Evaluación de redes neuronales convolucionales Fiabilidad en función de sus Datos Representación

Annachiara Ruospo , Alberto Bosio†, Alessandro Ianne , Ernesto Sanchez Dipartimento di Automatica e Informatica, Politecnico di Torino, Italia †INL, Ecole Centrale de Lyon, Lyon, Francia {annachiara.ruospo, alessandro.ianne, ernesto.sanchez }@polito.it alberto.bosio@ec-lyon.fr

Resumen: las aplicaciones críticas para la seguridad se basan con frecuencia en algoritmos de aprendizaje profundo. En particular, las redes neuronales convolucionales (CNN) se implementan comúnmente en aplicaciones de conducción autónoma para realizar tareas complejas como el reconocimiento de objetos y la clasificación de imágenes.

Garantizar la confiabilidad de las CNN se está convirtiendo en un requisito urgente ya que se comportan constantemente en entornos humanos. Una tendencia común y reciente es reemplazar las CNN de precisión completa para dar paso a modelos más optimizados que explotan paradigmas de aproximación como el tipo de datos de ancho de bits reducido. Si, por un lado, esto está a punto de convertirse en una solución sólida para reducir la huella de memoria y los requisitos informáticos, puede afectar negativamente a la capacidad de recuperación de las CNN. La intención de este trabajo es evaluar la confiabilidad de un sistema basado en CNN cuando se utilizan anchos de bits reducidos para los parámetros de la red (es decir, pesos sinápticos). El enfoque evalúa el impacto de las fallas permanentes en las CNN mediante la adopción de varios esquemas de ancho de bits y tipos de datos, es decir, punto flotante y punto fijo. Esto determina el equilibrio entre la precisión de la CNN y los bits necesarios para representar los pesos de la red. La caracterización se realiza a través de un entorno de inyección de fallas construido en el marco de código abierto darknet. Los resultados experimentales muestran los efectos de las inyecciones de fallas permanentes en los pesos de LeNet-5 CNN.

Términos del índice : aprendizaje profundo, prueba, confiabilidad, inyección de fallas, seguridad, Automotor

1 INTRODUCCIÓN

Deep Learning [1] es actualmente uno de los modelos predictivos más utilizados en el campo del aprendizaje automático. En este sentido, las redes neuronales convolucionales (CNN) están ganando popularidad debido a su excelente desempeño en la resolución de problemas de aprendizaje complejos [2]. De hecho, proporcionan muy buenos resultados para muchas tareas, como el reconocimiento de objetos en imágenes/videos, descubrimiento de fármacos, procesamiento de lenguaje natural hasta juegos [3]–[5]. Las CNN son un subconjunto de las redes neuronales profundas (DNN) e incluyen varias capas: convolucional, no lineal, agrupada y totalmente conectada. Su nombre proviene de la operación lineal matemática entre matrices llamada convolución.

Las redes neuronales pueden considerarse robustas, desde una perspectiva teórica, debido a su naturaleza iterativa y aprendizaje. proceso [6]. Sin embargo, especialmente cuando se implementan en aplicaciones críticas para la seguridad, como la conducción autónoma [7], se debe evaluar su resiliencia. Más en detalle, la probabilidad de que una falla de hardware o software pueda causar una falla del sistema.

Como regla general, el análisis de confiabilidad para dispositivos electrónicos está regulado por estándares que dependen del dominio de aplicación (por ejemplo, IEC 61508 para sistemas industriales, DO-254 para aviónica, ISO 26262 para automoción) [8]. Dado que las CNN se implementan cada vez más en aplicaciones automotrices, es necesario descubrir cómo asignar los requisitos de los estándares automotrices, es decir, ISO 26262, a los sistemas de aprendizaje profundo. Un primer intento es la evaluación de la confiabilidad a través de campañas de inyección de fallas. Por lo general, para mejorar la confiabilidad de los dispositivos electrónicos en el dominio autónomo, las soluciones de prueba en el campo se integran y activan en modo misión para detectar posibles fallas permanentes antes de que puedan producir una falla. Ejemplos de tales soluciones de prueba son las técnicas de diseño para la capacidad de prueba (p. ej., BIST), los enfoques funcionales de autoprueba (p. ej., autoprueba basada en software [9]), o una combinación de ambos. Independientemente de la solución de prueba adoptada, el punto clave es la cobertura de falla lograda con respecto a los modelos de falla adoptados. Una mayor cobertura de fallas conduce a garantizar un mayor nivel de seguridad.

Las tendencias recientes en el área de investigación de redes neuronales convolucionales muestran una adopción creciente de modelos más optimizados que utilizan un tipo de datos de ancho de bits reducido en la fase de entrenamiento o inferencia. De hecho, una limitación importante sobre la adopción de la versión más nueva de las CNN es la memoria requerida para almacenar los parámetros de la red. Por ejemplo, VGG-Net [10], una de las implementaciones más profundas, requiere 500 MB de memoria, un valor que supera la posibilidad de muchas plataformas de hardware restringidas. En los últimos años, la Computación Aproximada (AxC) se ha convertido en un importante campo de investigación para mejorar tanto la velocidad como el consumo de energía en sistemas embebidos y de alto rendimiento [11]. Al relajar la necesidad de operaciones completamente precisas o completamente deterministas, la computación aproximada mejora sustancialmente la eficiencia energética y reduce los requisitos de memoria. Diversas técnicas para el cálculo aproximado.

978-1-7281-9535-3/20/$31.00 ©2020 IEEE
DOI 10.1109/DSD51259.2020.00109

ing aumenta el espacio de diseño al proporcionar otro conjunto de perillas de diseño para las compensaciones de rendimiento y precisión. Por ejemplo, la ganancia de energía entre una operación de 8 bits de baja precisión adecuada para la visión y una operación de punto flotante de doble precisión de 64 bits necesaria para el cálculo científico de alta precisión puede alcanzar hasta 50 veces si se considera el almacenamiento , transporte y computación. La ganancia en eficiencia energética (el número de cálculos por Joule) es aún mayor ya que se reduce considerablemente el retraso de las operaciones básicas.
Tener operadores más simples también reduce el costo de implementación, lo que permite que la red use más recursos en paralelo.

Al tomar prestadas estas ideas del campo de la computación aproximada, el principal desafío es encontrar una representación de datos adecuada para las CNN que se ajuste bien a las restricciones de la aplicación y el hardware. Las CNN se prestan bien a las técnicas de computación aproximada, especialmente con implementaciones de aritmética de punto fijo o de punto flotante de baja precisión, lo que expone un gran paralelismo de grano fino. Por ejemplo, en [12] los autores describen una red binaria que explota solo dos valores {−1, 1} para la representación de pesos. Otra solución propuesta es la red ternaria [13]; cuantifica los pesos en 3 valores diferentes {−1, 0, 1}.
Finalmente, XNOR-Net [14] utiliza una metodología ligeramente diferente: todos los cálculos se realizan mediante operaciones de conteo de bits y XNOR, al mismo tiempo que se reduce la precisión de todos los operandos involucrados durante el cálculo.

En este punto, la pregunta puede sonar trivial: ¿son esos modelos optimizados lo suficientemente confiables para tolerar fallas que se propagan por todo el sistema? Empieza a ser necesario evaluar el comportamiento de las CNN en un escenario defectuoso para determinar si aún se pueden implementar de manera segura en un sistema crítico para la seguridad. Estas dudas se justifican si se considera el creciente escalamiento tecnológico en la fabricación de chips. Debido a la reducción de los transistores, las plataformas de hardware más nuevas son más complejas y, al mismo tiempo, más susceptibles a fallas, aunque más rápidas.

El objetivo final de este trabajo es caracterizar el impacto de las fallas permanentes que afectan a una Red Neuronal Convolucional por medio de campañas de inyección de fallas, cuando se utiliza una representación más compacta para describir los parámetros de la red. Analizamos diferentes implementaciones de la misma arquitectura CNN, cuando se explotan diferentes tipos de datos. Además, este trabajo tiene como objetivo estudiar el nivel de criticidad de las capas de la red así como identificar todos los Safe Faults Application Dependent (SFAD), aquellos fallos que no producen ningún fallo en el modo operativo (ISO 26262).

El resto del documento está organizado de la siguiente manera. El trabajo de investigación de vanguardia y las principales contribuciones se destacan en la Sección 1.1. La Sección 2 presenta el estudio de caso, centrándose en la configuración del experimento y la técnica de conversión de peso. La Sección 3 describe el marco de inyección de fallas, mientras que la Sección 4 proporciona los resultados experimentales. Finalmente, la Sección 5 concluye el artículo delineando algunas de las posibles direcciones futuras de investigación.

1.1 Trabajo relacionado

En la literatura, cada vez se presta más atención a la Fiabilidad de las Redes Neuronales. Dependiendo de múltiples factores, como la tipología de inyección de fallas, el nivel de abstracción, los modelos de fallas, es posible identificar diferentes conjuntos de actividades de investigación interesantes.

Un conjunto significativo se enfoca en analizar un modelo de falla específico: el error suave (es decir, cambio de bit). En [15], los autores evalúan la confiabilidad de una CNN ejecutada en tres arquitecturas de GPU diferentes (Kepler, Maxwell y Pascal).
La inyección de errores suaves se ha realizado exponiendo las GPU que ejecutan la CNN bajo haces de neutrones controlados.
Un enfoque similar pero más amplio se detalla en [16], donde los autores evalúan la confiabilidad de un DNN de 54 capas (NVIDIA DriveWorks) a través de experimentos de inyección de fallas para fallas permanentes y pruebas de haz de neutrones acelerados para errores transitorios. Las fallas se inyectan en los pesos de la red y en las imágenes de entrada. Todas las inferencias se ejecutan en la GPU Volta y solo tienen como objetivo los valores de coma flotante. Más adelante, los autores presentan en [17] un análisis diferente.
Caracterizan la propagación de errores leves desde el hardware al software de aplicación de diferentes CNN. Las inyecciones se realizan mediante el uso de un simulador de red neuronal profunda basado en un marco de simulador de código abierto, Tiny CNN [18]. Gracias a la flexibilidad del simulador, es posible caracterizar cada capa para un análisis más preciso. Un marco diferente se muestra en [19]: Ares, un marco ligero de inyección de fallas de redes neuronales profundas.

Los autores presentan un estudio empírico sobre la resiliencia de tres tipos destacados de DNN (totalmente conectados, CNN y unidad recurrente cerrada). En particular, se centran en dos tipos de datos de punto fijo para cada red: Q3,13 , es decir, 3 bits enteros y 13 fraccionarios, y Q2,6. Sus experimentos demuestran que el tipo de datos Q2,6 optimizado es 10 veces más tolerante a fallas. La razón radica en el hecho de que el rango más grande e innecesario de valores enteros aumenta la posibilidad de que ocurran fallas. Vale la pena señalar que este resultado está en línea con nuestros resultados recopilados, presentados en la Sección 4.2. Es una tendencia común explorar los cálculos de punto fijo para sistemas integrados de potencia ultrabaja con un presupuesto de potencia limitado [20]. Finalmente, en [21], los autores analizan la confiabilidad de un acelerador de redes neuronales profundas siguiendo un enfoque de síntesis de alto nivel (HLS). Caracterizan los efectos de las fallas permanentes y transitorias mediante la explotación de un marco de inyector de fallas integrado en el diseño RTL del acelerador. Las fallas se inyectan durante los ciclos de inferencia solo en un subconjunto de registros: aquellos que se encargan de almacenar pesos, valores de entrada e intermedios utilizados a lo largo del trabajo de inferencia, sin considerar los efectos de las fallas en las otras unidades de ruta de datos. En cuanto a la representación de datos, realizan los experimentos adoptando únicamente un modelo de baja precisión de punto fijo de 16 bits, alegando una pérdida de precisión insignificante con respecto a un modelo de datos de precisión completa.

La principal contribución de este artículo es un análisis exhaustivo sobre el comportamiento de las CNN en función de su representación de datos. En un trabajo anterior [22], evaluamos el impacto de las fallas permanentes que afectan a las CNN a través de campañas de inyección de fallas de software. Sin embargo, solo se ha considerado una representación de punto flotante. En comparación con los análisis del estado del arte [19] [21], se proporciona un espectro más amplio de representaciones de punto fijo (cinco tipologías que van desde 32 bits hasta 6 bits). Con este objetivo, se inyectan fallas permanentes en relación con la capa corrupta y el tipo de datos.

2 ESTUDIO DE CASO

En esta sección se pretende presentar el caso de estudio. Como se indicó anteriormente, explotamos el marco DNN de código abierto de darknet [23]. Implementado en lenguaje C y CUDA, es adecuado para realizar despliegues de extremo a extremo de arquitecturas de redes neuronales de una manera muy sencilla. Además, proporciona un entorno muy simple donde se pueden ejecutar varias configuraciones de redes neuronales profundas para realizar trabajos de entrenamiento o inferencia. Como se indicó anteriormente, el trabajo se centra en una categoría de DNN, denominada Redes neuronales convolucionales (CNN). Estas redes artificiales son ampliamente explotadas en campos como la clasificación de imágenes y la detección de objetos.

Entre todas las posibles CNN existentes, nuestro interés recae en LeNet-5 [24], un conocido clasificador para tareas de reconocimiento de dígitos escritos a mano introducido por Y. Lecun et al en 1998. La arquitectura de red está compuesta por 1 capa de entrada, 5 capas ocultas y 1 capa de salida, cuya tipología va desde capas Convolucionales, Totalmente Conectadas y Max-Pool. Para la evaluación de la confiabilidad de la red, el comportamiento de la categoría de capa Max Pool está fuera del alcance de nuestro análisis debido al hecho de que no se implementan operaciones aritméticas.

Para ejecutar los experimentos, se seleccionó la base de datos MNIST [25], un conjunto de datos bien conocido que se utiliza para evaluar la precisión de los nuevos modelos emergentes. Está compuesto por 60.000 imágenes para entrenamiento y 10.000 para test/validación del modelo, codificadas en 28×28 píxeles en escala de grises. Sin embargo, para reducir los costos y el tiempo computacional, se seleccionó aleatoriamente una carga de trabajo de 2023 imágenes del conjunto de prueba/validación del MNIST para todos los experimentos. Además, dado que nos estamos centrando en la fase de inferencia y en la respuesta de la red en un escenario defectuoso, se ha adoptado un conjunto de pesos preentrenados. Está disponible en el sitio web de darknet e incluye todos los pesos en punto flotante de 32 bits. La figura 1 destaca la distribución de valores de estos pesos preentrenados. Como se evidencia, todos los valores están en el rango de -0.6 a 0.6 con la mayoría de ellos alrededor de cero.

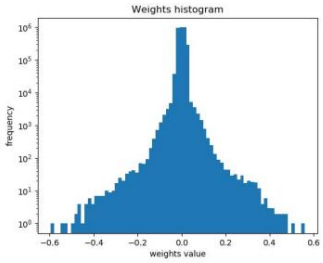


Fig. 1: Distribución de Valores de Pesos Pre-Entrenados.

La confiabilidad de la CNN mencionada anteriormente se evaluó ejecutando muchos experimentos de inyección de fallas con dos tipos de datos diferentes. El primer conjunto de experimentos considera la CNN de destino con pesos de punto flotante de 32 bits, mientras que el segundo explota una representación de punto fijo con un ancho de bits que va desde 32 hasta 6 bits. El siguiente subapartado (2.1) profundizará en la metodología seguida para pasar de números de punto flotante a números de punto fijo, es decir, la conversión. Luego, la Sección 3 establecerá los dos entornos de inyección de fallas.

2.1 Conversión de pesos En el

campo de las redes neuronales, un enfoque común para reducir el ancho de bits de los pesos y los valores de activación es adoptar esquemas de cuantificación [26] [27]. Sin embargo, el marco darknet no admite operaciones con punto fijo, lo que obliga al usuario a ejecutar inferencias solo con números de punto flotante. Por esta razón, resultó más conveniente realizar una conversión en línea de los pesos de la red. Esto nos llevó a modificar ligeramente el código fuente para admitir las inferencias con una representación de menor precisión. Todas las conversiones entre punto flotante y punto fijo se han llevado a cabo mediante la integración de una biblioteca de código abierto en el marco de darknet: la biblioteca libfixmath [28].

Dado que el objetivo final es caracterizar la propagación de fallas a través de la red (acelerar los cálculos y compactar el tamaño del modelo están fuera del alcance de este trabajo), realizamos estas conversiones en línea mientras mantenemos todas las operaciones internas en punto flotante (Figura 2). Los beneficios de este enfoque son dobles: en primer lugar, no es necesario cambiar la estructura del marco cada vez que se deben realizar nuevos experimentos con un tipo de datos diferente; segundo, permite cambiar la representación sin volver a entrenar el modelo CNN para cada tipo de datos, explotando el mismo conjunto de parámetros entrenados. De esta forma, la evaluación de la confiabilidad de CNN es más rápida; es posible cambiar entre experimentos con diferente formato numérico en un tiempo razonable.

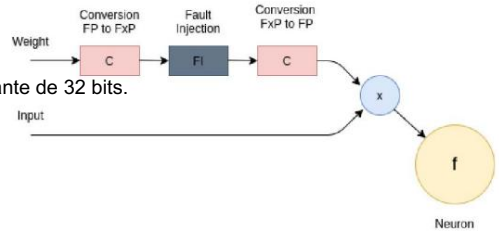


Fig. 2: Conversiones de pesos en línea.

En aras de la exhaustividad, describimos cómo se aplica la conversión en línea de un peso de punto flotante antes de llegar a una sola neurona (Figura 2). El esquema aplicado funciona de la siguiente manera:

1) El peso se convierte de punto flotante a una representación de punto fijo previamente seleccionada.
2) El peso de punto fijo se corrompe de acuerdo con una lista de fallas elegida y una ubicación de falla, es decir, se inyecta la falla.
3) El peso de punto fijo se vuelve a convertir a la representación de punto flotante para preservar la implementación nativa del marco. De tal manera, el valor obtenido refleja el mismo valor corrupto de punto fijo, mientras sigue siendo un dato de punto flotante.
4) El peso se multiplica por el valor de entrada.
5) La neurona realiza los cálculos aritméticos.

Aunque todas las operaciones de la red se ejecutan entre variables de punto flotante, debe señalarse que el

Escenario	Tipo de datos	Ancho de bit	Bits fraccionarios	[%] Pérdida de precisión
A	punto fijo	32	16	0
B	punto fijo	18	16	0
C	punto fijo	16	8	0.18
D	punto fijo	10	8	0
E	punto fijo	6	4	2.96

TABLA 1: [%] Pérdida de precisión de LeNet-5 de punto fijo.

se conserva la pérdida de precisión causada por la primera conversión. De hecho, al pasar de una representación de alta precisión (es decir, punto flotante) a uno de baja precisión (es decir, punto fijo), estamos presenciando un efecto de error de truncamiento. Entonces, convirtiendo volver de un rango estrecho de valor (punto fijo) a un rango más amplio uno (coma flotante), el error de truncamiento aún permanece. Este es importante para justificar la elección de la metodología de conversión. Además, calculamos la pérdida de precisión de la red. como resultado de la adopción de pesos de punto fijo. Como resaltados en la Tabla 1, cinco escenarios diferentes han sido analizado. La segunda columna de la tabla subraya la ancho de bit de peso total, incluyendo siempre 1 bit para el signo. La tercera columna muestra la cantidad de bits asignados a la parte fraccionaria después del punto de base. Los bits restantes se utilizan para representar la parte entera. Para calcular el exactitud de la red cuando los pesos se representan en diferentes formatos de ancho de bits, la inferencia de todas las imágenes perteneciente al conjunto de validación de la base de datos MNIST (10.000) se han ejecutado en LeNet-5, claramente sin inyectar ningún fallas, es decir, en un escenario dorado. Resultó que el la pérdida de precisión es cero en el 60% de los casos, y menor que el 3% en el 40% restante.

3 AVERÍA INYECCIÓN

La intención de la sección es describir primero la Inyección de Falla entorno construido en el marco darknet. Entonces, más se proporcionan detalles para los dos estudios de caso: las representaciones de pesos de punto flotante y de punto fijo.

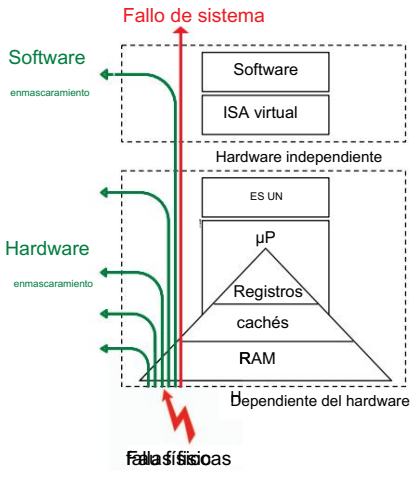


Fig. 3: Capas del sistema y propagación de fallas.

El sistema de hardware puede verse afectado por fallas causadas por defectos físicos de fabricación. Como se destaca en la Figura 3,

```
1 ejecute CNN (CNN, predicción dorada);
2 para ( i =0, i < FLo . tamaño ( ) , i ++) {
3 fallo de inyección (Flo [i], CNN);
4 ejecutar CNN (CNN, predicción defectuosa);
5 comparar (predicción defectuosa, predicción dorada);
6 fallo de liberación (Flo [i], CNN);
7 }
```

Listado 1: Pseudocódigo de inyección de fallas

las fallas podrían propagarse a través de las diferentes estructuras de hardware que componen el sistema completo. Sin embargo, podría suceder que están enmascarados durante la propagación ya sea en el a nivel tecnológico o arquitectónico [29]. cuando una falla llega a la capa de software del sistema, puede corromper los datos, instrucciones o el flujo de control. Estos errores pueden afectar la correcta ejecución del software produciendo resultados erróneos o impidiendo la ejecución de la aplicación que conduce a una terminación anormal o bloqueo de la aplicación. El software la pila puede desempeñar un papel importante en el enmascaramiento de errores; en el Al mismo tiempo, este fenómeno es implícitamente importante para el confiabilidad del sistema pero un desafío difícil para los ingenieros de prueba que tienen que garantizar la seguridad de sus sistemas. Como se indica en la introducción, el objetivo de este trabajo es investigar la efecto de fallas permanentes en la capa de software para ser independiente de la arquitectura de hardware que ejecuta el CNN (es decir, CPU, GPU o acelerador HW). como permanente falla, consideramos el modelo Stuck-at Fault (SaF) en 0/1 (SaF0 y SaF1). La ubicación de la falla (F Lo) está definida por (1).

$$F\,Lo =< Capa, Conexión, Bit, P\,olaridad > \quad (1)$$

donde Capa corresponde a la capa CNN, Conexión es el borde que conecta un nodo de la capa y Bit es uno los bits del peso asociado a la Conexión. Finalmente, la P olaridad puede ser '0' o '1' dependiendo del SaF. La culpa El inyector realmente funciona en la capa de software, y su pseudocódigo se proporciona en el seudocódigo P (1). corresponde a un simple inyector de fallas en serie que modifica la topología de CNN como se describe en (1).

El proceso de inyección de fallas consiste en lo siguiente: una vez que la CNN está completamente entrenada, se realiza una carrera dorada recopilando los resultados dorados (también conocido como predicción dorada), línea 1 en 1. Luego, se realiza el proceso de inyección de falla real. El paso inicial requiere generar la lista de fallas a ser inyectado. Esta lista de fallas debe verse como una lista de lugares dónde inyectar las fallas como se describió anteriormente. Entonces, para cualquier falla en la lista de fallas (línea 2 en 1), una predicción se realiza la ejecución y los resultados se recopilan y nombran como predicción defectuosa. Es necesario subrayar que las fallas son inyectados independientemente de su polaridad (atascado en 0 o atascado en 1). Una vez que se fija la ubicación de la falla, el bit objetivo se invierte (si es 0 se convierte en 1 y viceversa). De esta manera, no distinguir entre el efecto singular de los dos modelos de falla mientras se obtiene una gran flexibilidad para la enorme cantidad de simulaciones realizadas. En este punto (línea 5 en 1), el los resultados obtenidos se comparan con los esperados, y los resultados registrados para un análisis posterior.

En detalle, la función compare del Pseudocódigo (1) clasifica la predicción/clasificación de la CNN defectuosa

escribir el dorado. La clasificación se hace de la siguiente manera:

- Enmascarado: no se observa diferencia con la CNN defectuosa y el dorado.
- Observado: se observa una diferencia con el defectuoso CNN y el dorado. dependiendo de cuanto los resultados divergen, los clasificamos además como:
  - Seguro: la puntuación de confianza del elemento mejor clasificado varía menos de +/-5% respecto al dorado;
  - Inseguro: la puntuación de confianza del elemento mejor clasificado varía en más del +/-5 % con respecto al dorado. uno, o el elemento mejor clasificado predicho por el defectuoso CNN es diferente de lo predicho por el golden uno. Como ya se discutió en [17] este es el más falla crítica observada;

Como se informó en la sección de introducción, uno de los objetivos de este trabajo es identificar la "Aplicación de fallas seguras Dependiente" (SFAD) según la norma ISO 26262.
En este escenario, las fallas SFAD pueden ser calculadas por la unión entre falla enmascarada y segura observada como se muestra en (2).

$$SF\ AD = M\ solicitado \quad Falla\ segura\ observada \qquad (2)$$

3.1 Inyección de coma flotante

En el primer conjunto de experimentos, los pesos de conexión de LeNet-5 se representan como punto flotante binario de precisión simple formato según el estándar IEEE 754 (Figura 4).

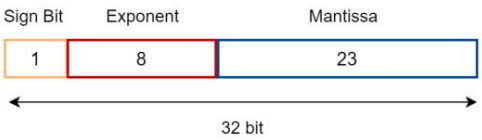


Fig. 4: Estándar de punto flotante de precisión simple IEEE 754.

Las inyecciones se han realizado sólo en los cuatro Capas de LeNet-5 que realizan cálculos aritméticos, es decir, el dos Convolucionales y los dos Completamente Conectados. Mesa 2 proporciona detalles acerca de la configuración, así como la lista de fallas de cada capa. Las dos primeras columnas de la tabla. presentar las capas objetivo; el tercero especifica la cantidad de sus pesos de conexión. El número de posibles fallas. se calcula como la multiplicación entre las conexiones número (columna 3) y el tamaño del peso (columna 4) multiplicado por 2 (es decir, atascado en 0 y atascado en 1).

Capa 0	Detalle	Conexiones Bit-Ancho 2.400 51.200		#Fallas	#inyecciones
	convolucional		32	153.600	9,039
	convolucional		32	3.276.800	9,576
	Totalmente conectado	3.211.264	32	205.520.896	9,604
2 4 6	Totalmente conectado	10.240	32	655.360	9,465

TABLA 2: Lista de fallas de LeNet-5 para la inyección de coma flotante.

Como señala la columna 5, el número total es muy alto reflejado en una campaña de inyección de fallas no manejable Tiempo de ejecución. Por lo tanto, para reducir la inyección de fallas tiempo de ejecución, seleccionamos aleatoriamente un subconjunto de fallas. A obtener resultados estadísticamente significativos con un margen de error del 1% y un nivel de confianza del 95%, un promedio de 9K Se deben considerar las inyecciones de fallas. Los numeros precisos se dan en la última columna de la Tabla 2 y tienen calculado usando el enfoque presentado en [30].
Además, es necesario subrayar que las inyecciones se han realizado en bits seleccionados al azar de todos los Pesos de conexión de punto flotante de 32 bits.

3.2 Inyección de Punto Fijo

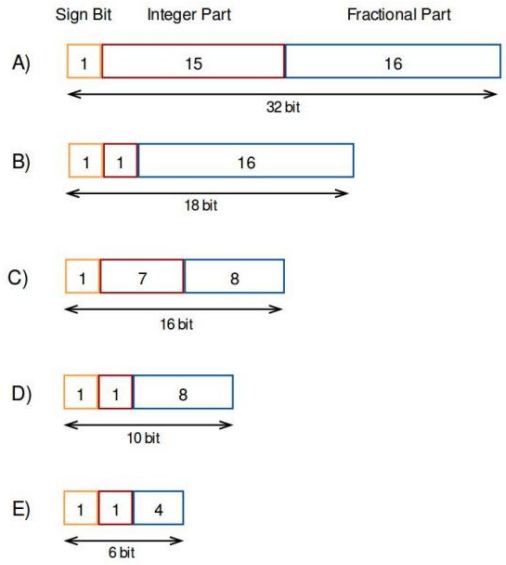


Fig. 5: Representación de datos de punto fijo.

En el segundo conjunto de experimentos, la conexión LeNet-5 los pesos se representan en un formato de punto fijo. Con el fin de reunir un espectro más amplio de resultados, ajustamos los pesos anchos de bits. Como se mencionó en la Sección 2.1, cinco diferentes Se han elegido formatos de punto fijo de tamaño decreciente: desde 32 hasta 6 bits (Figura 5). Estos emplean todo un bit para el signo, un número fijo de bits para la parte decimal y el fraccionario, es decir, los bits posteriores al punto fijo. Todas las conversiones de peso se realizan como se describe en la Sección 2.1, durante la fase de inferencia mediante el mapeo en tiempo de ejecución los valores de punto flotante a punto fijo. si de por un lado, esta técnica permite realizar los experimentos sin recurrir a computacional intensivo externo bibliotecas de punto fijo para las operaciones aritméticas; es cabe mencionar que requiere mayor esfuerzo durante el fase de inyección de fallas. De hecho, la representación correcta de los datos debe tenerse en cuenta cada vez que se coloca una falta.

Las inyecciones de fallas, como para el escenario de punto flotante, se ejecutan solo para las capas aritméticas de la red: los convolucionales y los totalmente conectados. Detalles de la tabla 3 la lista de fallas para cada capa y la cantidad de fallas que han sido inyectados. Como en el caso de los datos de punto flotante formato, la cantidad de fallas se calcula multiplicando la cantidad de las conexiones de pesos de la capa de destino con el ancho de bits de la representación de datos, multiplicado por 2 (ambos se debe considerar el 1 atascado y el 0 atascado). Estos

Capa		L0	L2	L4	L6
Detalle		Convolucional	Convolucional	Completamente Conectado	Completamente Conectado
Conexiones		2.400	51.200	3.211.264	10,240
A	ancho de bits	32	32	32	32
	#Fallas	153.600	3.276.800	205.520.896	655,360
	#Inyección	7.680	163.840	102.760	32,768
B	ancho de bits	18	18	18	18
	#Fallas	86.400	1.843.200	115.605.504	368,640
	#Inyección	4.080	87.040	54.591	17,408
C	ancho de bits	16	16	16	—
	#Fallas	76.800	1.638.400	102.771.968	327,680
	#Inyección	3.840	81.920	51.380	16,384
D	ancho de bits	10	10	10	10
	#Fallas	48.000	1.024.000	64.225.280	204,800
	#Inyección	2.160	46.080	28.901	9,216
mi	ancho de bits				6
	#Fallas	6	6 614.400	6 38.535.168	122,880
	#Inyección	28.800 1.200	25.600	16.056	5,120

TABLA 3: Lista de fallas de LeNet-5 para la inyección de punto fijo.

se proporcionan detalles para los cinco experimentos (ABC DE) realizados al disminuir el ancho de bits. Por el amor de claridad, las inyecciones se han realizado al azar seleccionando el bit defectuoso dentro del ancho de bits del peso. Para reducir los costos computacionales, solo un subconjunto de fallas ha sido elegido, de acuerdo con el trabajo presentado en [30]. Concretamente, se inyectó el 5% de fallos atascados en Capa 0 - Capa 2 - Capa 6, con respecto al total cantidad de fallas permanentes. Teniendo en cuenta la enorme cantidad de conexiones de la Capa 4, sería casi imposible para simular fallas el 5% de las fallas, por lo tanto, un subconjunto más pequeño con una cantidad razonable de fallas permanentes fue seleccionada arriba.

4 RESULTADOS EXPERIMENTALES

La intención principal de esta sección es reportar los datos recopilados resultados experimentales para punto fijo y punto flotante tipos de datos.

4.1 Representación de punto flotante

Se llevó a cabo un primer conjunto de experimentos aprovechando la configuración predeterminada de darknet, donde los pesos se representan en un formato de coma flotante de 32 bits. Todas las inyecciones han sido realizado considerando una carga de trabajo de 2023 imágenes. Tabla 4 representa el promedio de falla observada insegura (UOF) (la los más críticos) para cada capa objetivo de Lenet-5. Los resultados experimentales demuestran que convolucional las capas son menos confiables ante la presencia de fallas permanentes (Cuadro 4). De hecho, su porcentaje de error es aproximadamente 2x veces de las dos capas totalmente conectadas (L4-L6). Esto significa que los errores permanentes que afectan a las primeras capas de la Red Neuronal Convolucional impactan negativamente en todo el proceso de extracción de características. Como se afirma en [17], los errores que causar una gran desviación en los valores de activación son más probable que produzca una falla en la salida. De hecho, si las fallas son colocado dentro de una capa inicial, genera un gran número desviación, que la red ya no puede corregir. Seguramente el error se amplifique cada vez que fluya el valor a través de las sucesivas capas.

Una representación más precisa del porcentaje UOF la tendencia para todas las capas (L0-L2-L4-L6) se destaca en la Figura 6. En aras de la simplicidad, trazamos solo gráficos que representan el porcentaje de falla observada insegura (UOF) ya que parece para representar la métrica de clasificación más significativa. Además, un análisis más profundo de los resultados subraya que todos

Punto flotante 32 bits	[%] UOF
L0 convolucional	0.0137
Convolucional L2	0.0139
L4 Totalmente Conectado	0.0071
L6 Totalmente Conectado	0.0063

TABLA 4: [%] Promedio de falla observada insegura (UOF) para Experimentos de coma flotante.

las fallas observadas inseguras (UOF) se han debido a fallas afectando los 8 bits utilizados para almacenar el exponente (es decir, de bit 30 hasta el bit 23 de los datos de coma flotante). La señal y los bits de mantisa no tienen impactos significativos, es decir, conducían a fallas enmascaradas o seguras observadas.

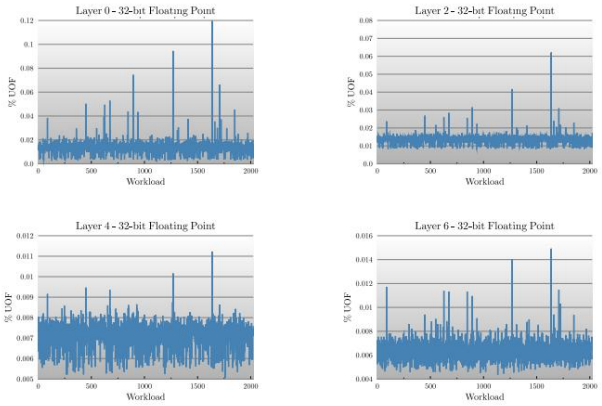


Fig. 6: [%] Porcentaje de fallas observadas inseguras (UOF) para una CNN de coma flotante.

Los resultados recopilados para los datos de punto flotante son bastante interesante ya que están mostrando una tendencia diferente con con respecto al efecto de los errores suaves. De hecho, los resultados de Los experimentos de error suave muestran que las capas convolucionales son se supone que es el más resistente a la presencia de una falla, de acuerdo con los resultados mostrados en [17]. Esto se debe al hecho de que su función es extraer las características de la imagen de origen, mientras que se supone que las capas conectadas completas son las menos resilientes porque clasifican las características extraídas por el primeros dos niveles. Por otra parte, estos resultados parecen confirmar la conclusión de [15] en la que los autores afirman esta tendencia.

4.2 Representación de punto fijo

Con respecto al conjunto de experimentos de punto fijo, toda la falla Las inyecciones se han realizado por separado en cada uno de los escenarios antes mencionados: caso ABCDE de nuestro caso estudiar. Se proporcionan resultados experimentales para cada uno de estos y, más notablemente, nos llevó a las siguientes consideraciones:

1) Las capas totalmente conectadas (L4 y L6) son, en principio, menos crítico y más resistente a los errores permanentes (Cuadro 5). Al comparar las capas convolucionales (L0 - L2) con los dos Totalmente Conectados (L4 - L6), emerge que, a primera vista, para las capas Totalmente Conectadas hay es una reducción de más del 50% del error medio, en

Punto fijo [%] UOF	A (32 bits) B (18 bits) C (16 bits) D (10 bits) E (6 bits)				
L0 convolucional	0.1617	0.0011	0.1126	0.0023	0.0070
L2 convolucional	0.1513	0.0006	0.0894	0.0011	0.0034
L4 Totalmente conectado	0.0545	0.00004	0.0152	0.00007	0.0044
L6 Totalmente conectado	0.0742	0.0011	0.0530	0.0022	0.0036

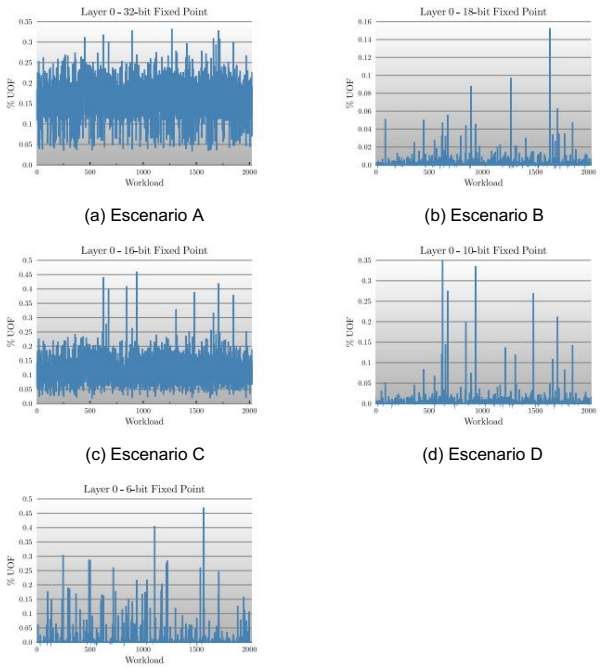
TABLA 5: [%] Promedio de fallas observadas inseguras (UOF) para Experimentos de Punto Fijo.

las cinco representaciones diferentes de punto fijo (excepto para B y D, debido a su peculiar estructura que se profundizará en los apartados siguientes). esto parece estar en línea con lo que se afirma para los resultados de coma flotante (Sección 4.1).

2) La primera capa convolucional L0 es la más crítica uno. Entre las cuatro capas, presenta la más alta porcentaje de falla observada insegura (UOF). Una falta afectando la primera capa convolucional de una CNN, más probablemente produce una salida incorrecta. En principio surge que la confiabilidad de la red se vea menos afectada ya que las fallas afectan a las últimas capas totalmente conectadas. En particular, La Figura 7 presenta una vista general del comportamiento de la red cuando fallas permanentes afectan L0. De hecho, por En aras de la concisión, sólo los datos pertenecientes a la primera capa convolucional (L0), para cada uno de los formato considerado de punto fijo (Figuras 7a-7b-7c-7d-7e). Lo que sale inmediatamente al observar los cinco gráficos es el comportamiento diferente de la red cuando frente a diferentes anchos de bits. es inmediatamente evidente que los peores escenarios son A y C, respectivamente de 32 bits y formatos de punto fijo de 16 bits, donde la cantidad de bits reservado para la parte entera es igual a los reservados por la parte fraccionaria del peso. El error medio de UOF es igual a 0.1617 y 0.1126 respectivamente para A y C, aproximadamente 100 veces mayor que los escenarios B, C y E (ver Tabla 5, segunda fila).

3) Los resultados experimentales prueban que hay un punto donde ya no es conveniente reducir el ancho de bits de los pesos. El último escenario E es la demostración de la afirmación anterior. De hecho, en ese caso particular, los pesos se representan a través de sólo 6 bits (1 para el signo, 1 para la parte entera y 4 para la fraccionaria). El promedio de UOF es de 1.6x veces hasta 110x veces mayor que B y D. Esto demuestra que por reduciendo el ancho de bits, la precisión y la fiabilidad de la red también podría reducirse.

En base a estas consideraciones, podría afirmarse que la La mejor opción de ancho de bit de punto fijo se encuentra en B (1 bit para el signo, 1 bit para entero, 16 bits para la parte fraccionaria) y D (1 bit para el signo, 1 bit para el entero, 8 bits para el fraccionario parte). Estos dos formatos tienen en común sólo un bit para la parte entera antes del punto de base. De hecho, el promedio el porcentaje de UOF es el más bajo entre los cinco escenarios (ver columna B y D de la Tabla 5). En particular, el más bajo porcentaje de UOF se obtiene al adoptar 18 bits para la representación del peso (B). Al reducir aún más el ancho de bits hasta 10 bits, el error aumenta ligeramente.



(a) Escenario A (b) Escenario B

(c) Escenario C (d) Escenario D

(e) Escenario E

Fig. 7: [%] Porcentaje de fallas observadas inseguras (UOF) para la Primera Capa Convolucional (L0) con una Representación de Ancho de Bit Decreciente.

4.3 Comparación de punto flotante y punto fijo

El propósito de este trabajo es brindar una visión general de la confiabilidad de una CNN bien conocida cuando un ancho de bits reducido se adopta para los pesos. Para cumplir con los nuevos requisitos de optimización de CNN, la confiabilidad de la red de destino es analizado cuando se utilizan dos tipos de datos diferentes: punto flotante y punto fijo. Se pueden sacar muchas conclusiones de comparando la Tabla 4 y la Tabla 5. De acuerdo con los resultados experimentales, una representación de punto flotante de 32 bits parece ser más confiable que uno de punto fijo de 32 bits (A, Tabla 5). Este último presenta un error promedio de 10 veces mayor que el formato de punto flotante. Todo esto podría ser justificado por la estructura innata de los dos tipos de datos: el La parte más sensible del punto flotante resulta ser la parte exponencial (8 bits). Por otro lado, el punto fijo de 32 bits contiene 15 bits para la parte entera. Si considerando que la distribución de pesos se agrupa alrededor del cero (Sección 2), 15 bits para la parte entera hace que la red sea realmente poco fiable y propenso a errores. Además, la representación de punto flotante de 32 bits sigue funcionando mejor que una representación de punto fijo de 16 bits (C, Tabla 5), donde se utilizan 7 bits para representar la parte entera. Por lo tanto, si el diseñador pretende reducir el ancho de bits del peso de la red, una buena idea es adoptar una representación de punto fijo donde sólo 1 bit es dedicado a los números enteros. Este resultado, como se describe en el La Sección 1.1, es consistente con la investigación publicada en [19]. En efecto, si observando el error medio de B y D (Tabla 5), es unas 10 veces menor que el punto flotante de 32 bits representación.

5 CONCLUSIONES

Este artículo presenta un marco de caracterización para analizar el impacto de las fallas permanentes que afectan a una red neuronal convo lucional destinada a ser implementada en el dominio automotriz. La caracterización se realiza mediante campañas de inyección de fallas en el marco de código abierto de darknet [23]. Todos los experimentos se realizan a nivel de software con el objetivo de ser independientes de las arquitecturas de hardware y, en conjunto, derivar una caracterización común del comportamiento de las CNN afectadas por fallas permanentes. Esto podría considerarse un resultado interesante ya que el diseñador, a partir de estos resultados, podría seleccionar el tipo de datos más conveniente para su aplicación.

Con respecto a la representación de coma flotante, los análisis demuestran que es importante cuidar principalmente los 8 bits de exponente (es decir, desde el bit 30 hasta el bit 23) que causan comportamientos críticos. Una técnica de redundancia puede funcionar bien para cubrir solo las partes críticas del sistema. Sin embargo, la descripción de las soluciones de prueba, así como las técnicas de redundancia utilizadas para endurecer la CNN, están fuera del alcance de este trabajo. Para concluir, el análisis propuesto tiene la ventaja de ser independiente del hardware: el diseñador tendrá la oportunidad de seleccionar cuidadosamente la arquitectura del hardware teniendo en cuenta los bits más críticos de la representación dada. En consecuencia, el ingeniero de pruebas podrá centrar los esfuerzos de prueba solo en las partes críticas para reducir el costo de la solución de prueba.

EXPRESIONES DE GRATITUD

Este trabajo se ha fundado parcialmente en el contexto del proyecto ODeLe IDEXLYON (ANR-16-IDEX-0005).

REFERENCIAS

[1] Y. LeCun, Y. Bengio y G. Hinton, "Aprendizaje profundo", Naturaleza, vol. 521, pp. 436 EP 05 2015. [En línea]. Disponible: http://dx.doi.org/10.1038/nature14539 [2] S. Albawi, TA Mohammed y S. Al-Zawi, "Understanding of a convolutional neural network", en la Conferencia Internacional de Ingeniería y Tecnología de 2017 (ICET), 2017, págs. 1–6.

[3] Avances recientes en aprendizaje profundo para la investigación del habla en Microsoft. Conferencia internacional IEEE sobre acústica, voz y procesamiento de señales (ICASSP), mayo de 2013. [En línea]. Disponible: https://www.microsoft.com/en-us/research/publication/recent advances-in-deep-learning-for-speech-research-at-microsoft/ [4] A. Krizhevsky, I. Sutskever, y GE Hinton, "Clasificación de Imagenet con redes neuronales convolucionales profundas", en Actas de la 25.ª Conferencia internacional sobre sistemas de procesamiento de información neuronal - Volumen 1, ser. NIPS'12. EE. UU.: Curran Associates Inc., 2012, págs. 1097–1105. [En línea]. Disponible: http://dl.acm.org/citation.cfm?id=2999134.2999257 [5] D. Silver, A. Huang, CJ Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser , I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel y D. Hassabis, "Dominar el juego de go con redes neuronales profundas y búsqueda de árboles", Nature, vol. 529, págs. 484 EP 01 2016.

–,

[En línea]. Disponible: http://dx.doi.org/10.1038/nature16961 [6] W. Sung, "Resiliencia de redes neuronales profundas bajo cuantificación". CoRR, vol. abs/1511.06488, 2015.

[7] C. Chen, A. Seff, A. Kornhauser y J. Xiao, "Conducción profunda: posibilidad de aprendizaje para la percepción directa en la conducción autónoma", en Actas de la Conferencia internacional IEEE 2015 sobre visión por computadora (ICCV), ser. ICCV'15. Washington, DC, EE. UU.: IEEE Computer Society, 2015, págs. 2722–2730. [En línea]. Disponible: http://dx.doi.org/10.1109/ICCV.2015.312

[8] R. Cantoro, A. Firrincieli, D. Piumatti, M. Restifo, E. Sanchez y MS Reorda, "Acerca de la identificación de fallas funcionalmente no comprobables en línea en núcleos de microprocesadores para aplicaciones críticas para la seguridad" 2018 IEEE 19th Latin-American Test Symposium (LATS), págs. 1–6, 2018.

[9] M. Psarakis, D. Gizopoulos, E. Sanchez y MS Reorda, "Autoevaluación basada en software de microprocesador", IEEE Design and Test of Computers, vol. 27, págs. 4 a 19, 2010.

[10] K. Simonyan y A. Zisserman, "Redes convolucionales muy profundas para el reconocimiento de imágenes a gran escala", 2014.

[11] S. Mittal, "Un estudio de técnicas para la computación aproximada", Cómputo ACM. Surv., vol. 48, núm. 4, págs. 62:1–62:33, marzo de 2016. [En línea]. Disponible: http://doi.acm.org/10.1145/2893356 [12] M. Courbariaux, Y. Bengio y J.-P. David, "Binaryconnect: Entrenamiento de redes neuronales profundas con pesos binarios durante la propagación", 2015.

[13] C. Zhu, S. Han, H. Mao y WJ Dally, "Cuantificación ternaria entrenada", 2016.

[14] M. Rastegari, V. Ordonez, J. Redmon y A. Farhadi, "Xnor net: Clasificación de Imagenet usando redes neuronales convolucionales binarias", 2016.

[15] F. dos Santos, L. Draghetti, L. Weigel, L. Carro, P. Navaux y P. Rech, "Evaluación y mitigación de errores de software en la detección de objetos basada en redes neuronales en tres arquitecturas gpu" págs. 169–176, 6 de junio de 2017.

[16] A. Lotfi, S. Hukerikar, K. Balasubramanian, P. Racunas, N. Saxena, R. Bramley e Y. Huang, "Resiliencia de las redes de detección de objetos automotrices en arquitecturas gpu", en la Prueba internacional IEEE de 2019 Conferencia (ITC), 2019, págs. 1–9.

[17] G. Li, SKS Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer y SW Keckler, "Comprender la propagación de errores en aceleradores y aplicaciones de redes neuronales de aprendizaje profundo (dnn)", en Procedimientos de la Conferencia Internacional de Informática, Redes, Almacenamiento y Análisis de Alto Rendimiento, ser. SC'17. Nueva York, NY, EE. UU.: ACM, 2017, págs. 8:1–8:12. [En línea]. Disponible: http://doi.acm.org/10.1145/3126908.3126964 [18] Tiny-cnn. [En línea]. Disponible: https://github.com/nyanp/tiny

CNN

[19] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, SK Lee, N. Mulholland, D. Brooks y G. Wei, "Ares: un marco para cuantificar la resiliencia de las redes neuronales profundas" en 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), 2018, pp. 1–6.

[20] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza y L. Benini, "Un motor de navegación visual basado en dnn de 64 mw para nanodrones autónomos", IEEE Internet of Things Journal, pag. 1–1, 2019. [En línea]. Disponible: http://dx.doi.org/10.1109/JIOT.2019.2917066

[21] B. Salami, O. Unsal y A. Cristal, "Sobre la resiliencia de rtl nn aceleradores: caracterización y mitigación de fallas", 2018.

[22] A. Bosio, P. Bernardi, A. Ruospo y E. Sánchez, "Un análisis de confiabilidad de una red neuronal profunda", en Simposio de prueba latinoamericano (LATS) de IEEE de 2019, 2019, págs. 1 a 6.

[23] J. Redmon, "Darknet: redes neuronales de código abierto en c", http://pjreddie.com/darknet/, 2013–2016.

[24] Y. Lecun, L. Bottou, Y. Bengio y P. Haffner, "Aprendizaje basado en gradientes aplicado al reconocimiento de documentos", Actas del IEEE, vol. 86, núm. 11, págs. 2278–2324, noviembre de 1998.

[25] [En línea]. Disponible: https://github.com/ashitani/darknet mnist [26] A. Chatterjee y LR Varshney, "Hacia la cuantificación óptima de las redes neuronales", en Simposio internacional sobre teoría de la información (ISIT) de IEEE de 2017, 2017, págs. 1162 –1166.

[27] R. Ding, Z. Liu, RDS Blanton y D. Marculescu, "Redes neuronales profundas cuantificadas para la inferencia basada en hardware de eficiencia energética", en la 23.ª Conferencia de Automatización del Diseño de Asia y el Pacífico Sur (ASP-DAC) de 2018, 2018, págs. 1 a 8.

[28] [En línea], "Biblioteca Libfixmath", https://github.com/Petteri Aimonen/libfixmath, 2020.

[29] M. Kooli, F. Kaddachi, GD Natale y A. Bosio, "Evaluación de la confiabilidad del sistema consciente de caché y registro basada en el análisis de la vida útil de los datos", en 2016 IEEE 34th VLSI Test Symposium (VTS), abril de 2016, págs. 1 a 6.

[30] R. Leveugle, A. Calvez, P. Maistri y P. Vanhauwaert, "Inyección estadística de fallas: error cuantificado y confianza", en 2009 Design, Automation Test in Europe Conference Exhibition, abril de 2009, págs. 502–506.