Un análisis de confiabilidad de una red neuronal profunda

1,2Alberto Bosio
[1]INL, Ecole Centrale de Lyon
Lyon, Francia
alberto.bosio@ec-lyon.fr

2Paolo Bernardi, 2Annachiara Ruospo, 2Ernesto Sanchez
2DAUIN, Politecnico di Torino
Torino, Italia
{paolo.bernardi, annachiara.ruospo, ernesto.sanchez}@polito.it

Resumen—El aprendizaje profundo, y en particular su implementación mediante redes neuronales convolucionales (CNN), es actualmente uno de los modelos predictivos más utilizados para aplicaciones críticas para la seguridad, como la asistencia a la conducción autónoma en el reconocimiento de peatones, objetos y estructuras. Hoy en día, garantizar la confiabilidad de estas innovaciones se está volviendo muy importante ya que involucran vidas humanas. Una de las peculiaridades de las CNN es la resiliencia inherente a los errores debido a la naturaleza iterativa del proceso de aprendizaje. En este trabajo presentamos una metodología para evaluar el impacto de las fallas permanentes que afectan a las CNN explotadas para aplicaciones automotrices. Dicha caracterización se realiza a través de un entorno de inyección de fallas basado en el marco DNN de código abierto de darknet. Se muestran resultados sobre campañas de inyección de fallas donde fallas permanentes están afectando los pesos de conexión en LeNet y Yolo; el comportamiento de la CNN corrompida se clasifica según la criticidad de la desviación introducida.

Términos del índice: aprendizaje profundo, prueba, confiabilidad, inyección de fallas, Seguridad, Automotriz

I. INTRODUCCIÓN

El aprendizaje profundo [1], y en particular las redes neuronales convolucionales (CNN), son actualmente uno de los modelos predictivos más utilizados en el campo del aprendizaje automático. En la mayoría de los casos, las CNN brindan muy buenos resultados para muchas tareas complejas, como el reconocimiento de objetos en imágenes/videos, el descubrimiento de fármacos, el procesamiento del lenguaje natural hasta juegos complejos [2]–[4].

Una de las características peculiares de las CNNs es la resiliencia inherente a los errores debido a su naturaleza iterativa y proceso de aprendizaje [5]. Por lo tanto, estas técnicas ahora se utilizan mucho para aplicaciones críticas para la seguridad, como la conducción autónoma [6]. Para utilizar un dispositivo electrónico en una aplicación crítica para la seguridad, se debe evaluar la confiabilidad. Más en detalle, se calcula la probabilidad de que una falla pueda causar una falla.
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) [7].

Por lo general, las soluciones de prueba en el campo deben integrarse y activarse 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), enfoques funcionales de autoprueba (p. ej., autoprueba basada en software).

978-1-7281-1756-0/19/$31.00 ©2019 IEEE

[8]), 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. Se calcula como la relación entre el número de fallos detectados por la solución de prueba sobre el número de posibles fallos. La lista de fallos incluye todos los fallos posibles excepto aquellos que no pueden producir ningún fallo en el modo operativo. Algunos ejemplos se pueden encontrar en [9]. En la terminología ISO26262, estas fallas se denominan "Dependiente de la aplicación de fallas seguras" (SFAD). Como se indica en [7], identificar SFAD es crucial, porque permite eliminarlos de la lista de fallas y enfocar los esfuerzos de prueba hacia las fallas que conducen a fallas de la aplicación, únicamente.

El objetivo de este artículo es caracterizar el impacto de las fallas permanentes que afectan a una CNN mediante una campaña de inyección de fallas en el marco DNN de código abierto de la red oscura [10].

En la literatura, pocos trabajos se enfocaron en la confiabilidad de DNN, pero solo para errores leves (es decir, cambio de bit). En [11], los autores evaluaron 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.

En [12], los autores presentan un análisis más detallado. Caracterizaron la propagación de errores leves desde el hardware al software de aplicación de diferentes CNN. Las inyecciones se realizaron mediante el uso de un simulador DNN basado en el marco del simulador DNN de código abierto, Tiny-CNN [13]. Gracias a la flexibilidad del simulador, fue posible caracterizar cada capa para tener un análisis más preciso.

En nuestro estudio, consideramos la inyección de fallas permanentes con el objetivo final de evaluar la distribución SFAD y clasificar la criticidad de una desviación de predicción de CNN según la capa corrupta. Las principales contribuciones de este manuscrito con respecto al estado del arte se enumeran a continuación: • La inyección de fallas se lleva a cabo en la capa

de software, para ser independiente de cualquier arquitectura HW potencial que finalmente ejecute la CNN; • Los efectos de las fallas se clasifican en SFAD

(enmascarados), Seguros e Inseguros, de acuerdo con varios criterios.

• La Inyección de fallas permite identificar

las capas más sensibles para las cuales se puede diseñar un mecanismo de seguridad.

Se han caracterizado dos topologías de CNN diferentes utilizando el marco general de la red oscura, las CNN de LeNet y Yolo.

Los resultados experimentales muestran costos computacionales limitados para lograr una buena precisión en la clasificación del comportamiento defectuoso.

El resto del documento está organizado de la siguiente manera. Sección II presenta los antecedentes de los dos casos de estudio y la Sección III detalla el entorno de inyección de fallas. Experimental los resultados se presentan en la Sección IV mientras que la Sección V concluye el papel.

II. FONDO

En esta sección se presentan los casos de estudio utilizados en este trabajo. Como se indicó anteriormente, explotamos el código abierto de darknet Framework DNN [10] implementado en lenguaje C. Red oscura, como la mayoría de las CNN, se compone de tres tipos de capas: Convolución capa, capas de agrupación máxima y capa totalmente conectada (para clasificación). Dos CNN se utilizan como estudios de caso. La primera uno es LeNeT [14] utilizado como clasificador de dígitos escritos a mano tarea de reconocimiento (utilizamos la base de datos MNSIT de formación y validación). El segundo, llamado YOLO [15], es un Marco neural convolucional profundo capaz de detectar objetos en tiempo real, analizando hasta 45 fotogramas por segundo. YOLO es un predictor que CNN explota para detectar un determinado conjunto de objetos (es decir, de nuevo la lista de objetos reconocibles provino de [10]). En la Figura 1 se muestran los resultados de la predicción resaltando los objetos identificados, incluido el nombre del objeto y el área ocupación. Este ejemplo presenta una imagen que contiene tres objetos relevantes para ser detectados: un perro, una bicicleta y una auto. Además, el automóvil se reconoce además como un camión para un total de cuatro objetos detectados.

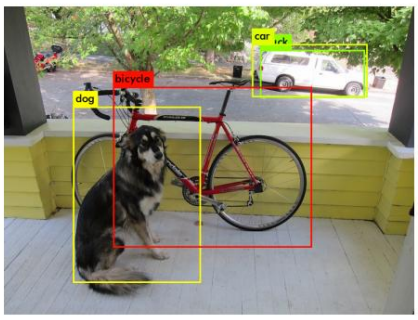


Fig. 1: Resultado de la predicción de Yolo CNN

Los detalles sobre las topologías de las redes se dan en la Tabla I y II. Para ambas tablas, la capa numérica se informa en la primera columna. El tipo de capa se especifica en la segunda columna como Convolución (Conv) o Totalmente conectado (FC). Entonces, el tercero columna detalla el número de conexiones para cada capa; mientras la última columna indica el ancho de bit utilizado para almacenar cada peso. Como se indica en la tabla, todos los pesos están representados como números de coma flotante de 32 bits.

TABLA I: Características de LeNet

Tipo de capa	Conexiones	Ancho	
0	conversión	2400	32
1	conversión	51200	32
2	FC	3211264	32
3	FC	10240	32

TABLA II: Características de Tiny YOLO

Tipo de capa	Conexiones	Ancho	
0	conversión	432	32
	conversión	4608	32
1	conversión	73728	32
2	conversión	294912	32
3	conversión	1179648	32
4	conversión	4718592	32
5	conversión	262144	32
6	conversión	1179648	32
7	conversión	130560	32
8	conversión	32768	32
9	FC	884736	32
10 11	FC	65280	32

tercero AVERÍA INYECCIÓN

Esta sección proporciona los detalles de un marco de Inyección de fallas construido en el marco de Darknet. Este es un generico entorno que se utiliza como un motor genérico para ejecutar varios tipos de redes neuronales. En nuestro caso, los resultados serán informado sobre los casos de estudio introducidos en el fondo párrafo. En términos generales, el sistema de hardware puede ser afectados por fallas causadas por defectos físicos de fabricación. Las fallas se propagan a través de las diferentes estructuras de hardware. componiendo el sistema completo (ver Fig. 2). Sin embargo, las fallas pueden ser enmascarado durante esta propagación ya sea en el tecnológico o en nivel arquitectónico [16]. Cuando una falla llega a la capa de software del sistema, puede corromper datos, instrucciones o el control fluir. Estos errores pueden afectar la correcta ejecución del software. produciendo resultados erróneos o impidiendo la ejecución de la aplicación que conduce a la terminación o aplicación anormal colgar. La pila de software puede desempeñar un papel importante en errores de enmascaramiento; al mismo tiempo, este fenómeno es implícitamente importante para la confiabilidad del sistema, pero un desafío difícil para ingenieros de pruebas que tienen que garantizar la seguridad de sus sistemas.

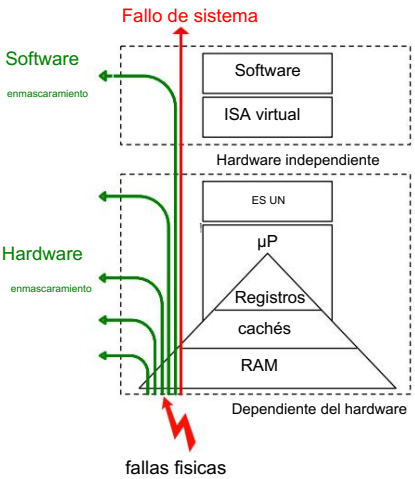


Fig. 2: Capas del sistema y propagación de fallas.

Como se indicó en la introducción, el objetivo de este trabajo es investigar el efecto de fallas permanentes en la capa de software

para ser independiente de la arquitectura del hardware
ejecutar la CNN (es decir, CPU, GPU o acelerador HW). Como
falla permanente, consideramos el modelo Stuck-at Fault (SaF)
en 0/1 (SaF0 y SaF1). La ubicación de la falla (F Lo) es
definida por (1).

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

donde Capa corresponde a la capa CNN, Conexión
es el borde que conecta un nodo de la capa y el bit
es uno de los bits del peso asociado a la Conexión.
Finalmente, la P olaridad puede ser '0' o '1' dependiendo del SaF.

El Fault Injector realmente funciona en la capa de software, y
su pseudocódigo se proporciona en el P seudo - Código (1). Él
corresponde a un simple inyector serial averiado que modifica el
Topología CNN como se describe en (1).

```
1 ejecute CNN (CNN, goldenprediction);
2 para ( i = 0 i < FLo · tamaño ( ) , yo ++ ) {
3 injectfault ( Flo [ i ] , CNN) ;
4 ejecutar CNN (CNN, predicción defectuosa);
5 comparar (predicción defectuosa, predicción dorada);
6 releasefault ( Flo [ i ] , CNN);
7 }
```

Listado 1: Pseudocódigo de inyección de fallas

El proceso de inyección de fallas consiste en lo siguiente: Una vez
la CNN está totalmente entrenada, se realiza una carrera dorada recogiendo
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 inyectar. esta falla
la lista debe verse como una lista de lugares donde inyectar las fallas
como se describió anteriormente. Luego, para cualquier falla en la lista de fallas
(línea 2 en 1), se realiza una ejecución de predicción y los resultados
recogidos y nombrados como predicción_defectuosa. En este punto (línea
5 en 1), se comparan los resultados obtenidos 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 escritura CNN defectuosa
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 respecto a la CNN defectuosa.
  y el dorado. Dependiendo de cuánto 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 de los mejores clasificados
    elemento varía en más de +/-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 [12] este es el más
    falla crítica observada;

Como se informó en la introducción, el objetivo de este trabajo es
para identificar el "dependiente de la aplicación de fallas seguras" (SFAD)
de acuerdo con la norma ISO 26262. Las fallas del SFAD pueden ser
simplemente calculado por la unión entre la falla enmascarada y la falla
observada segura como se muestra en (2).

SF AD = M solicitado Falla segura_observada _ (2)

IV. RESULTADOS EXPERIMENTALES

En esta sección se reportan los resultados experimentales recopilados.
Considerando que, las siguientes subsecciones presentarán los resultados para
las dos CNN. Finalmente, la última subsección discute sobre la
resultados obtenidos.

TABLA III: Lista de fallas de LeNet

Ancho de	conexiones de capa	#Faults		#inyecciones
0	2400	32	153600	9039
	51200	32	3276800	9576
1	3211264	32	205520896	9604
2 3	10240	32	655360	9465

TABLA IV: Lista de fallas de Tiny YOLO

Ancho de	conexiones de capa	#Faults		#inyecciones
0	432	32	27648	7128
	4608	32	294912	9301
1	73728	32	4718592	9584
2	294912	32	18874368	9599
3	1179648	32	75497472	9603
4	4718592	32	301989888	9604
5	262144	32	16777216	9599
6	1179648	32	75497472	9603
7	130560	32	8355840	9593
8	32768	32	2097152	9560
9	884736	32	56623104	9602
10 11	65280	32	4177920	9582

Las tablas III y IV presentan el tamaño de la lista de fallas de cada capa
de las dos topologías CNN. El número de posibles fallas.
es simplemente la multiplicación entre el número de conexiones
(columna 2) y el tamaño del peso multiplicado por 2 (es decir, atascado en 0 y
atascado en 1). Como se muestra en la columna 4, 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 de
1% y un nivel de confianza del 95%, y promedio de falla de 9K
las inyecciones deben ser consideradas. Los números exactos son
en la última columna de las tablas III y IV y fueron
calculado utilizando el enfoque presentado en [17].

A. LeNet

Para LeNet usamos los pesos preentrenados disponibles de
[18]. Para la campaña de inyección, seleccionamos al azar 37
imágenes de validación de la base de datos MNIST. Figuras 4, 5
y 6 representan los resultados obtenidos en términos de porcentaje de
Enmascarado, Seguro Observado e Inseguro Observado. Para cada gráfico,
diferenciamos las inyecciones por capa (de 0 a 3). Como lo
se puede ver, el porcentaje de fallas enmascaradas es mayor para
capa 2 y 3 (es decir, las capas totalmente conectadas), mientras que la primera
dos capas han mostrado un porcentaje más bajo de fallas enmascaradas.
Se puede realizar un análisis similar para las fallas observadas inseguras.
Las capas completamente conectadas muestran el porcentaje más bajo de Inseguro

Fallos observados. Esto significa que las capas menos "críticas" son las que están completamente conectadas (para la topología de LeNet).

En cuanto a las fallas Inseguras Observadas (Figura 6), nos gustaría resaltar que su porcentaje es muy bajo, variando desde 0.4% hasta 1.8%. Además, al considerar la “Capa 0”, las dos caídas del porcentaje de fallas observadas inseguras (correspondientes a la carga de trabajo 17 y 23) solo representan una diferencia del 1,4%, lo que significa que podemos considerar esta variación como insignificante. Esto significa que no podemos notar una dependencia particular en la carga de trabajo de entrada.

Este resultado es bastante interesante porque muestra una tendencia diferente con respecto al efecto de los errores leves. De hecho, se supone que las capas convolucionales (es decir, las dos primeras capas de LeNet) son las más resistentes a la presencia de una falla, según los resultados que se muestran en [12]. 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 resistentes porque clasifican las características extraídas por los dos primeros niveles. Por otro lado, estos resultados parecen confirmar la conclusión de [11] en la que los autores afirman esta tendencia.

Sin embargo, en [12] el resultado presentó otra tendencia: se encontró una resiliencia ligeramente mayor para la capa convolucional. Esto puede explicarse por dos factores: en primer lugar el modelo de fallas, aquí experimentamos fallas permanentes mientras que en [12] el modelo de fallas era transitorio, y la topología de la red.

La última parte de los experimentos realizados en LeNet es el análisis de los bits más críticos de la variable que almacena los pesos. Como se describió anteriormente, los pesos se representan como un formato de punto flotante binario de precisión simple como lo describe el estándar IEEE 754 como se muestra en la figura 3.

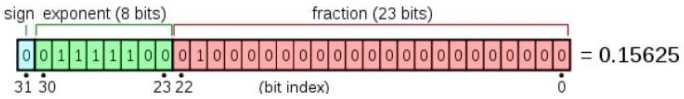


Fig. 3: Ejemplo de IEEE 754.

Como era de esperar, todas las fallas observadas no seguras se debieron a fallas que afectaron los 8 bits utilizados para almacenar el exponente (es decir, desde el bit 30 hasta el bit 23). Los bits de signo y mantisa no tienen impactos significativos (es decir, dieron lugar a fallos enmascarados o de observación segura).

B. YOLO

Para YOLO, usamos los pesos preentrenados disponibles en [10] y, más detalladamente, usamos yolov3-tiny.weights. Las cargas de trabajo también se descargaron de [10] y constan de siete imágenes diferentes. Para estos experimentos, mantenemos la misma clasificación de fallas que para LeNet, con solo una excepción.

LeNet clasifica la imagen de entrada como un dígito preciso. Por lo tanto, solo se consideraron los resultados de rango superior más altos. Usando YOLO, podemos tener más de un objeto que se puede detectar en la imagen, por ejemplo, podemos tener varios autos presentes en la imagen. Por lo tanto, tenemos que actualizar la definición de fallas observadas inseguras de la siguiente manera:

1) El número de objetos detectados es diferente del predicción áurea y defectuosa;

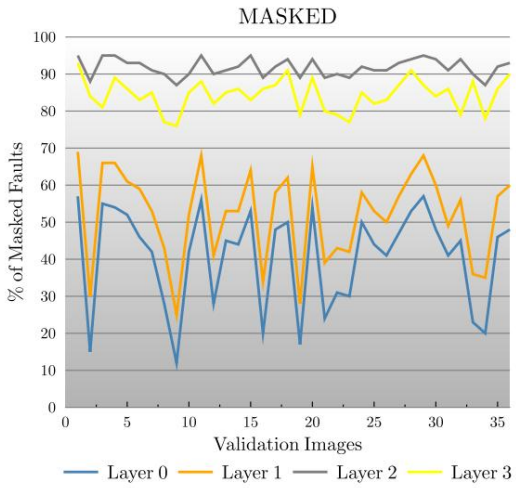


Fig. 4: Fallos enmascarados de LeNet.

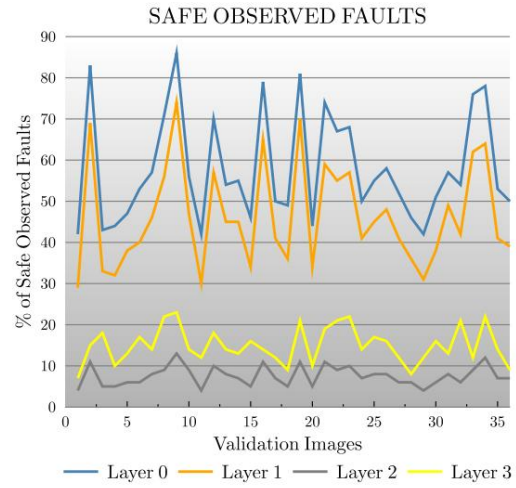


Fig. 5: Fallos observados de LeNet Safe.

2) El número de objetos detectados es igual, pero la etiqueta asociada con ellos es diferente (es decir, detección incorrecta); 3) La "ubicación" del elemento mejor clasificado varía en más de +/-5% con respecto al dorado.

Las otras definiciones (Enmascarado y Seguro) no cambian.

Para aclarar mejor esta definición, recurramos al ejemplo que se muestra en la figura 10 y comparemos con la predicción dorada que se muestra en la figura 1.

En el caso de fallos Safe Observed. De hecho, el YOLO defectuoso es capaz de detectar correctamente los cuatro objetos, pero su "ubicación" es ligeramente diferente (menos del 5%) que el dorado. El término “ubicación” significa el rectángulo que identifica el objeto en la imagen.

Por el contrario, la figura 10b y la figura 10c presentan casos de fallas observadas inseguras. En estas imágenes podemos notar que la CNN solo reconoce dos objetos (la bicicleta y el auto)

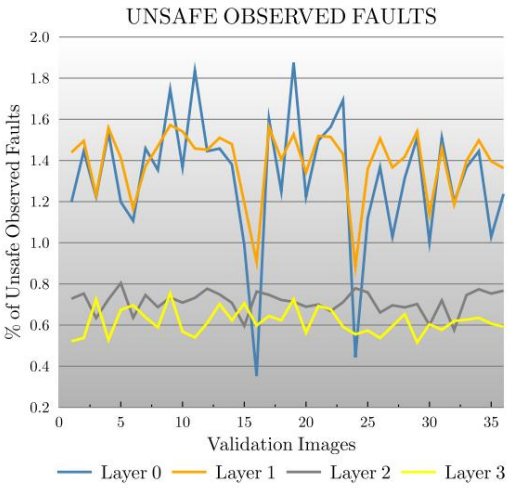


Fig. 6: Fallos observados de LeNet Unsafe.

y la predicción es incorrecta en términos generales.

Las figuras 7 y 8 muestran los resultados obtenidos en términos de porcentaje de fallas enmascaradas y no seguras observadas. Al contrario de los experimentos de LeNet, no tenemos un porcentaje significativo de fallos Safe Observed (es decir, el porcentaje obtenido fue inferior al 0,05 % de media). Como se puede observar, tenemos una población diferente de fallas enmascaradas en comparación con los datos de LeNet. En primer lugar, la distribución de las fallas enmascaradas no varía tanto entre las capas, a excepción de la primera. Por lo tanto, no podemos afirmar que las capas completamente conectadas sean las menos "críticas", ya que también la mayoría de las capas convolucionales mostraron un alto porcentaje de fallas enmascaradas. Por otro lado, también es interesante notar que dependiendo de la carga de trabajo (es decir, la imagen de entrada trazada en el eje X), la distribución de fallas enmascaradas cambia significativamente, mientras que para LeNet no fue el caso.

Un comentario final está relacionado con la imagen número 4 en la que podemos notar un pico de fallas observadas inseguras para todas las capas, pero especialmente para la "Capa 0". En cuanto a los resultados de LeNet, la "capa 0" es la capa más sensible a la carga de trabajo de entrada. Sin embargo, en este caso, la variación no se puede despreciar ya que observamos un aumento del 10% de fallas observadas inseguras. Por lo tanto, investigamos profundamente la carga de trabajo de entrada número 4. La figura 9 muestra un ejemplo de fallas observadas inseguras, donde en lugar de detectar los caballos, la red defectuosa detecta una oveja. En primer lugar, la imagen de entrada es "compleja" ya que hay muchos caballos para detectar. Para esta imagen específica, las fallas parecen conducir a más fallas observadas inseguras para la "capa 0". Esto significa que, según la topología de la red, la carga de trabajo de entrada puede desempeñar un papel importante.

Una vez más, este resultado es interesante en el sentido de que demuestra claramente que no es posible generalizar los resultados. Significa que cada CNN debe analizarse para identificar los elementos más críticos (es decir, las capas). A pesar de la diferencia entre los resultados obtenidos por las dos redes, el análisis del bit más crítico en los pesos confirma los resultados de LeNet ya que nuevamente obtuvimos que todas las fallas Inseguras Observadas son debido a fallas que afectan a los 8 bits utilizados para almacenar el exponente (es decir, desde el bit 30 hasta el bit 23). Los bits de signo y mantisa no tienen impactos significativos (es decir, dieron lugar a fallos enmascarados o de observación segura).

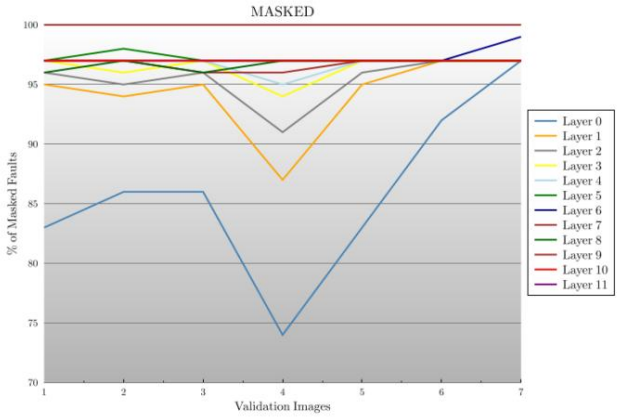


Fig. 7: Fallos enmascarados YOLO.

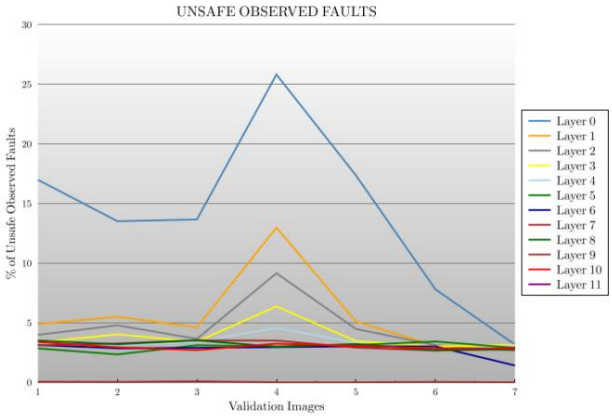


Fig. 8: Fallos observados inseguros de YOLO.

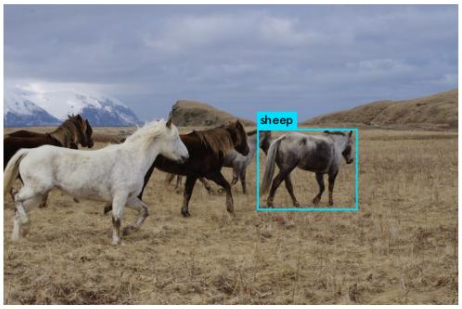
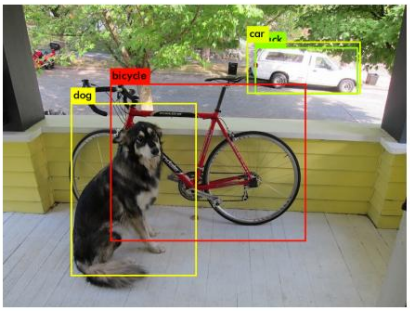


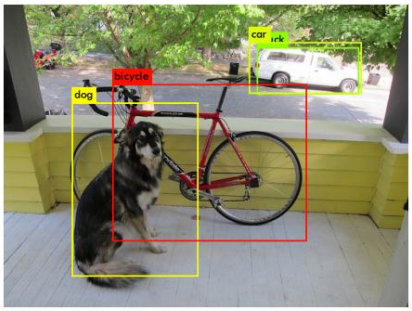
Fig. 9: Carga de trabajo de YOLO #4.

C. Discusión

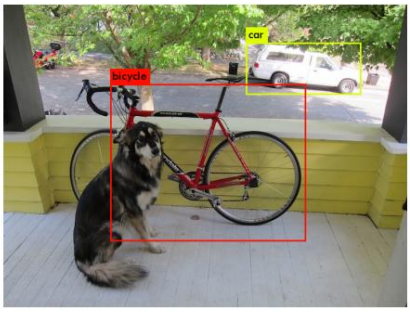
De acuerdo con los resultados presentados, la conclusión general sobre la criticidad de las CNN analizadas está relacionada con los bits de las variables que representan los pesos. Más en particular,



(a) Predicción Dorada



(b) Predicción segura de fallas observadas:



(c) Predicción de fallas observadas inseguras - fuerte corrupción del resultado de la predicción

reconocimiento faltante Fig. 10: Ejemplo de predicciones de YoLo.

el análisis mostró que tenemos que ocuparnos solo de 8 bits (es decir, los bits de exponente desde el bit 30 hasta el bit 23) que causan comportamientos críticos. Esta podría ser una solución si la redundancia se aplica solo en la parte crítica del sistema. Desde el punto de vista de la prueba, la conclusión puede ser la misma, en el sentido de que nuestra solución de prueba tiene que ser capaz de probar principalmente solo estos bits en particular. En este documento, no se presenta la descripción de la solución de prueba ni las técnicas de redundancia utilizadas para endurecer la CNN. De hecho, nuestro análisis tiene la ventaja de ser independiente del hardware, por lo que el diseñador tendrá la oportunidad de seleccionar cuidadosamente la arquitectura HW teniendo en cuenta cuidadosamente los bits más críticos. En consecuencia, el ingeniero de pruebas podrá centrar el esfuerzo de prueba solo en las partes críticas para reducir el costo de la solución de prueba.

V. CONCLUSIONES

En este artículo, presentamos un marco de caracterización para analizar el impacto de las fallas permanentes que afectan a la CNN destinada a ser utilizada en aplicaciones automotrices. La caracterización se realizó mediante una campaña de inyección de fallas en el marco DNN de código abierto de darknet [10]. Los experimentos se realizaron a nivel de software con el objetivo de ser independientes de la arquitectura HW y derivar una caracterización común del comportamiento de estas redes en presencia de estas fallas. Creemos que este es un resultado importante ya que el diseñador, a partir de estos resultados, podría seleccionar la arquitectura HW más conveniente para una CNN dada.

REFERENCIAS

[1] Y. LeCun, Y. Bengio y G. Hinton, “Aprendizaje profundo”, Nature, vol. 521, pp. 436 EP 05 2015. [En línea]. Disponible: http://dx.doi.org/10.1038/nature14539

[2] 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/ [3]

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

[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 y D. Hassabis, "Dominar el juego de ir con redes neuronales profundas y búsqueda de árboles" Naturaleza, vol. 529, pp. 484 EP 01 2016. [En línea]. Disponible: http://dx.doi.org/10.1038/nature16961

[5] W. Sung, "Resistencia de redes neuronales profundas bajo cuantificación". CoRR, vol. abs/1511.06488, 2015.

[6] 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 sobre visión por computadora (ICCV) de 2015, 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

[7] 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 19 Latin -Simposio Americano de Pruebas (LATS), págs. 1 a 6, 2018.

[8] 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.

[9] P. Bernardi, M. Bonazza, E. Sanchez, MS Reorda y O. Ballan, "Identificación de fallas funcionalmente no comprobables en línea en núcleos de procesadores integrados", en Actas de la Conferencia sobre Diseño, Automatización y Pruebas en Europa , ser. FECHA '13. San José, CA, EE. UU.: EDA Consortium, 2013, págs. 1462–1467. [En línea]. Disponible: http://dl.acm.org/citation.cfm?id=2485288.2485636 [10] J. Redmon, “Darknet: redes neuronales de código abierto en c,” http://pjreddie.com/darknet/, 2013–2016.

[11] 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.

[12] 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 [13] Tiny-cnn. [En línea]. Disponible: https://github.com/nyanp/tiny-cnn [14] 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.

[15] J. Redmon, SK Divvala, Girshick y A. Farhadi, "Solo mirPas una vez: detección unificada de objetos en tiempo real", CoRR, vol. abs/1506.02640, 2015. [En línea]. Disponible: http://arxiv.org/abs/1506.02640

[16] M. Kooli, F. Kaddachi, GD Natale y 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), abril de 2016, págs. 1–6.

[17] 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 .

[18] [En línea]. Disponible: https://github.com/ashitani/darknetmnist