

1/8

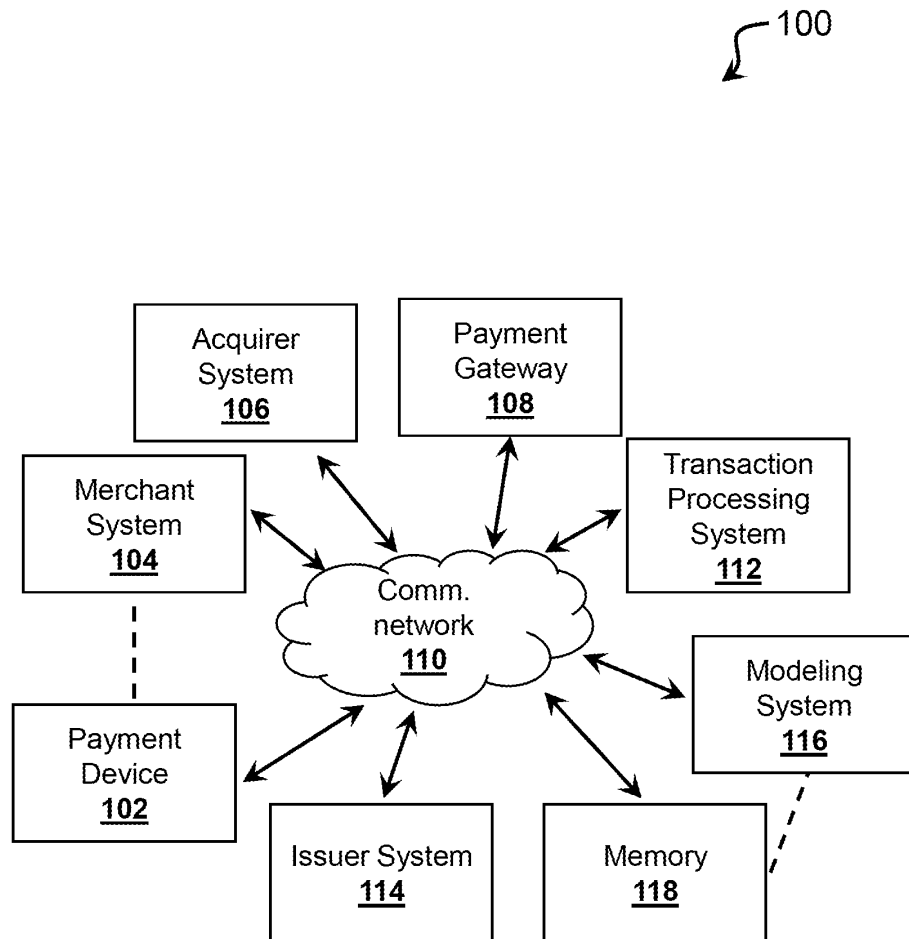


FIG. 1

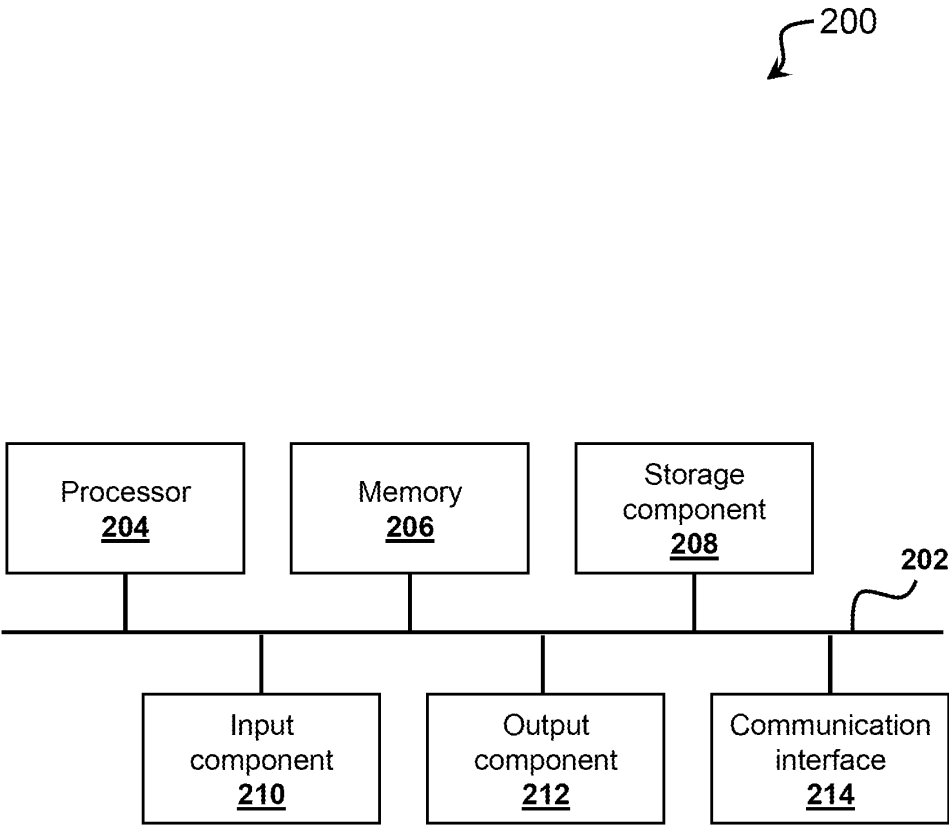


FIG. 2

3/8

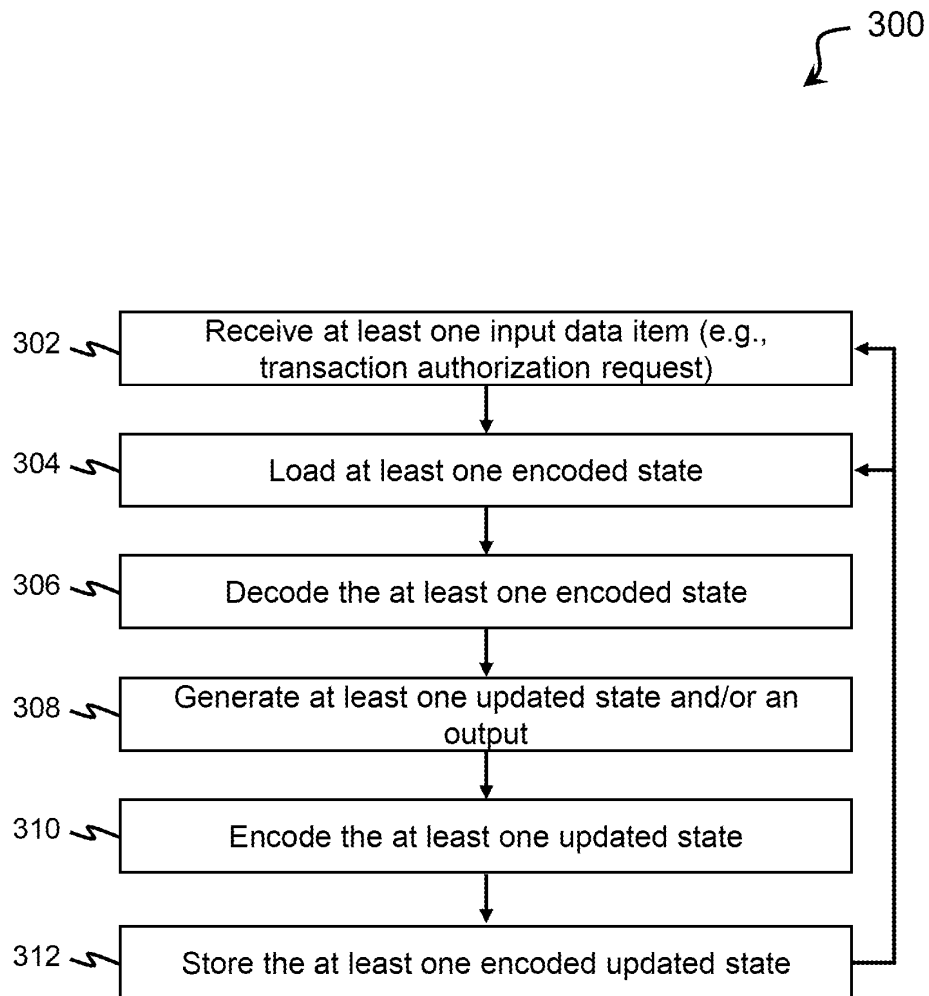


FIG. 3

4/8

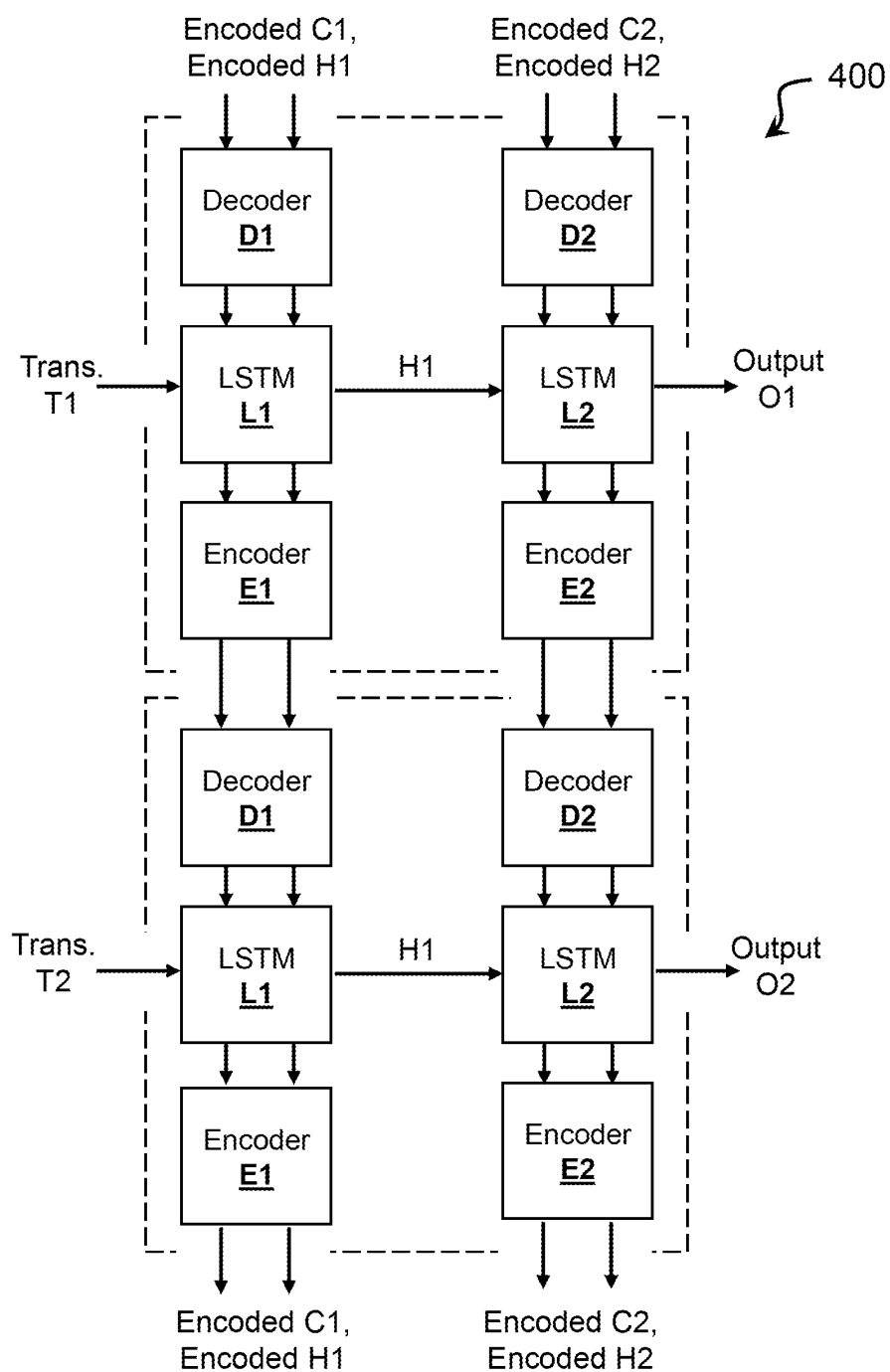


FIG. 4

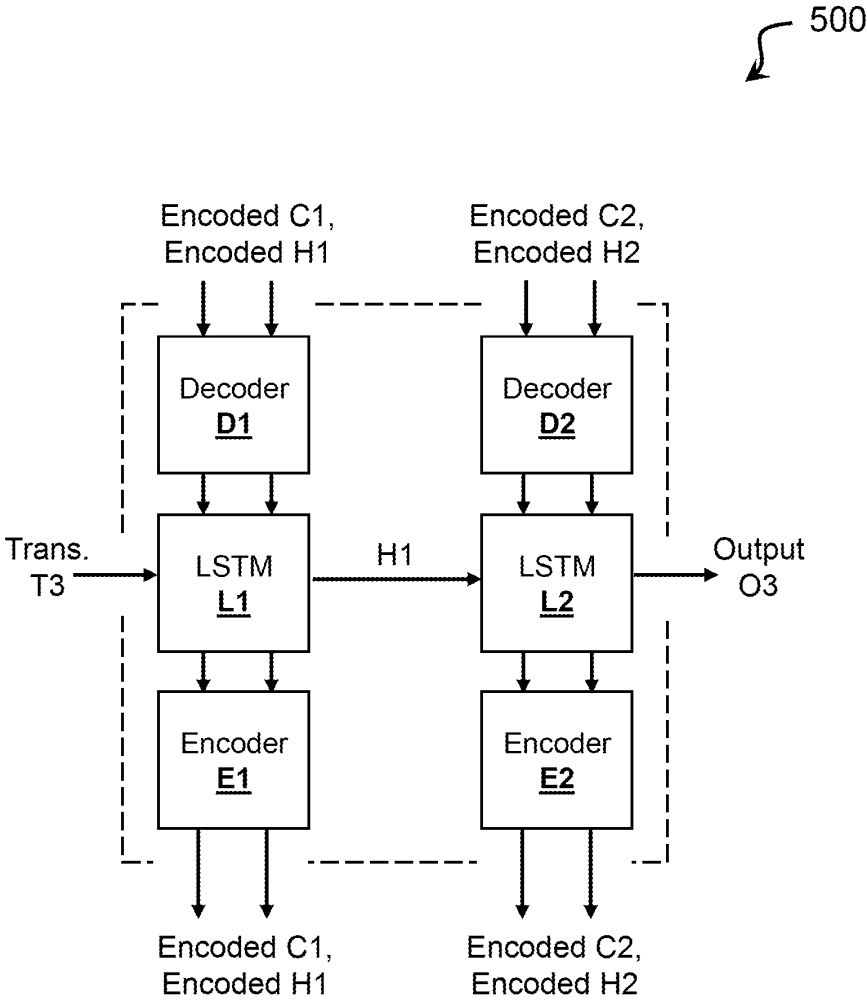


FIG. 5

6/8

```
weights_ec = Variable (d1, d2)
biases_ec = Variable (d2)
function EC(x):
    return sigmoid(add(matmul(x, weights_ec), biases_ec))
end function

weights_dc = Variable (d2, d1)
biases_dc = Variable (d1)
function DC(x):
    return sigmoid(add(matmul(x, weights_dc), biases_dc))
end function

weights_eh = Variable (d1, d2)
biases_eh = Variable (d2)
function EH(x):
    return sigmoid(add(matmul(x, weights_eh), biases_eh))
end function

weights_dh = Variable (d2, d1)
biases_dh = Variable (d1)
function DH(x):
    return sigmoid(add(matmul(x, weights_dh), biases_dh))
end function
```

FIG. 6

7/8

```
While t < T:
    inputs = X[t]
    recovered_states = (DC(enC), DH(enH))
    outputs, orig_states = LSTM_layer(inputs, recovered_states)
    append outputs into output_ta
    use outputs to calculate loss and optimize DC, EC, DH, EH, and LSTM_Layer
    C, H = orig_states
    enC = EC(C)
    enH = EH(H)
    t = t + 1
```

FIG. 7

8/8

```
If X has previous stored states in memory:
    Load enC and enH from memory
else:
    Initialize enC and enH states with zero states having dimension d2
    recovered_states = (DC(enC), DH(enH))
    inputs = X[0]
    outputs, orig_states = LSTM_layer(inputs, recovered_states)
    C, H = orig_states
    enC = EC(C)
    enH = EH(H)
store enC and enH into memory.
```

FIG. 8