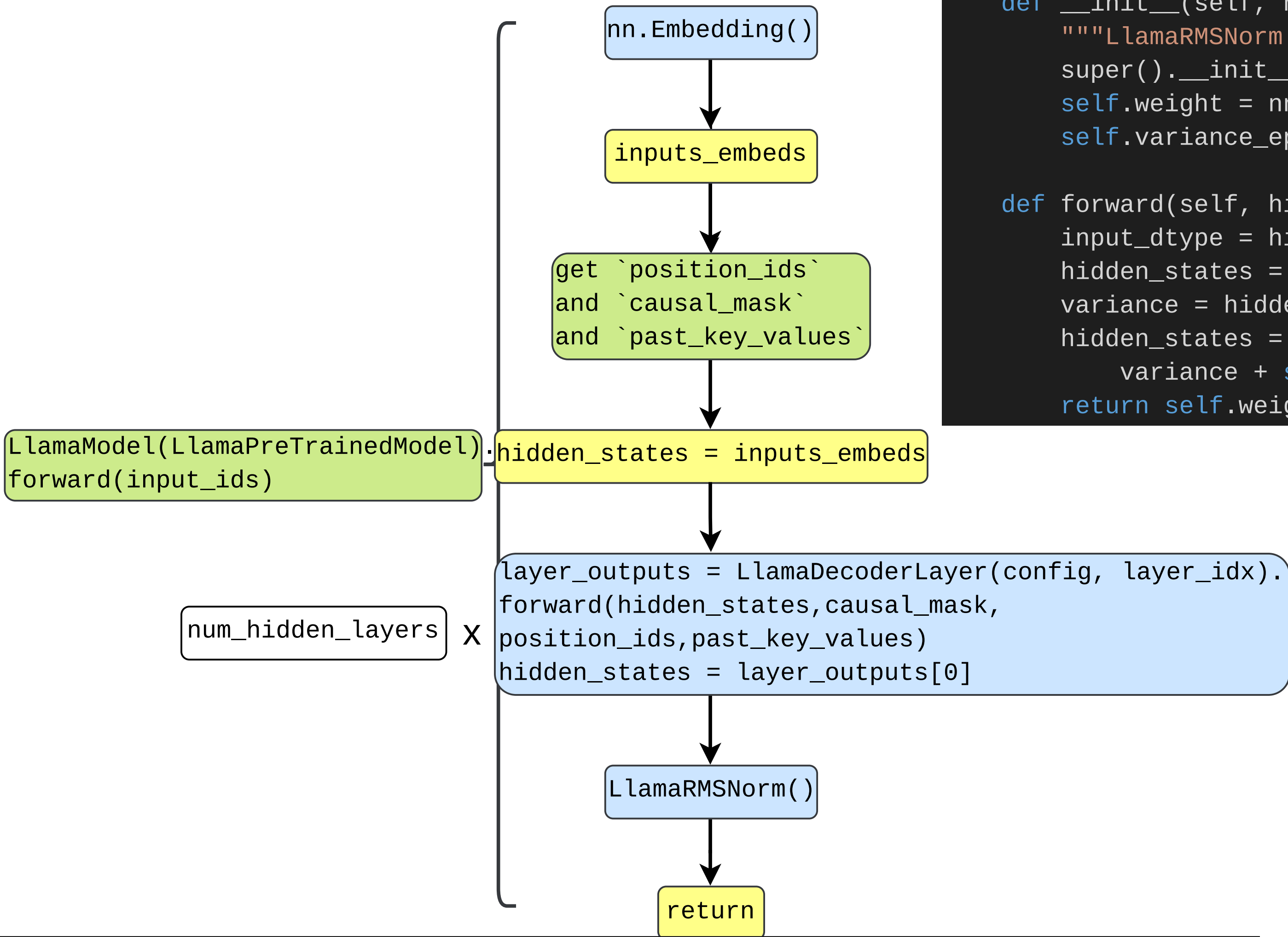
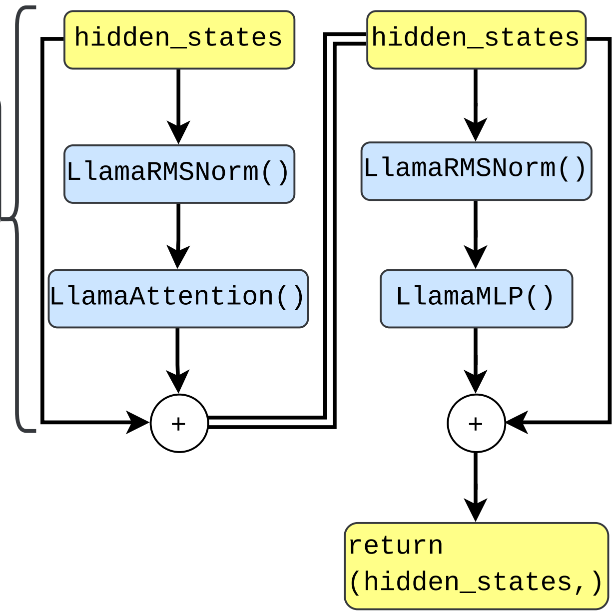
```python
class LlamaRMSNorm(nn.Module):
    def __init__(self, hidden_size, eps=1e-6):
        """LlamaRMSNorm is equivalent to T5LayerNorm"""
        super().__init__()
        self.weight = nn.Parameter(torch.ones(hidden_size))
        self.variance_epsilon = eps

    def forward(self, hidden_states):
        input_dtype = hidden_states.dtype
        hidden_states = hidden_states.to(torch.float32)
        variance = hidden_states.pow(2).mean(-1, keepdim=True)
        hidden_states = hidden_states * torch.rsqrt(
            variance + self.variance_epsilon)
        return self.weight * hidden_states.to(input_dtype)
```

$$a = \frac{a}{\sqrt{\frac{\sum a_i^2}{n} + \epsilon}} w$$






```python
class LlamaMLP(nn.Module):
    def __init__(self, config):
        super().__init__()
        self.config = config
        self.hidden_size = config.hidden_size
        self.intermediate_size = config.intermediate_size
        self.gate_proj = nn.Linear(self.hidden_size, self.intermediate_size, bias=False)
        self.up_proj = nn.Linear(self.hidden_size, self.intermediate_size, bias=False)
        self.down_proj = nn.Linear(self.intermediate_size, self.hidden_size, bias=False)
        self.act_fn = ACT2FN[config.hidden_act]

    def forward(self, x):
        down_proj = self.down_proj(self.act_fn(self.gate_proj(x)) * self.up_proj(x))
        return down_proj
```

$$W_{down}(f(W_{gate}x) \otimes W_{up}x)$$

$$\mathrm{FFN}_{\mathrm{GLU}}(x, W, V, W_2) = (\sigma(xW) \otimes xV)W_2$$
$$\mathrm{FFN}_{\mathrm{Bilinear}}(x, W, V, W_2) = (xW \otimes xV)W_2$$
$$\mathrm{FFN}_{\mathrm{ReGLU}}(x, W, V, W_2) = (\max(0, xW) \otimes xV)W_2$$
$$\mathrm{FFN}_{\mathrm{GEGLU}}(x, W, V, W_2) = (\mathrm{GELU}(xW) \otimes xV)W_2$$
$$\mathrm{FFN}_{\mathrm{SwiGLU}}(x, W, V, W_2) = (\mathrm{Swish}_1(xW) \otimes xV)W_2$$