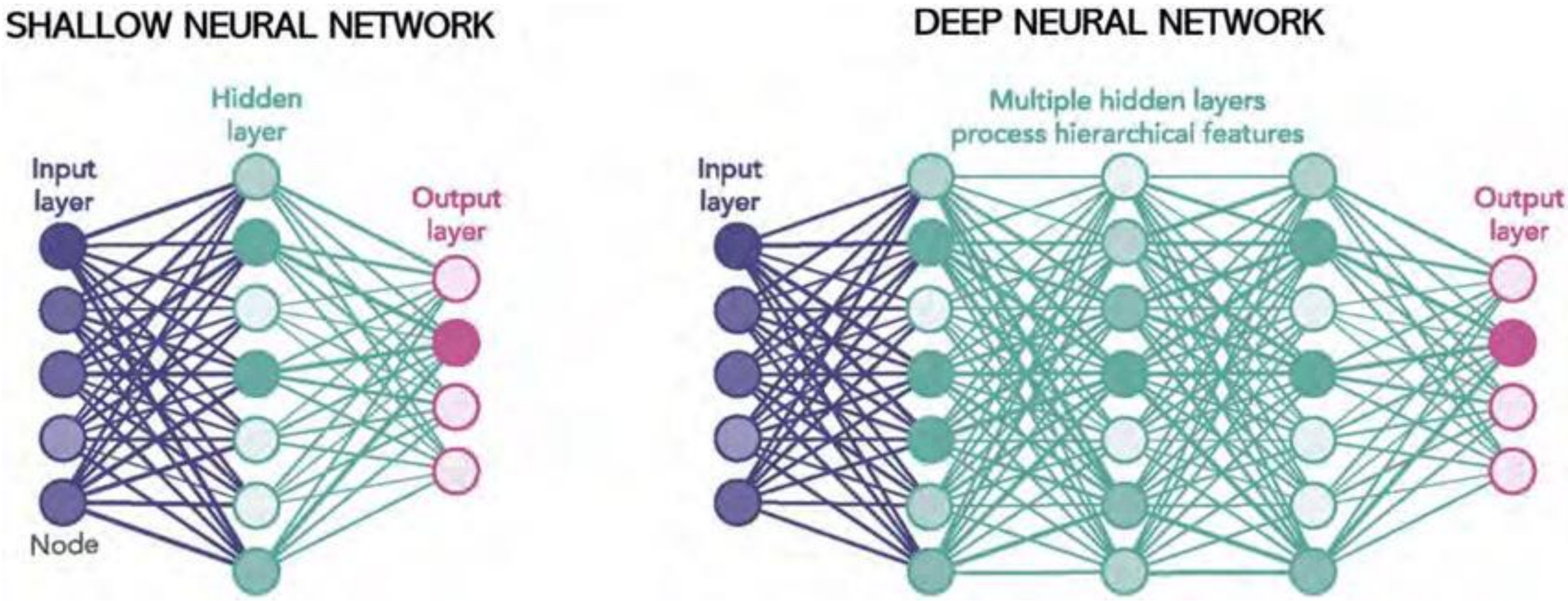
Redes Neuronales y Aprendizaje Profundo

Bloque1: Feedforward Neural Networks

From Shallow to Deep NN



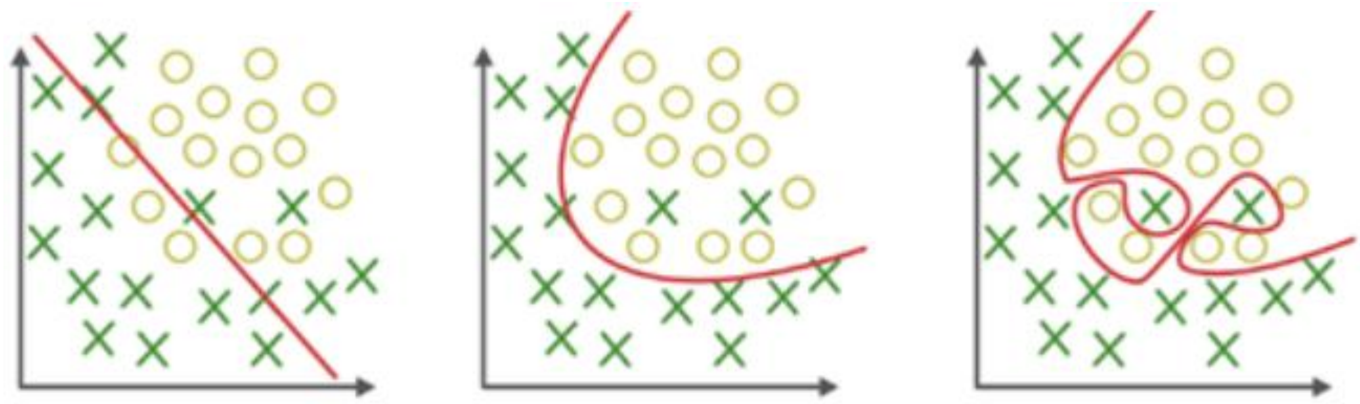
- **Objectives**
  - To analyze the phenomena of **Overfitting and Underfitting** to understand how model complexity affects generalization performance.
  - To investigate regularization techniques for preventing poor generalization.
  - To evaluate the role of **Parameter Initialization** in maintaining gradient flow and preventing numerical instability in deep architectures

- **Objectives**
  - To understand **Early Stopping** to strategically halt the optimization process at the point of maximum generalization
  - To explore **Quantization** as a technique to map high-precision parameters to discrete spaces for efficient edge-device inference.

- **Contents**
  - Overfitting and Underfitting
  - Regularization techniques
  - Parameter initialization
  - Early Stopping
  - Quantization

Overfitting and Underfitting

- **Gradient Descent. Epochs**
  - **Epochs**
  - In training a neural network if too many epochs are used the model may have “overfitting” with the training data set.



under-fitting **appropiate-fitting** **over-fitting**

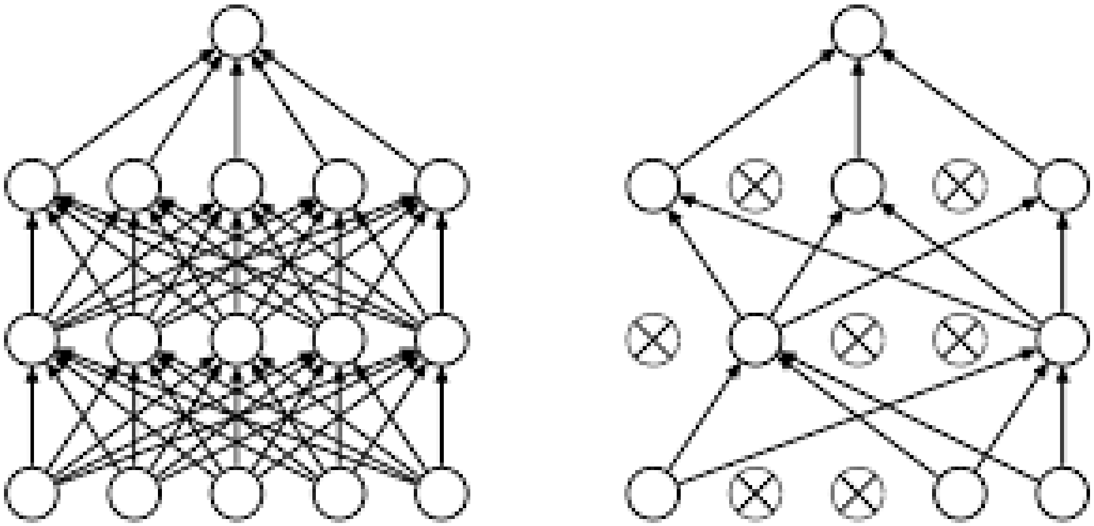
Overfitting and Underfitting

- **Overfitting and Underfitting**
  - **Overfitting:** Learns training data too well but has poor generalization in inference.
  - **Underfitting:** Does not learn training data and misses characteristics and patterns of the data, leading to poor performance in both training and test datasets.
  - Possible solutions:

- K-fold cross-validation
- Data augmentation
- Model selection
- Hyperparameters selection

Regulatization techniques

- ## Dropout
  - Dropout is a **regularization technique** used in neural networks **to prevent overfitting**.
  - It works by **randomly deactivating** (or "**dropping out**") a **fraction of neurons during training**, forcing the model to learn more robust and generalizable features.



(a) Standard Neural Net (b) After applying dropout.

- ## Weight Decay
  - Weight Decay penalizes the sum of squared weights.
  - Prevents weights from **becoming too large** and leads to more **stable models**.

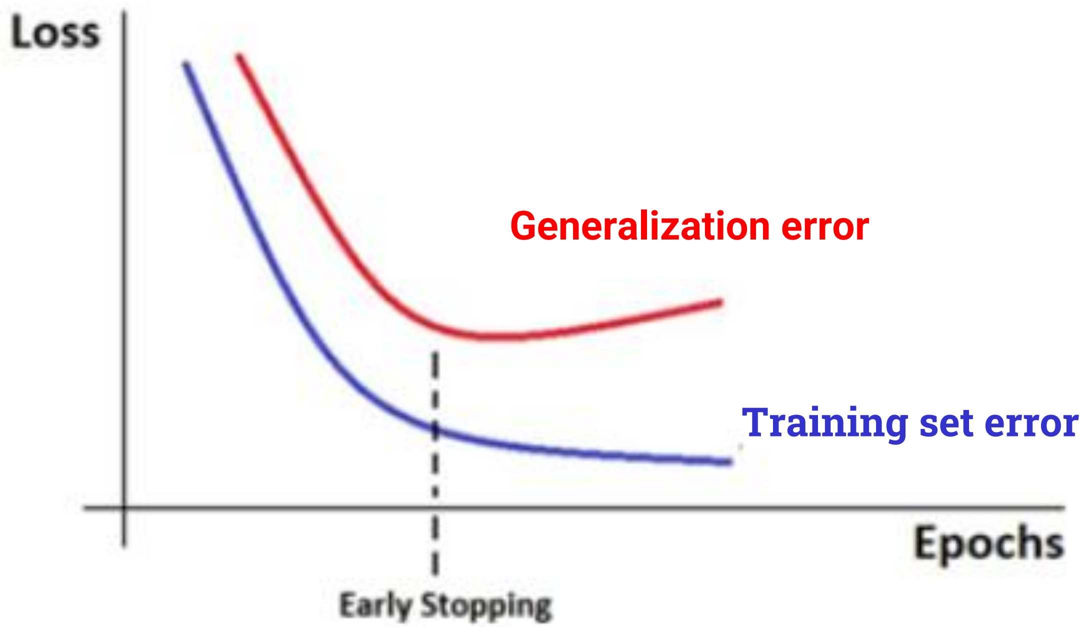
$$\boldsymbol{w} = (\boldsymbol{1} - \lambda)\boldsymbol{w} - \boldsymbol{\alpha}\Delta \boldsymbol{C_0}$$

  - Useful if we do not want the model to focus on local noise of new inputs. Force the network to learn only relevant features in the training set.

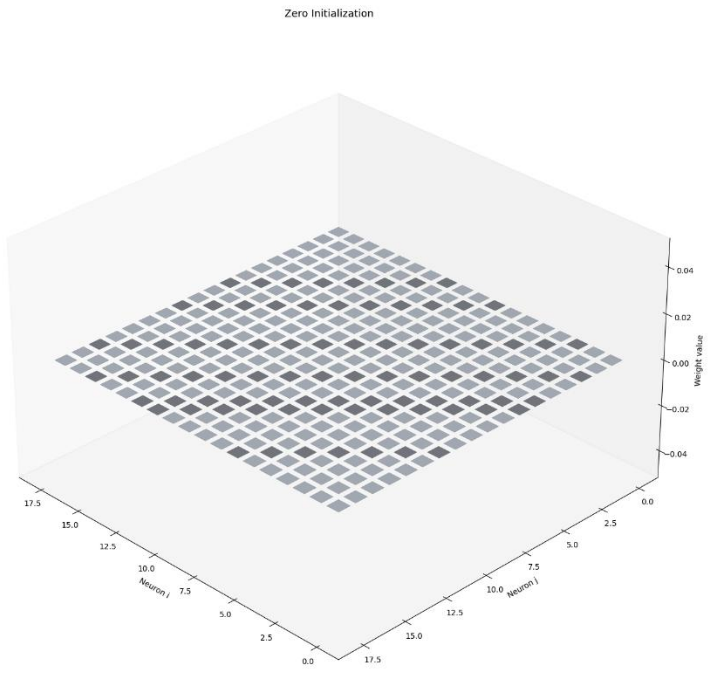
Regularization techniques

- **Data Augmentation**
  - Generates **additional** examples and reduces overfitting **increasing the variability** of the dataset
  - In small tabular datasets we can apply data augmentation by generating samples with the same distribution or adding noise.

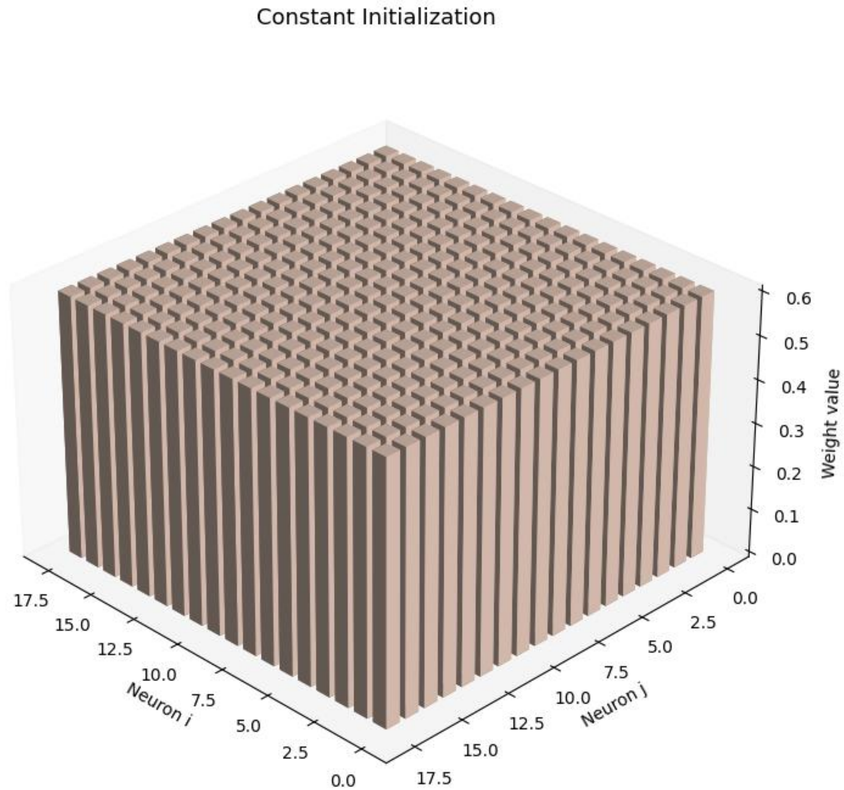
- **Early stopping**
  - A very high number of epochs can reduce the training error but increase the generalization error.



- **Zero initialization**
  - **Zero initialization** initializes weights with 0. This produces that the all the neurons of the NN learns the same.

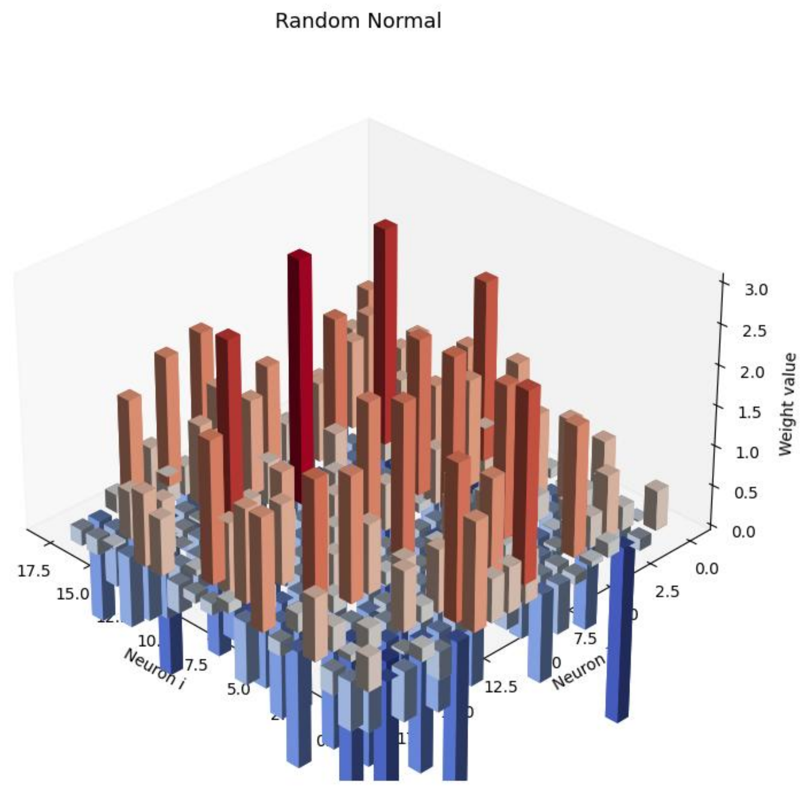


- **Constant initialization**
  - **Constant initialization** initializes weights with the same value. Has the same problem of Zero initialization.

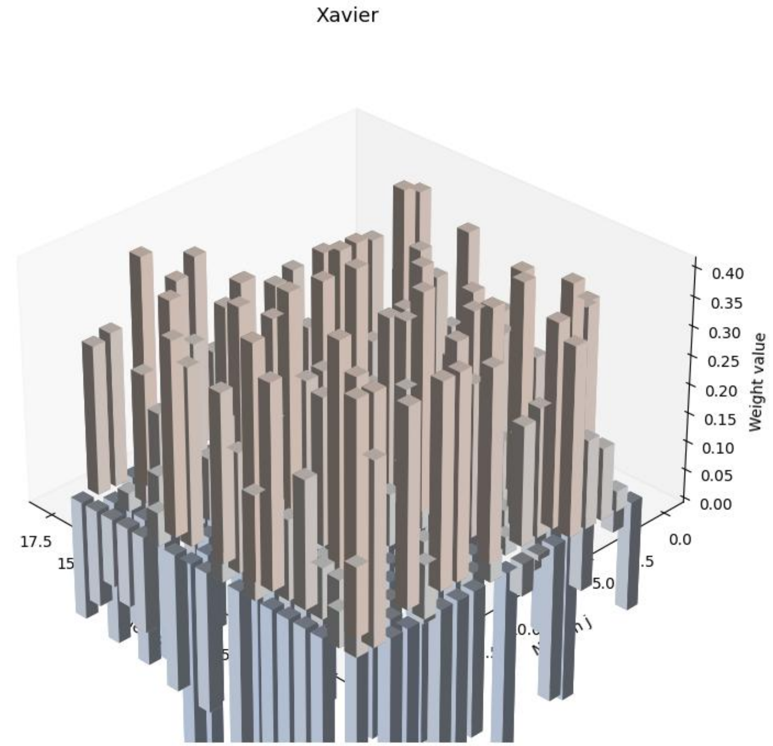


Parameter Initialization

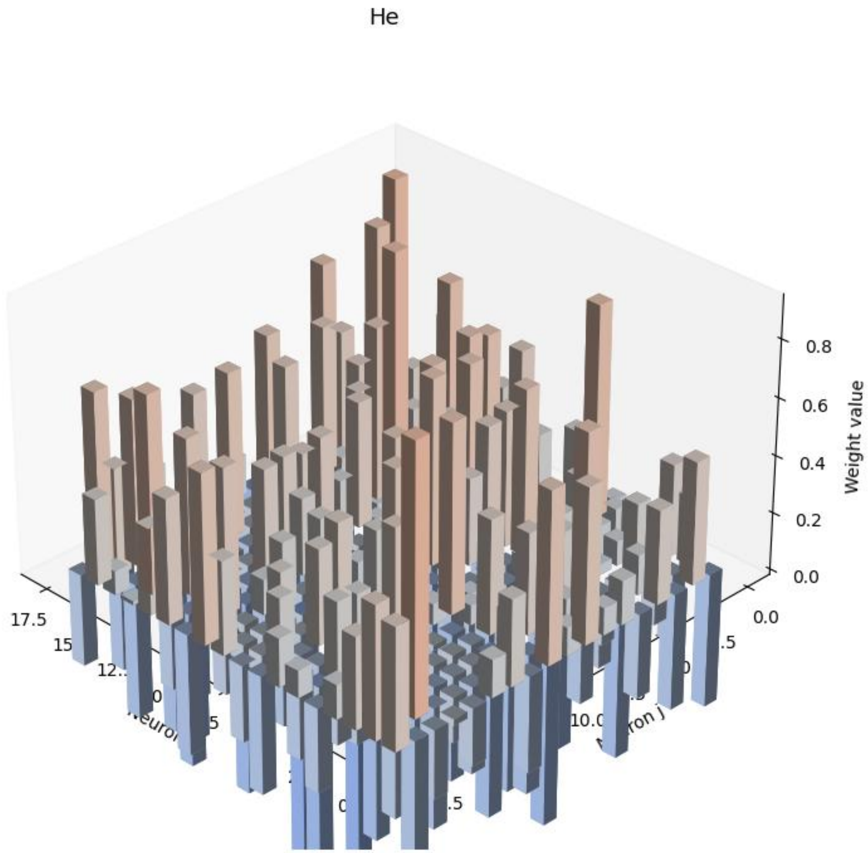
- **Random normal initialization**
  - **Random normal (Gaussian) initialization** initializes weights with the same value. Has the same problem of Zero initialization.



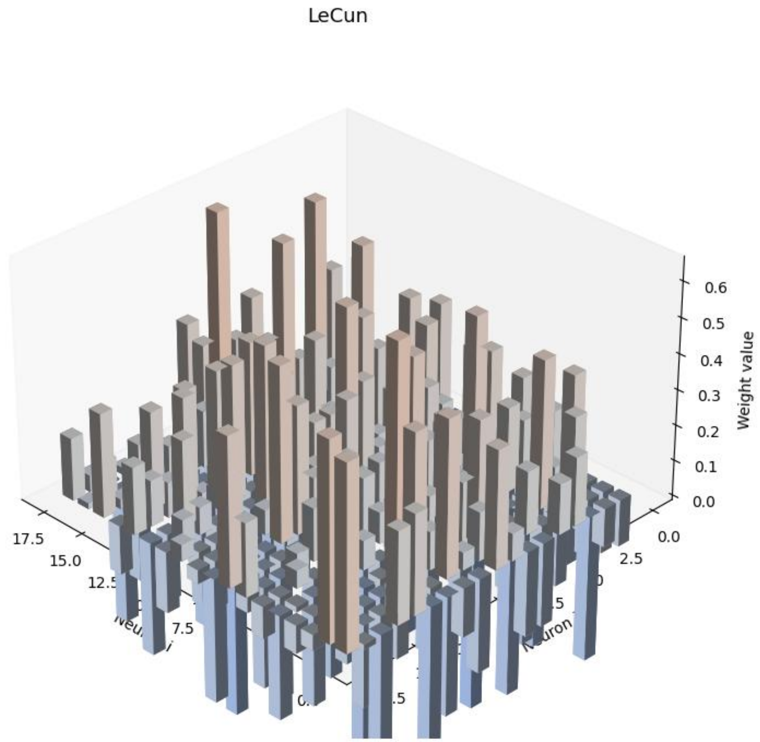
- **Xavier initialization**
  - One of the main objectives of **Xavier's initialization method** (also known as **Glorot initialization**) in a neural network is to ensure that the gradients during training are neither too large nor too small.



- **He initialization**
  - **He initialization (Normal)** is optimized for layers that use ReLU activation functions. Same variance for both inputs and outputs in NN.



- **Lecun initialization**
  - **Lecun initialization (Normal)** is optimized for layers that use SELU/sigmoid/tanh activation functions. Same variance for both inputs and outputs in NN.



Quantization

- ## Quantization
  - **Quantization** is a technique that **reduces** the precision of weights and activations. For instance, from Float32 to smaller formats such as INT8.
  - Some studies achieved good results although a quantization of **1.58 bits**.
  - Remember, 8 bits = 1 byte, FP32= 4 bytes = 32 bits. 32x times cheaper.

The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits

Shuming Ma Hongyu Wang‡ Lingxiao Ma Lei Wang Wenhui Wang

Shaohan Huang Li Dong Ruiping Wang Jilong Xue Furu Wei⋄

https://aka.ms/GeneralAI

Equal contribution. ⋄ Corresponding author. S. Ma, L. Ma, L. Wang, W. Wang, S. Huang, L. Dong, J. Xue, F. Wei are with Microsoft Research. H. Wang and R. Wang are with University of Chinese Academy of Sciences.

Quantization

- **Quantization Types**
  - **Post-Training Quantization (PTQ)** quantizes a trained model.
  - **Quantization-Aware Training (QAT)** simulates quantization during training to reduce accuracy loss.
  - In **Weight Quantization** only weights are reduced in precision.
  - In **Activation Quantization** are reduced in precision.
  - **Bias Quantization** quantizes biases (less common).

Quantization

- **Exercise 1. Symmetric Uniform Quantization**

  - **Domain**

    $n = 8$, **number of bits for the representation**

    $q \in Z$, **quantized values are integers**

    $$\boxed{q_{min} = -2^{n-1} \qquad q_{max} = 2^{n-1} - 1}$$

  - **Scale**

    $\alpha =$ max(**abs(x)), max value in original data**

    $s = \dfrac{\alpha}{q_{max}}$, **scale factor that maps x**

  - **Input = x = [1.23, - 0.87]**

  - **Quantization**

    $q = round(\dfrac{x}{s})$, **rounds to nearest integer**