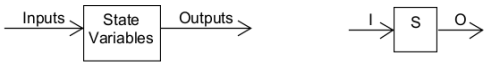
Internet of Things amb IA aplicada a la Indústria 4.0. Curs Online en Directe

Àrea: Transformació Digital

INTRODUCCIÓ A LES XARXES NEURONALS

Teoria General de Sistemes



$$(O, S_{post}) = F\ (I, S_{pre})$$

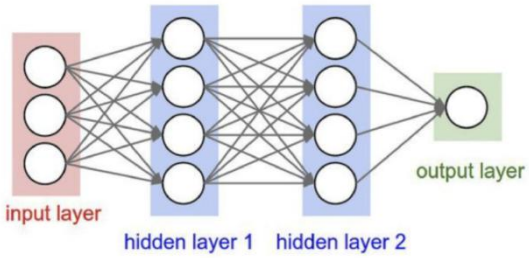
Segons el marc conceptual d'Ashby, quansevol sistema es por modelar en termes d'entrada, sortida i estat. Això es pot representar de forma genèrica amb una funció matemàtica F. Un sistema és dinàmic si varia al llarg del temps, per exemple connectant un input a un rellotge extern, altrament el sistema és estàtic. Si hi ha sortides connetades a entrades llavors el sistema és de llaç tancat (retroalimentat), altrament és de llaç obert. L'evolució de les variables pot ser continua o discreta.

Teorema de Hornik

Qualsevol funció matemàtica es pot aproximar amb una xarxa neuronal, llevades certes situacions de singularitat. Per tant, pràcticament qualsevol sistema es pot modelar amb una xarxa neuronal. Les xarxes neuronals són predictors univesals, en canvi les regressions lineals, polinòmiques, interpolacions, o els K-NN (K-Nearest Networks, mètode de predicció que utilitza els punts més propers en l'espai de dades per predir el valor d'un nou punt, on "K" és el nombre de veïns que es tenen en compte) no ho són.

Com funciona una xarxa neuronal?

Quan una neurona de la xarxa rep informació, aquesta informació és una combinació de valors que poden provenir d'altres neurones. La neurona fa una suma d'aquests valors i llavors aplica una funció d'activació per decidir si envia un senyal cap endavant o no.



Les funcions d'activació en les xarxes neuronals actuen com una mena d'interruptor que decideix si una neurona s'activa o no, és a dir, si envia un senyal als altres neurones o si es manté "en silenci". Aquest procés és clau perquè les xarxes neuronals aprenguin i puguin fer prediccions. La funció d'activació

converteix aquesta suma (que podria ser qualsevol nombre, com per exemple 5, -3 o 100) en un valor que sigui més útil i manejable per a la xarxa. Això ajuda la xarxa a funcionar correctament i a prendre decisions de manera eficient.

Si no existissin funcions d'activació, la xarxa només faria càlculs molt simples i lineals (com sumar i multiplicar), però no podria fer tasques complexes com reconèixer imatges o entendre textos. Les funcions d'activació donen a les xarxes neuronals la capacitat de "aprendre" patrons no lineals i de prendre decisions basades en aquests patrons.

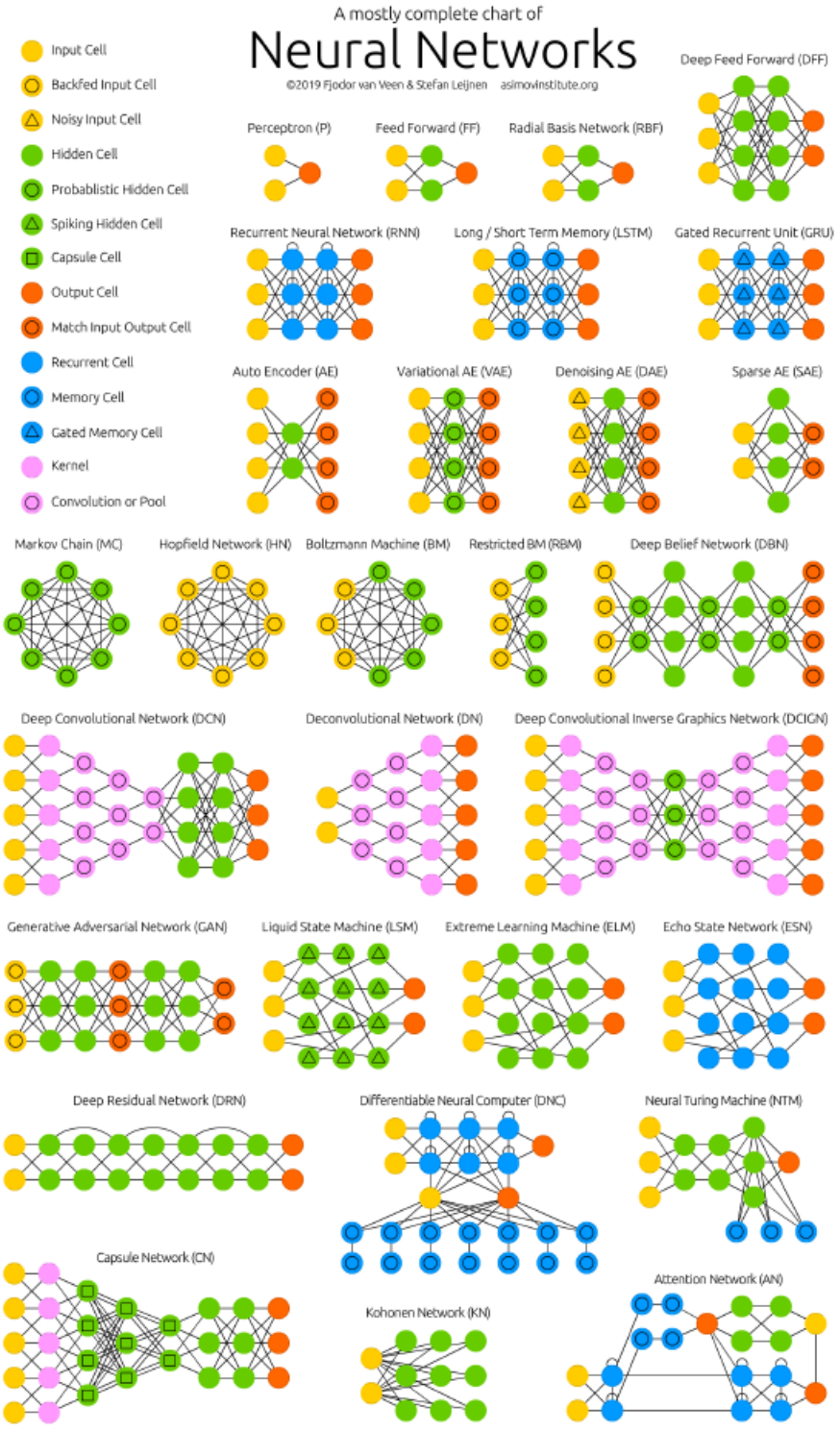
Hiperparàmetres d'una xarxa neuronal

Els hiperparàmetres d'una xarxa neuronal són els paràmetres de configuració que es decideixen abans de començar a entrenar la xarxa. Són com les "regles del joc" que defineixen com s'entrenarà la xarxa, però no són valors que la xarxa aprèn automàticament; es fixen per avançat.

Alguns exemples d'hiperparàmetres comuns són:

1. **Nombre de capes**: Quantes capes de neurones té la xarxa. Més capes poden permetre que la xarxa aprengui patrons més complexos, però també poden fer que l'entrenament sigui més lent o difícil.
2. **Nombre de neurones per capa**: Quantes neurones hi ha a cada capa. Això afecta la capacitat de la xarxa per processar informació i reconèixer patrons.
3. **Taxa d'aprenentatge** (*learning rate*): És la velocitat amb què la xarxa ajusta els seus paràmetres interns (els pesos) a mesura que aprèn. Si és massa alta, la xarxa pot aprendre de manera poc precisa, i si és massa baixa, pot aprendre massa a poc a poc. Fncionament: La xarxa fa una predicció (per exemple, intenta endevinar si una imatge és un gat o un gos), després es compara la predicció amb la resposta correcta: Si s'equivoca, ajusta una mica els paràmetres per acostar-se més a la resposta correcta segons la taxa d'aprenentatge. Si és alta, l'ajust serà gran, si és baixa, l'ajust serà petit.
4. **Mida del *batch***: Quan s'entrena la xarxa, sovint no se li passen totes les dades alhora. En lloc d'això, se li donen petits subconjunts (*batches*) de dades. La mida del *batch* és quantes dades se li donen a la vegada perquè vagi aprenent.
    1. **Mides petites de *batch***
        1. Avantatges: entrenament inicial ràpid, millor generalització, menys ús de memòria.
        2. Desavantatges: actualitzacions sorolloses, convergència menys estable, més *epochs* necessàries.
    2. **Mides grans de *batch***:
        1. Avantatges: actualitzacions més estables, entrenament més precís, millor ús dels recursos si hi ha suficient memòria.
5. Desavantatges: risc d'*overfitting*, necessitat de més memòria, entrenament més lent per *epoch*.
6. **Nombre d'epochs**: Quantes vegades se li passaran totes les dades d'entrenament per la xarxa. Una *epoch* és un cicle complet d'aprenentatge, un conjunt de batchs. Més *epochs* poden ajudar la xarxa a aprendre millor, però massa poden fer que sobreentreni (que aprengui massa els detalls dels exemples, però no generalitzi bé amb dades noves).
7. **Funció d'activació**: És la funció matemàtica que cada neurona utilitza per decidir si "s'activa" o no en resposta a una entrada. Diferents funcions d'activació poden afectar com la xarxa aprèn a reconèixer patrons.
8. **Optimitzador i funció de pèrdua:** L'optimitzador és l'algoritme que diu a la xarxa com ajustar els paràmetres per millorar-ne el rendiment. Utilitza el gradient per ajustar lleugerament els paràmetres de la xarxa a la direcció oposada, és a dir, cap a on la pèrdua disminueix. Aquest procés es repeteix moltes vegades fins que la xarxa troba la configuració de paràmetres que minimitza la pèrdua (calculada segons la funció de pèrdua).
9. **Mètriques**: Les mètriques són indicadors numèrics que ens diuen què tan bé està fent la xarxa neuronal la seva tasca. Per exemple, si la xarxa classifica imatges de gats i gossos, una mètrica podria ser el percentatge d'imatges que classifica correctament.

Topologies de xarxes neuronals



Tipus de cel·les en les xarxes neuronals:

1. Input Cell (Cèl·lula d'entrada): Neurona que rep la informació inicial. És la primera capa de la xarxa i és on entren les dades.

2. Backfed Input Cell (Cèl·lula d'entrada retroalimentada): Neurona que no només rep dades d'entrada, sinó que també pot rebre informació de capes posteriors de la xarxa.

3. Noisy Input Cell (Cèl·lula d'entrada sorollosa): Neurona que rep dades d'entrada amb soroll o incertesa, simulant variabilitat o errors en les dades.

4. Hidden Cell (Cèl·lula amagada): Neurona que es troba entre la capa d'entrada i la capa de sortida. Processa les dades però no és visible directament per als usuaris.

5. Probabilistic Hidden Cell (Cèl·lula amagada probabilística): Neurona que aplica un procés probabilístic per decidir el seu comportament, utilitzada en models amb incertesa.

6. Spiking Hidden Cell (Cèl·lula amagada d'impacte): Neurona que funciona amb pulsos o impactes, imitant com funcionen les neurones del cervell.

7. Capsule Cell (Cèl·lula càpsula): Neurona agrupada que processa la informació en una estructura més complexa, capturant relacions més riques entre les dades.

8. Output Cell (Cèl·lula de sortida): Neurona que produeix el resultat final de la xarxa, després de tot el processament intern.

9. Match Output Cell (Cèl·lula de coincidència de sortida): Neurona que compara la sortida de la xarxa amb algun patró o objectiu esperat per veure si coincideix.

10. Recurrent Cell (Cèl·lula recurrent): Neurona que pot utilitzar la seva pròpia sortida com a entrada en els cicles futurs, fent que la xarxa tingui memòria.

11. Memory Cell (Cèl·lula de memòria): Neurona que guarda informació durant un cert temps i la fa servir més endavant.

12. Gated Memory Cell (Cèl·lula de memòria amb porta): Neurona que controla quan guardar o deixar passar informació a través d'una "porta", com si fos un interruptor.

13. Kernel (Nucli): En xarxes neuronals convolucionals, el nucli és el filtre que s'utilitza per extreure característiques d'una imatge o senyal.

14. Convolution or Pool (Convolució o Agrupació): Operacions utilitzades per simplificar i extreure les característiques més rellevants d'una imatge o un conjunt de dades grans. La convolució és una operació matemàtica que transforma dues funcions en una tercera funció que representa la magnitud de superposició de les dues funcions originals. A la física, on hi hagi un sistema lineal amb un "principi de superposició", apareix una operació de convolució. A l'estadística, una mitjana mòbil ponderada és una convolució.

Tipus de xarxes neuronals:

1. Perceptron (P): La forma més senzilla de xarxa neuronal, amb una sola capa d'entrada i una capa de sortida. Cas d'ús: Classificar dades simples linealment separables, com detectar si un punt està per sobre o per sota d'una recta.

2. Feed Forward (FF): Xarxa on les dades flueixen només en una direcció, de les entrades cap a les sortides, sense bucles. Cas d'ús: Classificar correus com spam o no spam.

3. Radial Basis Network (RBF): Xarxa que utilitza funcions radials per classificar o ajustar dades, sovint utilitzada per reconeixement de patrons. Cas d'ús: Reconeixement de rostres bàsic utilitzant vectors de característiques.

4. Deep Feed Forward (DFF): Una versió més complexa de la xarxa "Feed Forward", amb múltiples capes entre l'entrada i la sortida. Cas d'ús: Predir el preu d'habitatges a partir de múltiples característiques (superfície, ubicació, habitacions…).

5. Recurrent Neural Network (RNN): Xarxa que té connexions internes, permetent que les dades circulin per la xarxa i guardin informació anterior. Cas d'ús: Predicció de sèries temporals simples, com preveure valors futurs d'un senyal de temperatura.

6. Long Short-Term Memory (LSTM): Una variant de les RNN dissenyada per recordar informació durant més temps i evitar problemes amb la pèrdua de memòria. Cas d'ús: Predicció de text (predictive text typing), com completar frases en un mòbil.

7. Gated Recurrent Unit (GRU): Similar a les LSTM, però amb una estructura més senzilla per gestionar informació a llarg termini. Cas d'ús: Traducció automàtica bàsica amb models seqüència-a-seqüència.

8. Auto Encoder (AE): Xarxa utilitzada per comprimir les dades a una forma més senzilla i, després, reconstruir-les. Cas d'ús: Reducció de dimensionalitat d'imatges per compressió sense supervisió.

9. Variational AE (VAE): Una versió de l'Auto Encoder que afegeix variabilitat probabilística, per a generar dades noves similars a les originals. Cas d'ús: Generar cares sintètiques realistes (imatges de persones inexistents).

10. Denoising AE (DAE): Xarxa que s'entrena per eliminar soroll de les dades, útil per netejar senyals o imatges. Cas d'ús: Eliminar soroll en fotografies antigues o imatges de baixa qualitat i convertir-les amb aparent alta qualitat.

11. Sparse AE (SAE): Xarxa que es limita a utilitzar només una petita part de les seves neurones, permetent un processament més eficient. Cas d'ús: Detecció de característiques latents en dades d'imatges (com bordes i textures), afavorint representacions eficients.

12. Markov Chain (MC): Model basat en la probabilitat que cada pas depèn només del pas anterior, utilitzat per prediccions de seqüències. Cas d'ús: Predir la paraula següent en textos molt simples basats només en la paraula anterior.

13. Hopfield Network (HN): Xarxa recurrent que s'utilitza principalment per emmagatzemar patrons i recuperar-los de manera associativa. Cas d'ús: Recuperar una imatge o patró "trencat", com recordar un patró complet a partir d'una versió parcial.

14. Boltzmann Machine (BM): Xarxa probabilística que intenta trobar les relacions més probables entre les dades. Cas d'ús: Modelatge de dades complexes amb incertesa, com patrons d'activació neuronal (Xarxa neuronal al servei d'una neurona)

15. Restricted BM (RBM): Versió simplificada de la Boltzmann Machine, amb restriccions a les connexions entre neurones. Cas d'ús: Recomanació de pel·lícules (p. ex. en sistemes tipus Netflix, abans dels models moderns).

16. Deep Belief Network (DBN): Xarxa profunda que combina moltes capes de neurones, aprenent característiques de manera jeràrquica. Cas d'ús: Reconeixement d'escriptura manual (MNIST) en els primers sistemes previs a les CNN.

17. Deep Convolutional Network (DCN): Xarxa neuronal especialment bona per treballar amb imatges, utilitza convolucions per extreure característiques d'aquestes. Cas d'ús: Reconeixement d'objectes en imatges (p. ex. classificar gats i gossos).

18. Deconvolutional Network (DN): Xarxa que fa el procés invers a les convolucions, reconstruint imatges a partir de característiques extretes. Cas d'ús: Reconstrucció d'imatges, p. ex. convertir una representació comprimida en una imatge completa.

19. Deep Convolutional Inverse Graphics Network (DCIGN): Xarxa que utilitza convolucions per generar representacions gràfiques, útil per a reconeixement d'objectes en imatges. Cas d'ús: Reconèixer i reconstruir objectes en 3D a partir d'imatges 2D (gràfics inversos).

20. Generative Adversarial Network (GAN): Xarxa que utilitza dues parts: una que genera noves dades i una altra que les jutja, ajudant a crear contingut molt realista. Cas d'ús: Generar imatges ultrarrealistes, com paisatges o cares humanes.

21. Liquid State Machine (LSM): Xarxa recurrent que emula el comportament del cervell per processar dades canviants en el temps. Cas d'ús: Processament de senyals en temps real, com reconeixement de paraules en veu contínua.

22. Extreme Learning Machine (ELM): Xarxa amb entrenament molt ràpid, sovint utilitzada per a classificació o reconeixement de patrons simples. Cas d'ús: Classificació instantània en temps real amb molt poques dades (p. ex. sistemes industrials ràpids).

23. Echo State Network (ESN): Xarxa recurrent amb una estructura interna complexa que es manté fixa durant l'entrenament, utilitzada per tasques de predicció. Cas d'ús: Predicció d'esdeveniments en sèries temporals caòtiques (clima, dades financeres no lineals).

24. Deep Residual Network (DRN): Xarxa profunda amb connexions especials que permeten que la informació es transmeti sense degradar-se, facilitant l'entrenament de xarxes molt profundes. Cas d'ús: Detecció d'objectes en imatges profundes i complexes, entrenant xarxes de moltes capes (50, 101, 152…).

25. Differentiable Neural Computer (DNC): Xarxa que combina la capacitat d'aprenentatge d'una xarxa neuronal amb l'emmagatzematge de dades, imitant com funcionen els ordinadors. Cas d'ús: Resoldre laberints o problemes d'arbre on cal memòria estructurada.

26. Neural Turing Machine (NTM): Xarxa neuronal capaç de manipular memòria externa, com un ordinador que pot recordar i manipular informació. Cas d'ús: Realitzar càlculs seqüencials com copiar, ordenar o duplicar seqüències de números.

27. Capsule Network (CN): Xarxa que agrupa neurones en "càpsules" per capturar relacions més complexes entre les dades, útil per reconèixer objectes en diferents angles. Cas d'ús: Reconèixer objectes sota rotacions fortes (angles diferents), com xifres o formes.

28. Kohonen Network (KN): Xarxa que s'utilitza per a la classificació i agrupació de dades, aprèn a organitzar informació de manera no supervisada. Cas d'ús: Agrupar dades en mapes 2D, com mapes de clients segons comportament de compra.

29. Attention Network (AN): Xarxa que aprèn a prestar atenció a parts específiques de la informació, molt útil per processar seqüències, com en traducció automàtica o reconeixement de veu. Cas d'ús: Traducció automàtica moderna (per exemple, el model seq2seq amb attention original de Bahdanau).

Llibreries populars open-source de xarxes neuronals

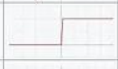
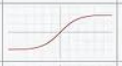
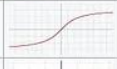
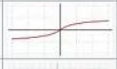
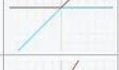
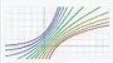
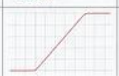
TensorFlow: Desenvolupada per Google és el més establert. És molt popular a grans empreses i té una comunitat molt gran d'usuaris. Sovint es considera més adequat per a projectes a gran escala i producció. (https://github.com/tensorflow/tensorflow) (https://github.com/eloquentarduino/python-eloquent-tensorflow).

PyTorch: Desenvolupada per Facebook (Meta) i Uber, és més recent i es considera més fàcil d'aprendre i fer servir. És molt popular en la investigació i en la comunitat acadèmica. (https://github.com/pytorch/pytorch).

YOLO: (You Only Look Once) Desenvolupada per la Univ. de Washington i l'empresa Ultralytics, amb una càmera és capaç d'analitzar les imatges d'aquesta en temps real i dir indicant la probabilitat quins objectes hi ha a l'escena, com ara cotxes, persones, senyals de trànsit, semàfors, etc. (https://github.com/ultralytics/yolov5)

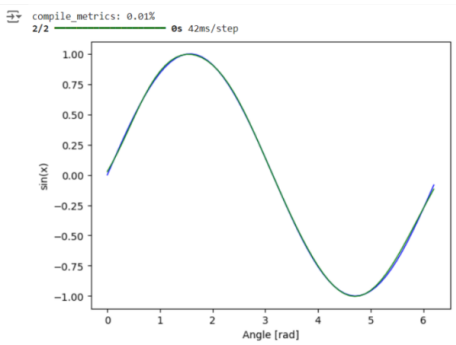
Eina popular: **Teachable Machine** (https://teachablemachine.withgoogle.com)

Funcions d'activació

Name	Plot	Equation	Derivative (with respect to x)	Range
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ [1]	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$
ElliotSig[9][10][11] Softsign[12][13]		$f(x) = \frac{x}{1 + \lvert x \rvert}$	$f'(x) = \frac{1}{(1 + \lvert x \rvert)^2}$	$(-1, 1)$
Inverse square root unit (ISRU)[14]		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1 + \alpha x^2}}\right)^3$	$\left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}}\right)$
Inverse square root linear unit (ISRLU)[14]		$f(x) = \begin{cases} \frac{x}{\sqrt{1 + \alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \left(\frac{1}{\sqrt{1 + \alpha x^2}}\right)^3 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$\left(-\frac{1}{\sqrt{\alpha}}, \infty\right)$
Square Nonlinearity (SQNL)[11]		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 0 \\ -1 & : x < -2.0 \end{cases}$	$f'(x) = 1 \mp \frac{x}{2}$	$(-1, 1)$
Rectified linear unit (ReLU)[15]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$
Bipolar rectified linear unit (BReLU)[16]		$f(x_i) = \begin{cases} ReLU(x_i) & \text{if } i \bmod 2 = 0 \\ -ReLU(-x_i) & \text{if } i \bmod 2 \neq 0 \end{cases}$	$f'(x_i) = \begin{cases} ReLU'(x_i) & \text{if } i \bmod 2 = 0 \\ -ReLU'(-x_i) & \text{if } i \bmod 2 \neq 0 \end{cases}$	$(-\infty, \infty)$
Leaky rectified linear unit (Leaky ReLU)[17]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Parameteric rectified linear unit (PReLU)[18]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$[2]
Randomized leaky rectified linear unit (RReLU)[19]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \text{ [3]} \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$
Exponential linear unit (ELU)[20]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$	$(-\alpha, \infty)$
Scaled exponential linear unit (SELU)[21]		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ with $\lambda = 1.0507$ and $\alpha = 1.67326$	$f'(\alpha, x) = \lambda \begin{cases} \alpha(e^x) & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$
S-shaped rectified linear activation unit (SReLU)[22]		$f_{t_l, a_l, t_r, a_r}(x) = \begin{cases} t_l + a_l(x - t_l) & \text{for } x \leq t_l \\ x & \text{for } t_l < x < t_r \\ t_r + a_r(x - t_r) & \text{for } x \geq t_r \end{cases}$ t_l, a_l, t_r, a_r are parameters.	$f'_{t_l, a_l, t_r, a_r}(x) = \begin{cases} a_l & \text{for } x \leq t_l \\ 1 & \text{for } t_l < x < t_r \\ a_r & \text{for } x \geq t_r \end{cases}$	$(-\infty, \infty)$
Adaptive piecewise linear (APL)[23]		$f(x) = \max(0, x) + \sum_{s=1}^{S} a_i^s \max(0, -x + b_i^s)$	$f'(x) = H(x) - \sum_{s=1}^{S} a_i^s H(-x + b_i^s)$ [4]	$(-\infty, \infty)$
SoftPlus[24]		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, \infty)$
Bent identity		$f(x) = \frac{\sqrt{x^2 + 1} - 1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2 + 1}} + 1$	$(-\infty, \infty)$
Sigmoid-weighted linear unit (SiLU)[25] (a.k.a. Swish[26])		$f(x) = x \cdot \sigma(x)$ [5]	$f'(x) = f(x) + \sigma(x)(1 - f(x))$ [6]	$[\approx -0.28, \infty)$
SoftExponential[27]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1 - \alpha(x + \alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x} - 1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1 - \alpha(\alpha + x)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$
Soft Clipping[28]		$f(\alpha, x) = \frac{1}{\alpha} \log \frac{1 + e^{\alpha x}}{1 + e^{\alpha(x-1)}}$	$f'(\alpha, x) = \frac{1}{2} \sinh\left(\frac{p}{2}\right) \operatorname{sech}\left(\frac{px}{2}\right) \operatorname{sech}\left(\frac{p}{2}(1 - x)\right)$	$(0, 1)$
Sinusoid[29]		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$	$[\approx -.217234, 1]$
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$	$(0, 1]$

Exemple amb TensorFlow (Google Colab https://colab.research.google.com): Predicció funció sinus

```
# Canviar entorn d'execució a la versió 2025.07
# Instal·lacions prèvies
# !pip install eloquent-tensorflow

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
import matplotlib.pylab as plt
from google.colab import files
from eloquent_tensorflow import convert_model

# 1. Create dataset
x = np.arange(0, np.pi * 2, 0.1)
y = np.sin(x)

# Keras expects 2D input: (samples, features)
x_train = x.reshape(-1, 1)
y_train = y.reshape(-1, 1)

# 2. Create the model
model = Sequential()
model.add(Dense(5, input_shape=(1,), activation='tanh'))
model.add(Dense(5, activation='tanh'))
model.add(Dense(1, activation='linear'))

# 3. Compile the model
opt = tf.keras.optimizers.SGD(learning_rate=0.01)
model.compile(loss='mean_squared_error', optimizer=opt,metrics=['mean_squared_error'])

# 4. Fit the model
model.fit(x_train, y_train, epochs=300, batch_size=1, verbose=1)

# 5. Evaluate the model
scores = model.evaluate(x_train, y_train, verbose=1)
print('%s: %.2f%%' % (model.metrics_names[1], scores[1] * 100))

# 6. Make predictions
y_pred = model.predict(x_train)

# Optional: plot
plt.plot(x, y, label='True sin(x)')
plt.plot(x, y_pred.flatten(), label='Model prediction')
plt.legend()
plt.show()
```




```
# Descarregar model entrenat
model.save_weights('model.weights.h5')
files.download('model.weights.h5')

# Generar codi eloquent per a ESP32
print(convert_model(model))

# Testing amb bessó digital WOKWI d'ESP32
# https://wokwi.com/projects/449284053394935809
```

Altres exemples

ESP32 CAM Eloquent

https://eloquentarduino.com/posts/esp32-cam-object-detection

Raspberry Pi YOLOv5

https://www.youtube.com/watch?v=pIUFXGxR_-Q&t=10m30s