Pontificia Universidad Javeriana
Maestría
Aprendizaje Profundo

Aprendizaje Profundo Taller 2

Elkin Daniel Prada Gómez
elkind.prada@javeriana.edu.co

William Peña
walexanderpena@javeriana.edu.co

1 Resumen

En el siguiente trabajo se analizará la construcción de dos modelos basados en redes neuronales convolucionales para el procesamiento de imágenes de señales de tránsito, con el fin de poner en práctica los conceptos obtenidos en la asignatura, aprendizaje profundo y evaluar la diferencia entre 2 modelos con diferentes capas y parámetros de procesamiento. Para realizar este trabajo, se hace uso del dataset Traffic Sign de la página Kaggle, y se procede a realizar la construcción del modelo siguiendo las etapas de comprensión del dataset, construcción del dataset, elaboración del modelo, entrenamiento, ajustes y análisis de resultados. Las librerías a usar para la construcción de los modelos son, Keras, Tensorflow de Python. De igual manera, se realizará la construcción en la plataforma Colab para su desarrollo colaborativo.

2 Comprensión del dataset

Para iniciar con la compresión del dataset, se realiza la descarga del dataset Traffic Sign Classification and Recognition de Kaggle, el cual tiene un peso de 26.96MB y contiene imágenes de prueba y validación sobre señales de tránsito.

2.1 ¿Qué información presenta el dataset?

El conjunto de datos contiene 6348 etiquetas de categorías etiquetadas manualmente. Las etiquetas incluyen las siguientes 10 categorías correspondientes a diez clases diferentes de señales de tráfico.

Todos los datos han sido divididos manualmente en un conjunto de entrenamiento y un conjunto de pruebas de acuerdo con la proporción, y se proporciona el archivo json de anotación del conjunto de entrenamiento correspondiente. El conjunto de datos contiene dos carpetas de entrenamiento y prueba, así como un archivo train.json unificado para guardar las anotaciones de todos los conjuntos de datos de entrenamiento. Las etiquetas de imagen guardan el diccionario serializado en formato json, y todas las anotaciones se guardan en train.json bajo la clave llamada "annotations" en el archivo.

2.2 ¿Describir las características de las imágenes?

Las imágenes tienen un tamaño de 224 x 224 píxeles y vienen en formato JPG, estás imágenes vienen divididas en diferentes paquetes que llamaremos categorías o clases. Cada imagen tiene una particularidad distinta, ya que la calidad de la imagen cambia así como la posición de la misma, unas vienen centradas, otras no, incluso algunas vienen un poco más opacas o en una orientación distinta.

(a) GuideSign (b) M1 (c) M4 (d) M5

(e) M6 (f) M7 (g) P1 (h) P10_50

(i) P12 (j) W1

2.3 Indicar conjuntos de entrenamiento y pruebas

Al descargar el dataset, se puede evidenciar que son separadas las imágenes en dos paquetes, train_dataset para las imágenes de entrenamiento y test_dataset para las imágenes de prueba. Sumando la cantidad de las imágenes de ambas carpetas, dan un total de 6024 imágenes de entrenamiento y 324 imágenes de prueba. Sin embargo, no vienen en la proporción que indica el taller, ya que en este caso tenemos 95% imágenes de entrenamiento y 5% imágenes de prueba aproximadamente.

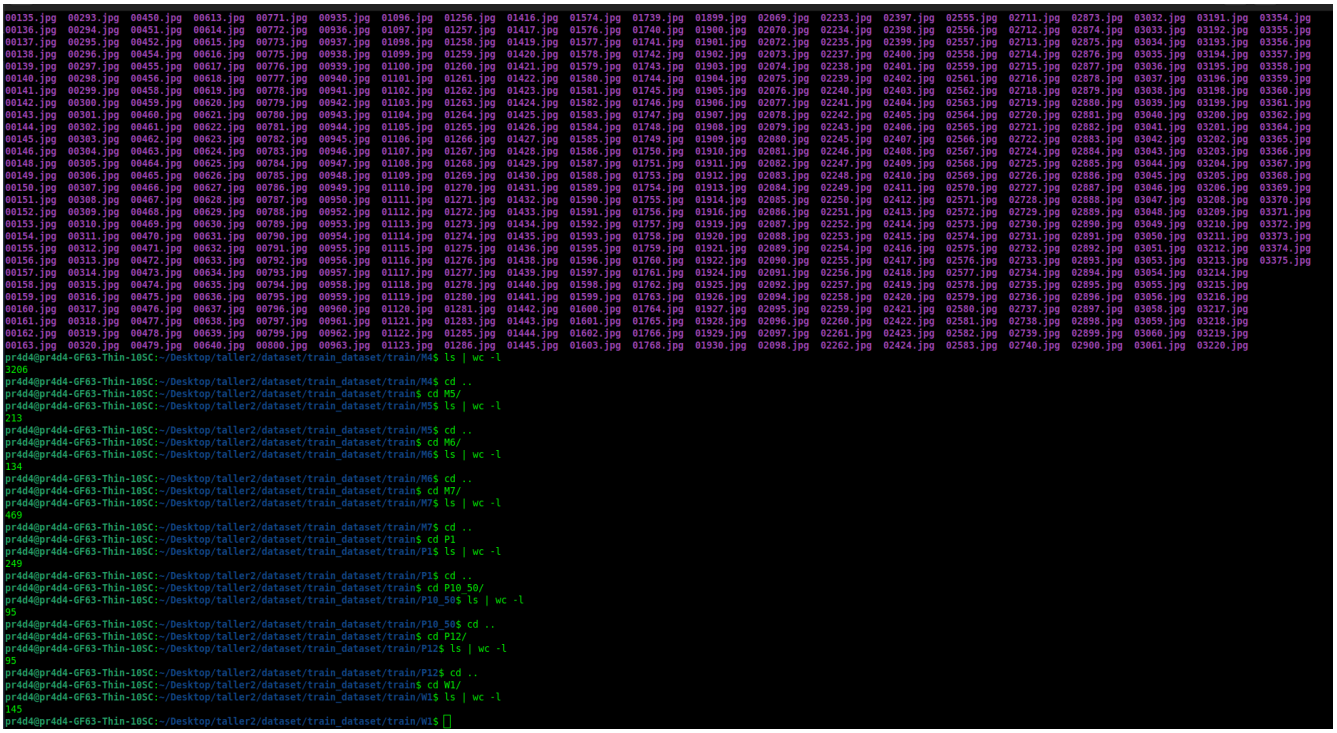
2.4 Indicar las clases de clasificación

El conjunto de datos, viene en 10 clases diferentes, las cuales son: "GuideSign", "M1", "M4", "M5", "M6", "M7", "P1", "P10_50", "P12", "W1". Como podemos observar a continuación:

```
In [53]: class_list = os.listdir("dataset/train_dataset/train")
         class_list

Out[53]: ['M6', 'W1', 'P10_50', 'P1', 'M7', 'GuideSign', 'P12', 'M4', 'M5', 'M1']
```

Figure 1: Se cuentan las imágenes de cada paquete



Clases o Categorías			
Categoría	Entrenamiento	Pruebas	Total
GuideSign	1171	62	1233
M1	247	14	261
M4	3206	169	3375
M5	213	12	225
M6	134	8	142
M7	469	25	494
P1	249	14	263
P10_50	95	6	101
P12	95	6	101
W1	145	8	153

3 Construcción del dataset

Para construir el dataset conforme a lo especificado en el taller, es necesario distribuir las imágenes de forma manual, para que sea un 80% de imágenes de entrenamiento y un 20% de imágenes de prueba. Se hace uso de la siguiente serie de comandos de linux para mover las imágenes en las carpetas correspondientes y así cambiar el dataset por 80% - 20% teniendo en cuenta la proporción de distribución de imágenes para que por cada clase se tenga el mismo porcentaje de imágenes respecto a las demás y a todo el dataset. De esta manera, el dataset queda construido de la siguiente forma:

Figure 2: Comandos para mover las imagenes

```
ls | wc -l
find train_dataset/train/GuideSign/ -maxdepth 1 -type f | head -185 | xargs -I{} mv {} test_dataset/test/GuideSign/
find train_dataset/train/M1/ -maxdepth 1 -type f | head -38 | xargs -I{} mv {} test_dataset/test/M1/
find train_dataset/train/M4/ -maxdepth 1 -type f | head -506 | xargs -I{} mv {} test_dataset/test/M4/
find train_dataset/train/M5/ -maxdepth 1 -type f | head -33 | xargs -I{} mv {} test_dataset/test/M5/
find train_dataset/train/M6/ -maxdepth 1 -type f | head -20 | xargs -I{} mv {} test_dataset/test/M6/
find train_dataset/train/M7/ -maxdepth 1 -type f | head -74 | xargs -I{} mv {} test_dataset/test/M7/
find train_dataset/train/P1/ -maxdepth 1 -type f | head -39 | xargs -I{} mv {} test_dataset/test/P1/
find train_dataset/train/P10_50/ -maxdepth 1 -type f | head -14 | xargs -I{} mv {} test_dataset/test/P10_50/
find train_dataset/train/P12/ -maxdepth 1 -type f | head -14 | xargs -I{} mv {} test_dataset/test/P12/
find train_dataset/train/W1/ -maxdepth 1 -type f | head -23 | xargs -I{} mv {} test_dataset/test/W1/
```

Clases o Categorías			
Categoría	Entrenamiento	Pruebas	Total
GuideSign	986	247	1233
M1	209	52	261
M4	2700	675	3375
M5	180	45	225
M6	114	28	142
M7	395	99	494
P1	210	53	263
P10_50	81	20	101
P12	81	20	101
W1	122	31	153

4 Elaboración del modelo

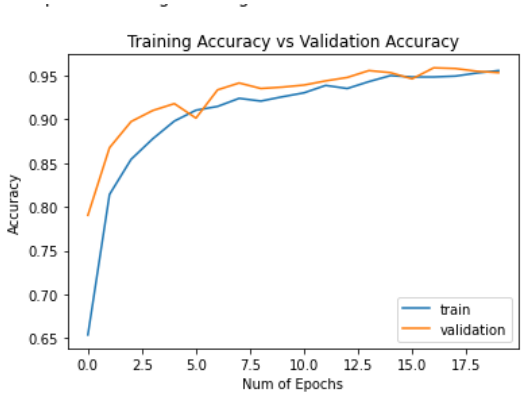
El modelo se elabora con una Red Neuronal Convolucional. En este caso, una vez se extraen las características y se seleccionan las clases con las que se desea trabajar, se procede a construir el modelo e iniciar un entrenamiento de clasificación y de pruebas. En este caso, para el modelo 1, se tienen en cuenta las siguientes características de la arquitectura, que serán datos que usaremos como parámetros al construir la red neuronal apoyados por las librerías de Python.

- Capa convolucional con 32 detectores de características, tamaño de kernel de 5 x 5 y una función de activación lineal rectificada (ReLu) de paso 1 con panding.
- Capa de max pooling de tamaño 5x5
- Capa de Flattening
- Capa full conectada con 100 neuronas y función de activación lineal rectificada (ReLu).
- Capa de salida del mismo número de las clases a clasificar y función de activación softmax

5 Entrenamiento

El modelo se entrenó de acuerdo a las consideraciones del taller, en este caso, durante 20 épocas con un tamaño de bloque de 200 (Batch), los resultados obtenidos se muestran a continuación, luego de la etapa de entrenamiento y pruebas.

A continuación, Podemos apreciar en las gráficas de validación el desempeño del algoritmo a lo largo de cada una de las épocas con respecto a nuevos datos en el proceso de entrenamiento, y la validación loss donde revisamos la medida de valuación de la función de perdida, es decir, el error del modelo en la fase de validación.



(a) Gráfica de entrenamiento y pruebas (Accuracy)

(b) Gráfica de entrenamiento y pruebas (Loss)

6 Elaboración del modelo 2

El modelo empleado para la red neuronal convolucional tiene la siguiente configuración

- Capa convolucional con 48 detectores de características, tamaño de kernel de 3x3, funcion de activación lineal rectificada (ReLU) de paso 1 y con panding.
- Capa de max pooling de tamaño 2x2
- Capa convolucional con 96 detectores de características, tamaño de kernel de 3x3, funcion de activación lineal rectificada (ReLU) de paso 1 y con panding.
- Capa convolucional con 96 detectores de características, tamaño de kernel de 3x3, funcion de activación lineal rectificada (ReLU) de paso 1 y con panding.
- Capa de max pooling de tamaño 2x2
- Capa de Flattening
- Capa full conectada con 100 neuronas y funcion de activación lineal rectificada (ReLU)
- Capa full conectada con 100 neuronas y funcion de activación lineal rectificada (ReLU)
- Capa de salida del mismo numero de las clases a clasificar y función de activación softmax.

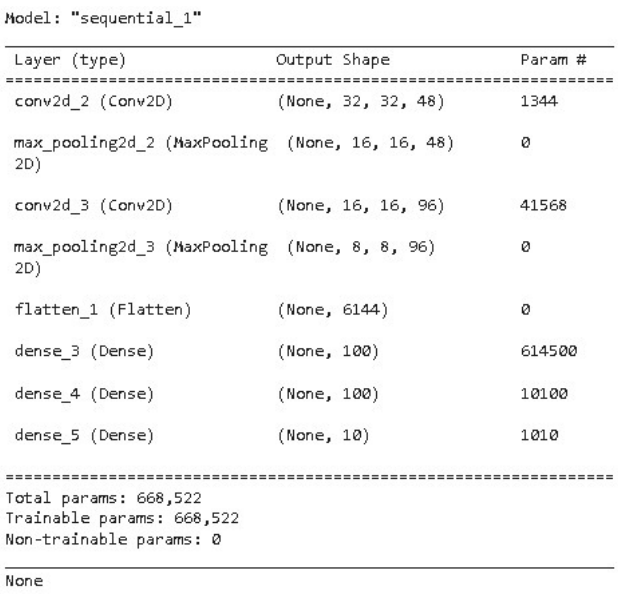


Figure 3: Parámetros del modelo CNN en la implementación con TF y Keras

7 Entrenamiento 2

El modelo se entrenó de acuerdo a las consideraciones del taller, en este caso, durante 20 épocas con un tamaño de bloque de 200 (Batch), los resultados obtenidos se muestran a continuación, luego de la etapa de entrenamiento y pruebas:

- loss: 0.0619
- accuracy: 0.9818
- val_loss: 0.2320
- val_accuracy: 0.9467

De acuerdo con estos resultados la red neuronal convolucional ha logrado una alta precisión en el conjunto de entrenamiento, con una precisión del 98.18% y una pérdida del 0.0619, por su parte la precisión en el conjunto de validación es ligeramente menor, pero aún así es alta, con una precisión del 94.67% y una pérdida del 0.2320.

La diferencia entre la precisión del conjunto de entrenamiento y la precisión del conjunto de validación indica que la red podría estar sufriendo de sobreajuste (overfitting), es decir, que está memorizando demasiado los datos de entrenamiento y no está generalizando bien para nuevos datos, puede ser necesario evaluar la red en un conjunto de datos completamente nuevo para confirmar su capacidad de generalización y evitar el sobreajuste.

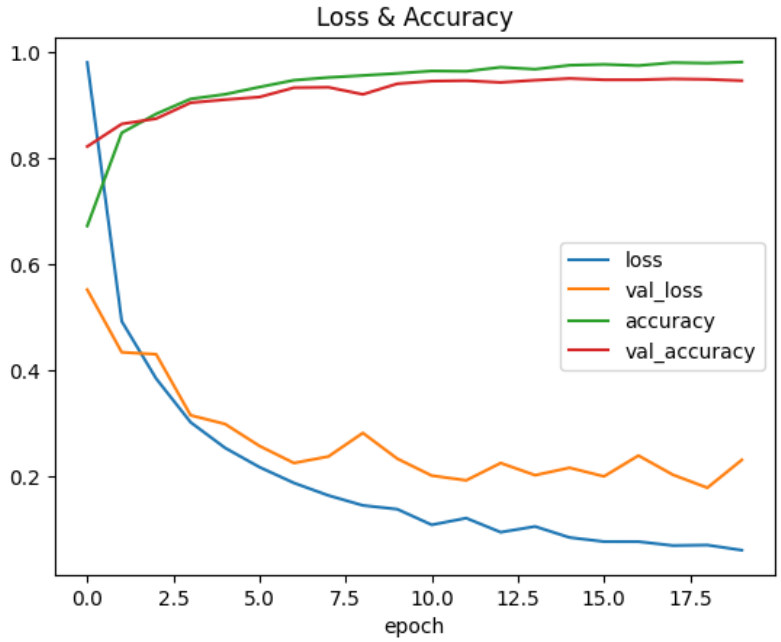


Figure 4: Resultado del proceso de entrenamiento de la CNN

8 Ajustes

No se puede determinar con certeza el balanceo de los datos sólo a partir de la matriz de confusión, ya que ésta sólo muestra la distribución de las predicciones y las etiquetas verdaderas. Sin embargo, se puede notar que algunas clases tienen un número muy pequeño de muestras, con menos de 10 muestras cada una. Esto puede indicar que el conjunto de datos no está equilibrado y puede llevar a una clasificación deficiente en estas clases. Sería importante revisar la distribución de clases en el conjunto de datos y, en caso de que haya un desequilibrio, considerar técnicas de equilibrado de datos, como aumentación de datos o submuestreo/sobremuestreo de clases minoritarias.

9 Análisis de Resultados

9.1 Modelo 1

Para el modelo 1, con la matriz de confusión se visualiza el desempeño de la red supervisada. Podemos observar por cada columna de la matriz, el número de predicciones por clase y en cada fila apreciar las instancias de los valores reales.

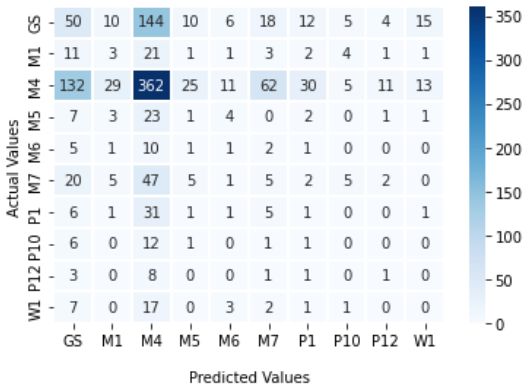
El modelo 1, tiene una precisión baja, aunque detecta algunas clases, se logra confundir con otra, esto quiere decir que el modelo no logra clasificar de forma correcta todos los casos.

9.2 Modelo 2

```
Evaluate on test data
11/11 [==============================] - 1s 121ms/step - loss: 0.1982 - accuracy: 0.9568
test loss, test acc: [0.19824513792991638, 0.9567901492118835]
```

Figure 5: Resultado del proceso de validación de la CNN

En este caso, podemos ver que el modelo tiene un rendimiento muy bueno para la mayoría de las clases, con altos números en la diagonal principal de la matriz. Sin embargo, hay algunas clases donde el modelo tiene dificultades



(a) Matriz de confusión

	precision	recall	f1-score	support
GuideSign	0.18	0.20	0.19	247
M1	0.06	0.06	0.06	52
M4	0.53	0.54	0.53	675
M5	0.02	0.02	0.02	45
M6	0.05	0.04	0.04	28
M7	0.05	0.05	0.05	99
P1	0.02	0.02	0.02	53
P10_50	0.00	0.00	0.00	20
P12	0.07	0.05	0.06	20
W1	0.00	0.00	0.00	31
accuracy			0.33	1270
macro avg	0.10	0.10	0.10	1270
weighted avg	0.33	0.33	0.33	1270

(b) Reporte

para hacer predicciones precisas. Por ejemplo, la clase 0 (GuideSign) tiene 60 predicciones correctas y solo una incorrecta, mientras que la clase 2 (M4) tiene 167 predicciones correctas y solo dos incorrectas. Por otro lado, hay clases con un número relativamente pequeño de muestras, como la clase 6 (P12) y la clase 7 (W1), por lo que la capacidad del modelo para clasificar estas clases puede ser menos confiable debido a la falta de datos.

En general, la matriz de confusión muestra que el modelo tiene un rendimiento sólido en la mayoría de las clases, pero puede haber áreas donde se pueda mejorar el rendimiento a través de la recopilación de datos adicionales o la optimización del modelo.

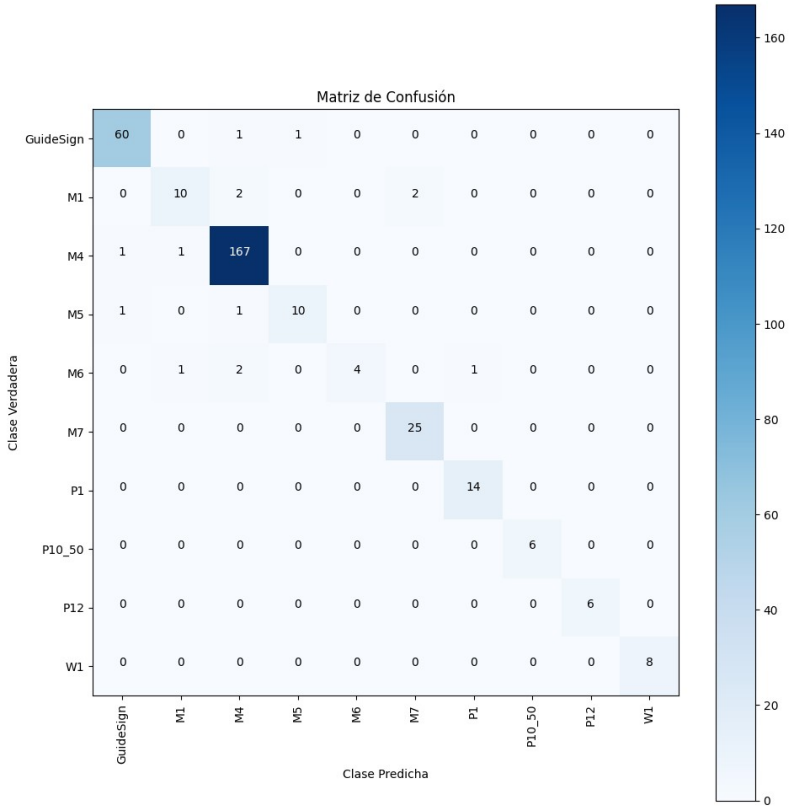


Figure 6: Matriz de confusión de la CNN

10 Conclusiones

Se puede concluir que el modelo tiene un buen desempeño para clasificar algunas de las clases, mientras que para otras clases, la precisión disminuye pero no de manera drástica.

Por ejemplo, podemos ver que la clase 2 (M4) fue clasificada con alta precisión, ya que sólo hubo un error en la clasificación de 169 imágenes. De manera similar, las clases 6 y 7 (P1 y P10_50) fueron clasificadas con precisión perfecta, sin errores. Sin embargo, para algunas otras clases, como la clase 0 (GuideSign), la precisión es menor, ya que hubo 2 errores en la clasificación de 62 imágenes.

Podría ser necesario analizar más a fondo el modelo y los datos de entrenamiento para determinar las posibles causas de estas imprecisiones y tomar medidas para mejorar el modelo.

Referencias

https://www.tensorflow.org/guide/keras/train_and_evaluate?hl=es-419