

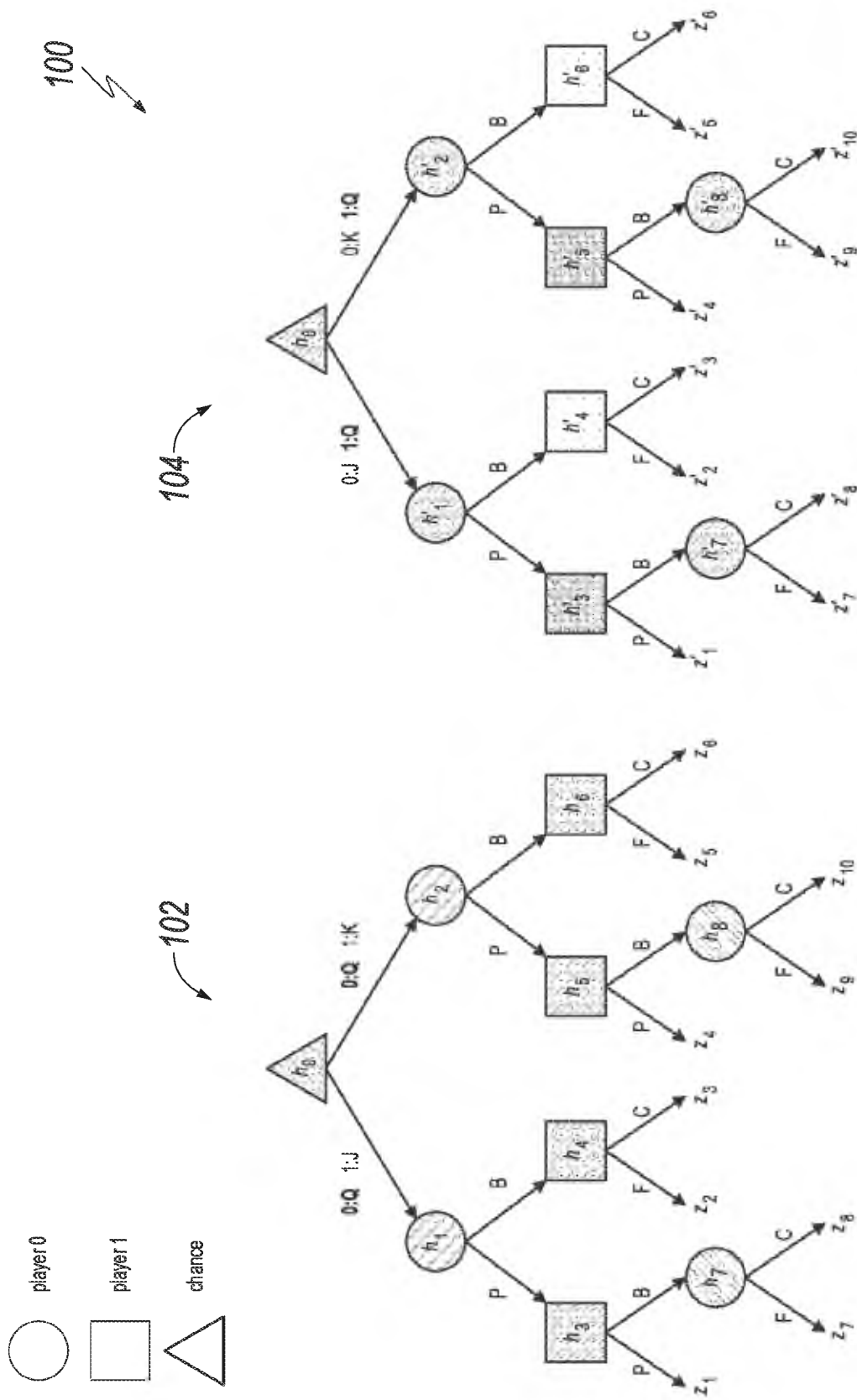


FIG. 1

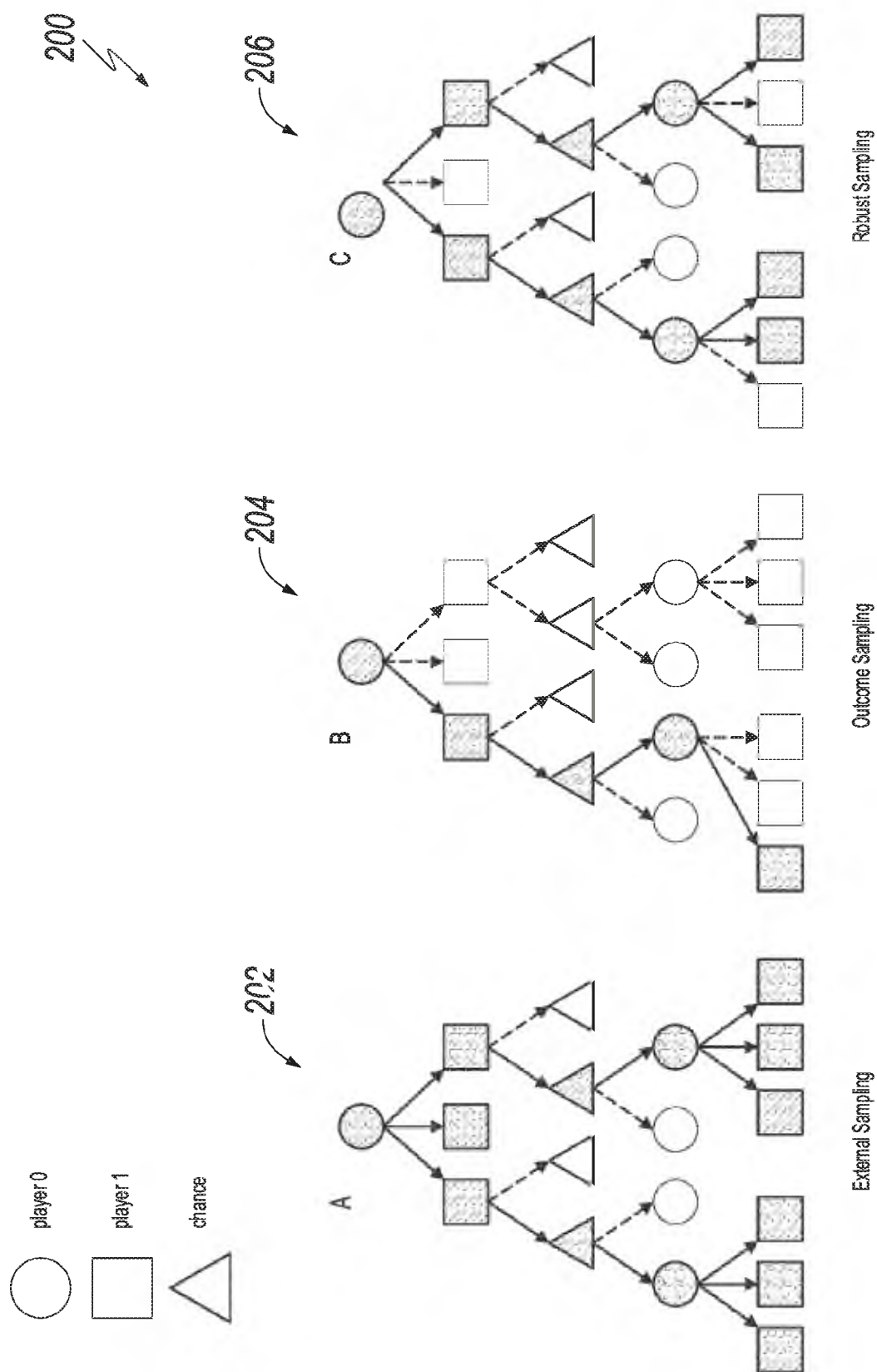


FIG. 2

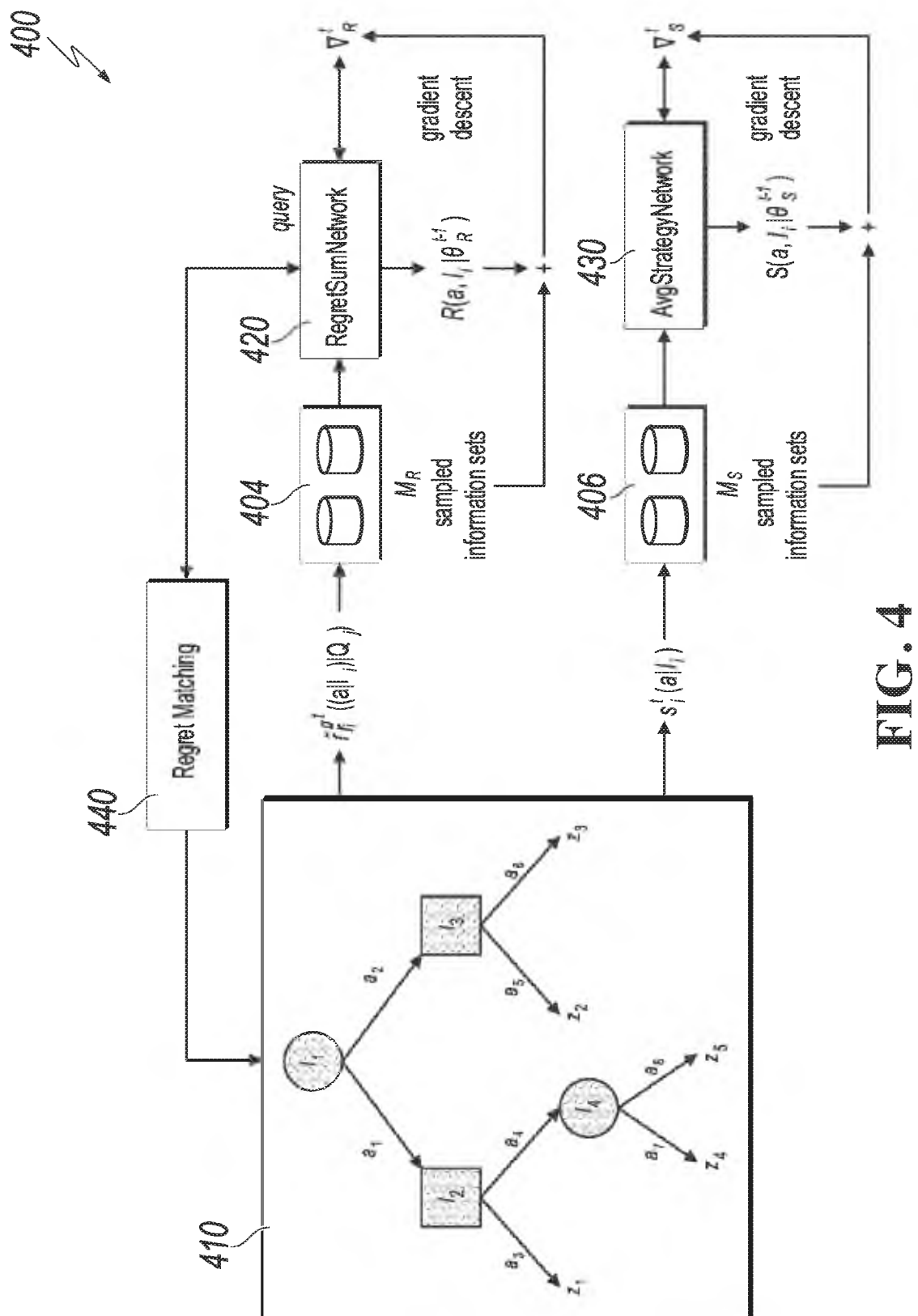
Algorithm: Robust Sampling Monte Carlo Counterfactual Regret Minimization

```

1 Function RS-MCCFR-ALG ( t ):
2   for i = 1 to t do
3     RS-MCCFR ( t,  $\emptyset$ , 0, 1, 1 )
4     RS-MCCFR ( t,  $\emptyset$ , 1, 1, 1 )
5
6 Function RS-MCCFR ( t, h, i,  $\pi_i$ ,  $\pi_i^{rs(k)}$  ):
7    $I_i \leftarrow I_i(h)$  ▷ information set at state h
8   if  $h \in Z$  then
9     return  $\frac{u_i(h)}{\pi_i^{rs(k)}}$  ▷ return game payoff
10  else if  $P(h) = -1$  then
11     $a \sim \sigma_{-i}(I_i)$  ▷ sample an action from  $\sigma_{-i}(h)$ 
12    return RS-MCCFR( t, ha, i,  $\pi_i$ ,  $\pi_i^{rs(k)}$  )
13  else if  $P(h) = i$  then
14     $\sigma_i(I_i) \leftarrow \text{CalculateStrategy}( R_i^t(\cdot|I_i), I_i )$  ▷ calculate current strategy
15     $v_i(h) \leftarrow 0, r_i(\cdot|I_i) \leftarrow \vec{0}, s_i(\cdot|I_i) \leftarrow \vec{0}$  ▷  $r_i(\cdot|I_i)$  and  $s_i(\cdot|I_i)$  are two vectors over  $A(I_i)$ 
16     $A_i^{rs(k)}(I_i) \leftarrow$  sampling k different actions according to  $\sigma_i^{rs(k)}$ 
17    For  $a \in A_i^{rs(k)}(I_i)$  do
18       $v_i(a|h) \leftarrow \text{RS-MCCFR}( t, ha, i, \pi_i, \sigma_i(a|I_i), \pi_i^{rs(k)}, \sigma_i^{rs(k)}(a|I_i) )$ 
19       $v_i(h) \leftarrow v_i(h) + v_i(a|h)\sigma_i(a|I_i)$  ▷ update counterfactual value
20    For  $a \in A_i^{rs(k)}(I_i)$  do
21      if  $t == 0$  then
22         $R_i^t(\cdot|I_i) \leftarrow R_i^{t-1}(\cdot|I_i) + (v_i(a|h) - v_i(h))$  ▷ update cumulative regret
23         $S_i^t(a|I_i) \leftarrow S_i^{t-1}(a|I_i) + \pi_i^a(I_i)\sigma_i(a|I_i)$  ▷ update average strategy numerator
24      else
25         $R_i^t(\cdot|I_i) \leftarrow (v_i(a|h) - v_i(h))$  ▷ update cumulative regret
26         $S_i^t(a|I_i) \leftarrow \pi_i^a(I_i)\sigma_i(a|I_i)$  ▷ update average strategy numerator
27  else
28     $\sigma_{-i}(I_i) \leftarrow \text{CalculateStrategy}( R_{-i}^t(\cdot|I_i), I_i )$  ▷ calculate current strategy
29     $a \sim \sigma_{-i}(I_i)$  ▷ sample an action from  $\sigma_{-i}(I_i)$ 
30    return RS-MCCFR( t, ha, i,  $\pi_i$ ,  $\pi_i^{rs(k)}$  )
31
32 Function CalculateStrategy (  $R_i(\cdot|I_i), I_i$  ):
33   sum  $\leftarrow \sum_{a \in A(I_i)} \max(R_i(a|I_i), 0)$ 
34   For  $a \in A(I_i)$  do
35      $\sigma_i(a|I_i) = \frac{\max(R_i(a|I_i), 0)}{\text{sum}}$  if sum > 0 else  $\frac{1}{|A(I_i)|}$ 
36   return  $\sigma_i(I_i)$ 
37

```

FIG. 3

**FIG. 4**

500



Algorithm 1: Counterfactual Regret Minimization with Two Deep Neural Networks

```

1 Function Agent (  $T, b$  ):
2   For  $t = 1$  to  $T$  do
3     if  $t = 1$  and using warm starting then
4       initialize
5          $\theta_R^t$  and  $\theta_S^t$  from an existing checkpoint
6          $t \leftarrow t + 1$  ▷ skip cold starting
7     else
8       initialize  $\theta_R^t$  and  $\theta_S^t$  randomly
9        $M_R^t, M_S^t \leftarrow$  sampling methods ▷ such as Algorithm3
10      sum aggregate value in  $M_R$  by information set ▷ according to the Lemma 5 and Equation 13
11      remove duplicated records in  $M_S$ 
12      if use reservoir method then
13         $M_R \leftarrow M_R^t, M_S \leftarrow M_S^t$ 
14         $\theta_R^t \leftarrow \text{NerualAgent}( R(\cdot | \theta_R^{t-1}), M_R^t, \theta_R^{t-1}, \beta_R^* )$  ▷ update  $\theta_R^t$  using Algorithm2
15         $\theta_S^t \leftarrow \text{NerualAgent}( S(\cdot | \theta_S^{t-1}), M_S^t, \theta_S^{t-1}, \beta_S^* )$  ▷ update  $\theta_S^t$  using Algorithm2
16  return  $\theta_R^t, \theta_S^t$ 

```

FIG. 5

600



Algorithm 2: Optimization of Deep Neural Network

```

1 Function NeuralAgent ( $f(\cdot|\theta^{t-1}), M, \theta^{t-1}, \beta^*$ )
2   initialize optimizer, scheduler
3    $\theta^T \leftarrow \theta^{T-1}, l_{best} \leftarrow \infty, t_{best} \leftarrow 0$ 
4   For  $t = 1$  to  $\beta_{epoch}$  do
5      $loss \leftarrow []$ 
6     For each training epoch do
7        $\{x^{(i)}, y^{(i)}\}_{i=1}^m \sim M$ 
8        $batch\_loss \leftarrow \frac{1}{m} \sum_{i=1}^m (f(x^{(i)}|\theta^{T-1}) + y^{(i)} - f(x^{(i)}|\theta^T))^2$ 
9       back propagation batch_loss with learning rate  $lr$ 
10      clip gradient of  $\theta^T$  to  $[-\epsilon, \epsilon]^d$ 
11      optimizer(batch_loss)
12      loss.append(batch_loss)
13       $lr \leftarrow scheduler(lr)$ 
14      if  $avg(loss) < \beta_{loss}$  then
15         $\theta_{best}^T \leftarrow \theta^T$ , early stopping
16      else if  $avg(loss) < l_{best}$  then
17         $l_{best} = avg(loss), t_{best} \leftarrow t, \theta_{best}^T \leftarrow \theta^T$ 
18      if  $t - t_{best} > \beta_{re}$  then
19         $lr \leftarrow \beta_{lr}$ 
20 return  $\theta^T$ 

```

▷ gradient descent optimizer and learning rate scheduler
 ▷ warm starting from the checkpoint of the last iteration
 ▷ initialize loss as an empty list
 ▷ sampling a mini-batch from M
 ▷ d is the dimension of θ^T
 ▷ reduce learning rate adaptively when loss has stopped improving
 ▷ if loss is small enough, using early stopping mechanism
 ▷ reset learning rate to escape from potential saddle point or local minima

FIG. 6

Algorithm 3: Mini-Batch RS-MCCFR with Double Neural Networks

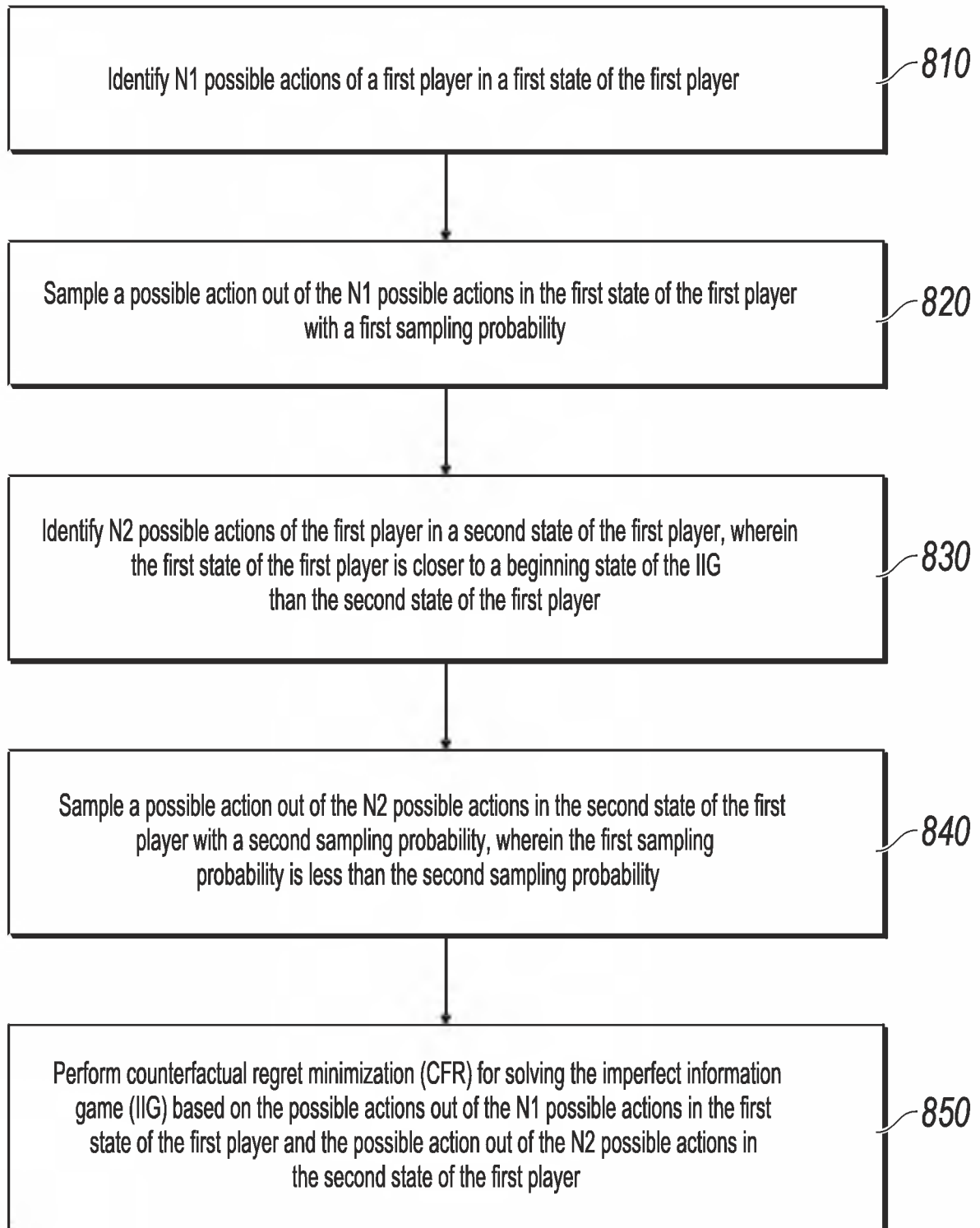
```

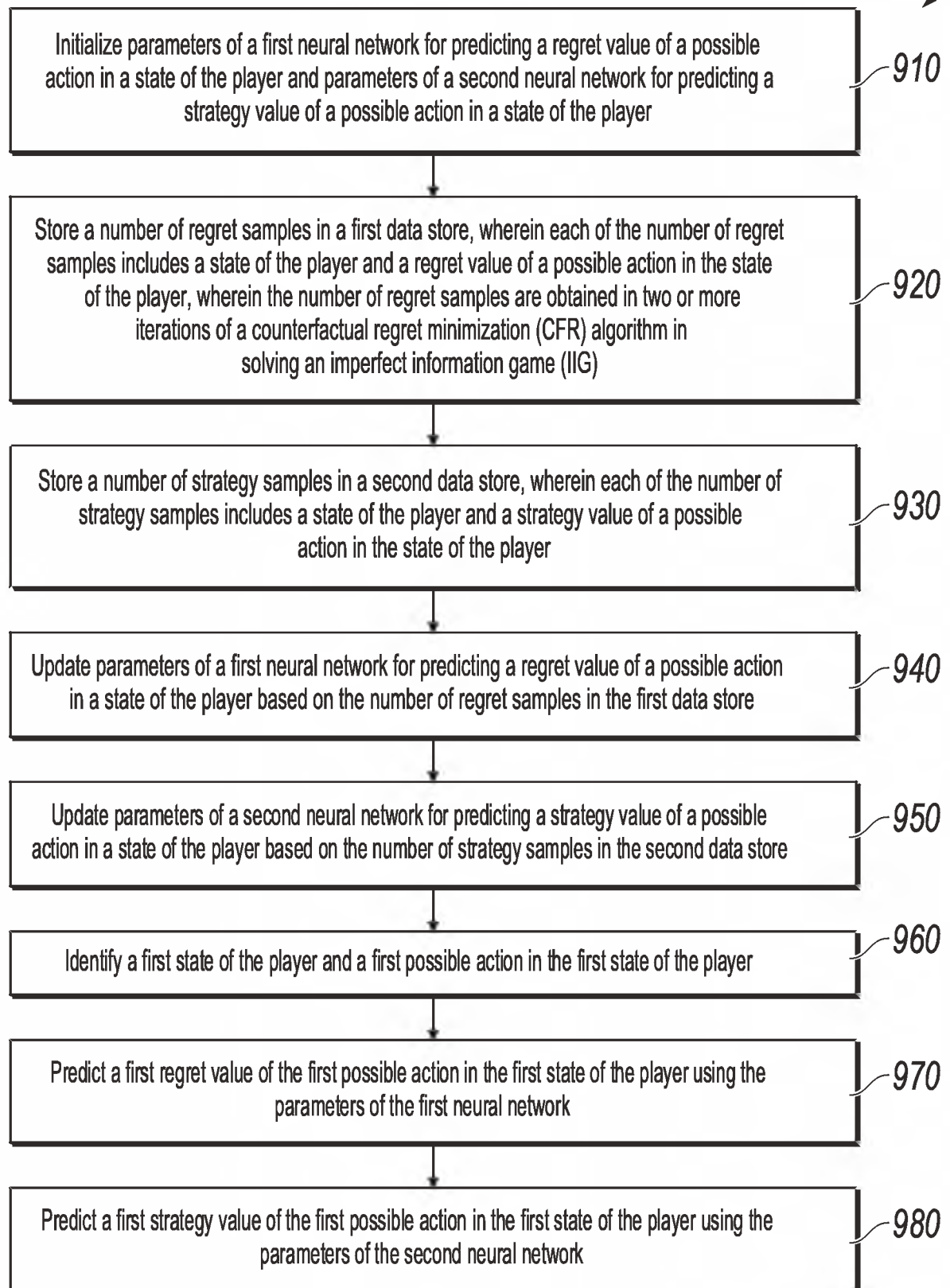
1 Function Mini-Batch-MCCFR-NN( $t$ ):
2    $M_R^t \leftarrow \emptyset, M_S^t \leftarrow \emptyset$ 
3   For all  $i = 1$  to  $b$  do in parallel then
4      $\text{MCCFR-NN}(t, \emptyset, 0, 1, 1)$ 
5      $\text{MCCFR-NN}(t, \emptyset, 1, 1, 1)$ 
6   return  $M_R^t, M_S^t$ 
7
8 Function MCCFR-NN( $t, h, i, \pi_i, \pi_i^{rs(k)}$ ):
9    $I_i \leftarrow I_i(h)$  ▷ information set at state  $h$ 
10  if  $h \in Z$  then
11    return  $\frac{u_i(h)}{\pi_i^{rs(k)}}$  ▷ return game payoff
12  else if  $P(h) = -1$  then
13     $a \sim \sigma_{-i}(I_i)$  ▷ sample an action from  $\sigma_{-i}(h)$ 
14    return  $\text{MCCFR-NN}(t, ha, i, \pi_i, \pi_i^{rs(k)})$ 
15  else if  $P(h) = i$  then
16     $\hat{R}_i(\cdot|I_i) \leftarrow R(\cdot, I_i | \theta_R^i)$  if  $t > 1$  else  $\vec{0}$  ▷ inference the vector of cumulative regret  $\mathbf{A} \mathbf{a} \in A(I_i)$ 
17     $\sigma_{-i}(I_i) \leftarrow \text{CalculateStrategy}(\hat{R}_i(\cdot|I_i), I_i)$  ▷ calculate current strategy
18     $v_i(h) \leftarrow 0, r_i(\cdot|I_i) \leftarrow \vec{0}, s_i(\cdot|I_i) \leftarrow \vec{0}$  ▷  $r_i(\cdot|I_i)$  and  $s_i(\cdot|I_i)$  are two vectors over  $A(I_i)$ 
19     $A^{rs(k)}(I_i) \leftarrow$  sampling  $k$  different actions according to  $\sigma_i^{rs(k)}$ 
20    For  $a \in A^{rs(k)}(I_i)$  do
21       $v_i(a|h) \leftarrow \text{MCCFR-NN}(t, ha, i, \pi_i, \sigma_i(a|I_i), \pi_i^{rs} \sigma_i^{rs(k)}(a|I_i))$ 
22       $v_i(h) \leftarrow v_i(h) + v_i(a|h) \sigma_i(a|I_i)$  ▷ update counterfactual value
23    For  $a \in A^{rs(k)}(I_i)$  do
24       $r_i(a|I_i) \leftarrow v_i(a|h) - v_i(h)$  ▷ update cumulative regret
25       $s_i(a|I_i) \leftarrow \pi_i^o(I_i) \sigma_i(a|I_i)$  ▷ update average strategy numerator
26    Store updated cumulative regret tuple ( $I_i, r_i(\cdot|I_i)$ ) in  $M_R$ 
27    Store updated current strategy dictionary ( $I_i, s_i(\cdot|I_i)$ ) in  $M_S$ 
28    return  $v_i(h)$ 
29  else
30     $\hat{R}_{-i}(\cdot|I_i) \leftarrow R(\cdot, I_i | \theta_R^i)$  if  $t > 1$  else  $\vec{0}$  ▷ inference cumulative regret
31     $\sigma_{-i}(I_i) \leftarrow \text{CalculateStrategy}(\hat{R}_{-i}(\cdot|I_i), I_i)$  ▷ calculate current strategy
32     $a \sim \sigma_{-i}(I_i)$  ▷ sample an action from  $\sigma_{-i}(I_i)$ 
33    return  $\text{MCCFR-NN}(t, ha, i, \pi_i, \pi_i^{rs(k)})$ 
34
35 Function CalculateStrategy( $\hat{R}_i(\cdot|I_i), I_i$ ):
36    $sum \leftarrow \sum_{a \in A(I_i)} \max(\hat{R}_i(a|I_i), 0)$ 
37   For  $a \in A(I_i)$  do
38      $\sigma_i(a|I_i) = \frac{\max(\hat{R}_i(a|I_i), 0)}{sum}$  if  $sum > 0$  else  $\frac{1}{|A(I_i)|}$ 
39   return  $\sigma_i(I_i)$ 
40

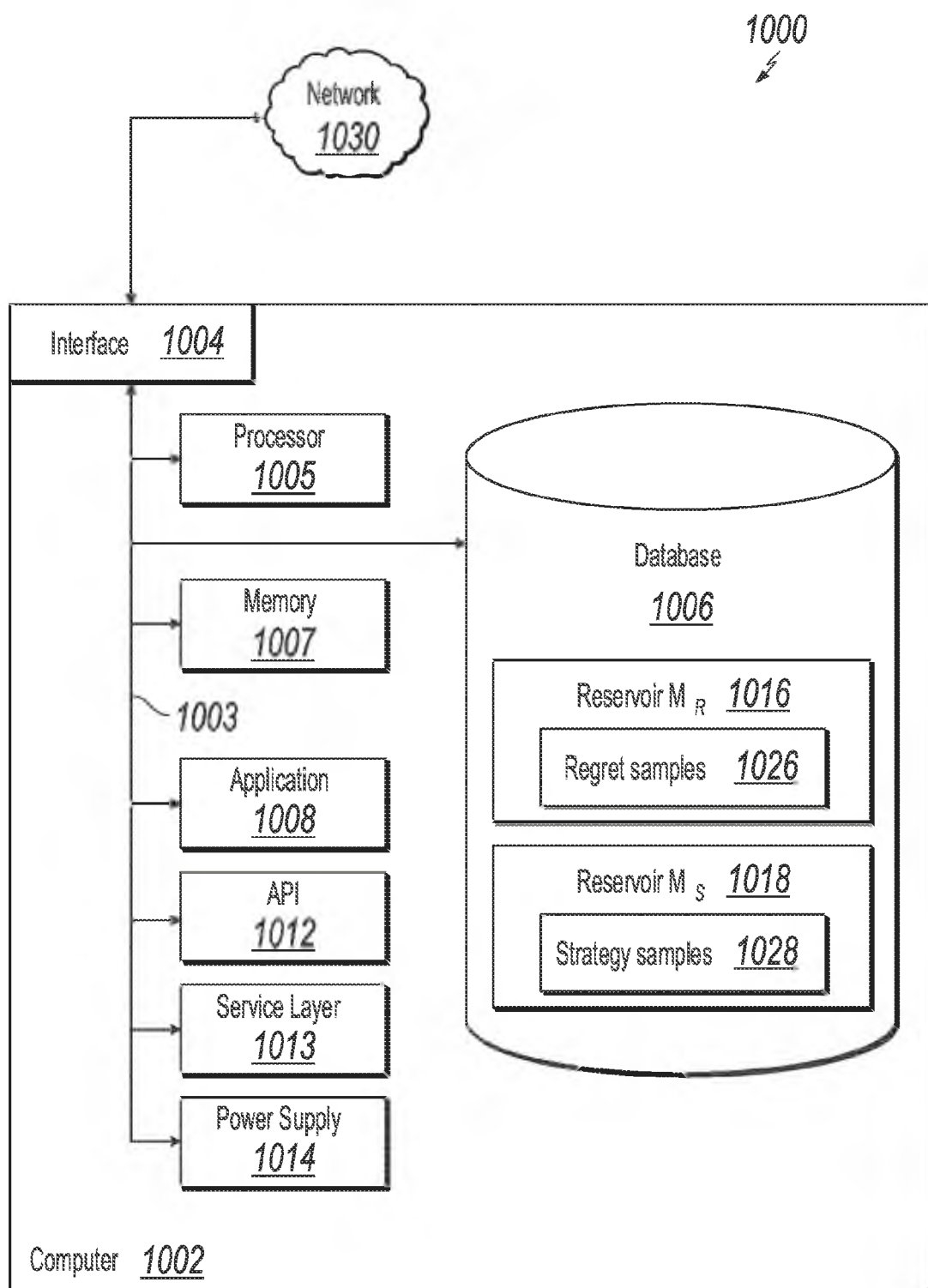
```

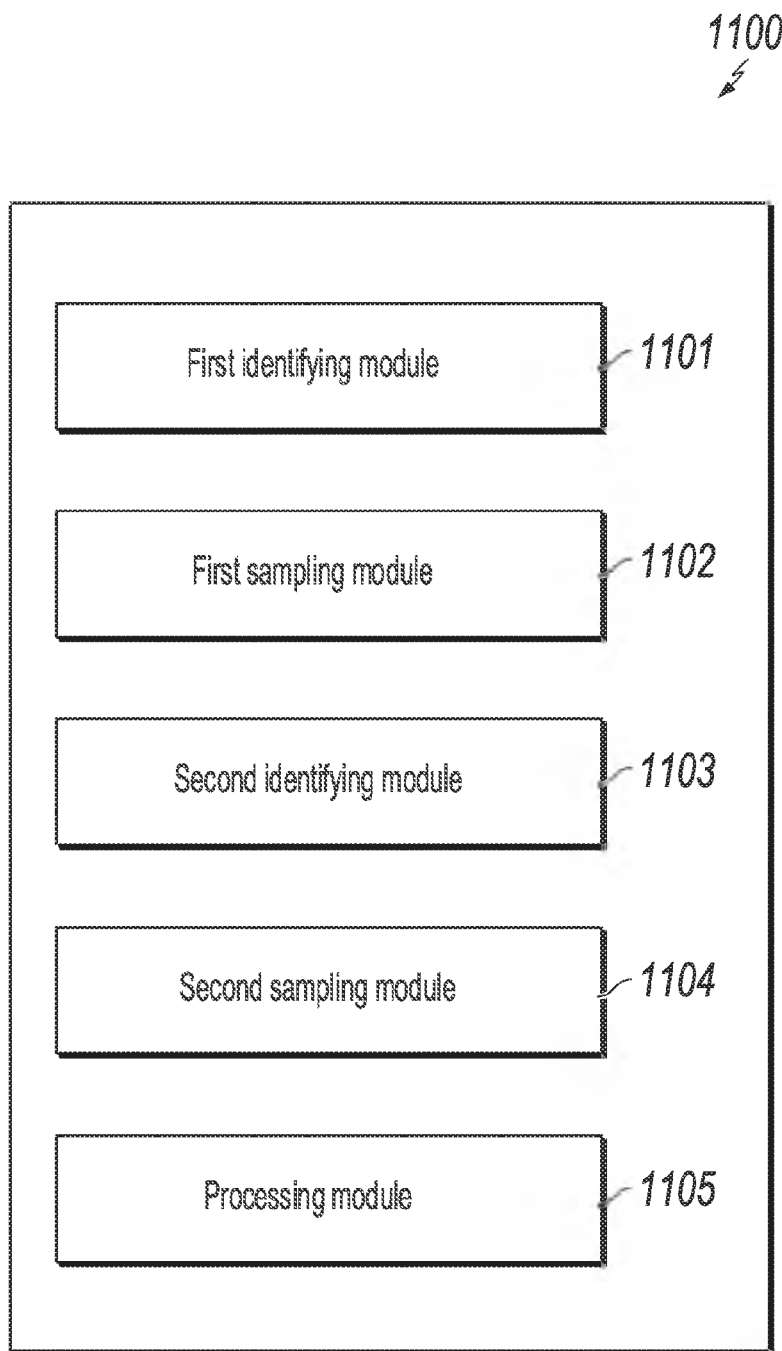
FIG. 7

700

**FIG. 8**

**FIG. 9**

**FIG. 10**

**FIG. 11**

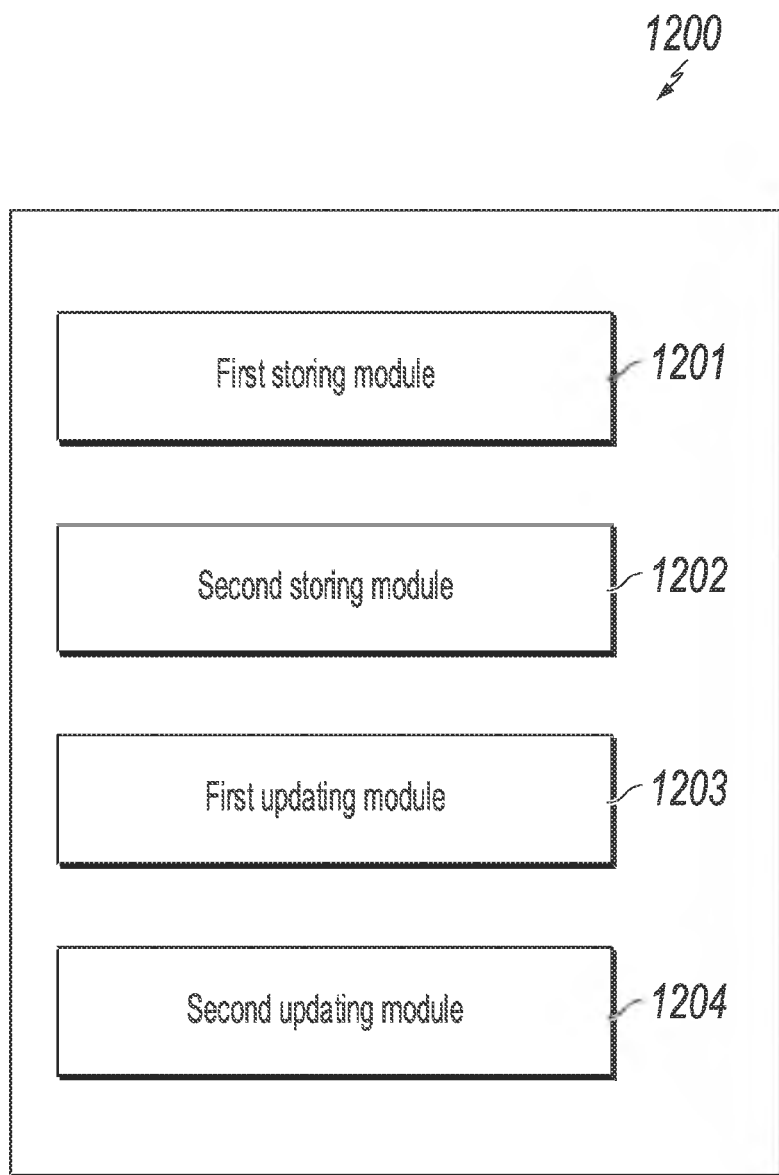


FIG. 12