Activation Functions

- **What are Activation Finctions?**

تخيل إن الشبكة العصبية بتتعلم تتعرف على صور القطط والكلاب

وظيفة ال activation functions

هي إنها تساعد النموذج انه يقدر يفرق بين الصور وازاي يتعامل مع أشكال مختلفة للبيانات بشكل ذكي

- Activation functions in the context of neural network are mathematical equations that determine the output of each neuron based on its input. They introduce non-linearity into the model, allowing it to capture complex patterns and relationships in the data.
- Without these functions, a neural network would essentially act like a linear model, limiting its capability to learn and generalize from intricate datasets.

وبالتالي دا بيفسر احنا ليه بنستخدم ال Activation Functions
لما بنعمل processing لل inputs من خلال ال Layers في ال ANN
و بتحولها بطرق معينة مما يؤدي إلى نتائج غير خطية
التعقيد ده مهم لأن المشاكل اللي بنحلها في العالم الحقيقي غالبًا فيها علاقات معقدة وتفاعلات.
لو مفيش activation functions
الشبكة كلها هتشتغل زي معادلة خطية واحدة، وده هيقلل من قدرتها على حل مشاكل معقدة زي التعرف على الصور أو معالجة اللغة وخلافه

- A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

- **Types of Activation Functions**

ليه أصلا عندنا ليها أنواع؟

لأن كل دالة بتقدم خصائص مختلفة تساعد ال ANNانها تتعلم بشكل أفضل على حسب نوع البيانات

أو ال Task ال عايزين نعمله

وبالتالي اختيار الدالة المناسبة بيأثر بشكل كبير على أداء النموذج النهائي

1. **Sigmoid / Logistic Function**

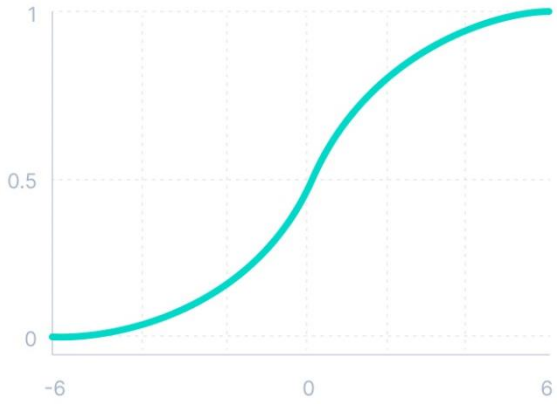
   The sigmoid function maps any real-valued number into the range of 0 to 1.

   It has an S-shaped curve.

   Often used in binary classification problems.

مفيدة لما يكون عندنا مشكلة تصنيف ثنائية (زي تصنيف صورة إذا كانت قطة أو كلب)

لأنها بتحول القيم المدخلة لأي رقم بين 0 و1

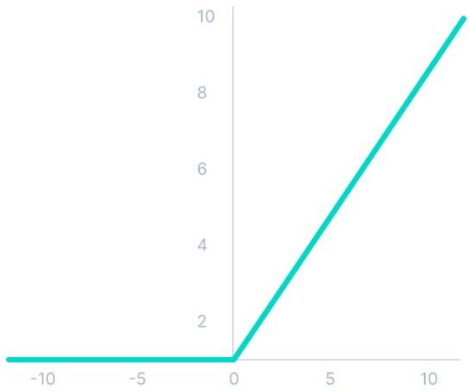


$$f(x) = \frac{1}{1 + e^{-x}}$$

2. **ReLU Function**

   ReLU stands for Rectified Linear Unit.

   The ReLU function outputs the input directly if it is positive; otherwise, it outputs zero.



$$f(x) = max\ (0, x)$$

if x < 0 → f(x) is 0

if x >= 0 → f(x) is z

There are multiple variants of the **ReLU** activation function, each designed to improve performance in certain situations

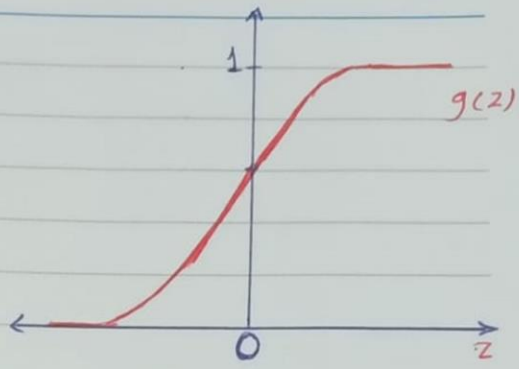
- **Leaky ReLU**
- **Parametric ReLU (PreLU)**
- **Exponential ReLU (ELU)**
- **Scaled Exponential Linear Unit (SELU**)

These variants aim to improve the learning process in deep networks, especially when dealing with large-scale datasets or specific architectures.

- Binary classification

• Sigmoid → y = 0 | 1

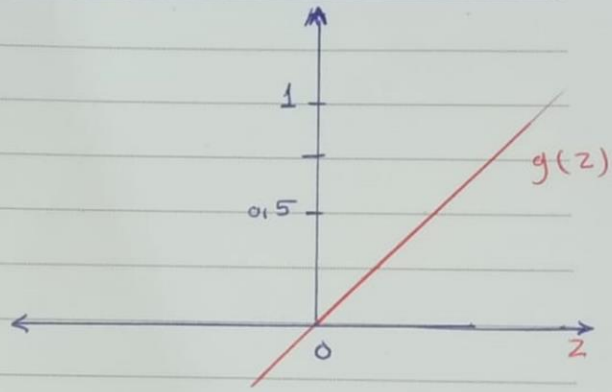
x_1	x_2	y
		0
		1
		0



- Regression

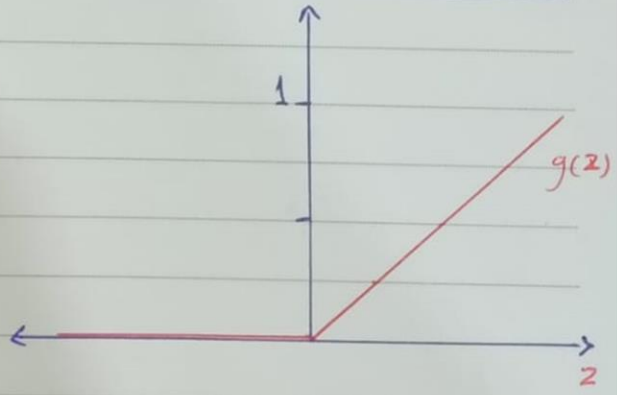
• Linear activation function

→ y = +/-



- Regression

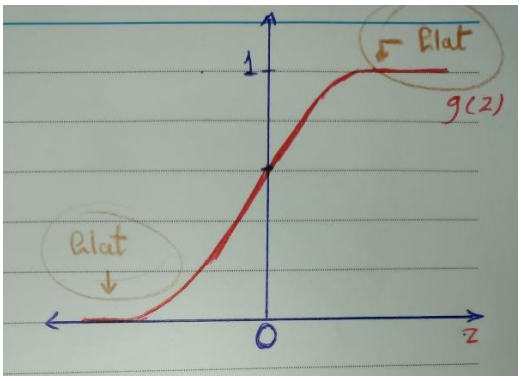
• ReLU → y = 0 or +

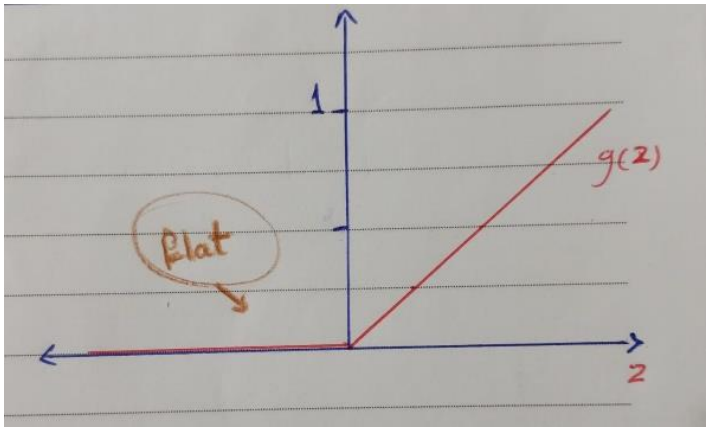


The one exception that you do use a sigmoid activation function in the output layer if you have a binary classification problem.

So why is that? Well, there are a few reasons..

- First, if you compare the ReLU and the sigmoid activation functions, the ReLU is a bit faster to compute because it . just requires computing max of 0, z .. whereas the sigmoid requires taking an exponentiation and th en a inverse and so on, and so it's a little bit less efficient.
- The second reason which turns out to be even more important is that the ReLU function goes flat only in one part of the graph; here on the left is completely flat, whereas the sigmoid activation function, it goes flat in two places. It goes flat to the left of the graph and it goes flat to the right of the graph. If you're using gradient descent to train a neural network, then when you have a function that is fat in a lot of places, gradient descents would be really slow.





لما نقارن بين RELU & Sigmoid

هنلاقي ان ال RELU أسرع لأنها عملية حسابها بسيطة (أكبر من صفر أو لا)

لكن ال Sigmoid بتحتاج لحسابات معقدة زي الأس والانعكاس، وده بيأثرعلى سرعة التدريب

3. Softmax

- converts a vector of numbers into a vector of probabilities.
- used when we have a problem of multi-class classification (ideal for classification tasks with multiple categories).

على عكس ال Sigmoid ال بتحول القيم بين (0,1)للفئتين

Softmaxبتحول كل قيمة من ال inputs لقيمة احتمالية بين ال (0,1)

بحيث يكون مجموع كل الاحتمالات يساوي1

Example : If we have a model classifying an image into three categories (e.g., Cat, Dog, or Rabbit), **Softmax** will give you probabilities for each category based on the model's confidence:

- Cat: 0.7
- Dog: 0.2
- Rabbit: 0.1

In this case, the **Softmax** output shows that the model is 70% confident the image is a Cat.

طيب ازاي بقا ال Softmax بتشتغل ؟

خلينا ناخد مثال بسيط ..

تخيل ان النموذج بيحاول يتعرف على صورة ويحدد إذا كانت :

{ Cat , Dog , Rabbit }

ال Model عطانا قيم لكل لكل فئة كالتالي :

z_1 (Cat) = 2.0

z_2 (Dog) = 1.0

z_3 (Rabbit) = 0.1

هنبدأ بعد كدا بحساب الأس لكل قيمة

بنحسب قيمة (e) مرفوعة لأس كل قيمة:

e^(z_1) = e^2.0 ≈ 7.39

e^(z_2) = e^1.0 ≈ 2.72

e^(z_3) = e^0.1 ≈ 1.1

بنضيف القيم اللي حسبناها في الخطوة الأولى عشان نحسب المجموع الكلي:

7.39+2.72+1.11 = 11.22

وأخيرا بنحسب النسبة لكل فئة

بنقسم كل قيمة من القيم على المجموع الكلي عشان نحصل على احتمال كل فئة:

Cat = 7.39 / 11.22 ≈ 0.66 (66%)

Dog = 2.72 / 11.22 ≈ 0.24 (24%)

Rabbit = 1.11 / 11.22 ≈ 0.10 (10%)

كدا نقدر نقول ان ال Modelمتأكد ان الصورةوليكن مثلا قطة بنسبة 66%

- Note: The activation functions covered in this document are some of the most commonly used ones in neural networks. There are additional activation functions that may be suitable for specific types of models or tasks, and as the field of deep learning evolves, more options continue to emerg. This work will remain a work in progress and will be updated to include new insights and functions in the future

(ان شاء الله تعالى)