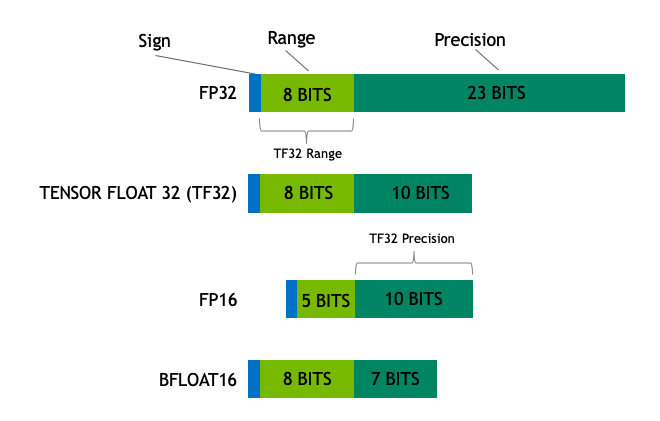
**QLoRa & PEFT**

**Floating Points Structure:**  
  
Sign (1 bit): This bit indicates the sign of the number. If the sign bit is 0, the number is positive; if it is 1, the number is negative. This simple binary distinction allows for the representation of both positive and negative numbers.

Exponent (8 bits): The exponent is used to determine the scale of the number. It's stored in a biased form, meaning that a certain value (in FP32, it's 127) is subtracted from the actual exponent to get the stored exponent. This allows for the representation of both very large and very small numbers. The exponent essentially dictates how far and in which direction (positive for large numbers, negative for small numbers) to move the decimal point in the mantissa.

Mantissa (or significand) (23 bits): The mantissa holds the significant digits of the number. It's essentially the "detail" part of the number, representing the actual digits without regard to magnitude. In FP32, it is normalized, meaning that it's always adjusted to be in a specific range (usually 1 ≤ mantissa < 2 for normalized numbers, but 0 ≤ mantissa < 1 for subnormal numbers). The leading 1 (before the decimal point in normalized numbers) is implicit and not stored in the 23 bits.



**PEFT**

🤗 PEFT (Parameter-Efficient Fine-Tuning) is a library for efficiently adapting large pretrained models to various downstream applications without fine-tuning all of a model’s parameters because it is prohibitively costly. PEFT methods only fine-tune a small number of (extra) model parameters - significantly decreasing computational and storage costs - while yielding performance comparable to a fully fine-tuned model. This makes it more accessible to train and store large language models (LLMs) on consumer hardware.

**QLoRA**

QLoRA uses 4-bit quantization to compress a pretrained language model. The LM parameters are then frozen and a relatively small number of trainable parameters are added to the model in the form of Low-Rank Adapters. During finetuning, QLoRA backpropagates gradients through the frozen 4-bit quantized pretrained language model into the Low-Rank Adapters. The LoRA layers are the only parameters being updated during training. Read more about LoRA in the original LoRA paper.

**bitsandbytes.functional:** Contains quantization functions and stateless 8-bit optimizer update functions. bitsandbytes.nn.modules: Contains stable embedding layer with automatic 32-bit optimizer overrides (important for NLP stability) bitsandbytes.optim: Contains 8-bit optimizers.

bnb\_config = BitsAndBytesConfig(

    load\_in\_4bit=True,

    bnb\_4bit\_use\_double\_quant=True,

    bnb\_4bit\_quant\_type="nf4",

    bnb\_4bit\_compute\_dtype=torch.bfloat16,

    device\_map = "auto"

)

**Lora Configuration** 🡪 Decides the size of the matrix that will be used to fine tune

https://huggingface.co/blog/4bit-transformers-bitsandbytes