《AI 神經網路资料》

<https://cloud.tencent.com/developer/article/1800954>

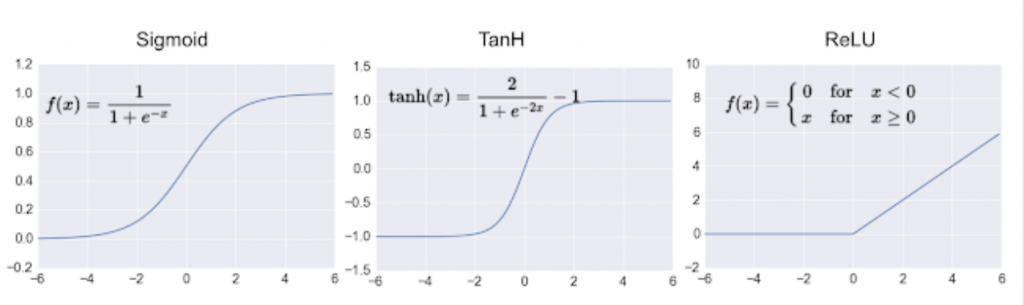
# 最常用的10个激活函数 (Sigmoid,Tanh,**ReLU**,Leaky Relu,PRelu,ELU,****Softmax**,**Swish,**Maxout,Softplus**)

<https://zhuanlan.zhihu.com/p/105722023>

* **Softmax 归一化指数函数** (hardmax, Numpy模块/ Sigmoid函数 )將各个输出节点的输出值范围映射到[0, 1]，***并且约束各个输出节点的输出值的和为1。***当然可以将输出为两个节点的二分类推广成拥有n个输出节点的n分类问题。

<https://zhuanlan.zhihu.com/p/424858561>

* **Sigmoid激活函数**(activation function**)**,它的域是所有实数的集合，它的范围是(0,1)。因此，如果函数的输入是一个非常大的负数或非常大的正数(-∞和 + ∞之间的任何数字)，则输出总是介于0和1之间，這樣将输入数据映射到某个范围内，再往下传递的好处是可以限制数据的扩张，防止数据过大导致的溢出风险。以Sigmoid函数为激活函数的神经元被称为sigmoid unit 。而由于Sigmoid函数是单调的，连续的，到处可微的 ，再加上它的导数可以用它自己来表示 的特性，使得当使用反向传播算法时，很容易推导出学习神经网络中的权重的更新方程。Sigmoid函数方式有可能會造成梯度消失，而且實際上電腦計算不容易，因為要用到指數，若資料或層數複雜時，則可能無法有效率的進行運算。(使用线性函数时，无法发挥多层网络带来的优势。因此，**为了发挥叠加层所带来的优势，激活函数必须使用非线性函数**。)
* **Tanh激活函数**是sigmoid向下平移、收縮的結果，其原理差不多，因此也會有共同的缺點。tanh 的输出间隔为 1，并且整个函数以 0 为中心，比 sigmoid 函数更好。
* **ReLU线性修正单元**(Rectified Linear Unit,ReLU)，也是***目前深层神经网络中经常使用的激活函数***，它模擬出一般生物上在遇到問題時，有些神經元並不會有反應，維持0的狀態，只會在有需要被使用的時候，才會有反應。再來可以不需要用到指數計算，只要判斷是否大於0即可。也能克服梯度消失問題，還能避免類神經網路的稀疏性等問題。计算速度快。*但当输入为负时，ReLU 完全失效。*
* **[Leaky ReLU 激活函數（P-Relu）](https://codingnote.cc/zh-tw/p/176736/" \l ":~:text=Leaky Relu %E5%87%BD%E6%95%B8%EF%BC%88P-Relu%EF%BC%89)**一种专门设计*用于解决 Dead ReLU 问题*(当输入为负时，ReLU 完全失效)的激活函数。
* **ELU激活函数** 的提出也解决了 ReLU 的问题。与 ReLU 相比，ELU 有负值，这会使激活的平均值接近零。均值激活接近于零可以使学习更快，因为它们使梯度更接近自然梯度。
* ****PReLU（Parametric ReLU）**激活函数**是 ReLU 的改进版本,shiyon使用参数通常为 0 到 1 之间的ML数字，并且通常相对较小。當爲负值域，PReLU 的斜率较小，这也可以避免 Dead ReLU 问题。
* ****Softmax**激活函数**，用于多类分类问题，當超过两个类标签则需要类成员关系。对于长度为 K 的任意实向量，Softmax 可以将其压缩为长度为 K，值在（0，1）范围内，并且向量中元素的总和为 1 的实向量，这意味着 Softmax 函数获得的各种概率彼此相关。主要缺点是：在零点不可微；及负输入的梯度为零，这意味着对于该区域的激活，权重不会在反向传播期间更新，因此会产生永不激活的死亡神经元。
* **Swish- 自门控(Self-Gated)激活函数**，使用相同的 gating 值来简化 gating 机制，这称为 self-gating。优点在于它只需要简单的标量输入，这使得Swish 之类的 self-gated激活函数能够轻松替换以单个标量为输入的激活函数（例如 ReLU），而无需更改隐藏容量或参数数量。Swish有助于防止慢速训练期间，梯度逐渐接近 0 并导致饱和。wish的平滑度在优化和泛化中起了重要作用。
* ****Maxout**激活函数，**是输入的最大值，因此只有 2 个 maxout 节点的多层感知机就可以拟合任意的凸函数。因此，由两个 Maxout 节点组成的 Maxout 层可以很好地近似任何连续函数。
* ****Softplus(**logistic/sigmoid) 激活函数**。类似于 ReLU 函数，但是相对较平滑，像 ReLU 一样是单侧抑制。它的接受范围很广：(0, + ∞)。



。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

<https://ithelp.ithome.com.tw/articles/10158350>

**XOR** (EOR or ExOR) (Exclusive-or gate) 逻辑算符互斥或, 與一般的邏輯或不同是数字逻辑中实现逻辑异或的逻辑门，當兩兩數值相同時為否，而數值不同時為真。

<https://baike.baidu.com/item/%E8%87%AA%E7%84%B6%E8%AF%AD%E8%A8%80%E5%A4%84%E7%90%86/365730>

**NLP**(Natural Language Processing)自然語言處理，利用人类交流所使用的自然语言与机器进行交互通讯的技术。

<https://zhuanlan.zhihu.com/p/25387613>

**Gradient descent**梯度下降法- 是一种基于函数性质的优化算法，希望利用其一阶性质持续向函数值下降最快的方向前进,找到使函数值达到全局最小的自变量值。(梯度下降法的缺点包括：靠近局部极小值时速度减慢。直线搜索可能会产生一些问题。可能会”之字型”下降。)

<https://medium.com/uxai/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E9%A6%AC%E6%8B%89%E6%9D%BE-075-%E5%8F%8D%E5%90%91%E5%82%B3%E6%92%AD-backpropagation-f1b612e003df>

<https://zhuanlan.zhihu.com/p/25609953>

**Backpropagation**反向传播算法- 构建比较复杂的神经网络模型，最重要的原因之一得益于反向传播算法。反向传播算法从输出端也就是损失函数开始向输入端基于链式法则计算梯度，然后通过计算得到的梯度，应用梯度下降算法迭代更新待优化参数。

反向傳播演算法靠著比較預測輸出和目標輸出的誤差，來決定要如何更新網路中的權重。通过學習率 (learning rate)，我們需要靠經驗來微調，更新規則很簡單，如遇權重增加時，誤差降低，那麼就增加權重；反之權重增加時，誤差也增加，那麼就降低權重。因此它通常被認為是一種監督式學習的方法，可以對每層迭代計算梯度，另外反向傳播要求神經元的啟動函數是可微分的。

<https://easyai.tech/ai-definition/cnn/>

# CNN (Convolutional Neural Networks) 卷积神经网络

<https://zhuanlan.zhihu.com/p/68487669>

## **FF**(Feed Forward)前馈神经网络,神经元是构成神经网络的基本单元，其主要是模拟生物神经元的结构和特性，接受一组输入信号并产出输出。常用的Sigmoid型激活函数有Logistics函数和Tanh函数；ReLU线性修正单元(Rectified Linear Unit,ReLU)，也是目前深层神经网络中经常使用的激活函数，Swish激活函数- 自门控(Self-Gated)激活函数。

(反馈网络反馈网络中神经元不但可以接收其它神经元的信号，也可以接收自己的反馈信号。和前馈网络相比，反馈网络中的神经元具有记忆功能，在不同的时刻具有不同的状态。称为记忆增强网络。)

<https://zhuanlan.zhihu.com/p/30844905>

**RNN** (Recurrent Neural Network）反復迴歸神經網路- 是一種用於處理序列數據的神經網路。相比一般的神經網路來說(前個輸入和後個輸入是沒有關係的)，**RNN** 能夠處理序列變化的數據(即前個的輸入和後個的輸入是有關聯的)。比如某個單詞的意思會因為上文提到的內容不同而有不同的含義，RNN就能夠很好地解決這類問題。(循环神经网络的隐藏层的值,不仅仅取决于当前这次的输入，还取决于上一次隐藏层的值。)

<https://zhuanlan.zhihu.com/p/52119092>

# NLP中的**RNN**-Seq2Seq(N vs M or Encoder-Decoder模型)与**attention注意力机制**

# <https://zhuanlan.zhihu.com/p/32085405>

# **LSTM**（Long short-term memory）长短期记忆是一种特殊的**RNN**，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，相比普通的RNN，LSTM能够在更长的序列中有更好的表现。

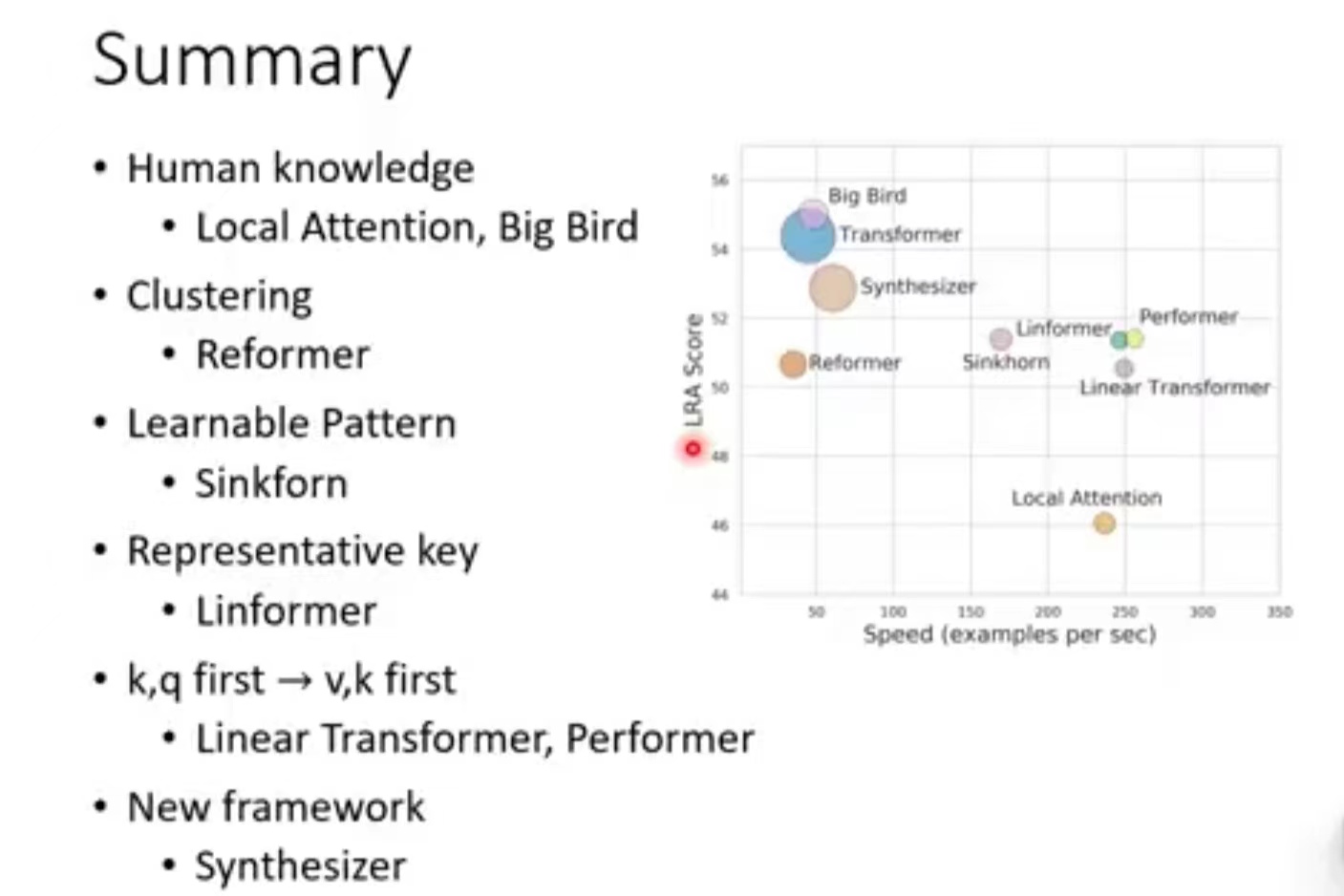
<https://zhuanlan.zhihu.com/p/32481747>

# **GRU**（Gate Recurrent Unit）是循环神经网络(**RNN**)的一种，GRU输入输出的结构与普通的RNN相似，其中的内部思想与LSTM相似。与LSTM相比，GRU内部少了”门控“，参数比LSTM少，但是却也能够达到与LSTM相当的功能。考虑到硬件的计算能力和时间成本，因而很多时候我们也就会选择更加”实用“的GRU。

<https://youtu.be/yHoAq1IT_og>

**Self-attention-** Local(Truncated)/ Stride/ Global/ Longformer=Local+Stride+Global / Big Bird= Longformer+(random)

Ps: Clustering(簇聚)/ ***[MLP](https://www.google.com/search?q=mlp+model&oq=mlp&aqs=chrome.1.69i57j0i512l7.3327j0j9&sourceid=chrome-mobile&ie=UTF-8" \l ":~:text=View%20all-,Multilayer,Perceptrons,-%2C%20or%20MLPs%20for)*** *[(Multilayer Perceptrons](https://www.google.com/search?q=mlp+model&oq=mlp&aqs=chrome.1.69i57j0i512l7.3327j0j9&sourceid=chrome-mobile&ie=UTF-8" \l ":~:text=View%20all-,Multilayer,Perceptrons,-%2C%20or%20MLPs%20for)*) / MLP-Mixer



# <https://zhuanlan.zhihu.com/p/82312421>

# **Transformer Architecture** 是一个利用attention注意力机制来提高模型训练速度的模型,里面主要有两部分组成：**Encoders** x6c 和 **Decoders** x6 (Seq2Seq or N vs M)。 encoder里边的结构是一个自注意力机制(**self-attention**)加上一个前馈神经网络（**feed forward**）。

**GNN** Graph Neural Network使用神经网络来学习图结构数据，提取和发掘图结构数据中的特征和模式，满足聚类、分类、预测、分割、生成等图学习任务需求的算法总称。按照图的种类划分，可以分为引文网络、社交网络、交通网络、图像、化合物分子结构、蛋白质网络等。按照应用领域划分，可以分为自然语言处理、图像处理、轨迹预测、物理化学和药物医学等。["

<https://medium.com/ai-academy-taiwan/%E6%8F%90%E7%85%89%E5%86%8D%E6%8F%90%E7%85%89%E6%BF%83%E7%B8%AE%E5%86%8D%E6%BF%83%E7%B8%AE-neural-architecture-search-%E4%BB%8B%E7%B4%B9-ef366ffdc818>

# **Neural Architecture design**神经架構設計- CNN 已經可以達到非常高甚至超越人類的精確度，但隨之而來的代價就是追求更深更大的網路所造成的高複雜度,如大量的記憶體需求，大量的運算硬體設備的挑战。(ChatGPT一次运算就要花费450万美元，2022年，OpenAI公司净亏损总额为5.45亿美元。此外，ChatGPT测算生成一条信息的成本在1.3美分左右，是目前传统搜索引擎的3到4倍，单次搜索成本过于高昂。)

<https://www.bnext.com.tw/article/60440/ai-strategy-04>

## **Training data**訓練數據- 訓練AI數據模型時，其實有三個要素，彼此互相影響。分別是： **問題的複雜度** 、 **AI模型複雜度** （Model Complexity），以及**數據複雜度** （Data Complexity）。

<https://chtseng.wordpress.com/2019/09/20/transfer-learning-fine-tuning/>

**Fine-tuning procedures微調過程**~ 实例

**Optimizers**- Adam/ SGD/ RMSProp/ SWA/ AdaTune

BERT

Muppets