Monitoreo en Proyectos de Machine Learning

Santiago Gacia Solarte[1] and Santiago Loazia Cardona[2]

[1]santiago.garcia_sol@uao.edu.co
[2]santiago.loaiza@uao.edu.co

I. Introducción

Tener un buen monitoreo permite la escalabilidad de un modelo y facilita la detección de cambios que puedan hacer este pierda su capacidad de clasificación. Además, ayuda a identificar posibles sesgos o errores durante la fase de entrenamiento, garantizando un modelo transparente y equitativo al clasificar y tomar decisiones basadas en su proceso de clasificación.

Un problema de no realizar un monitoreo adecuado a los algoritmos es la aparición de sesgos. Algunos prototipos de IA (Inteligencia Artificial) en NLP (Procesamiento de Lenguaje Natural) han mostrado sesgos notables, llegando al punto de que la misma IA podría considerarse racista. También se han dado casos en los que el sesgo de la IA ha llevado a respuestas homofóbicas, sexistas y machistas en su proceso de clasificación.

Aunque las IA, debido a un mal algoritmo de aprendizaje, pueden llegar a ofender a ciertos sectores, este problema es menor en comparación con el mal manejo del aprendizaje automático en ámbitos sensibles como la medicina. En este campo, las IA y el aprendizaje automático han contribuido significativamente en los diagnósticos, por lo que en estas herramientas médicas no hay margen para errores en la clasificación o predicción de los algoritmos de aprendizaje automático.

II. Diseño y entreno del modelo

En el proyecto ”Detección de neumonía”se desplegó una aplicación que servirá como herramienta para ayudar a los médicos en la detección de un tipo de fractura. Para ello, se optó por un modelo de capas convolucionales que permite clasificar entre seis tipos de fracturas: mano, dedo, muñeca, antebrazo, húmero, codo y hombro.

Para la clasificación, se opta por una red neuronal convolucional simple, la cual cuenta con: con tres capas convolucionales, tres capas de max pooling (despues de cada capas conbvolucional esta una de max poling) una capa flatten y dos capas densas. En la última capa, se especifica la cantidad de clases a clasificar. A continuación, se muestra cómo está estructurada la red neuronal para la clasificación.

```
model = Sequential()

model.add(Conv2D(32,(3,3),1,
          activation="relu",
          input_shape=(128,128,3)))
model.add(MaxPooling2D())
model.add(Conv2D(64,(3,3),1,
          activation="relu"))
model.add(MaxPooling2D())
model.add(Conv2D(32,(3,3),1,
          activation="relu"))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(7,activation='softmax'))
```

El modelo se compila usando *sparse categorical crossentropy* como función de pérdida y se entrena durante 20 épocas. Al final del entrenamiento, los valores de validación para la pérdida y la precisión muestran un comportamiento excelente, alcanzando 0.25 y 0.92, respectivamente.

III. Rendimiento

Una vez completado el entrenamiento, se calcula el rendimiento utilizando las principales métricas de evaluación: *Loss* y *Accuracy*. Esto genera dos curvas de entrenamiento. La primera es la de *Loss*, en la cual, tanto para el entrenamiento como para la validación, se observa una disminución constante, tal como se muestra en la Figura 1.

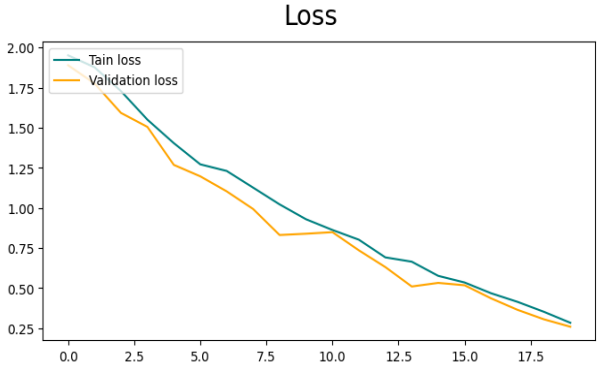

Figura 1. Curva de Perdida

De igual manera, se genera la curva de *Accuracy* tanto para el entrenamiento como para la validación, resultando en una gráfica que aumenta de manera constante.

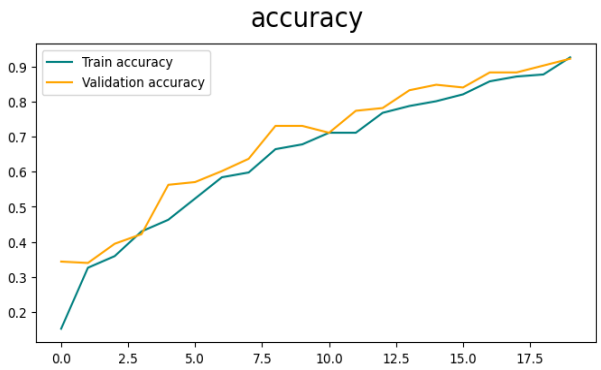

Figura 2. Curva de Exactitud

El comportamiento de estas curvas, que convergen hacia un mínimo o un máximo y muestran similitudes entre el entrenamiento y la validación, indica que el entrenamiento del modelo fue satisfactorio.

IV. Monitoreo

Para monitorear el modelo, es necesario recurrir a tres medidas principales: el *Recall*, que es la tasa de verdaderos positivos, y se calcula como $Recall = TP + FN/TP$, Esta métrica mide la capacidad del modelo para identificar correctamente las instancias positivas. En este caso, el modelo presenta un *Recall* de 0.99.

La segunda métrica es la precisión, definida por la fórmula: $Precision = TP/TP + FP$ donde TP es el número de verdaderos positivos y FP es el número de falsos positivos. La precisión mide la proporción de etiquetas identificadas como positivas por el modelo que realmente son positivas. En este caso, el modelo tiene una precisión de 0.99.

Finalmente, está la Exactitud (Accuracy), que se define como: $Accuracy = CorrectPrediction/Totalpredicciones$, Esto indica la capacidad del modelo para hacer predicciones correctas, tanto positivas como negativas, sobre el total de predicciones realizadas. En este caso, el modelo tiene una Exactitud de 0.98.

Por otro lado, se genera una matriz de confusión para evaluar cómo clasifica correctamente el modelo los diferentes labels. En este caso, se presenta la matriz de confusión en la Figura 3.

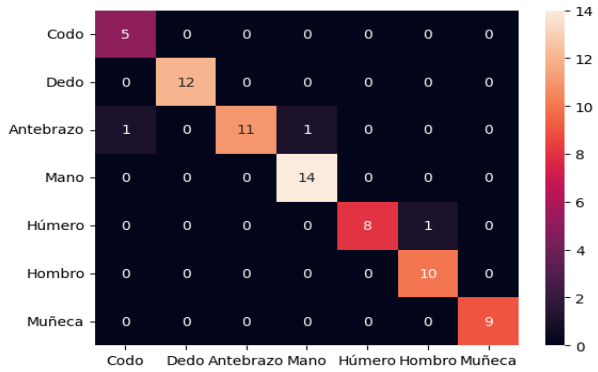

Figura 3. Matriz de confusion

En el caso de la clasificación de un conjunto de prueba, el modelo comete errores en tres casos específicos: un codo se clasifica incorrectamente como antebrazo, un antebrazo se clasifica como mano, y un hombro se clasifica como húmero.

Aunque se presentan tres errores en la clasificación, se puede argumentar que las imágenes mal etiquetadas muestran características similares a las de las etiquetas con las que fueron clasificadas. Esto puede deberse a que la fractura es poco apreciable o a que las radiografías de antebrazo, codo y húmero son muy similares, dado que muchas de las imágenes pertenecientes a estas etiquetas son radiografías del brazo completo, a continuación un ejemplo de las imagenes que pueden generar un confusion en la clasificacion.

V. Ejemplos

Figura 4. Posible caso de Codo etiquetado como antebrazo

Figura 5. Posible caso de Humero etiquetado como hombro

VI. Conclusiones

Como se observa durante todo el pipeline del entrenamiento del modelo, las redes convolucionales son bastante poderosas para extraer características de las imágenes, diferenciarlas y clasificarlas. A pesar de tener una arquitectura simple, con solo tres capas convolucionales, se logran obtener resultados bastante buenos.

Es posible mejorar los resultados y alcanzar una precisión del 100 % al contar con un conjunto de datos más amplio o con un conjunto de datos que utilice el mismo método de radiografías. El dataset utilizado muestra imágenes tomadas con diferentes métodos, lo que puede generar ruido en la red neuronal al realizar la clasificación.

Es importante aclarar que este tipo de aplicaciones, para ser utilizadas en un ámbito profesional, deben ser desarrolladas en compañía de un profesional de la salud, como un radiólogo, quien es el encargado de obtener los datos y ayudar a los ingenieros a realizar una etiquetación correcta de las imágenes. Posteriormente, el ingeniero o ingeniera se encargará del procesamiento de datos, la creación de la red neuronal, el desarrollo de la aplicación y su despliegue.

Como reflexión final, más modelos tienden a producir mejores resultados. En este caso, se optó por la creación y entrenamiento de un único modelo que permita identificar la parte del cuerpo afectada por una fractura. Sin embargo, es posible desarrollar otros tipos de redes neuronales que, por ejemplo, muestren la fractura en un mapa de calor, evalúen la gravedad de la fractura o indiquen si ha exisitido o no intervención médica. Aunque se puede considerar el uso de varios modelos para crear un sistema más robusto, la creación y entrenamiento de estos modelos adicionales requieren un tratamiento de datos diferente y una mayor capacidad de cómputo, lo cual puede no ser factible con las herramientas actuales.