Analytics Vidhya

LoRA and its variants

Instructor

Sourab Mangrulkar

Machine Learning Engineer at 🤗,
Creator of 🤗 PEFT

(Low-Rank Adaptation) LoRA Overview

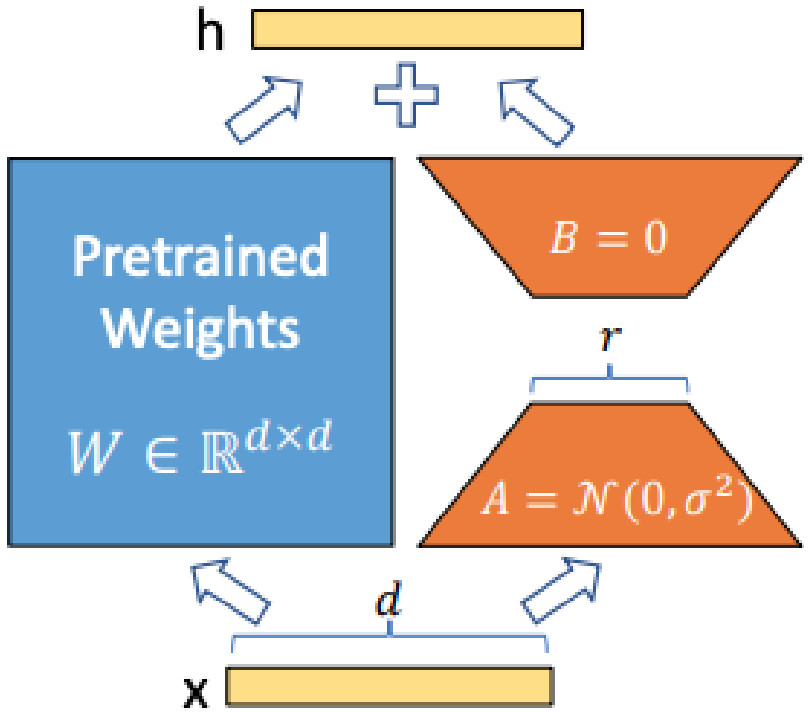

FIGURE EXPLAINING HOW LORA WORKS FROM ORIGINAL PAPER, FIGURE 1

LoRA Overview

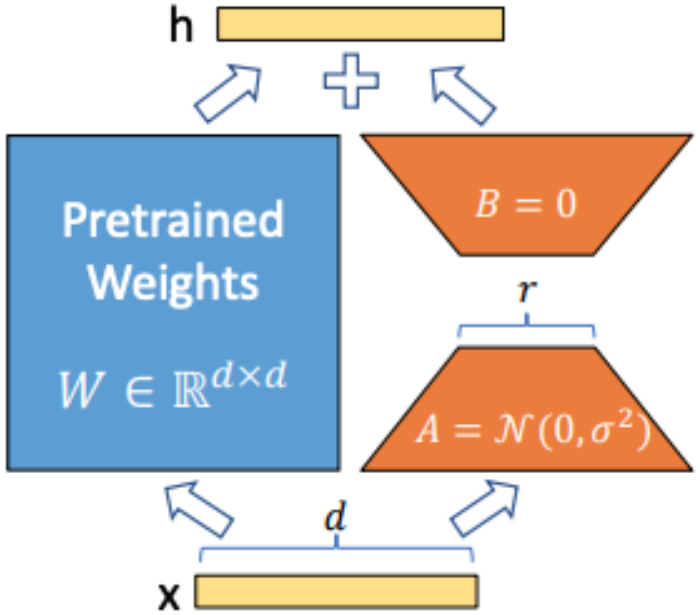

FIGURE EXPLAINING HOW LORA WORKS FROM ORIGINAL PAPER, FIGURE 1

```python
class LoraLayer(torch.nn.Module):
    def __init__(self, base_layer, r, alpha, f_in, f_out):
        super().__init__()
        self.base_layer = base_layer
        self.scaling = alpha/r
        self.lora_A = torch.nn.Linear(f_in, r, bias=False)
        self.lora_B = torch.nn.Linear(r, f_out, bias=False)

    def forward(self, x, *args, **kwargs):
        lora_output = self.lora_B(self.lora_A(x)) * self.scaling
        return self.base_layer(x, *args, **kwargs) + lora_output
```

LoRA Overview

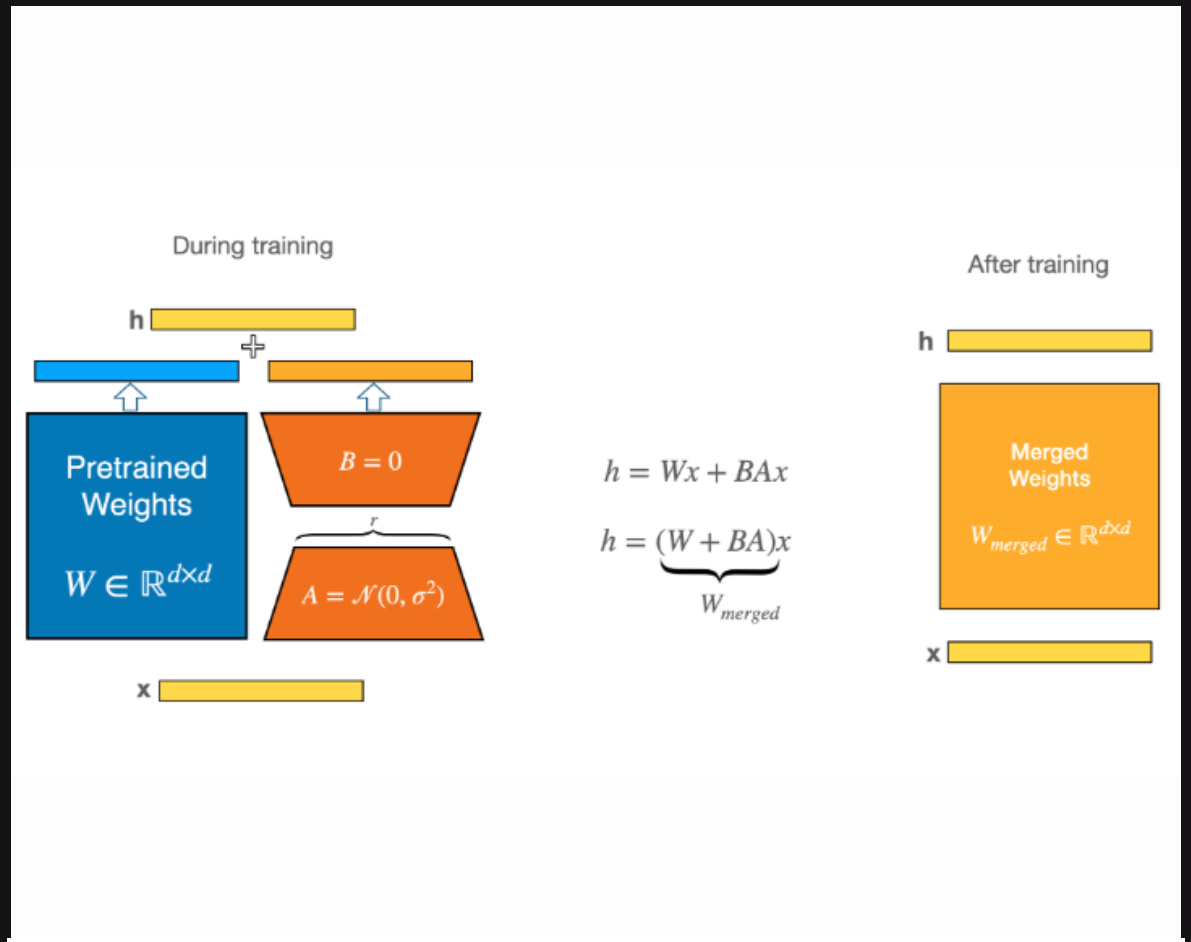

FIGURE ADAPTED FROM ORIGINAL PAPER, FIGURE 1 AND SECTION 4.1

No Additional Inference Latency

Step1: Train adapters adapted on your task

Step 2: Merge the adapter weights inside the base model and use it as a standalone model

Analytics Vidhya

LoHA, LoKR, AdaLoRA

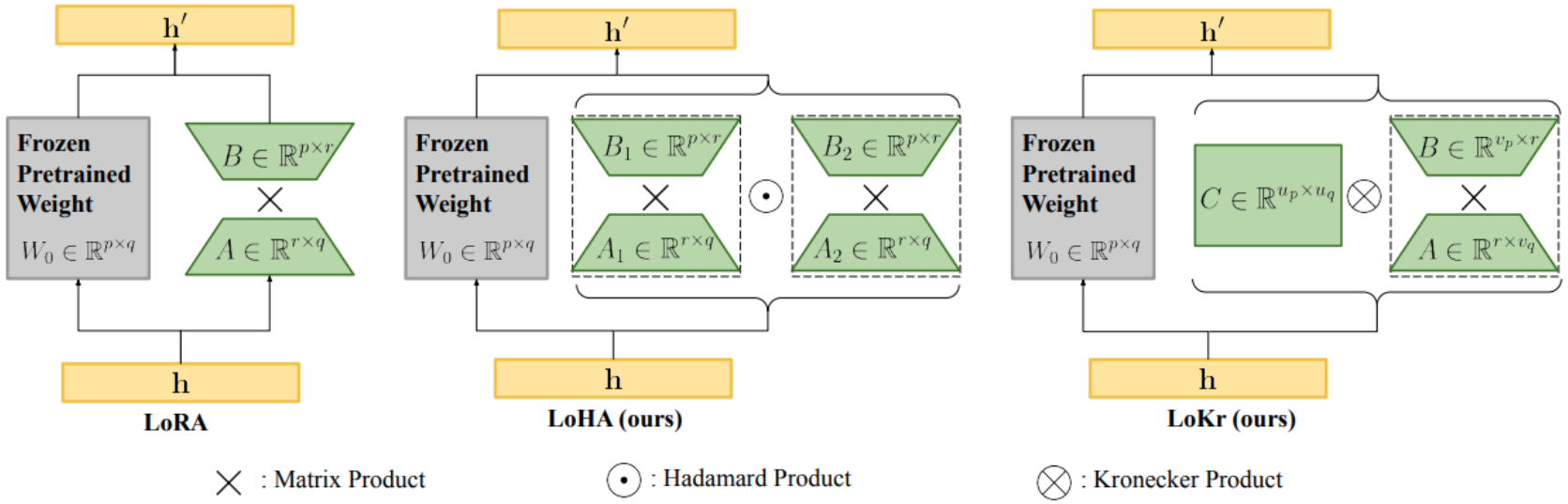

FIGURE 1 FROM ORIGINAL PAPER

LoRA variants

1. LoHA: LoRA with Hadamard Product.
2. LoKr: LoRA with Kronecker Product.
3. AdaLoRA: Adaptive budget allocation such that important layers having higher rank (more params) while pruning less important ones.

Analytics Vidhya

LoRA Finetuning cost

Finetuning Mistral-7B in mixed-precision using Adam Optimizer.
trainable: 21,549,136 || all params: 7,263,322,192 || trainable%: 0.296

Weights - 2 bytes / parameter

Gradients - 2 bytes / parameter

Optimizer state - 4 bytes / parameter (FP32 copy) + 8 bytes / parameter (momentum & variance estimates)

Total training cost: 16 bytes/parameter * 7 billion parameters * 0.0029 + 14 = 112 * 0.00296 + 14 GB ~ 14.4 GB