                                steps_per_epoch=len(train_set), # Pasos por época (número de lotes)
                                validation_data=val_set, # Conjunto de validación
                                validation_steps=len(val_set), # Pasos de validación (número de lotes)
                                epochs=20,
                                verbose=1)
```

En el conjunto de prueba, los resultados son los siguientes:

- Pérdida (Loss): 1.62. Con este valor podemos observar cómo se desvían las predicciones del modelo de las etiquetas reales en el train_set. Sería preferible que el modelo tuviera una pérdida más baja, este valor de la pérdida nos sugiere una mayor discrepancia entre las predicciones y las etiquetas reales.

- Precisión (Accuracy): 55%. La precisión nos muestra la proporción de predicciones correctas en relación con el total de muestras en el test_set. En este modelo, el 55% de las predicciones son correctas. Prácticamente el modelo hace predicciones correctas 1 vez, y luego hace predicciones incorrectas.

Dadas las métricas anteriores, se entrenó el modelo durante más épocas para que lograra mejorar las predicciones.

Modelo Mejorado

Este nuevo modelo pasa de 20 épocas a 30 épocas en el entrenamiento, la función de pérdida “categorical cross entropy”, optimizador “adam” y métrica de “accuracy” se mantiene igual. Y en esta versión final del modelo, el conjunto de datos fue entrenado en 5 lotes y por 30 epochs.

Métricas

Los resultados fueron los siguientes:

Tras haber hecho el entrenamiento se obtuvieron los siguientes resultados:

- La pérdida (loss) es 0.5298, indicando cuánto se desvían las predicciones del modelo de las etiquetas reales.

- La precisión (accuracy) es 78.06%, lo que significa que el 78.06% de las predicciones del modelo en el conjunto de entrenamiento son correctas.

Estos valores sugieren que el modelo tiene un rendimiento razonable en el conjunto de entrenamiento. Sin embargo, es importante evaluar el modelo en un conjunto de datos de prueba independiente para validar su capacidad de generalización.

En el conjunto de prueba, los resultados son los siguientes:

- Pérdida (Loss): 0.68. Este valor indica cuánto se desvían las predicciones del modelo de las etiquetas reales en el conjunto de datos de prueba. Una pérdida más baja sería preferible, pero el valor de 0.68 proporciona información sobre la discrepancia entre las predicciones y las etiquetas reales.

- Precisión (Accuracy): 64%. Esta métrica muestra la proporción de predicciones correctas en relación con el total de muestras en el conjunto de datos de prueba. En este caso, el 64% de las predicciones del modelo en el conjunto de prueba son correctas.

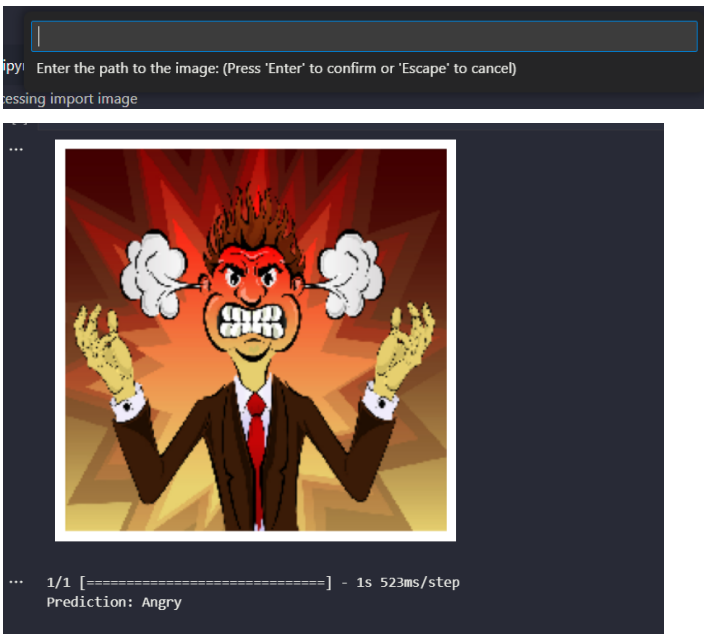
Predicciones con el Modelo

Se hicieron predicciones con el conjunto de pruebas y se obtuvo lo siguiente:

	precision	recall	f1-score
Angry	0.29	0.40	0.33
Happy	0.86	0.75	0.80
Sad	0.75	0.67	0.71

Una representación gráfica de los resultados de predicción del modelo es la siguiente:

Cabe destacar que el modelo se guardó para hacer predicciones con el modelo entrenado. El código del archivo titulado “userTest.ipynb” es una manera interactiva de hacer predicciones con el modelo entrenado en nuevas imágenes.



De la misma manera el código en el archivo titulado “app.py” crea un servidor en Flask donde se despliega una interfaz gráfica para mayor facilidad de subir imágenes y hacer predicciones.