



# Informe Lab 3




## Preparación de los datos

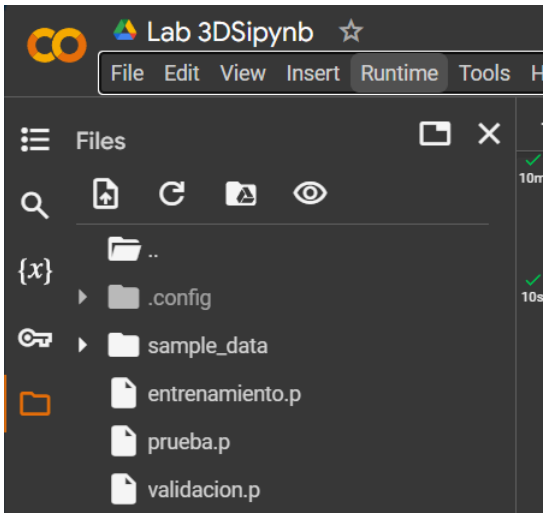
### Descargar el conjunto de datos

Se descargaron los datos

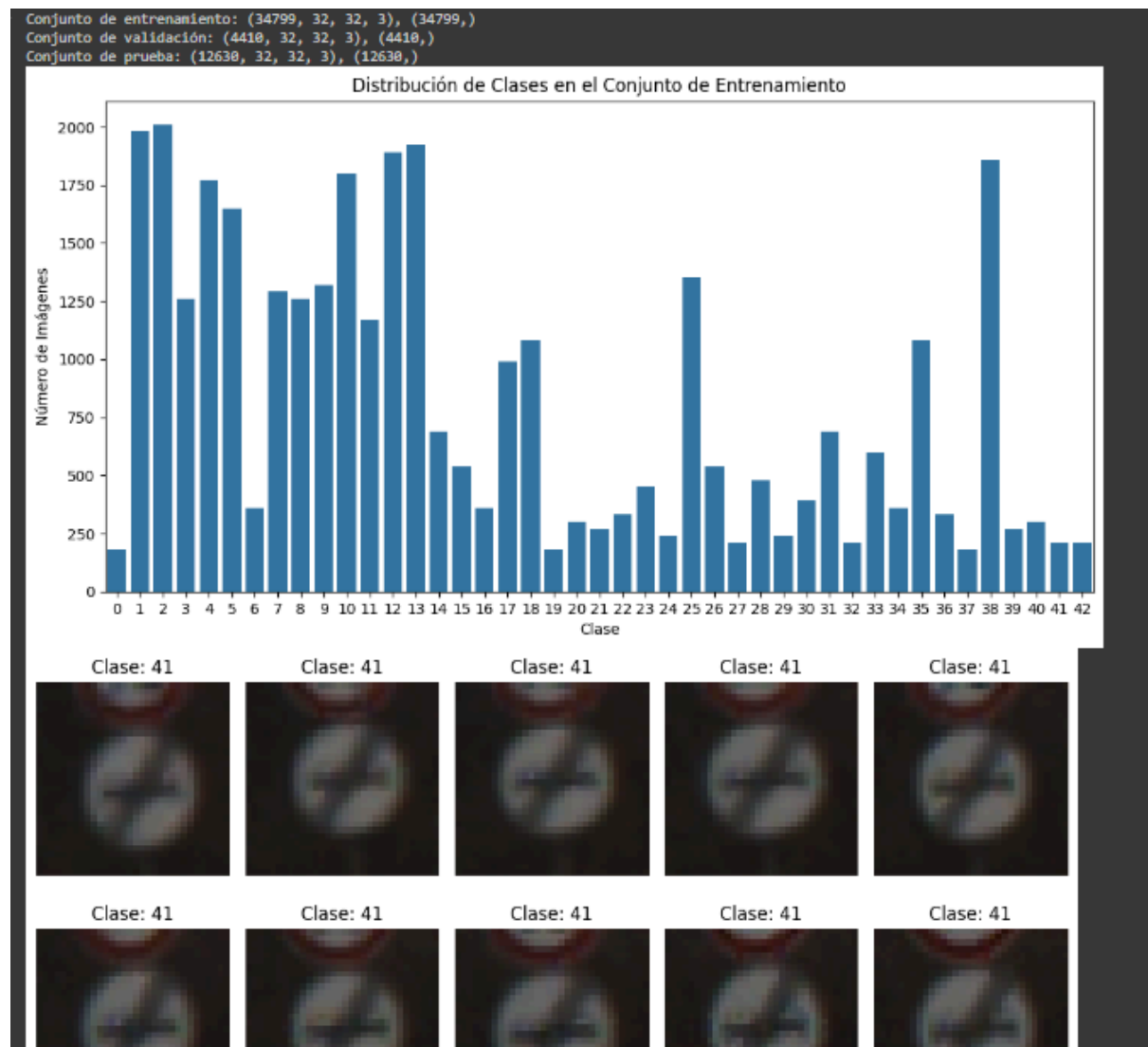
Name	Date modified	Type
 _MACOSX	8/10/2024 11:47 AM	File folder
 Datos_Rotulos_Trafico	8/10/2024 11:47 AM	File folder

### Dividir el conjunto de datos en entrenamiento, validación y prueba

 entrenamiento.p	8/10/2024 11:47 AM	P File
 prueba.p	8/10/2024 11:47 AM	P File
 validacion.p	8/10/2024 11:47 AM	P File



## Preprocesamiento de imágenes



De manera que se cargaron las imágenes, así mismo, se tuvieron que normalizar las imágenes al dividir las sobre 255.0 píxeles. Además, se codificaron en one hot encoded para las etiquetas y se setearon la cantidad de clases que eran 43.

## Implementación de la arquitectura Le-Net

### Presentar la arquitectura Le-Net en detalle

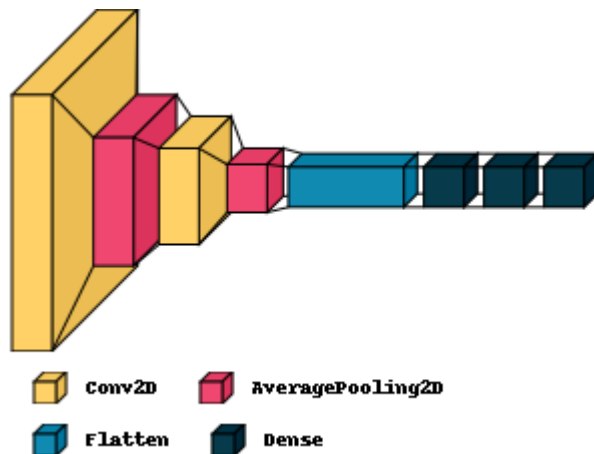
Consiste en las siguientes capas:

#### 1. Capa Convolutiva 1:

- Filtros: 6 filtros de tamaño 5x5.
- Función de Activación: ReLU (Rectified Linear Unit) introduce no linealidad al modelo, permitiendo aprender patrones más complejos.
- Padding: 'same' asegura que la salida tenga las mismas dimensiones que la entrada.

- Entrada: Imágenes de 32x32 píxeles con 3 canales de color (RGB).
  - Salida: Mapas de características de 32x32x6.
2. Capa de Pooling Promedio 1:
- Tamaño del Pooling: 2x2. Reduce las dimensiones de los mapas de características, disminuyendo la cantidad de parámetros y haciendo el modelo más robusto a pequeñas variaciones en la posición de las características.
  - Salida: Mapas de características de 16x16x6.
3. Capa Convolutiva 2:
- Filtros: 16 filtros de tamaño 5x5. Extrae características de mayor nivel de los mapas de características generados por la capa anterior.
  - Función de Activación: ReLU.
  - Padding: 'valid' significa que no se aplica padding, lo que reduce las dimensiones de la salida.
  - Salida: Mapas de características de 10x10x16.
4. Capa de Pooling Promedio 2:
- Tamaño del Pooling: 2x2. Reduce aún más las dimensiones de los mapas de características.
  - Salida: Mapas de características de 5x5x16.
5. Capa de Aplanamiento:
- Convierte los mapas de características multidimensionales en un vector unidimensional para que puedan ser procesados por las capas fully connected.
  - Salida: Vector de 400 elementos (5x5x16).
6. Capa Densa 1:
- Neuronas: 120. Primera capa fully connected, que aprende combinaciones lineales de las características extraídas por las capas convolucionales.
  - Función de Activación: ReLU.
  - Salida: Vector de 120 elementos.
7. Capa Densa 2:
- Neuronas: 84. Segunda capa fully connected, que refina aún más las representaciones aprendidas.
  - Función de Activación: ReLU.
  - Salida: Vector de 84 elementos.
8. Capa Densa 3 (Salida):
- Neuronas: 43. Capa de salida, que produce la probabilidad de que la imagen de entrada pertenezca a cada una de las 43 clases.
  - Función de Activación: Softmax. Asegura que la suma de las probabilidades de todas las clases sea 1.
  - Salida: Vector de 43 elementos, representando las probabilidades de cada clase.

## Diseño de la Le-Net



### Explicación de las fases

- Convolución: Aplica filtros a la imagen de entrada para extraer características como bordes, texturas, etc. Cada filtro produce un mapa de características.
- Función de Activación (ReLU): Introduce no linealidad al modelo, permitiendo aprender patrones más complejos. ReLU activa una neurona solo si su entrada es positiva, de lo contrario, la salida es cero.
- Pooling: Reduce las dimensiones de los mapas de características, disminuyendo la cantidad de parámetros y haciendo el modelo más robusto a pequeñas variaciones en la posición de las características.

## Construcción del modelo

La función de pérdida y el optimizador son dos componentes fundamentales en el entrenamiento de redes neuronales.

Función de Pérdida:

- Objetivo: Mide cuán bien está prediciendo el modelo en comparación con los valores reales.
- Importancia: Proporciona una señal al optimizador sobre la dirección en la que debe ajustar los pesos del modelo para mejorar las predicciones.

Optimizador:

- Objetivo: Ajusta los pesos del modelo para minimizar la función de pérdida.
- Importancia: Determina cómo el modelo aprende de los datos y converge hacia una solución óptima.

En resumen, la función de pérdida indica al optimizador cuán lejos está el modelo de la solución ideal, y el optimizador utiliza esta información para ajustar los pesos del modelo y mejorar su rendimiento. La elección de la función de pérdida y el optimizador depende del problema específico que se esté abordando.

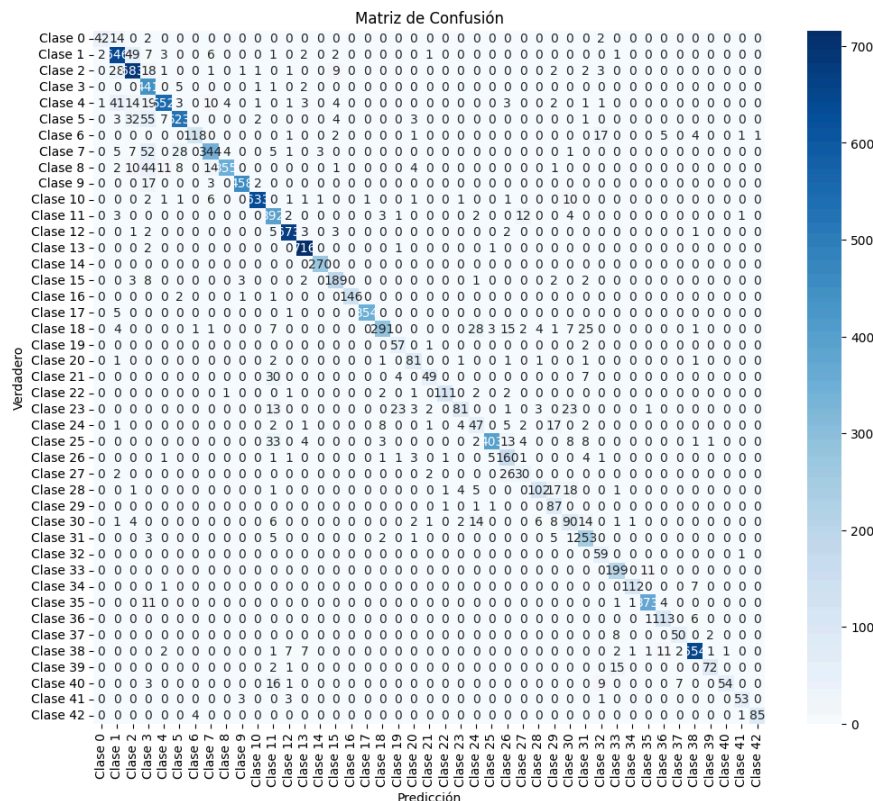
# Entrenamiento del modelo

Se generó un modelo exitoso

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 4)	496
average_pooling2d (AveragePooling2D)	(None, 16, 16, 4)	0
conv2d_1 (Conv2D)	(None, 12, 12, 16)	2,416
average_pooling2d_1 (AveragePooling2D)	(None, 6, 6, 16)	0
flatten (Flatten)	(None, 576)	0
dense (Dense)	(None, 120)	69,240
dense_1 (Dense)	(None, 84)	10,164
dense_2 (Dense)	(None, 41)	1,695
Total params: 85,911 (335.67 KB)		
Trainable params: 85,911 (335.67 KB)		

# Evaluación del modelo

	precision	recall	f1-score	support
Clase 0	0.93	0.70	0.80	60
Clase 1	0.85	0.90	0.88	720
Clase 2	0.85	0.91	0.88	750
Clase 3	0.64	0.98	0.78	450
Clase 4	0.95	0.84	0.89	660
Clase 5	0.92	0.83	0.87	630
Clase 6	0.96	0.79	0.86	150
Clase 7	0.89	0.76	0.82	450
Clase 8	0.98	0.79	0.87	450
Clase 9	0.98	0.95	0.97	480
Clase 10	0.99	0.96	0.97	660
Clase 11	0.75	0.93	0.83	420
Clase 12	0.97	0.98	0.97	690
Clase 13	0.97	0.99	0.98	720
Clase 14	0.99	1.00	0.99	270
Clase 15	0.88	0.90	0.89	210
Clase 16	1.00	0.97	0.99	150
Clase 17	1.00	0.98	0.99	360
Clase 18	0.94	0.75	0.83	390
Clase 19	0.66	0.95	0.78	60
Clase 20	0.81	0.90	0.85	90
Clase 21	0.86	0.54	0.67	90
Clase 22	0.97	0.93	0.95	120
Clase 23	0.87	0.54	0.67	150
Clase 24	0.46	0.52	0.49	90
Clase 25	0.98	0.84	0.90	480
Clase 26	0.70	0.89	0.78	180
Clase 27	0.59	0.50	0.54	60
Clase 28	0.88	0.68	0.77	150
Clase 29	0.61	0.97	0.75	90
Clase 30	0.56	0.60	0.58	150
Clase 31	0.79	0.94	0.85	270
Clase 32	0.63	0.98	0.77	60
Clase 33	0.87	0.95	0.91	210
Clase 34	0.97	0.93	0.95	120
Clase 35	0.96	0.96	0.96	390
Clase 36	0.85	0.94	0.89	120
Clase 37	0.85	0.83	0.84	60
Clase 38	0.97	0.95	0.96	690
Clase 39	0.95	0.80	0.87	90
Clase 40	0.98	0.60	0.74	90
Clase 41	0.93	0.88	0.91	60
Clase 42	0.99	0.94	0.97	90
accuracy			0.89	12630
macro avg	0.86	0.85	0.85	12630
weighted avg	0.90	0.89	0.89	12630



Se obtuvo una precisión en el conjunto de prueba del 89%.

De manera que con base de las métricas de precisión, recall y f1-score se revela que existen desempeños variados de manera que para algunas clases se tiende a predecir más que otras. Sin embargo, se encontraron rendimientos similares dependiendo de qué clase sea que serían:

- Mejores clases: 0, 9, 10, 13, 14, 16, 17, 22, 25, 35, 38 y 42 son las clases que muestran el mejor desempeño en todas las métricas, contando con una alta precisión, recall y f1-score. Esto implica que el modelo es excelente para identificar estas clases de manera que este modelo debería de ser tomado en cuenta para la exclusiva clasificación de estas clases.
- Decentes clases: 1, 2, 4, 5, 6, 7, 8, 15, 20, 26, 28, 31, 32, 36 y 37 son clases con un buen precisión y recall, pero sus f1-scores no son del todo buenos ya que se demuestra que existen falsos positivos y falsos negativos por lo que se podría mejorar el análisis de estas clases.
- Recall bajo: 0, 6, 7, 8, 21, 23, 24, 39 y 40, clases con un recall bajo, esto implica que el modelo no logra identificar los valores reales de estas clases. A pesar que la precisión sea alta, el hecho de tener un recall bajo implica que se necesita aumentar el número de ejemplos de entrenamiento y esto se podría lograr a partir de generar falsas imágenes mediante ajustes en los modelos y aumentarlos. También se podrían ajustar ciertos parámetros para que sea un poco más pulcro al momento de decidir.
- Precisión baja: 3, 11, 19, 26, 29 y 40, son clases con recall alto pero una precisión baja, esto implica que encuentra muchos falsos positivos, por lo que esto podría mejorarse a partir de encontrar alguna otra técnica de regularización o ajustar la arquitectura del modelo a partir de capas extra y tal vez a partir de capas de dropout.

- Pésimo desempeño: la clase 24 presenta todas las métricas por debajo de 0.5 por lo que el modelo no es apto para clasificarlo correctamente, probablemente se podría generar un modelo específico para esta clase.

Se puede concluir que el modelo es aceptable ya que clasifica correctamente el 89% de los valores, aunque las demás métricas indican que existe una variabilidad en el resto de clases tal cómo se pudo observar.

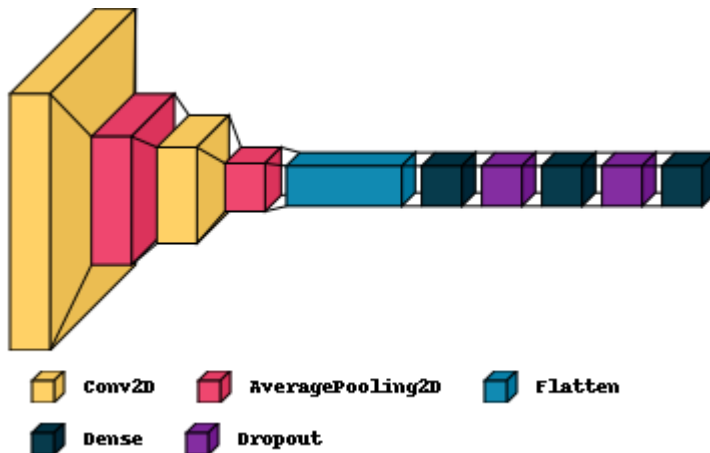
## Mejoras y experimentación

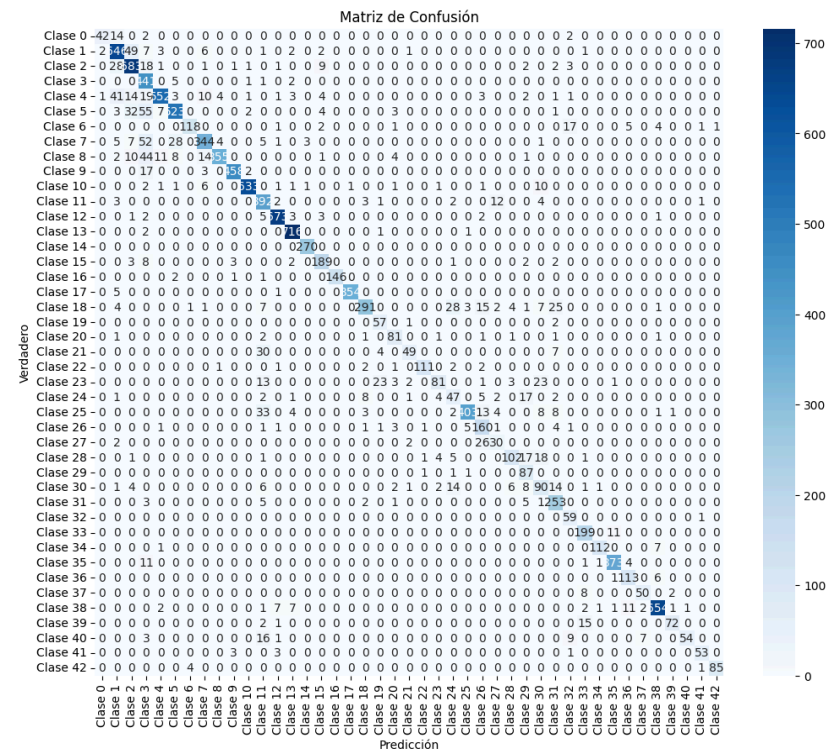
Se utilizaron capas de dropout para minimizar el overfitting de manera que se generara un modelo más potente, también se utilizó un modelo con un learning rate de 0.0005 de manera que fuera ligeramente mejor, además de que se utilizó en el learning rate un calendarizador para que fuera un poco más adaptable dependiendo de cómo se estuviera comportando el modelo.

Model: "sequential\_1"

Code cell output actions	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 10, 10, 4)	400
average_pooling2d_2 (AveragePooling2D)	(None, 10, 10, 4)	0
conv2d_3 (Conv2D)	(None, 10, 10, 16)	2,416
average_pooling2d_3 (AveragePooling2D)	(None, 5, 5, 16)	0
flatten_1 (Flatten)	(None, 576)	0
dense_3 (Dense)	(None, 128)	69,248
dropout (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	10,144
dropout_1 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 48)	3,600

Total params: 86,931 (335.67 KB)  
Trainable params: 86,931 (335.67 KB)  
Non-trainable params: 0 (0.00 B)





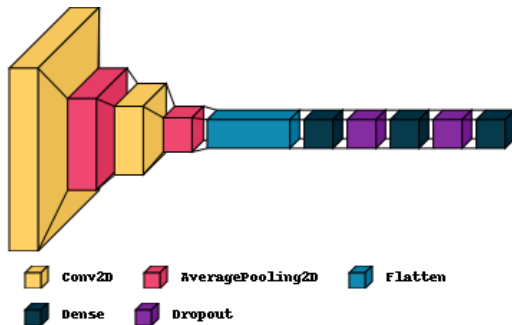
	precision	recall	f1-score	support
Clase 0	0.85	0.73	0.79	60
Clase 1	0.85	0.92	0.88	720
Clase 2	0.84	0.92	0.88	750
Clase 3	0.83	0.94	0.88	450
Clase 4	0.89	0.88	0.89	660
Clase 5	0.89	0.85	0.87	630
Clase 6	0.97	0.79	0.87	150
Clase 7	0.94	0.78	0.85	450
Clase 8	0.86	0.90	0.88	450
Clase 9	0.95	0.95	0.95	480
Clase 10	0.98	0.97	0.98	660
Clase 11	0.90	0.92	0.91	420
Clase 12	0.98	0.95	0.96	690
Clase 13	0.99	0.99	0.99	720
Clase 14	1.00	0.99	0.99	270
Clase 15	0.92	0.96	0.94	210
Clase 16	0.99	0.99	0.99	150
Clase 17	0.94	0.99	0.97	360
Clase 18	0.84	0.80	0.82	390
Clase 19	0.59	0.67	0.62	60
Clase 20	0.81	0.88	0.84	90
Clase 21	0.89	0.66	0.76	90
Clase 22	0.83	0.92	0.87	120
Clase 23	0.84	0.79	0.81	150
Clase 24	0.54	0.44	0.49	90
Clase 25	0.97	0.87	0.92	480
Clase 26	0.83	0.82	0.82	180
Clase 27	0.68	0.42	0.52	60
Clase 28	0.91	0.93	0.92	150
Clase 29	0.82	0.97	0.89	90
Clase 30	0.64	0.48	0.55	150
Clase 31	0.78	0.91	0.84	270
Clase 32	0.67	0.85	0.75	60
Clase 33	0.94	0.98	0.96	210
Clase 34	0.98	0.99	0.99	120
Clase 35	0.97	0.98	0.98	390
Clase 36	0.98	0.96	0.97	120
Clase 37	0.80	0.98	0.88	60
Clase 38	0.98	0.96	0.97	690
Clase 39	0.93	0.91	0.92	90
Clase 40	0.86	0.80	0.83	90
Clase 41	0.83	0.80	0.81	60
Clase 42	0.92	0.97	0.94	90
accuracy			0.90	12630
macro avg	0.87	0.86	0.86	12630
weighted avg	0.90	0.90	0.90	12630

El modelo fue ligeramente mejor que el modelo original en primer lugar teniendo una precisión en el conjunto de entrenamiento del 90%, siendo 1% superior al original. A su vez, al momento de evaluarlo dado que tuvo una mejora de 1% en casi todas las métricas.

Sin embargo, quisimos realizar un segundo experimento, evaluando cómo sería un modelo

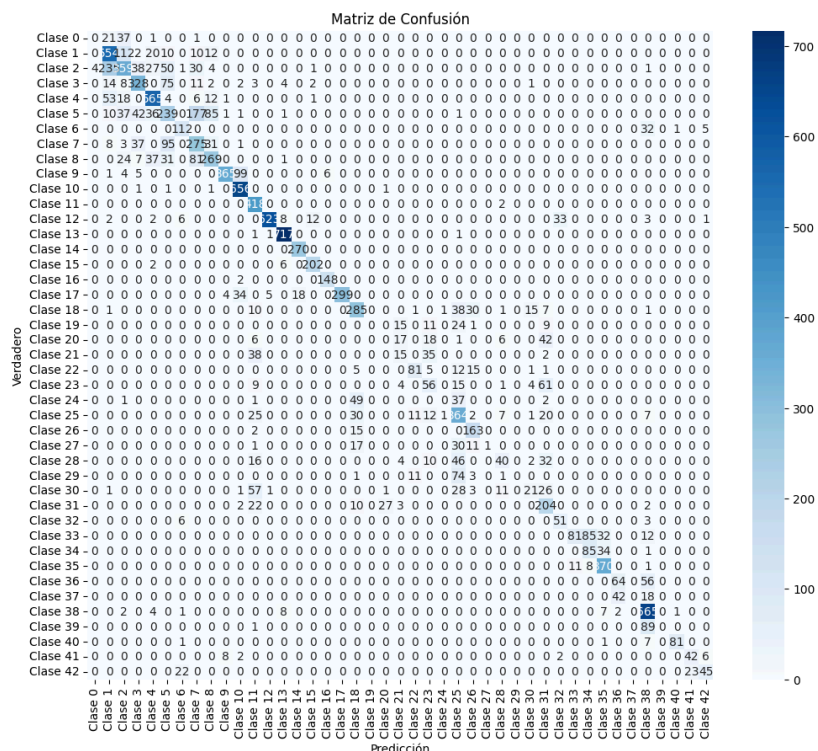


como este pero aplicando cambios en la generación de los datos con cambios en el rango de rotación, translación en xy, cortes en la imagen, zoom, volteo horizontal y vertical.  
Por lo que se hizo utilizando el mismo modelo experimental pero con esos cambios lo que resultó en el siguiente modelo, con misma estructura:



Model: "sequential_2"		
Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 16, 16, 4)	440
average_pooling2d_4 (AveragePooling2D)	(None, 8, 8, 4)	0
conv2d_5 (Conv2D)	(None, 16, 16, 16)	2,416
average_pooling2d_5 (AveragePooling2D)	(None, 8, 8, 16)	0
flatten_2 (Flatten)	(None, 512)	0
dense_6 (Dense)	(None, 100)	60,300
dropout_2 (Dropout)	(None, 100)	0
dense_7 (Dense)	(None, 50)	10,100
dropout_3 (Dropout)	(None, 50)	0
dense_8 (Dense)	(None, 2)	1,000

Reporte de Clasificación:				
	precision	recall	f1-score	support
Clase 0	0.00	0.00	0.00	60
Clase 1	0.62	0.77	0.68	720
Clase 2	0.59	0.48	0.53	750
Clase 3	0.71	0.73	0.72	450
Clase 4	0.81	0.86	0.83	660
Clase 5	0.47	0.38	0.42	630
Clase 6	0.75	0.75	0.75	150
Clase 7	0.47	0.61	0.53	450
Clase 8	0.65	0.60	0.62	450
Clase 9	0.96	0.76	0.85	480
Clase 10	0.82	0.99	0.90	660
Clase 11	0.69	1.00	0.81	420
Clase 12	0.99	0.90	0.94	690
Clase 13	0.96	1.00	0.98	720
Clase 14	0.94	1.00	0.97	270
Clase 15	0.93	0.96	0.94	210
Clase 16	0.96	0.99	0.97	150
Clase 17	1.00	0.83	0.91	360
Clase 18	0.69	0.73	0.71	390
Clase 19	0.00	0.00	0.00	60
Clase 20	0.00	0.00	0.00	90
Clase 21	0.26	0.17	0.20	90
Clase 22	0.78	0.68	0.72	120
Clase 23	0.38	0.37	0.38	150
Clase 24	0.00	0.00	0.00	90
Clase 25	0.69	0.76	0.73	480
Clase 26	0.71	0.91	0.80	180
Clase 27	1.00	0.02	0.03	60
Clase 28	0.58	0.27	0.37	150
Clase 29	0.00	0.00	0.00	90
Clase 30	0.47	0.14	0.22	150
Clase 31	0.50	0.76	0.60	270
Clase 32	0.59	0.85	0.70	60
Clase 33	0.88	0.39	0.54	210
Clase 34	0.48	0.71	0.57	120
Clase 35	0.83	0.95	0.89	390
Clase 36	0.59	0.53	0.56	120
Clase 37	0.00	0.00	0.00	60
Clase 38	0.74	0.96	0.84	690
Clase 39	0.00	0.00	0.00	90
Clase 40	0.98	0.90	0.94	90
Clase 41	0.65	0.70	0.67	60
Clase 42	0.79	0.50	0.61	90
accuracy			0.72	12630
macro avg	0.60	0.58	0.57	12630
weighted avg	0.70	0.72	0.70	12630



Tal como se observa, fue pauperrimo el experimento ya que el desempeño fue muy inferior al desempeño previo y puede ser a que al haber modificado tanto los datos de entrada el modelo esté tan sesgado a datos muy diferentes a los que se tenía con el dataset base. En pocas palabras, este modelo no sirve debido al contexto de las imágenes ya que son imágenes de señales, tal vez en imágenes un poco más interesantes estos movimientos servirían de algo.

Por ende, se concluye que el segundo modelo fue ligeramente superior al primero y el tercer modelo fue probablemente debido a la augmentación y modificación de los datos el pero modelo.