Analytics Vidhya

$(IA)^3$ and Bottleneck Adapters

Instructor

Sourab Mangrulkar

Machine Learning Engineer at 🤗,
Creator of 🤗 PEFT

Infused Adapter by Inhibiting and Amplyfing Inner Activations $(IA)^3$ Overview

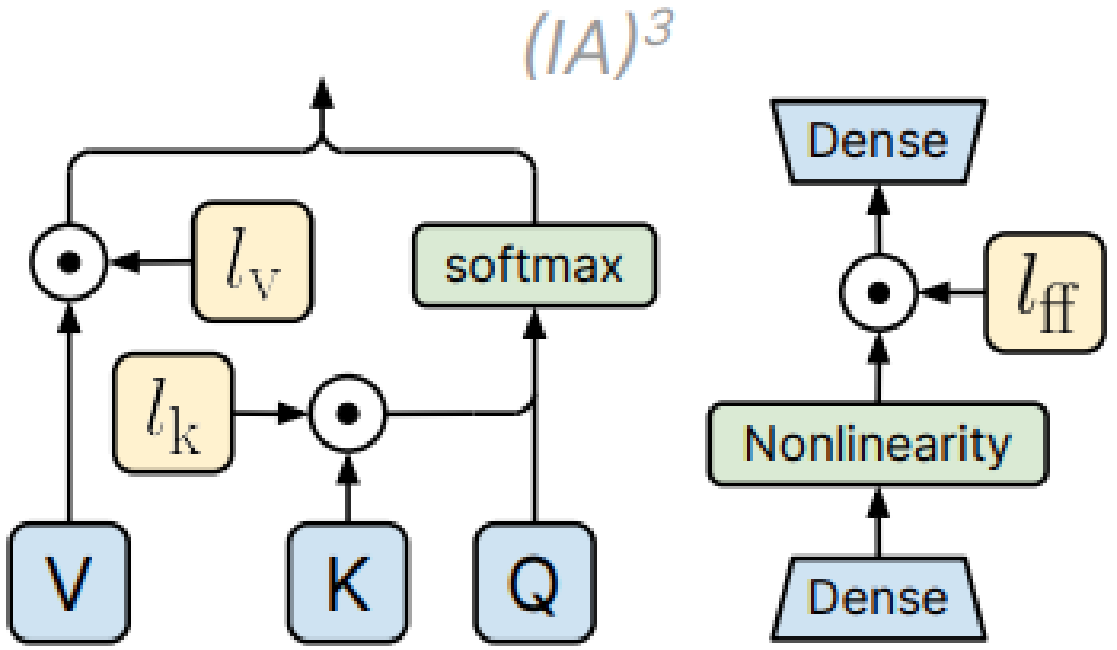

FIGURE EXPLAINING HOW $(IA)^3$ WORKS FROM ORIGINAL PAPER, FIGURE 1

$(IA)^3$ Overview

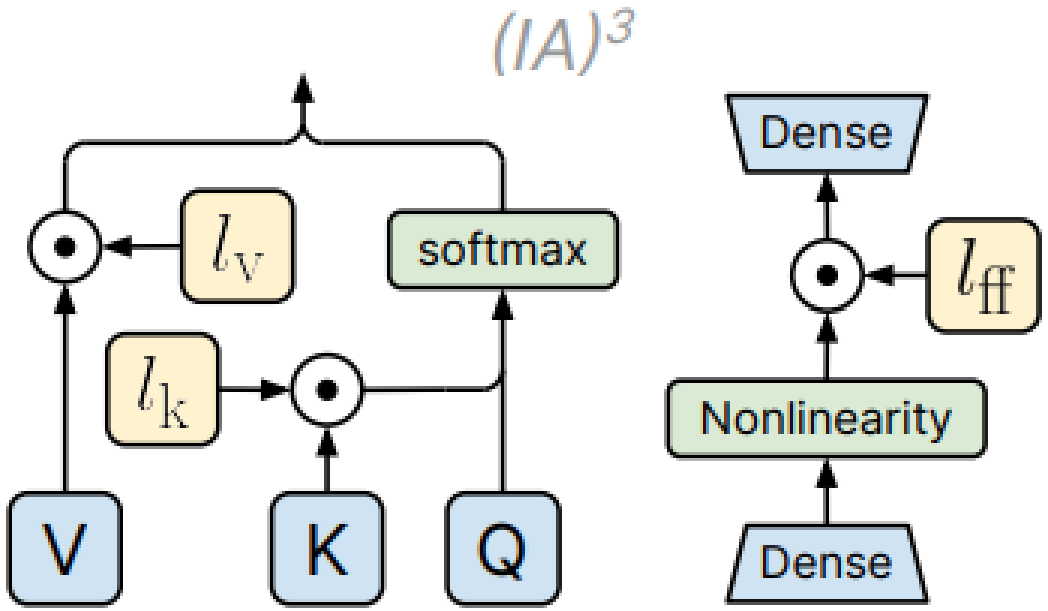

FIGURE EXPLAINING HOW $(IA)^3$ WORKS FROM ORIGINAL PAPER, FIGURE 1

```python
class IA3Layer(torch.nn.Module):
    def __init__(self, base_layer, transform_input, f_in, f_out):
        super().__init__()
        self.base_layer = base_layer
        self.transform_input = transform_input
        d = f_in if transform_input else f_out
        self.ia3 = torch.nn.Parameter(d)

    def forward(self, x, *args, **kwargs):
        if self.transform_input:
            x = x * self.ia3
            output = self.base_layer(x, *args, **kwargs)
        else:
            output = self.base_layer(x, *args, **kwargs)
            output = output * self.ia3
        return output
```

Analytics Vidhya

Bottleneck Adapters Overview

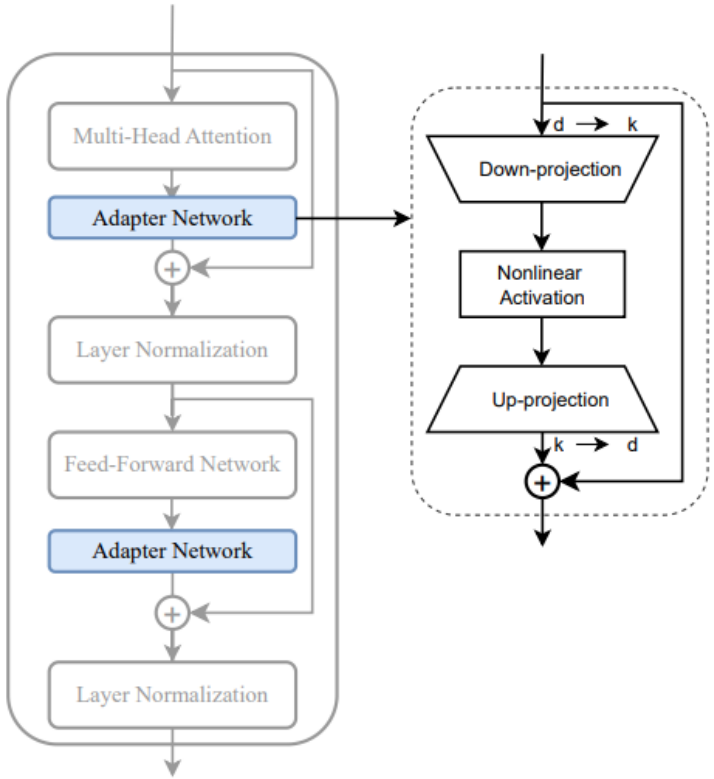

FIGURE EXPLAINING HOW SEQUENTIAL BOTTLENECK ADAPTERS WORKS FROM SURVEY PAPER, FIGURE 3

Bottleneck Adapters Overview

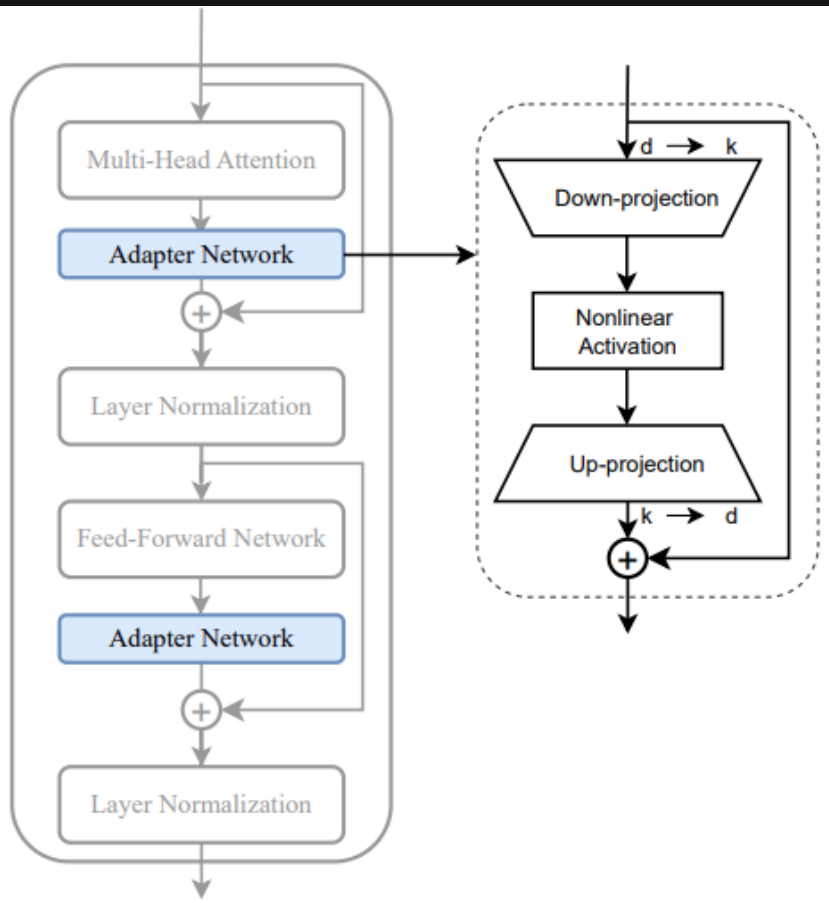

FIGURE EXPLAINING HOW SEQUENTIAL BOTTLENECK ADAPTERS WORKS FROM SURVEY PAPER, FIGURE 3

```python
class BottleneckAdapterLayer(torch.nn.Module):
    def __init__(self, k, f_in, f_out, act_fn):
        super().__init__()
        self.down_proj = torch.nn.Linear(f_in, k)
        self.up_proj = torch.nn.Linear(k, f_out)
        self.act_fn = act_fn

    def forward(self, x, *args, **kwargs):
        return self.up_proj(self.act_fn(self.down_proj(x)))
```