CLASIFICACIÓN (RECONOCIMIENTO) DE IMÁGENES DE FRUTAS Y VERDURAS

MANUELA GUTIÉRREZ CANO

UNIVERSIDAD DE ANTIOQUIA
FUNDAMENTOS DE DEEP LEARNING
FACULTAD DE INGENIERÍA
MEDELLÍN
2023

1. **Contexto de aplicación**

   El problema que se desea abordar trata de la clasificación de imágenes de frutas y verduras, el cual es útil cuando se tiene un volumen muy grande de imágenes y se requiere conocer a qué clase pertenece cada imagen por el nombre que recibe el objeto que está en una imagen.

2. **Variable objetivo de Machine Learning**

   La variable que se va a predecir es el nombre o la clase del objeto principal que se encuentra en cada una de la imágenes de frutas y verduras, por ejemplo si en una imagen hay una manzana la etiqueta o clase que el modelo de Deep Learning debería predecir es la clase 'manzana', y si en la imagen hay un objeto central que es una lechuga, la clase que se debe predecir es 'lechuga'.

3. **Dataset**

   Los datos constan de un conjunto de imágenes que están divididas en tres conjuntos a su vez que son: train (3115 imágenes pertenecientes a 36 clases), test (359 imágenes pertenecientes a 36 clases) y validation (351 imágenes pertenecientes a 36 clases). De lo anterior se dice que hay 36 clases distribuidas en frutas y verduras.

4. **Métrica de desempeño (de Machine Learning y de negocio)**

   Como medida de desempeño se tiene la medida de precisión o de accuracy de los datos, se desea conseguir a partir del entrenamiento de los modelos de Deep Learning un accuracy de al menos el 80% (0.8) que presente en las etapas de entrenamiento y de validación de los modelos.

5. **Referencias y resultados previos**

   Para este conjunto de datos en especial ya se han hecho varias aproximaciones de modelos para la clasificación de las imágenes y se han obtenido buenos resultados respecto a la métrica de accuracy concierne,

accuracies de más del 80% con modelos cuya arquitectura es muy compleja y también con modelos cuya arquitectura es más simple. Las arquitecturas están compuestas por capas desde la capa completamente conectada (densa), utilizando también capas para evitar el sobreajuste (drop) y también capas de convolución, capas de flatten, y la capa de salida en muchos trabajos previos ha sido una capa densa con sus respectivos parámetros. Cada trabajo realizado ha incluido ciertos valores de parámetros que no se han ajustado, pero que al asignarlos a cada una de las capas en cuestión brinda un buen desempeño que en muchas ocasiones fue mejor del esperado. Asimismo se ha hecho el entrenamiento de varios modelos mucho más complejos utilizando la técnica de Transfer Learning brindando resultados muy similares a los anteriormente mencionados.

6. Descripción de la estructura del notebook

El notebook llamado *Deep Learning Project* está compuesto por la carga de los datos que están disponibles en la página de *Kaggle* https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition/code?resource=download, seguido de las etapas de exploración y procesamiento de los datos en donde se configuran los conjuntos de datos (train, test y validation) para obtener un conjunto de cada uno con el fin de poder ingresarlos a los modelos de Deep Learning y que estos tengan la capacidad de predecir o generalizar correctamente, a continuación se muestra la estructura de los datos, las clases en las que están distribuidos, así como también la cantidad de clases que hay, la cantidad de muestras o imágenes que existe por conjunto de datos, y también se muestran algunas imágenes seleccionadas aleatoriamente del conjunto de datos de entrenamiento con el fin de poder visualizar cómo es que está conformado dicho conjunto. Luego sigue la etapa de modelación del problema que es en donde se puede ver varias arquitecturas que pueden modelar los datos para alcanzar el nivel deseado de exactitud, dichos modelos están compuestos por varias capas que con un trabajo conjunto producen buenos resultados, y por último se encuentra algunos modelos de Transfer Learning que son útiles para predecir las imágenes nuevas que llegan en el futuro.

7. Descripción de la solución

7.1 Exploración y preprocesado

Como se mencionó previamente, existen tres subconjuntos de datos en este problema que son de entrenamiento, evaluación y validación, cada uno de los cuales tiene su propia cantidad de imágenes y de clases. En el conjunto de entrenamiento se encontró que está compuesto por 3115 imágenes y 36 clases, el de evaluación tiene 359 imágenes y 36 clases y el de validación tiene 351 imágenes y 36 clases.

Las clases son: ['cabbage', 'mango', 'peas', 'sweetcorn', 'carrot', 'turnip', 'potato', 'chilli pepper', 'bell pepper', 'capsicum', 'tomato', 'jalepeno', 'kiwi', 'pineapple', 'spinach', 'banana', 'grapes', 'beetroot', 'raddish', 'soy beans', 'apple', 'sweetpotato', 'onion', 'ginger', 'garlic', 'lemon', 'paprika', 'cauliflower', 'lettuce', 'corn', 'eggplant', 'watermelon', 'orange', 'pear', 'cucumber', 'pomegranate'], que corresponden a diversas frutas y verduras.

Parte del conjunto de datos de entrenamiento se puede observar como sigue:

Figura 1. Conjunto de entrenamiento parcial

Las imágenes anteriores están en RGB (tres canales) y tienen dimensión de (64,64,3). Los tres canales presentes en las imágenes hace que estas tengan mejor visibilidad al momento de analizarlas.

7.2 Modelación, arquitectura y resultados

Para solucionar este problema se entrenó un modelo cuya arquitectura está compuesta por:

- 2 capas de convolución con 32 filtros, cada filtro con tamaño 3*3, la primera con *padding same,* con funciones de activación *relu* y con *input shape* (64,64,3).
- Una capa *MaxPool* con *pool_size* de 2 y *strides* 2.
- Una capa *Dropout* con tasa de 0.25 que ayuda a evitar el sobreajuste de los datos.
- 2 capas de convolución con los mismos parámetros antes mencionados, salvo el número de filtros de 64.
- Otra capa de *MaxPool* y de *Dropout* con los parámetros ya mencionados.
- Una capa *Flatten* que aplana los datos de entrada.
- 2 capas densas con 512 y 256 unidades y activación *relu*.
- Una capa *Dropout* con tasa de 0.5.
- Una capa densa que es la capa de salida con activación *softmax*.

Como se observa es una arquitectura que arroja resultados deseados con los parámetros mostrados. Las capas de *Dropout* hacen que el modelo no se ajuste totalmente a los datos de entrada, ya que si esto sucede no hay posibilidad de que dicho modelo tenga buena generalización de los datos, sino que tiene un buen aprendizaje de las muestras que se le ingresaron. Las capas *MaxPool* reducen las dimensiones de los mapas y por consiguiente se reduce la cantidad de parámetros que el modelo tiene que aprender, esta también ayuda a resumir la cantidad de características en una región de un mapa generado por una capa de convolución.

La arquitectura del modelo anterior se puede ver más fácilmente como sigue:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 32)        896

 conv2d_1 (Conv2D)           (None, 62, 62, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 dropout (Dropout)           (None, 31, 31, 32)        0

 conv2d_2 (Conv2D)           (None, 31, 31, 64)        18496

 conv2d_3 (Conv2D)           (None, 29, 29, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 14, 14, 64)        0

 flatten (Flatten)           (None, 12544)             0

 dense (Dense)               (None, 512)               6423040

 dense_1 (Dense)             (None, 256)               131328

 dropout_2 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 36)                9252

=================================================================
Total params: 6,629,188
Trainable params: 6,629,188
Non-trainable params: 0
_________________________________________________________________
```

Figura 2. Arquitectura

Se entrenó con 15 épocas para minimizar el tiempo de ejecución y a su vez reducir los recursos computacionales necesarios para ejecutar esta tarea, debido a que como se trata de un conjunto de datos grande, los tiempos también llegan a ser altos.

Los resultados obtenidos fueron:

Figura 3. Acc. de entre.

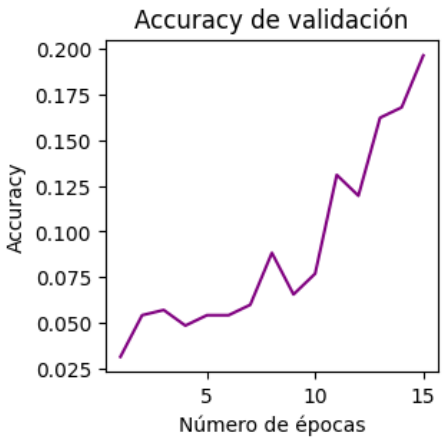

Figura 4. Acc. de vald.

En las gráficas mostradas se puede ver que ambas curvas tienden a subir, y se puede inferir que a mayor cantidad de épocas mayor va a ser el accuracy obtenido. Se necesita un entrenamiento con más épocas para conseguir un accuracy de al menos el 80%.

7.3 Transfer Learning

El uso de la técnica de Transfer Learning es muy útil, puesto que al tratarse de modelos pre-entrenados no se necesitan muchos recursos computacionales para entrenar los modelos en este problema y puede ayudar a reducir los tiempos que requieren el entrenamiento de diversos modelos.

Se entrenó con los modelos de Transfer Learning listados a continuación, que son algunos de los modelos más comunes en esta técnica:

1. ResNet50V2
2. VGG16
3. DenseNet121

Los resultados obtenidos fueron los siguientes:

1. ResNet50V2

   Se entrenó un modelo Res Net60V2 con dos capas densas de 52 y 104 neuronas, con función de activación *relu* y una capa de salida

densa de 36 neuronas con función de activación *softmax*, haciendo uso de un modelo funcional de *Keras*. Dichas capas densas tienden a conseguir mejores resultados cuando se entrenan con modelos de Transfer Learning debido a que esto ayuda al modelo a clasificar mejor los datos de entrada.

Los resultados en términos de desempeño en entrenamiento y validación fueron:

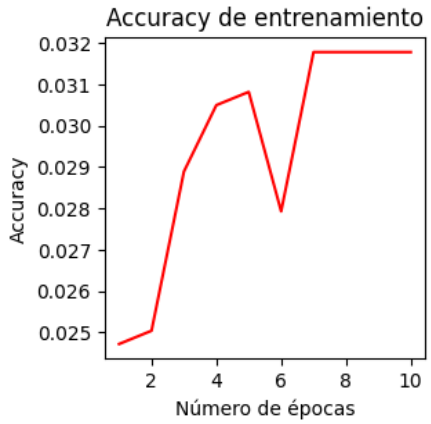

Figura 5. Acc. de entren

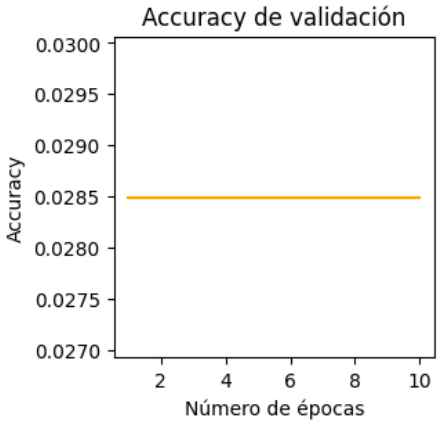

Figura 6. Acc de val

En la *Figura 5* y en la *Figura 6* se puede apreciar que la curva de desempeño de entrenamiento llega hasta un 32% con 10 épocas y la de validación se mantiene constante en un valor de 29%, ambas gráficas muestran que el valor de accuracy no depende del número de épocas, siempre va a haber una tendencia a ser una constante, lo que quiere decir que este modelo con estos parámetros no es el ideal en este problema.

2. VGG16

Se entrenó un modelo VGG16 con el mismo modelo anterior, pero con diferentes parámetros: 128 neuronas en la primera capa densa, 256 neuronas en la segunda capa densa, y 36 neuronas en la capa densa de salida y los resultados fueron:

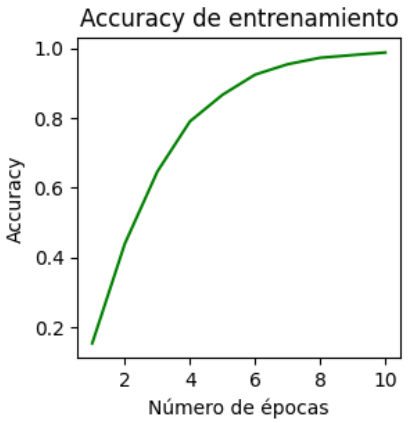

Figura 7. Acc de entr

Figura 8. Acc de val

En los diagramas anteriores se puede observar que las curvas de desempeño de entrenamiento y validación tienen un buen comportamiento, es decir se acercan a la métrica deseada, ambas curvas tienen una relación directamente proporcional entre el número de épocas y el accuracy obtenido. Si se aumenta el número de épocas puede aumentar también la medida de rendimiento.

El modelo de Transfer Learning VGG16 modela bien los datos.

3. DenseNet121

Se entrenó el modelo con las dos capas densas, con 250 y 285 neuronas y la capa densa de salida con 36 neuronas. Los resultados se muestran como sigue:

Figura 9. Acc de entre

Figura 10. Acc de val

Las gráficas anteriores muestran que tanto los accuracy de entrenamiento como de validación se comportan de manera similar. En el diagrama de entrenamiento, el número de épocas es directamente proporcional a la medida de desempeño, en el de validación hay varias variaciones ligeras en esta relación. Es un buen modelo para el problema de clasificación, se adapta bien.

8. Conclusiones de los resultados

- Todas las aproximaciones a los modelos descritas anteriormente modelan el comportamiento de los datos de alguna manera y unos mejor que otros, dependiendo de los parámetros que se le asignen a cada modelo y el número de iteraciones que se determinen también juega un papel trascendental en el problema, así como también de la arquitectura que se plantee y de qué tan compleja o sencilla sea.
- Según las curvas de desempeño que se dibujaron el mejor modelo de los que se entrenaron para este problema fue el modelo VGG16 que muestra una buena tendencia en la métrica de desempeño.
- Según los resultados obtenidos el número de iteraciones debería ser grande para que el modelo se logre entrenar correctamente y así lograr que las predicciones sean los más precisas posible.