Memoria del proyecto de Machine Learning

Clasificación de rocas mediante CNN y Transfer Learning

I. Introducción

La identificación de rocas es una labor que se ha realizado hasta hace no mucho con métodos analíticos como el rayado, la exposición a determinados tipos de sustancias químicas o la observación de sus patrones a pequeña escala con ayuda de un microscopio. Estos métodos requieren que la persona que desee clasificar una roca deba poseerla de manera física. Sin embargo, con el avance paulatino en las técnicas de Machine Learning y Deep Learning, podemos ser capaces de construir modelos de clasificación de rocas únicamente con fotografías de estas. Esto puede ser de gran ayuda si se desea identificar una gran cantidad de muestras, ya que el tiempo requerido para identificarlas por los medios analíticos sería mucho mayor.

El objetivo de este proyecto es construir un modelo de redes neuronales convolucionales, lo que nos permitirá clasificar muestras de rocas divididas en un total de 6 clases diferentes, tanto sedimentarias como volcánicas.

II. Dataset

El dataset utilizado se compone de un total de 6480 imágenes de rocas, divididas en conjuntos de entrenamiento, prueba y validación. La distribución de las imágenes es la siguiente:

- Conjunto de entrenamiento: 5667 imágenes
- Conjunto de prueba: 274 imágenes
- Conjunto de validación: 539 imágenes

Las 6 clases en las que se dividen los 3 conjuntos son los siguientes:

1. Conglomerados (Conglomerate)
2. Calizas dolomíticas (Dolomitic Limestone)
3. Calizas (Limestone)
4. Areniscas (Sandtone)
5. Lutitas (Shale (Mudstone))
6. Tezontles

Las 5 primeras clases descritas son rocas sedimentarias de tamaño de grano, porosidad y color distinto, mientras que los tezontles son rocas volcánicas porosa, muy ligeras y de color rojizo ampliamente usadas en construcción.

Origen de los datos: https://universe.roboflow.com/chris-qjtul/rocks-life/dataset/2

III. Preprocesamiento de los datos

Las imágenes fueron previamente reescaladas a 640 x 640 píxeles, además de haber sido aplicadas técnicas de 'Data augmentation', obteniendo 3 imágenes adicionales por imagen del dataset. Los cambios que se aplicaron a las nuevas imágenes son los siguientes:

1. Volteado horizontal
2. Saturación: entre -25% y +25%
3. Brillo: entre -15% y +15%
4. Difuminado: hasta 2.5 píxeles

Se ha realizado un histograma para poder observar la distribución de imágenes en todos los conjuntos con respecto a los 6 tipos de rocas del dataset (Figura 1).

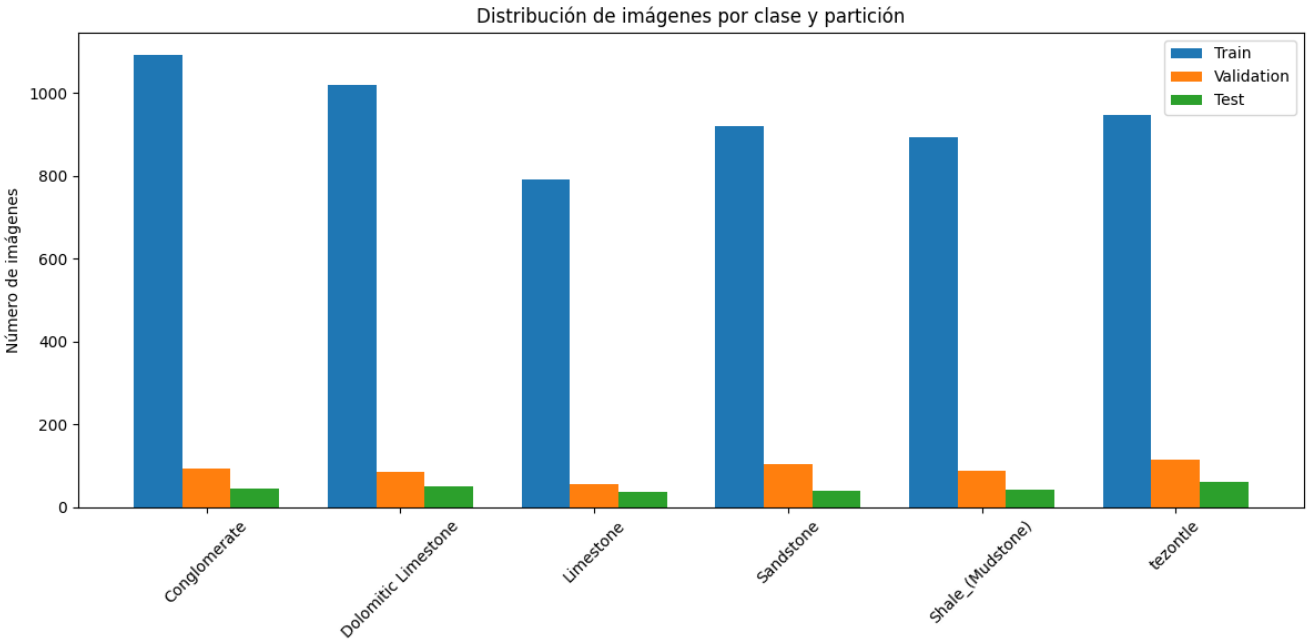

Figura 1: Histogramas de las distribuciones de las imágenes por tipo de roca y conjunto (train, validation y test).

Vemos que, en general, las imágenes están correctamente distribuidas en los tres conjuntos. El mayor desbalance ocurre en el conjunto de entrenamiento, donde existe una diferencia de alrededor del 27% entre las calizas y los conglomerados. Sin embargo, no es una diferencia sustancial ya que el balanceo de las clases no varía prácticamente el rendimiento del modelo.

Ya tenemos las imágenes reescaladas, clasificadas por tipo y con las transformaciones para aumentar la precisión del modelo, con lo que podemos empezar a construir el modelo.

IV. Modelado

En este proyecto se ha usado ResNet50, una red neuronal profunda con 50 capas con una estructura que introduce bloques residuales para poder así facilitar el entrenamiento de redes más profundas (Figura 2).

Las redes como ResNet introducen conexiones o atajos representadas en la imagen por flechas curvas (Figura 2). Estas permiten que la información pueda “saltar” capas, lo que facilita el entrenamiento en redes profundas.

El input suele ser, como en el caso de este proyecto, una imagen de resolución 224 x 224 y 3 canales de colores. La primera capa reduce el tamaño espacial de la imagen para extraer sus características básicas, seguido de un ‘pooling’, lo que reduce aún más el tamaño de la imagen.

Los primeros bloques (color amarillo) extraen características más simples con una capa inicial que reduce la dimensionalidad, una segunda capa que extrae las características y la última que restaura la dimensionalidad. La salida de cada bloque se suma a la entrada.

Con cada conjunto de bloques la cantidad de filtros se duplica, de modo que los bloques que poseen más filtros son los que captarán las características más complejas. A la salida de estos últimos bloques. Para nuestro estudio no queremos la capa final fc,3 puesto que tenemos más de 3 clases. Esto lo podemos conseguir añadiendo el parámetro ‘include_top=False’ a la hora de definir la red.

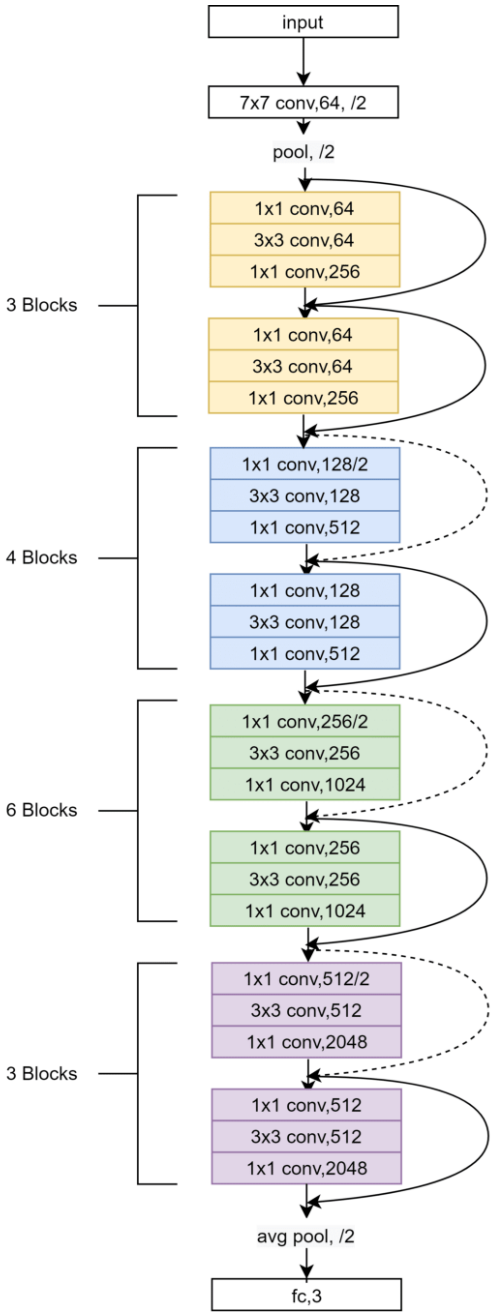

Figura 2: Esquema de la arquitectura de las capas de ResNet50.

Lo que obtenemos a la salida de este modelo es un tensor de tamaño (None, 7, 7, 2048), es decir, 2048 capas de activación de 7 x 7 para cada imagen. Tras el modelo de ResNet50, añadimos una capa de GlobalAveragePooling2D, la cual reemplaza a la capa ‘avg pool, /2’ anterior a la última capa (Figura 2). Esta capa reduce este tensor de salida a un vector plano (2048,).

La siguiente capa Dense está conectada con 128 neuronas que sirve como un clasificador intermedio que aprende combinaciones no lineales de las características extraídas por ResNet (Figura 3). Con el fin de reducir el ‘overfitting’ se apagan el 30% de las neuronas con ayuda de Dropout.

La última capa debe poseer el mismo número de clases en las que se desea clasificar cada imagen. Esta capa sustituye a la capa fc,3 (Figura 2).

```
base_model = ResNet50(input_shape=IMG_SIZE + (3,),
                      include_top=False,
                      weights='imagenet')

base_model.trainable = False

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(6, activation='softmax')  # 6 clases
])
```

Figura 3: Extracto de código que representa la arquitectura de la red neuronal.

Una vez construido el modelo debemos compilarlo (Figura 4). Se usará el optimizador Adam ya que suele ser el que mejores resultados da en tareas de clasificación de imágenes al ser muy robusto, eficiente y muy útil cuando se aplica 'fine-tuning' en modelos de ResNet50. Como tenemos etiquetas de tipo 'string' (como Limestone) convertidas a enteros a la hora del procesamiento de los datos, podemos usar como función de pérdida el 'sparse_categorical_crossentropy' (Figura 4). Nuestro objetivo es ver la relación entre el número de predicciones correctas con respecto al total de predicciones, de modo que usaremos 'accuracy' como métrica de medida de la precisión del modelo.

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Figura 4: Extracto de código que representa la compilación del modelo.

Con el fin de reducir el 'overfitting', se aplicará 'EarlyStopping', el cual nos ayuda a detener el entrenamiento si la pérdida del conjunto de validación no mejora después de 20 épocas (Figura 5). Para evitar usar datos sobreentrenados, introducimos el parámetro 'restore_best_weights=True' y así se capturarán los pesos del modelo donde se obtuvo el mejor rendimiento. A continuación, se definen un máximo de épocas (en nuestro caso 50) y se pasa a

entrenar el modelo introduciendo el conjunto de 'train' y el de validación para así evaluar el desempeño tras cada época (Figura 5).

```
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True
)

EPOCHS = 50

history = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=EPOCHS,
    callbacks=[early_stopping],
)
```

Figura 5: Extracto de código que representa la declaración del EarlyStopping y la etapa de entrenamiento del modelo.

Una vez obtenida la precisión en este primer entrenamiento, nos interesa ahora poder aumentar la eficacia de la clasificación. Para ello, aplicaremos un método de 'descongelamiento' de las capas profundas del modelo ResNet50 llamado 'fine-tuning'. Definimos que del total de capas que existen para ResNet50 en Keras (175), mantenemos congeladas las 120 primeras y descongelamos las 55 últimas (Figura 6). Decidimos descongelar las últimas capas ya que son las que detectan patrones más complejos y específicos, de modo que el modelo se ajusta de manera precisa a las características de los datos de imagen.

Para evitar que el modelo sobrescriba lo ya aprendido, debemos introducir una tasa de aprendizaje o 'learning rate' muy bajo (del orden de 1 E-5). Así, el ajuste de los pesos que ya existen dentro de ResNet50 será mínimo y los cambios serán leves. Cuando el primer entrenamiento termine (Figura 5), podemos retomar el entrenamiento con 'fine-tuning' desde la época donde el anterior modelo se detuvo con el 'EarlyStopping' gracias al parámetro 'initial_epoch' (Figura 6).

Cuando el entrenamiento con fine-tuning termina, podemos evaluar el rendimiento del modelo con el conjunto de prueba. Con los parámetros descritos anteriormente, obtenemos una precisión en test del 88%, siendo un porcentaje lo suficientemente alto como para considerarse un modelo de clasificación muy competente.

```python
base_model.trainable = True

fine_tune_at = 120
for layer in base_model.layers[:fine_tune_at]:
    layer.trainable = False

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

total_epochs = EPOCHS + 10

history_finetune = model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=total_epochs,
    initial_epoch=history.epoch[-1] + 1,
    callbacks=[early_stopping]
)
```

Figura 6: Extracto de código que representa el descongelamiento de las capas finales de ResNet50 y la compilación del nuevo modelo con un 'learning rate' muy bajo a partir de la época donde se detuvo el anterior modelo.

V. Predicción y resultados finales

Una vez hechas las predicciones en el conjunto de prueba, vamos a crear una matriz de confusión para poder observar si existen tipos de roca peor clasificados que otros, además de poder ver qué rocas son las que el modelo clasifica mejor (Figura 7).

Podemos ver que las calizas dolomíticas es el tipo de roca que posee peores predicciones, puesto que es la clase con mayor cantidad de falsos positivos. Esto puede deberse a que su textura y color son muy similares a las de otras rocas sedimentarias presentes en el dataset como las calizas, areniscas o algunos tipos de lutita. Por otro lado, otros tipos de roca como las areniscas poseen una coloración más anaranjada y patrones de estratificación que no se suelen encontrar en los demás tipos de roca del dataset. Las calizas son las rocas que mejor predice el modelo. Esto puede deberse a que poseen un color, forma y textura únicos que solamente podrían confundirse con sus variables dolomíticas. Por otro lado, los tezontles, a pesar de ser rocas volcánicas con características distintas a las demás rocas del dataset, no han sido clasificados tan bien como las calizas (Figura 7).

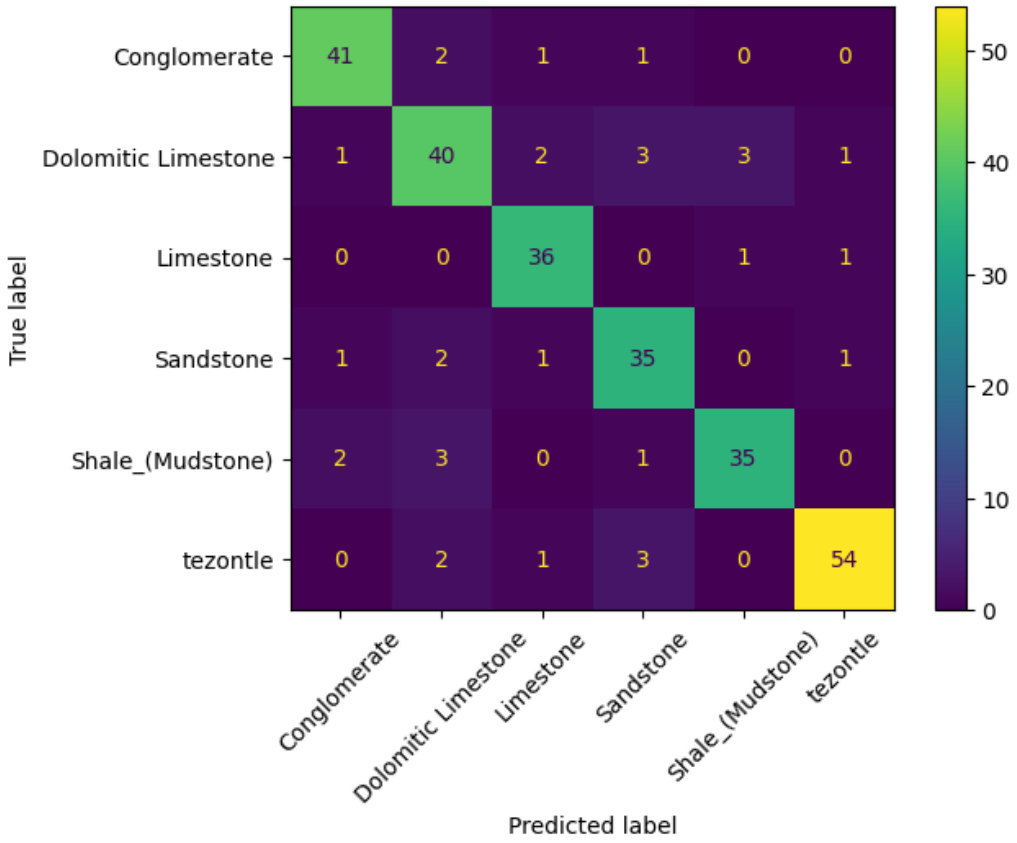

Figura 7: Matriz de confusión de las predicciones con los datos de prueba. Se aprecia que el modelo clasifica con gran precisión, a pesar de existir algunos tipos de roca con errores de predicción de más del 10%.

A continuación, veremos el porcentaje de confianza con el cual predice el modelo (Figura 8). Podemos observar que prácticamente la totalidad de las imágenes han sido predichas con más del 90 % de confianza. Esto nos permite concluir que el modelo detecta con claridad los patrones, formas, colores y las características complejas. Sin embargo, dentro de esas predicciones con alto grado de confianza están las predicciones erróneas. Del total de las 33 imágenes que fueron mal clasificadas (Figura 7), hay un total de 15 cuya clase fue asignada con un grado de confianza mayor a 90%.

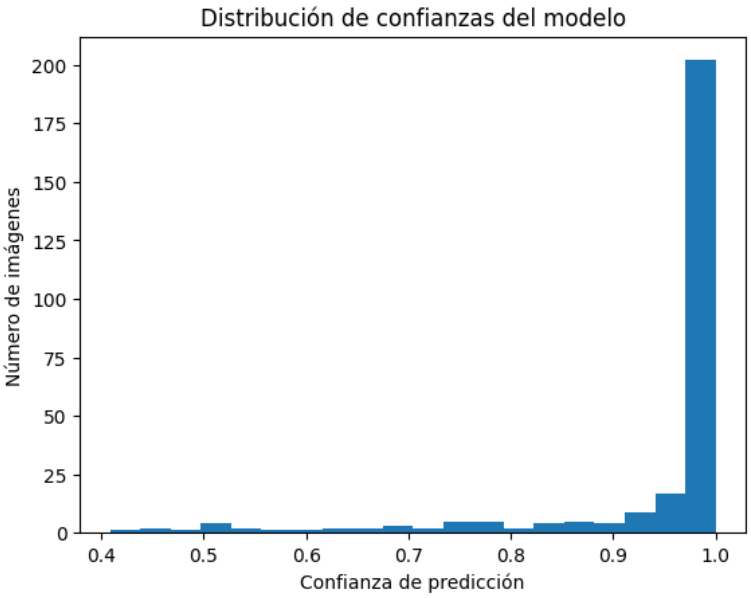

Figura 8: Histograma que muestra el nivel de confianza en las predicciones de las imágenes del conjunto de prueba. Se observa claramente como la práctica totalidad de los datos son predichos con una confianza cercana al 100%.

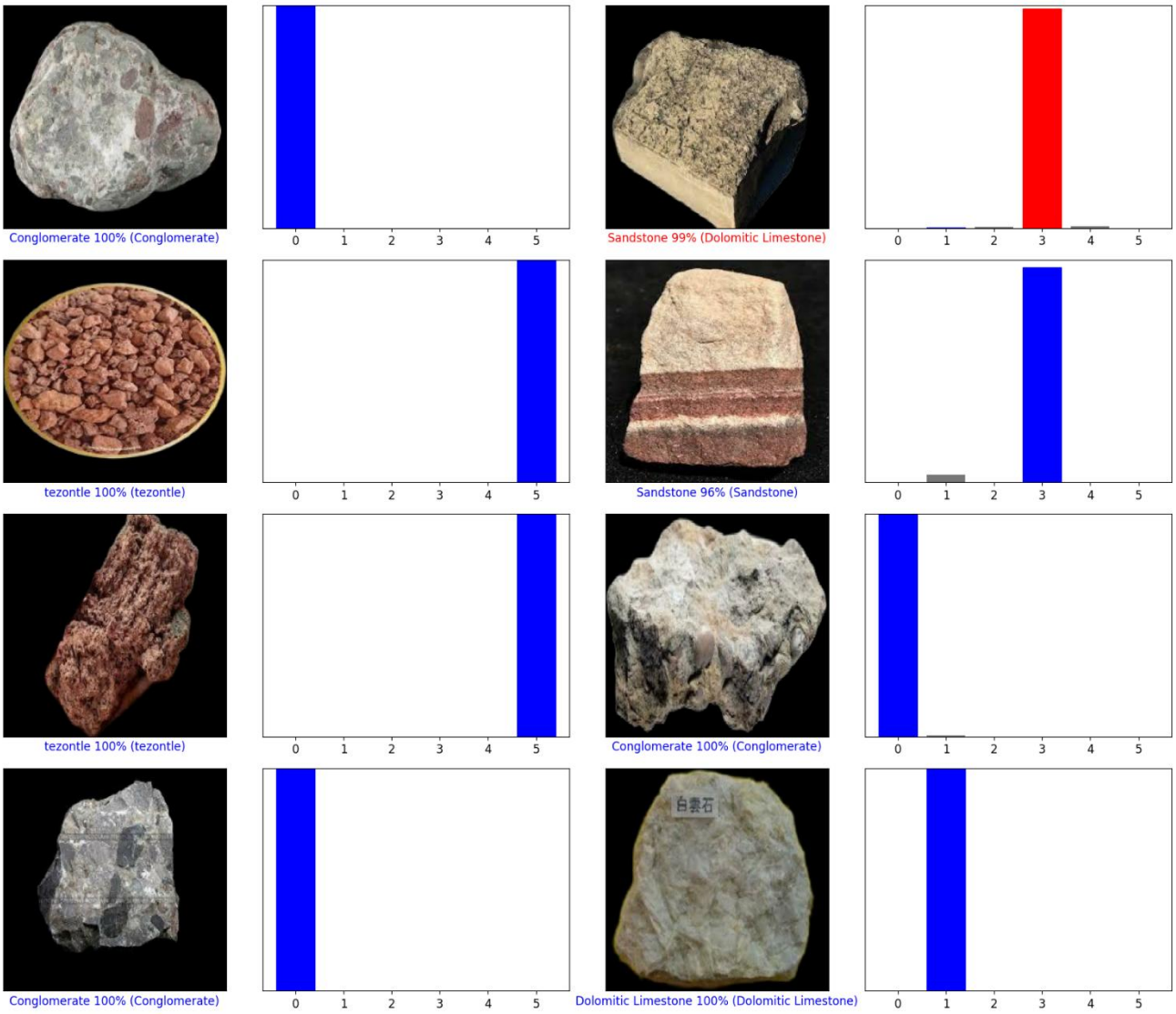

Figura 9: Fotografías de algunas de las muestras del conjunto de prueba junto con su correspondiente gráfico de barras de confianza.

Finalmente, con el fin de obtener una visión más clara de los resultados de clasificación, representaremos algunas de las fotografías del conjunto de prueba y sus correspondientes histogramas de confianza en las predicciones (Figura 9). Se observa claramente como todas las predicciones poseen un grado de confianza muy cercano al 100%, lo que se corresponde con lo visto en la figura 8. El modelo clasificó como arenisca una caliza dolomítica, lo que seguramente sea debido a que el modelo haya identificado que una de las características propias de las areniscas es su color amarillento o anaranjado. De este modo, al encontrarse con una muestra de ese color, su predicción sea tan clara, a pesar de que probablemente la muestra posea patrones o texturas propios de las calizas dolomíticas que el modelo no haya podido identificar.

VI. Conclusiones y futuros pasos

ResNet50 es un tipo de red neuronal convolucional que permite al usuario realizar clasificaciones de rocas de manera muy precisa. Su estructura compleja permite la captación de características texturales, colores y patrones propios de cada tipo de roca, de manera que el modelo ha sido capaz de reconocer la textura clástica de los conglomerados o la estratificación en las areniscas. Asimismo, las predicciones realizadas por el modelo poseen un grado de confianza muy cercano al 100%, lo que demuestra que ha conseguido detectar los patrones de reconocimiento de manera eficiente para sí poder inferir la clase a la que pertenece cada muestra.

Este proyecto se sustenta en gran parte por la calidad de las imágenes que se pretenden clasificar. Un dataset que posea pocas imágenes o en el cual no se haya llevado a cabo un proceso de aumento de datos (Data Augmentation), realizará una peor labor de clasificación. Del mismo modo, un correcto etiquetado de las rocas permitirá al modelo poder aprender las características inherentes a cada tipo de roca. En el caso de la realización de este proyecto, debido a la mala calidad de las imágenes del dataset inicial, se tuvo que tomar la decisión de cambiar de set de imágenes o uno donde las muestras estaban mejor etiquetadas y mejor escogidas. Esto condujo a un aumento considerable de la precisión de las predicciones utilizando el mismo modelo de clasificación.

Una manera de poder optimizar este modelo sería la revisión exhaustiva de cada imagen del dataset para sí poder identificar algunas muestras que puedan perjudicar en las predicciones del modelo. También podrían utilizarse otras redes neuronales como MobileNetV2 o DenseNet121, las cuales podrían aportar mejores resultados con los parámetros adecuados.

Partiendo de este proyecto, una futura aplicación relacionada con la geología sería una ampliación del dataset, aportando nuevas clases e imágenes para sí enriquecer al modelo y diversificar su grado de clasificación. Asimismo, en el ámbito de mi Tesis Doctoral, mi interés se centra en la identificación de trazas de titanio y litio en la superficie lunar utilizando técnicas de Machine Learning. Una de las aplicaciones abordadas podría ser la búsqueda de este mineral con ayuda de redes neuronales convolucionales como ResNet50 o EfficientNetB1.