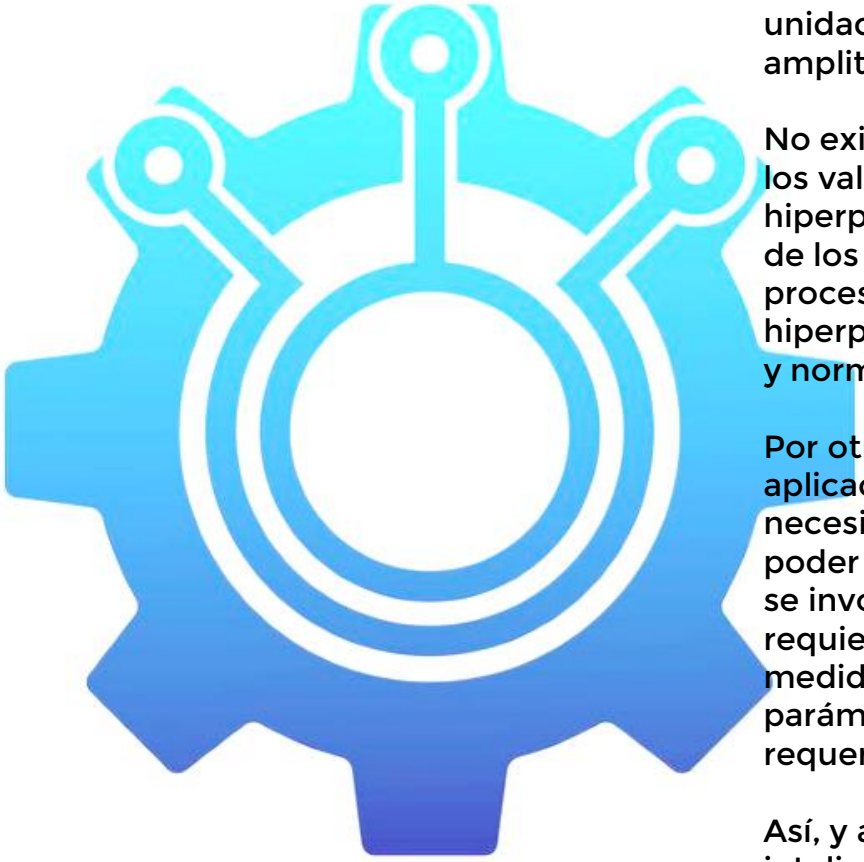

Universidad
Tecmilenio®

Universidad Tecmilenio

Proyectos de aprendizaje profundo

Técnicas de aprendizaje profundo
Hardware para el aprendizaje profundo

Introducción



Cuando se entrena una red neuronal, se deben tomar ciertas decisiones sobre algunos elementos importantes, como la cantidad de capas que debe tener la red, la cantidad de unidades o neuronas en cada capa, la tasa de aprendizaje o amplitud de paso, etc.

No existe una técnica o metodología definida para seleccionar los valores de estos elementos de la red, conocidos como hiperparámetros, entonces, una forma de mejorar la precisión de los modelos en aprendizaje profundo y de acelerar el proceso de entrenamiento es haciendo ajustes en los hiperparámetros, además de aplicar técnicas de regularización y normalización por lotes.

Por otro lado, es importante resaltar que en algunas de las aplicaciones donde se utiliza el aprendizaje profundo se necesitan grandes cantidades de operaciones de cómputo para poder analizar los datos. En redes neuronales profundas, donde se involucra una gran cantidad de parámetros también se requieren esfuerzos computacionales importantes y, en la medida que se busca mejorar el desempeño de un modelo, los parámetros crecen de forma exponencial, así como los requerimientos computacionales.

Así, y a pesar de que el futuro estará poblado de dispositivos inteligentes que necesitan plataformas de hardware económicas y de bajo consumo, el aprendizaje profundo a través de las redes neuronales están sujetas a una alta complejidad computacional, consumo energético y demanda de ancho de banda en memoria.

Explicación

Algoritmo de optimización

El **gradiente** de una función objetivo se puede utilizar en algoritmos de optimización cuando esta es diferenciable.

La **optimización** es un procedimiento para encontrar un conjunto de entrada a una función objetivo que resulte en una evaluación de función máxima o mínima. De manera más sencilla, encontrar las entradas que maximicen (o minimicen) el valor de una función.

Los **algoritmos de optimización** se utilizan en el proceso de entrenamiento de una red neuronal y modifican atributos de la red, como los pesos y la tasa de aprendizaje con el objetivo de minimizar el error.

Dependiendo del tipo de algoritmo de optimización, será el impacto en la reducción de las pérdidas, buscando los resultados más precisos posibles.

Explicación

Existen diferentes tipos de algoritmos de optimización (Velasco, 2020):

Gradiente descendente: es uno de los algoritmos básicos, pero mayormente utilizados en procesos de optimización. Es de los preferidos en problemas de regresión lineal y de clasificación, así como en retropropagación en redes neuronales.

La ecuación matemática del algoritmo es la siguiente: $\theta = \theta - \alpha . \nabla J(\theta)$

Gradiente descendente estocástico: es una variante del algoritmo de gradiente descendente y, a diferencia de este, actualiza los parámetros del modelo con mayor frecuencia, esto es, después del cálculo de las pérdidas (error) en cada instancia de entrenamiento.

La ecuación del algoritmo es la siguiente: $\theta = \theta - \alpha . \nabla J(\theta, x(i), y(i))$

Gradiente descendente por minilotes: es un algoritmo de mejora al gradiente descendente clásico y al estocástico. Los parámetros se actualizan después de cada lote en el que se subdivide el conjunto de datos de entrenamiento.

La ecuación que representa este algoritmo es la siguiente: $\theta = \theta - \alpha . \nabla J(\theta, B(i))$

Momento: es un algoritmo inventado para reducir la varianza que presenta el gradiente descendente estocástico y suavizar la convergencia. Acelera la convergencia en la dirección relevante y minimiza la fluctuación en la dirección irrelevante. El parámetro utilizado es el momento denotado por ***y***

La ecuación matemática del algoritmo es la siguiente:

$$V(t) = \gamma V(t-1) + \alpha . \nabla J(\theta)$$

$$\theta = \theta - V(t)$$

Estimación adaptativa del momento (Adam, por sus siglas en inglés): este algoritmo trabaja con momentos de primer y segundo orden, e realiza una búsqueda cuidadosa a un ritmo lento.

$$\hat{m} = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v} = \frac{v_t}{1 - \beta_2^t}$$

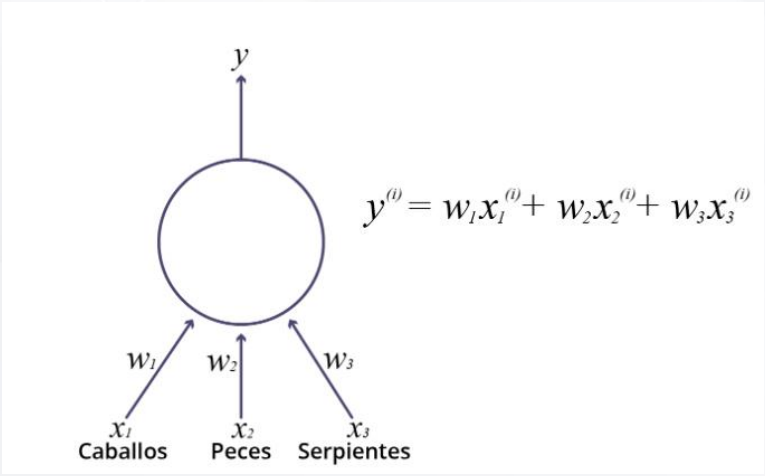
$$\theta_{t+1} = \theta - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}$$

Explicación

Además de los algoritmos de optimización que se presentan, están RMSProp, gradiente acelerado de Nesterov, Adagrad y AdaDelta (Li y Malik, 2017). La forma más eficiente de entrenar redes neuronales y en una menor cantidad de tiempo es con el optimizador Adam, aunque si se desea utilizar gradiente descendente, la mejor opción es la versión en minilotes.

Regularización

El entrenamiento de una red neuronal es un proceso en el que se obtienen los pesos de todas las conexiones en la red. A lo largo de este, se muestra a la red un gran número de ejemplos de datos de entrenamiento de donde aprenderá y, de forma iterativa, modificará o actualizará los valores de los pesos, con el objetivo de minimizar el error que se pueda cometer al entrenar con esos datos. Una vez que finalice el entrenamiento, la red será capaz de resolver tareas como para la que se entrenó.



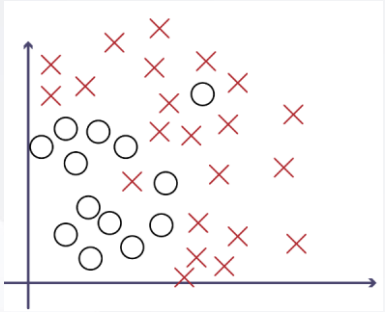
Neurona entrenada para identificar animales.

En la figura 1 se ejemplifica el entrenamiento de una neurona que se utilizará para identificar cierto tipo de animales de un universo. A partir de un conjunto de entrenamiento grande, es posible calcular lo que producirá esta neurona con la i-ésima instancia de entrenamiento, usando la expresión matemática mostrada.

Se pretende entrenar a la neurona para que escoja los pesos óptimos posibles (w_1, w_2, w_3), que minimicen los errores que puedan existir durante el proceso de entrenamiento y así identificar lo mejor posible al tipo de animal con el modelo obtenido.

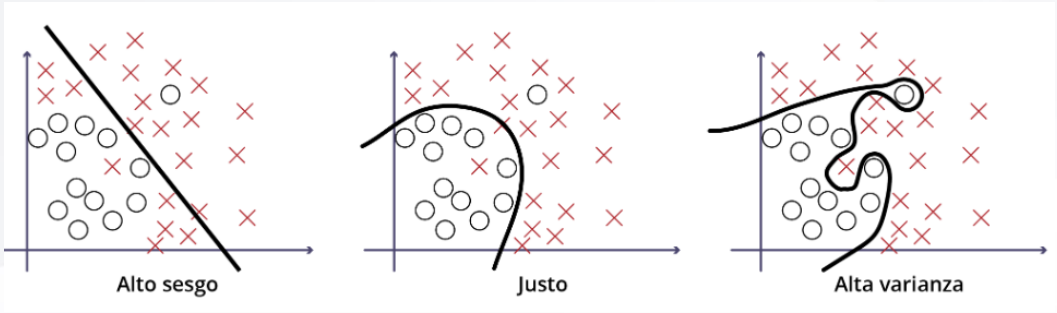
Explicación

Considera un conjunto de datos de entrenamiento cuya categoría es binaria, por ejemplo, el conjunto de datos de entrenamiento donde algunos elementos son manzanas y otros no. Al graficar tal conjunto de datos, la gráfica obtenida es como la de la figura 2; asume que las burbujas son manzanas y las "x" no lo son.



Es posible encontrar una función que permita identificar las dos clases posibles de los datos, por ejemplo, una línea recta. Con ella, el modelo presenta desajuste (underfitting) y sesgo grande. No funciona bien con los datos de entrenamiento ni con datos que aún no ha visto. Por otro lado, si la función ajusta perfectamente todos los puntos y puede clasificar todos los puntos dentro de la categoría que corresponda, es posible que el modelo presente alta varianza y sobreajuste o sobreaprendizaje (overfitting), por lo que no funcionará de forma adecuada con datos de prueba (invisibles).

El modelo que se pretende buscar debe estar entre estos dos extremos, un sesgo bajo, tanto en los datos de entrenamiento como en los de prueba. La idea se ejemplifica en la siguiente figura:



Explicación

La **regularización** es una técnica que hace pequeñas modificaciones al algoritmo de aprendizaje para que el modelo resultante generalice mejor y, con esto, mejorar el desempeño de este con datos aún no visibles (datos de prueba).

En el aprendizaje profundo, la regularización **modifica la función objetivo (o de costo)** cambiándola a una que incluya la pérdida (error) y un término de regularización, de la siguiente manera:

$$f_{error} = error + \lambda f(\theta)$$

Donde ***f(Ø)*** crece a medida que **Ø** crece y λ es el coeficiente de regularización que determina qué tanto se desea proteger al modelo de un sobreajuste. Si λ **=0** implica que no se desean tomar medidas para atender el sobreajuste, por el contrario, si λ es muy grande, el modelo dará prioridad a mantener **Ø** lo más pequeña posible en lugar de buscar valores de los parámetros que funcionen bien en el conjunto de entrenamiento.

Explicación

Existen **otras técnicas de regularización**, como el **abandono de neuronas** (dropout), **incremento de datos** (data augmentation) y **parada anticipada** (early stopping).

En la técnica de **abandono de neuronas** se mantienen ciertas neuronas activas con un valor de probabilidad asociado p (hiperparámetro). En cada iteración se seleccionan, de forma aleatoria, algunos nodos y se eliminan junto con todas sus conexiones entrantes y salientes.

En la técnica **parada anticipada** se entrena a la red iniciando con pesos pequeños y se detiene antes de que sufra un sobreajuste o sobreaprenda, es decir, cuando el desempeño sobre el conjunto de datos de validación empeore.

El **incremento de datos** es una técnica donde se utilizan diferentes "perspectivas" de los datos de entrenamiento para incrementar el tamaño de ese conjunto de datos y, con ello, reducir el sobreajuste (Hernández y König, 2018). Estas "perspectivas" no son otra cosa más que transformaciones sobre los datos originales que los presentan ligeramente modificados.

Ajuste de hiperparámetros

En los algoritmos de aprendizaje automático existen dos tipos de parámetros: los parámetros del modelo y los hiperparámetros. Los valores de los parámetros del modelo se aprenden durante el proceso de aprendizaje, por ejemplo, los pesos de una red neuronal.

Los hiperparámetros se utilizan para controlar el proceso de aprendizaje, por lo que deben configurarse antes de iniciar el entrenamiento. No pueden aprenderse directamente de los datos, se optimizan a lo largo del proceso de entrenamiento de la red.

Hay situaciones en las que el efecto del valor de un hiperparámetro sobre la capacidad del modelo no es claro y se sugiere utilizar algún tipo de metodología para lograr el ajuste de su valor de forma automática. Hay dos métodos populares para tal fin: **GridSearch y RandomSearch.**

Explicación

Supercomputadora

El aprendizaje profundo (AP) ha cambiado la vida del ser humano en gran medida, modificando la forma en la que interactúa con las computadoras y otras tecnologías por medio de la oferta de servicios novedosos de TI.

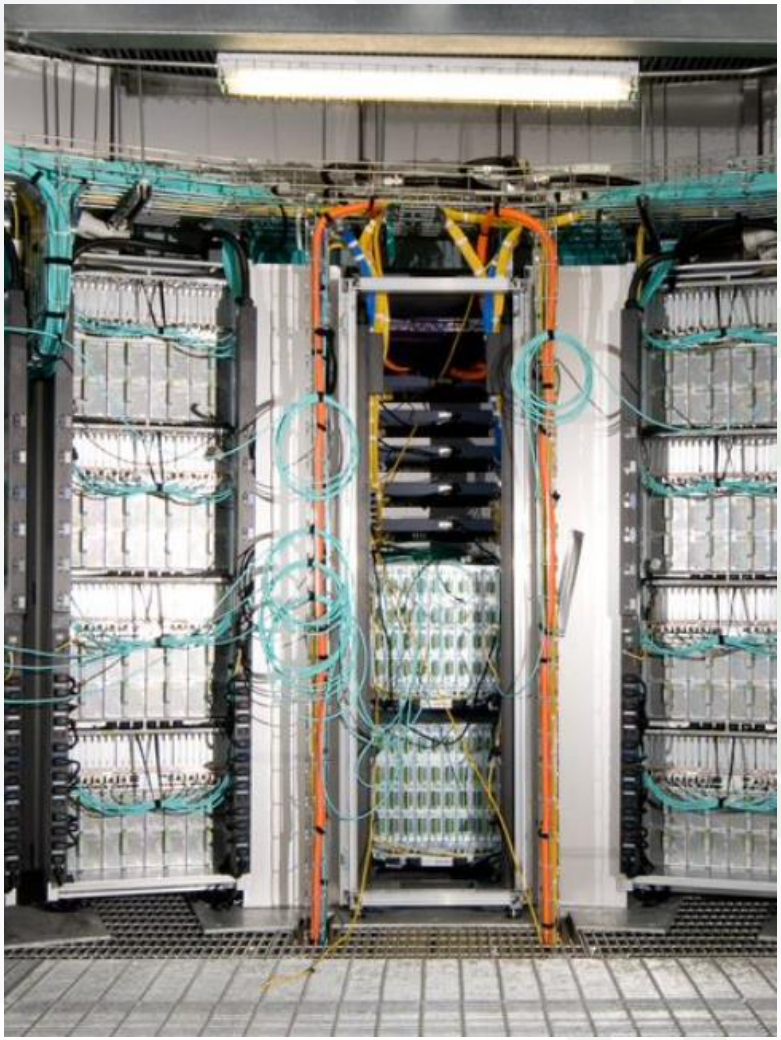
La visión computacional es tal vez el área que más se beneficia del aprendizaje profundo. Se detectan objetos en imágenes digitales para clasificarlas o categorizarlas con gran precisión. Estos avances se han convertido en mejoras en:

1.- Robótica.
2.- Drones autónomos.
3.- Sistemas de navegación asistida.

De manera similar, la identificación y reconocimiento de rostros se incluyen en muchos servicios de:
Seguridad y autenticación en entornos inteligentes,
Soluciones de Internet de las cosas (IoT, por sus siglas en inglés).

Además, existen otros avances en AP que están transformando los procedimientos de detección de enfermedades:

Leer y analizar tomas de rayos X,
Imágenes de resonancia magnética,
Electrocardiogramas y tomografías computarizadas.



Explicación

También hay ejemplos excepcionales del AP en el reconocimiento de voz como:

Cortana en Windows,
Alexa de Amazon
Siri de Apple.

Aplicaciones de texto como Google Translator son también resultados de aplicar AP en esa rama del procesamiento del lenguaje natural.

Actualmente existen unidades de procesamiento de gráficos (GPU, por sus siglas en inglés) con rendimientos del orden de decenas de teraflops capaces de procesar redes como la requerida para la identificación de objetos en un automóvil autónomo. Desafortunadamente, el consumo energético de estos GPU (entre 100 y 200 watts) superan cualquier presupuesto de energía de un sistema integrado típico. Aun y suponiendo que disipar 200 watts en un automóvil alimentado con una sola batería no fuera mayor problema, el costo de GPU tan potentes es una restricción (>1,000 USD).

La precisión y el desempeño de las redes neuronales están sujetos a una alta complejidad computacional, consumo energético y demanda de ancho de banda en memoria.

Por lo que algunos de los servicios construidos por el AP, de capacidades de cómputo y energía limitados, utilizan arquitecturas en la nube para dejar que las tareas de aprendizaje se realicen en supercomputadoras del lado de la nube.

Con el objetivo de satisfacer la demanda excesiva de poder computacional en AP, la industria de los semiconductores ha creado aceleradores de hardware de AP y procesadores específicos de dominio (Liu y Kin, 2021). Estos aceleradores son procesadores de propósito específico donde el data path (ruta de datos), el conjunto de instrucciones y la configuración del registro se adaptan a los requerimientos específicos y al modelo de flujo de datos de las redes neuronales, teniendo un mayor desempeño y eficiencia energética comparado con los procesadores de propósito general.

Explicación

Características necesarias para programar

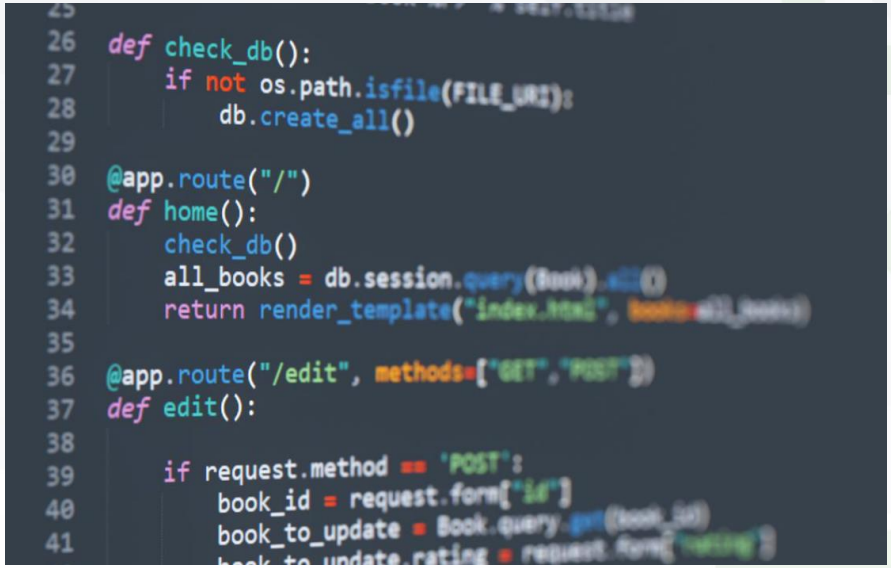
En esta era digital hay tecnologías que se han vuelto más populares que otras y de las que todo el mundo habla: inteligencia artificial (AI), aprendizaje automático (ML) y aprendizaje profundo (AP).

El ML es una tecnología que se centra en que las máquinas aprendan de forma independiente a partir de los datos, con poca o nula intervención humana o programación explícita. El AP es parte de ML y principalmente trata con redes neuronales (NN, por sus siglas en inglés). Las NN intentan emular el cerebro humano con el objetivo de producir resultados de la misma forma que la mente humana. Tanto el ML como el AP pertenecen a la IA.

Primero, se analiza el problema para determinar si es posible resolverlo con algoritmos tradicionales, en caso de que no, se puede optar por el AP.

Una vez que se haya decidido utilizar el AP, hay que asegurarse de tener conocimientos en:

Un lenguaje de programación apropiado para AP
Fundamentos de ciencias de la computación y estructura de datos
Matemáticas aplicadas
Tecnologías de Front End/UI y de despliegue
Plataformas de cómputo en la nube



Actividad

Durante el entrenamiento de redes de aprendizaje profundo, ¿qué técnicas de regularización utilizas?

¿Y cuáles técnicas de optimización?

Comparte las diferencias entre los algoritmos de búsqueda paramétrica para decidir cuál utilizar en problemas de aprendizaje profundo.

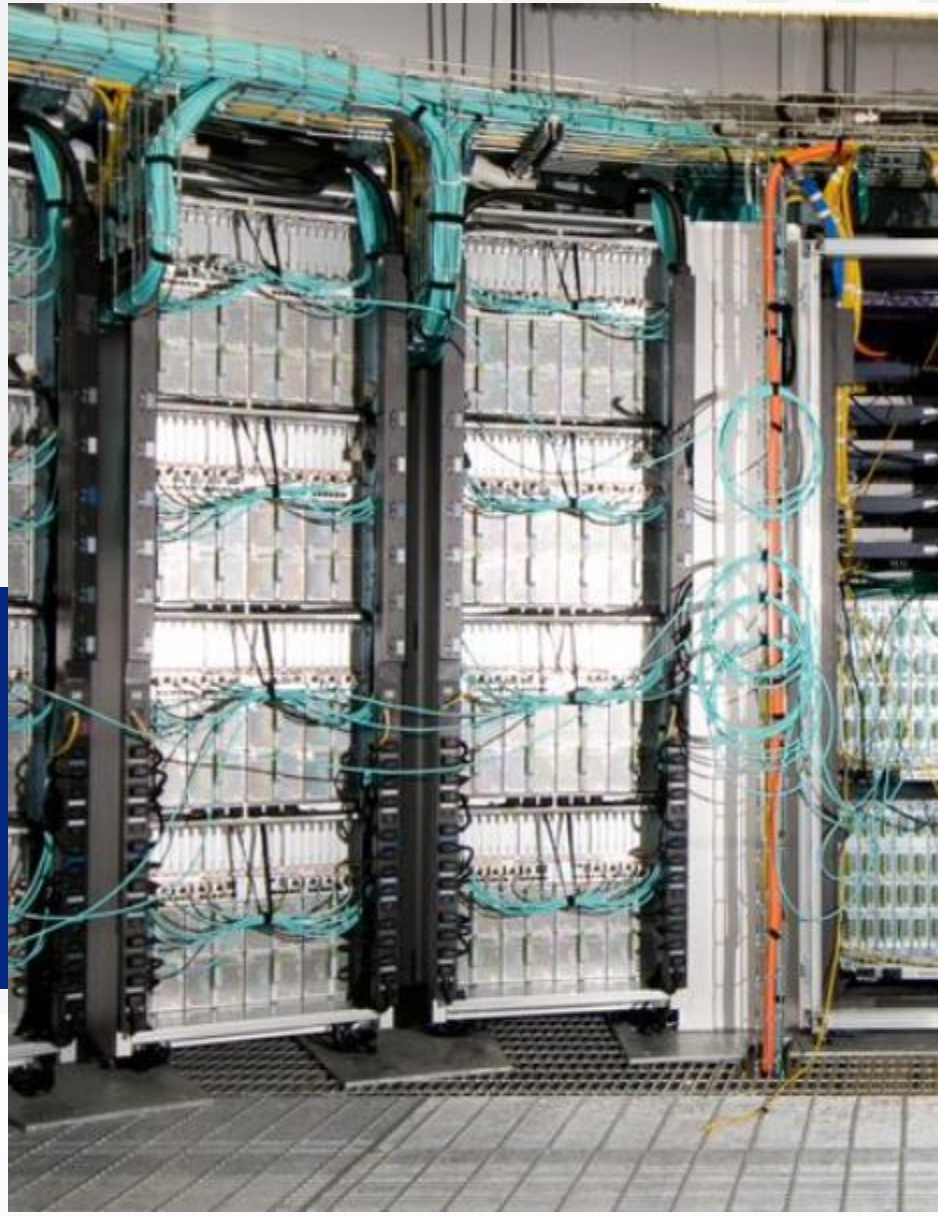
¿Qué tecnologías se utilizan en los aceleradores de hardware para su aplicación en soluciones basadas en aprendizaje profundo?

Identifica el conjunto de habilidades y conocimientos necesarios para programar redes neuronales.

Cierre

La arquitectura de una red no solo implica determinar la cantidad de capas que tendrá o de cuántas neuronas deben ser incluidas en cada una de ellas, también es importante identificar distintos valores de hiperparámetros que inciden directamente sobre el entrenamiento de la red y su desempeño.

Además, Dada la popularidad del AP en la industria y el uso de los modelos obtenidos en múltiples aplicaciones de la vida diaria, se adoptan alternativas de hardware que hagan más eficiente el proceso de entrenamiento y operación de los modelos, desde el uso de GPU y CPU hasta el desarrollo de aceleradores con FPGA y ASIC, incluso cómputo analógico.



Bibliografía

- Hernández, A., y König, P. (2018). Data augmentation instead of explicit regularization. Recuperado de http://arxiv.org/abs/1806.03852
- Li, K., y Malik, J. (2017). Learning to Optimize Neural Nets. Recuperado de http://arxiv.org/abs/1703.00441
- Liu, A., y Kin, O. (2021). Artificial Intelligence Hardware Design: Challenges and Solutions. Estados Unidos: Wiley-Blackwell.