Microprocesadores y Microsistemas 86 (2021) 104318

Listas de contenidos disponibles en ScienceDirect

Microprocesadores y Microsistemas

página de inicio de la revista: www.elsevier.com/locate/micpro

Investigando la representación de datos para Convolucional eficiente y confiable Redes neuronales

Annachiara Ruospo [un], Ernesto Sánchez, Marcello Traiolab, Ian O´Connor [b], alberto bosio [b]

[a] DAUIN - Politécnico de Turín, Italia
[b] Universidad de Lyon, ECL, INSA Lyon, CNRS, UCBL, CPE Lyon, INL, UMR5270, Francia

INFORMACIÓN DEL ARTÍCULO

MSC:
00-01
99-00

Palabras clave:
Cómputo aproximado
Redes neuronales convolucionales
Fiabilidad
Inyecciones de fallas

ABSTRACTO

Hoy en día, las Redes Neuronales Convolucionales (CNNs) son ampliamente utilizadas como modelos de predicción en diferentes campos, con un uso intensivo en sistemas críticos para la seguridad en tiempo real. Estudios recientes han demostrado que las fallas de hardware inducidas por una perturbación externa o efectos de envejecimiento pueden afectar significativamente la inferencia de CNN, lo que lleva a fallas en la predicción. Por lo tanto, garantizar la confiabilidad de las plataformas CNN es crucial, especialmente cuando se implementan en aplicaciones críticas. Se ha hecho un gran esfuerzo para reducir la huella de memoria y energía de las CNN, allanando el camino para la adopción de técnicas de computación aproximadas como la cuantificación, la precisión reducida, el peso compartido y la poda. Desafortunadamente, la computación aproximada reduce la redundancia intrínseca de las CNN, haciéndolas más eficientes pero menos resistentes a las fallas de hardware. El objetivo de este trabajo es doble. Primero, evaluamos la confiabilidad de una CNN cuando se utilizan anchos de bits reducidos y dos tipos de datos diferentes (punto flotante y punto fijo) para representar los parámetros de la red (es decir, pesos sinápticos). En segundo lugar, tenemos la intención de investigar el mejor compromiso entre el tipo de datos, la reducción del ancho de bits y la confiabilidad. La caracterización se realiza a través de un entorno de inyección de fallas basado en el marco de código abierto de darknet y se dirige a dos CNN: LeNet-5 y YOLO. Los resultados experimentales muestran que los datos de punto fijo proporcionan la mejor compensación entre la reducción del espacio de memoria y la resiliencia de CNN. En particular, para LeNet-5, logramos una reducción de 4 veces el espacio de memoria a costa de una confiabilidad ligeramente reducida (0,45 % de fallas críticas) sin volver a entrenar la

1. Introducción

En las últimas décadas, las Redes Neuronales Convolucionales (CNNs) han ganado popularidad debido a su excelente desempeño en la resolución de problemas complejos de aprendizaje [1]. De hecho, brindan muy buenos resultados para muchas tareas, como el reconocimiento de objetos en imágenes/videos, el descubrimiento de fármacos, el procesamiento del lenguaje natural y los juegos [2–4]. Se clasifican como una clase de redes neuronales profundas (DNN) [5] y su nombre se origina en la operación lineal matemática entre matrices llamada convolución.

Al ser modelos inspirados en el cerebro, las redes neuronales convolucionales tienen una resiliencia inherente atribuida a su estructura distribuida y paralela y la redundancia de espacio introducida debido al sobreaprovisionamiento [6].
Varios estudios han investigado la tolerancia a fallos de las DNN, por ejemplo [7], y han demostrado que las NN son muy robustas frente a los errores de cálculo [8,9]. Como consecuencia, los NN específicos de hardware, que van desde circuitos para aplicaciones de aprendizaje automático integradas hasta integración a gran escala personalizada (VLSI) de redes neuronales en silicio, se consideran tradicionalmente muy resistentes a fallas de hardware (fallas de HW).

Sin embargo, varios estudios recientes han demostrado que las fallas de HW inducidas por una perturbación externa (es decir, en un entorno hostil) o debido al desgaste del silicio y los efectos del envejecimiento pueden afectar significativamente la inferencia que conduce a fallas en la predicción de CNN [10–12] . Por lo tanto, garantizar la confiabilidad de las CNN es crucial, especialmente cuando se implementan en aplicaciones críticas para la seguridad y de misión crítica, como robótica, aeronáutica, atención médica inteligente y conducción autónoma [13] .
La confiabilidad se puede definir como la probabilidad de que una falla de HW provoque una falla [14]. El análisis de confiabilidad y su evaluación están regulados por estándares según el dominio de aplicación (por ejemplo, IEC 61508 para sistemas industriales, DO-254 para aviónica, ISO 26262 para automoción), y generalmente se evalúa a través de campañas de inyección de fallas (FI).

Hoy en día, para abordar los problemas de confidencialidad de los datos y las limitaciones de ancho de banda, la tendencia es llevar los sistemas basados en el aprendizaje profundo de la nube a los dispositivos de borde [15–17], como los dispositivos de Internet de las cosas (IoT), dado el creciente número de usuarios de Internet. -IoT conectado. Una de las principales ventajas es que alivia la latencia de comunicación que es inaceptable para decisiones críticas de seguridad en tiempo real, por ejemplo, en

Autor correspondiente.
Direcciones de correo electrónico: annachiara.ruospo@polito.it (A. Ruospo), ernesto.sanchez@polito.it (E. Sánchez), marcello.traiola@ec-lyon.fr (M. Traiola), ian.connor@ec-lyon.fr (I. O'Connor), alberto.bosio@ec-lyon.fr (A. Bosío).

https://doi.org/10.1016/j.micpro.2021.104318 Recibido el
29 de diciembre de 2020; Recibido en forma revisada el 1 de junio de 2021; Aceptado el 10 de julio de 2021
Disponible en línea el 11 de agosto de
2021 0141-9331/

conducción autónoma. Por esta razón, el diseño de aceleradores de hardware personalizados de circuitos integrados específicos de aplicaciones (ASIC) para admitir el movimiento de datos que consumen mucha energía, la velocidad de cómputo y los recursos de memoria que las CNN requieren para alcanzar su máximo potencial en el perímetro es crucial [18] .

Paralelamente a las evaluaciones de confiabilidad [19], se ha realizado un esfuerzo significativo para reducir la huella de memoria y energía de las CNN aprovechando el 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 reciente de las CNN es la memoria necesaria para almacenar los parámetros de la red (p. ej., pesos sinápticos). Por ejemplo, VGG Net [20] requiere 500 MB de memoria, una complejidad que simplemente no se ajusta al hardware restringido de muchos sistemas integrados. Recientemente, 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 integrados y de alto rendimiento [21]. Al relajar la necesidad de operaciones completamente precisas o completamente deterministas, AxC mejora sustancialmente la eficiencia energética y reduce el requisito de memoria. Varias técnicas para AxC aumentan el espacio de diseño al proporcionar otro conjunto de perillas de diseño para las compensaciones de rendimiento y precisión. A modo de 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 considerando el almacenamiento, el 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.

Por lo tanto, 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 AxC, especialmente con implementaciones de aritmética de punto fijo o de punto flotante de baja precisión, que exponen un gran paralelismo de grano fino. Por ejemplo, en [22] 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 [23]; cuantifica los pesos en 3 valores diferentes {−1, 0, 1}. Finalmente, XNOR-Net [24] utiliza una metodología ligeramente diferente: todos los cálculos se realizan a través de XNOR y operaciones de conteo de bits, al mismo tiempo que se reduce la precisión de los operandos involucrados durante el cálculo. En consecuencia, es necesario comprender si esos modelos optimizados son lo suficientemente confiables para tolerar fallas que se propagan por todo el sistema. Comienza a ser crucial 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 artículo es caracterizar el impacto de las fallas permanentes que afectan a una CNN mediante campañas de inyección de fallas, cuando se utilizan tipos de datos personalizados para representar los parámetros de la red.
Analizamos diferentes implementaciones de la misma arquitectura de CNN, cuando se explotan diferentes tipos de datos, e identificamos la mejor representación que lleva a lograr una reducción de 4 veces la huella de memoria con la mayor resistencia a fallas. Se presentan dos estudios de caso: el primero apunta a LeNet-5, una popular CNN, el último se enfoca en una CNN más profunda (YOLO) utilizada para la tarea de detección de objetos en tiempo real.

El resto del documento está estructurado de la siguiente manera: después de resumir los trabajos relacionados en el campo, se destaca nuestra principal contribución (Sección 1.1). La Sección 2 presenta la metodología, centrándose en la descripción del tipo de datos personalizado y la técnica de conversión de datos. En la misma sección, se describen tanto el escenario como el entorno de inyección de fallas. A continuación, en la Sección 3, presentamos los dos casos de estudio (es decir, LeNet-5 y YOLO), junto con los resultados experimentales provenientes de las campañas de FI. Finalmente, la Sección 4 concluye el artículo y describe algunas de las posibles direcciones futuras de investigación.

1.1. Obras relacionadas

En la literatura, se presta cada vez más atención a la confiabilidad de la red neuronal. Dependiendo de múltiples factores, como la tipología de FI, el nivel de abstracción y 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 [25,26], los autores realizaron un análisis profundo de la confiabilidad de CNN cuando se utilizan valores de punto flotante de 32 bits como tipo de datos para la representación de pesos, resultó que el bit 30 es el más crítico entre los 32 en general. Curiosamente, encontramos los mismos resultados con nuestra metodología como se muestra en las siguientes secciones. En [27,28], 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 [29], donde los autores evalúan la confiabilidad de un DNN de 54 capas (NVIDIA DriveWorks) a través de experimentos de inyección de fallas y pruebas de haz de neutrones acelerados para fallas permanentes y transitorias, respectivamente. Las fallas se inyectan en los pesos de las DNN y en las imágenes de entrada. Todas las inferencias se ejecutan en Volta GPU solo con valores de coma flotante de 32 bits.

En el futuro, Li et al. presente en [30] 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 DNN basado en un marco de simulación de código abierto, Tiny-CNN [31]. Gracias a la flexibilidad del simulador, es posible caracterizar cada capa para un análisis más preciso. En este artículo, se consideran las CNN con seis tipos de datos diferentes: punto flotante de precisión doble de 64 bits, punto flotante de precisión simple de 32 bits, punto flotante de precisión media de 16 bits, punto fijo de 32 bits (con dos diferentes puntos de base) y punto fijo de 16 bits. En general, concluyen que, cuanto mayor sea el rango de valores dinámicos del tipo de datos de la red, mayor será la probabilidad de tener grandes desviaciones en los valores en caso de fallas que conduzcan a predicciones incorrectas.

Además, se muestra un marco diferente en [32]: Ares, un marco ligero de inyección de fallas de DNN. Los autores presentan un estudio empírico sobre la resiliencia de tres tipos destacados de DNN (totalmente conectadas, 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 optimizado Q2,6 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 ocurra una falla. Vale la pena señalar que este resultado está en línea con nuestros resultados recopilados, presentados en la Sección 3. Es una tendencia común explorar cálculos de punto fijo para sistemas integrados de potencia ultrabaja con un presupuesto de potencia limitado [33]. Finalmente, en [34], los autores analizan la confiabilidad de un acelerador DNN 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 el ciclo 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 utilizada, realizan los experimentos adoptando solo 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 total.

Además, también vale la pena mencionar la contribución de investigación de [35] y [36], donde los autores investigan la confiabilidad de las CNN que explotan la representación de datos enteros de 16 y 8 bits. Luego, moviéndose hacia una dimensión de datos aún más pequeña, los trabajos relacionados en [37] y [38] explotan anchos de bits reducidos. El primero utiliza tipos de datos de punto fijo de 5 y 3 bits y una representación binaria. Este último realiza análisis de evaluación de confiabilidad en una red neuronal binaria, donde solo se usa 1 bit para representar los parámetros (pesos y sesgos).

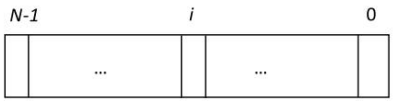


Fig. 1. Tipo de datos personalizado.

En este sentido, la principal contribución de este artículo es un análisis exhaustivo del comportamiento de las CNN en función de su representación de datos. En un trabajo anterior [11,39], evaluamos el impacto de las fallas permanentes que afectan a las CNN a través de campañas de software FI. En comparación con los análisis del estado del arte [11,32,34,39], se proporciona un espectro más amplio de representaciones de punto fijo y flotante (cinco tipologías que van desde 32 bits hasta 8 bits). Como resultado, identificamos la mejor representación de datos que condujo a la mayor reducción del uso de memoria y la mayor resistencia a fallas.

2. Metodología

Esta sección presenta la metodología adoptada. Explotamos el marco DNN de código abierto de red oscura [40]. Implementado en lenguaje C y CUDA, es adecuado para realizar despliegues de arquitecturas de redes neuronales de extremo a extremo de una manera muy sencilla. Además, proporciona un entorno muy simple donde se pueden ejecutar varias configuraciones de DNN, incluidas las CNN, ya sea para realizar trabajos de entrenamiento o de inferencia. En nuestro trabajo, modificamos el marco de la red oscura para (i) aproximar el DNN e (ii) inyectar fallas en el momento de la inferencia.

2.1. Aproximación del tipo de datos DNN

En el campo de las redes neuronales, un enfoque de aproximación común es reducir la precisión y el tipo de datos de los pesos y valores de activación.
Más en detalle, tenemos la intención de utilizar representaciones personalizadas de punto flotante y punto fijo con diferente precisión (es decir, ancho de bits) en el momento de la inferencia.

El marco darknet se aprovecha solo de los tipos de datos de punto flotante de 32 bits. Por lo tanto, modificamos el código fuente de la darknet para permitir conversiones de tipos de datos. Todas las conversiones entre el punto flotante estándar de 32 bits y el tipo de datos personalizado se han llevado a cabo mediante la integración de dos bibliotecas de código abierto en el marco de darknet : la biblioteca libfixmath [41] para administrar el punto fijo y la biblioteca FloatX .

La figura 1 ilustra nuestro tipo de datos personalizado. Se define como sigue:

- : determina el ancho de bits de los datos;
- : determina la dinámica y la precisión del tipo de dato dependiendo de la representación de los datos:
  - Coma flotante: es el ancho de la mantisa, el ancho del − 1 − es el exponente;
  - Punto fijo: es el ancho fraccionario, − 1 − es el número entero ancho.

Dado que el objetivo final es caracterizar la propagación del efecto de falla a través de la red (acelerar los cálculos y compactar el tamaño del modelo están fuera del alcance de este trabajo), realizamos conversiones en línea mientras mantenemos todas las operaciones internas en punto flotante (Fig . . 2). Los beneficios que se derivan 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 DNN 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 diferentes formatos numéricos en un tiempo razonable. Para confirmar la idoneidad de la elección, realizamos un experimento preliminar para confirmar que el uso de la conversión en línea no introduce diferencias de precisión importantes al tiempo que brinda beneficios en el tiempo de ejecución. Con este fin, en los experimentos, convertimos el representación de datos del marco darknet desde punto flotante de 32 bits hasta punto fijo de 32 bits. De esta forma, todas las operaciones se realizaron en el dominio aritmético de punto fijo utilizando la biblioteca libfixmath [41].
Ejecutamos la inferencia de 70.000 imágenes de la base de datos MNIST con la CNN LeNet-5 (más detalles en la Sección 3.1) con ambas versiones, es decir, la de punto flotante original y la de punto fijo modificada.
Específicamente, no capacitamos a la CNN. Los resultados de los experimentos mostraron que el error de precisión promedio de la versión de punto fijo con respecto a la de punto flotante fue de −0.01%, es decir, un ligero aumento de precisión. Además, mientras que el tiempo de ejecución del experimento de la versión de punto fijo fue de 8566 s, es decir, ≈0,122 segundos por imagen, el tiempo de ejecución de la versión original de punto flotante fue de 3280 s, es decir, ≈ 0,047 segundos por imagen. Por lo tanto, el uso del marco de coma flotante de 32 bits de darknet con conversión en línea nos permite ejecutar experimentos 2,6 veces más rápidos con respecto a la conversión del tipo de datos de darknet, sin incurrir en diferencias significativas en la precisión de la inferencia.

En aras de la exhaustividad, describimos cómo se aplica la conversión en línea de un peso de punto flotante de 32 bits antes de llegar a una sola neurona (Fig. 2). El esquema aplicado funciona de la siguiente manera:

1. El peso se convierte de punto flotante estándar de 32 bits a una representación de tipo de datos personalizada dada.
2. El peso del tipo de datos personalizado se corrompe de acuerdo con una lista de fallas elegida y una ubicación de falla, es decir, se inyecta la falla.
3. El peso del tipo de datos personalizado se vuelve a convertir a la representación estándar de punto flotante de 32 bits 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.

Si bien todas las operaciones de red se ejecutan entre variables de coma flotante de 32 bits, debe destacarse que se conserva la pérdida de precisión provocada por la primera conversión. De hecho, al pasar de la representación estándar de punto flotante de 32 bits a una de baja precisión (por ejemplo, punto fijo de 16 bits), estamos presenciando un efecto de error de truncamiento.
Luego, al volver a convertir de un rango estrecho de valor a uno más amplio, el error de truncamiento aún permanece.

2.2. Inyector de falla

La intención de la sección es describir la inyección de fallas (FI) escenario y el entorno construido en el marco darknet .

Escenario FI: La Fig. 3 ilustra el escenario en el que se ejecuta la campaña de inyección de fallas. En primer lugar, trabajamos con la CNN entrenada con tipos de datos de punto flotante de 32 bits, llamados Estándar. La red entrenada se aproxima mediante una representación de tipo de datos personalizada, denominada Personalizada. Las salidas de inferencia se almacenan (es decir, el "Estándar dorado" y el "Personalizado dorado") y se comparan para determinar la Pérdida de precisión debida a la aproximación. La campaña de inyección de fallas se lleva a cabo en la CNN personalizada y las salidas de inferencia defectuosas se almacenan en el registro "Personalizado defectuoso". Este último se compara luego con el "Golden Custom" y el "Golden Std" para evaluar la resiliencia.

La CNN personalizada (es decir, aproximada) está destinada a reemplazar la CNN estándar en dispositivos de borde/recursos limitados. Por lo tanto, tenemos que evaluar la confiabilidad de la CNN personalizada con respecto a la CNN estándar. Por otro lado, también es posible entrenar directamente el tipo de datos personalizado CNN. En este caso, la referencia con respecto a la confiabilidad que se debe evaluar es la propia CNN personalizada.

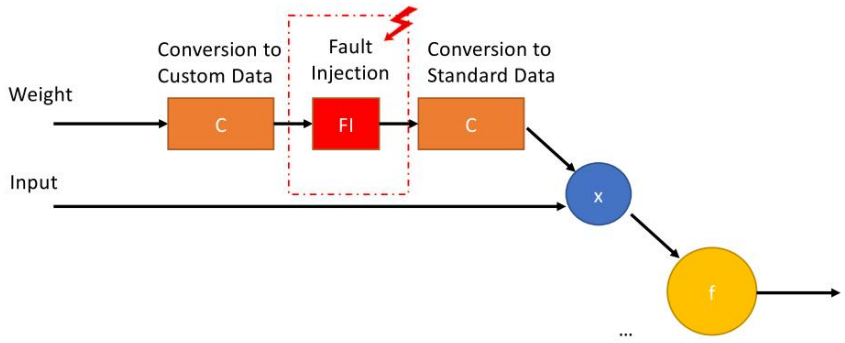


Fig. 2. Conversiones de pesos en línea.

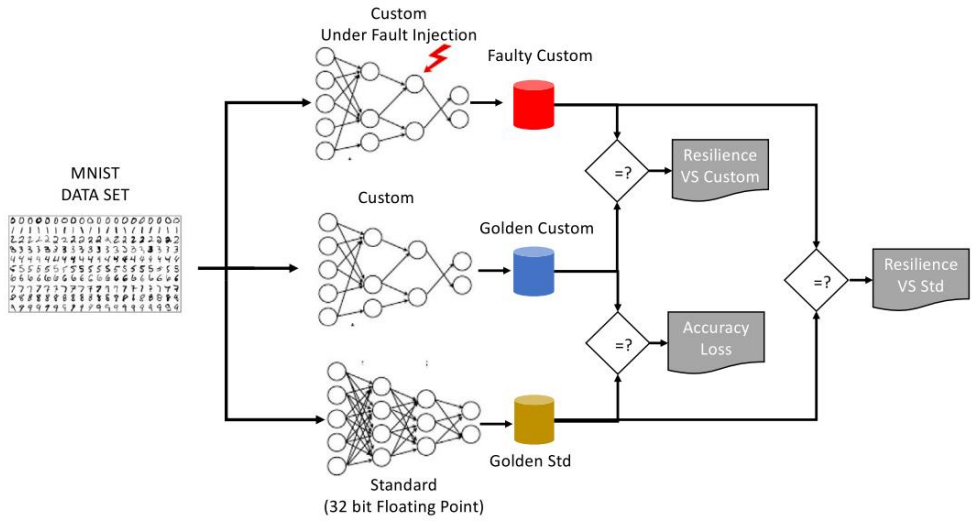


Fig. 3. Escenario de inyección de fallas.

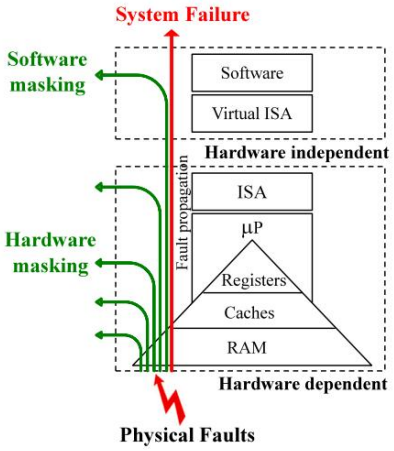


Fig. 4. Capas del sistema y propagación de fallas.

Entorno FI: El sistema de hardware puede verse afectado por fallas causadas por defectos físicos de fabricación. Como destaca la Fig. 4 , las fallas podrían propagarse a través de las diferentes estructuras de hardware que componen el sistema completo. Sin embargo, podría suceder que estén enmascarados durante la propagación ya sea a nivel tecnológico o arquitectónico nivel [42]. Cuando una falla llega a la capa de software del sistema, puede corromper datos, instrucciones o el flujo de control. Estos errores pueden impactar la correcta ejecución del software al producir resultados erróneos o evitar que la ejecución de la aplicación provoque una terminación anómala o que la aplicación se cuelgue. La pila de software puede desempeñar un papel importante papel en el enmascaramiento de errores; al mismo tiempo, este fenómeno está implícitamente importante para la confiabilidad del sistema pero un desafío difícil para los ingenieros que tienen que garantizar la seguridad de sus sistemas.

Como se indicó en la Introducción, las fallas de HW pueden ser transitorias o permanentes inducidas por perturbaciones externas (es decir, en un entorno hostil) o debido al desgaste del silicio y los efectos del envejecimiento. Es importante hacer hincapié una vez más, que realizamos una inyección de fallas en los pesos sinápticos que son valores constantes (es decir, nunca reescritos en la memoria). Significa que, incluso para fallas transitorias, una vez que se dispara la falla, se comporta exactamente como uno permanente ya que la celda de memoria invertida no será reescrito Por esta razón, como modelo de falla objetivo, consideramos el modelo Stuck at Fault ( ) en 0/1 (0 y 1). La ubicación de la falla ( ) está definida por (1).

$$= \quad , \quad , \quad , \qquad (1)$$

donde corresponde a la capa CNN, es el borde conectando un nodo del y es uno de los bits del peso asociado con el puede ser '0' . Finalmente, el o '1' dependiendo del SaF. El Fault Injector realmente funciona en el capa de software, y su pseudocódigo se proporciona en el (Listado 1). Corresponde a un simple inyector serial averiado que modifica la topología CNN como se describe en (1).

El proceso de FI sigue el escenario representado en la Fig. 3 y consiste en lo siguiente: una vez que la CNN está completamente entrenada, se realiza una carrera dorada recolectando los resultados dorados (es decir, predicción dorada estándar y personalizada), es decir, las líneas 1 y 2 en el Listado 1. Luego, la inyección de falla real se realiza el proceso. El paso inicial requiere generar la lista de

```
1 run_CNN (CNN, estándar, golden_prediction_std); ,
ejecutar_CNN (CNN, golden_prediction_cs t ) ; 2
personalizado 3 para ( i =0; i < FLo . tamaño ( ) ;
i ++) { 4 inyectar_CNN (CNN, falla_CNN); 5 run_CNN (CNN,
predicción _ defectuosa personalizada); 6 comparar
(falla_prediccion, golden_prediction_c st); 7 comparar (falla_predicción,
oro_predicción_std); 8 comunicado CNN);
_ falla (Flo [ i ] ,
9 }
```

Listado 1: Pseudocódigo de inyección de fallas

fallas a inyectar. Esta lista de fallas debe verse como una lista de lugares para inyectar las fallas, como se describió anteriormente. Luego, para cualquier falla en la lista de fallas (línea 3), se realiza una ejecución de predicción y los resultados se recopilan y nombran como predicción_defectuosa. Es necesario subrayar que las fallas se inyectan independientemente de su polaridad (atascadas en 0 o atrapadas 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 forma, no distinguimos entre el efecto singular de los dos modelos de falla y obtenemos una gran flexibilidad para la considerable cantidad de simulaciones realizadas. En este punto (líneas 6 y 7), los resultados obtenidos se comparan con los esperados (nuevamente para la CNN estándar y personalizada), y los resultados se registran para un análisis posterior.

En detalle, la función comparar del Pseudo-código (Listado 1) clasifica la predicción/clasificación de la CNN defectuosa con respecto a la dorada. La clasificación depende del tipo de CNN. En nuestro artículo, consideramos dos tipos de CNN: (i) un clasificador y (ii) un detector de objetos. Con respecto al clasificador, las salidas de la CNN defectuosa se etiquetan de la siguiente manera:

- Enmascarado: No se observa diferencia entre la CNN defectuosa y la dorada.
- Observado: Se observa una diferencia entre la CNN defectuosa y la dorada. Dependiendo de cuánto diverjan los resultados, los clasificamos además como:
  - Bueno: La puntuación de confianza de los primeros clasificados es superior respecto a la CNN dorada. En otras palabras, la CNN defectuosa proporciona una mejor inferencia que la dorada;
  - Aceptar: La puntuación de confianza del elemento mejor clasificado se reduce en menos del 5% con respecto a la CNN dorada;
  - Advertencia: La puntuación de confianza del elemento mejor clasificado se reduce en más de un 5% con respecto a la CNN dorada;
  - Crítico: La predicción top-1 es diferente. En otras palabras, la CNN defectuosa hace una inferencia incorrecta.

Desde una perspectiva de evaluación de la seguridad, consideramos tres clases de fallas Críticas, Advertencia y Aceptar como eventos que reducen la seguridad de la CNN. De hecho, cada vez que ocurre una de esas clases de fallas, la predicción principal 1 es diferente (Crítico) o el nivel de confianza de la predicción principal 1 disminuye (Advertencia, Aceptar). Por otro lado, las dos clases de fallo Enmascarado y Bueno dejan inalterada la seguridad de la CNN (Enmascarado) o incluso la mejoran (Bueno).

Por otro lado, los efectos de las fallas inyectadas en el detector de objetos CNN se clasifican de manera diferente. La salida de la CNN es una imagen que tiene cuadros delimitadores que indican la detección de los objetos. Para evaluar si dos cuadros delimitadores se superponen, utilizamos la métrica de intersección sobre unión (IoU) como se define en [12]. IoU es la relación del área de intersección sobre el área de unión de dos cuadros delimitadores. Cuanto más cerca esté IoU de 1, mayor será la superposición de los dos cuadros delimitadores. Gracias a la métrica IoU, podemos redefinir el resultado de la falla de la siguiente manera:

- Enmascarado: No se observa diferencia entre la CNN defectuosa y la dorada.
- Observado: Se observa una diferencia entre la CNN defectuosa y la dorada. Al usar el IoU calculado entre las casillas del CNN dorado y las del defectuoso, clasificamos los resultados observados de la siguiente manera:
  - Aceptar: El IoU es inferior a 1 y superior a 0,95;
  - Advertencia: el IoU es inferior a 0,95 y superior a 0,9;
  - Crítico: el número de cuadros delimitadores es diferente o la etiqueta asociada con los cuadros no coincide con el Buenos. En otras palabras, la CNN defectuosa identificó objetos incorrectos. Además, si el IoU es inferior a 0,9, la falla se clasifica como crítica, lo que significa que la CNN defectuosa identifica correctamente los objetos, pero no puede ubicarlos con la precisión suficiente.

No consideramos el resultado "Bueno" ya que no tiene sentido para la detección de objetos. Vale la pena señalar que la clasificación de fallas utilizada para el detector de objetos es más estricta que la utilizada para el clasificador. De hecho, consideramos que la tarea de detección de objetos es más crítica que la tarea de clasificación. Desde una perspectiva de evaluación de la seguridad, consideramos las fallas que caen en las clases Crítica, Advertencia y Aceptar como eventos que reducen la seguridad de la CNN. Por el contrario, las fallas enmascaradas dejan inalterada la seguridad de la CNN.

3 resultados experimentales

Esta sección primero detalla los dos casos de estudio y la falla relacionada. campañas de inyección, luego se discuten los resultados experimentales.

3.1 Primer estudio de caso

Entre las CNN existentes, nuestro interés recae en LeNet-5 [43], un clasificador bien conocido 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 ejecutar los experimentos, se seleccionó la base de datos MNIST [44], 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 el costo y el tiempo computacional, se seleccionó aleatoriamente una carga de trabajo de 2023 imágenes del conjunto de datos de prueba/validación del MNIST para 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 una representación de punto flotante de 32 bits.

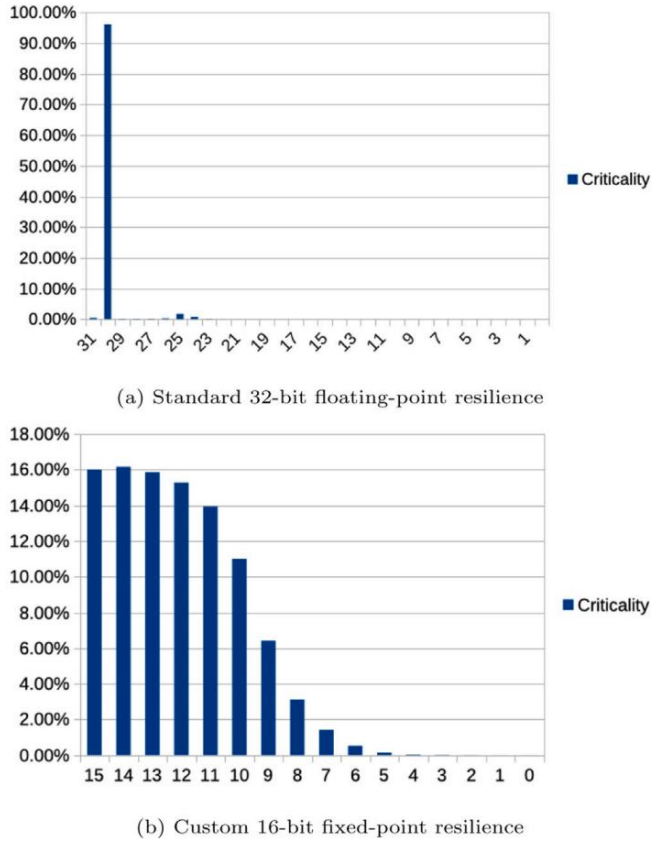
3.2 Tipo de datos personalizado

La selección del tipo de datos personalizado no es trivial. Recurramos a un ejemplo para ilustrar este punto. En nuestro trabajo preliminar [11], analizamos diferentes tipos de datos de punto fijo en términos de resistencia a fallas. Uno de ellos se configuró con = 16 y = 8, lo que significa que se dedicaron 8 bits para representar la parte fraccionaria y 8 bits para la parte entera. higos. 5a y 5b muestran la "criticidad" de cada bit del tipo de datos. Se puede ver que en el tipo de datos de coma flotante de 32 bits, solo un bit (es decir, el bit 30) es el principal responsable de los comportamientos defectuosos observados críticos: hasta el 95% de los comportamientos defectuosos observados críticos se debe a un falla en el bit 30. Por otro lado, en la representación de punto fijo de 16 bits (con 8 bits para los enteros y 8 bits para la parte fraccionaria), el número de bits responsables de los comportamientos defectuosos Observados Críticos es seis. Esto significa que la CNN personalizada tiene una huella de memoria menor del 50%, pero también muestra una menor resiliencia con respecto a la CNN estándar, ya que una mayor cantidad de bits defectuosos puede afectar seriamente los resultados de la inferencia.

tabla 1

Pérdida de precisión del tipo de datos LeNet-5 [%].

Guión	Tipo de datos	ancho de bits	Codificación de bits	[%] Pérdida de precisión
FP32	Punto flotante	32	1 signo, 8 exponente, 23 fraccionario 1 signo,	Árbitro.
FP16	Punto flotante	[illegible]	5 exponente, 10 fraccionario 1 signo, 4	0%
FP8	Punto flotante	8	exponente, 3 fraccionario 1 entero, 31	0,02%
FxP32	Punto fijo	32	fraccionario 1 entero, 15	0%
FxP16	Punto fijo	[illegible]	fraccionario 1 entero, 7	0%
FxP8	Punto fijo	8	fraccionario	0,04%



(a) Estándar 32-bit floating-point resilience

(b) Custom 16-bit fixed-point resilience

Fig. 5. Bits críticos.

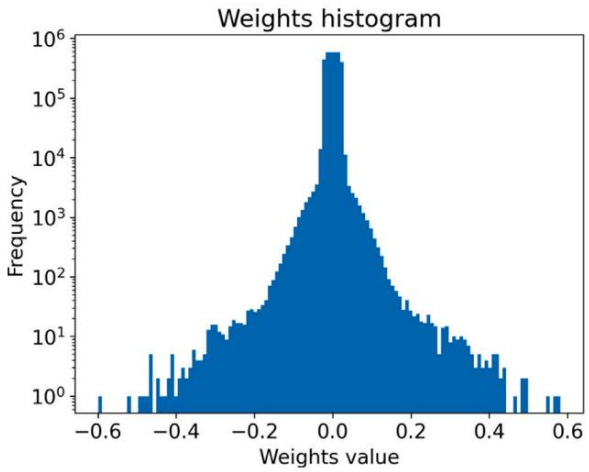


Fig. 6. Distribución de valores de pesos preentrenados para la red LeNet-5 .

Para seleccionar cuidadosamente la representación de datos personalizados, primero analizamos la distribución de peso de LeNet-5. Se ilustra en la Fig. 6 y evidencia que todos los valores están en el rango −0.6 a 0.6 con la mayoría de ellos alrededor de cero. De este análisis, simplemente deducimos que el tipo de datos no necesita una dinámica más alta, mientras que se prefiere una alta precisión. Por lo tanto, seleccionamos los tipos de datos personalizados informados en la Tabla 1. Dos datos Se utilizan tipos, el fijo y el flotante con diferente ancho de bit. Además, calculamos la pérdida de precisión de la CNN resultante de la adopción de pesos de tipos de datos personalizados. Como se destaca en la Tabla 1, Se han analizado cinco escenarios diferentes. La segunda columna de la tabla informa el tipo de datos utilizados en cada campaña de FI, mientras que la la tercera columna informa el ancho de bits de los pesos. la cuarta columna muestra la cantidad de bits asignados para codificar las diferentes partes de el número, es decir, signo, exponente y partes fraccionarias en el caso de representaciones de punto flotante, y la parte entera y fraccionaria en el caso de los datos de punto fijo. Para calcular la precisión de la CNN en los diferentes escenarios, la inferencia de las imágenes pertenecientes al conjunto de validación de la base de datos MNIST (10.000 imágenes) se ha ejecutado en LeNet-5, claramente sin inyectar fallas, es decir, en un escenario dorado. Los resultados mostraron que solo al reducir el ancho de bit a 8 bits, el la red exhibió cierta pérdida de precisión. En detalle, para la red con pesos codificados mediante el uso de variables de coma flotante de 8 bits (escenario FP8), la pérdida de precisión fue del 0,02%, mientras que fue del 0,04% cuando los pesos se codificaron utilizando variables de punto fijo de 8 bits (escenario FxP8).

3.3 Lista de fallas

En esta sección, discutimos la complejidad de la CNN considerada en términos de inyección de fallas. Además, ilustramos el enfoque que utilizamos para gestionar esta complejidad mediante la realización de una estadística subconjunto significativo de inyecciones de fallas. Para esta evaluación compleja, considere solo cuatro capas LeNet-5 que realizan cálculos aritméticos involucrando pesos entrenables, es decir, dos convolucionales y dos completamente Conectado. De hecho, consideramos la resiliencia de la CNN frente a fallas golpeando la memoria, donde se almacenan los pesos. La Tabla 2 proporciona detalles sobre la configuración, así como la lista de fallas de cada capa. Las dos primeras filas (etiquetadas "Capa" y "Detalle") de la tabla presentan las capas de destino; el tercero ("Conexiones") especifica la cantidad de sus pesos de conexión. El número de fallas posibles se calcula como la multiplicación entre el número de conexiones ("Conexiones") y el tamaño del peso ("Bit-width"). Como señalan las filas "#Faults" , el número total de posibles fallas es muy alto y esto se refleja en un tiempo de ejecución de la campaña FI no manejable. Por lo tanto, para reducir la falla tiempo de ejecución de la inyección, podemos seleccionar aleatoriamente un subconjunto de fallas. A obtener resultados estadísticamente significativos con un margen de error del 1% y un nivel de confianza del 99%, un promedio de 15.6k inyecciones de fallas han a considerar para escenarios de 32 bits (FP32 y FxP32), 15k para escenarios de 16 bits escenarios (FP16 y FxP16), y 13.8k para escenarios de 8 bits (FP8 y FxP8). Los números precisos se dan en las filas de la Tabla 2 etiquetadas #Inyecciones" y se han calculado utilizando el método presentado en [45]. En detalle, recurrimos a la siguiente fórmula:

$$= \frac{}{1 + \frac{2}{2} \frac{-1}{0.25}} \quad (2)$$

donde es el número total de ubicaciones de fallas (es decir, fila #Faults of Tabla 2), es el margen de error deseado (1%), y depende de la

Tabla 2
Lista de fallas de LeNet-5 para campañas de inyección de fallas.

	Capa	L0	L2	L4	L6
	Detalle	convolucional	convolucional	Totalmente	Totalmente conectado
	Conexiones	2400	51,200	Conectado 3.211.264	10,240
Escenarios FP32, FxP32	ancho de bits	32	32	32	32
	#Fallas	76.800	1.638.400	102.760.448	327,680
	#inyecciones	13.678	16.474	16.638	15,837
Escenarios FP16, FxP16	ancho de bits	16	16	16	16
	#Fallas	38.400	819.200	51.380.224	163,840
	#inyecciones	11.610	16.310	16.636	15,107
Escenarios FP8, FxP8	ancho de bits	8	8	8	8
	#Fallas	19,200	409.600	25.690.112	81,920
	#inyecciones	8915	15.991	16.630	13,831

nivel de confianza deseado (t = 2.58 corresponde al 99% de confianza nivel [45]). ecuación (2) tiene un valor asintótico horizontal (→ ∞) igual a 16.641, lo que limita el número de inyecciones de falla necesarias para lograr una evaluación con un margen de error del 1% y una confianza nivel del 99%. Además, vale la pena subrayar que las inyecciones son se realiza seleccionando aleatoriamente el bit defectuoso entre todos los bits del pesos de conexión.

Para realizar los experimentos de FI, las conversiones de peso se realizan como se describe en la Sección 2.1, es decir, los pesos se asignan a el tipo personalizado y luego reconvertido al formato original. En particular, cada vez que se coloca una falla, el peso seleccionado aleatoriamente el valor debe convertirse a la representación personalizada, inyectado con la falla, y reconvertido al formato original para realizar el inferencia. Por un lado, este enfoque requiere una sobrecarga durante la fase de inyección de fallas; sin embargo, por otro lado, esta técnica permite realizar los experimentos sin introducir gastos generales debido a puntos fijos y flotantes personalizados computacionalmente intensivos librerías para las operaciones aritméticas.

3.4 Resultados de inyección de fallas

Realizamos dos conjuntos de experimentos (ver Fig. 3). En el primero, evaluamos la confiabilidad usando como referencia el Estándar 32-bit CNN de coma flotante. Esto es útil en el caso de que un diseñador quiere aproximar la CNN (es decir, cambiar su tipo de datos y/o ancho de bits) después de haber sido entrenado. En el segundo, evaluamos la confiabilidad de la CNN utilizando como referencia la CNN sin fallas personalizada. Esto es útil en el caso de que un diseñador quiera entrenar directamente al CNN de tipo de datos/ancho de bits personalizado. Para discutir los resultados, nos referimos a la clasificación presentada en la Sección 2.2. En particular, queremos evaluar la seguridad de las diferentes versiones de CNN, cuando están sujetas a fallas. Por lo tanto, consideramos fallas en las clases Critical, Warning y Accept como eventos que reducen la seguridad de CNN. La suma de estas contribuciones es representado por el símbolo '<' en las tablas. Por el contrario, consideramos las fallas en las clases Enmascarado y Bueno como eventos ya sea dejando la seguridad de la CNN inalterada o incluso mejorándola. La suma de estas contribuciones está representado por el símbolo '≥' en las tablas.

Los resultados del primer conjunto de experimentos (es decir, tener el Estándar CNN de punto flotante de 32 bits como referencia) se muestran en la Tabla 3 donde cada fila corresponde a una de las variantes de CNN (FP32-FP16-FP8-FxP32-FxP16-FxP8 definidas en la Tabla 1). Cada columna corresponde a un clase de comportamiento defectuoso como se describe en la Sección 2.2. En primer lugar, podemos señalar una resiliencia diferente a las fallas según el tipo de datos: flotante versus fijo. Más en detalle, el efecto de disminución de la seguridad es menor para el punto fijo que para los puntos flotantes, para un ancho de bits dado. Como ejemplo podemos recurrir a los escenarios FP32 y FxP32 (32 bits CNN): el efecto de aumento (decremento) de la seguridad varía del 69% (31%) de la versión en coma flotante (escenario FP32) al 74% (26%) de la versión en coma fija (escenario FxP8). Esto corresponde a una diferencia del 5%. La diferencia promedio entre las versiones de coma flotante y de punto fijo con con respecto al efecto de aumento/disminución de la seguridad en las tres variantes (32, 16 y 8 bits) es 8,96% sobre todas las capas. Esto puede ser visto comparando los escenarios FP32 con FxP32, FP16 con FxP16, y FP8 con FxP8, en términos del efecto promedio de aumento/disminución de la seguridad variación (columnas 8 y 9). ¿Vale la pena resaltar que, en general términos, el efecto de disminución de la seguridad es crítico sólo en unos pocos casos. El el porcentaje de fallos críticos es siempre inferior al 3,42% para todas las variantes. En particular, las variantes de punto fijo tienen un porcentaje muy pequeño de fallos críticos, siempre inferior al 0,46%. Además, la contribución de Las buenas fallas en el efecto de aumento de la seguridad resultan significativas, especialmente para las versiones de 16 y 8 bits. Como ejemplo, en el escenario FxP8 para la capa L0, observamos un efecto de aumento de la seguridad en un 52,21% de los casos, con un 52,18% de Buenas faltas.

Además, el ancho de bits juega un papel importante para la fiabilidad: cuanto menor sea el ancho de bits, menor será la resiliencia. Por lo tanto, un diseñador que quería usar una versión más eficiente de la CNN (huella de memoria reducida) debe tener en cuenta que también sería menos resistente con respecto a la CNN original (FP32). Sin embargo, vale la pena remarcando también que el uso de la representación de datos de punto fijo, en lugar de la contraparte de punto flotante, proporciona los mejores resultados en términos de equilibrio entre resiliencia y eficiencia. Esto se informa en el últimas dos columnas de la Tabla 3. Por ejemplo, podemos comparar escenarios FP8 y FxP8 (CNN de 8 bits) para la capa L0: observamos una pérdida de seguridad respecto al FP32 del 37% en la versión de punto flotante (escenario FP8) y solo del 16% en la versión de punto fijo (escenario FxP8). Por lo tanto, elegir la CNN en el escenario FxP8, es decir, punto fijo de 8 bits (1 bit para entero y 7 bits para fraccionario), permite al diseñador para compactar la huella de memoria por un factor de 4x mientras se reduce el seguridad sólo en un 16%. Además, mirando más de cerca, la ocurrencia de fallas críticas en el escenario FxP8 incluso disminuye de 1.32% de FP32 al 0,45%, mientras que en el escenario B aumenta al 3,41%. Además, para escenario FxP32 (CNN de punto fijo de 32 bits), se ha observado que el CNN logra mejorar la seguridad con respecto al escenario FP32 para el misma huella de memoria para las capas L0 y L2 (+5,34 % y +1,43 %, respectivamente). Por lo tanto, simplemente cambiando el tipo de datos CNN a un punto fijo la representación puede mejorar su resiliencia para algunas capas.

La Tabla 4 informa los resultados del segundo conjunto de experimentos (es decir, teniendo la CNN personalizada como referencia). Mientras que en el primer set comparamos los resultados de FI de cada escenario a los obtenidos con el sistema libre de fallas CNN de punto flotante de 32 bits (FP32), en este conjunto de experimentos comparamos los resultados de cada escenario a los resultados obtenidos con los correspondientes CNN personalizado sin fallas. Este escenario corresponde a la formación directa la CNN de tipo de datos personalizado, por lo que la confiabilidad debe evaluarse con respecto a la propia CNN personalizada. Para obtener más información, consulte la Fig. 3. Tenga en cuenta que el la fila relacionada con la versión FP32 es la misma en ambos experimentos conjuntos En general, las tendencias destacadas en el primer conjunto de experimentos se observan también en este escenario: (i) la seguridad se ve afectada por la reducción del ancho de bit; (ii) para las versiones de CNN de punto fijo, observamos una mayor Disminución elegante de la seguridad en comparación con las versiones de coma flotante. Además eso, un efecto interesante es que para variantes personalizadas de punto flotante (FP16 y FP8) la diferencia entre los dos conjuntos de experimentos (Tablas 3 y 4) en cuanto a la ganancia de Seguridad con respecto al FP32 es mayor que para los de punto fijo (FxP32, FxP16 y FxP8). por ejemplo, en capa L0, para la variante FP8 (coma flotante de 8 bits) la ganancia de seguridad con

Tabla 3
Resultados de inyección de fallas de LeNet-5 con Golden Std.

Capa	Datos	Observado				Enmascarado	Seguridad		Ganancia contra FP32	
		Crítico	Advertencia	Aceptar	Bien		<	≥	Seguridada	Memoria
L0	FP32	1,32%	0,06%	29,96%	28,98%	39,68%	31,34%	68,66%	–	–
	FP16	2,61%	0,12%	53,28%	41,27%	2,71%	56,01%	43,98%	−24,67%	2X
	FP8	3,41%	0,91%	64,13%	31,53%	0,02%	68,45%	31,55%	−37,11%	4X
	FxP32	0,03%	0,04%	25,93%	25,54%	48,47%	26,00%	74,01%	+5,34%	0
	FxP16	0,05%	0,08%	49,98%	46,94%	2,96%	50,11%	49,90%	−18,77%	2X
	FxP8	0,45%	0,60%	46,74%	52,18%	0,03%	47,79%	52,21%	−16,45%	4X
L2	FP32	1,37%	0,02%	23,88%	23,39%	51,34%	25,27%	74,73%	–	–
	FP16	2,59%	0,08%	55,00%	38,57%	3,76%	57,67%	42,33%	−32,40%	2X
	FP8	1,10%	0,91%	65,86%	32,10%	0,03%	67,87%	32,13%	−42,60%	4X
	FxP32	0.01%	0.01%	23,82%	23,62%	52,54%	23,84%	76,16%	+1,43%	0
	FxP16	0,03%	0,02%	49,87%	46,61%	3,47%	49,92%	50,08%	−24,65%	2X
	FxP8	0,42%	0,45%	46,57%	52,53%	0,03%	47,44%	52,56%	−22,17%	4X
L4	FP32	0.71%	0,00%	3,85%	3,86%	91,57%	4,56%	95,44%	–	–
	FP16	0,84%	0,13%	57,79%	36,12%	5,13%	58,75%	41,25%	−54,19%	2X
	FP8	0,49%	0,06%	66,69%	32,72%	0,04%	67,24%	32,76%	−62,67%	4X
	FxP32	0,00%	0,00%	10,99%	11,49%	77,52%	10,99%	89,01%	−6,43%	0
	FxP16	0,00%	0,00%	49,40%	45,59%	5,01%	49,40%	50,60%	−44,84%	2X
	FxP8	0,40%	0,36%	46,38%	52,82%	0,04%	47,14%	52,86%	−42,58%	4X
L6	FP32	0,61%	0.01%	7,69%	8,14%	83,54%	8,32%	91,68%	–	–
	FP16	1,19%	0,03%	56,34%	37,65%	4,78%	57,57%	42,43%	−49,25%	2X
	FP8	1,76%	0,40%	65,07%	32,74%	0,04%	67,22%	32,78%	−58,91%	4X
	FxP32	0.01%	0,04%	13,50%	14,11%	72,33%	13,55%	86,45%	−5,24%	0
	FxP16	0,03%	0,08%	49,15%	46,11%	4,63%	49,26%	50,74%	−40,94%	2X
	FxP8	0,44%	0,53%	46,28%	52,72%	0,04%	47,25%	52,75%	−38,93%	4X

aDiferencia del efecto de aumento de la seguridad entre un escenario dado y el FP32.

Tabla 4
Resultados de la inyección de fallas de LeNet-5 con Golden Custom.

Capa	Datos	Observado				Enmascarado	Seguridad		Ganancia contra FP32	
		Crítico	Advertencia	Aceptar	Bien		<	≥	Seguridada	Memoria
L0	FP32	1,32%	0,06%	29,96%	28,98%	39,68%	31,34%	68,66%		–
	FP16	2,61%	0,12%	42,10%	40,86%	14,30%	44,84%	55,16%	−13,50%	2X
	FP8	3,30%	0,88%	47,08%	44,67%	4,07%	51,26%	48,74%	−19,91%	4X
	FxP32	0,03%	0,04%	24,45%	23,81%	51,67%	24,51%	75,49%	+6,83%	0
	FxP16	0,05%	0,08%	41,96%	40,81%	17,10%	42,09%	57,91%	−10,75%	2X
	FxP8	0.11%	0,15%	49,03%	46,94%	3,76%	49,29%	50,71%	−17,95%	4X
L2	FP32	1,37%	0,02%	23,88%	23,39%	51,34%	25,27%	74,73%		–
	FP16	2,59%	0,08%	35,66%	34,59%	27,07%	38,34%	61,66%	−13,07%	2X
	FP8	0,96%	0,89%	46,21%	43,17%	8,77%	48,06%	51,94%	−22,79%	4X
	FxP32	0.01%	0.01%	21,41%	20,93%	57,64%	21,43%	78,57%	+3,84%	0
	FxP16	0,03%	0,02%	38,70%	37,74%	23,52%	38,75%	61,25%	−13,47%	2X
	FxP8	0,06%	0,05%	47,94%	45,79%	6,17%	48,05%	51,95%	−22,77%	4X
L4	FP32	0.71%	0,00%	3,85%	3,86%	91,57%	4,56%	95,44%		–
	FP16	0,84%	0,13%	6,21%	6,23%	86,59%	7,17%	92,83%	−2,61%	2X
	FP8	0,32%	0,06%	10,45%	10,11%	79,06%	10,83%	89,17%	−6,26%	4X
	FxP32	0,00%	0,00%	4,06%	4,02%	91,92%	4,06%	95,94%	+0,50%	0
	FxP16	0,00%	0,00%	7,83%	7,75%	84,42%	7,83%	92,17%	−3,27%	2X
	FxP8	0.01%	0,00%	10,67%	10,33%	78,99%	10,68%	89,32%	−6,12%	4X
L6	FP32	0,61%	0.01%	7,69%	8,14%	83,54%	8,32%	91,68%		–
	FP16	1,19%	0,03%	11,47%	12,63%	74,67%	12,70%	87,30%	−4,39%	2X
	FP8	1,59%	0,39%	15,87%	17,10%	65,05%	17,85%	82,15%	−9,53%	4X
	FxP32	0.01%	0,04%	7,15%	7,27%	85,52%	7.21%	92,79%	+1.11%	0
	FxP16	0,03%	0,08%	13,48%	13,72%	72,69%	13,59%	86,41%	−5,27%	2X
	FxP8	0,05%	0,16%	17,25%	18,27%	64,27%	17,47%	82,53%	−9,15%	4X

*Diferencia de efecto de aumento de seguridad entre un escenario dado y FP32.

con respecto a FP32 es −37.1% para el primer conjunto de experimentos (Tabla 3) y −19,9% para el segundo conjunto de experimentos (Tabla 4), es decir un 17,2% diferencia. Por el contrario, para la variante FxP8 (punto fijo de 8 bits) el Safety la ganancia con respecto a FP32 es −16,4% para el primer conjunto de experimentos (Tabla 3) y -17,9% para el segundo conjunto de experimentos (Tabla 4), que es una diferencia del 1,5%. En promedio, la diferencia entre el dos conjuntos de experimentos para variantes de punto flotante (FP16 y FP8) sobre todas las capas es 33.72%, mientras que para variantes personalizadas de punto fijo (FxP32, FxP16 y FxP8) es 14,81%. En la práctica, esto significa que un diseñador que optó por aproximarse a la versión original de CNN estándar FP32 utilizando una variante de punto flotante personalizada sin volver a entrenarla, estaría expuesta a una mayor degradación de la seguridad que mediante el uso de la alternativa de punto fijo con el mismo ancho de bits. Cuando un entrenamiento se realiza directamente en el CNN de tipo de datos personalizado, la diferencia de degradación de seguridad entre el variantes de punto flotante y las de punto fijo es menor. Sin embargo, los de punto fijo aún garantizan una ocurrencia de falla menos crítica, es decir, menos del 0,12%.

Finalmente, como resultado del análisis, pensamos que en general, la CNN en el escenario FxP8 proporciona los mejores resultados en términos de consumo de memoria

Tabla 5

Pérdida de precisión del tipo de datos YOLO [%].

Guión	Tipo de datos	ancho de bits	Codificación de bits	[%] Pérdida de precisión
FP32	Punto flotante	32	1 signo, 8 exponente, 23 fraccionario 1	Árbitro.
FP16	Punto flotante	decidir	signo, 5 exponente, 10 fraccionario 1 signo,	42% enmascarado, 58% aceptable
FP8	Punto flotante		4 exponente, 3 fraccionario 3 entero, 29	28% advertencia, 72% crítico
FxP32	Punto fijo	8 32	fraccionario 3 entero, 13	100% enmascarado
FxP16	Punto fijo	decidir	fraccionario 3 entero, 5	42% enmascarado, 58% aceptable
FxP8	Punto fijo	8	fraccionario	100% crítico

Enmascarado: IoU = 1; aceptable: 0,95 < IoU < 1; advertencia: 0,9 ≤ IoU ≤ 0,95; crítico: IoU < 0,9 o diferentes objetos reconocidos.

Tabla 6

Lista de fallas de YOLO para campañas de inyección de fallas.

Capa	Conexiones	FP32 y FxP32 Ancho de bit = 32		FP16 y FxP16 Ancho de bit = 16		FP8 y FxP8 Ancho de bit = 8	
		#Fallas	#inyecciones	#Fallas	#inyecciones	#Fallas	#inyecciones
L0	432	13.824	7.551	6.912	4.884	3.456	2,862
L2	4.608	147.456	14.954	73.728	13.577	36.864	11,466
L4	18.432	589.824	16.184	294.912	15.752	147.456	14,954
L6	73.728	2.359.296	16.524	1.179.648	16.410	589.824	16,184
L8	294.912	9.437.184	16.612	4.718.592	16.583	2.359.296	16,524
L10	1.179.648	37.748.736	16.634	18.874.368	16.626	9.437.184	16,612
L12	4.718.592	150.994.944	16.639	75.497.472	16.637	37.748.736	16,634
L13	262.144	8.388.608	16.608	4.194.304	16.575	2.097.152	16,510
L14	1.179.648	37.748.736	16.634	18.874.368	16.626	9.437.184	16,612
L15	130.560	4.177.920	16.575	2.088.960	16.509	1.044.480	16,380
L18	32.768	1.048.576	16.381	524.288	16.129	262.144	15,648
L21	884.736	28.311.552	16.631	14.155.776	16.621	7.077.888	16,602
L22	65.280	2,0 88,960	16.509	1.044,48 0	16.380	522.240	16,127

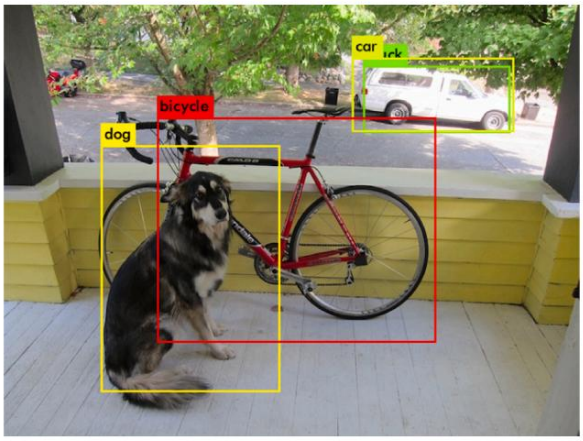


Fig. 7. Resultado de la predicción de YOLO CNN.

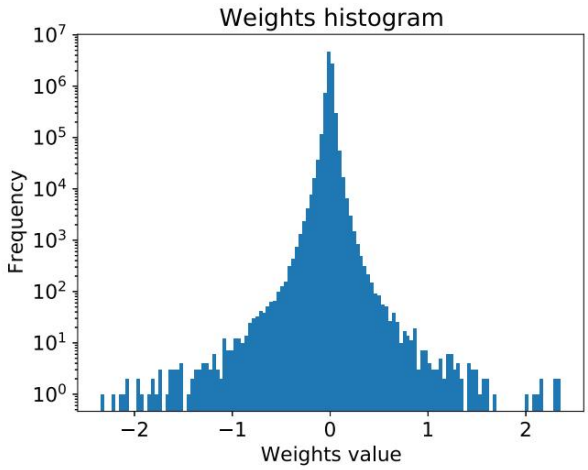


Fig. 8. Distribución de valores de pesos preentrenados para la red YOLO .

reducción, es decir, 4X, con una resiliencia significativa, es decir, menos del 0,45% fallas críticas.

3.5 Segundo estudio de caso

El segundo caso de estudio se dirige a YOLO [46], una CNN capaz de detectar objetos en tiempo real, analizando hasta 45 fotogramas por segundo. yolo es un predictor CNN utilizado para detectar un determinado conjunto de objetos (es decir, el lista de objetos reconocibles provino de [40]). En la Fig. 7, la predicción Los resultados se muestran resaltando los objetos identificados, incluido el nombre del objeto y ocupación del área. Este ejemplo presenta una imagen que contiene tres objetos relevantes para ser detectados: un perro, una bicicleta y un coche. Además, el automóvil se reconoce además como un camión por un total de cuatro objetos detectados.

3.6 Tipo de datos personalizado

Para seleccionar cuidadosamente la representación de datos personalizados, primero analizamos la distribución del peso de YOLO. Se muestra en la Fig. 8. Como se evidencia, todos los valores están en el rango −2.35 a 2.36 con la mayoría de ellos alrededor de cero. Por lo tanto, también para esta CNN, el tipo de datos no necesita una dinámica más alta. mientras que se prefiere una alta precisión. Por lo tanto, seleccionamos la costumbre tipo de datos informado en la Tabla 5.

Además, calculamos la pérdida de precisión de la CNN resultante de la adopción de tipos de datos personalizados. Como se destaca en la Tabla 5, cinco Se han analizado diferentes escenarios. La segunda columna de la tabla. informa el tipo de datos utilizado en cada escenario, mientras que la tercera columna informa el ancho de bits de los pesos. La cuarta columna muestra la cantidad de bits asignados para codificar las diferentes partes del número, es decir, signo, exponente y partes fraccionarias en el caso de punto flotante representaciones, y parte entera y fraccionaria en el caso de las de punto fijo. Para calcular la pérdida de precisión de la red en los diferentes escenarios, la inferencia de 7 imágenes se ha ejecutado en YOLO, claramente sin inyectar faltas, es decir, en un escenario dorado. los resultados son reportados de acuerdo con la clasificación reportada en la Sección 2.2 para el detector de objetos Los resultados muestran que para el escenario FxP32 no hay

Tabla 7

Resultados de inyección de fallas de YOLO con Golden Std.

Nivel	Datos	Observado			Enmascarado	Seguridad		Ganancia contra FP32	
		Crítico	Advertencia	Aceptar		<	=	Seguridada	Memoria
L0	FP32	20,17%	0,66%	16,25%	62,92%	37,08%	62,92%	–	–
	FP16	30,70%	1,04%	41,32%	26,95%	73,05%	26,95%	−35,97%	2x
	FP8	48,46%	49,88%	1,66%	0,00%	100,00%	0,00%	−62,92%	4x
	FxP32	13,35%	0,54%	12,43%	73,68%	26,32%	73,68%	10,76%	0
	FxP16	27,23%	1,11%	44,52%	27,14%	72,86%	27,14%	−35,78%	2x
L2	FP32	6,81%	0,10%	14,46%	78,63%	21,37%	78,63%	–	–
	FP16	12,07%	0,17%	49,85%	37,91%	62,09%	37,91%	−40,72%	2x
	FP8	16,67%	78,65%	4,68%	0,00%	100,00%	0,00%	−78,63%	4x
	FxP32	7,40%	0,43%	12,22%	79,95%	20,05%	79,95%	1,33%	0
	FxP16	14,38%	0,84%	51,95%	32,83%	67,17%	32,83%	−45,80%	2x
L4	FP32	5,05%	0,05%	11,24%	83,66%	16,34%	83,66%	–	–
	FP16	9,88%	0,10%	49,32%	40,70%	59,30%	40,70%	−42,97%	2x
	FP8	13,30%	82,95%	3,75%	0,00%	100,00%	0,00%	−83,66%	4x
	FxP32	6,32%	0,35%	11,86%	81,47%	18,53%	81,47%	−2,19%	0
	FxP16	12,45%	0,70%	52,81%	34,04%	65,96%	34,04%	−49,62%	2x
L6	FP32	4,02%	0.01%	7,59%	88,37%	11,63%	88,37%	–	–
	FP16	8,29%	0,04%	49,55%	42,13%	57,87%	42,13%	−46,25%	2x
	FP8	9,48%	88,49%	2,02%	0,00%	100,00%	0,00%	−88,37%	4x
	FxP32	4,49%	0,26%	11,04%	84,21%	15,79%	84,21%	−4,17%	0
	FxP16	8,82%	0,52%	53,75%	36,91%	63,09%	36,91%	−51,47%	2x
L8	FP32	3,41%	0.01%	3,94%	92,65%	7,35%	92,65%	–	–
	FP16	7,15%	0,02%	50,53%	42,30%	57,70%	42,30%	−50,35%	2x
	FP8	7,06%	91,51%	1,44%	0,00%	100,00%	0,00%	−92,65%	4x
	FxP32	2,84%	0,19%	9,22%	87,75%	12,25%	87,75%	−4,90%	0
	FxP16	5,49%	0,38%	54,32%	39,81%	60,19%	39,81%	−52,84%	2x
L10	FP32	3,26%	0,00%	1,70%	95,04%	4,96%	95,04%	–	–
	FP16	6,42%	0,02%	52,08%	41,49%	58,51%	41,49%	−53,55%	2x
	FP8	4,93%	94,16%	0,91%	0,00%	100,00%	0,00%	−95,04%	4x
	FxP32	1,67%	0.11%	6,85%	91,37%	8,63%	91,37%	−3,67%	0
	FxP16	3,14%	0,21%	54,33%	42,32%	57,68%	42,32%	−52,72%	2x
L12	FP32	3,17%	0,00%	0,84%	95,98%	4,02%	95,98%	–	–
	FP16	6,18%	0,07%	52,75%	41,00%	59,00%	41,00%	−54,98%	2x
	FP8	3,69%	95,54%	0,77%	0,00%	100,00%	0,00%	−95,98%	4x
	FxP32	0,76%	0,06%	5,34%	93,85%	6,15%	93,85%	−2,13%	0
	FxP16	1,39%	0,08%	54,87%	43,66%	56,34%	43,66%	−52,32%	2x
L13	FP32	3,27%	0.01%	3,37%	93,36%	6,64%	93,36%	–	–
	FP16	6,76%	0,04%	51,46%	41,74%	58,26%	41,74%	−51,62 %	2x
	FP8	6,09%	92,80%	1,10%	0,00%	100,00%	0,00%	−93,36 %	4x
	FxP32	1,86%	0,18%	9,08%	88,87%	11,13%	88,87%	−4,49%	0
	FxP16	3,61%	0,37%	55,13%	40,88%	59,12%	40,88%	−52,47%	2x
L14	FP32	3,13%	0.01%	1,25%	95,61%	4,39%	95,61%	–	–
	FP16	5,83%	0,07%	52,96%	41,13%	58,87%	41,13%	−54,47%	2x
	FP8	3,92%	96,08%	0,00%	0,00%	100,00%	0,00%	−95,61%	4x
	FxP32	1,03%	0,13%	5,96%	92,89%	7,11%	92,89%	−2,72%	0
	FxP16	2,05%	0,30%	54,55%	43,10%	56,90%	43,10%	−52,50%	2x
L15	FP32	0,82%	0.01%	0,19%	98,97%	1,03%	98,97%	–	–
	FP16	1,75%	0,04%	65,39%	32,82%	67,18%	32,82%	−66,15%	2x
	FP8	1,05%	98,95%	0,00%	0,00%	100,00%	0,00%	−98,97%	4x
	FxP32	0,34%	0,06%	0,36%	99,24%	0,76%	99,24%	0,26%	0
	FxP16	0,57%	0.11%	56,69%	42,63%	57,37%	42,63%	−56,35%	2x
L18	FP32	3,26%	0,02%	0,19%	96,53%	3,47%	96,53%	–	–
	FP16	6,10%	0,04%	53,82%	40,04%	59,96%	40,04%	−56,49%	2x
	FP8	5,55%	92,84%	1,61%	0,00%	100,00%	0,00%	−96,53%	4x
	FxP32	0,98%	0,14%	0,89%	98,00%	2,00%	98,00%	1,48%	0
	FxP16	2,04%	0,26%	57,61%	40,10%	59,90%	40,10%	−56,43%	2x
L21	FP32	3,22%	0,00%	0,06%	96,72%	3,28%	96,72%	–	–
	FP16	5,61%	0,02%	53,96%	40,41%	59,59%	40,41%	−56,31%	2x
	FP8	3,83%	94,51%	1,67%	0,00%	100,00%	0,00%	−96,72%	4x
	FxP32	0,37%	0,06%	0,56%	99,02%	0.98%	99,02%	2,29%	0
	FxP16	0,63%	0,17%	57,73%	41,47%	58,53%	41,47%	−55,25%	2x
L22	FP32	0,44%	0,00%	0,03%	99,52%	0,48%	99,52%	–	–
	FP16	0,89%	0,00%	56,66%	42,45%	57,55%	42,45%	−57,07%	2x
	FP8	1,14%	98,83%	0,03%	0,00%	100,00%	0,00%	−99,52%	4x
	FxP32	0,14%	0,02%	0,04%	99.79%	0,21%	99.79%	0,27%	0
	FxP16	0,29%	0,02%	57,05%	42,63%	57,37%	42,63%	−56,89%	2x

aDiferencia del efecto de aumento de la seguridad entre un escenario dado y el FP32.

Tabla 8

Resultados de la inyección de fallas de Yolo con Golden Custom.

Nivel	Datos	Observado			Enmascarado	Seguridad		Ganancia contra FP32	
		Crítico	Advertencia	Aceptar		<	=	Seguridada	Memoria
L0	FP32	20,17%	0,66%	16,25%	62,92%	37,08%	62,92%	–	–
	FP16	30,70%	1,03%	28,10%	40,16%	59,84%	40,16%	−22,76%	2x
	FP8	42,08%	5,63%	37,02%	15,27%	84,73%	15,27%	−47,64%	4x
	FxP32	13,35%	0,54%	12,43%	73,68%	26,32%	73,68%	10,76%	0
	FxP16	27,23%	1,13%	23,50%	48,14%	51,86%	48,14%	−14,78%	2x
L2	FP32	6,81%	0,10%	14,46%	78,63%	21,37%	78,63%	–	–
	FP16	12,07%	0,18%	23,04%	64,71%	35,29%	64,71%	−13,92%	2x
	FP8	14,19%	2,64%	38,61%	44,55%	55,45%	44,55%	−34,07%	4x
	FxP32	7,40%	0,43%	12,22%	79,95%	20,05%	79,95%	1,33%	0
	FxP16	14,39%	0,84%	23,60%	61,16%	38,84%	61,16%	−17,46%	2x
L4	FP32	5,05%	0,05%	11,24%	83,66%	16,34%	83,66%	–	–
	FP16	9,88%	0,10%	18,29%	71,73%	28,27%	71,73%	−11,94%	2x
	FP8	11,84%	1,70%	30,70%	55,76%	44,24%	55,76%	−27,90%	4x
	FxP32	6,32%	0,35%	11,86%	81,47%	18,53%	81,47%	−2,19%	0
	FxP16	12,45%	0,70%	22,51%	64,34%	35,66%	64,34%	−19,33%	2x
L6	FP32	4,02%	0.01%	7,59%	88,37%	11,63%	88,37%	–	–
	FP16	8,29%	0,04%	13,34%	78,33%	21,67%	78,33%	−10,04%	2x
	FP8	8,37%	1,25%	22,66%	67,72%	32,28%	67,72%	−20,65%	4x
	FxP32	4,49%	0,26%	11,04%	84,21%	15,79%	84,21%	−4,17%	0
	FxP16	8,82%	0,53%	20,99%	69,66%	30,34%	69,66%	−18,72%	2x
L8	FP32	3,41%	0.01%	3,94%	92,65%	7,35%	92,65%	–	–
	FP16	7,15%	0,02%	7,80%	85,03%	14,97%	85,03%	−7,61%	2x
	FP8	6,22%	1,04%	16,41%	76,33%	23,67%	76,33%	−16,31%	4x
	FxP32	2,84%	0,19%	9,22%	87,75%	12,25%	87,75%	−4,90%	0
	FxP16	5,49%	0,38%	17,48%	76,65%	23,35%	76,65%	−16,00%	2x
L10	FP32	3,26%	0,00%	1,70%	95,04%	4,96%	95,04%	–	–
	FP16	6,42%	0,02%	3,89%	89,67%	10,33%	89,67%	−5,36%	2x
	FP8	4,42%	0,52%	10,04%	85,01%	14,99%	85,01%	−10,03%	4x
	FxP32	1,67%	0.11%	6,85%	91,37%	8,63%	91,37%	−3,67%	0
	FxP16	3,14%	0,21%	12,41%	84,23%	15,77%	84,23%	−10,81%	2x
L12	FP32	3,17%	0,00%	0,84%	95,98%	4,02%	95,98%	–	–
	FP16	6,18%	0,07%	2,20%	91,55%	8,45%	91,55%	−4,44%	2x
	FP8	3,44%	0,23%	7,94%	88,39%	11,61%	88,39%	−7,59%	4x
	FxP32	0,76%	0,06%	5,34%	93,85%	6,15%	93,85%	−2,13%	0
	FxP16	1,39%	0,08%	9,90%	88,63%	11,37%	88,63%	−7,35%	2x
L13	FP32	3,27%	0.01%	3,37%	93,36%	6,64%	93,36%	–	–
	FP16	6,76%	0,04%	6,42%	86,78%	13,22%	86,78%	−6,58 %	2x
	FP8	5,20%	1,18%	14,59%	79,03%	20,97%	79,03%	−14,33 %	4x
	FxP32	1,86%	0,18%	9,08%	88,87%	11,13%	88,87%	−4,49%	0
	FxP16	3,61%	0,37%	16,92%	79,09%	20,91%	79,09%	−14,27%	2x
L14	FP32	3,13%	0.01%	1,25%	95,61%	4,39%	95,61%	–	–
	FP16	5,83%	0,07%	2,66%	91,43%	8,57%	91,43%	−4,18%	2x
	FP8	3,60%	0,25%	4,60%	91,54%	8,46%	91,54%	−4,06%	4x
	FxP32	1,03%	0,13%	5,96%	92,89%	7,11%	92,89%	−2,72%	0
	FxP16	2,05%	0,30%	11,29%	86,35%	13,65%	86,35%	−9,25%	2x
L15	FP32	0,82%	0.01%	0,19%	98,97%	1,03%	98,97%	–	–
	FP16	1,75%	0,04%	0,34%	97,87%	2,13%	97,87%	−1,10%	2x
	FP8	0,92%	0,17%	0,46%	98,45%	1,55%	98,45%	−0,53%	4x
	FxP32	0,34%	0,06%	0,36%	99,24%	0,76%	99,24%	0,26%	0
	FxP16	0,57%	0,12%	0,57%	98,74%	1,26%	98,74%	−0,23%	2x
L18	FP32	3,26%	0,02%	0,19%	96,53%	3,47%	96,53%	–	–
	FP16	6,10%	0,04%	0,38%	93,48%	6,52%	93,48%	−3,05%	2x
	FP8	5,22%	0,86%	10,56%	83,35%	16,65%	83,35%	−13,18%	4x
	FxP32	0,98%	0,14%	0,89%	98,00%	2,00%	98,00%	1,48%	0
	FxP16	2,04%	0,26%	2,01%	95,70%	4,30%	95,70%	−0,83%	2x
L21	FP32	3,22%	0,00%	0,06%	96,72%	3,28%	96,72%	–	–
	FP16	5,61%	0,02%	0,14%	94,23%	5,77%	94,23%	−2,50%	2x
	FP8	3,62%	0,48%	9,10%	86,80%	13,20%	86,80%	−9,92%	4x
	FxP32	0,37%	0,06%	0,56%	99,02%	0.98%	99,02%	2,29%	0
	FxP16	0,63%	0,17%	1,19%	98,00%	2,00%	98,00%	1,28%	2x
L22	FP32	0,44%	0,00%	0,03%	99,52%	0,48%	99,52%	–	–
	FP16	0,89%	0,00%	0,04%	99,07%	0,93%	99,07%	−0,45%	2x
	FP8	1,13%	0,06%	0,21%	98,61%	1,39%	98,61%	−0,91%	4x
	FxP32	0,14%	0,02%	0,04%	99.79%	0,21%	99.79%	0,27%	0
	FxP16	0,29%	0,02%	0,10%	99,59%	0,41%	99,59%	0,07%	2x

aDiferencia del efecto de aumento de la seguridad entre un escenario dado y el FP32.

degradación, para los tipos de datos de 16 bits (tanto FP16 como FxP16) en el 42 % de los casos (3 imágenes de 7) no hay degradación y en el 58 % de los casos (4 imágenes de 7) hay una degradación aceptable. degradación (es decir, todos los objetos se reconocen correctamente y la métrica IoU está entre 0,95 y 1). Finalmente, cuando se utiliza el tipo de datos de coma flotante de 8 bits (FP8), la CNN puede ofrecer resultados utilizables (es decir, clasificados como advertencia, 0,9 ≤ IoU ≤ 0,95) para el 28 % de las entradas (2 imágenes de 7) y no puede producir resultados correctos (críticos) para el resto (5 imágenes de 7). Por el contrario, con el tipo de datos de punto fijo de 8 bits (FxP8), la CNN no puede proporcionar los resultados correctos en absoluto, es decir, para todas las imágenes de entrada, la salida se clasificó como crítica .

3.7 Lista de fallas

En esta sección, presentamos la complejidad de YOLO en términos de inyección de fallas. En cuanto a LeNet-5, también para YOLO consideramos solo las capas que realizan cálculos aritméticos que involucran pesos entrenables, es decir, las trece capas convolucionales. La Tabla 6 proporciona detalles sobre la configuración, así como la lista de fallas de cada capa. La primera columna (etiquetada "Capa") informa las capas de destino; el segundo ("Conexiones") especifica el número de pesos de conexión. El número de posibles fallas se calcula como la multiplicación entre el número de conexiones ("Conexiones") y el tamaño del peso ("Bit-width").

Como señalan las columnas "#Faults" , el número total de posibles fallas es muy alto y esto se refleja en un tiempo de ejecución de la campaña de inyección de fallas no manejable. Como se hizo con LeNet-5, seleccionamos un subconjunto de fallas para reducir el tiempo de ejecución de inyección de fallas. Para cada capa, inyectamos el número de fallas reportadas en las columnas "#Inyecciones". El número de fallas a inyectar se obtuvo usando el enfoque presentado en [45] (ver Sección 3.3) y es estadísticamente representativo con un margen de error del 1% y un nivel de confianza del 99%. En promedio, se inyectan 15,7k fallas en cada capa para escenarios de 32 bits (FP32 y FxP32), 15,3k para escenarios de 16 bits (FP16 y FxP16) y 14,8k para escenarios de 8 bits (FP8 y FxP8). Las inyecciones se realizan seleccionando aleatoriamente el bit defectuoso entre todos los bits de los pesos de conexión.

3.8 Resultados de inyección de fallas

En esta sección, informamos los resultados de la campaña FI en YOLO CNN. En línea con la campaña LeNet-5 FI, se llevan a cabo dos conjuntos de experimentos. En primer lugar, evaluamos la confiabilidad usando como referencia la CNN estándar de punto flotante de 32 bits y luego usando la personalizada sin fallas. Los resultados se reportan en términos de la clasificación presentada en la Sección 2.2. A continuación, definimos las fallas que pertenecen a las clases Critical, Warning y Accept como eventos que reducen la seguridad de la CNN. La suma de estas contribuciones está representada por el símbolo '<' en las tablas. Por otro lado, consideramos las fallas enmascaradas como eventos que dejan inalterada la seguridad de la CNN. Así, su contribución está representada por el símbolo '=' en las Tablas.

La Tabla 7 informa los resultados del primer conjunto de campañas de FI.

Mientras que para LeNet-5 descubrimos que las versiones de punto fijo de la CNN tenían un nivel de seguridad promedio más alto (8,96 %) con respecto a las contrapartes de punto flotante, para YOLO la diferencia no es tan significativa. De hecho, para YOLO, las versiones de punto fijo de la CNN tienen un nivel medio de seguridad ligeramente inferior, es decir, −0,44%, con respecto a las versiones de punto flotante. Esto se calcula para las versiones de 32 y 16 bits, ya que la de punto fijo de 8 bits siempre arroja errores críticos (consulte la Tabla 5). Si incluimos también las versiones de 8 bits, las versiones de punto fijo tienen un nivel medio de seguridad inferior al −3,42% en comparación con las de punto flotante. Esto probablemente se deba a la diferente distribución de los valores de peso preentrenados para la red YOLO en comparación con LeNet-5 (compare la Fig. 8 con la Fig. 6). De hecho, para YOLO, la necesidad de más bits para la parte entera de los pesos reducía la precisión de la representación.

Además, cabe destacar que, en términos generales, YOLO resulta mucho menos resistente que LeNet-5. De hecho, mientras que para LeNet-5 el porcentaje de fallos críticos siempre es inferior al 3,42%, la red YOLO alcanza hasta el 48,46% de fallos críticos (capa L0, escenario FP8) y, como ya se ha comentado, la versión FxP8 produce fallos críticos incluso en un escenario libre de fallos. Esto probablemente se deba a la definición mucho más estricta que usamos para las fallas críticas. De hecho, para LeNet 5 clasificamos una falla como crítica solo cuando la predicción top-1 era incorrecta; por el contrario, para YOLO una falla se clasifica como crítica también cuando un objeto está correctamente clasificado pero no está perfectamente ubicado (IoU<0.9).

Finalmente, como en el caso de LeNet-5, también para YOLO reducir el ancho de bits implica reducir la resiliencia de CNN.

En la Tabla 8, informamos los resultados del segundo conjunto de campañas de FI, donde comparamos los resultados de cada escenario con los resultados obtenidos con la CNN personalizada sin fallas correspondiente.

En primer lugar, queda inmediatamente claro que, también para esta campaña de FI, la seguridad de la DNN se ve afectada por la reducción del ancho de bits. En segundo lugar, también en este caso, YOLO exhibe una alta incidencia de fallas críticas, es decir, hasta un 42,08 % en la capa L0 en el escenario FP8. Finalmente, para YOLO notamos el mismo fenómeno que observamos para LeNet-5: para las variantes personalizadas de punto flotante (FP16 y FP8), la diferencia entre los dos conjuntos de experimentos (Tablas 7 y 8) en términos de ganancia de seguridad con respecto a FP32 es mayor que para los de punto fijo (FxP32, FxP16 y FxP8).

De hecho, en promedio, la diferencia para las variantes de punto flotante en todas las capas es del 59,38 %, mientras que para las variantes personalizadas de punto fijo es del 13,92 %. Como ya se mencionó, esto significa que la aproximación a la versión FP32 CNN mediante el uso de una variante de punto flotante personalizada sin volver a entrenar expone a una mayor degradación de la seguridad que mediante el uso de la misma alternativa de punto fijo de ancho de bits.

4. Conclusiones

Este artículo presenta un marco de caracterización para analizar el impacto de las fallas permanentes que afectan una red neuronal convolucional destinada a ser implementada en sistemas críticos para la seguridad y con recursos limitados. La caracterización se realiza mediante campañas de inyección de fallas en el framework open source darknet . Los experimentos se realizan a nivel de software con el objetivo de ser independientes de la arquitectura del 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 CNN. En comparación con trabajos anteriores, mostramos que dependiendo de la definición del tipo de datos, podemos impactar profundamente la confiabilidad de la CNN. En particular, los resultados de la campaña de inyección de fallas mostraron que YOLO es menos resistente que LeNet-5, según las clasificaciones de fallas utilizadas. Además, para LeNet 5 CNN, los datos de punto fijo ofrecen un mejor equilibrio entre la reducción del espacio ocupado en la memoria y la resiliencia de la red en comparación con los datos de punto flotante. En conclusión, dependiendo del ancho de la distribución de los valores de ponderación de CNN, un diseñador puede decidir utilizar el tipo de datos de punto fijo o de punto flotante, con diferentes anchos de bits, para obtener diferentes compensaciones entre resiliencia y ganancia de recursos Por ejemplo, el uso del tipo de datos de punto fijo de 8 bits para LeNet-5 proporciona una reducción de 4 veces el espacio de memoria a un costo de menos del 0,45 % de fallas críticas. En el futuro, tenemos la intención de investigar la confiabilidad de los DNN mediante la explotación de otras representaciones de datos, como el tipo de datos enteros.

Declaración de competencia de intereses

Los autores declaran que no tienen intereses financieros en competencia ni relaciones personales conocidas que pudieran haber influido en el trabajo informado en este documento.

Referencias

[1] S. Albawi, TA Mohammed, S. Al-Zawi, Comprensión de una red neuronal convolucional, en: 2017 Conferencia Internacional sobre Ingeniería y Tecnología (ICET), 2017, pp. 1–6.

[2] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, A. Acero, Avances recientes en aprendizaje profundo para la investigación del habla en Microsoft, IEEE Conferencia Internacional sobre Acústica, Habla y Procesamiento de Señales (ICASSP).

[3] A. Krizhevsky, I. Sutskever, 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, en: NIPS'12, Curran Associates Inc. , EE. UU., 2012, págs. 1097–1105, URL http://dl.acm.org/citation.cfm? id=2999134.2999257.

[4] 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, D. Hassabis, Dominar el juego de Go con redes neuronales profundas y búsqueda de árboles, Nature 529 (2016) 484 EP –. URL http: //dx.doi.org/10.1038/nature16961.

[5] Y. LeCun, Y. Bengio, G. Hinton, Aprendizaje profundo, Nature 521 (2015) 436 EP –. URL http://dx.doi.org/10.1038/nature14539.

[6] W. Sung, Resiliencia de redes neuronales profundas bajo cuantificación, 2015, CoRR abs/1511.06488.

[7] C. Sequin, R. Clay, Tolerancia a fallas en redes neuronales artificiales, en: 1990 IJCNN International Joint Conference on Neural Networks, vol. 1, 1990, págs. 703–708, http://dx.doi.org/10.1109/IJCNN.1990.137651.

[8] EB Tchernev, RG Mulvaney, DS Phatak, Investigando la tolerancia a fallas de las redes neuronales, Neural Comput. 17 (7) (2005) 1646–1664, http://dx.doi.org/10.1162/0899766053723096 .

[9] J. Vialatte, F. Leduc-Primeau, Un estudio de la robustez del aprendizaje profundo frente a fallas informáticas, 2017, CoRR abs/1704.05396. arXiv:1704.05396. URL http: //arxiv.org/abs/1704.05396.

[10] C. Torres-Huitzil, B. Girau, Tolerancia a fallas y errores en redes neuronales: una revisión, IEEE Access 5 (2017) 17322–17341, http://dx.doi.org/10.1109/ACCESS. 2017.2742698.

[11] A. Ruospo, A. Bosio, A. Ianne, E. Sanchez, Evaluación de la confiabilidad de las redes neuronales convolucionales según su representación de datos, en: 2020 23rd Euromicro Conference on Digital System Design (DSD), 2020, pp. 672– 679, http://dx.doi.org/10.1109/DSD51259.2020.00109.

[12] A. Lotfi, S. Hukerikar, K. Balasubramanian, P. Racunas, N. Saxena, R. Bramley, Y. Huang, Resiliencia de las redes de detección de objetos automotrices en arquitecturas GPU, en: IEEE International Test Conference (ITC) de 2019, 2019, págs. 1 a 9, http://dx.doi. org/10.1109/ITC44170.2019.9000150.

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

[14] G. Di Natale, D. Gizopoulos, S. Di Carlo, A. Bosio, R. Canal (Eds.), Cross-Layer Reliability of Computing Systems, Instituto de Ingeniería y Tecnología, 2020, http:// dx .doi.org/10.1049/PBCS057E, URL https://biblioteca-digital.theiet. org/content/books/cs/pbcs057e.

[15] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, F. Aleotti, S. Mattoccia, percepción de profundidad monocular en microcontroladores para aplicaciones de borde, IEEE Trans. Sistema de circuitos Tecnología de vídeo. (2021) 1, http://dx.doi.org/10.1109/TCSVT. 2021.3077395.

[16] G. Ottavi, A. Garofalo, G. Tagliavini, F. Conti, L. Benini, D. Rossi, Un procesador RISC-V de precisión mixta para la inferencia DNN de borde extremo, en: Simposio anual de la IEEE Computer Society de 2020 sobre VLSI (ISVLSI), 2020, págs. 512–517, http://dx.doi.org/10.1109/ISVLSI49217.2020.000-5.

[17] L. Ravaglia, M. Rusci, A. Capotondi, F. Conti, L. Pellegrini, V. Lomonaco, D. Mal toni, L. Benini, Compensaciones de memoria-latencia-precisión para el aprendizaje continuo en un RISC -V extreme-edge node, en: 2020 IEEE Workshop on Signal Processing Systems (SiPS), 2020, pp. 1–6, http://dx.doi.org/10.1109/SiPS50750.2020.9195220.

[18] B. Moons, R. Uytterhoeven, W. Dehaene, M. Verhelst, 14.5 imaginan: Un procesador de red neuronal convolucional escalable de frecuencia y precisión de voltaje dinámico en paralelo de 0,26 a 10TOPS/W en FDSOI de 28 nm, en: 2017 IEEE International Solid-State Circuits Conference (ISSCC), 2017, pp. 246–247, http://dx.doi.org/10.1109/ISSCC.2017.7870353.

[19] C. Schorn, A. Guntoro, G. Ascheid, Predicción precisa de resiliencia neuronal para una gestión de confiabilidad flexible en aceleradores de redes neuronales, en: 2018 Design, Automation Test in Europe Conference Exhibition (DATE), 2018, pp. 979– 984, http://dx.doi.org/10.23919/DATE.2018.8342151.

[20] K. Simonyan, A. Zisserman, Redes convolucionales muy profundas para redes a gran escala reconocimiento de imágenes, 2014, arXiv:1409.1556.

[21] S. Mittal, Estudio de técnicas para computación aproximada, ACM Comput. sobrev. 48 (4) (2016) 62:1–62:33, http://dx.doi.org/10.1145/2893356, URL http: //doi.acm.org/10.1145/2893356.

[22] M. Courbariaux, Y. Bengio, J.-P. David, BinaryConnect: Entrenamiento de redes neuronales profundas con pesos binarios durante las propagaciones, 2015, arXiv:1511.00363.

[23] C. Zhu, S. Han, H. Mao, WJ Dally, Cuantificación ternaria entrenada, 2016, arXiv:1612.01064.

[24] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-Net: Clasificación de ImageNet mediante redes neuronales convolucionales binarias, 2016, arXiv:1603. 05279.

[25] MA Neggaz, I. Alouani, S. Niar, F. Kurdahi, ¿Son las CNN lo suficientemente confiables para aplicaciones críticas? Un estudio exploratorio, IEEE Des. Prueba 37 (2) (2020) 76–83, http://dx.doi.org/10.1109/mdat.2019.2952336.

[26] MA Neggaz, I. Alouani, PR Lorenzo, S. Niar, Un estudio de confiabilidad sobre CNN para sistemas integrados críticos, en: 2018 IEEE 36th International Conference on Computer Design (ICCD), IEEE, 2018, http:// dx .doi.org/10.1109/iccd.2018. 00077.

[27] F. dos Santos, L. Draghetti, L. Weigel, L. Carro, P. Navaux, P. Rech, Evaluación y mitigación de errores leves en la detección de objetos basada en redes neuronales en tres arquitecturas de GPU, 2017, págs. 169–176.

[28] Y. Ibrahim, H. Wang, J. Liu, J. Wei, L. Chen, P. Rech, K. Adam, G. Guo, Errores leves en los aceleradores DNN: una revisión exhaustiva, Microelectron. confiable 115 (2020) 113969, http://dx.doi.org/10.1016/j.microrel.2020.113969.

[29] A. Lotfi, S. Hukerikar, K. Balasubramanian, P. Racunas, N. Saxena, R. Bramley, Y. Huang, Resiliencia de las redes de detección de objetos automotrices en arquitecturas GPU, en: IEEE International Test Conference (ITC) de 2019, 2019, págs. 1 a 9.

[30] G. Li, SKS Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, SW Keckler, Comprender la propagación de errores en aceleradores y aplicaciones de redes neuronales de aprendizaje profundo (DNN), en: Procedimientos de la Conferencia internacional sobre computación, redes, almacenamiento y análisis de alto rendimiento, en: SC '17, ACM, Nueva York, NY, EE. UU., 2017, pp. 8:1–8:12, http://dx.doi.org/ 10.1145/3126908.3126964, URL http://doi.acm.org/10.1145/3126908.3126964.

[31] Tiny-CNN, 2020, URL https://github.com/nyanp/tiny-cnn.

[32] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, SK Lee, N. Mulholland, D. Brooks, 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.

[33] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, L. Benini, Un motor de navegación visual basado en DNN de 64 mW para nano drones autónomos, IEEE Internet Things J. (2019) 1 , http://dx.doi.org/10.1109/jiot.2019.2917066.

[34] B. Salami, O. Unsal, A. Cristal, Sobre la resiliencia de los aceleradores RTL NN: Caracterización y mitigación de fallas, 2018, arXiv:1806.09679.

[35] Y. He, P. Balaprakash, Y. Li, Fidelity: Marco de análisis de resiliencia eficiente para aceleradores de aprendizaje profundo, en: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), IEEE, Atenas, Grecia, 2020, págs. 270–281, http://dx.doi.org/10.1109/MICRO50266.2020.00033.

[36] F. Libano, P. Rech, B. Neuman, J. Leavitt, MJ Wirthlin, JS Brunhaver, Cómo la precisión de datos reducida y el grado de paralelismo afectan la confiabilidad de las redes neuronales convolucionales en FPGA, IEEE Trans. Núcleo ciencia (2021) 1, http://dx.doi.org/10.1109/TNS.2021.3050707, (Acceso temprano).

[37] M. Sabbagh, C. Gongye, Y. Fei, Y. Wang, Evaluación de la resistencia a fallas de las redes neuronales profundas comprimidas, en: Conferencia internacional IEEE sobre software y sistemas integrados (ICESS) de 2019, IEEE, Las Vegas, NV , EE. UU., 2019, págs. 1 a 7, http://dx.doi.org/10.1109/ICESS.2019.8782505.

[38] B. Du, S. Azimi, C. de Sio, L. Bozzoli, L. Sterpone, Sobre la confiabilidad de la implementación de redes neuronales convolucionales en FPGA basado en SRAM, en: Simposio internacional IEEE de 2019 sobre tolerancia a fallas y defectos en VLSI and Nanotechnology Systems (DFT), IEEE, Noordwijk, Países Bajos, 2019, págs. 1 a 6, http://dx.doi.org/10.1109/DFT.2019.8875362.

[39] A. Bosio, P. Bernardi, A. Ruospo, 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.

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

[41] [En línea], biblioteca Libfixmath, 2020, https://github.com/Petteri-Aimonen/libfixmath.

[42] M. Kooli, F. Kaddachi, GD Natale, A. Bosio, Evaluación de la confiabilidad del sistema con reconocimiento de caché y registro basada en el análisis de la vida útil de los datos, en: 2016 IEEE 34th VLSI Test Symposium (VTS), 2016, págs. 1 –6, http://dx.doi.org/10.1109/VTS.2016. 7477299.

[43] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Aprendizaje basado en gradientes aplicado al reconocimiento de documentos, Proc. IEEE 86 (11) (1998) 2278–2324, http://dx.doi.org/10.1109/5.726791 .

[44] Y. LeCun, et al., La base de datos MNIST, 2020, URL https://github.com/ashitani/ darknet_mnist.

[45] R. Leveugle, A. Calvez, P. Maistri, P. Vanhauwaert, Inyección estadística de fallas: error cuantificado y confianza, en: 2009 Design, Automation Test in Europe Conference Exhibition, 2009, pp. 502–506, http : //dx.doi.org/10.1109/FECHA. 2009.5090716.

[46] J. Redmon, SK Divvala, RB Girshick, A. Farhadi, Solo miras una vez: detección unificada de objetos en tiempo real, 2015, CoRR abs/1506.02640. arXiv:1506.02640. URL http://arxiv.org/abs/1506.02640.

Annachiara Ruospo recibió el M.Sc. Licenciada en Ingeniería Informática del Politecnico di Torino, Italia, en 2018, donde actualmente cursa el Ph.D. grado con el Departamento de Control e Ingeniería Informática. Sus principales intereses de investigación incluyen la prueba y verificación de dispositivos integrados modernos y la evaluación de confiabilidad de SoC orientados a IA.

Ernesto Sánchez recibió el título de ingeniero electrónico de la Universidad Javeriana, Bogotá, Colombia, en 2000, y el Ph.D. Licenciado en ingeniería informática por el Politecnico di Torino, Italia, en 2006, donde actualmente es profesor asistente en el Departamento de Control e Ingeniería Informática. Sus principales intereses de investigación incluyen pruebas de microprocesadores y computación evolutiva.

Marcello Traiola recibió el Ph.D. grado en Ingeniería Informática de la Universidad de Montpellier, Francia, en 2019 y el M.Sc. Licenciado en Ingeniería Informática cum laude de la Universidad de Nápoles Federico II, Italia, en 2016. Actualmente es investigador postdoctoral en el Instituto de Nanotecnología de Lyon, Ecole Centrale de Lyon, en Francia. Sus principales temas de investigación son los paradigmas informáticos emergentes con especial interés en el diseño, las pruebas y la fiabilidad. Es miembro de IEEE.

Ian O'Connor (IEEE S'95–M'98–SM'07) es Profesor de Diseño de Sistemas Heterogéneos y Nanoelectrónicos en el Departamento de Ingeniería Electrónica, Eléctrica y de Control de la Ecole Centrale de Lyon, Francia. Actualmente es jefe del grupo de Diseño de Sistemas Heterogéneos en el Instituto de Nanotecnología de Lyon y Director de la red de investigación SoC2. Desde 2008, también ocupa un puesto de profesor adjunto en la Ecole Polytechnique de Montréal, Canadá. Sus intereses de investigación incluyen arquitecturas informáticas novedosas basadas en tecnologías emergentes, asociadas a métodos para la exploración del diseño. Es autor o coautor de más de 200 capítulos de libros, publicaciones en revistas, ponencias en congresos y patentes, ha ocupado diversos puestos de responsabilidad en la organización de numerosos congresos internacionales y ha sido líder de paquete de trabajo o coordinador científico de varios proyectos nacionales y europeos. También se desempeña como experto en IFIP (Federación Internacional para el Procesamiento de la Información) WG10.5 (Diseño e Ingeniería de Sistemas Electrónicos). Anteriormente se desempeñó como experto en el Observatorio Francés de Micro y Nano Tecnologías (OMNT) y ALLISTENE

Alberto Bosio (Miembro, IEEE) recibió el Ph.D. Licenciado en ingeniería informática por el Politecnico di Torino, Turín, Italia, en 2006. Actualmente es profesor titular en el Instituto de Nanotecnología de Lyon (INL), École Centrale de Lyon, Lyon, Francia. Publicó artículos que abarcan diversas disciplinas, incluidas pruebas, verificación, confiabilidad, computación aproximada y tecnologías emergentes.