batch size = 10
learning rate = 0.0001
cross entropy
adam
epochs = 10

batch size = 10
learning rate = 0.0001
cross entropy
adam
epochs = 50

Tecnológico de Monterrey

Campus Querétaro

Asignatura

Inteligencia artificial avanzada para la ciencia de datos

Proyecto

Implementación de un modelo de deep learning. (Portafolio Implementación)

Nombre de la alumna

Frida Lizett Zavala Pérez

Matricula

A01275226

26 de Noviembre del 2023

Implementación de un modelo de deep learning. (Portafolio Implementación)

Implementación de un modelo de deep learning. (Portafolio Implementación)

Problema

El objetivo de este proyecto es implementar una arquitectura de aprendizaje profundo. La estructura seleccionada es una red neuronal diseñada para solucionar un problema de clasificación de imágenes multiclase, en dónde se realizará la distinción de 10 clases diferentes de gemas y piedras preciosas.

Conjunto de datos

Se trabajó con un conjunto de de datos extraído de Kaggle, llamado *Gemstones Images*. Este dataset contiene más de 3200 imágenes de gemas, las cuales están agrupadas en 87 clases diferentes, dónde cada una cuenta con una carpeta de entrenamiento y una de prueba. Para fines de esta implementación se seleccionaron aleatoriamente 10 de estas clases y se complementaron con una tercera carpeta para validación, dichas imágenes de validación fueron recolectadas manualmente desde el buscador de google. Contando así con un conjunto de datos de 10 clases, dondé cada una tiene de 30 a 40 imágenes para entrenamiento (332 imágenes, siendo el 78.12% del total), de 4 a 5 para prueba (43 imágenes lo que representa el 10.12%) y 5 de validación (50 imágenes, que es el 11.76%). Todas éstas imágenes son de tamaños diferentes y están en formato jpg.

Arquitectura del modelo

Modelo base

A partir del transfer learning, se utilizó la arquitectura VGG16 como modelo base pre entrenado, sin incluir las capas densas superiores. Se congelaron las capas convolucionales para evitar su actualización durante el entrenamiento y se agregó una capa densa para la clasificación multiclase.

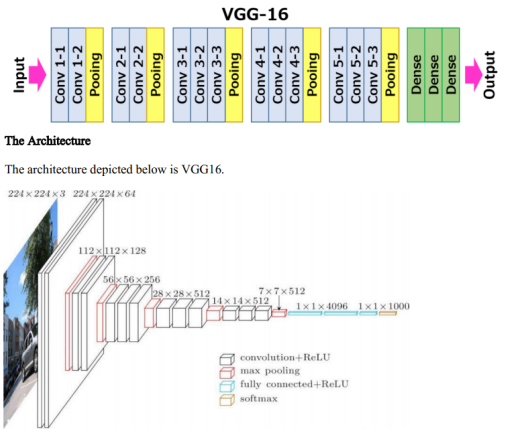

Figura 1. Arquitectura VGG16

La arquitectura VGG16 es una red convolucional muy popular que se utiliza para aplicaciones de visión por computadora, comúnmente la clasificación de imágenes. Su uso resulta beneficioso en casos en los que se requiere construir un modelo y se cuenta con un dataset pequeño o se busca acelerar el entrenamiento y generar un menor costo a nivel computacional. Inicia con una capa de entrada que recibe las imágenes, después usa capas convolucionales para aprender las características y capas de pooling para reducir el tamaño de salida, seguido de una capa flatten, la cual convierte la salida en un vector unidimensional, para poder pasar

a capas conectadas que son las que a partir de las características aprendidas previamente realizan las tareas, tales como la clasificación. Finalmente se tiene la capa de salida, que en el caso de la clasificación, ésta tendrá tantas neuronas como número de clases a identificar. La función de activación empleada entre las capas es RELU (Rectified Linear Unit).

Model: "sequential_8"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
flatten_8 (Flatten)	(None, 25088)	0
dense_22 (Dense)	(None, 256)	6422784
dense_23 (Dense)	(None, 10)	2570

Figura 2. Arquitectura de la red generada 1

Con un modelo secuencial se le añadieron 3 capas más al modelo precargado. Una capa flatten para convertir la salida en un vector unidimensional, una capa densa con 256 neuronas y función de activación ReLU la cual ayuda a introducir no linealidades en la red y la capa de salida, con un número de neuronas igual al número de clases en el dataset, su activación es softmax usado para la clasificación multiclase.

Entrenamiento inicial

El modelo se entrenó durante 10 épocas con un tamaño de lote (batch size) de 32, una tasa de aprendizaje (learning rate) y optimizador predeterminado, considerando como función de costo entropía cruzada categórica. Durante este proceso de entrenamiento, se utilizó un generador de datos que aplicó transformaciones de aumento de datos en el conjunto de entrenamiento, con la finalidad de obtener mejores resultados.

Resultados iniciales

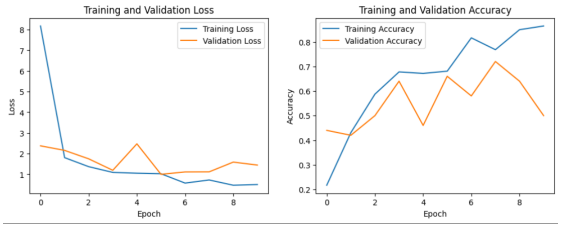

Figura 3. Gráficos de pérdida y precisión durante el entrenamiento 1

Matriz de confusión. Conjunto de validación

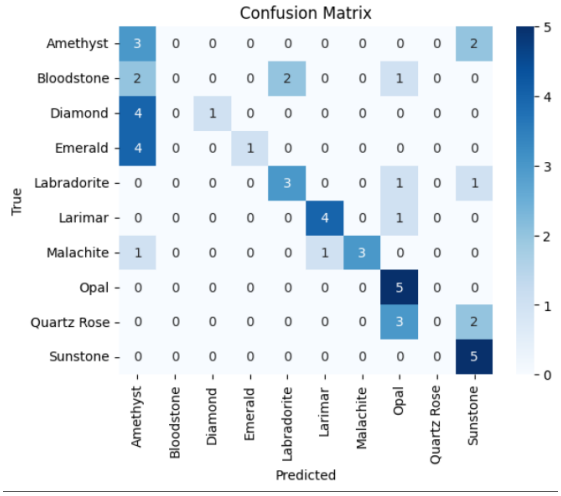

Figura 4. Matriz de confusión de validación 1

	precision	recall	f1-score	support
Amethyst	0.21	0.60	0.32	5
Bloodstone	0.00	0.00	0.00	5
Diamond	1.00	0.20	0.33	5
Emerald	1.00	0.20	0.33	5
Labradorite	0.60	0.60	0.60	5
Larimar	0.80	0.80	0.80	5
Malachite	1.00	0.60	0.75	5
Opal	0.45	1.00	0.62	5
Quartz Rose	0.00	0.00	0.00	5
Sunstone	0.50	1.00	0.67	5
accuracy			0.50	50
macro avg	0.56	0.50	0.44	50
weighted avg	0.56	0.50	0.44	50

Figura 5. Resultados de validación 1

Predicciones iniciales

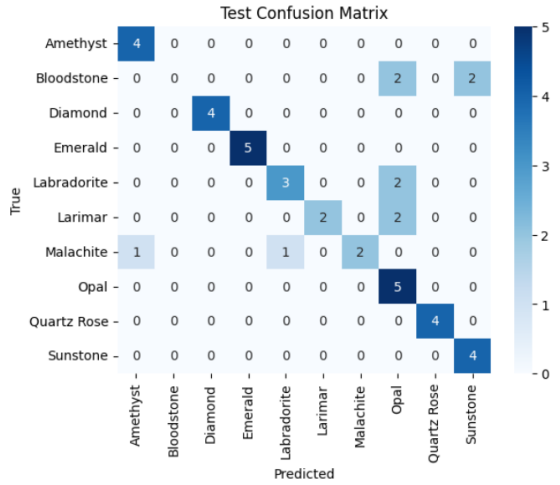

Figura 6. Figura 4. Matriz de confusión de prueba 1

	precision	recall	f1-score	support
Amethyst	0.80	1.00	0.89	4
Bloodstone	0.00	0.00	0.00	4
Diamond	1.00	1.00	1.00	4
Emerald	1.00	1.00	1.00	5
Labradorite	0.75	0.60	0.67	5
Larimar	1.00	0.50	0.67	4
Malachite	1.00	0.50	0.67	4
Opal	0.45	1.00	0.62	5
Quartz Rose	1.00	1.00	1.00	4
Sunstone	0.67	1.00	0.80	4
accuracy			0.77	43
macro avg	0.77	0.76	0.73	43
weighted avg	0.76	0.77	0.73	43

Figura 7. Resultados de prueba 1

El modelo evaluado con el conjunto de datos de validación presentó una precisión del 50% mientras que con el conjunto de prueba fue del 77%, siendo que no predijo de manera tan óptima. Sin embargo se observa que efectivamente el modelo es capaz de generar algunas predicciones correctas, lo que indica que se puede seguir bajo la misma estructura ajustando diversos parámetros para mejorar la calidad de las predicciones que se generan.

Mejora del modelo

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_5 (Flatten)         (None, 25088)             0

 dense_15 (Dense)            (None, 256)               6422784

 dropout_10 (Dropout)        (None, 256)               0

 dense_16 (Dense)            (None, 128)               32896

 dropout_11 (Dropout)        (None, 128)               0

 dense_17 (Dense)            (None, 10)                1290

=================================================================
Total params: 21171658 (80.76 MB)
Trainable params: 6456970 (24.63 MB)
Non-trainable params: 14714688 (56.13 MB)
```

Figura 8. Arquitectura de la red generada 2

Regularización

Para mejorar el rendimiento del modelo se aplicó la técnica de dropout, para ello se añadieron dos capas de drop out al 0.5, con una capa conectada de 128 neuronas y activación ReLU en medio de ellas. La técnica de drop out es utilizada para evitar el sobreajuste de la red, ya que debido al tamaño del dataset empleado, este modelo es propenso a presentarlo.

Ajuste de hiperparámetros

En cuanto a los cambios aplicados en los hiperparámetros, en el entrenamiento del modelo se aplicó el optimizador Adam, para aprovechar su capacidad de adaptar la tasa de aprendizaje de cada parámetro mientras que ayuda con la eficiencia computacional, con una tasa de aprendizaje inicial de 0.0001, manteniendo la función de pérdida de entropía cruzada categórica y el tamaño de lote de 32, ya que después de algunas pruebas se observó que en este caso un tamaño de lote menor no significaba una diferencia relevante en los resultados. En esta ocasión la red fue entrenada durante 50 épocas para darle tiempo suficiente al modelo para capturar

mejor los patrones y así mejorar la generalización de las predicciones.

Resultados mejorados

Figura 9. Gráficos de pérdida y precisión durante el entrenamiento 2

Se mantiene una pérdida baja al final del entrenamiento, y la precisión tiene un comportamiento esperado, sin presentar sobreajuste.

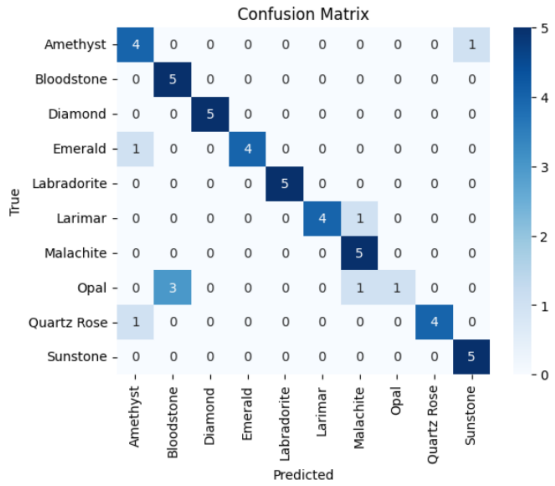

Figura 10. Figura 4. Matriz de confusión de validación 2

	precision	recall	f1-score	support
Amethyst	0.67	0.80	0.73	5
Bloodstone	0.62	1.00	0.77	5
Diamond	1.00	1.00	1.00	5
Emerald	1.00	0.80	0.89	5
Labradorite	1.00	1.00	1.00	5
Larimar	1.00	0.80	0.89	5
Malachite	0.71	1.00	0.83	5
Opal	1.00	0.20	0.33	5
Quartz Rose	1.00	0.80	0.89	5
Sunstone	0.83	1.00	0.91	5
accuracy			0.84	50
macro avg	0.88	0.84	0.82	50
weighted avg	0.88	0.84	0.82	50

Figura 11. Resultados de validación 2

Predicciones finales. Conjunto de prueba

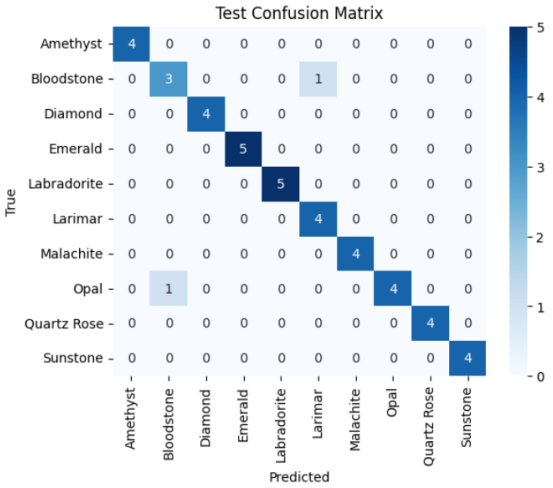

Figura 12. Matriz de confusión de prueba 2

	precision	recall	f1-score	support
Amethyst	1.00	1.00	1.00	4
Bloodstone	0.75	0.75	0.75	4
Diamond	1.00	1.00	1.00	4
Emerald	1.00	1.00	1.00	5
Labradorite	1.00	1.00	1.00	5
Larimar	0.80	1.00	0.89	4
Malachite	1.00	1.00	1.00	4
Opal	1.00	0.80	0.89	5
Quartz Rose	1.00	1.00	1.00	4
Sunstone	1.00	1.00	1.00	4
accuracy			0.95	43
macro avg	0.96	0.96	0.95	43
weighted avg	0.96	0.95	0.95	43

Figura 13. Resultados de prueba 2

Se observa que efectivamente el modelo presentó grandes mejoras en las predicciones, teniendo una precisión del 84% con los datos de validación, generando muy pocos errores en las predicciones, y un desempeño aún mejor con los datos de train, alcanzando un 95% de precisión, lo cual indica que efectivamente se mejoró el desempeño del modelo.

Ejemplo de predicciones

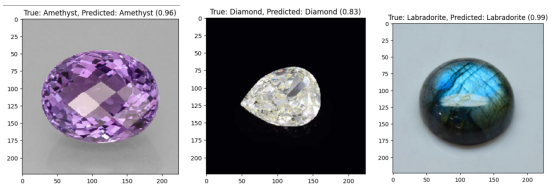

Figura 14. Predicciones

Conclusiones

En este proyecto, se abordó con éxito el problema de clasificación de gemas y piedras mediante el uso de técnicas de aprendizaje profundo, utilizando frameworks para el entrenamiento del modelo. Se logró una mejora significativa en el rendimiento del modelo mediante la aplicación de regularización y ajuste de hiperparámetros, obteniendo de esta manera mejores predicciones. Entendí con claridad el efecto de los cambios aplicados y pude aplicar el conocimiento adquirido sobre las diversas técnicas aprendidas.

Referencias

Gemstones Images. (2020, March 21). https://www.kaggle.com/datasets/lsind18/gemstones-images

Red neuronal convolucional VGG-16 - MATLAB vgg16 - MathWorks América Latina. (n.d.). https://la.mathworks.com/help/deeplearning/ref/vgg16.html

Anexos

El dataset empleado para el modelo al igual que dos imágenes extra para generar predicciones, pueden encontrarse en la siguiente carpeta:
images