Resolvendo o Caixeiro Viajante como um jogo através de Reinforcement Learning

Caixeiro Viajante, Deep Q-Learning, Policy Gradients

Kevin Takano

Tópicos Especiais em Aprendizagem de Máquina e Mineração de Dados
Instituto de Computação - IComp
Universidade Federal do Amazonas - UFAM

takano@icomp.ufam.edu.br

12 de Dezembros de 2017
Manaus - Amazonas, Brasil

Roteiro

Roteiro

Objetivo de Projeto

- Resolver um problema de otimização combinatória **através de Reinforcement Learning**.
- Estudar o **funcionamento prático** das técnicas clássicas de Deep Reinforcement Learning:
  - Deep Q-Learning.
  - Stochastic Policy Gradient.

Objetivo de Projeto

- Resolver um problema de otimização combinatória **através de Reinforcement Learning**.
- Estudar o **funcionamento prático** das técnicas clássicas de Deep Reinforcement Learning:
  - Deep Q-Learning.
  - Stochastic Policy Gradient.

Objetivo de Projeto

- Resolver um problema de otimização combinatória **através de Reinforcement Learning**.
- Estudar o **funcionamento prático** das técnicas clássicas de Deep Reinforcement Learning:
  - Deep Q-Learning.
  - Stochastic Policy Gradient.

Objetivo de Projeto

- Resolver um problema de otimização combinatória **através de Reinforcement Learning**.
- Estudar o **funcionamento prático** das técnicas clássicas de Deep Reinforcement Learning:
  - Deep Q-Learning.
  - Stochastic Policy Gradient.

Objetivo de Projeto

- Resolver um problema de otimização combinatória **através de Reinforcement Learning**.
- Estudar o **funcionamento prático** das técnicas clássicas de Deep Reinforcement Learning:
  - Deep Q-Learning.
  - Stochastic Policy Gradient.

Roteiro

Motivação

- Problemas combinatórios são encontrados **em largas áreas** da ciência.
- Sabe-se que **há uma grande dificuldade** em encontrar a solução ótima para estes problemas. (NP-Dificuldade)
- Problemas combinatórios também necessitam de **conhecimento especializado**.
  - Muitos dos mesmos problemas do mundo real são **resolvidos várias vezes**, porém, diferindo-se apenas nas **estrutura dos dados**.

Motivação

- Problemas combinatórios são encontrados **em largas áreas** da ciência.
- Sabe-se que **há uma grande dificuldade** em encontrar a solução ótima para estes problemas. (NP-Dificuldade)
- Problemas combinatórios também necessitam de **conhecimento especializado**.
  - Muitos dos mesmos problemas do mundo real são **resolvidos várias vezes**, porém, diferindo-se apenas nas **estrutura dos dados**.

Motivação

- Problemas combinatórios são encontrados **em largas áreas** da ciência.
- Sabe-se que **há uma grande dificuldade** em encontrar a solução ótima para estes problemas. (NP-Dificuldade)
- Problemas combinatórios também necessitam de **conhecimento especializado**.
  - Muitos dos mesmos problemas do mundo real são **resolvidos várias vezes**, porém, diferindo-se apenas nas **estrutura dos dados**.

Motivação

- Problemas combinatórios são encontrados **em largas áreas** da ciência.
- Sabe-se que **há uma grande dificuldade** em encontrar a solução ótima para estes problemas. (NP-Dificuldade)
- Problemas combinatórios também necessitam de **conhecimento especializado**.
  - Muitos dos mesmos problemas do mundo real são **resolvidos várias vezes**, porém, diferindo-se apenas nas **estrutura dos dados**.

Motivação

- Problemas combinatórios são encontrados **em largas áreas** da ciência.
- Sabe-se que **há uma grande dificuldade** em encontrar a solução ótima para estes problemas. (NP-Dificuldade)
- Problemas combinatórios também necessitam de **conhecimento especializado**.
  - Muitos dos mesmos problemas do mundo real são **resolvidos várias vezes**, porém, diferindo-se apenas nas **estrutura dos dados**.

Motivação

- Aprendizado de Máquina procura resolver os problemas através de busca por **generalização** dos dados e/ou busca por **padrões**.
- É interessante resolver os **problemas de combinatória** de forma **genérica**.
  - Isso reduz a dependência sobre o **conhecimento específico** de um problema e o **retrabalho** de implementação.
- Em 2017, o Google publicou um paper que resolve o **problema da Mochila 0-1** e o problema do **Caixeiro Viajante** utilizando-se uma **mesma** rede neural. *Rede Neural Combinatória* [Irwan Bello, 2017]
- Ainda em 2017, um paper publicado demonstrou que problemas de otimização em grafos podem ser resolvidos por **Reinforcement Learning e Graph Embedding**. [Anjun Dai, 2017]

Motivação

- Aprendizado de Máquina procura resolver os problemas através de busca por **generalização** dos dados e/ou busca por **padrões**.
- É interessante resolver os **problemas de combinatória** de forma **genérica**.
  - Isso reduz a dependência sobre o **conhecimento específico** de um problema e o **retrabalho** de implementação.
- Em 2017, o Google publicou um paper que resolve o **problema da Mochila 0-1** e o problema do **Caixeiro Viajante** utilizando-se uma **mesma** rede neural. *Rede Neural Combinatória* [Irwan Bello, 2017]
- Ainda em 2017, um paper publicado demonstrou que problemas de otimização em grafos podem ser resolvidos por **Reinforcement Learning e Graph Embedding**. [Anjun Dai, 2017]

Motivação

- Aprendizado de Máquina procura resolver os problemas através de busca por **generalização** dos dados e/ou busca por **padrões**.
- É interessante resolver os **problemas de combinatória** de forma **genérica**.
  - Isso reduz a dependência sobre o **conhecimento específico** de um problema e o **retrabalho** de implementação.
- Em 2017, o Google publicou um paper que resolve o **problema da Mochila 0-1** e o problema do **Caixeiro Viajante** utilizando-se uma **mesma** rede neural. *Rede Neural Combinatória* [Irwan Bello, 2017]
- Ainda em 2017, um paper publicado demonstrou que problemas de otimização em grafos podem ser resolvidos por **Reinforcement Learning e Graph Embedding**. [Anjun Dai, 2017]

Motivação

- Aprendizado de Máquina procura resolver os problemas através de busca por **generalização** dos dados e/ou busca por **padrões**.
- É interessante resolver os **problemas de combinatória** de forma **genérica**.
  - Isso reduz a dependência sobre o **conhecimento específico** de um problema e o **retrabalho** de implementação.
- Em 2017, o Google publicou um paper que resolve o **problema da Mochila 0-1** e o problema do **Caixeiro Viajante** utilizando-se uma **mesma** rede neural. *Rede Neural Combinatória* [Irwan Bello, 2017]
- Ainda em 2017, um paper publicado demonstrou que problemas de otimização em grafos podem ser resolvidos por **Reinforcement Learning e Graph Embedding**. [Anjun Dai, 2017]

Motivação

- Aprendizado de Máquina procura resolver os problemas através de busca por **generalização** dos dados e/ou busca por **padrões**.
- É interessante resolver os **problemas de combinatória** de forma **genérica**.
  - Isso reduz a dependência sobre o **conhecimento específico** de um problema e o **retrabalho** de implementação.
- Em 2017, o Google publicou um paper que resolve o **problema da Mochila 0-1** e o problema do **Caixeiro Viajante** utilizando-se uma **mesma** rede neural. *Rede Neural Combinatória* [Irwan Bello, 2017]
- Ainda em 2017, um paper publicado demonstrou que problemas de otimização em grafos podem ser resolvidos por **Reinforcement Learning e Graph Embedding**. [Anjun Dai, 2017]

Motivação

- Aprendizado de Máquina procura resolver os problemas através de busca por **generalização** dos dados e/ou busca por **padrões**.
- É interessante resolver os **problemas de combinatória** de forma **genérica**.
  - Isso reduz a dependência sobre o **conhecimento específico** de um problema e o **retrabalho** de implementação.
- Em 2017, o Google publicou um paper que resolve o **problema da Mochila 0-1** e o problema do **Caixeiro Viajante** utilizando-se uma **mesma** rede neural. *Rede Neural Combinatória* [Irwan Bello, 2017]
- Ainda em 2017, um paper publicado demonstrou que problemas de otimização em grafos podem ser resolvidos por **Reinforcement Learning e Graph Embedding**. [Anjun Dai, 2017]

Roteiro

Definição do PCV Clássico

- Definição: Dado um conjunto de n cidades e uma matriz de distância d_{ij} entre elas, o Problema do Caixeiro Viajante consiste em estabelecer uma rota para um caixeiro, iniciando seu percurso em uma cidade, chamada cidade de origem, passar por todas as demais $n-1$ cidades uma única vez e retornar a cidade de origem percorrendo a menor distância possível.

Definição do PCV Clássico

- Definição: Dado um conjunto de n cidades e uma matriz de distância d_{ij} entre elas, o Problema do Caixeiro Viajante consiste em estabelecer uma rota para um caixeiro, iniciando seu percurso em uma cidade, chamada cidade de origem, passar por todas as demais $n-1$ cidades uma única vez e retornar a cidade de origem percorrendo a menor distância possível.

Modelagem como um problema de Reinforcement Learning

- O problema do Caixeiro Viajante pode ser modelado como um jogo de PACMAM. [Clark, 2016]
  - O objetivo do game PACMAN é **acumular um maior** número de pontos **desviando-se** dos fantasmas.
  - De forma similar, o caixeiro viajante pode ser modelado **como um jogo** considerando-se o caixeiro como **um agente** em que o objetivo consiste em **passar em todas as cidades** (ou items) em **um menor número** de passos.
  - Similar ao Caixeiro Viajante Físico (*The Physical Travelling salesman*).
    - Obstáculos.

Modelagem como um problema de Reinforcement Learning

- O problema do Caixeiro Viajante pode ser modelado como um jogo de PACMAM. [Clark, 2016]
  - O objetivo do game PACMAN é **acumular um maior** número de pontos **desviando-se** dos fantasmas.
  - De forma similar, o caixeiro viajante pode ser modelado **como um jogo** considerando-se o caixeiro como **um agente** em que o objetivo consiste em **passar em todas as cidades** (ou items) em **um menor número** de passos.
  - Similar ao Caixeiro Viajante Físico (*The Physical Travelling salesman*).
    - Obstáculos.

Modelagem como um problema de Reinforcement Learning

- O problema do Caixeiro Viajante pode ser modelado como um jogo de PACMAM. [Clark, 2016]
  - O objetivo do game PACMAN é **acumular um maior** número de pontos **desviando-se** dos fantasmas.
  - De forma similar, o caixeiro viajante pode ser modelado **como um jogo** considerando-se o caixeiro como **um agente** em que o objetivo consiste em **passar em todas as cidades** (ou items) em **um menor número** de passos.
  - Similar ao Caixeiro Viajante Físico (*The Physical Travelling salesman*).
    - Obstáculos.

Modelagem como um problema de Reinforcement Learning

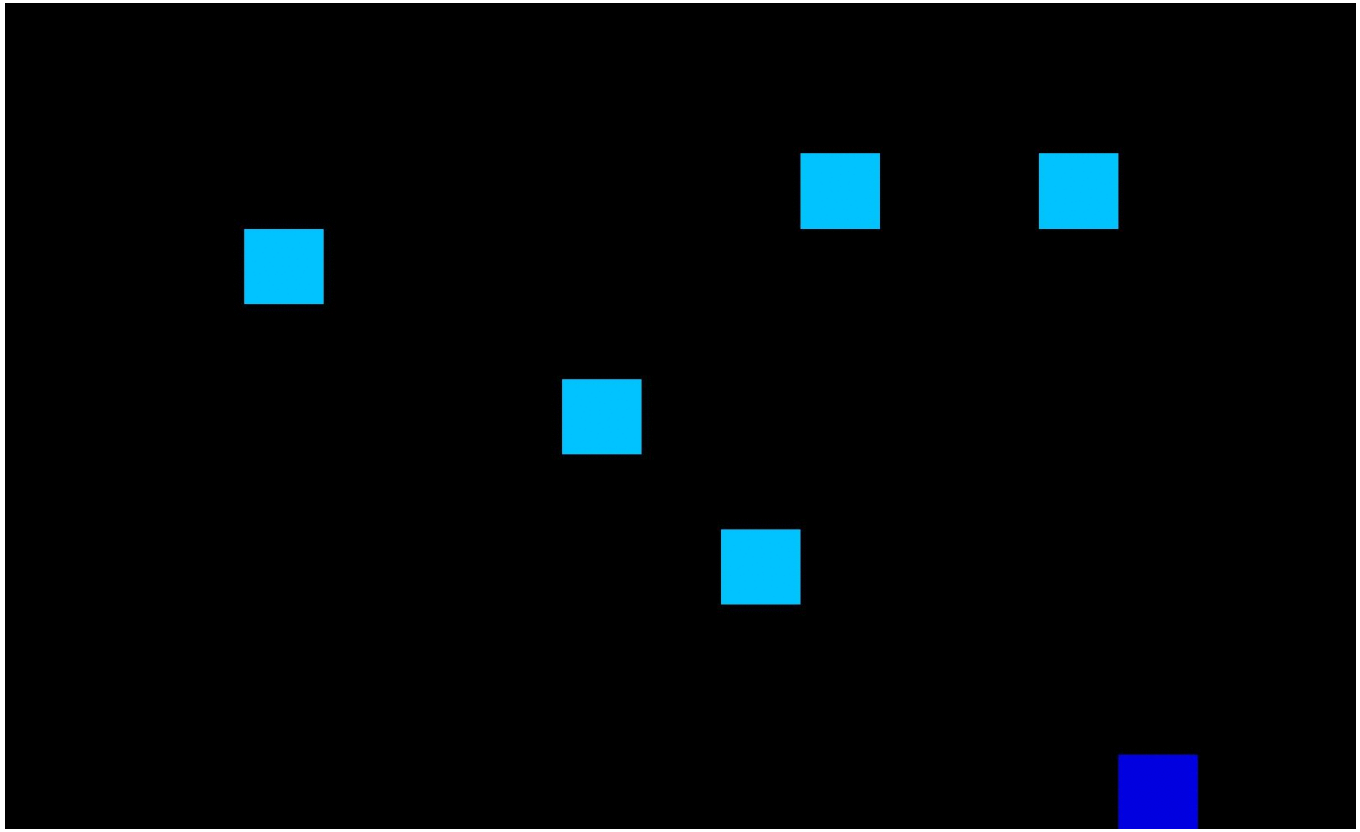
- O problema do Caixeiro Viajante pode ser modelado como um jogo de PACMAM. [Clark, 2016]
  - O objetivo do game PACMAN é **acumular um maior** número de pontos **desviando-se** dos fantasmas.
  - De forma similar, o caixeiro viajante pode ser modelado **como um jogo** considerando-se o caixeiro como **um agente** em que o objetivo consiste em **passar em todas as cidades** (ou items) em **um menor número** de passos.
  - Similar ao Caixeiro Viajante Físico (*The Physical Travelling salesman*).
    - Obstáculos.

Modelagem como um problema de Reinforcement Learning

- O problema do Caixeiro Viajante pode ser modelado como um jogo de PACMAM. [Clark, 2016]
  - O objetivo do game PACMAN é **acumular um maior** número de pontos **desviando-se** dos fantasmas.
  - De forma similar, o caixeiro viajante pode ser modelado **como um jogo** considerando-se o caixeiro como **um agente** em que o objetivo consiste em **passar em todas as cidades** (ou items) em **um menor número** de passos.
  - Similar ao Caixeiro Viajante Físico (*The Physical Travelling salesman*).
    - Obstáculos.

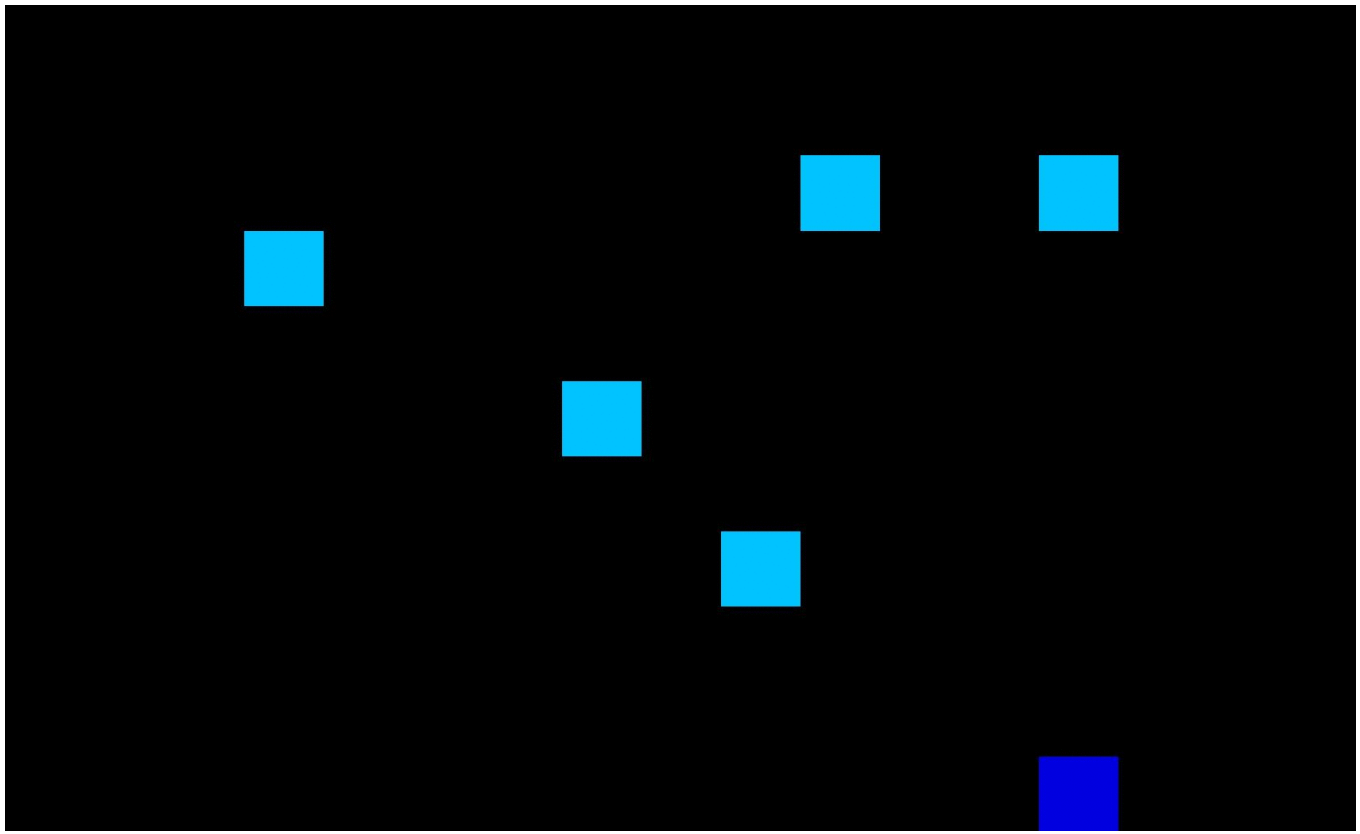
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



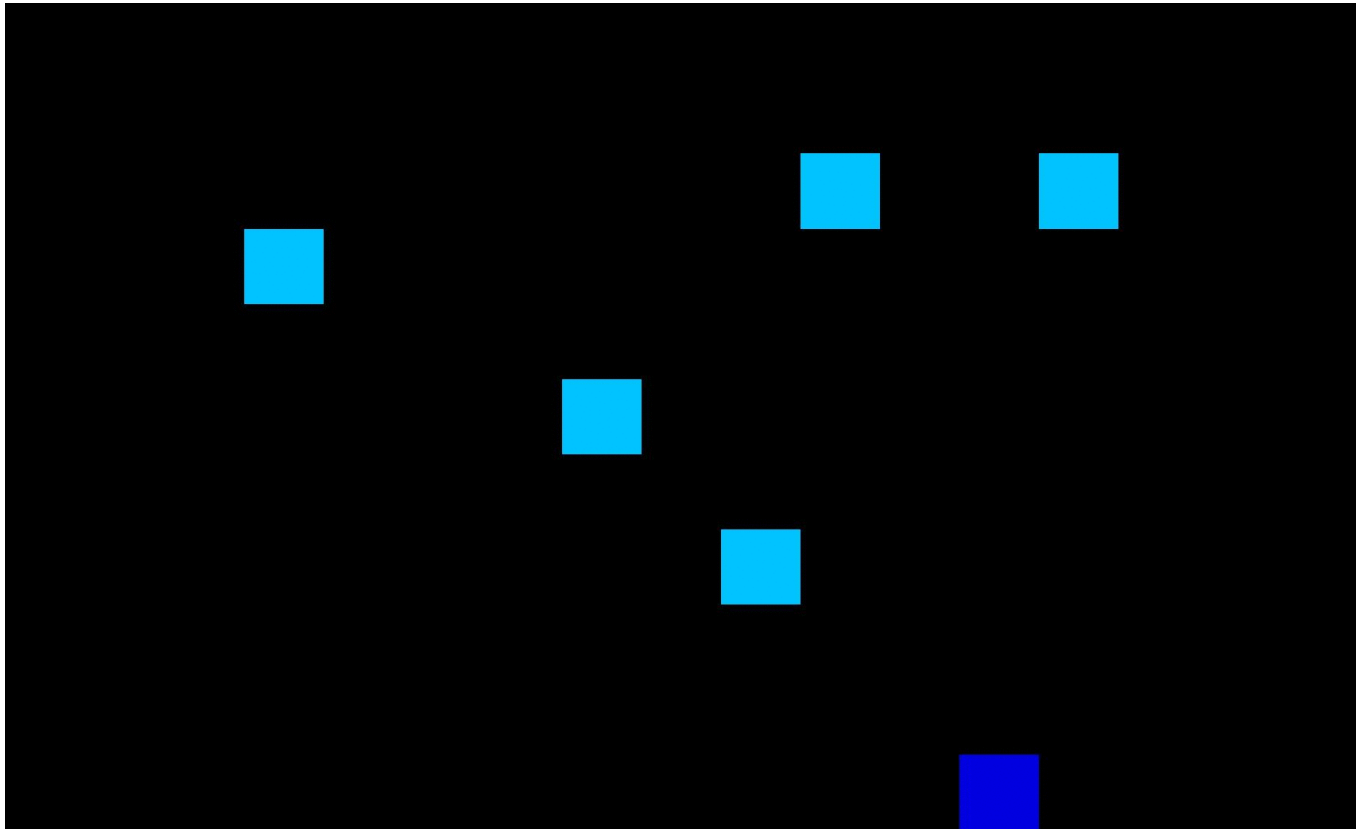
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



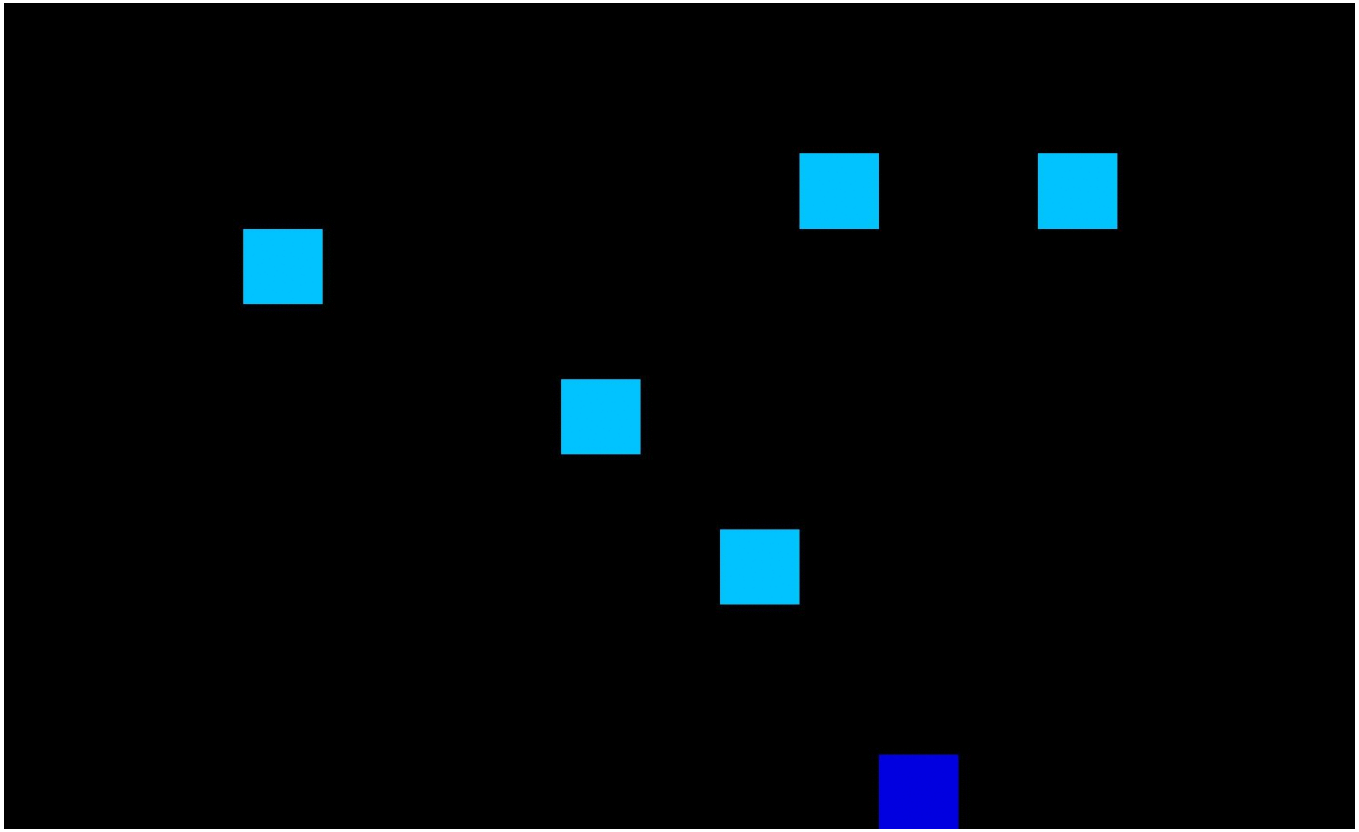
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



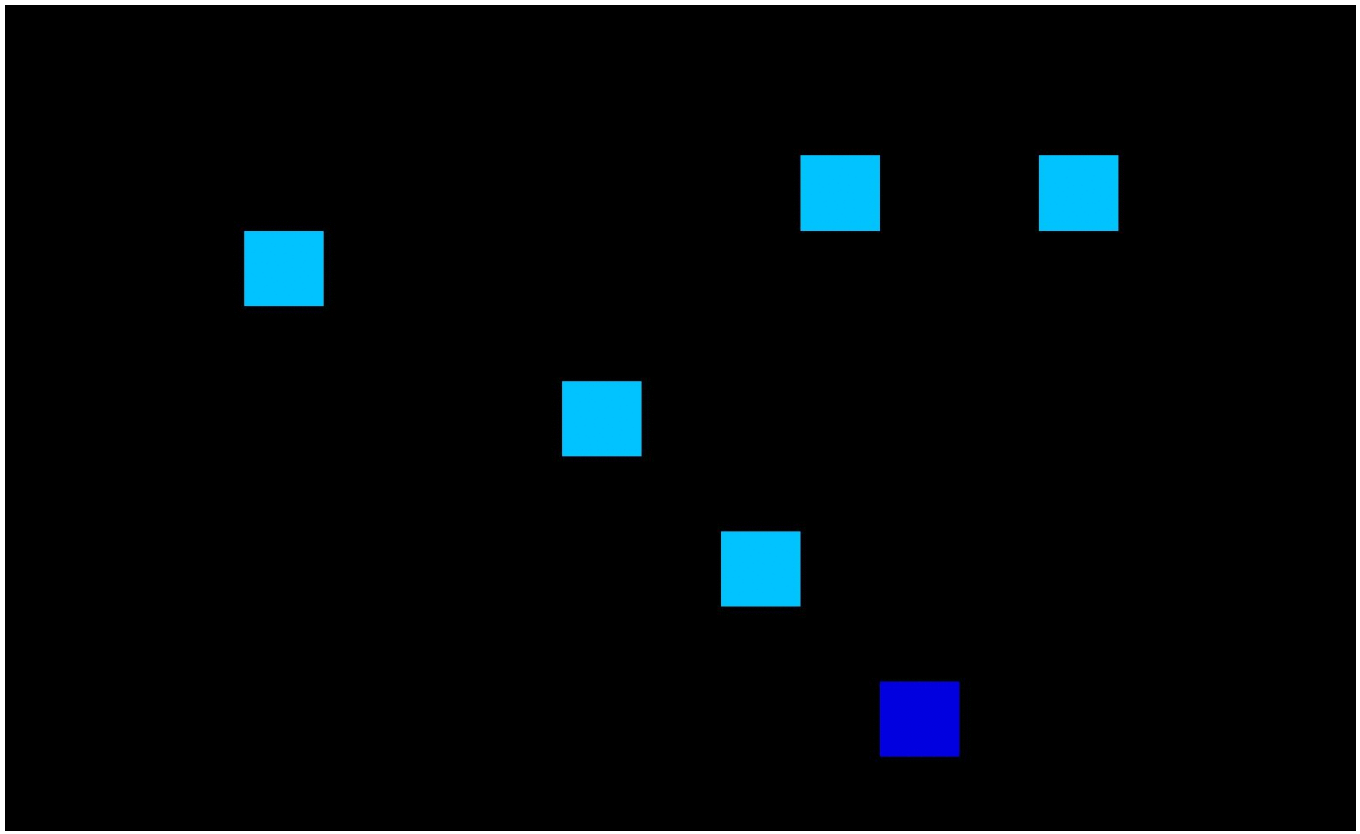
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



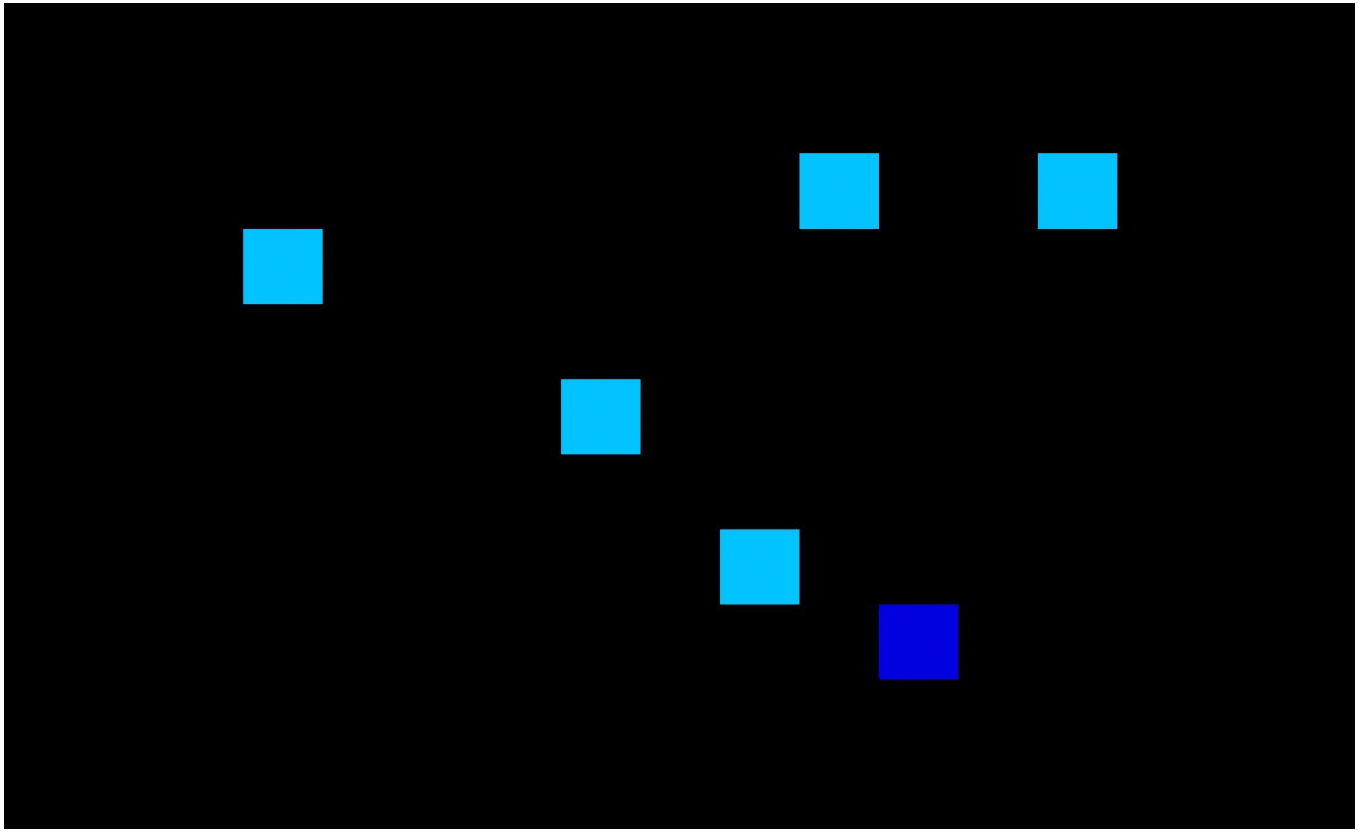
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



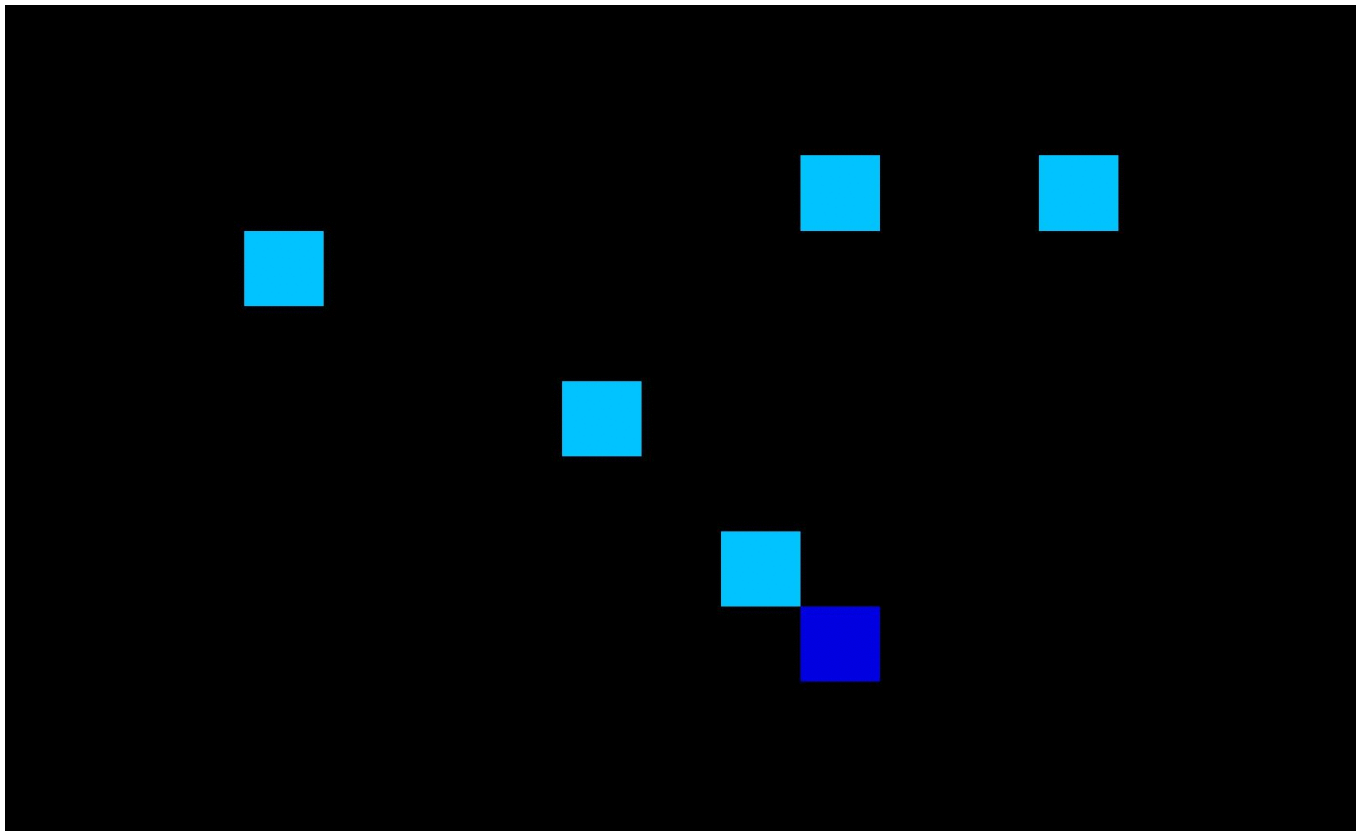
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



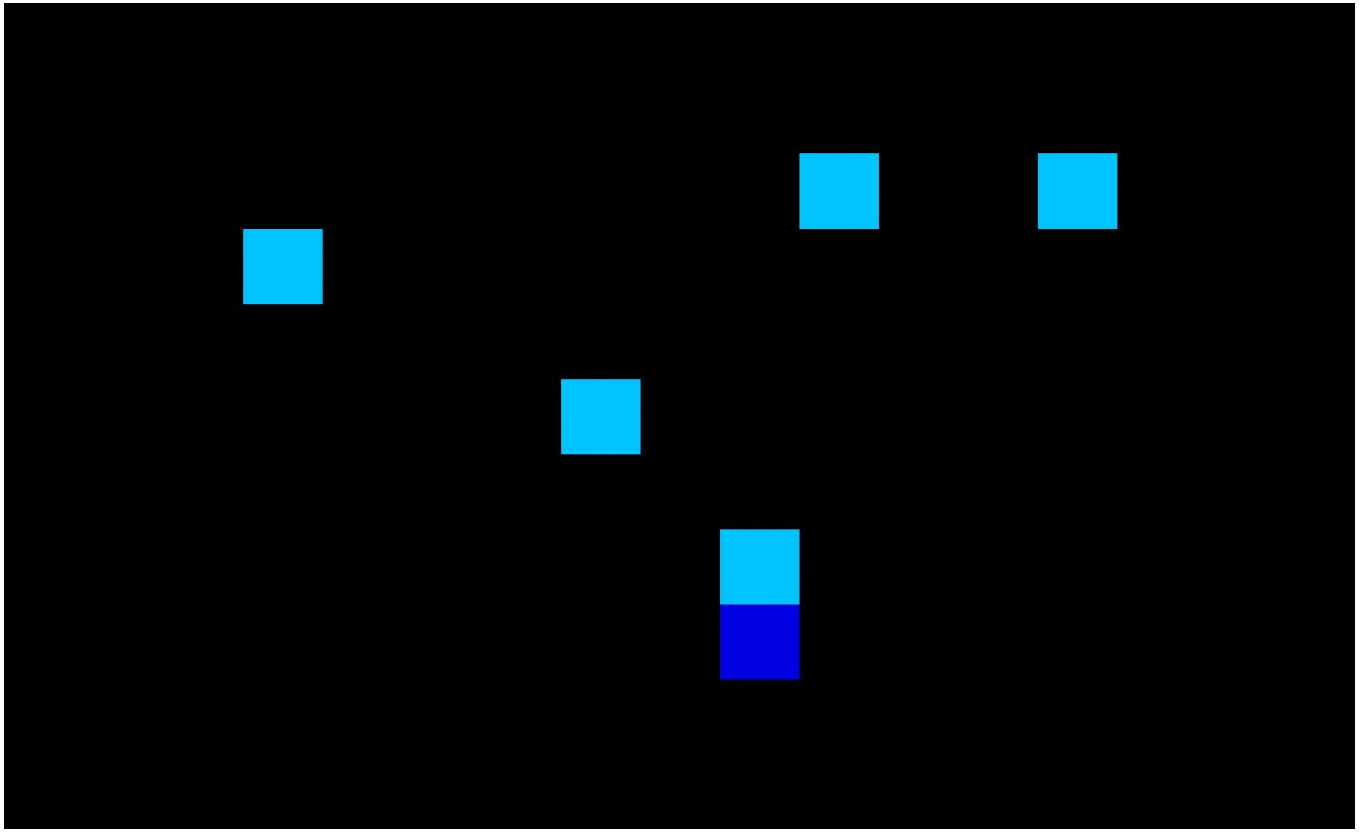
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



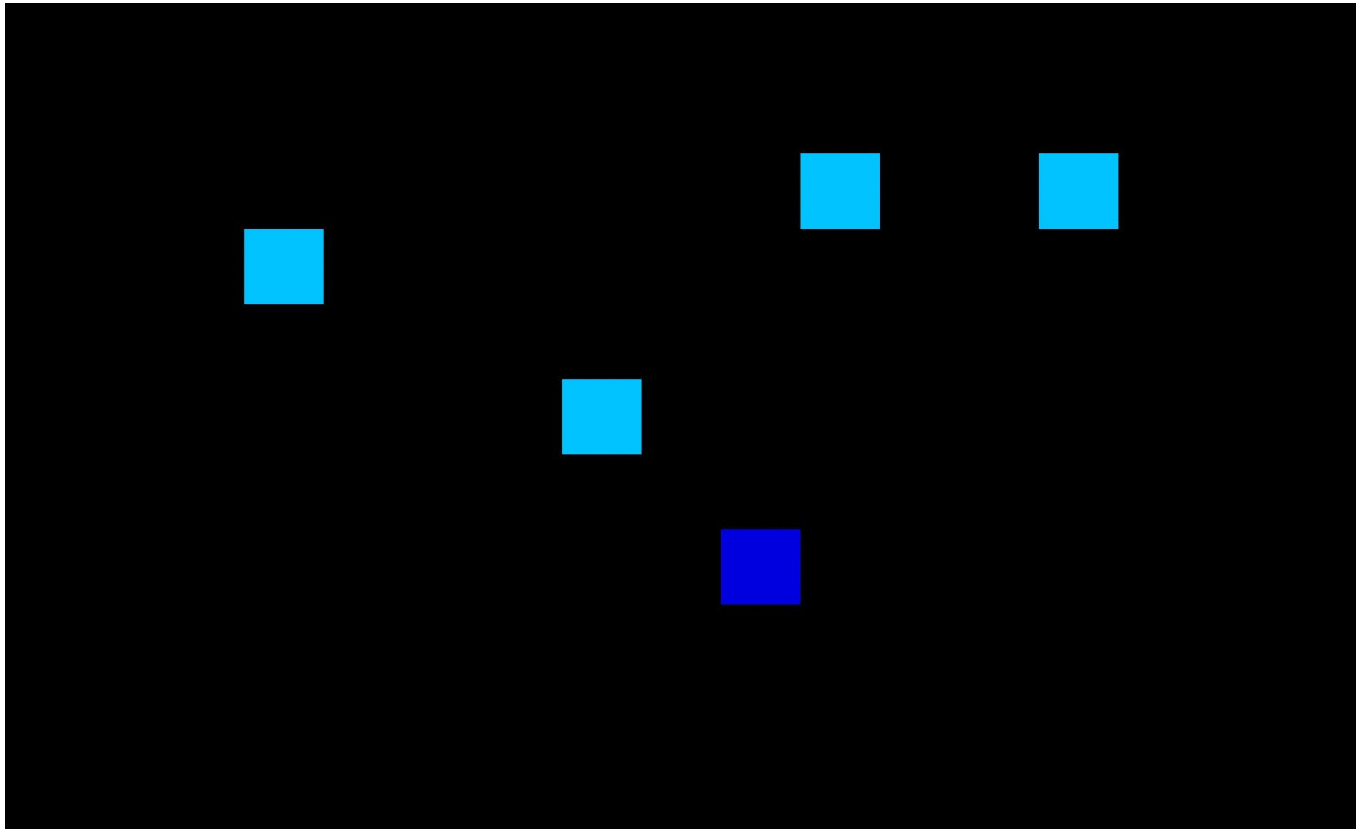
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



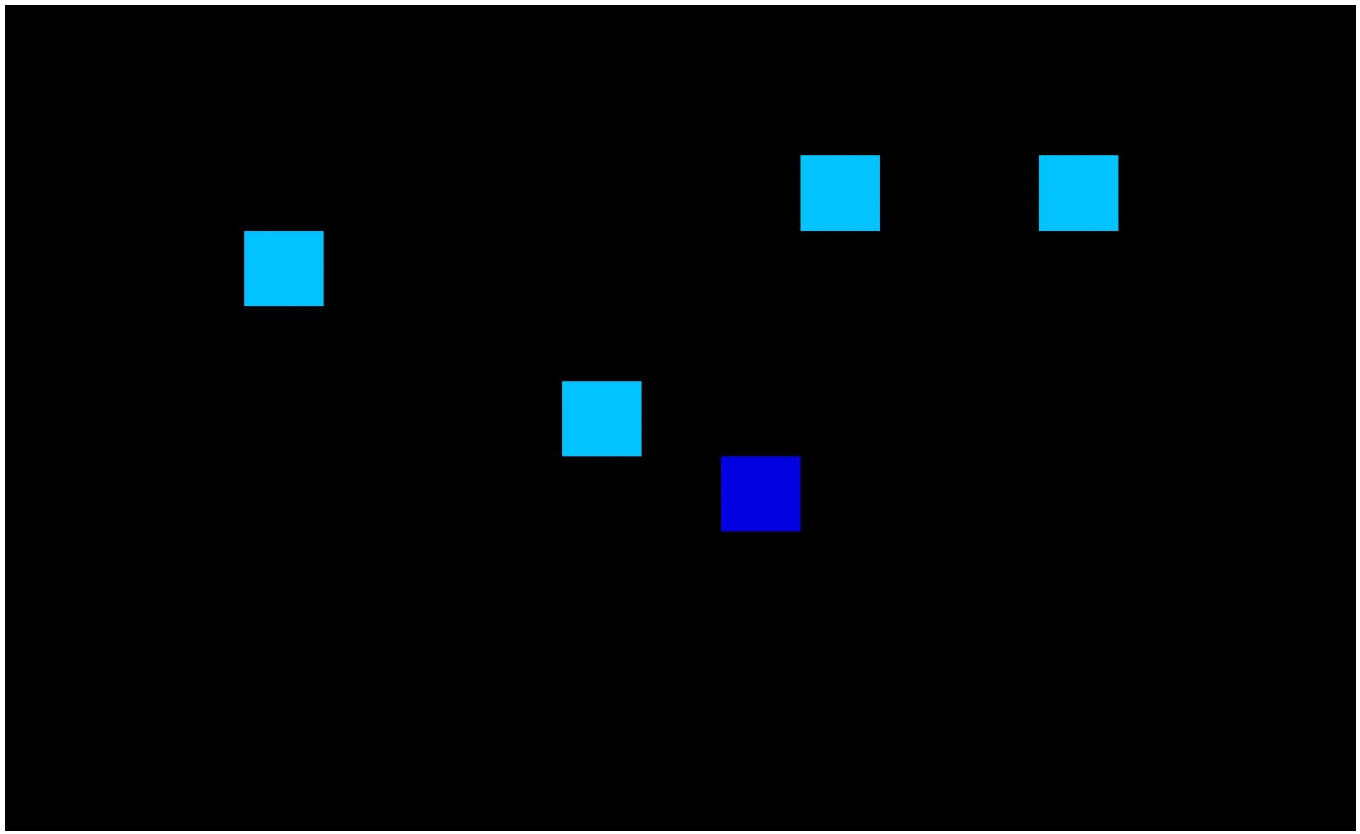
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



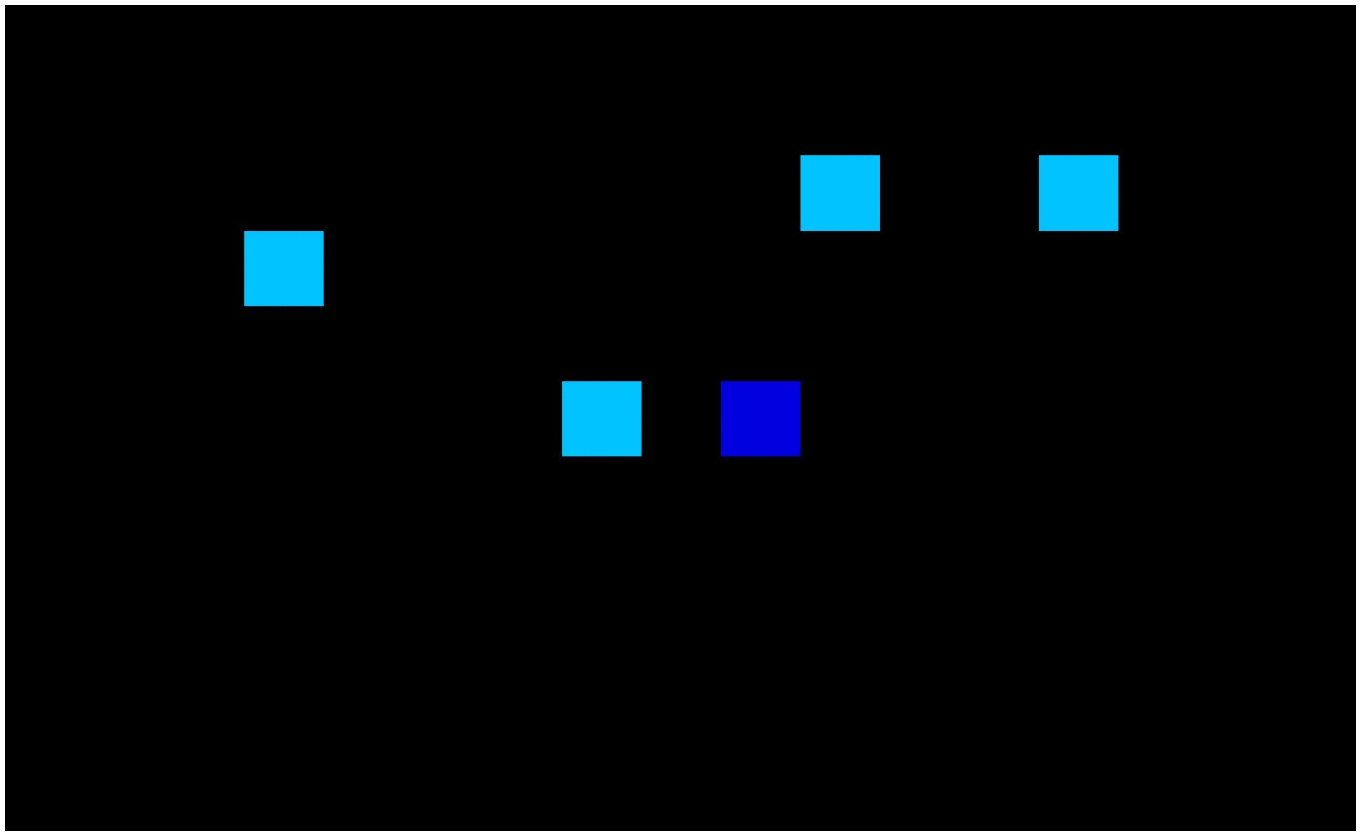
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



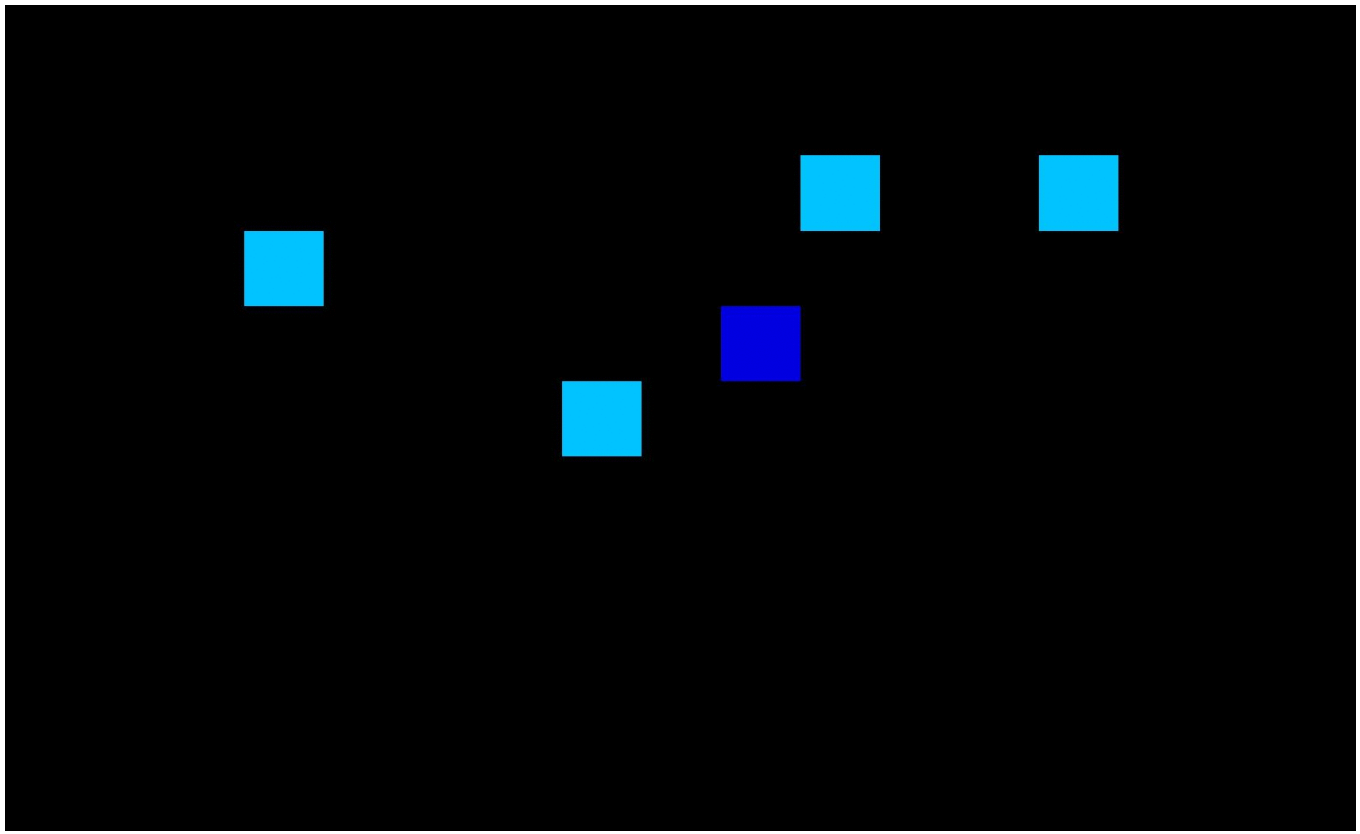
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



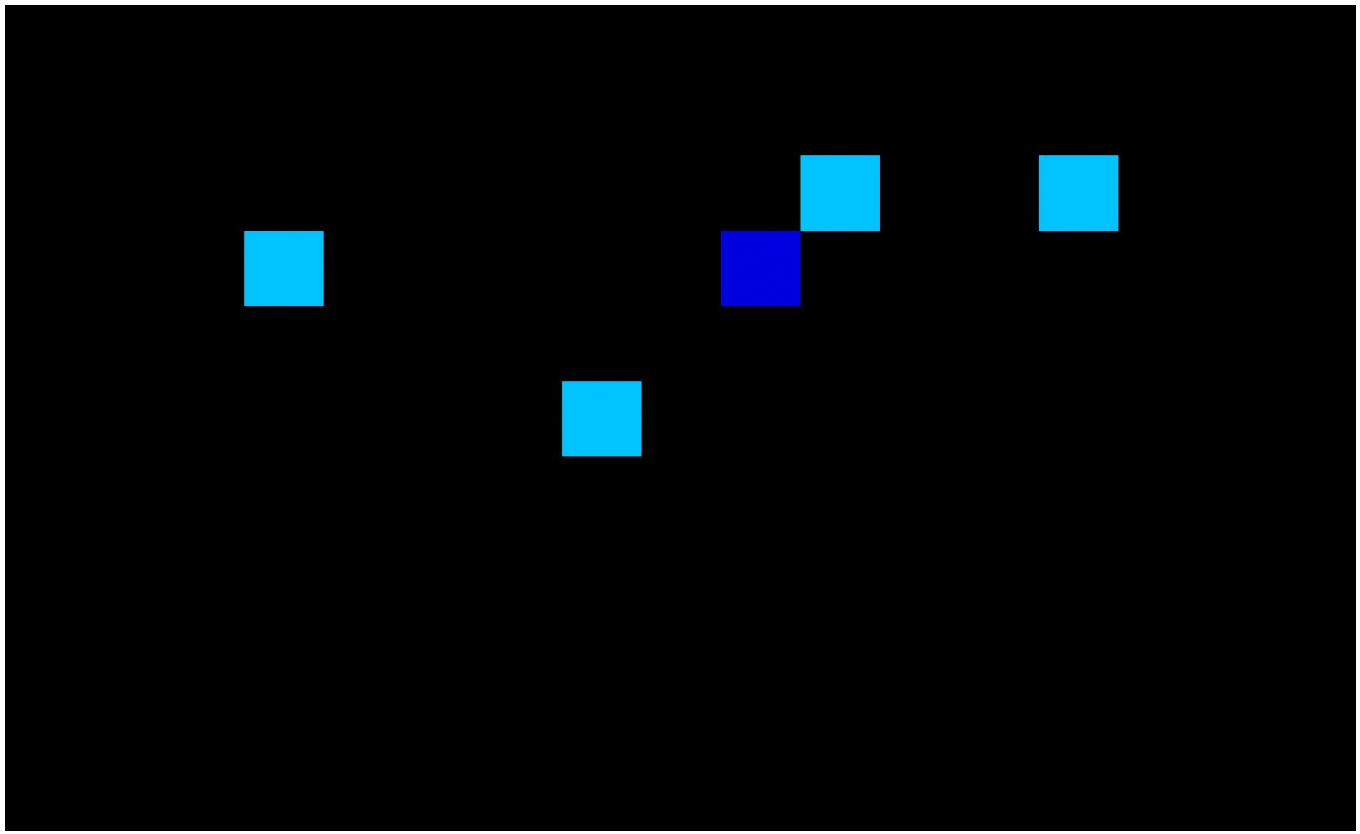
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



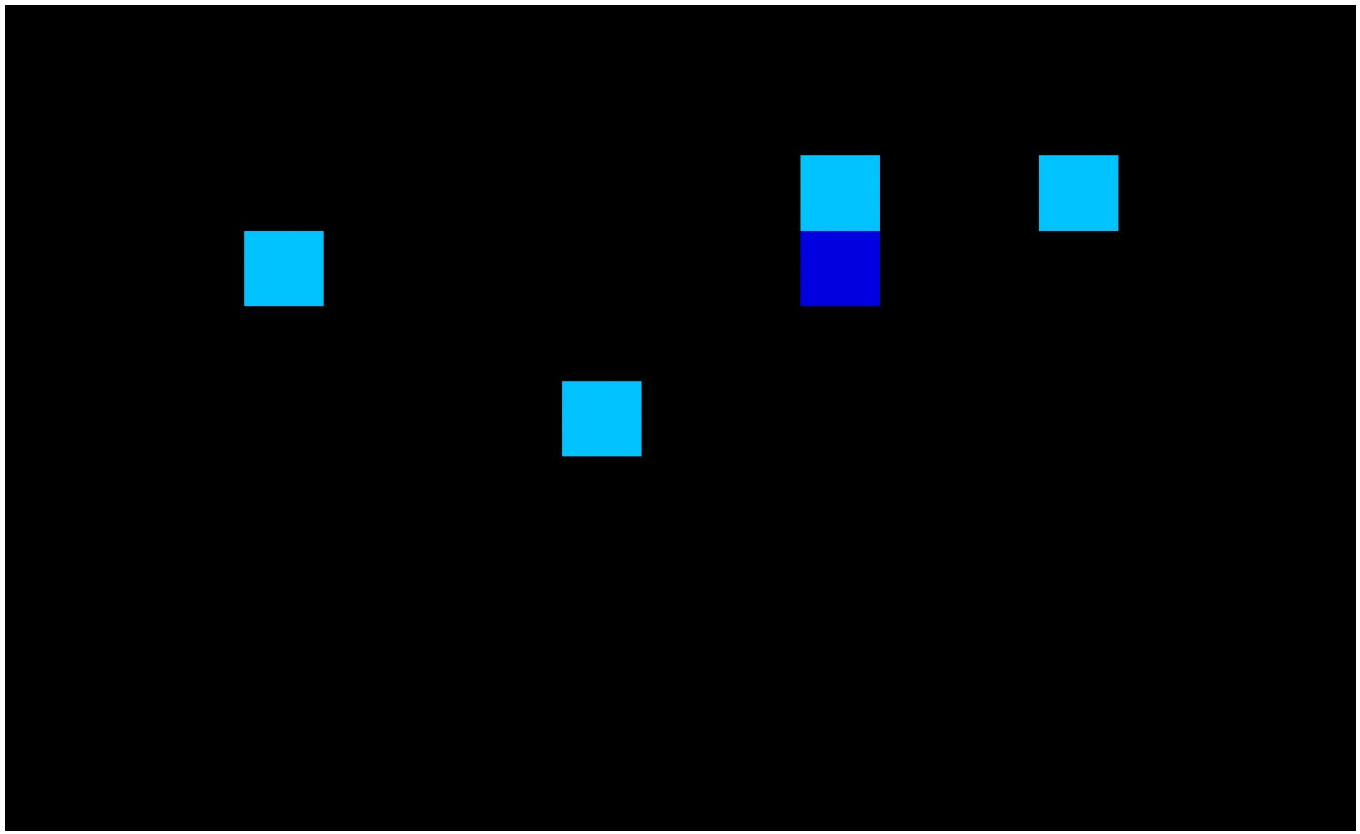
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



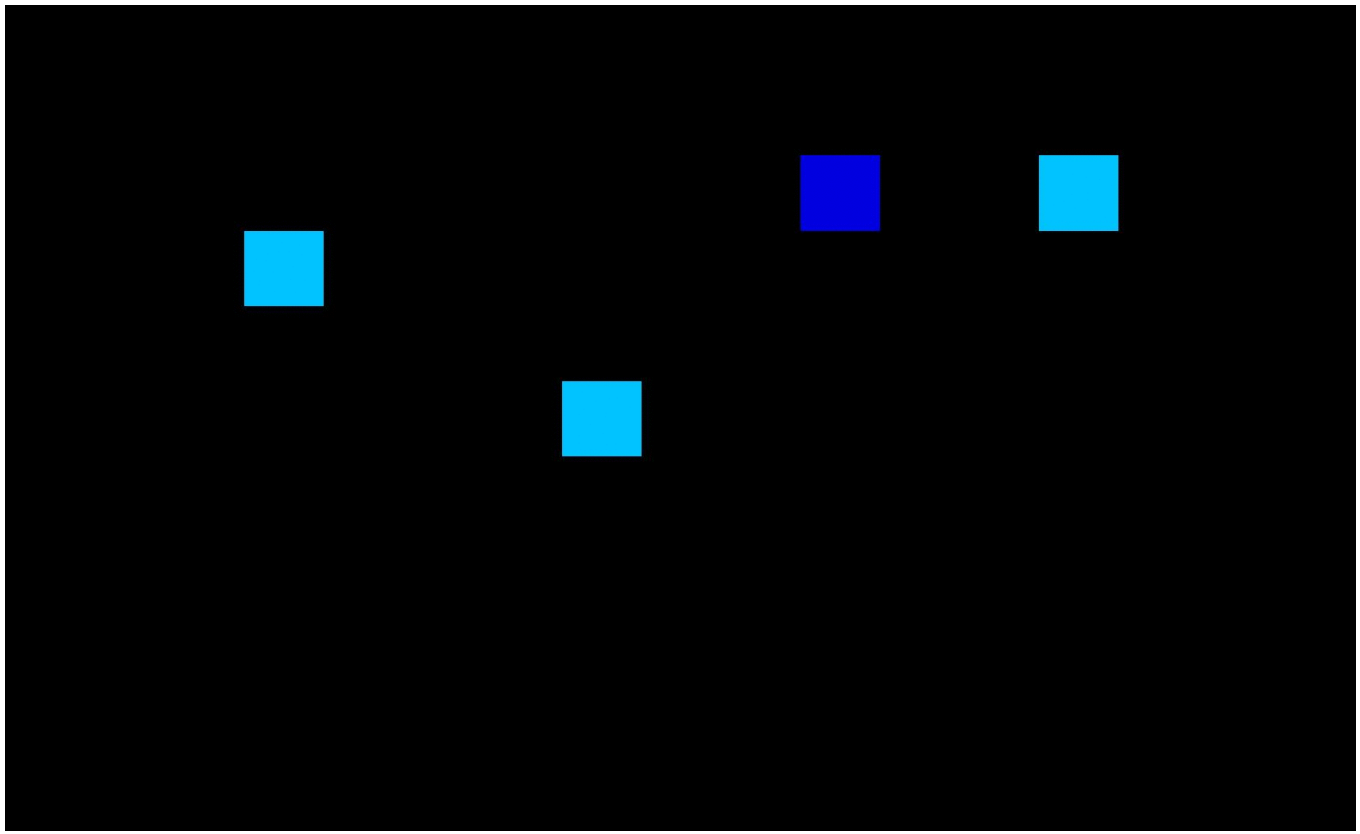
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



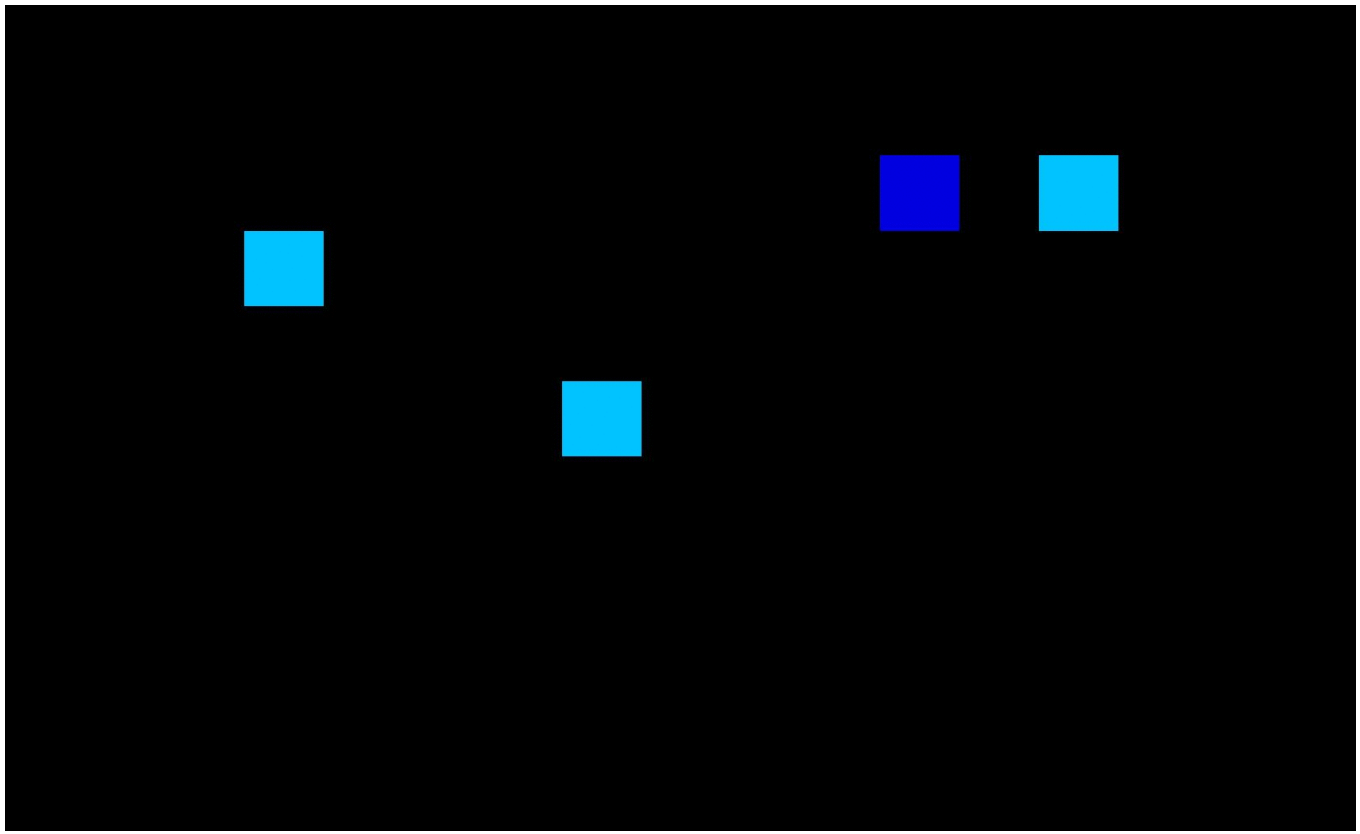
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



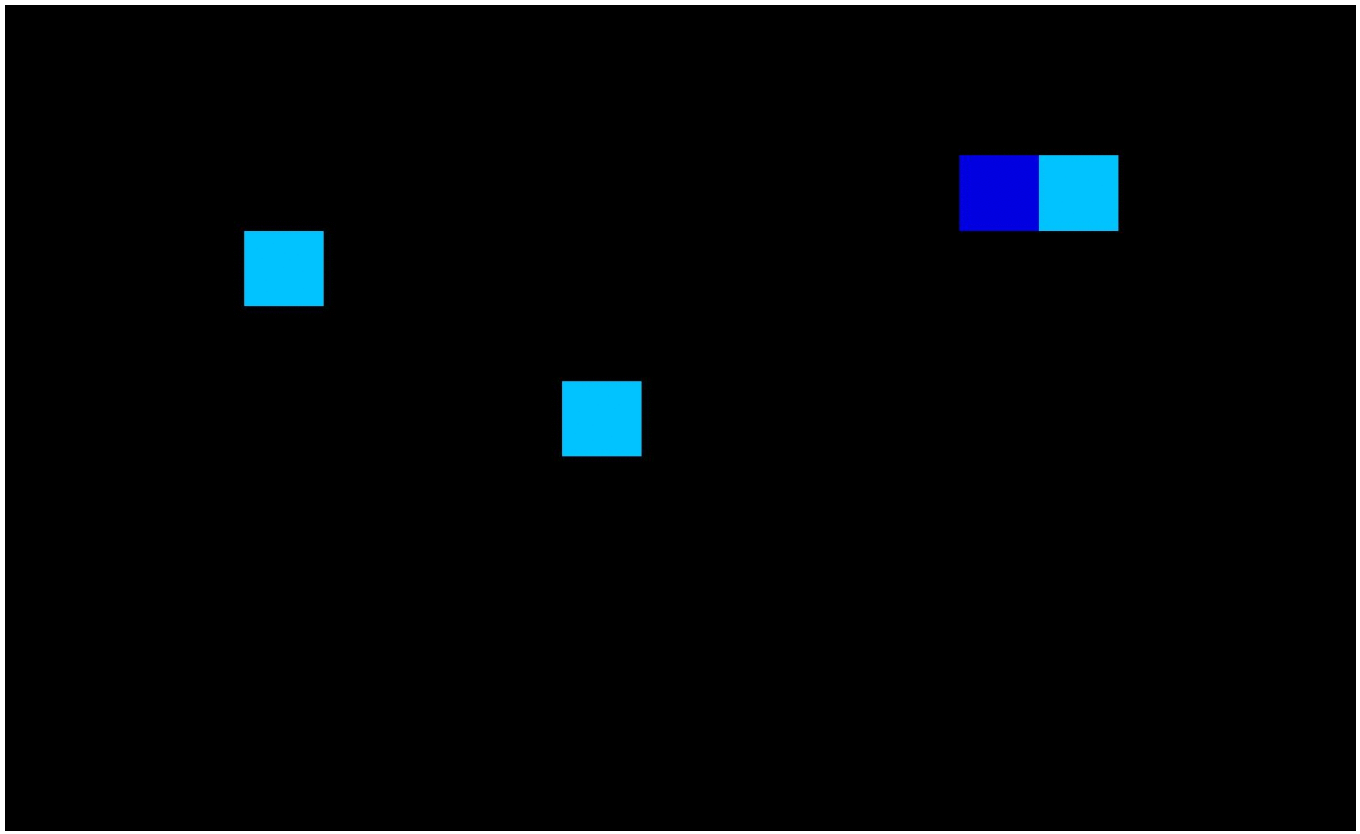
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



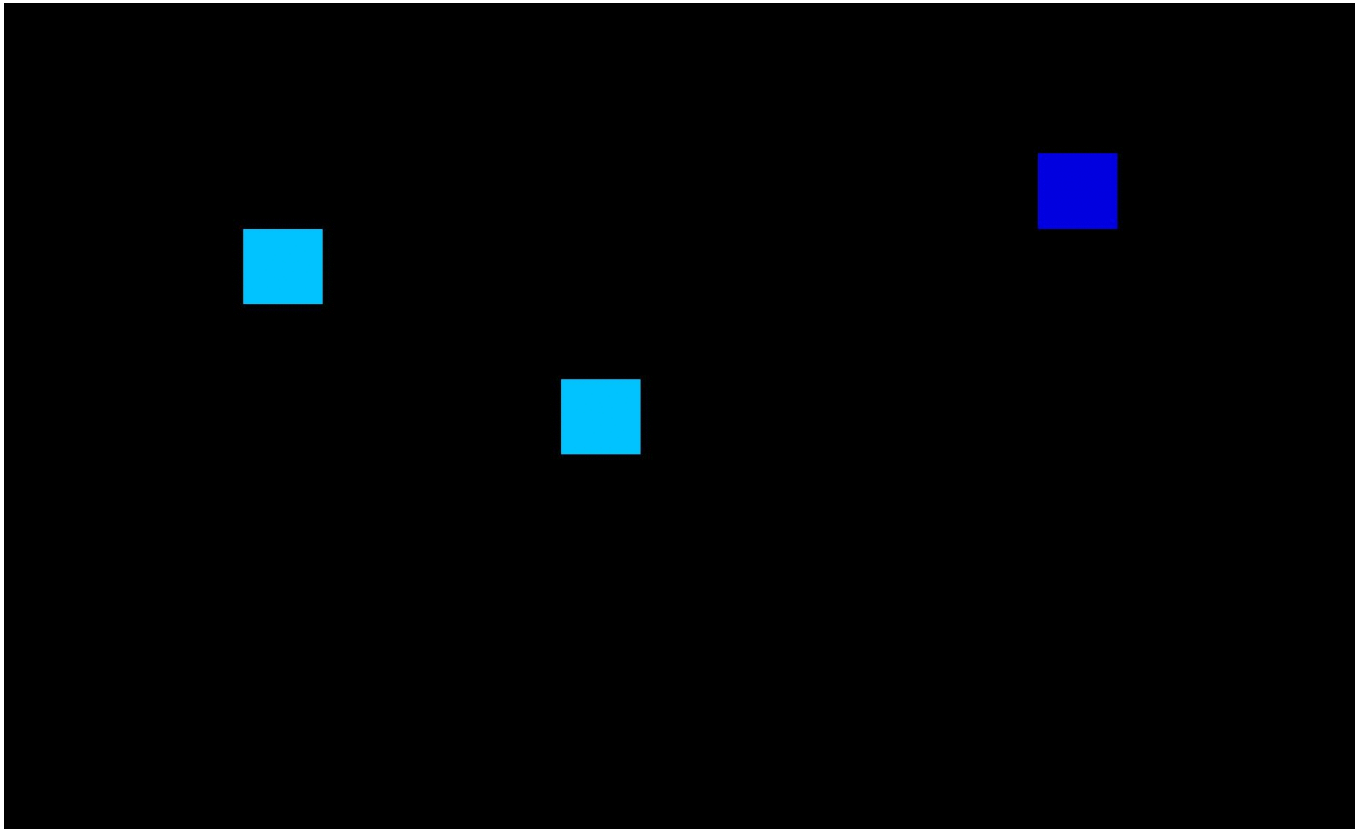
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



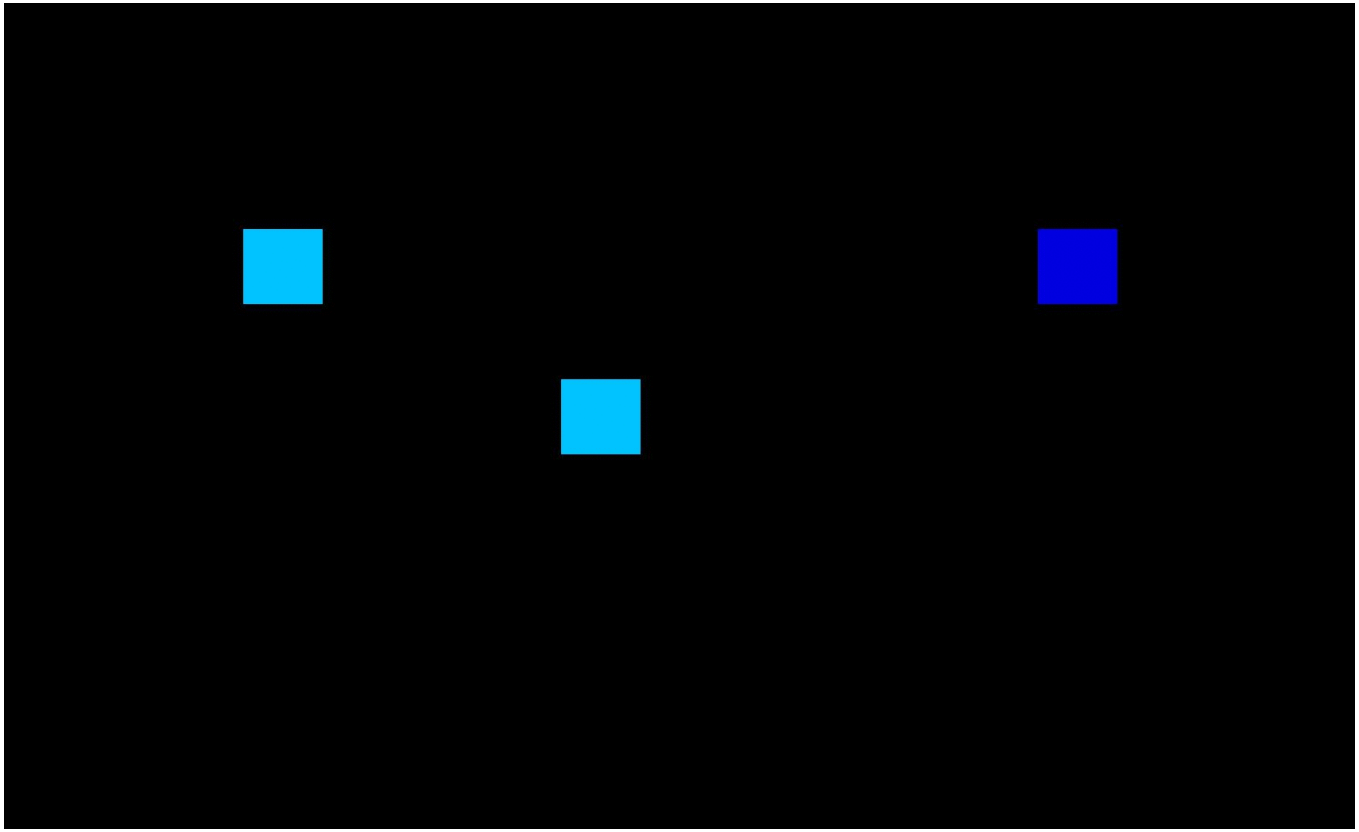
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



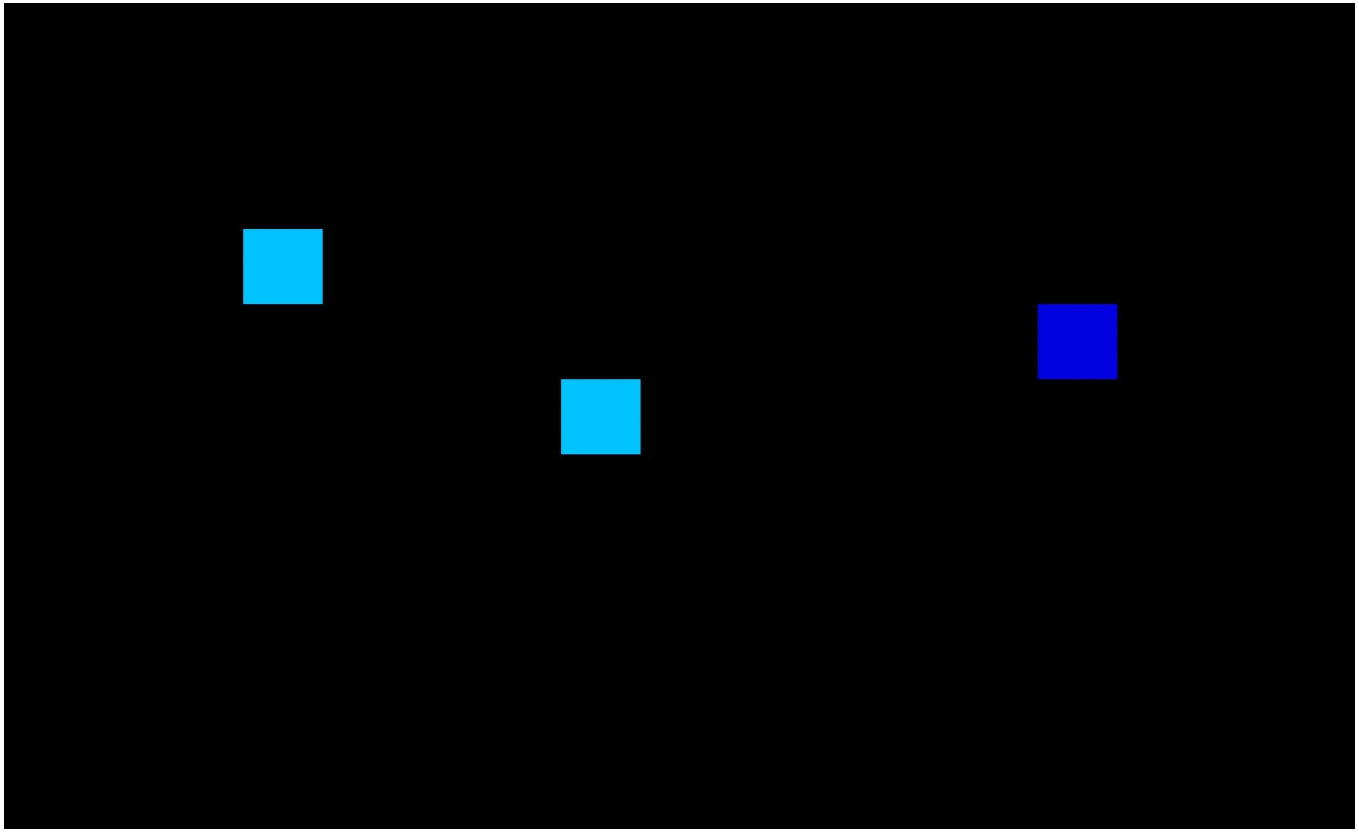
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



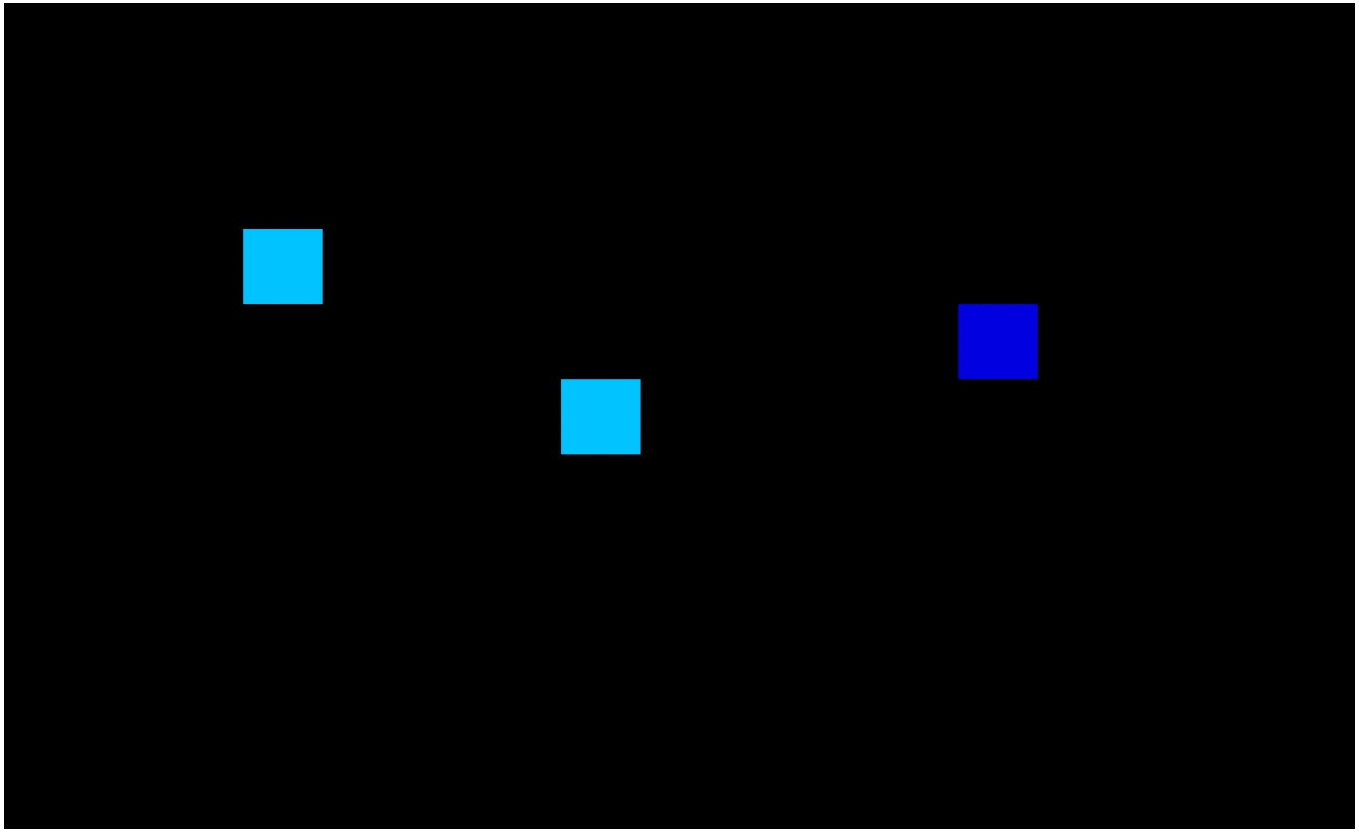
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



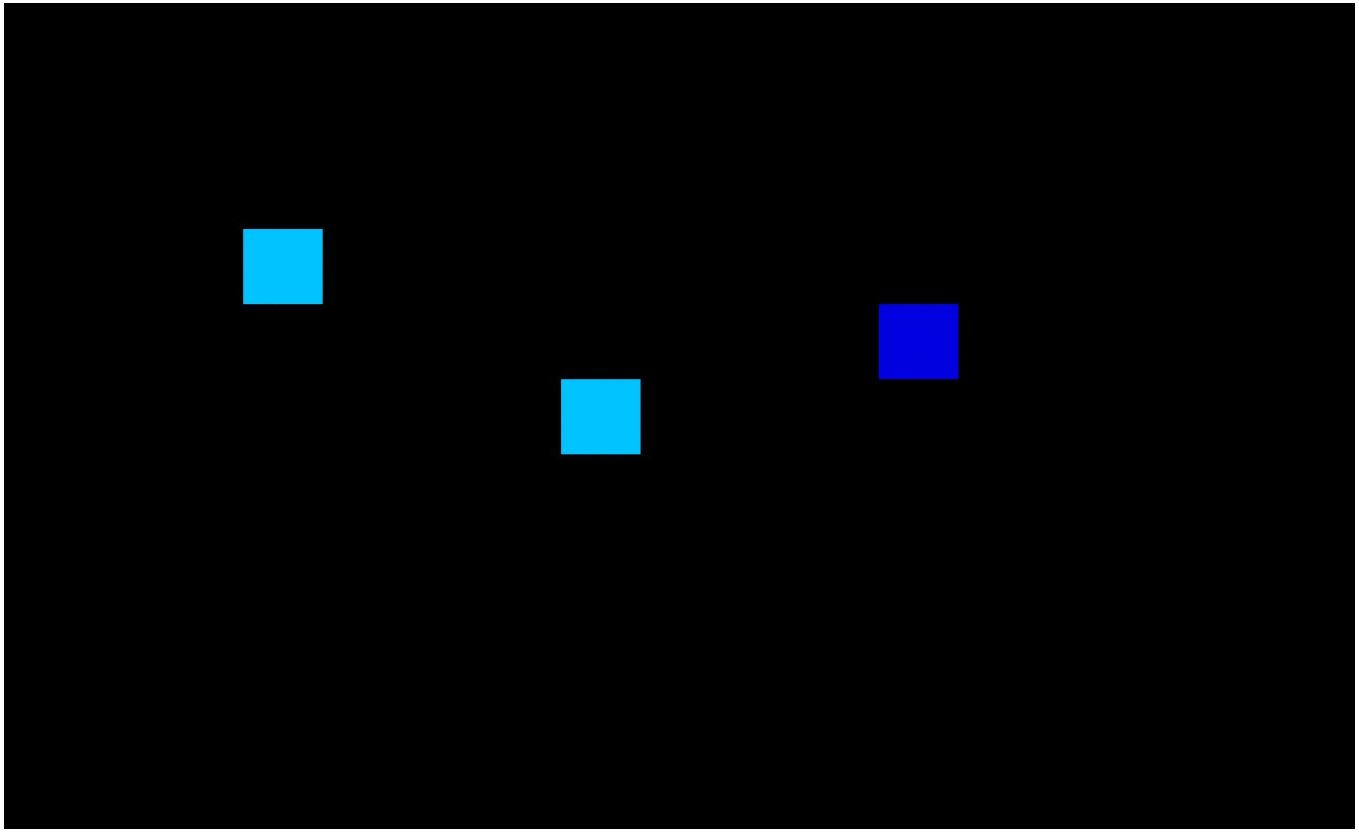
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



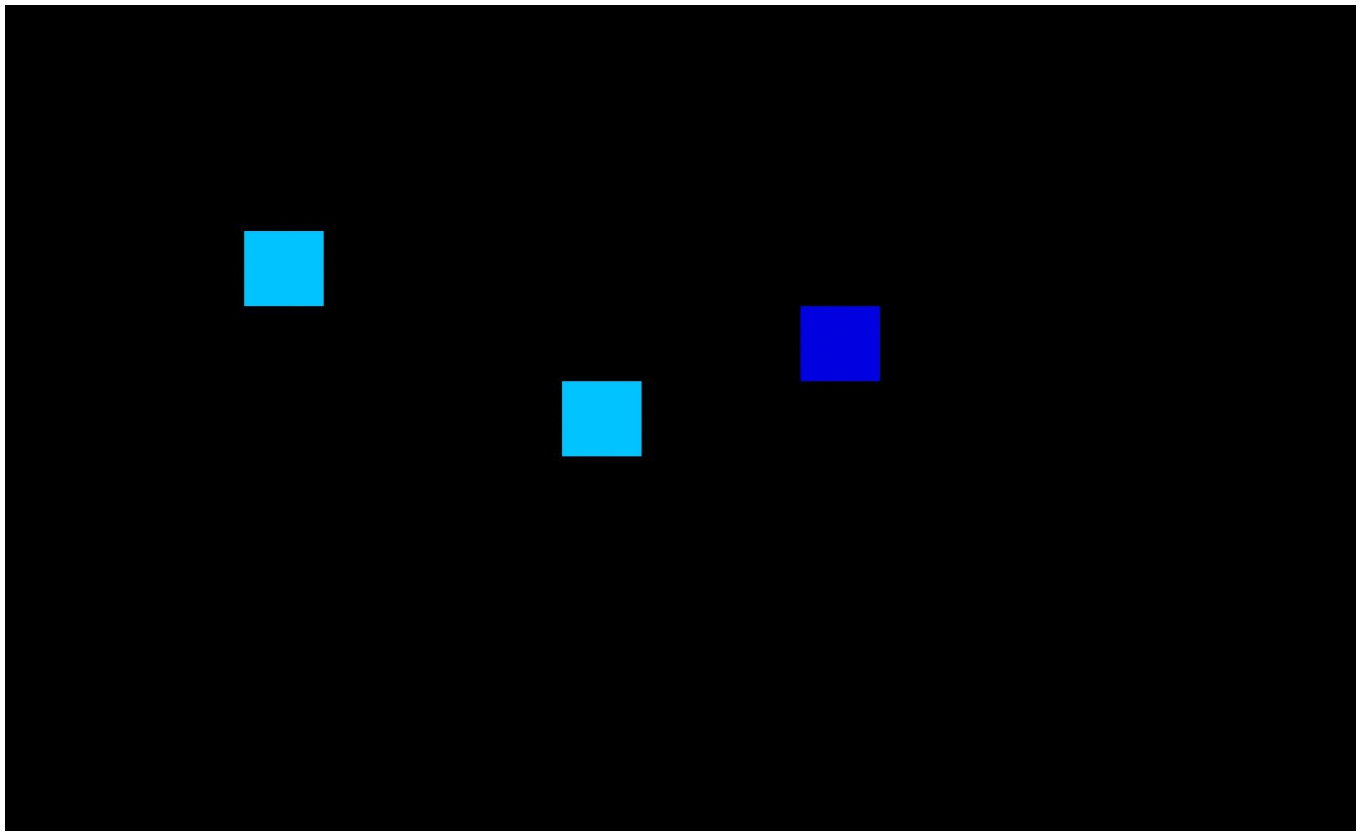
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



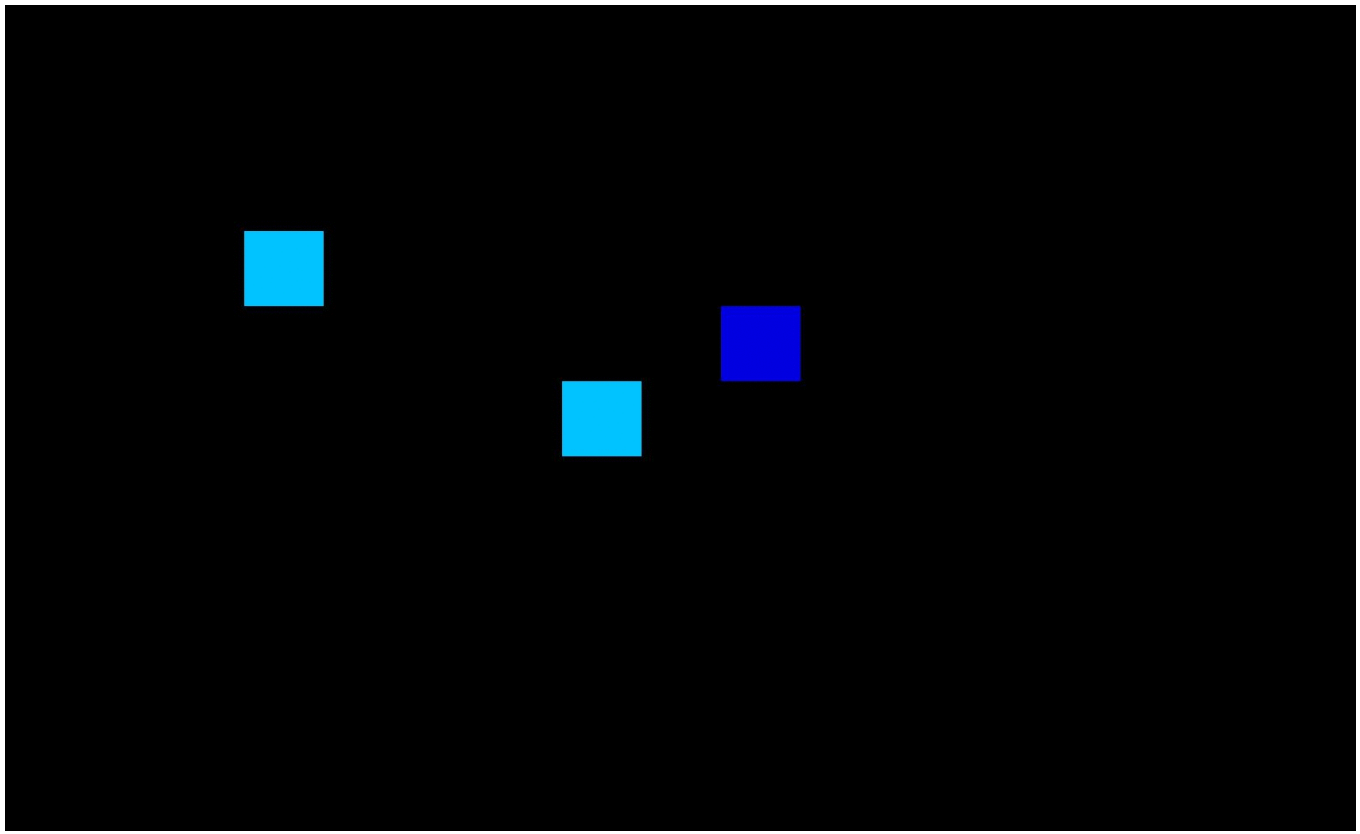
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



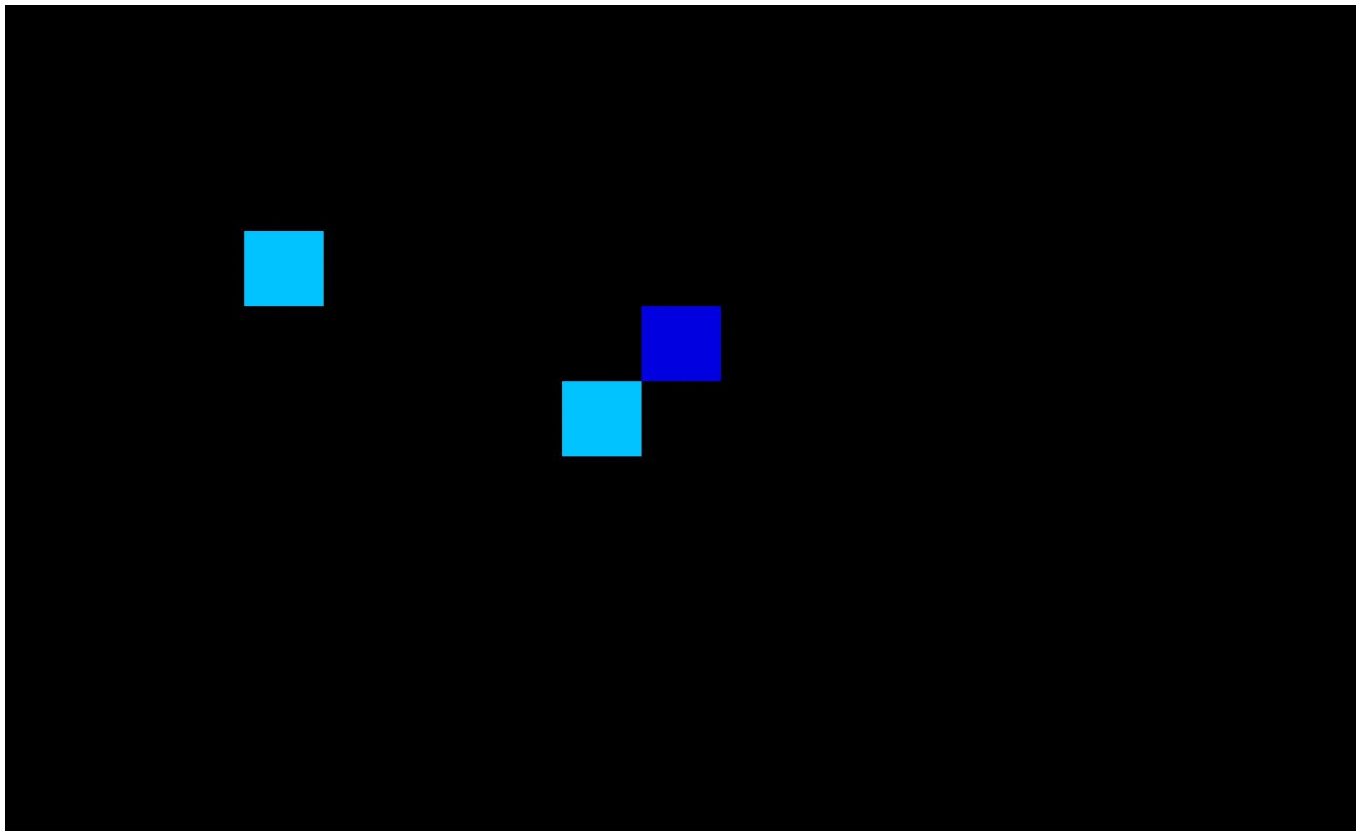
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



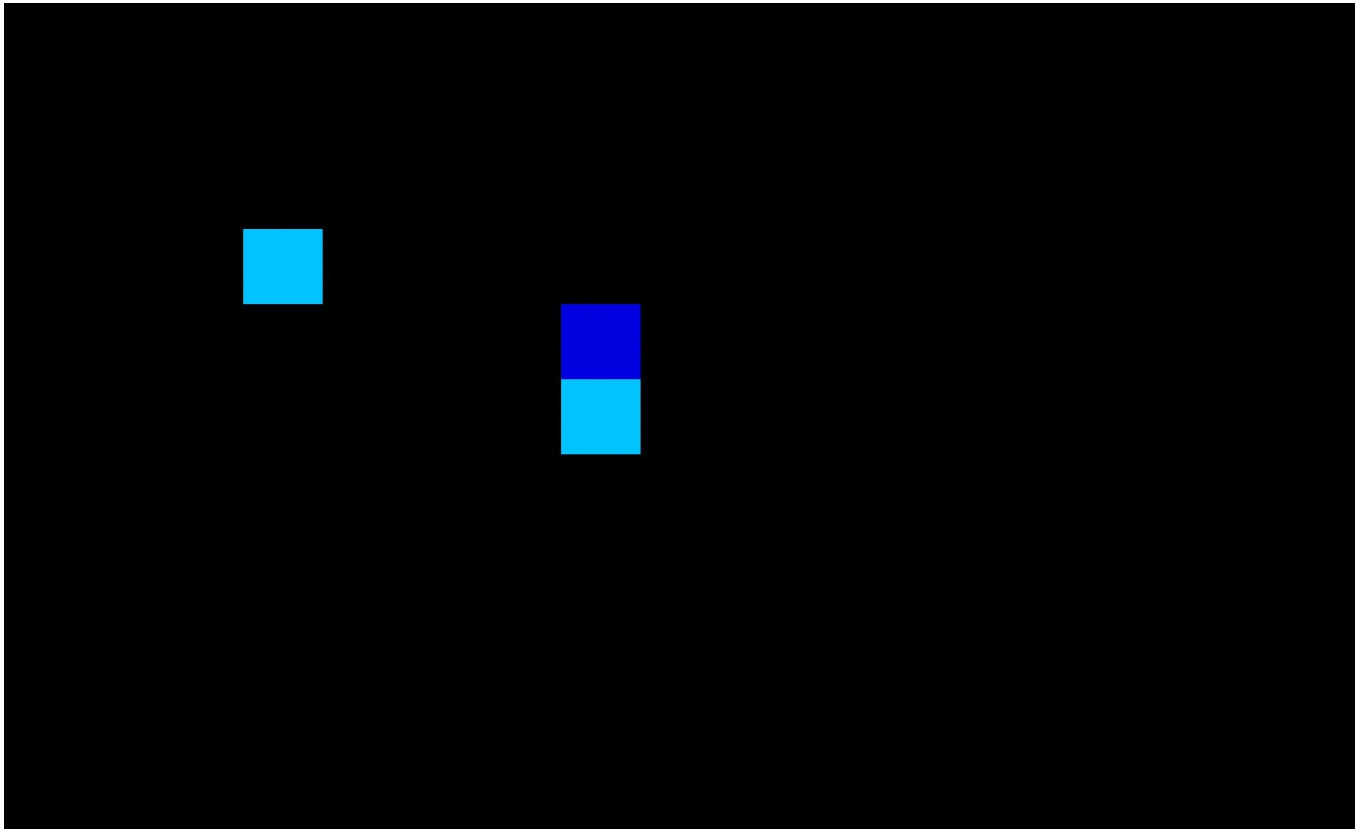
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



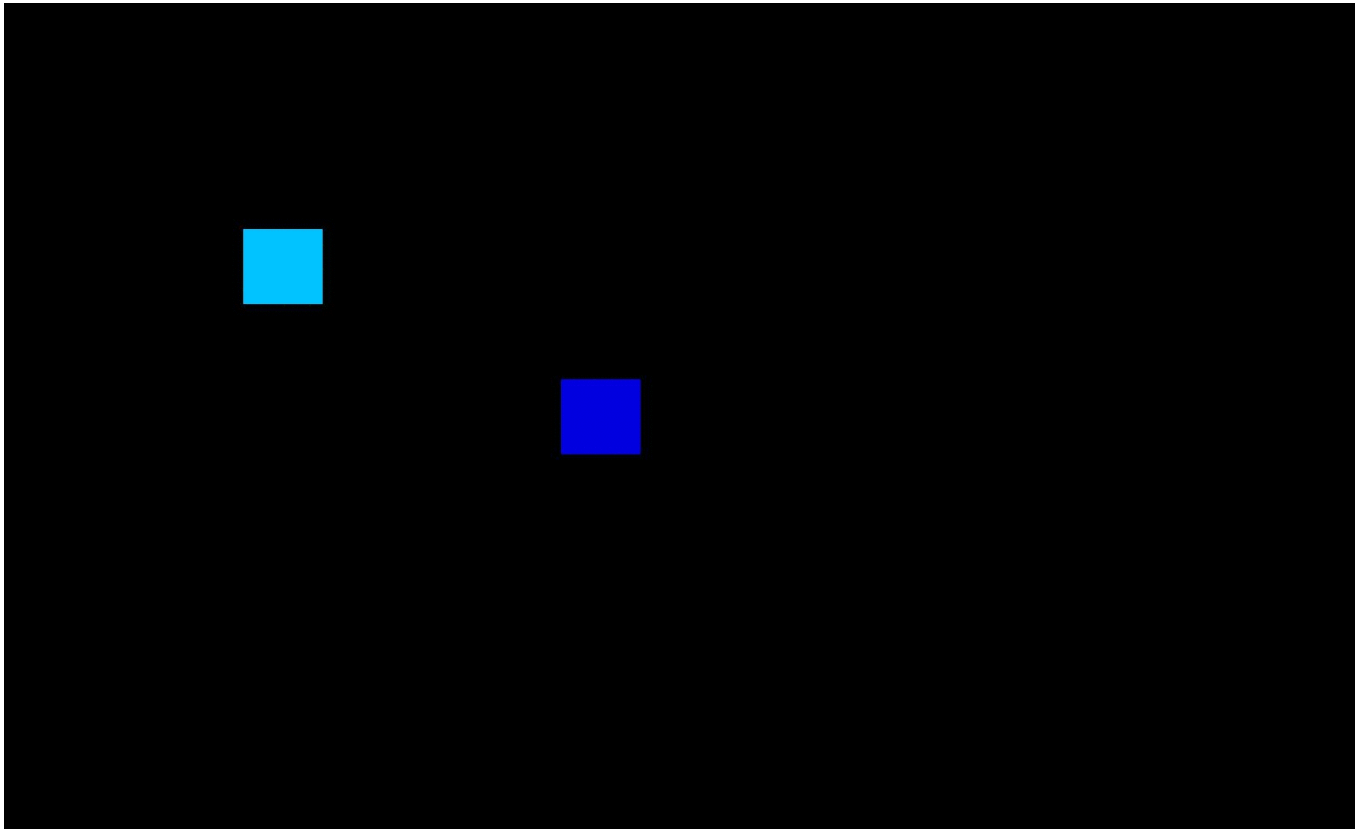
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



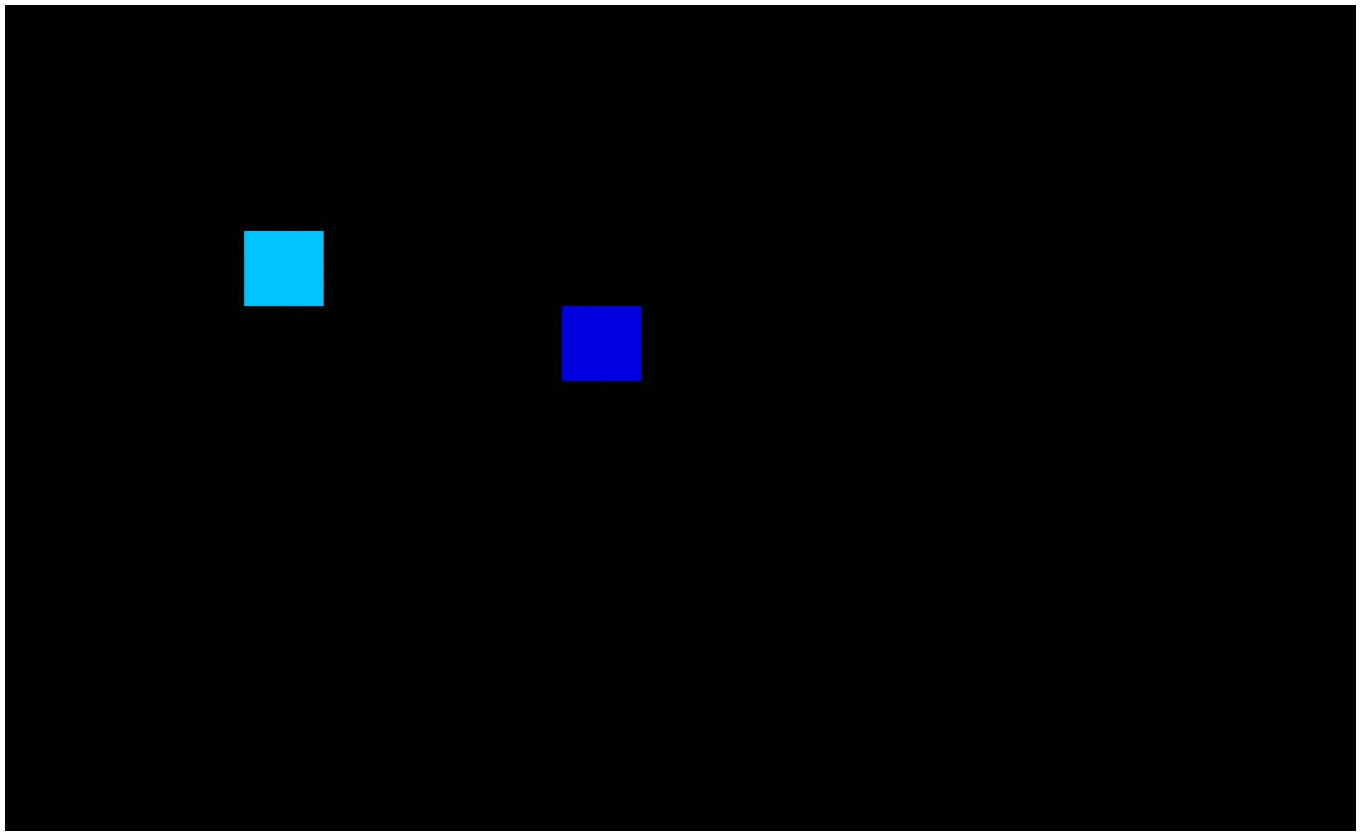
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



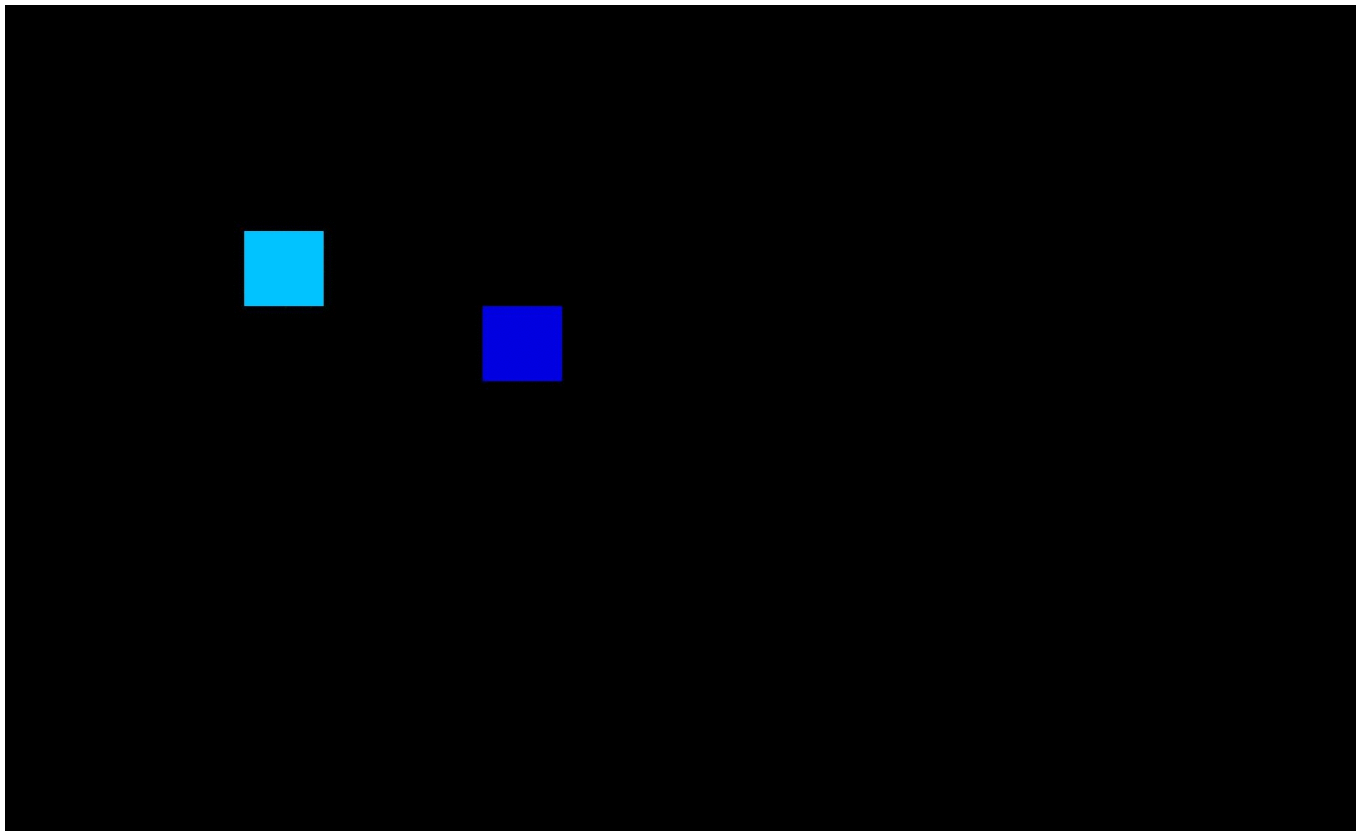
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



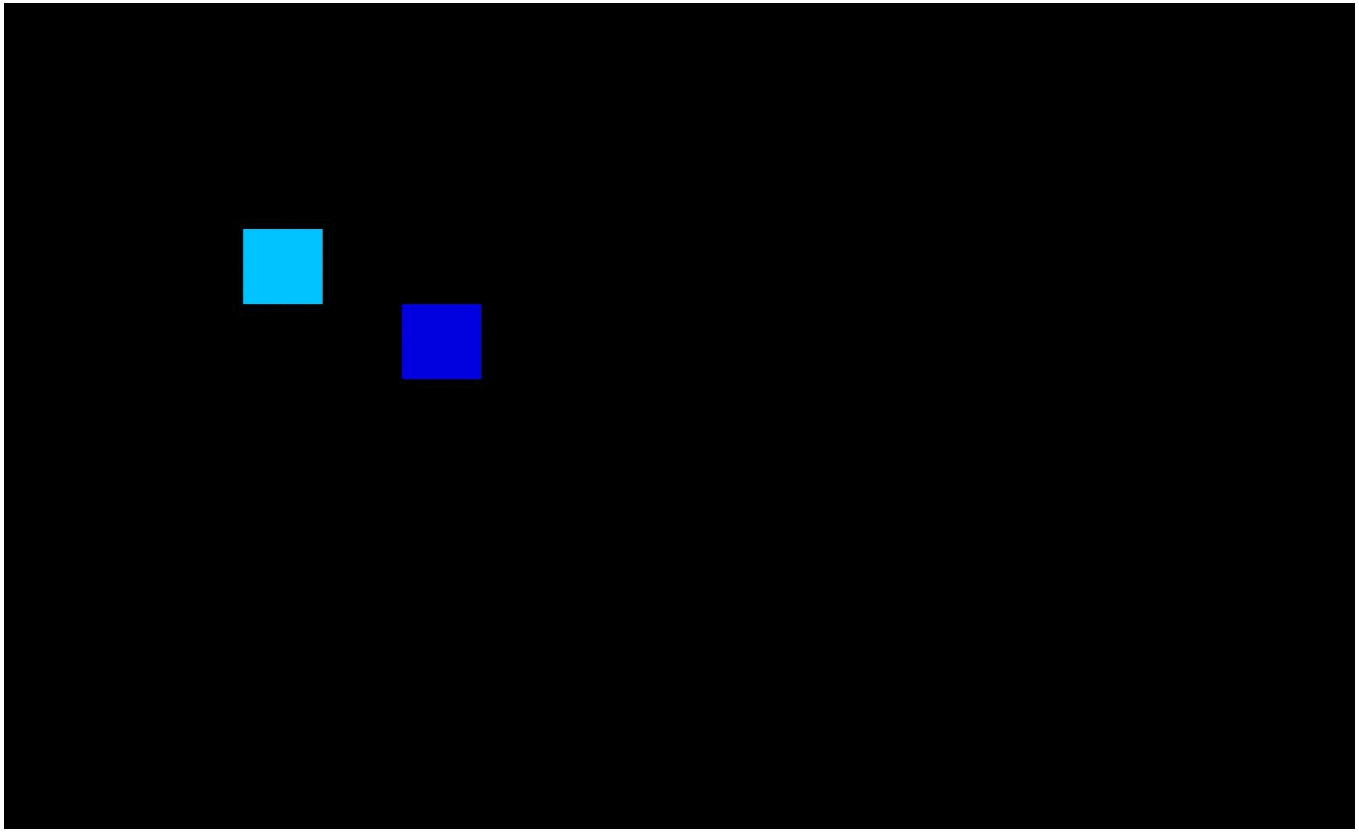
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



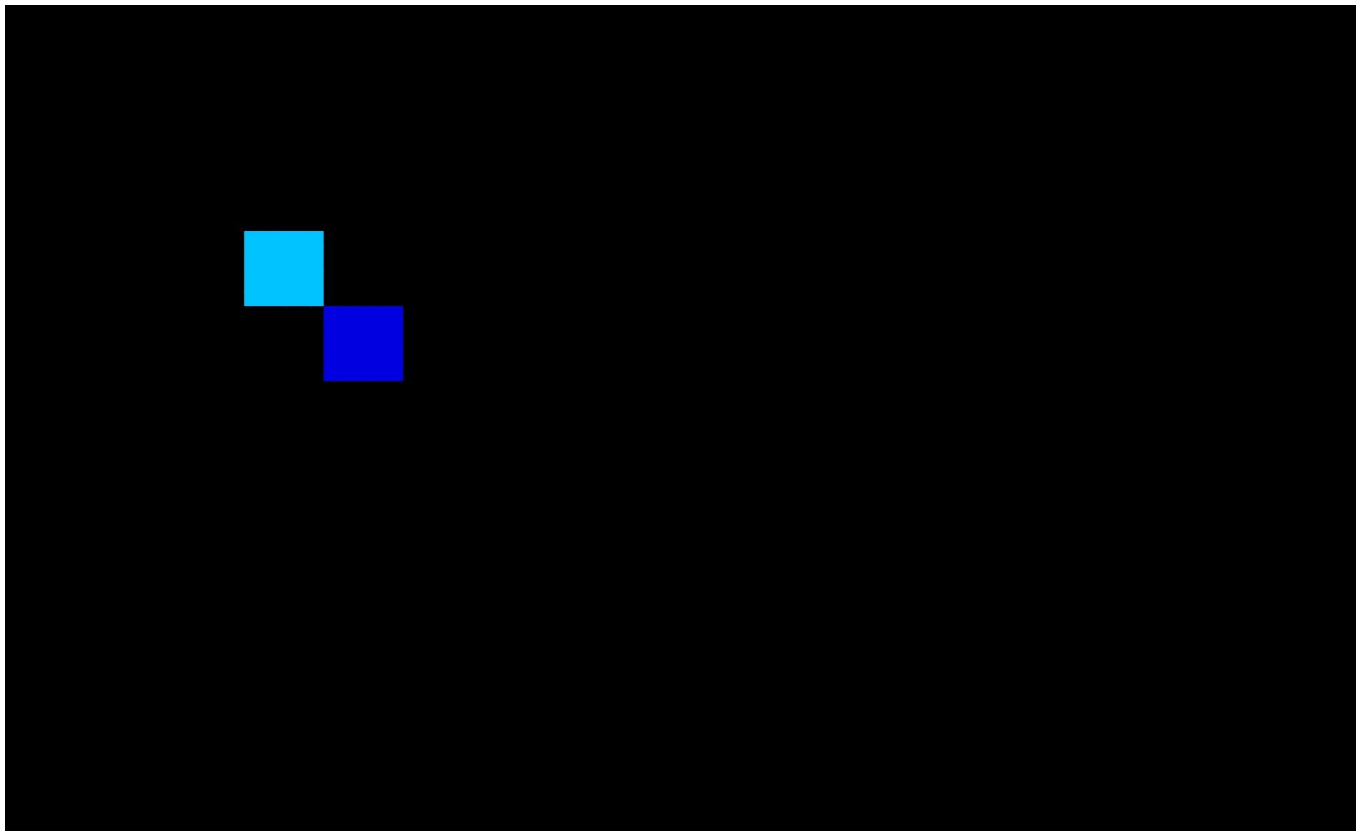
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



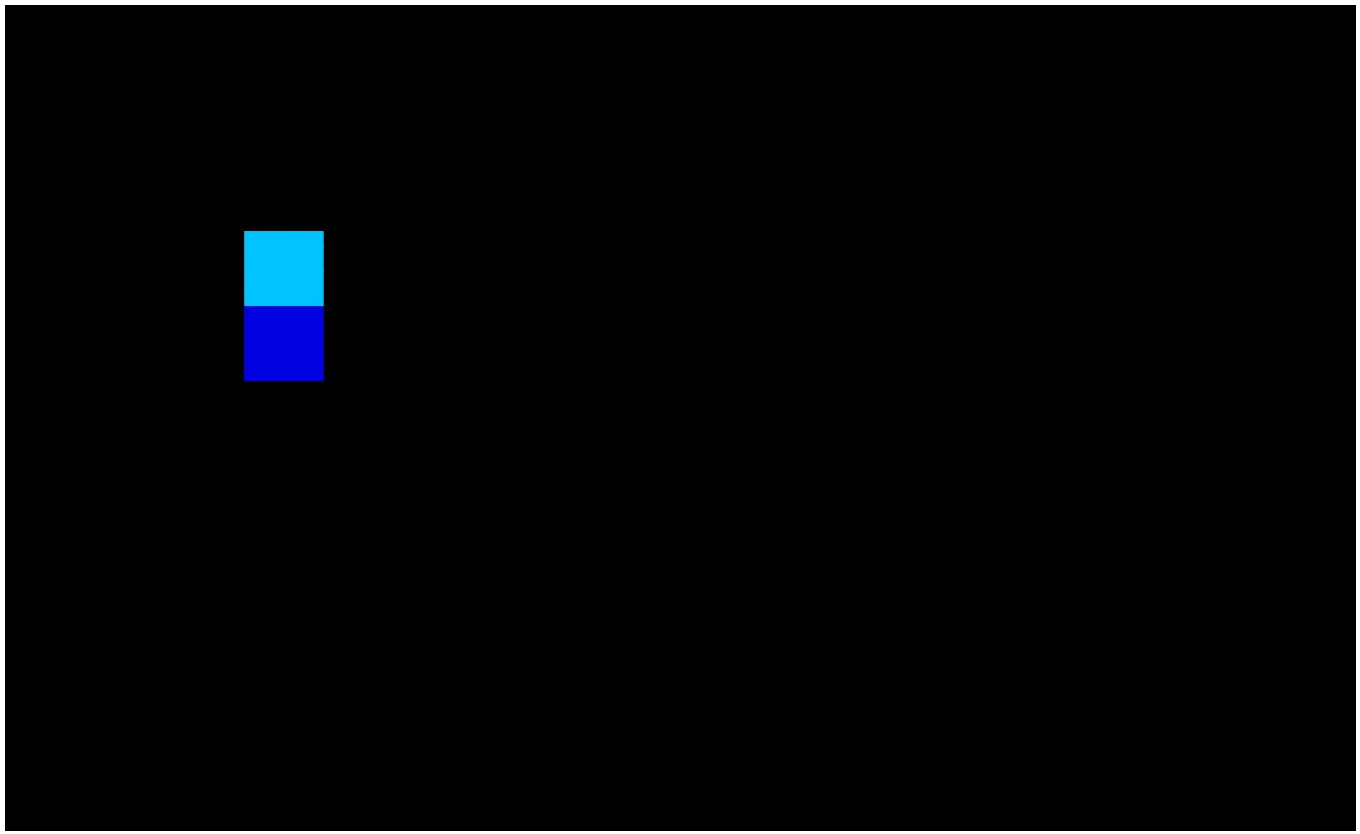
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



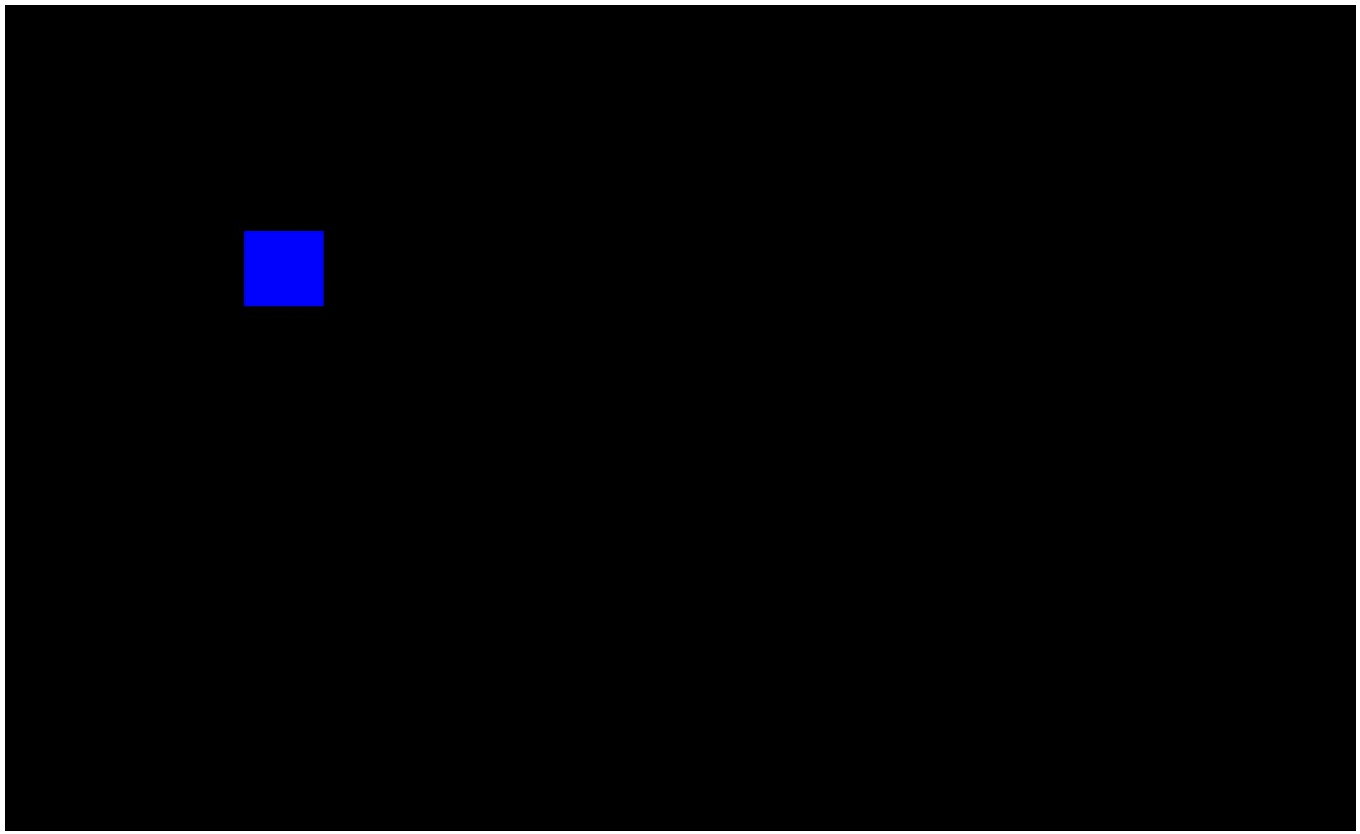
Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



Modelagem como um problema de Reinforcement Learning

- Exemplo de resultado gerado pela rede de [Clark, 2016] (Double Deep Q-Learning).



Modelagem como um problema de Reinforcement Learning

- Pode-se atrelar um *reward* a cada ações do agente.
- Na modelagem por [Clark, 2016] contêm os seguintes estados e recompensas:
  - 3 para todo **item coletado**.
  - -0.1 para **todo passo do agente**.
  - -1.5 para toda vez que o agente encostar em uma parede.
  - -1.5 para toda vez em que o agente não se movimentar.
- Desta forma o agente procurará **coletar os itens** de forma em que o caminho (ou número de passos) seja o **menor possível**.

Modelagem como um problema de Reinforcement Learning

- Pode-se atrelar um *reward* a cada ações do agente.
- Na modelagem por [Clark, 2016] contêm os seguintes estados e recompensas:
  - 3 para todo **item coletado**.
  - -0.1 para **todo passo do agente**.
  - -1.5 para toda vez que o agente encostar em uma parede.
  - -1.5 para toda vez em que o agente não se movimentar.
- Desta forma o agente procurará **coletar os itens** de forma em que o caminho (ou número de passos) seja o **menor possível**.

Modelagem como um problema de Reinforcement Learning

- Pode-se atrelar um *reward* a cada ações do agente.
- Na modelagem por [Clark, 2016] contêm os seguintes estados e recompensas:
  - 3 para todo **item coletado**.
  - -0.1 para **todo passo do agente**.
  - -1.5 para toda vez que o agente encostar em uma parede.
  - -1.5 para toda vez em que o agente não se movimentar.
- Desta forma o agente procurará **coletar os itens** de forma em que o caminho (ou número de passos) seja o **menor possível**.

Modelagem como um problema de Reinforcement Learning

- Pode-se atrelar um *reward* a cada ações do agente.
- Na modelagem por [Clark, 2016] contêm os seguintes estados e recompensas:
  - 3 para todo **item coletado**.
  - -0.1 para **todo passo do agente**.
  - -1.5 para toda vez que o agente encostar em uma parede.
  - -1.5 para toda vez em que o agente não se movimentar.
- Desta forma o agente procurará **coletar os itens** de forma em que o caminho (ou número de passos) seja o **menor possível**.

Modelagem como um problema de Reinforcement Learning

- Pode-se atrelar um *reward* a cada ações do agente.
- Na modelagem por [Clark, 2016] contêm os seguintes estados e recompensas:
  - 3 para todo **item coletado**.
  - -0.1 para **todo passo do agente**.
  - -1.5 para toda vez que o agente encostar em uma parede.
  - -1.5 para toda vez em que o agente não se movimentar.
- Desta forma o agente procurará **coletar os itens** de forma em que o caminho (ou número de passos) seja o **menor possível**.

Roteiro

Técnicas Aplicadas

- [Clark, 2016] resolveu o problema através das técnicas:
  - Double Deep Q-Learning.
  - Prioritized Replay.
  - Fixed Target Network.
- Para o trabalho prático resolveu-se através do *Stochastic Policy Gradient*.

Técnicas Aplicadas

- [Clark, 2016] resolveu o problema através das técnicas:
  - Double Deep Q-Learning.
  - Prioritized Replay.
  - Fixed Target Network.
- Para o trabalho prático resolveu-se através do *Stochastic Policy Gradient*.

Técnicas Aplicadas

- [Clark, 2016] resolveu o problema através das técnicas:
  - Double Deep Q-Learning.
  - Prioritized Replay.
  - Fixed Target Network.
- Para o trabalho prático resolveu-se através do *Stochastic Policy Gradient*.

Técnicas Aplicadas

- [Clark, 2016] resolveu o problema através das técnicas:
  - Double Deep Q-Learning.
  - Prioritized Replay.
  - Fixed Target Network.
- Para o trabalho prático resolveu-se através do *Stochastic Policy Gradient*.

Técnicas Aplicadas

- [Clark, 2016] resolveu o problema através das técnicas:
  - Double Deep Q-Learning.
  - Prioritized Replay.
  - Fixed Target Network.
- Para o trabalho prático resolveu-se através do *Stochastic Policy Gradient*.

Técnicas Aplicadas

- [Clark, 2016] resolveu o problema através das técnicas:
  - Double Deep Q-Learning.
  - Prioritized Replay.
  - Fixed Target Network.
- Para o trabalho prático resolveu-se através do *Stochastic Policy Gradient*.

Double Deep Q-Learning

- Deep Q-Learning:
  - Q-Learning na prática gasta uma enorme quantidade de memória e tempo computacional.
  - Uma forma de diminuir a entrada para obter os Q-values é utilizar redes neurais.
  - Utiliza-se então a saída da função de Q-Learning como target: $L = 1/2[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$
  - A primeira expressão $r + \gamma max_{a'} Q(s', a')$ refere-se a saída do target esperado (Q-Target) e a expressão $Q(s, a)$ refere-se a previsão feita pela rede neural.

Double Deep Q-Learning

- Deep Q-Learning:
  - Q-Learning na prática gasta uma enorme quantidade de memória e tempo computacional.
  - Uma forma de diminuir a entrada para obter os Q-values é utilizar redes neurais.
  - Utiliza-se então a saída da função de Q-Learning como target: $L = 1/2[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$
  - A primeira expressão $r + \gamma max_{a'} Q(s', a')$ refere-se a saída do target esperado (Q-Target) e a expressão $Q(s, a)$ refere-se a previsão feita pela rede neural.

Double Deep Q-Learning

- Deep Q-Learning:
  - Q-Learning na prática gasta uma enorme quantidade de memória e tempo computacional.
  - Uma forma de diminuir a entrada para obter os Q-values é utilizar redes neurais.
  - Utiliza-se então a saída da função de Q-Learning como target: $L = 1/2[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$
  - A primeira expressão $r + \gamma max_{a'} Q(s', a')$ refere-se a saída do target esperado (Q-Target) e a expressão $Q(s, a)$ refere-se a previsão feita pela rede neural.

Double Deep Q-Learning

- Deep Q-Learning:
  - Q-Learning na prática gasta uma enorme quantidade de memória e tempo computacional.
  - Uma forma de diminuir a entrada para obter os Q-values é utilizar redes neurais.
  - Utiliza-se então a saída da função de Q-Learning como target: $L = 1/2[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$
  - A primeira expressão $r + \gamma max_{a'} Q(s', a')$ refere-se a saída do target esperado (Q-Target) e a expressão $Q(s, a)$ refere-se a previsão feita pela rede neural.

Double Deep Q-Learning

- Deep Q-Learning:
  - Q-Learning na prática gasta uma enorme quantidade de memória e tempo computacional.
  - Uma forma de diminuir a entrada para obter os Q-values é utilizar redes neurais.
  - Utiliza-se então a saída da função de Q-Learning como target: $L = 1/2[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$
  - A primeira expressão $r + \gamma max_{a'} Q(s', a')$ refere-se a saída do target esperado (Q-Target) e a expressão $Q(s, a)$ refere-se a previsão feita pela rede neural.

Double Deep Q-Learning

- Deep Q-Learning:
  - Q-Learning na prática gasta uma enorme quantidade de memória e tempo computacional.
  - Uma forma de diminuir a entrada para obter os Q-values é utilizar redes neurais.
  - Utiliza-se então a saída da função de Q-Learning como target: $L = 1/2[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$
  - A primeira expressão $r + \gamma max_{a'} Q(s', a')$ refere-se a saída do target esperado (Q-Target) e a expressão $Q(s, a)$ refere-se a previsão feita pela rede neural.

Double Deep Q-Learning

- Deep Q-Learning:
  - Q-Learning na prática gasta uma enorme quantidade de memória e tempo computacional.
  - Uma forma de diminuir a entrada para obter os Q-values é utilizar redes neurais.
  - Utiliza-se então a saída da função de Q-Learning como target: $L = 1/2[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$
  - A primeira expressão $r + \gamma max_{a'} Q(s', a')$ refere-se a saída do target esperado (Q-Target) e a expressão $Q(s, a)$ refere-se a previsão feita pela rede neural.

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função max na fórmula para alcançar o Q-$target$.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função *max* na fórmula para alcançar o *Q-target*.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função *max* na fórmula para alcançar o *Q-target*.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função *max* na fórmula para alcançar o *Q-target*.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função *max* na fórmula para alcançar o *Q-target*.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função *max* na fórmula para alcançar o *Q-target*.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função *max* na fórmula para alcançar o *Q-target*.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Double Deep Q-Learning

- Double Deep Q-Learning:
  - Um dos problemas causados pelo Deep Q Learning é que o agente tem tendência de calcular exageradamente (maximizar o bias) da função Q, devido a função *max* na fórmula para alcançar o *Q-target*.
  - Para diminuir a instabilidade, a técnica de Double Deep Q-Learning (DDQN) propõe-se o uso de duas redes:
    - Uma rede primária Q com parâmetros θ' atualizados em que escolhe uma ação.
    - Uma rede alvo Q^2 com parâmetros θ antigos que gera o Q-target a partir da ação escolhida por Q.
  - Abaixo a nova equação para gerar o Q-Target:

$$\text{Q-Target} = r + \gamma * Q^2(s', argmax(Q(s', a, \theta'), \theta))$$

Deep Q-Learning

- Fixed Target Network:
  - Uma outra forma de diminuir as oscilações da rede é fixar os parâmetros w^- usados para alcançar o valor de target do Q-Learning, isto é $r + \gamma max_{a'} Q(s', a', w^-)$.
  - Por consequência, pode-se otimizar a média de erro quadrática entre a rede neural e o valor de target do Q-learning: $L(w) = E_{s,a,r,s' \sim D}[(r + \gamma max_{a'} Q(s', a', w^-) - Q(s, a, w)^2]$
  - E assim, periodicamente atualizar os parâmetros $w^- \leftarrow w$.

Deep Q-Learning

- Fixed Target Network:
  - Uma outra forma de diminuir as oscilações da rede é fixar os parâmetros w^- usados para alcançar o valor de target do Q-Learning, isto é $r + \gamma max_{a'} Q(s', a', w^-)$.
  - Por consequência, pode-se otimizar a média de erro quadrática entre a rede neural e o valor de target do Q-learning: $L(w) = E_{s,a,r,s' \sim D}[(r + \gamma max_{a'} Q(s', a', w^-) - Q(s, a, w)^2]$
  - E assim, periodicamente atualizar os parâmetros $w^- \leftarrow w$.

Deep Q-Learning

- Fixed Target Network:
  - Uma outra forma de diminuir as oscilações da rede é fixar os parâmetros w^- usados para alcançar o valor de target do Q-Learning, isto é $r + \gamma max_{a'} Q(s', a', w^-)$.
  - Por consequência, pode-se otimizar a média de erro quadrática entre a rede neural e o valor de target do Q-learning: $L(w) = E_{s,a,r,s' \sim D}[(r + \gamma max_{a'} Q(s', a', w^-) - Q(s, a, w)^2]$
  - E assim, periodicamente atualizar os parâmetros $w^- \leftarrow w$.

Deep Q-Learning

- Fixed Target Network:
  - Uma outra forma de diminuir as oscilações da rede é fixar os parâmetros w^- usados para alcançar o valor de target do Q-Learning, isto é $r + \gamma max_{a'} Q(s', a', w^-)$.
  - Por consequência, pode-se otimizar a média de erro quadrática entre a rede neural e o valor de target do Q-learning: $L(w) = E_{s,a,r,s' \sim D}[(r + \gamma max_{a'} Q(s', a', w^-) - Q(s, a, w)^2]$
  - E assim, periodicamente atualizar os parâmetros $w^- \leftarrow w$.

Deep Q-Learning

- Fixed Target Network:
  - Uma outra forma de diminuir as oscilações da rede é fixar os parâmetros w^- usados para alcançar o valor de target do Q-Learning, isto é $r + \gamma max_{a'} Q(s', a', w^-)$.
  - Por consequência, pode-se otimizar a média de erro quadrática entre a rede neural e o valor de target do Q-learning: $L(w) = E_{s,a,r,s' \sim D}[(r + \gamma max_{a'} Q(s', a', w^-) - Q(s, a, w)^2]$
  - E assim, periodicamente atualizar os parâmetros $w^- \leftarrow w$.

Prioritized Replay

- Uma outra forma de diminuir a instabilidade da rede é utilizar o *Prioritized Replay*.
- Durante o treinamento pode-se guardar toda a experiência $< s, a, r, s' >$ na memória de forma aleatória.
- Ao invés de utilizar minibatches de forma aleatória, uma outra estratégia seria recorrer aos minibatches que façam a rede aprender mais.
- A ideia é assinalar para cada replay a probabilidade deste de ser adicionado ao minibatch.
- A probabilidade de cada replay pode ser determinada pelo erro absoluto da rede neural.

Prioritized Replay

- Uma outra forma de diminuir a instabilidade da rede é utilizar o *Prioritized Replay*.
- Durante o treinamento pode-se guardar toda a experiência $< s, a, r, s' >$ na memória de forma aleatória.
- Ao invés de utilizar minibatches de forma aleatória, uma outra estratégia seria recorrer aos minibatches que façam a rede aprender mais.
- A ideia é assinalar para cada replay a probabilidade deste de ser adicionado ao minibatch.
- A probabilidade de cada replay pode ser determinada pelo erro absoluto da rede neural.

Prioritized Replay

- Uma outra forma de diminuir a instabilidade da rede é utilizar o *Prioritized Replay*.
- Durante o treinamento pode-se guardar toda a experiência $< s, a, r, s' >$ na memória de forma aleatória.
- Ao invés de utilizar minibatches de forma aleatória, uma outra estratégia seria recorrer aos minibatches que façam a rede aprender mais.
- A ideia é assinalar para cada replay a probabilidade deste de ser adicionado ao minibatch.
- A probabilidade de cada replay pode ser determinada pelo erro absoluto da rede neural.

Prioritized Replay

- Uma outra forma de diminuir a instabilidade da rede é utilizar o *Prioritized Replay*.
- Durante o treinamento pode-se guardar toda a experiência $< s, a, r, s' >$ na memória de forma aleatória.
- Ao invés de utilizar minibatches de forma aleatória, uma outra estratégia seria recorrer aos minibatches que façam a rede aprender mais.
- A ideia é assinalar para cada replay a probabilidade deste de ser adicionado ao minibatch.
- A probabilidade de cada replay pode ser determinada pelo erro absoluto da rede neural.

Prioritized Replay

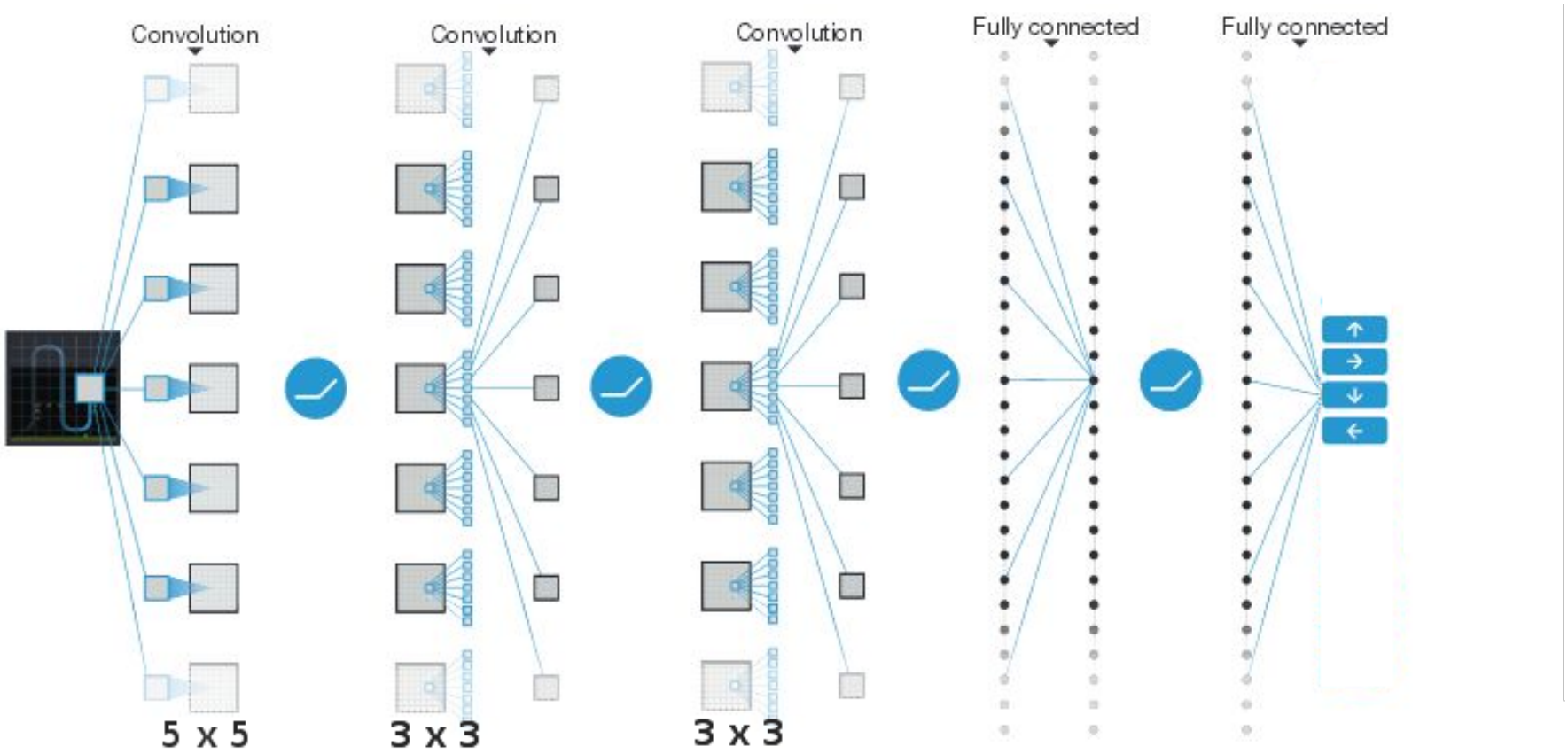
- Uma outra forma de diminuir a instabilidade da rede é utilizar o *Prioritized Replay*.
- Durante o treinamento pode-se guardar toda a experiência $< s, a, r, s' >$ na memória de forma aleatória.
- Ao invés de utilizar minibatches de forma aleatória, uma outra estratégia seria recorrer aos minibatches que façam a rede aprender mais.
- A ideia é assinalar para cada replay a probabilidade deste de ser adicionado ao minibatch.
- A probabilidade de cada replay pode ser determinada pelo erro absoluto da rede neural.

Prioritized Replay

- Uma outra forma de diminuir a instabilidade da rede é utilizar o *Prioritized Replay*.
- Durante o treinamento pode-se guardar toda a experiência $< s, a, r, s' >$ na memória de forma aleatória.
- Ao invés de utilizar minibatches de forma aleatória, uma outra estratégia seria recorrer aos minibatches que façam a rede aprender mais.
- A ideia é assinalar para cada replay a probabilidade deste de ser adicionado ao minibatch.
- A probabilidade de cada replay pode ser determinada pelo erro absoluto da rede neural.

Deep Q-Learning

- Rede implementada por [Clark, 2016].



Stochastic Policy Gradient

- Sabe-se algoritmos de Policy gradients **funcionam melhor** que Q-Learning quando tem seus parâmetros **bem ajustados**. [Karpathy, 2016]
- Diferente de Q-Learning, métodos de policy gradients tentam aprender a política de **forma direta**, são *on-policy*.
- Policy Gradients interpreta a função de política como uma **distribuição de probabilidade em ações** $q(a, s; \theta)$.
  - Portanto, define a **probabilidade da próxima ação** ser a dado a entrada s, parametrizada por θ.
  - Os parâmetros *theta* **podem ser aproximados** por uma rede neural.

Stochastic Policy Gradient

- Sabe-se algoritmos de Policy gradients **funcionam melhor** que Q-Learning quando tem seus parâmetros **bem ajustados**. [Karpathy, 2016]
- Diferente de Q-Learning, métodos de policy gradients tentam aprender a política de **forma direta**, são *on-policy*.
- Policy Gradients interpreta a função de política como uma **distribuição de probabilidade em ações** $q(a, s; \theta)$.
  - Portanto, define a **probabilidade da próxima ação** ser a dado a entrada s, parametrizada por θ.
  - Os parâmetros *theta* **podem ser aproximados** por uma rede neural.

Stochastic Policy Gradient

- Sabe-se algoritmos de Policy gradients **funcionam melhor** que Q-Learning quando tem seus parâmetros **bem ajustados**. [Karpathy, 2016]
- Diferente de Q-Learning, métodos de policy gradients tentam aprender a política de **forma direta**, são *on-policy*.
- Policy Gradients interpreta a função de política como uma **distribuição de probabilidade em ações** $q(a, s; \theta)$.
  - Portanto, define a **probabilidade da próxima ação** ser a dado a entrada s, parametrizada por θ.
  - Os parâmetros *theta* **podem ser aproximados** por uma rede neural.

Stochastic Policy Gradient

- Sabe-se algoritmos de Policy gradients **funcionam melhor** que Q-Learning quando tem seus parâmetros **bem ajustados**. [Karpathy, 2016]
- Diferente de Q-Learning, métodos de policy gradients tentam aprender a política de **forma direta**, são *on-policy*.
- Policy Gradients interpreta a função de política como uma **distribuição de probabilidade em ações** $q(a, s; \theta)$.
  - Portanto, define a **probabilidade da próxima ação** ser a dado a entrada s, parametrizada por θ.
  - Os parâmetros *theta* **podem ser aproximados** por uma rede neural.

Stochastic Policy Gradient

- Sabe-se algoritmos de Policy gradients **funcionam melhor** que Q-Learning quando tem seus parâmetros **bem ajustados**. [Karpathy, 2016]
- Diferente de Q-Learning, métodos de policy gradients tentam aprender a política de **forma direta**, são *on-policy*.
- Policy Gradients interpreta a função de política como uma **distribuição de probabilidade em ações** $q(a, s; \theta)$.
  - Portanto, define a **probabilidade da próxima ação** ser a dado a entrada s, parametrizada por θ.
  - Os parâmetros *theta* **podem ser aproximados** por uma rede neural.

Stochastic Policy Gradient

- Sabe-se algoritmos de Policy gradients **funcionam melhor** que Q-Learning quando tem seus parâmetros **bem ajustados**. [Karpathy, 2016]
- Diferente de Q-Learning, métodos de policy gradients tentam aprender a política de **forma direta**, são *on-policy*.
- Policy Gradients interpreta a função de política como uma **distribuição de probabilidade em ações** $q(a, s; \theta)$.
  - Portanto, define a **probabilidade da próxima ação** ser a dado a entrada s, parametrizada por θ.
  - Os parâmetros *theta* **podem ser aproximados** por uma rede neural.

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Stochastic Policy Gradient

- Na aprendizagem supervisionada tradicional o objetivo é maximizar $\sum_i log\ p(y_i|x_i)$.
  - Onde x_i é a entrada e y_i o rótulo.
- Em Policy Gradients, a função de perda é definida de **forma muito parecida**, porém há duas principais diferenças:
  - **Não se sabe** sabe o rótulo correto y_i. Então é atribuído uma ação a_i como **rótulo falso**, para toda vez que a política receber a entrada x_i.
  - Modula-se a função de perda de **forma multiplicativa** baseado no resultado final, uma vez que pretende-se que **haja acréscimo** da *log* probabilidade das ações que funcionaram e **descrescimento** das que não.
- De forma resumida, deseja se maximizar a função de perda no stochastic policy gradient de forma:

$$\sum_i r_i\ log\ p(a_i|x_i)$$

Roteiro

Código Base

- A implementação da rede e função de perda foi implementada com base no código de [Dirko Coetsee, 2017].
- O policy gradients foi implementado considerando o ambiente de [Clark, 2016] como black box.
  - Única mudança: rewards e número máximo de repetições de passos do agente.
- Para implementação utilizou-se uma rede densa com oito camadas escondidas.

Código Base

- A implementação da rede e função de perda foi implementada com base no código de [Dirko Coetsee, 2017].
- O policy gradients foi implementado considerando o ambiente de [Clark, 2016] como black box.
  - Única mudança: rewards e número máximo de repetições de passos do agente.
- Para implementação utilizou-se uma rede densa com oito camadas escondidas.

Código Base

- A implementação da rede e função de perda foi implementada com base no código de [Dirko Coetsee, 2017].
- O policy gradients foi implementado considerando o ambiente de [Clark, 2016] como black box.
  - Única mudança: rewards e número máximo de repetições de passos do agente.
- Para implementação utilizou-se uma rede densa com oito camadas escondidas.

Código Base

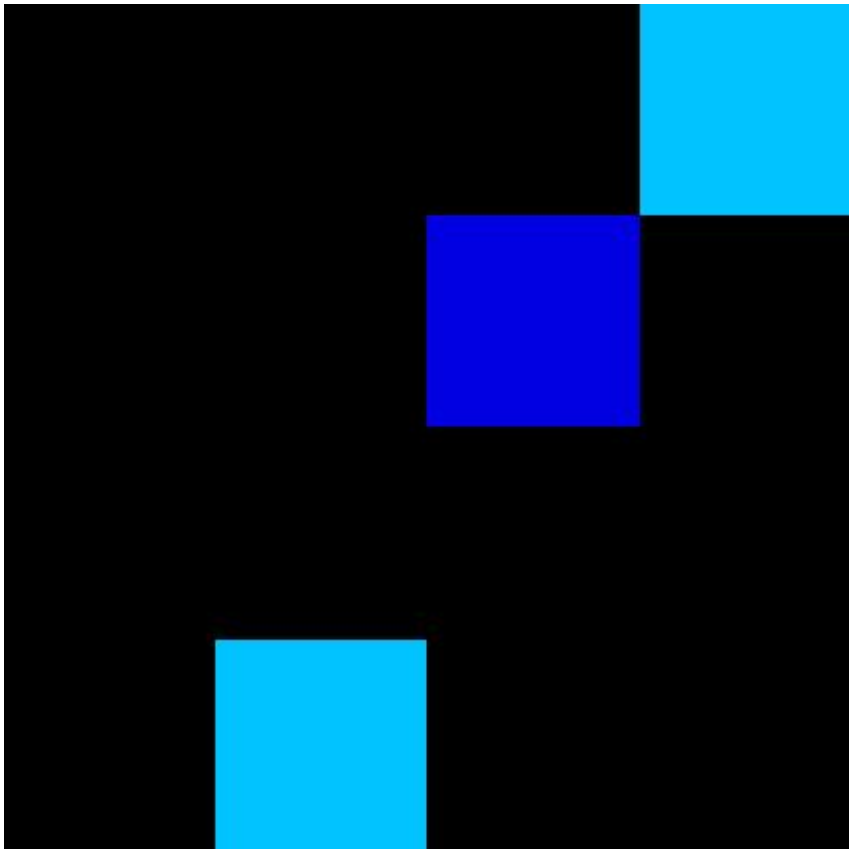
- A implementação da rede e função de perda foi implementada com base no código de [Dirko Coetsee, 2017].
- O policy gradients foi implementado considerando o ambiente de [Clark, 2016] como black box.
  - Única mudança: rewards e número máximo de repetições de passos do agente.
- Para implementação utilizou-se uma rede densa com oito camadas escondidas.

Código Base

- A implementação da rede e função de perda foi implementada com base no código de [Dirko Coetsee, 2017].
- O policy gradients foi implementado considerando o ambiente de [Clark, 2016] como black box.
  - Única mudança: rewards e número máximo de repetições de passos do agente.
- Para implementação utilizou-se uma rede densa com oito camadas escondidas.

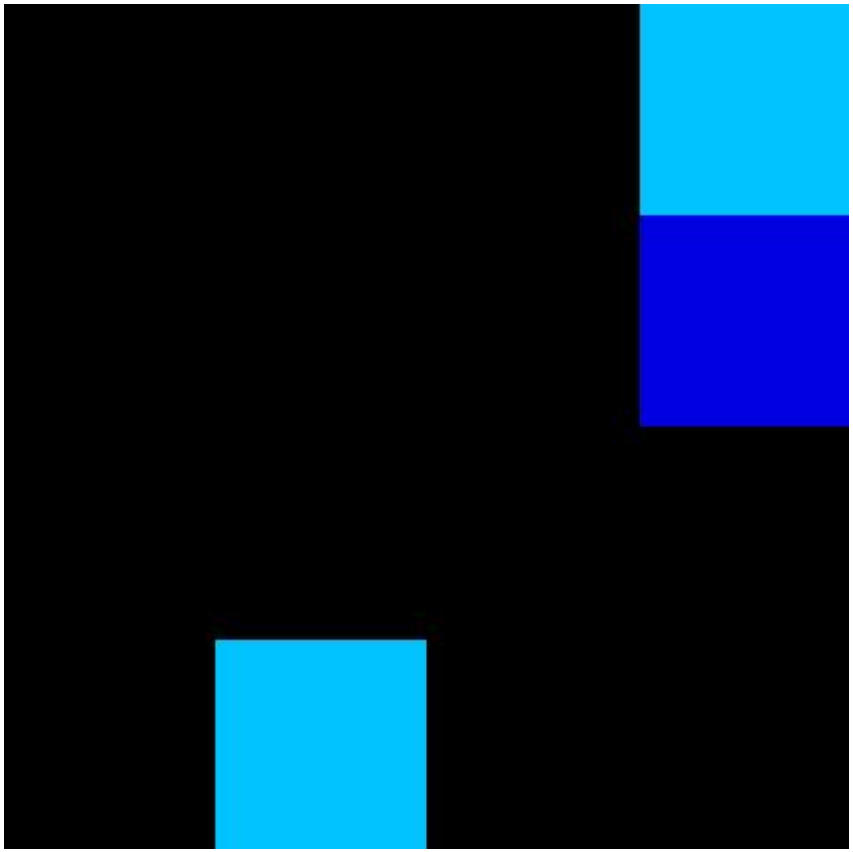
Máximo Local

- Exemplo de Máximo Local.



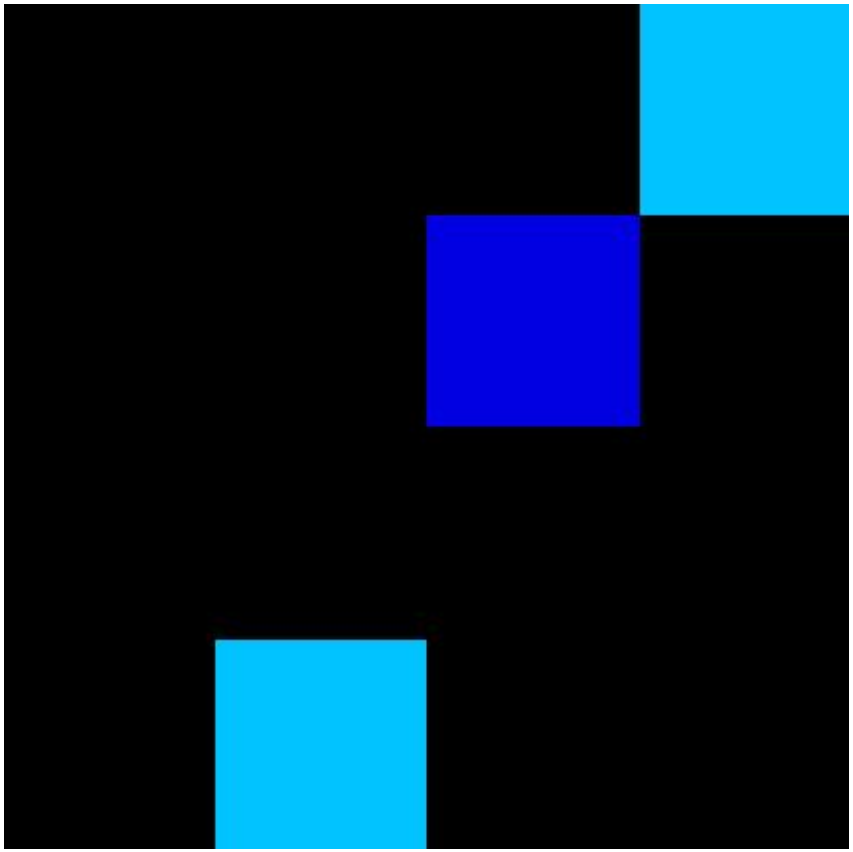
Máximo Local

- Exemplo de Máximo Local.



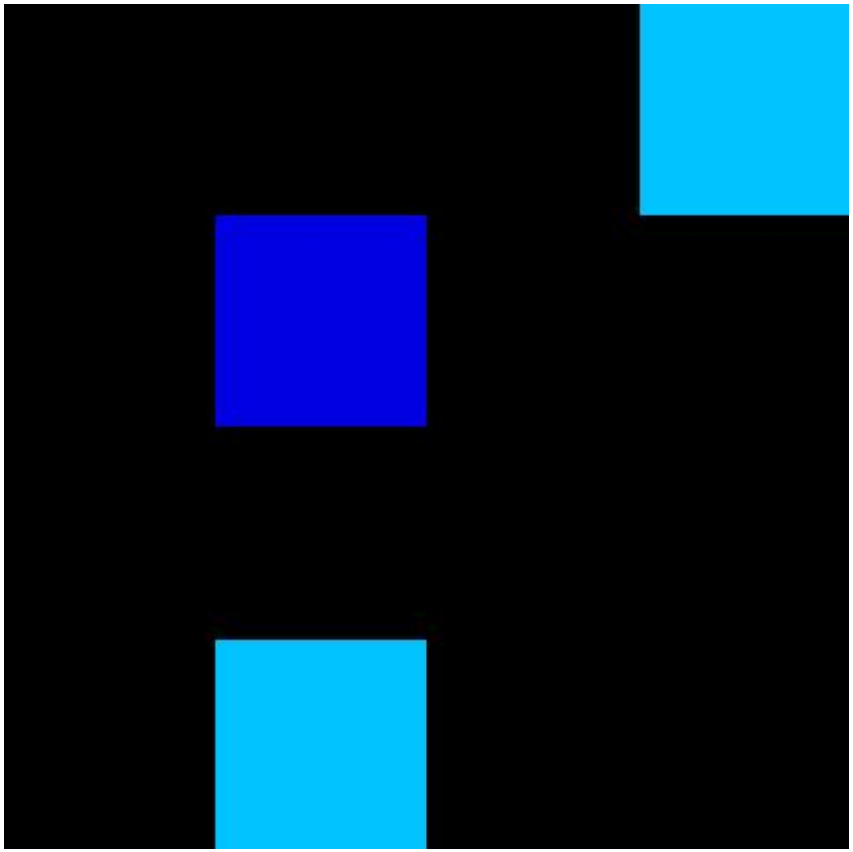
Máximo Local

- Exemplo de Máximo Local.



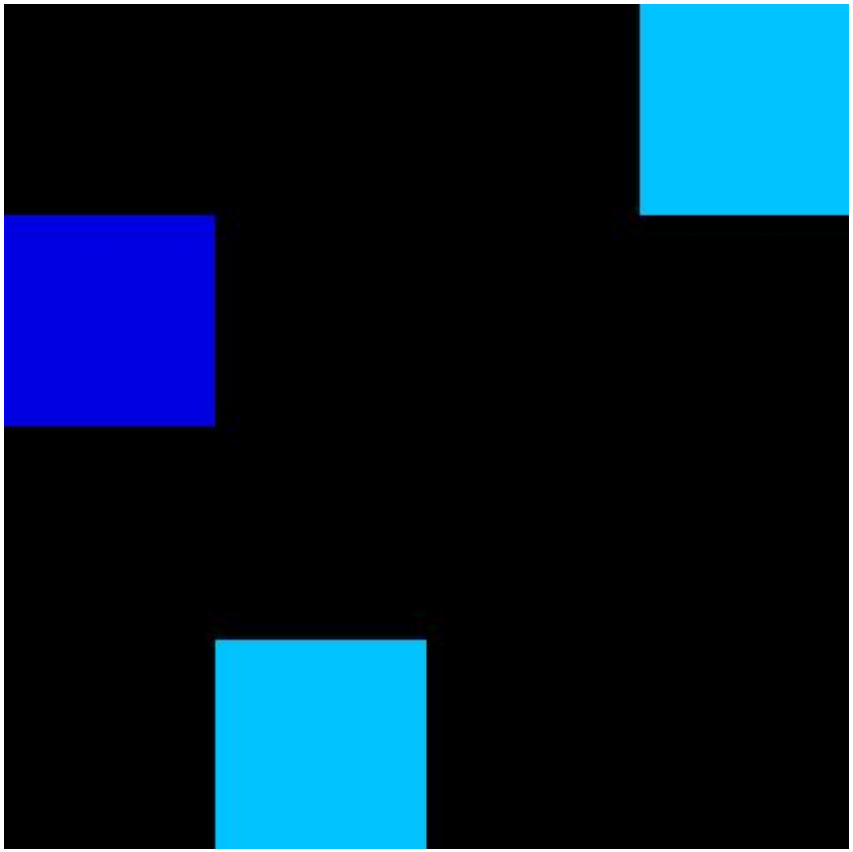
Máximo Local

- Exemplo de Máximo Local.



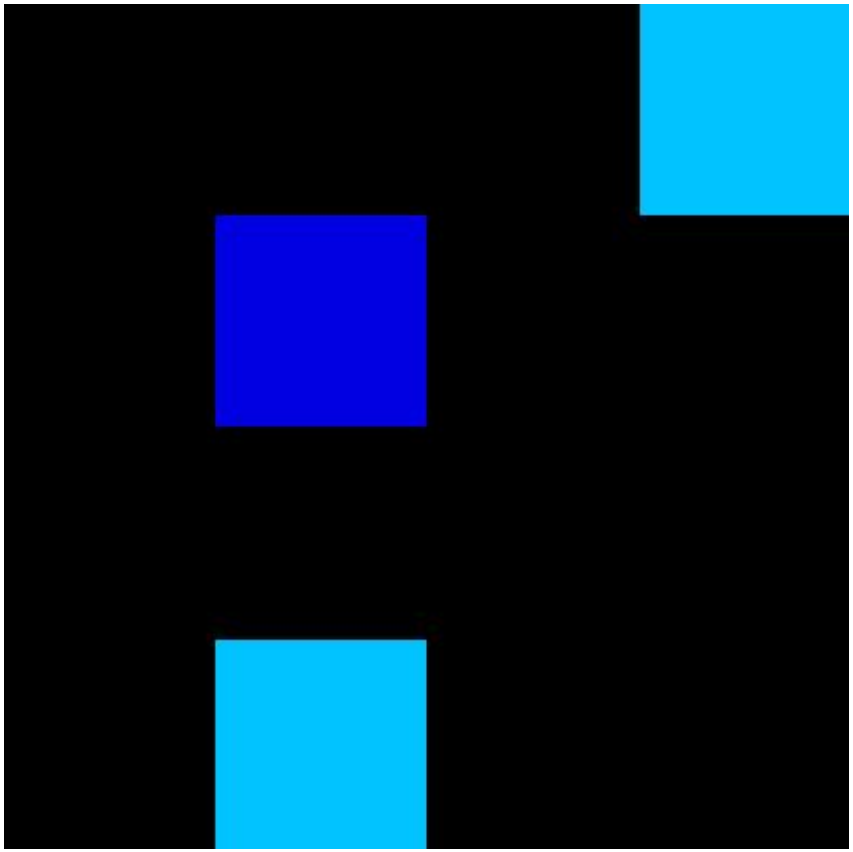
Máximo Local

- Exemplo de Máximo Local.



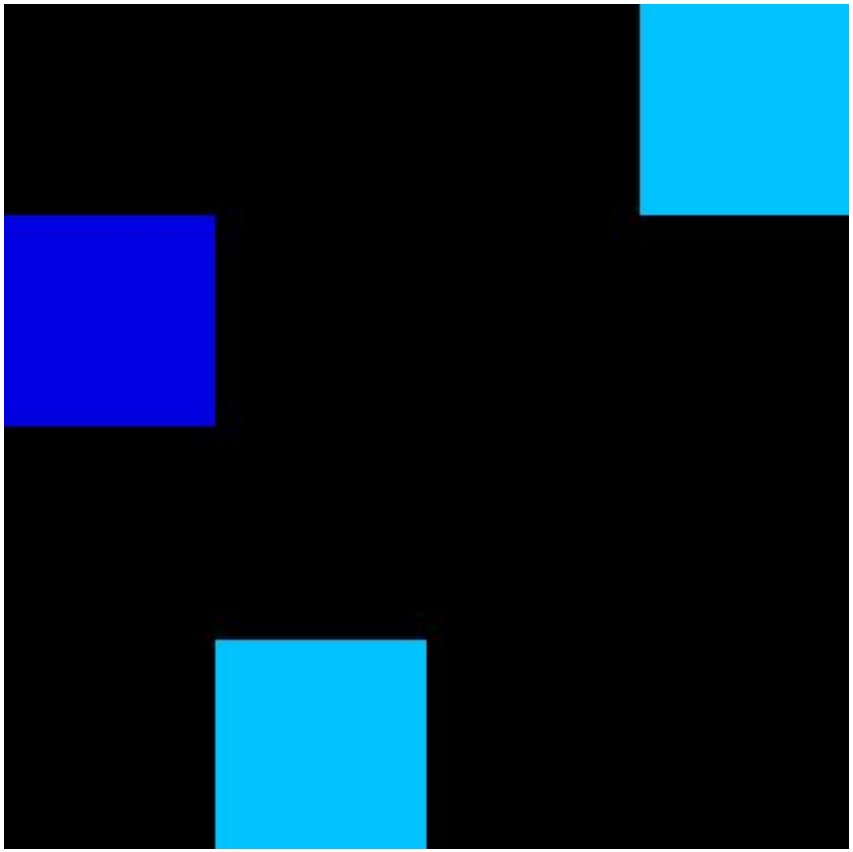
Máximo Local

- Exemplo de Máximo Local.



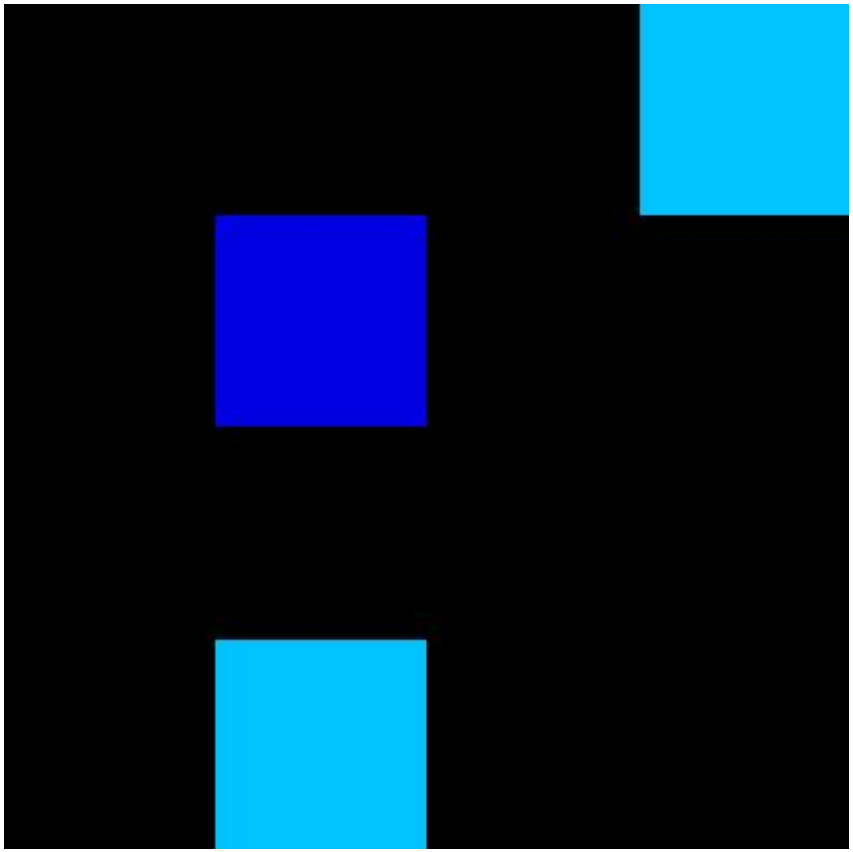
Máximo Local

- Exemplo de Máximo Local.



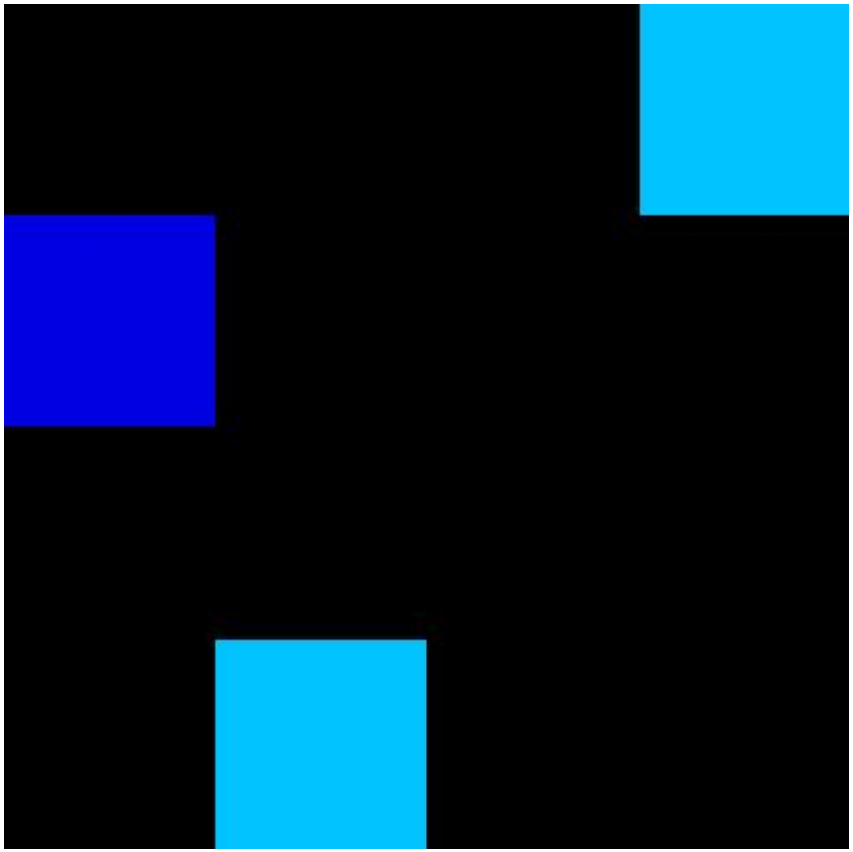
Máximo Local

- Exemplo de Máximo Local.



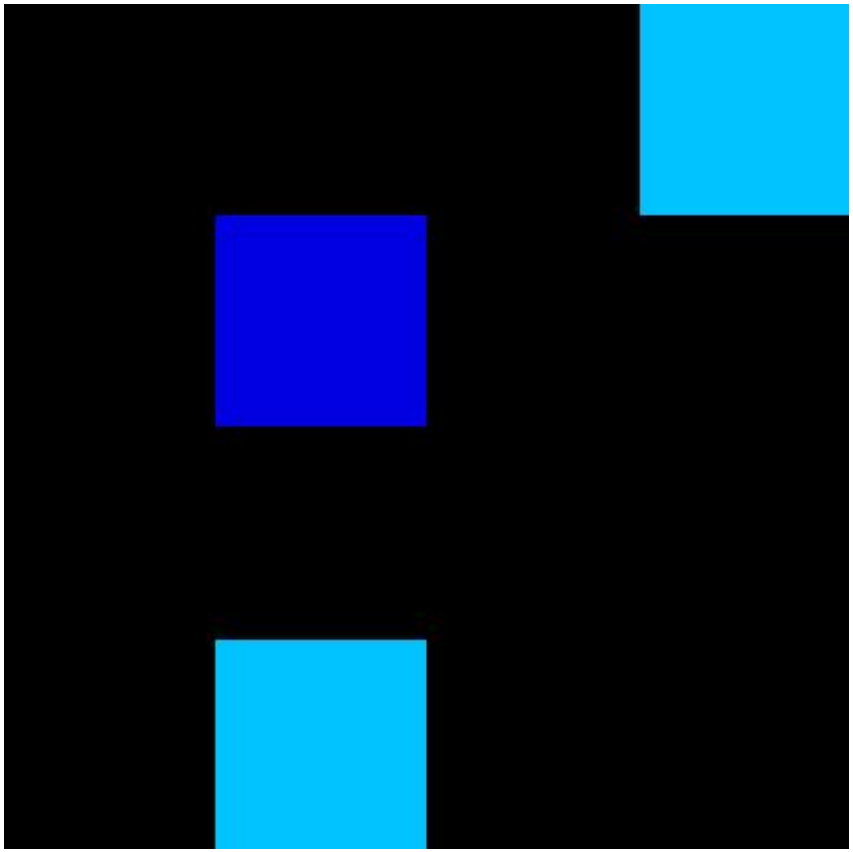
Máximo Local

- Exemplo de Máximo Local.



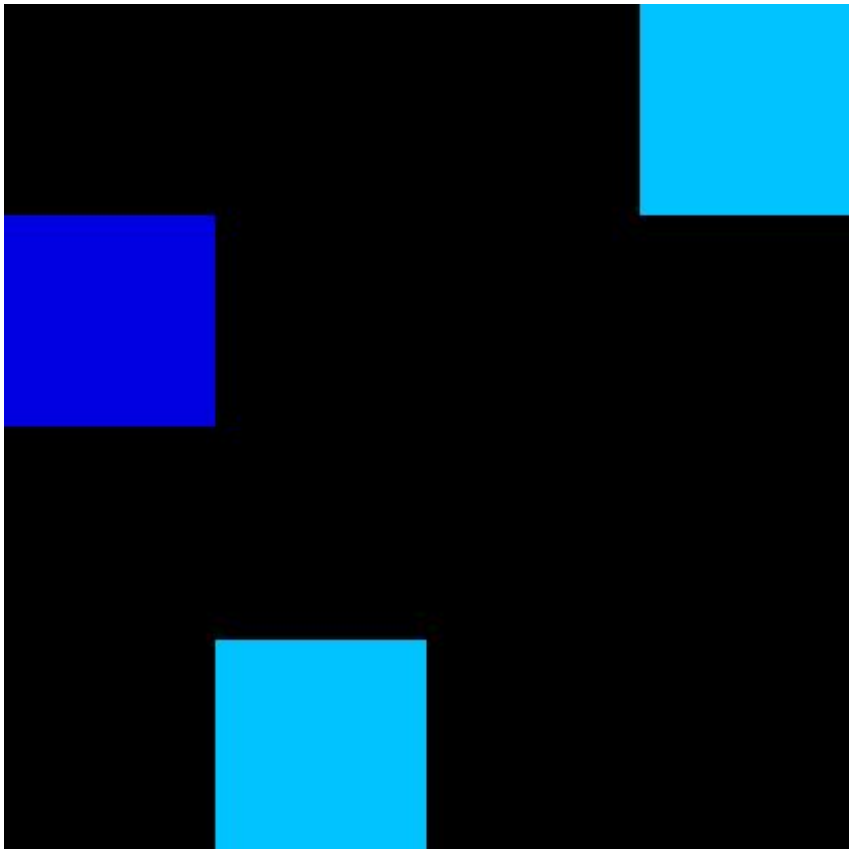
Máximo Local

- Exemplo de Máximo Local.



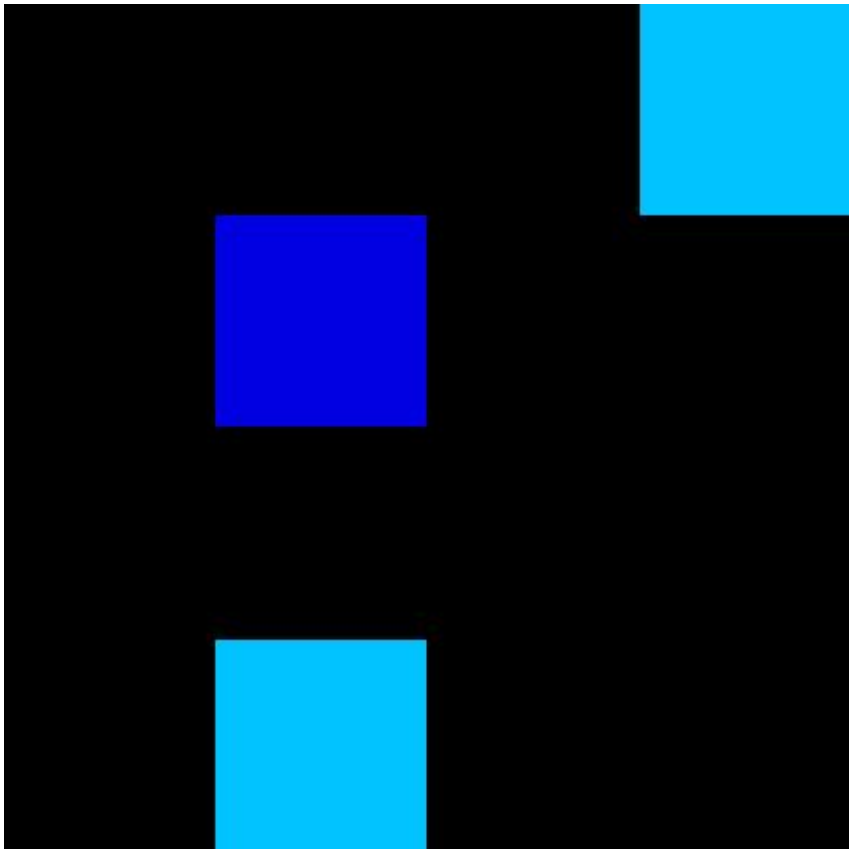
Máximo Local

- Exemplo de Máximo Local.



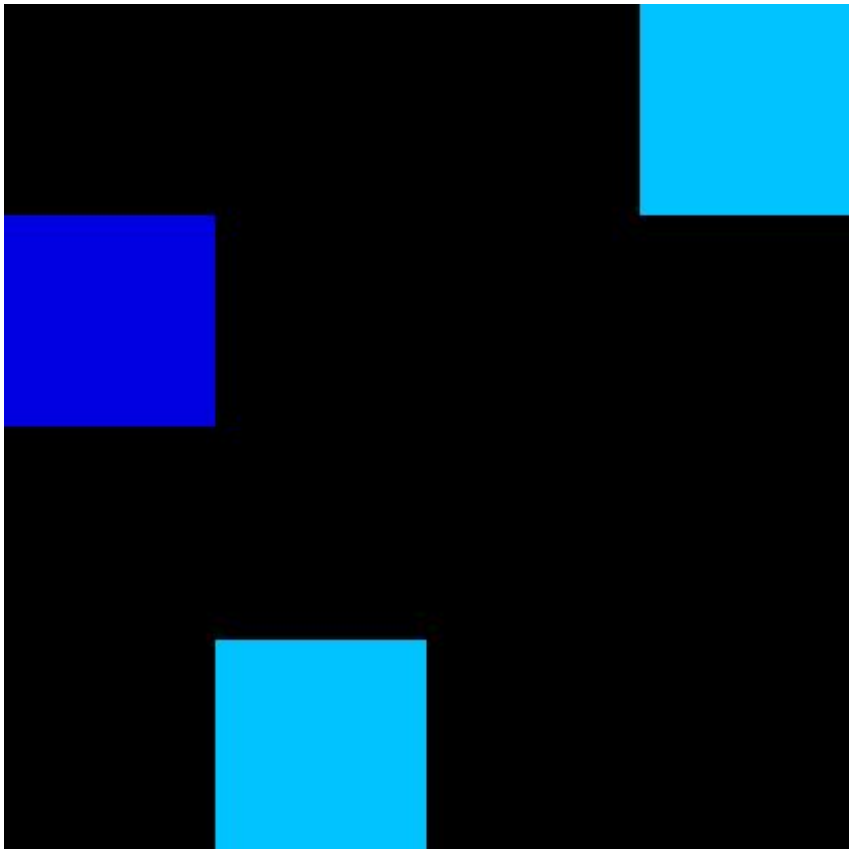
Máximo Local

- Exemplo de Máximo Local.



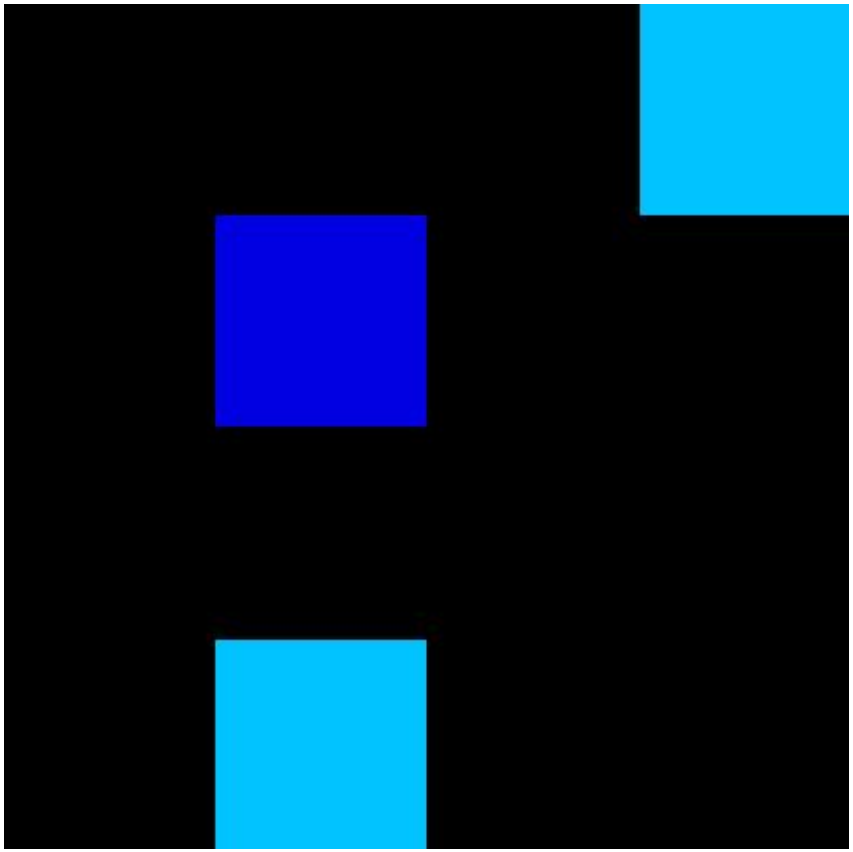
Máximo Local

- Exemplo de Máximo Local.



Máximo Local

- Exemplo de Máximo Local.



Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Parâmetros de controle

- A execução do algoritmo seguiu-se nas seguintes configurações:
  - Matriz de entrada: $[8 \times 8, 6 \times 6, 4 \times 4]$.
  - Discount Factor: 0.99.
  - Número de Iterações de treino: 50000
  - Número de Trajetórias: 20.
  - Número de Itens: 5
  - * Número máximo de movimentos: (ÁREA_MATRIZ) * (N_ITEMS * 4).
  - * Número Máximo de Repetições de Movimentos: (NÚMERO_MÁXIMO_MOVIMENTOS * (0.4)).
  - * Número de épocas de ajuste da rede: (N_TRAJETORIAS) / 2.
  - Epsilon mínimo: 0.05.
  - Número máximo de repetições: (N_TRAJETORIAS * N_GAMES) / 300 = 3333.

Epsilon Restore e Coletagem de Trajetórias

- Policy gradients tem uma tendência maior a encontrar **máximos locais**.
- Para tentar sair dos máximos locais, procurou-se **restaurar o valor do Epsilon** toda vez que o agente encontrar um máximo local.
- O máximo local pode ser encontrado se o agente **ultrapassar** um certo número de estados de término do jogo.
  - São os estados de término: [max_moves, repeated_actions, hit_boundry, hit_wall,all_fix_collected].
  - Se em uma dessas posições ela ultrassar o número máximo de repetições permitidas (3333), então o epsilon irá ser restaurado e decrementado a cada jogada.
- Para aumentar a velocidade de treino, implementou-se a coletagem de trajetória de **forma assíncrona**.

Epsilon Restore e Coletagem de Trajetórias

- Policy gradients tem uma tendência maior a encontrar **máximos locais**.
- Para tentar sair dos máximos locais, procurou-se **restaurar o valor do Epsilon** toda vez que o agente encontrar um máximo local.
- O máximo local pode ser encontrado se o agente **ultrapassar** um certo número de estados de término do jogo.
  - São os estados de término: [max_moves, repeated_actions, hit_boundry, hit_wall,all_fix_collected].
  - Se em uma dessas posições ela ultrassar o número máximo de repetições permitidas (3333), então o epsilon irá ser restaurado e decrementado a cada jogada.
- Para aumentar a velocidade de treino, implementou-se a coletagem de trajetória de **forma assíncrona**.

Epsilon Restore e Coletagem de Trajetórias

- Policy gradients tem uma tendência maior a encontrar **máximos locais**.
- Para tentar sair dos máximos locais, procurou-se **restaurar o valor do Epsilon** toda vez que o agente encontrar um máximo local.
- O máximo local pode ser encontrado se o agente **ultrapassar** um certo número de estados de término do jogo.
  - São os estados de término: [max_moves, repeated_actions, hit_boundry, hit_wall,all_fix_collected].
  - Se em uma dessas posições ela ultrassar o número máximo de repetições permitidas (3333), então o epsilon irá ser restaurado e decrementado a cada jogada.
- Para aumentar a velocidade de treino, implementou-se a coletagem de trajetória de **forma assíncrona**.

Epsilon Restore e Coletagem de Trajetórias

- Policy gradients tem uma tendência maior a encontrar **máximos locais**.
- Para tentar sair dos máximos locais, procurou-se **restaurar o valor do Epsilon** toda vez que o agente encontrar um máximo local.
- O máximo local pode ser encontrado se o agente **ultrapassar** um certo número de estados de término do jogo.
  - São os estados de término: [max_moves, repeated_actions, hit_boundry, hit_wall,all_fix_collected].
  - Se em uma dessas posições ela ultrassar o número máximo de repetições permitidas (3333), então o epsilon irá ser restaurado e decrementado a cada jogada.
- Para aumentar a velocidade de treino, implementou-se a coletagem de trajetória de **forma assíncrona**.

Epsilon Restore e Coletagem de Trajetórias

- Policy gradients tem uma tendência maior a encontrar **máximos locais**.
- Para tentar sair dos máximos locais, procurou-se **restaurar o valor do Epsilon** toda vez que o agente encontrar um máximo local.
- O máximo local pode ser encontrado se o agente **ultrapassar** um certo número de estados de término do jogo.
  - São os estados de término: [max_moves, repeated_actions, hit_boundry, hit_wall,all_fix_collected].
  - Se em uma dessas posições ela ultrassar o número máximo de repetições permitidas (3333), então o epsilon irá ser restaurado e decrementado a cada jogada.
- Para aumentar a velocidade de treino, implementou-se a coletagem de trajetória de **forma assíncrona**.

Epsilon Restore e Coletagem de Trajetórias

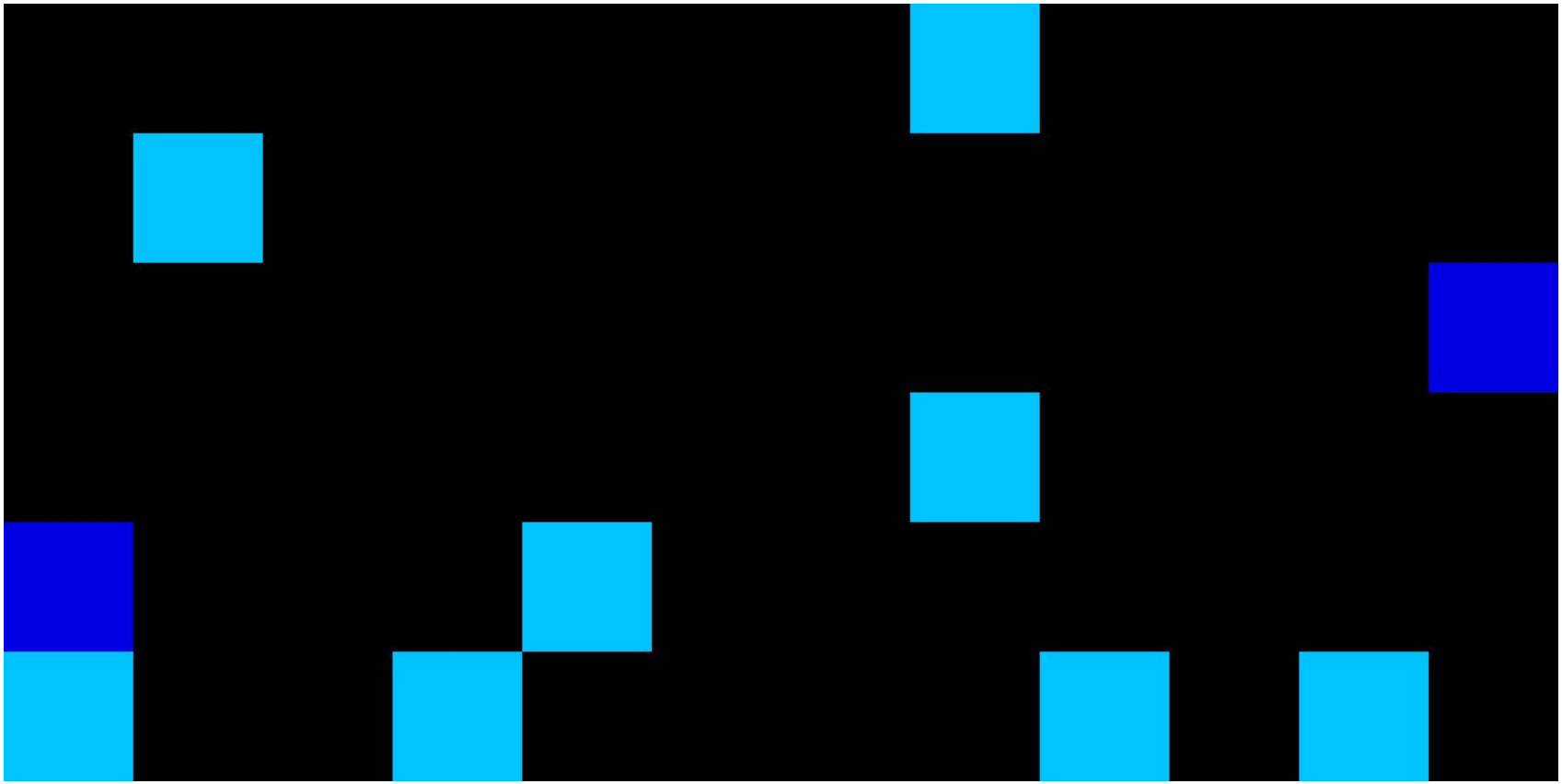
- Policy gradients tem uma tendência maior a encontrar **máximos locais**.
- Para tentar sair dos máximos locais, procurou-se **restaurar o valor do Epsilon** toda vez que o agente encontrar um máximo local.
- O máximo local pode ser encontrado se o agente **ultrapassar** um certo número de estados de término do jogo.
  - São os estados de término: [max_moves, repeated_actions, hit_boundry, hit_wall,all_fix_collected].
  - Se em uma dessas posições ela ultrassar o número máximo de repetições permitidas (3333), então o epsilon irá ser restaurado e decrementado a cada jogada.
- Para aumentar a velocidade de treino, implementou-se a coletagem de trajetória de **forma assíncrona**.

Epsilon Restore e Coletagem de Trajetórias

- Policy gradients tem uma tendência maior a encontrar **máximos locais**.
- Para tentar sair dos máximos locais, procurou-se **restaurar o valor do Epsilon** toda vez que o agente encontrar um máximo local.
- O máximo local pode ser encontrado se o agente **ultrapassar** um certo número de estados de término do jogo.
  - São os estados de término: [max_moves, repeated_actions, hit_boundry, hit_wall,all_fix_collected].
  - Se em uma dessas posições ela ultrassar o número máximo de repetições permitidas (3333), então o epsilon irá ser restaurado e decrementado a cada jogada.
- Para aumentar a velocidade de treino, implementou-se a coletagem de trajetória de **forma assíncrona**.

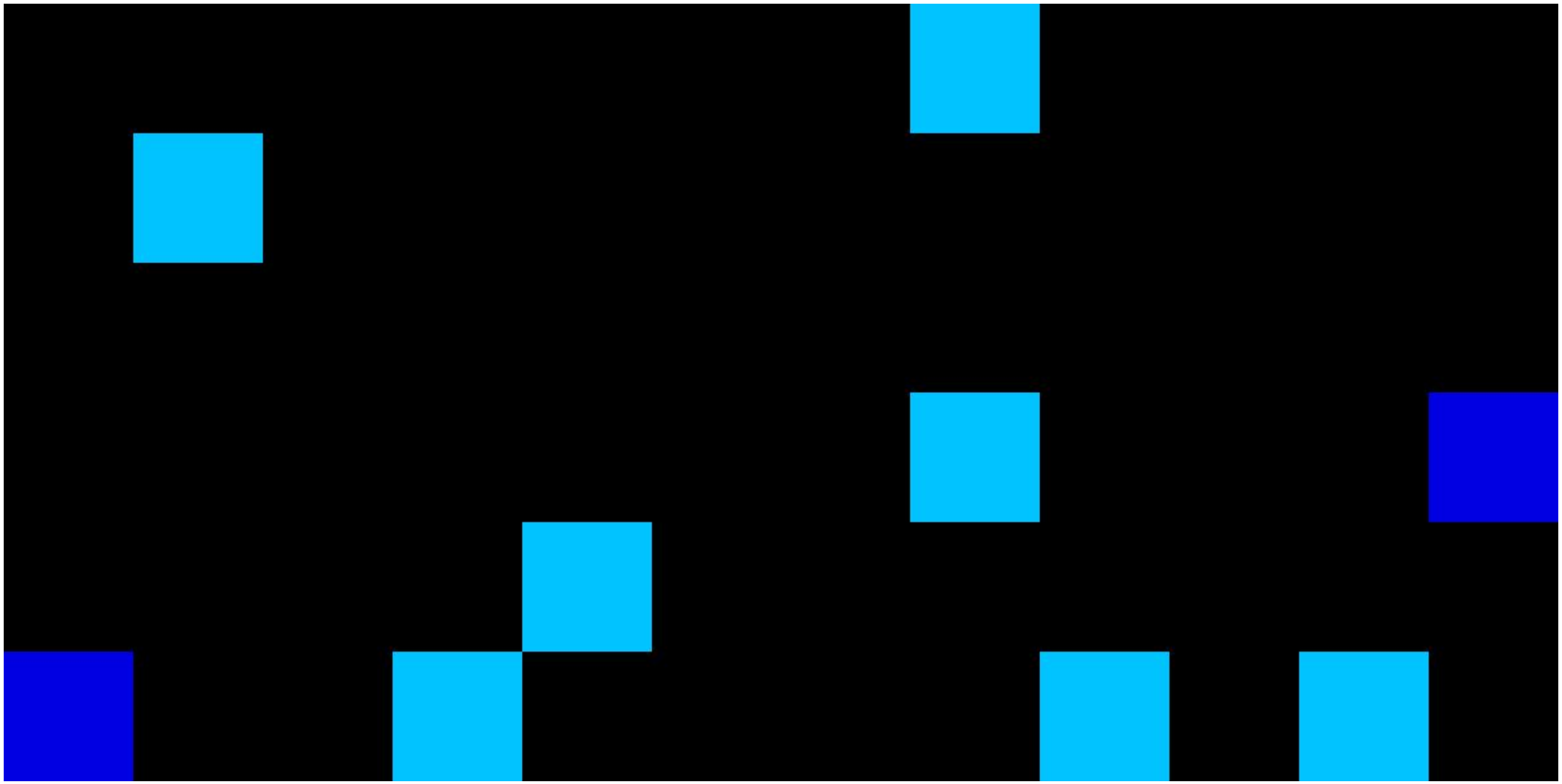
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



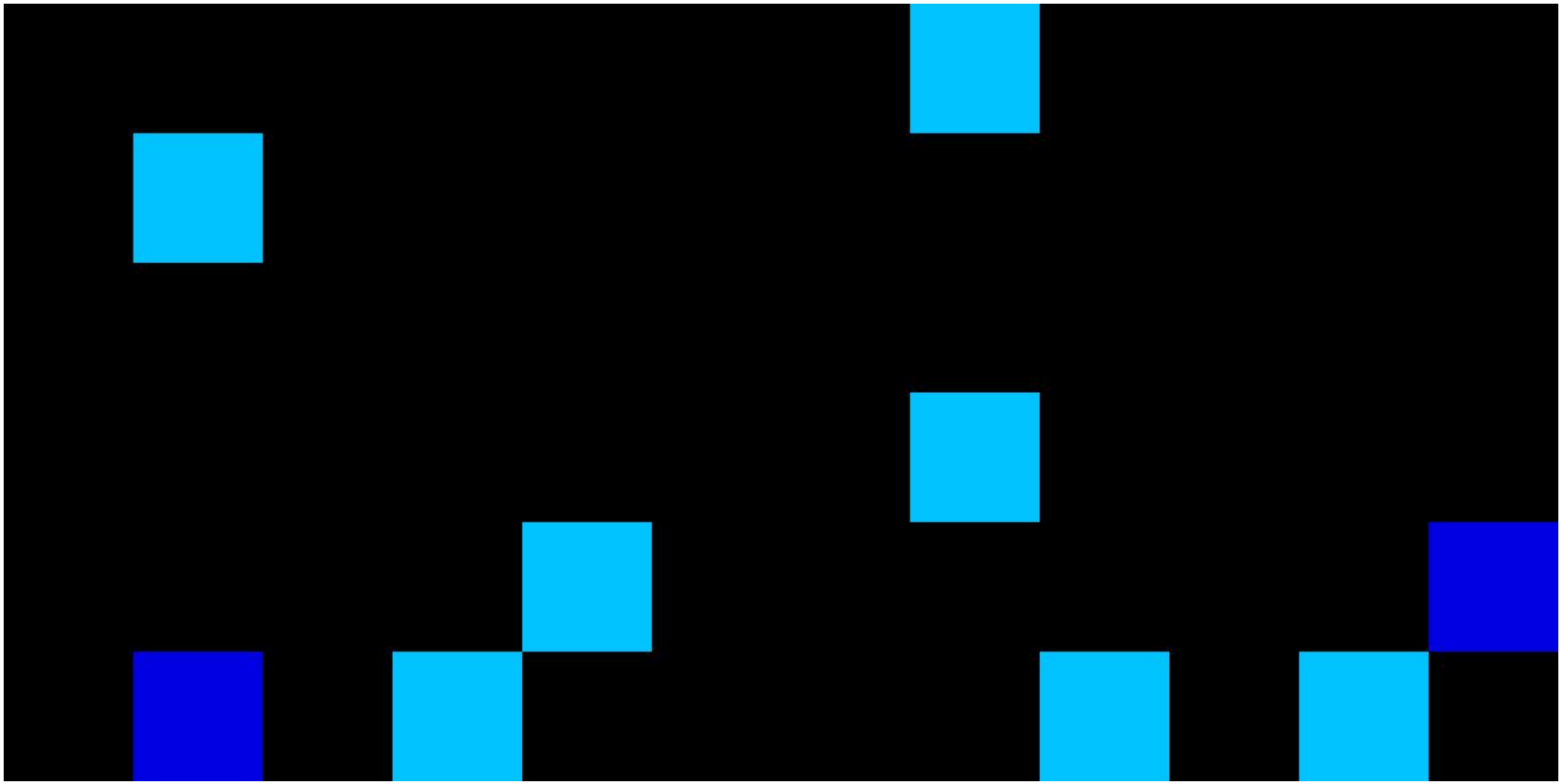
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



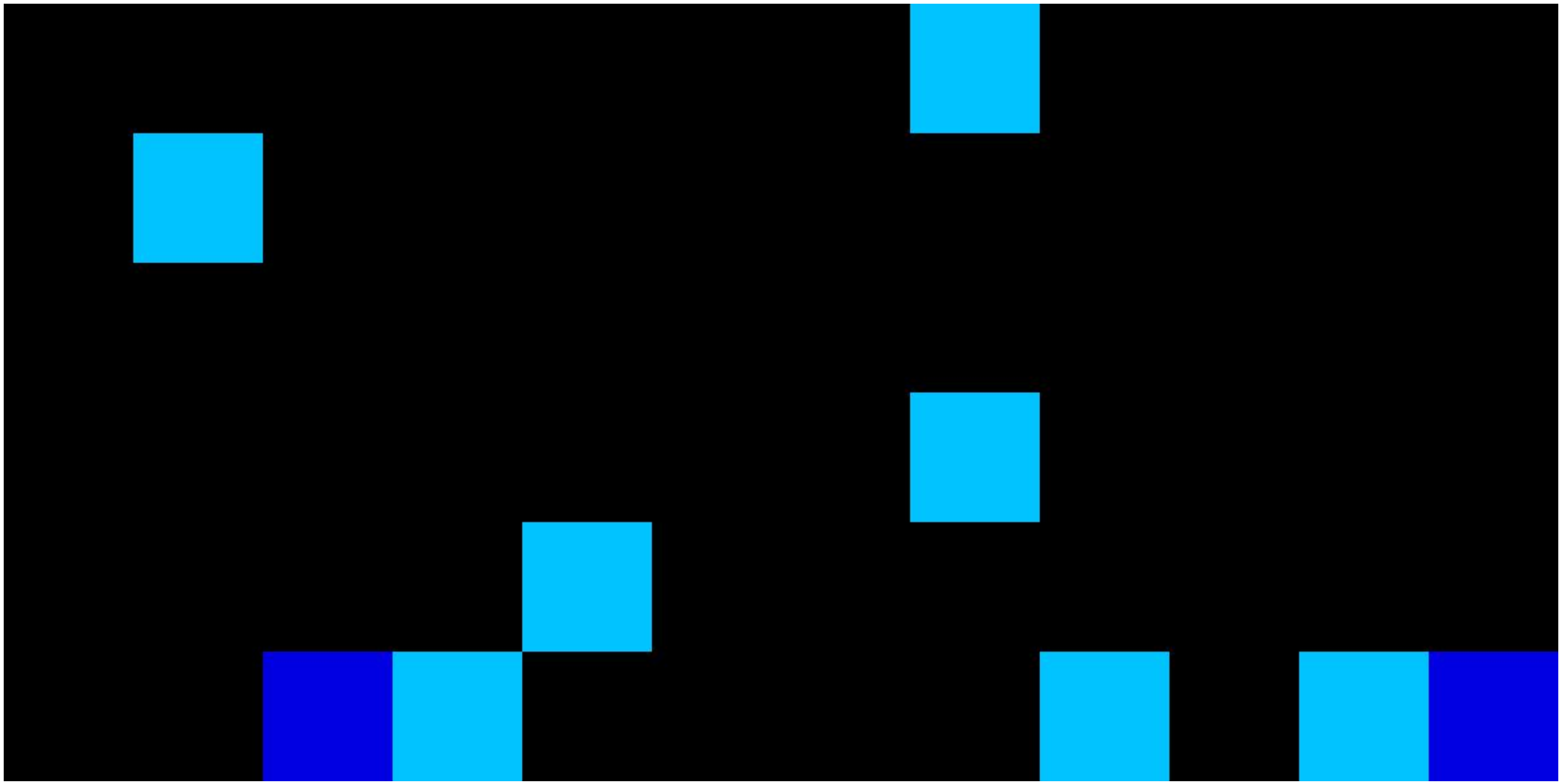
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



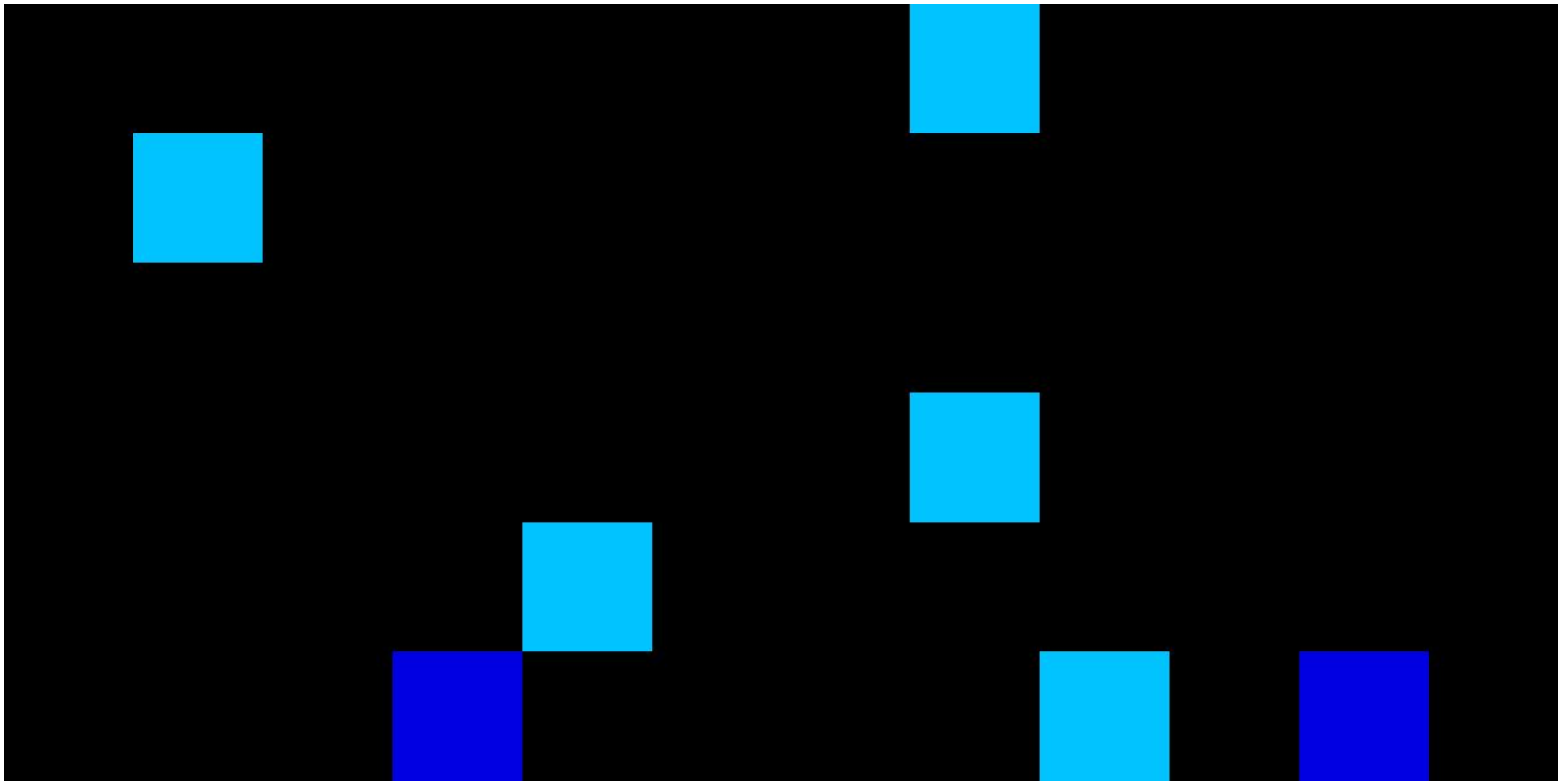
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



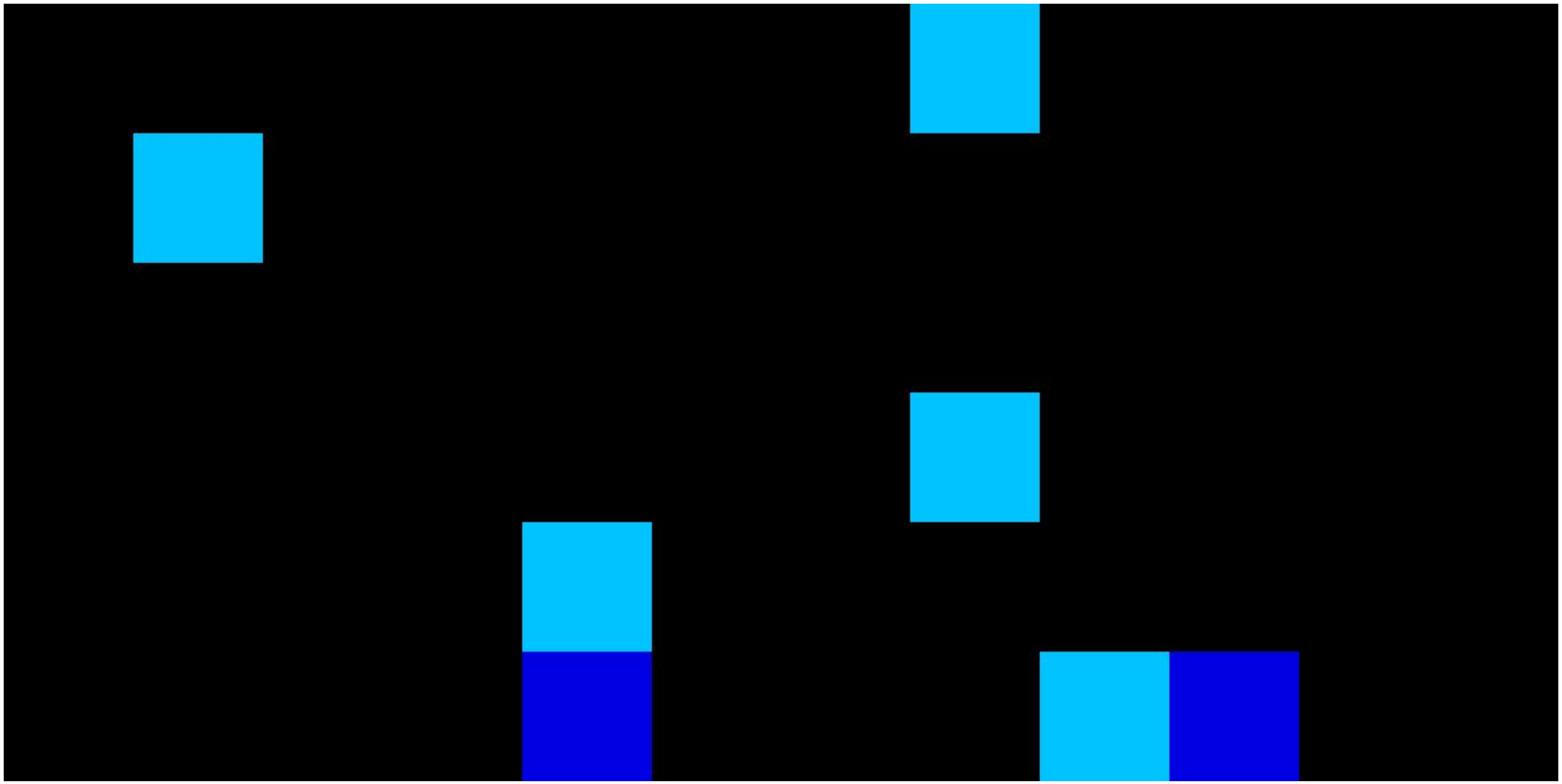
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



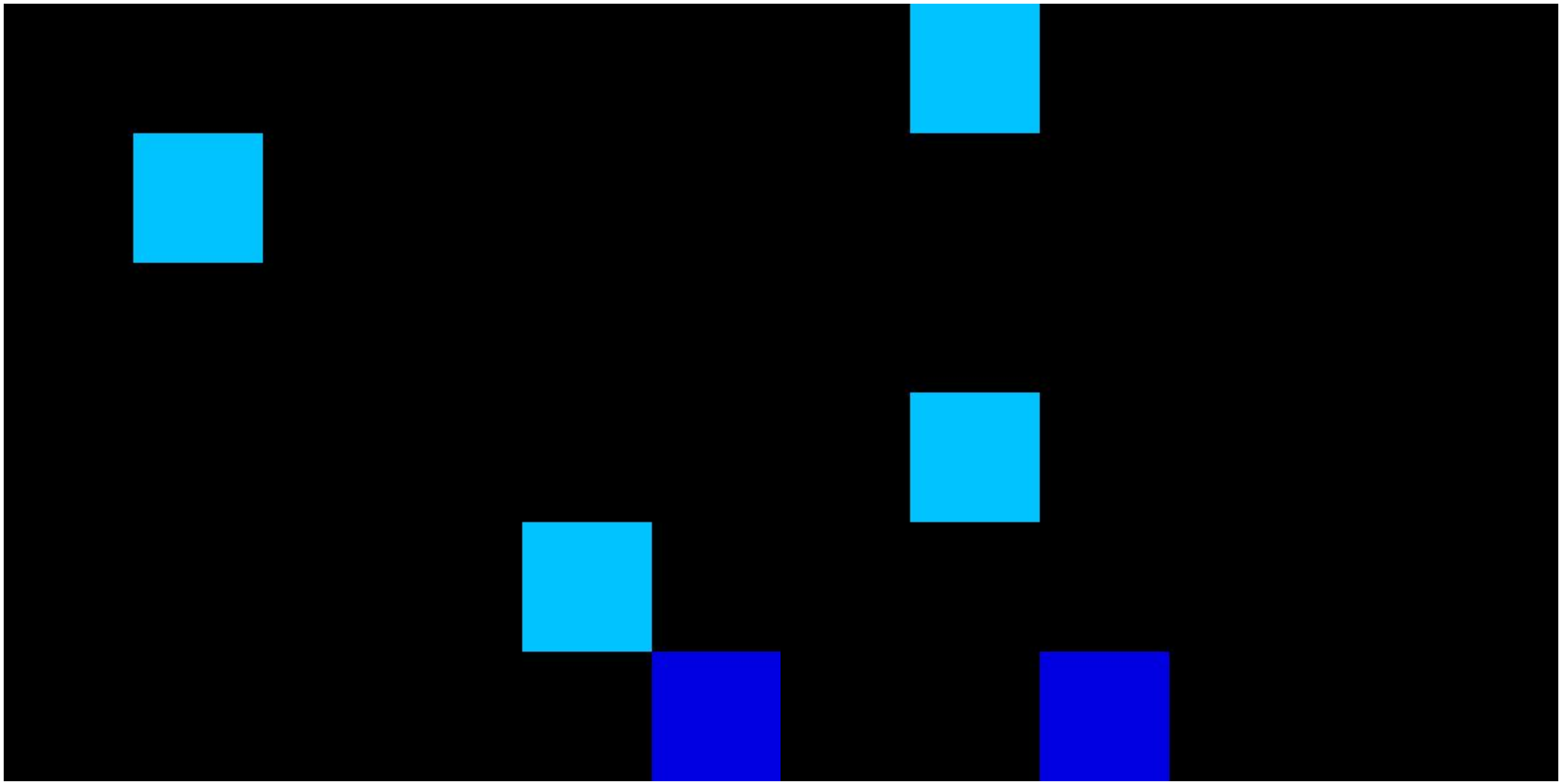
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



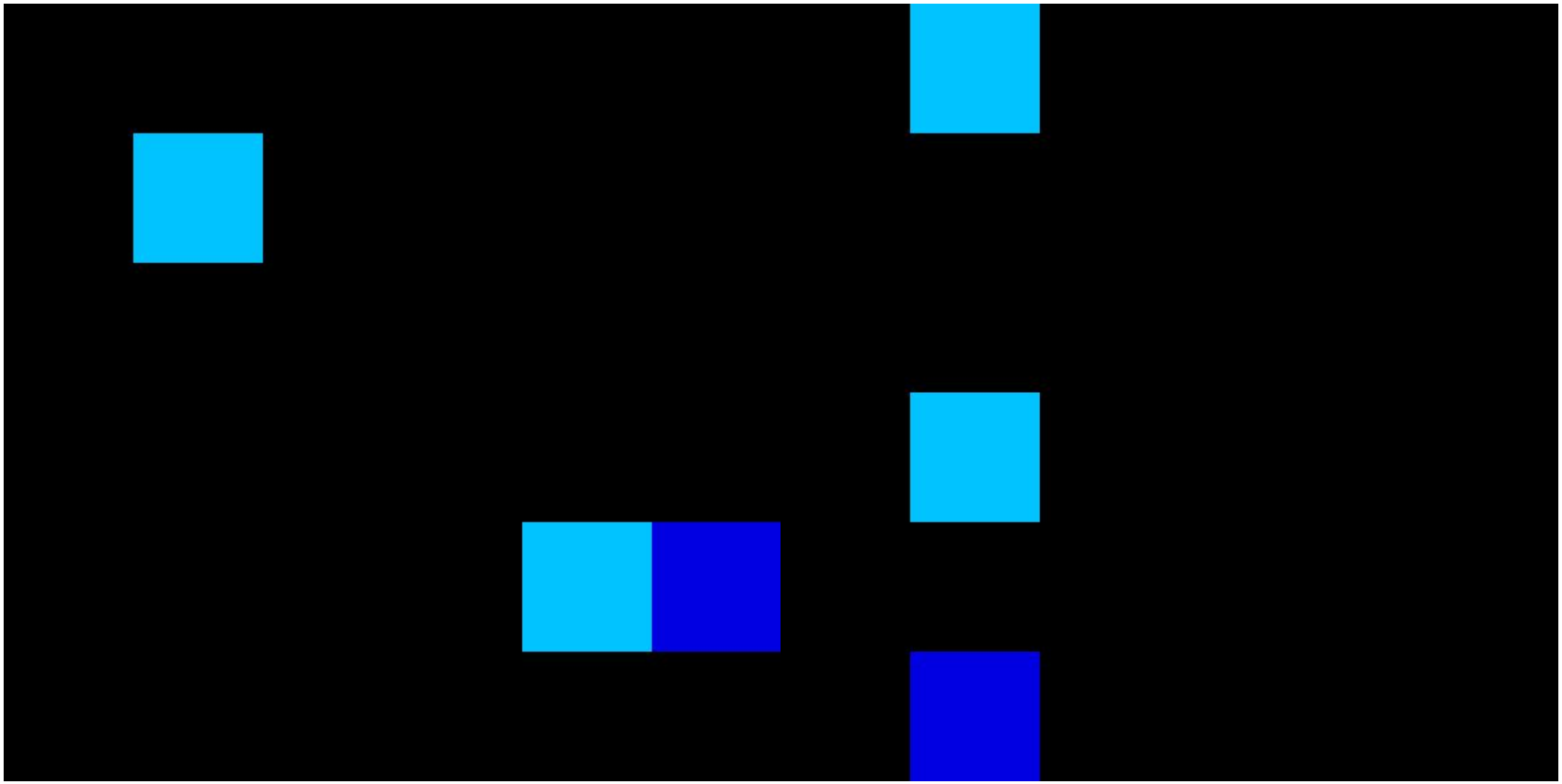
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



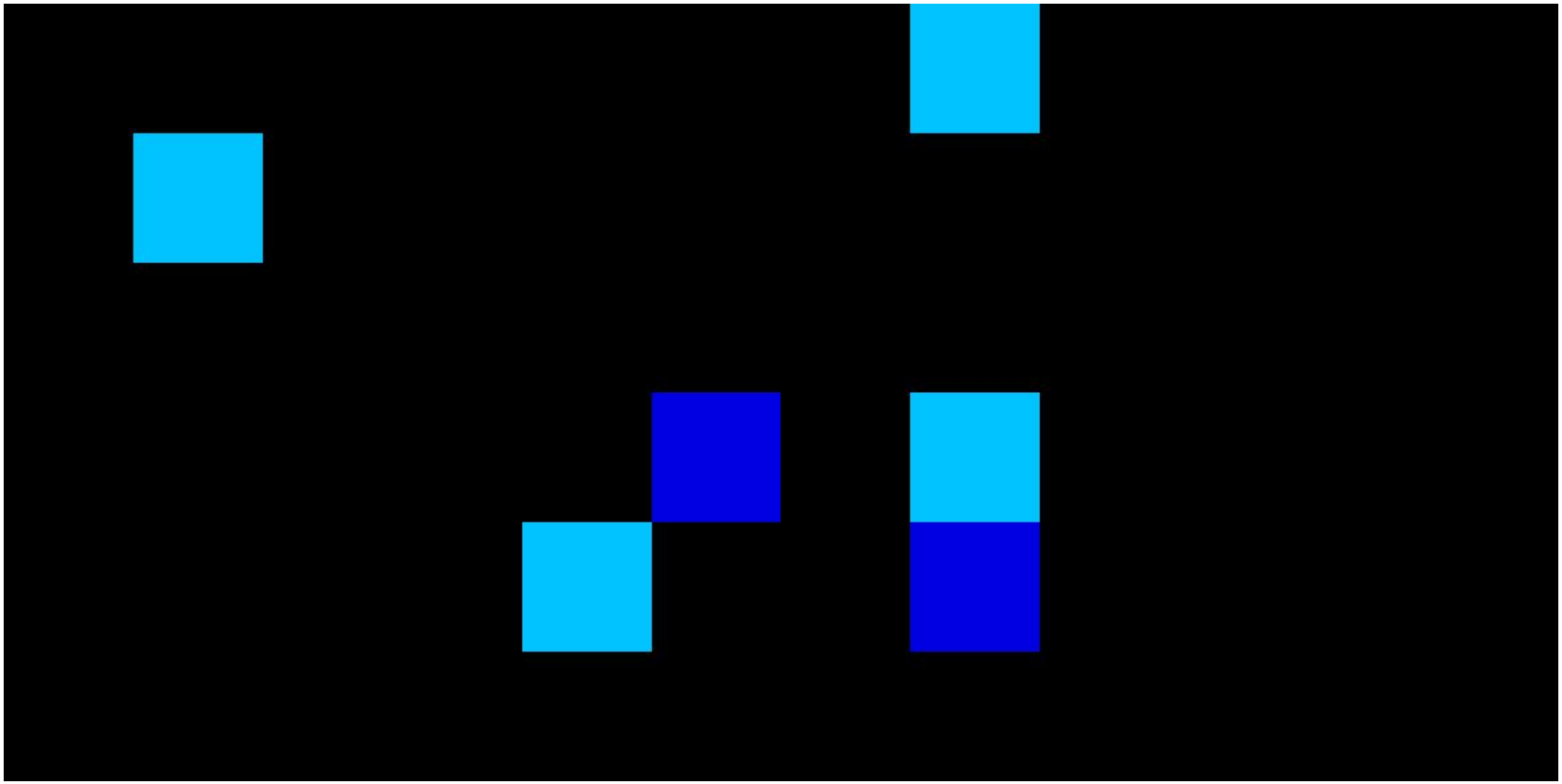
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



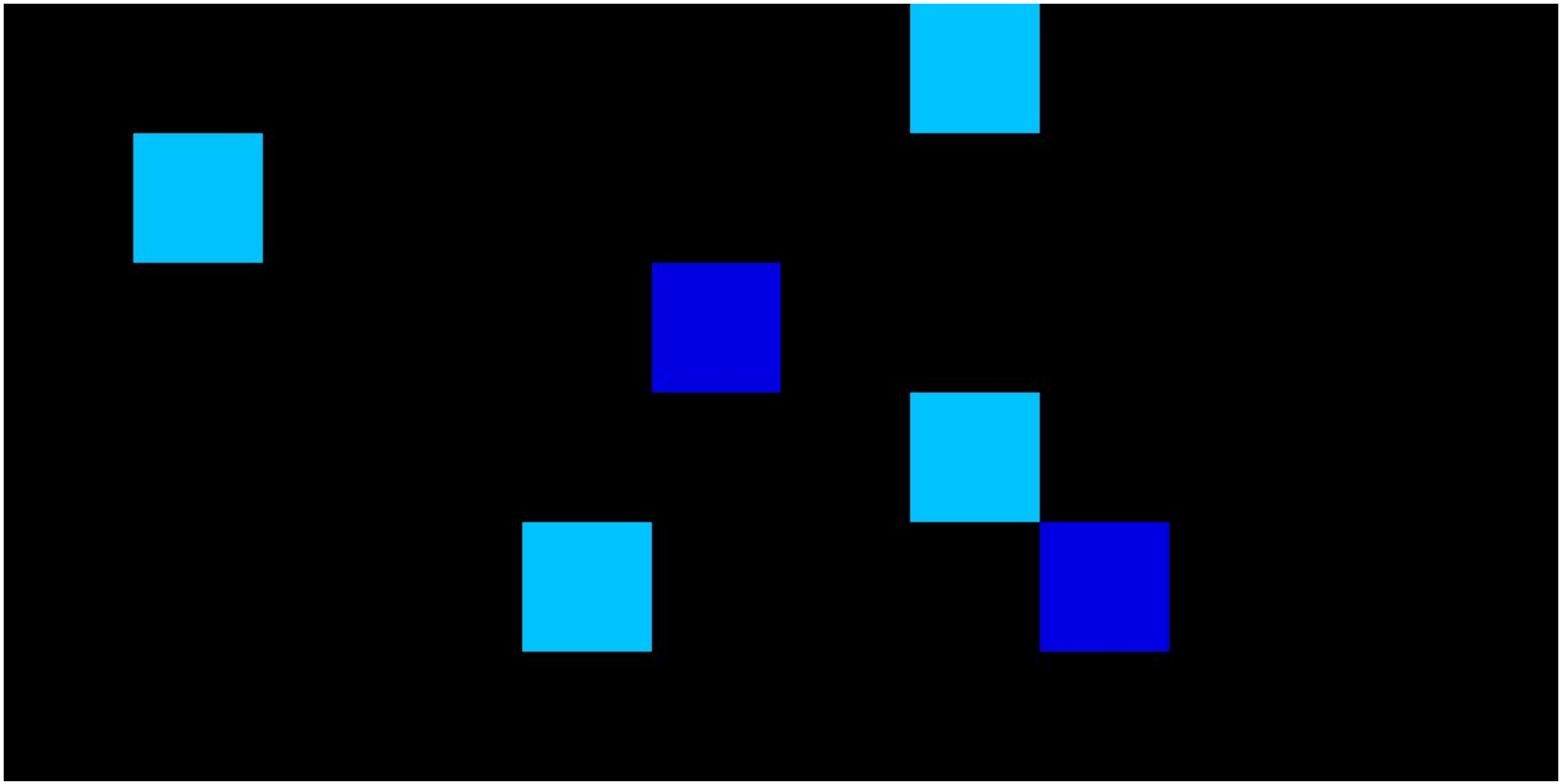
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



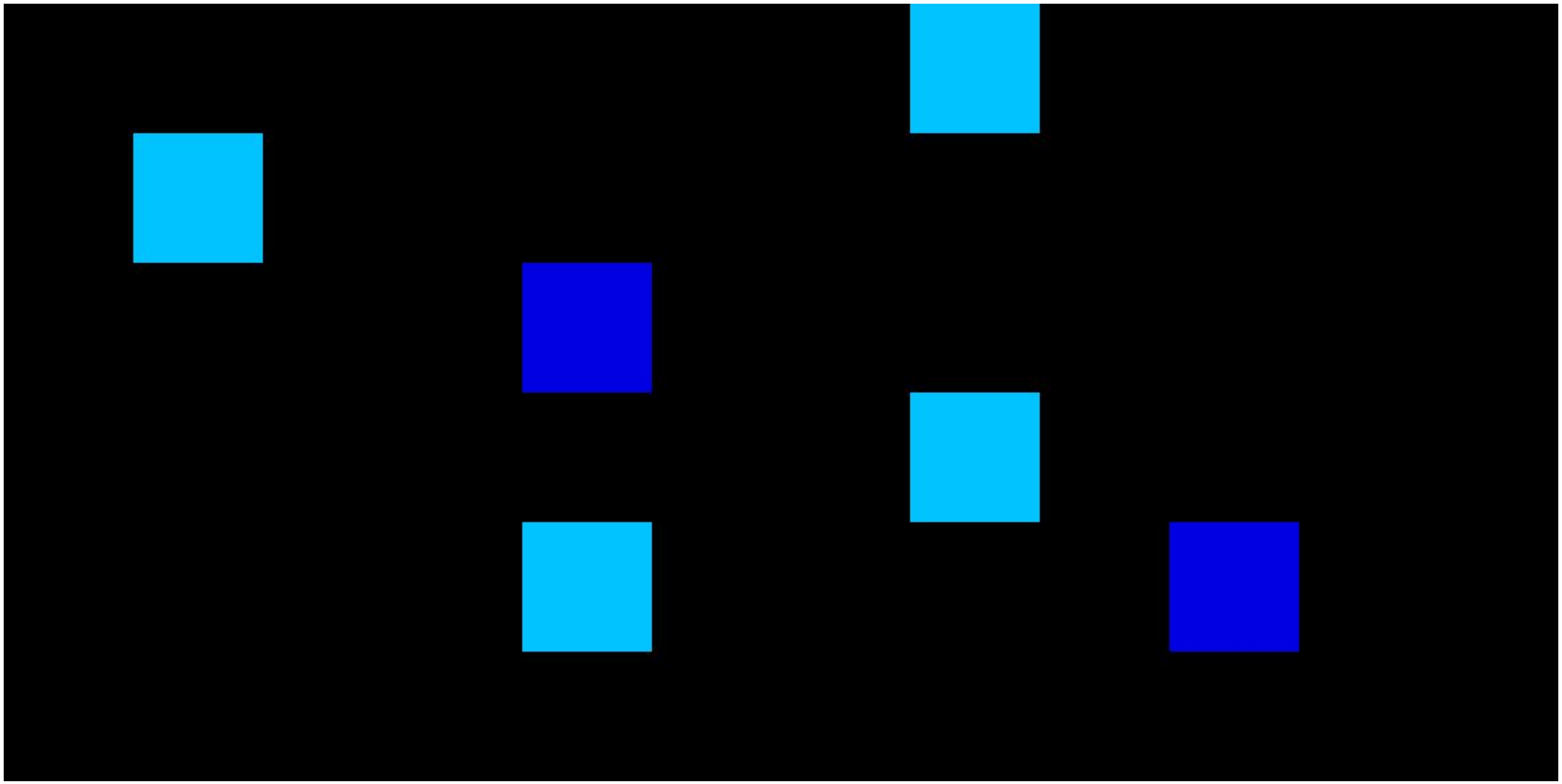
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



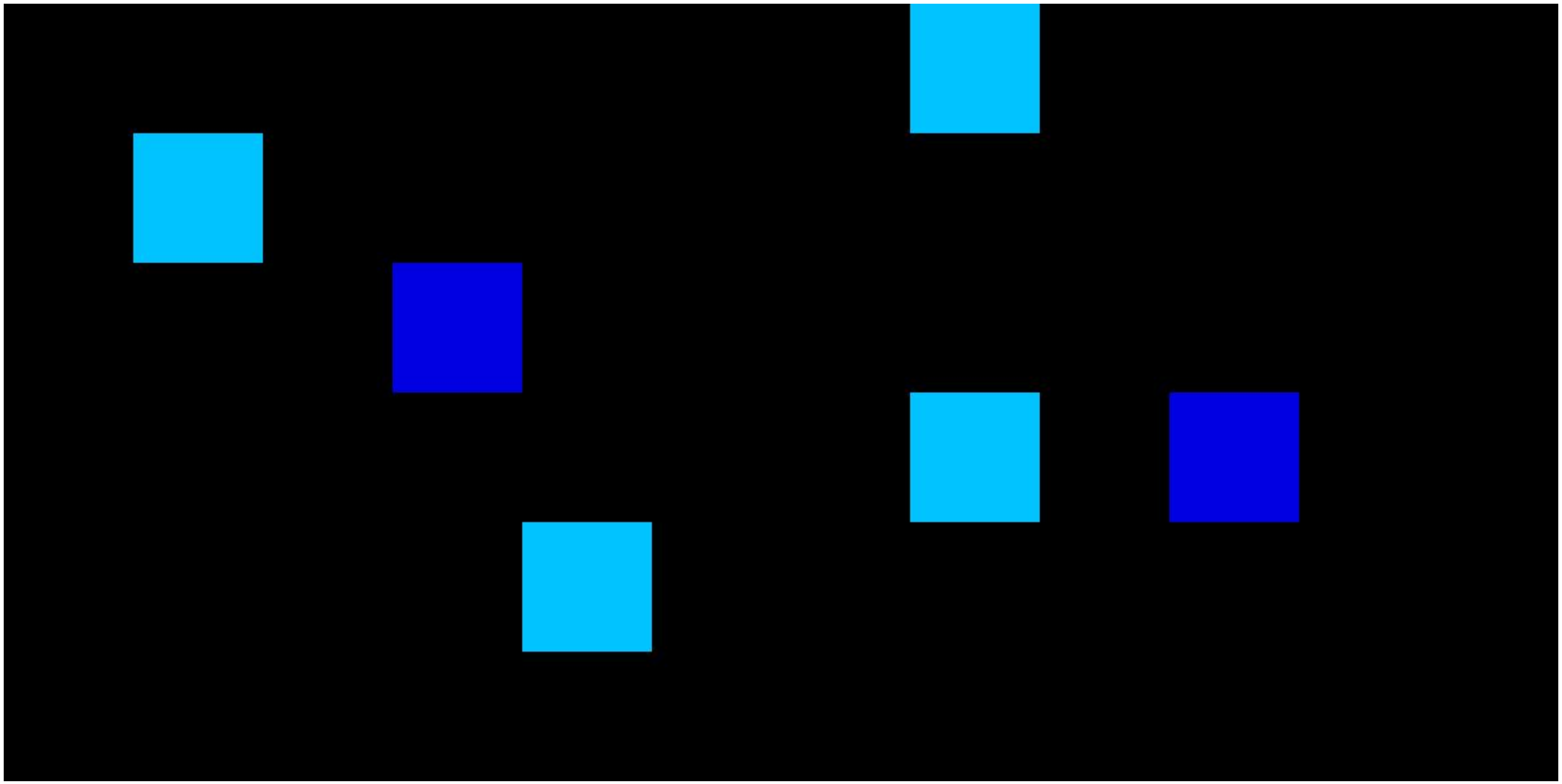
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



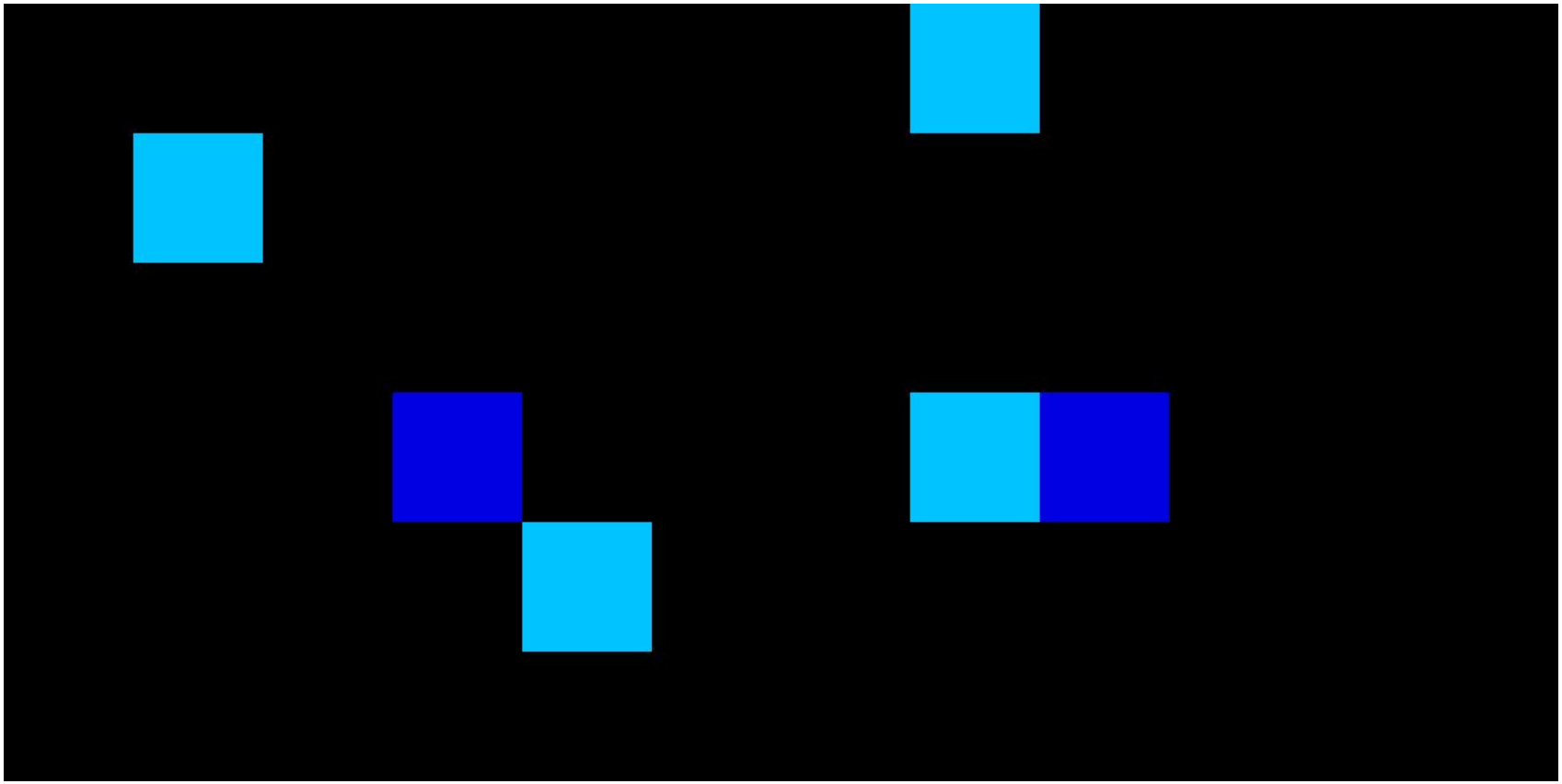
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



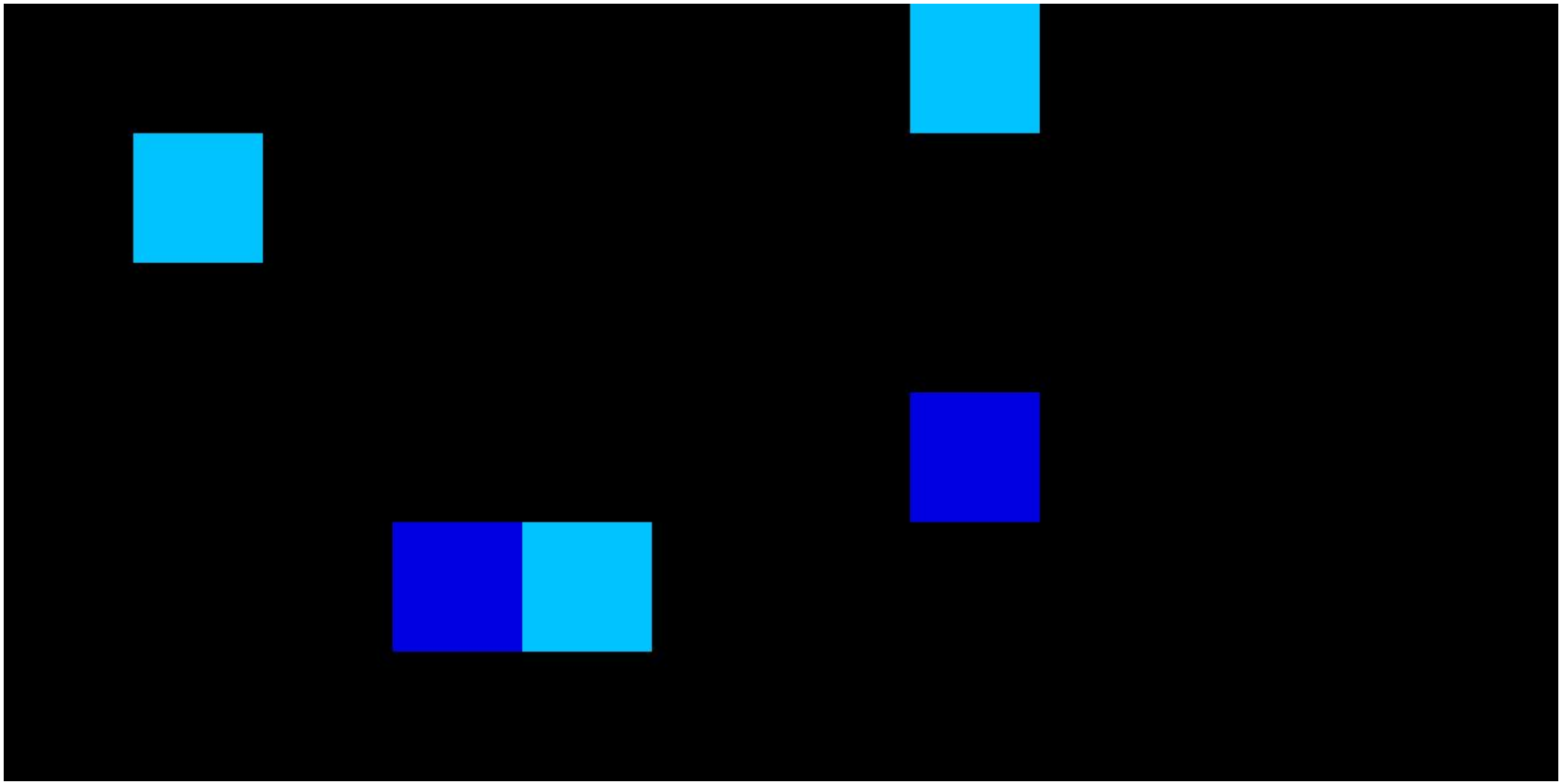
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



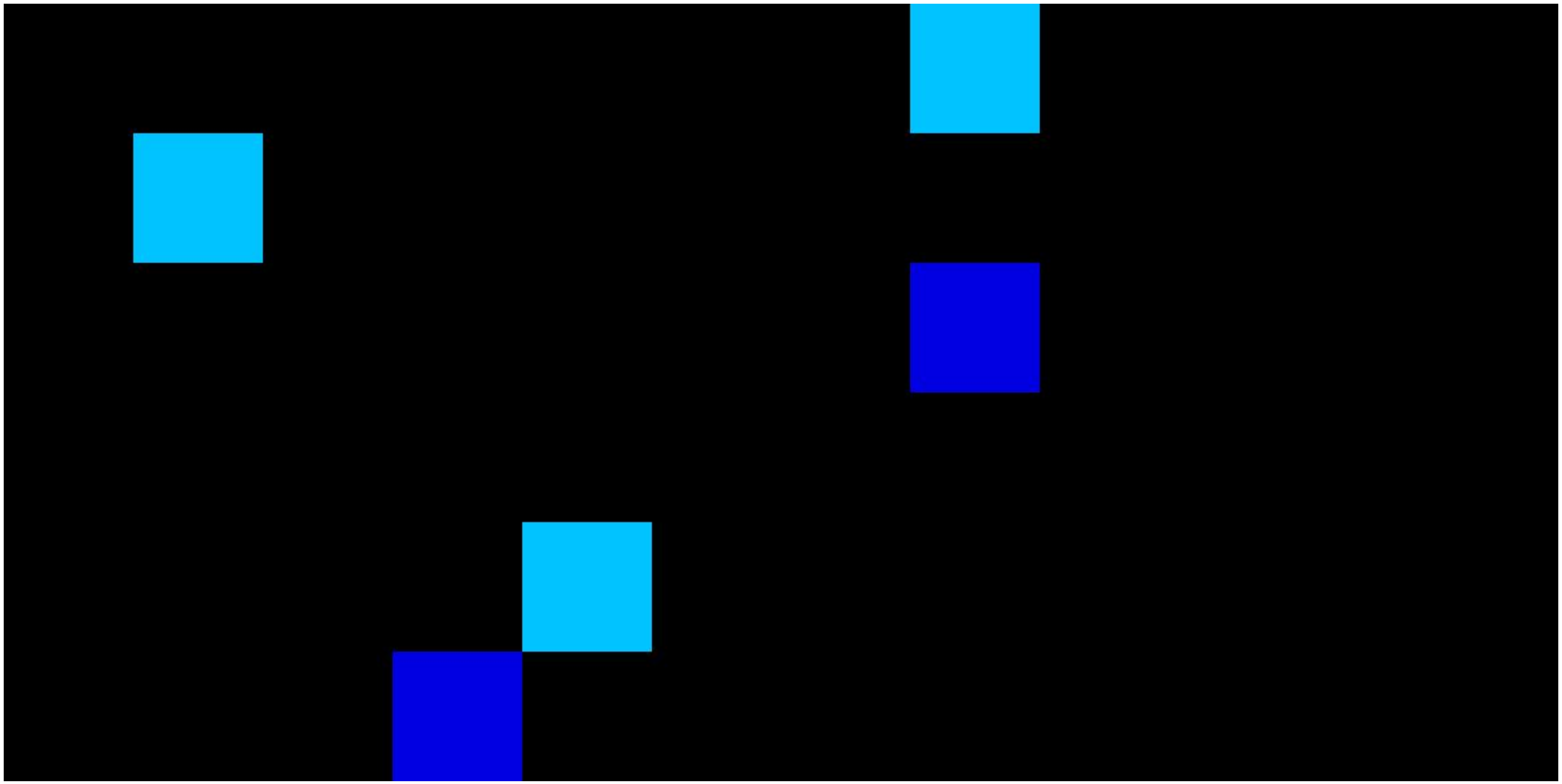
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



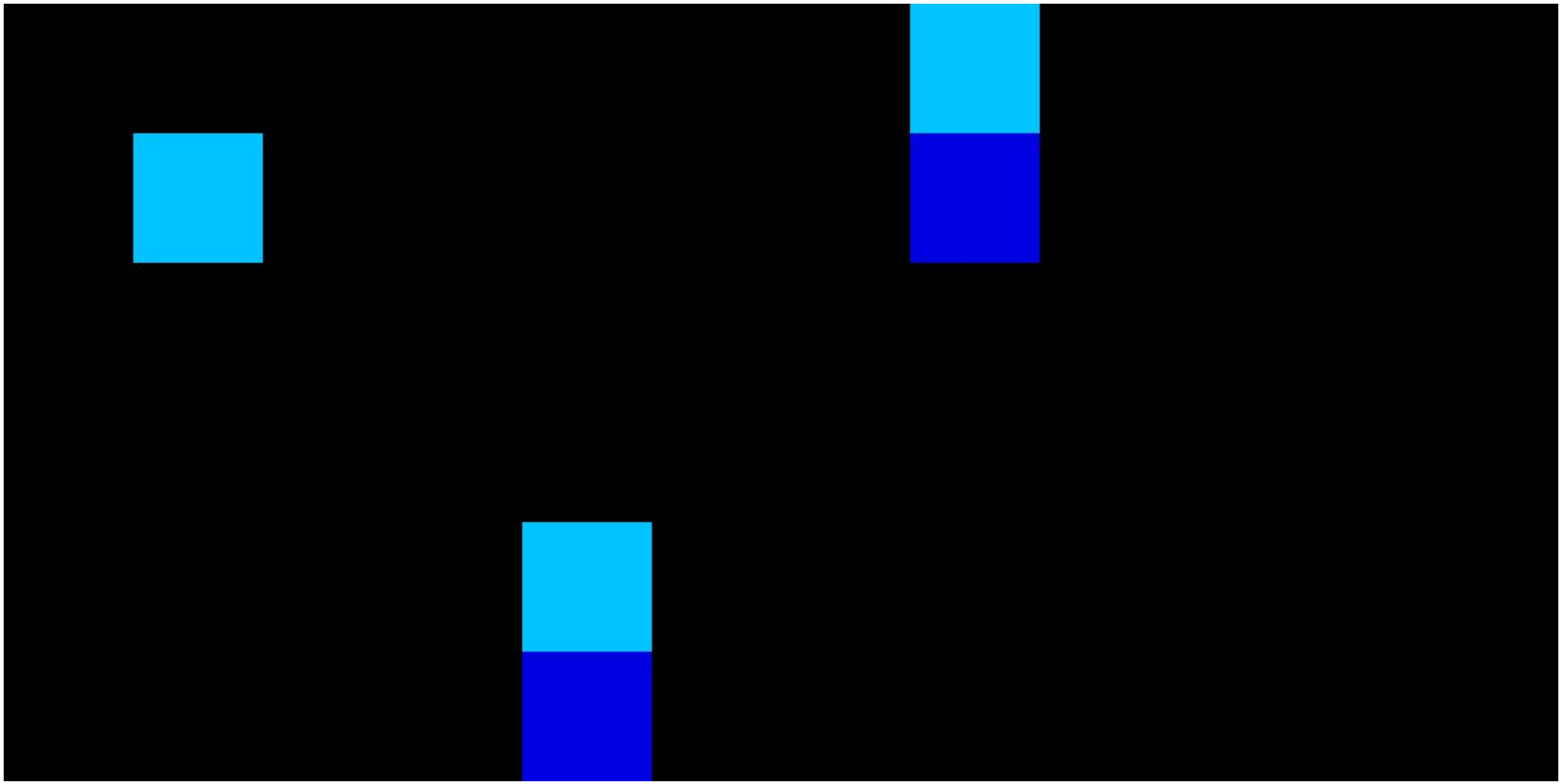
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



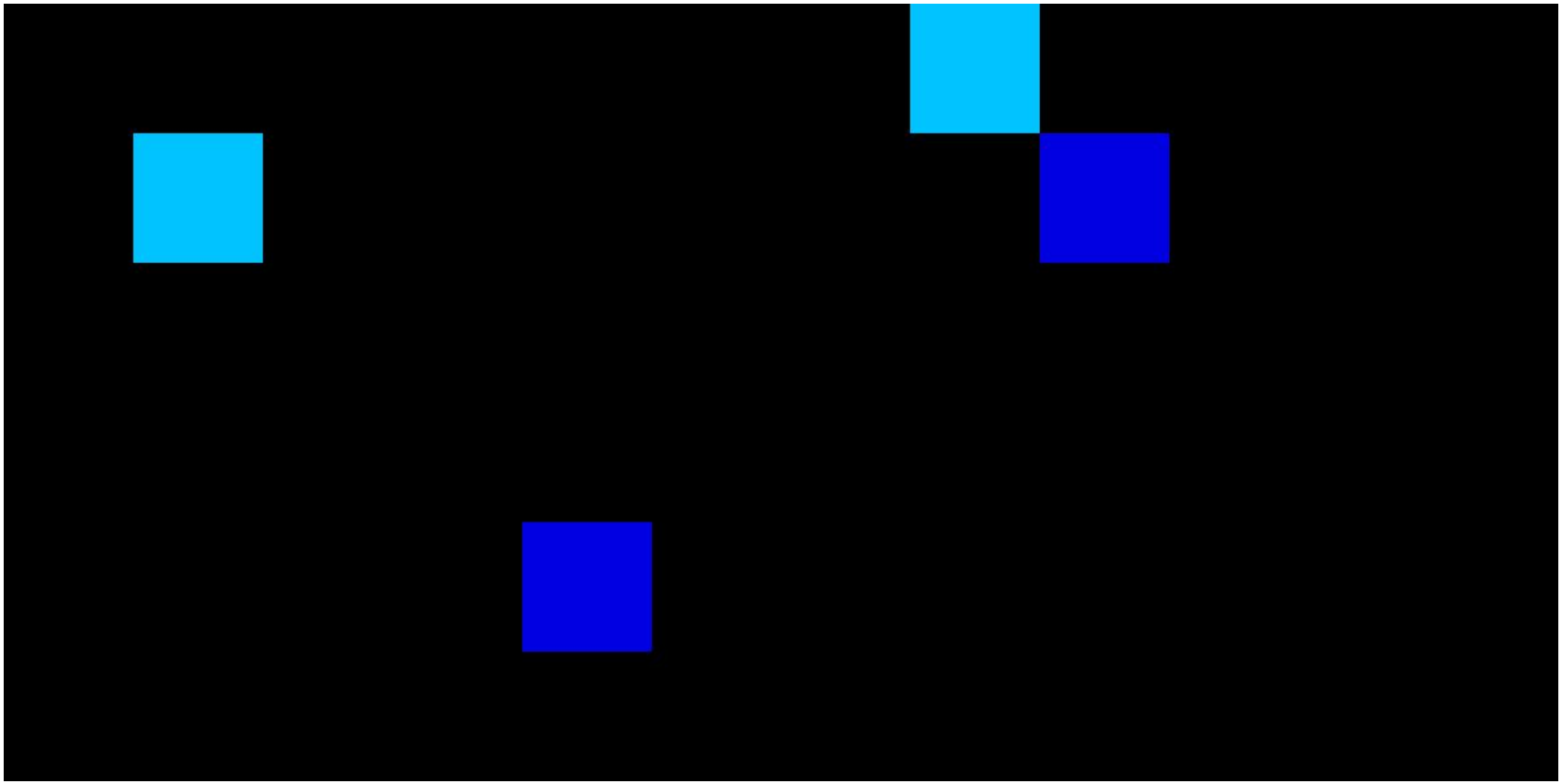
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



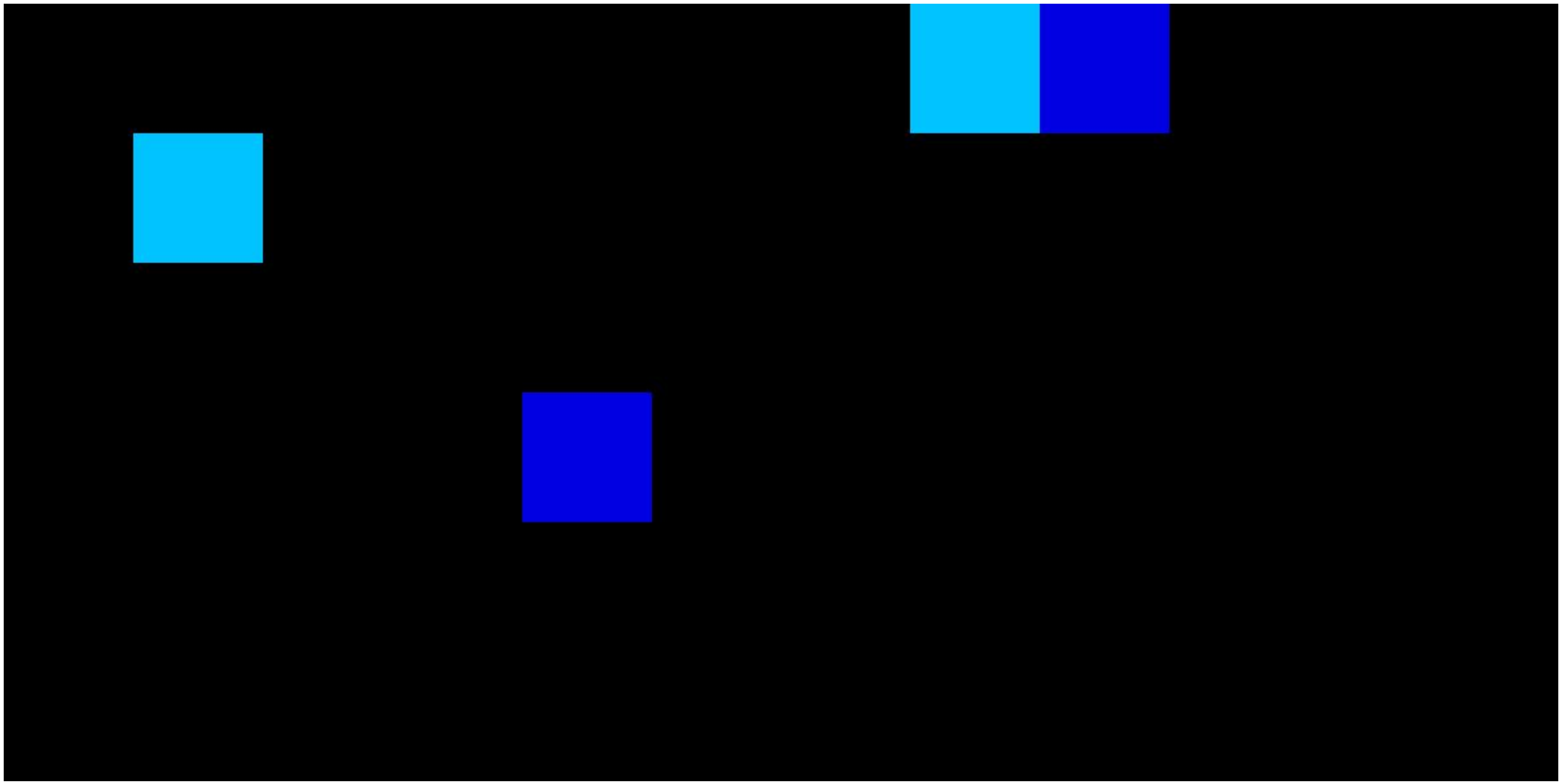
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



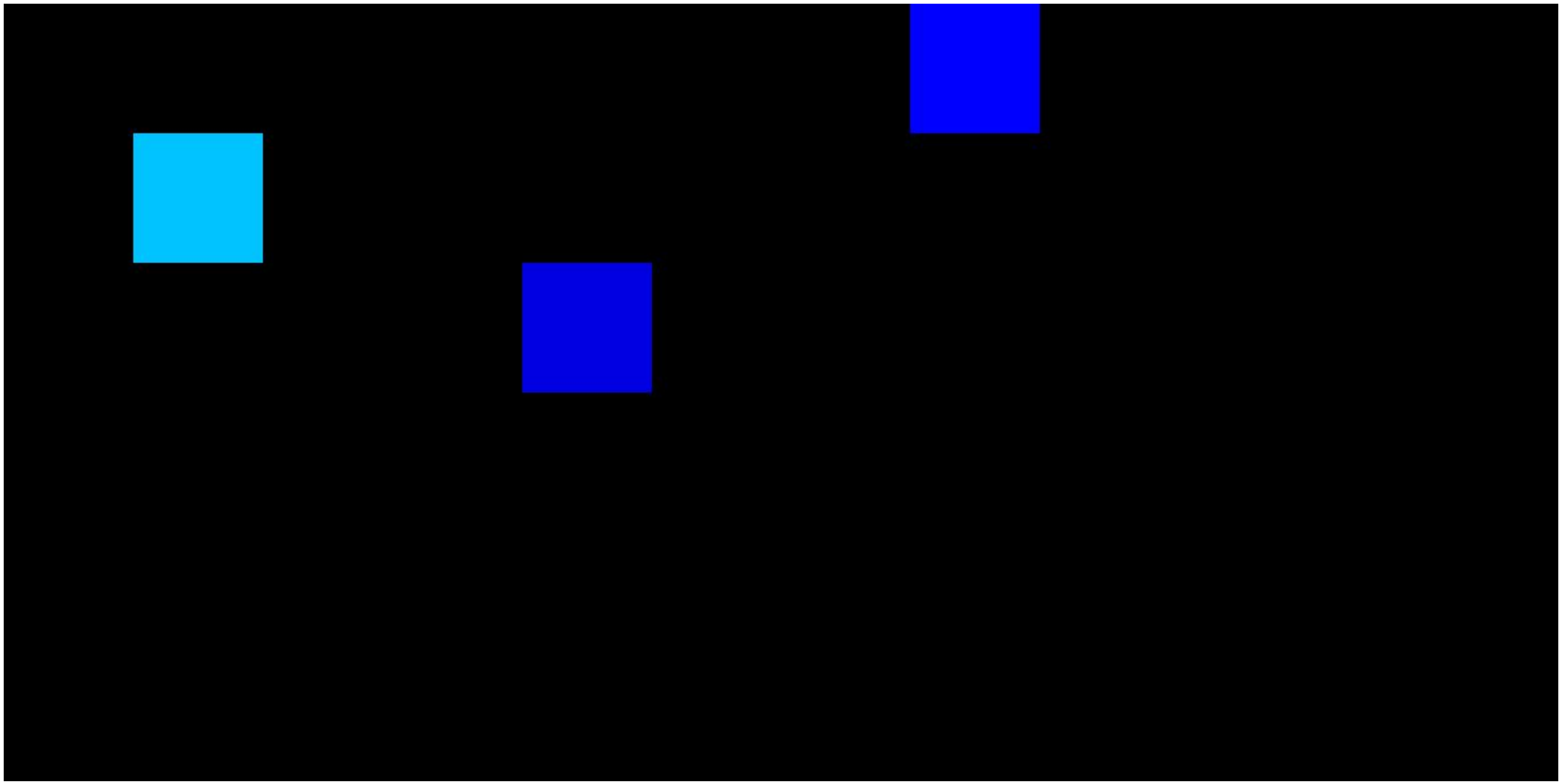
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



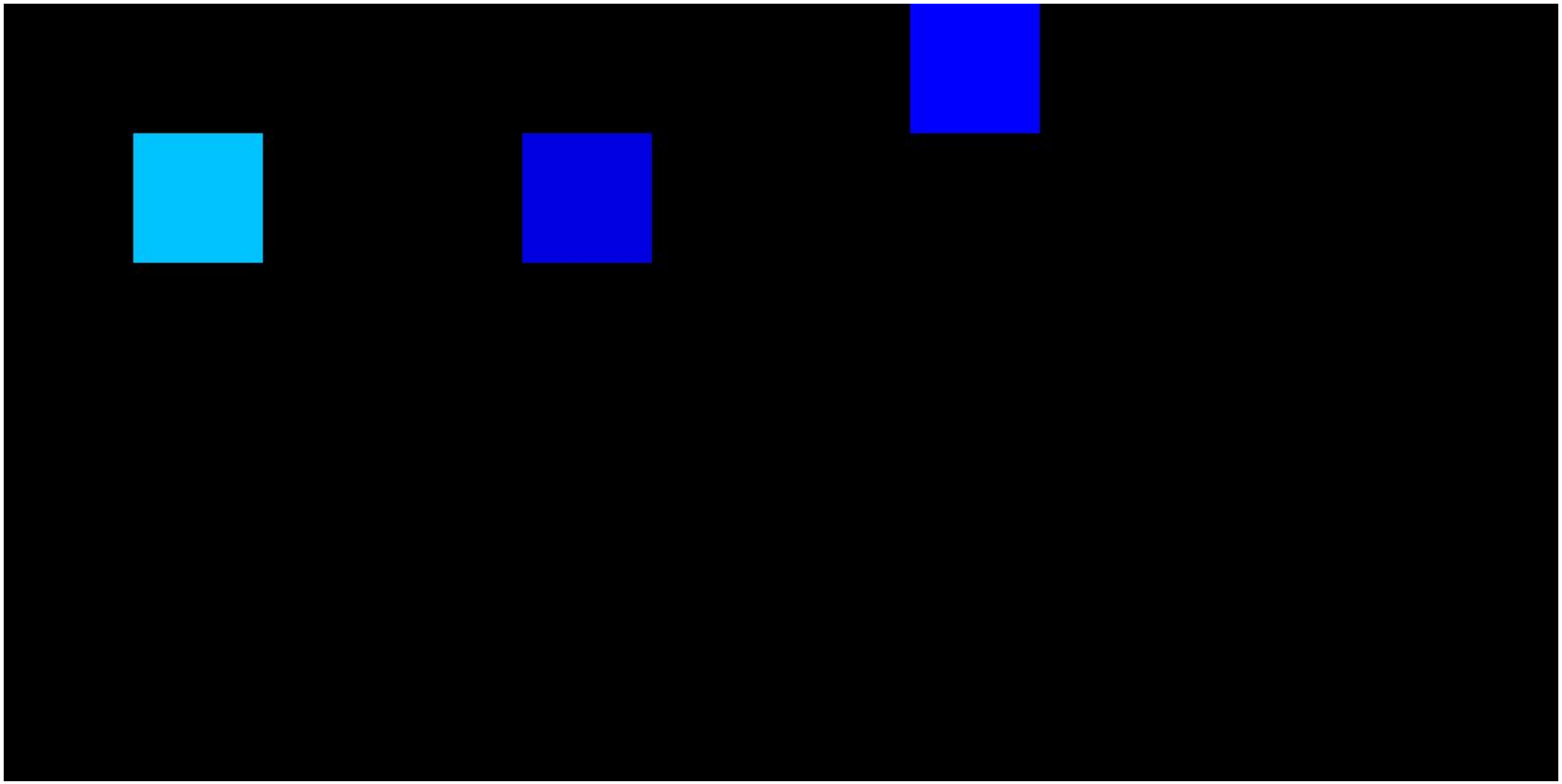
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



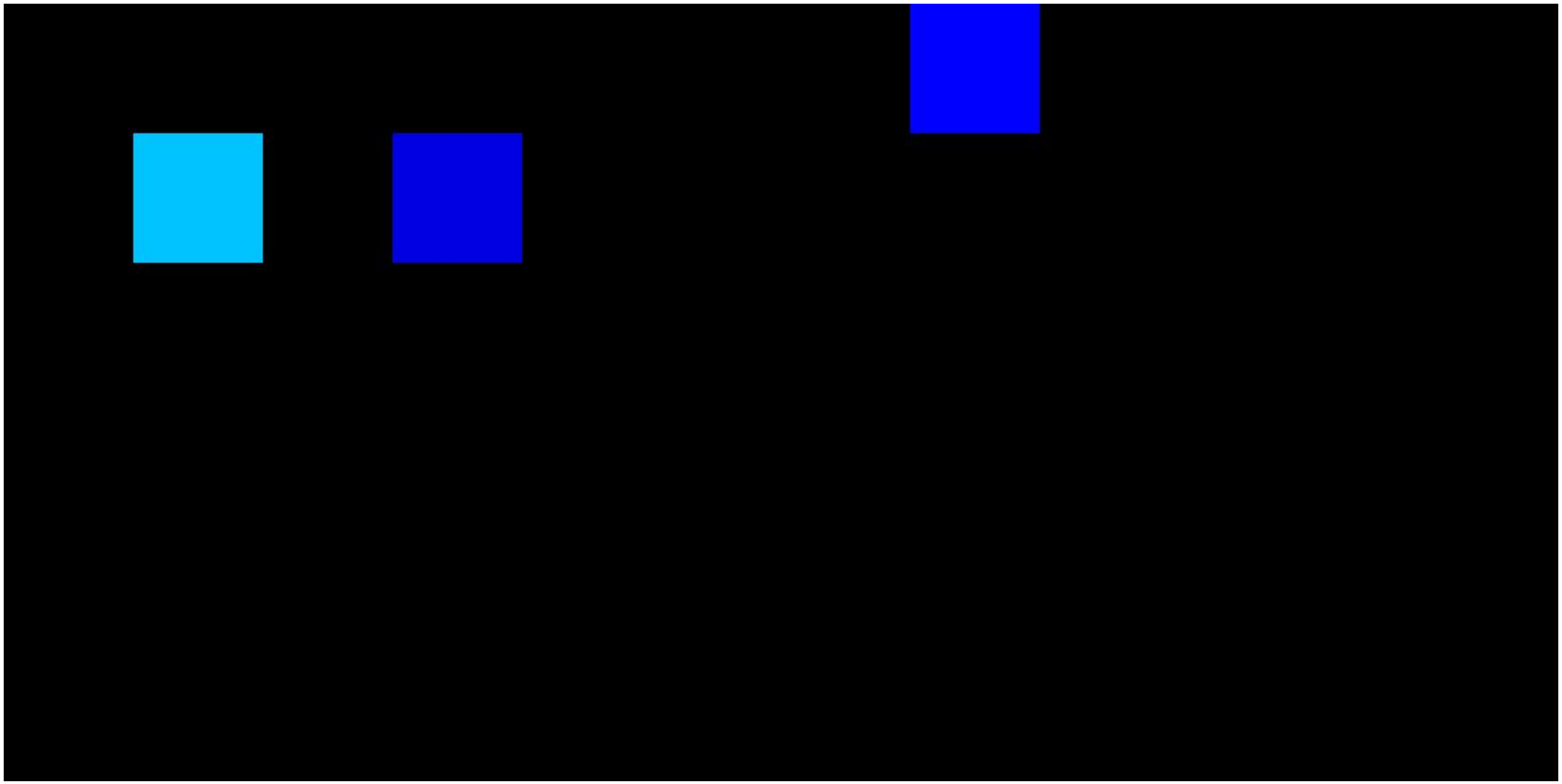
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



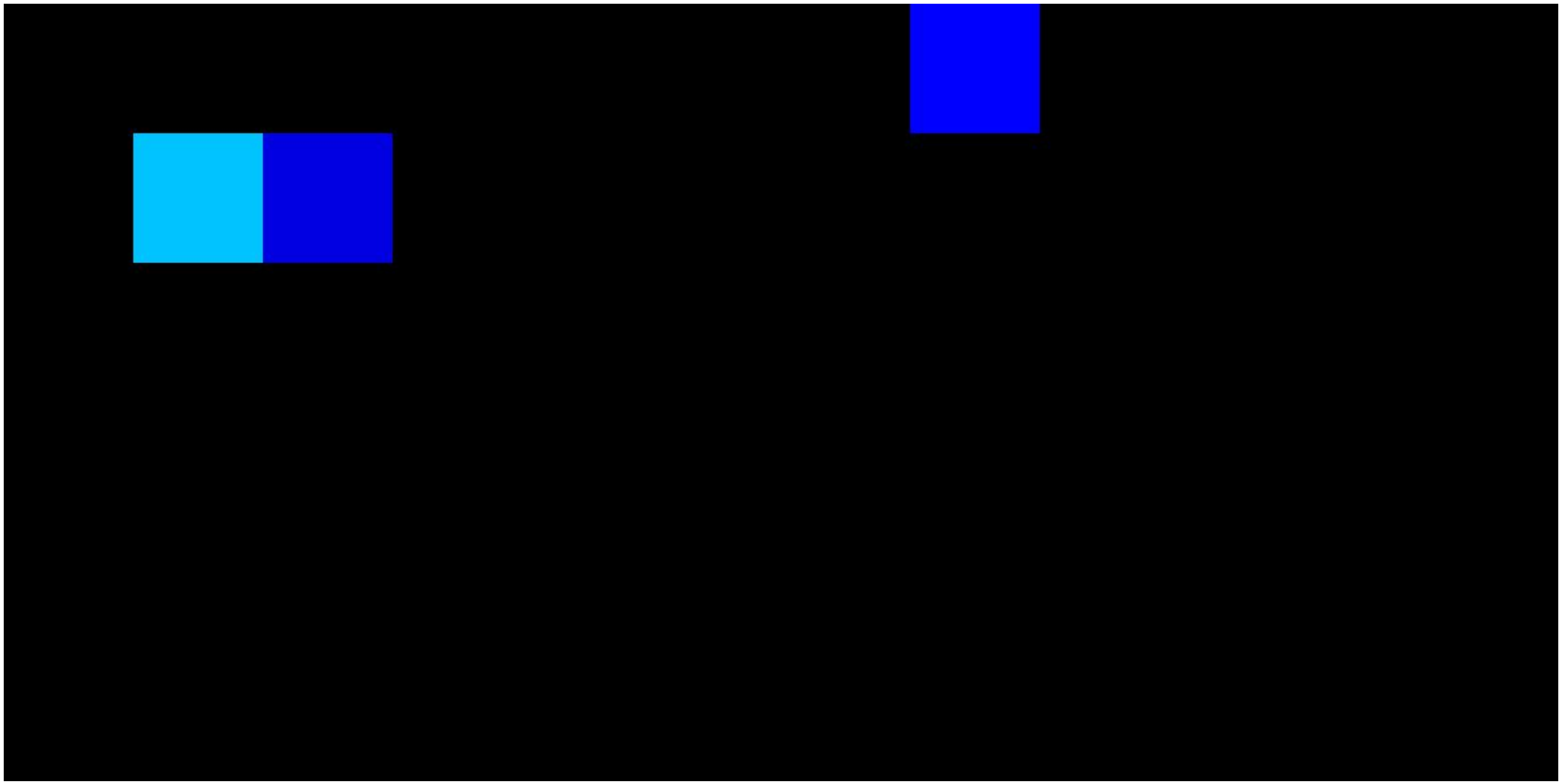
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



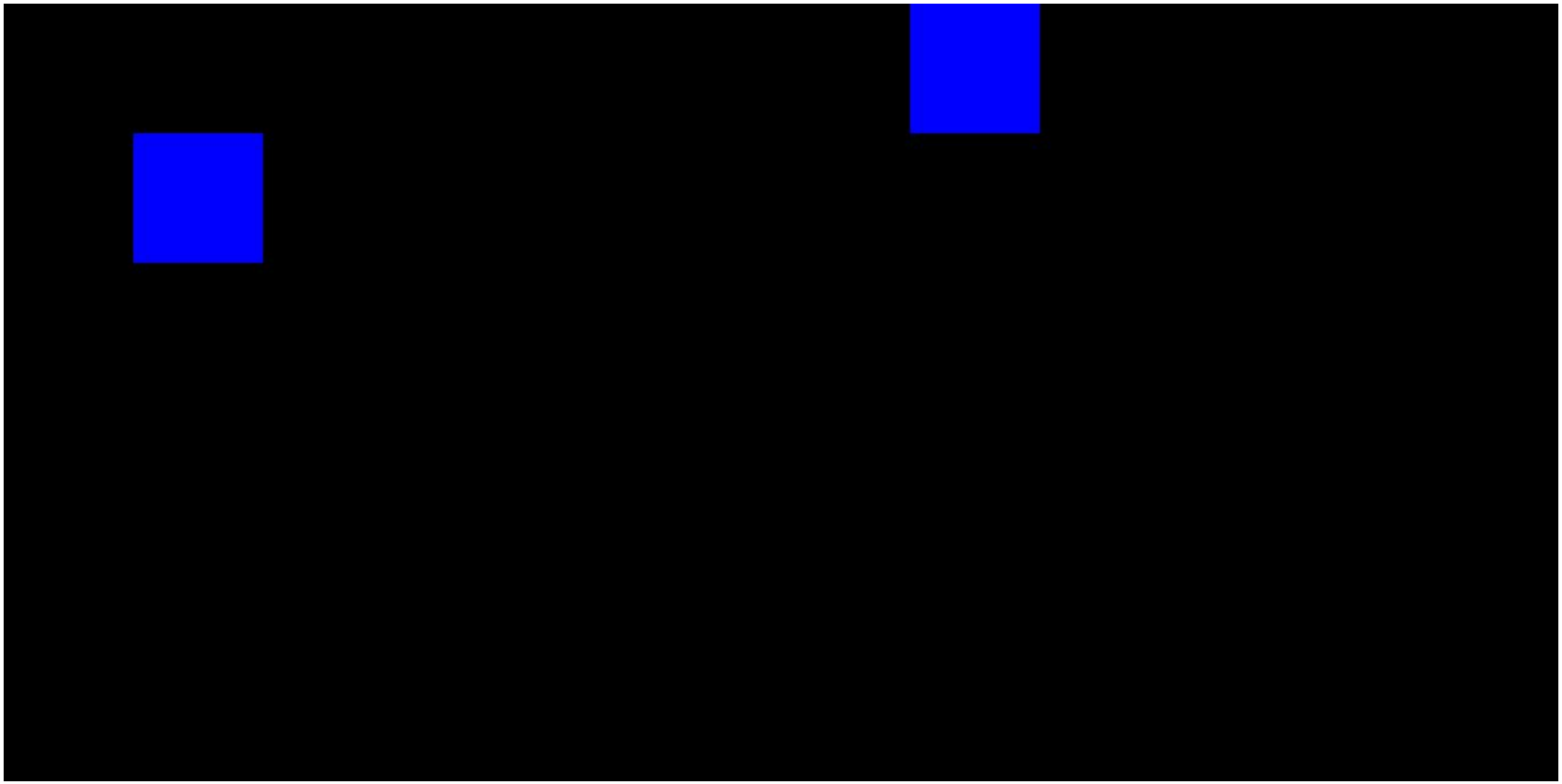
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



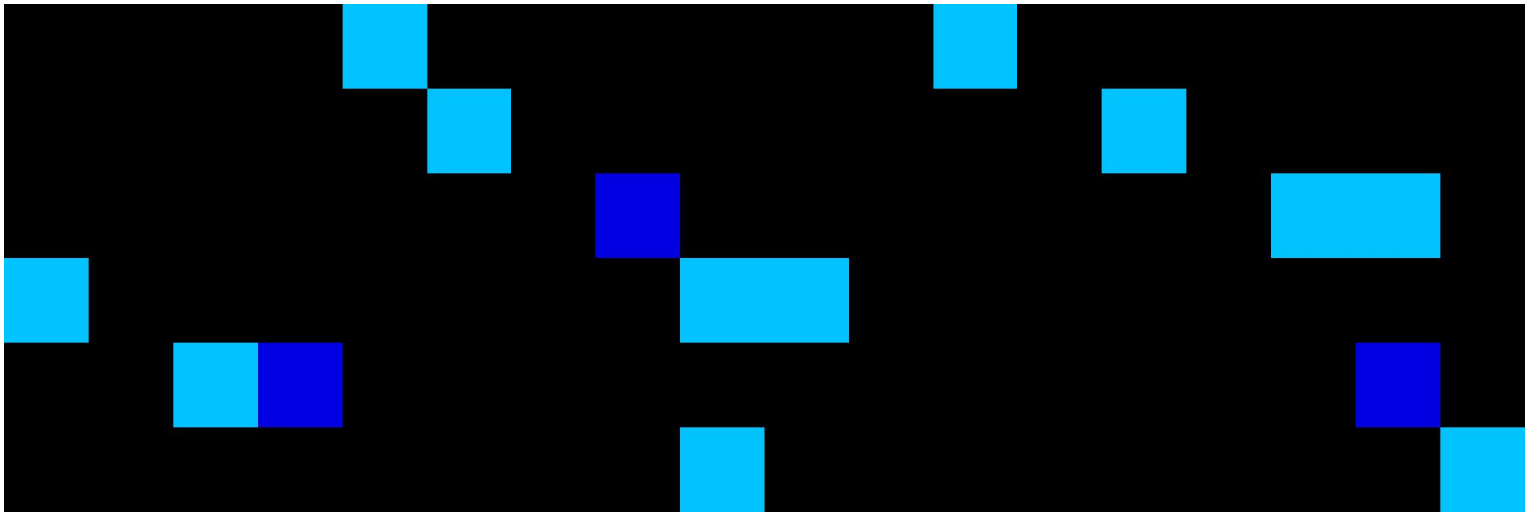
Resultados

- 6×6. Um milhão de épocas (1 dia), Acurácia de 11.88 (limite superior: $4 \times 3 = 12$).



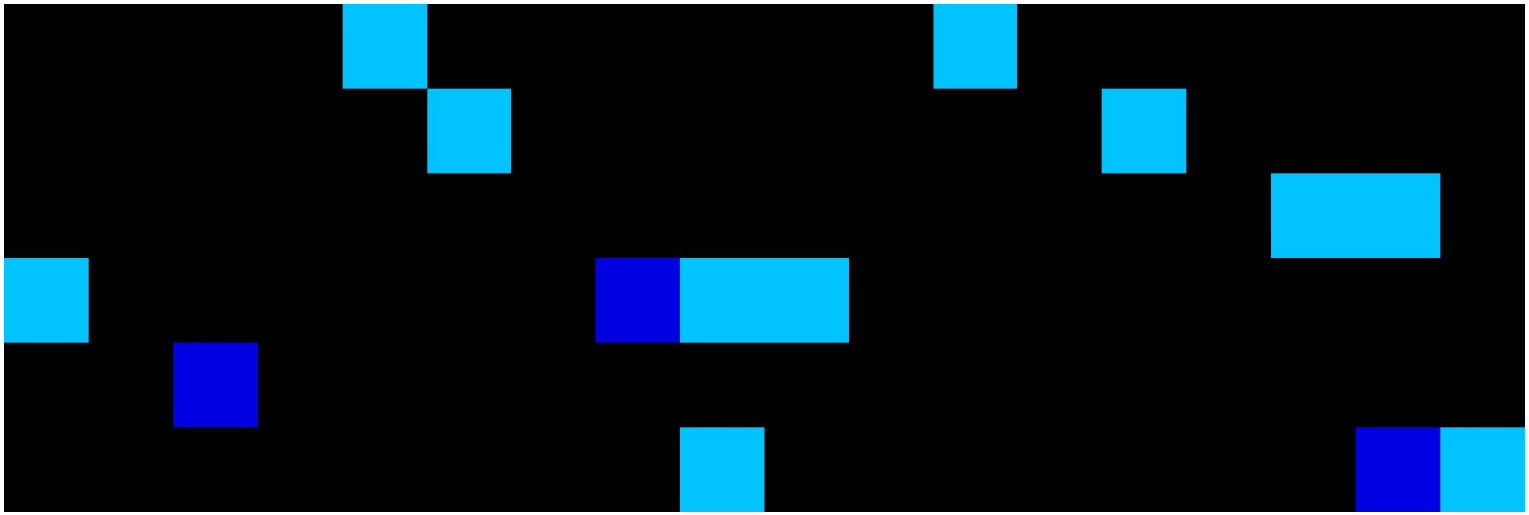
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



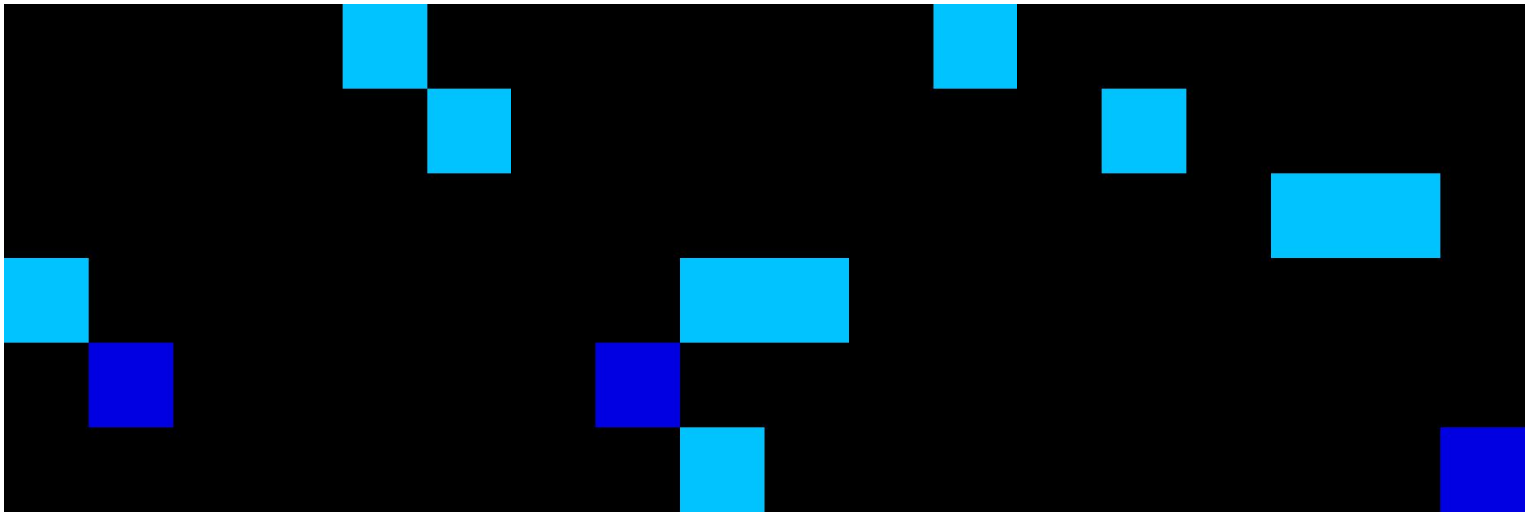
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



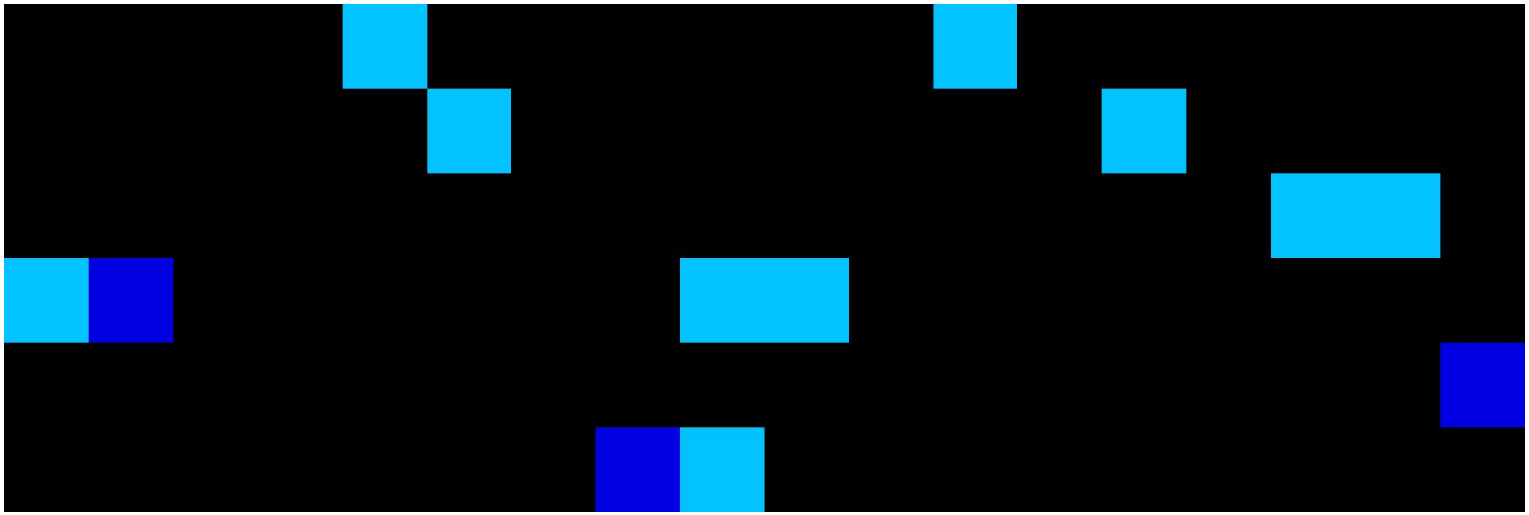
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



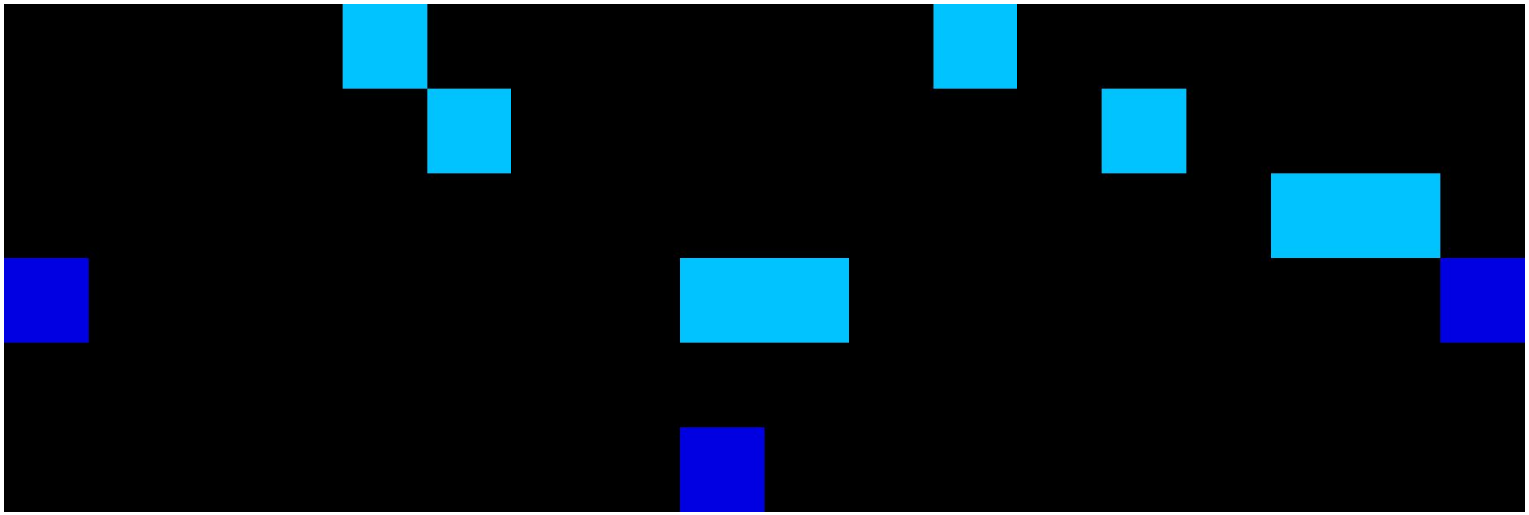
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



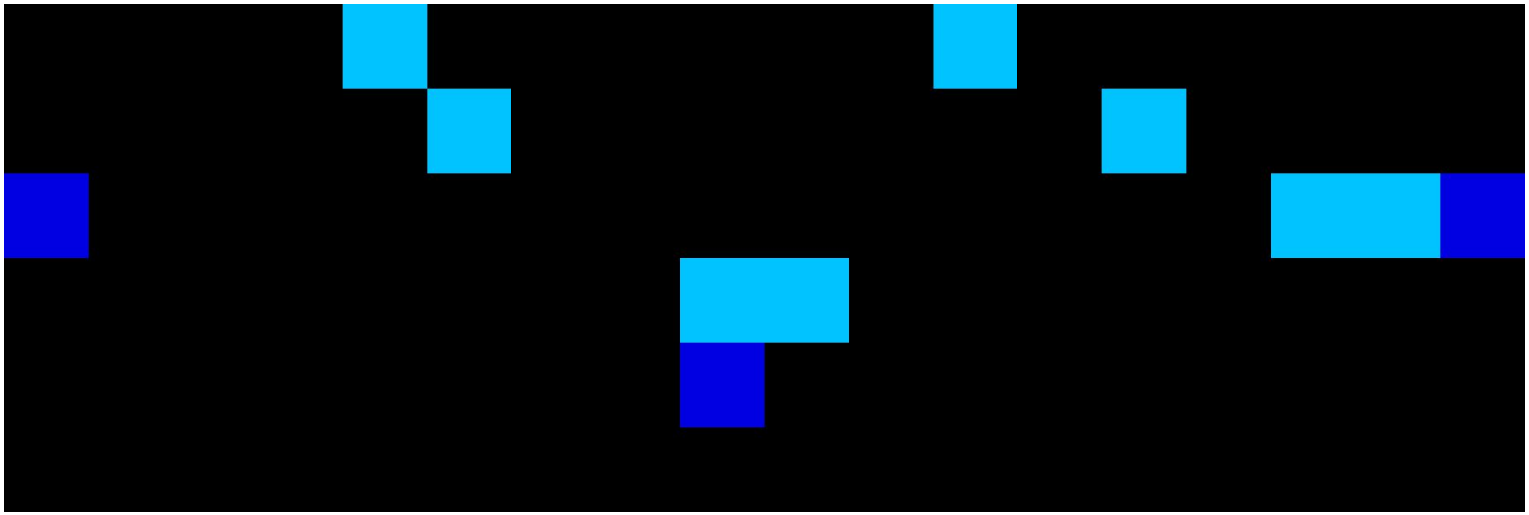
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



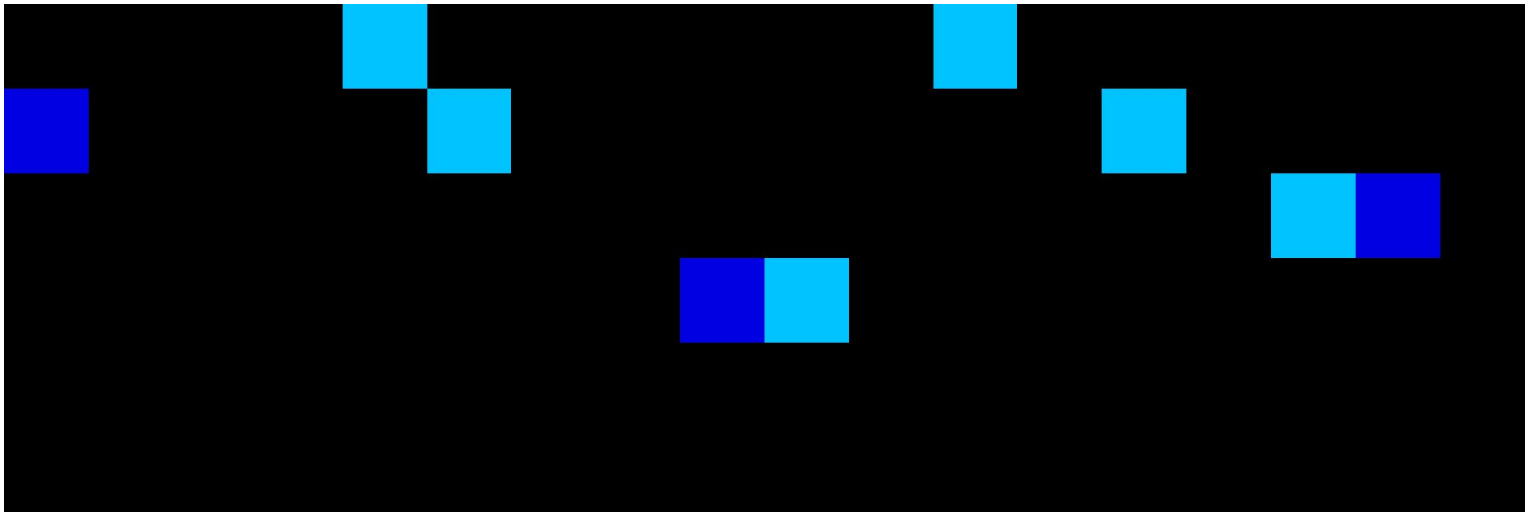
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



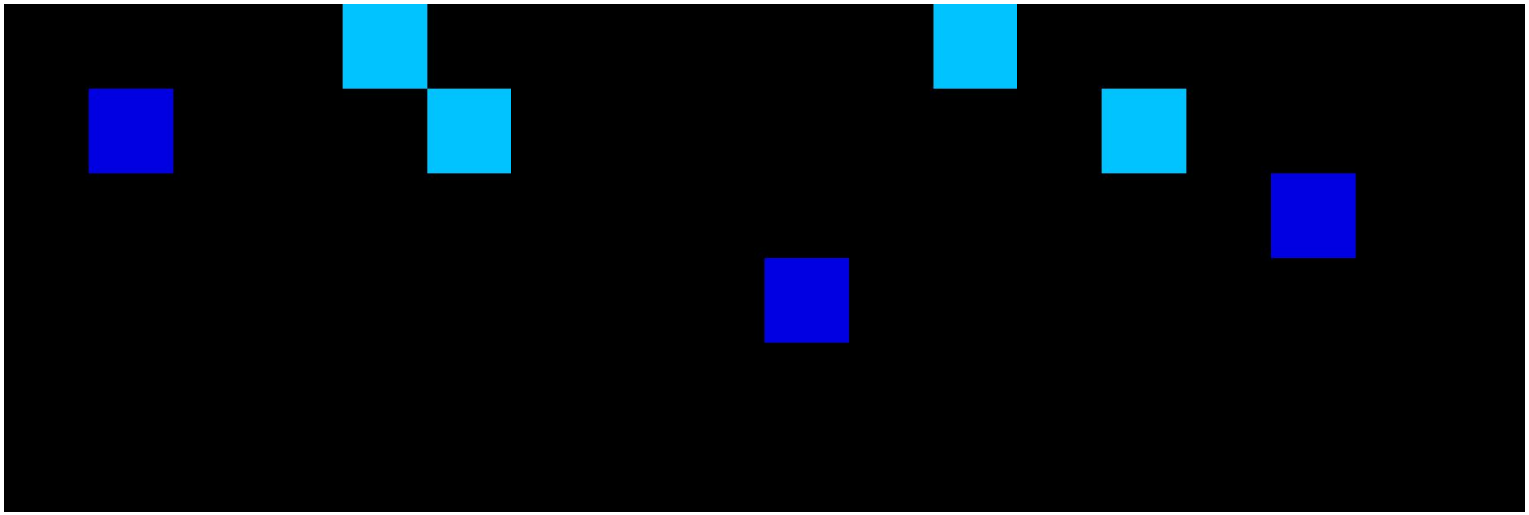
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



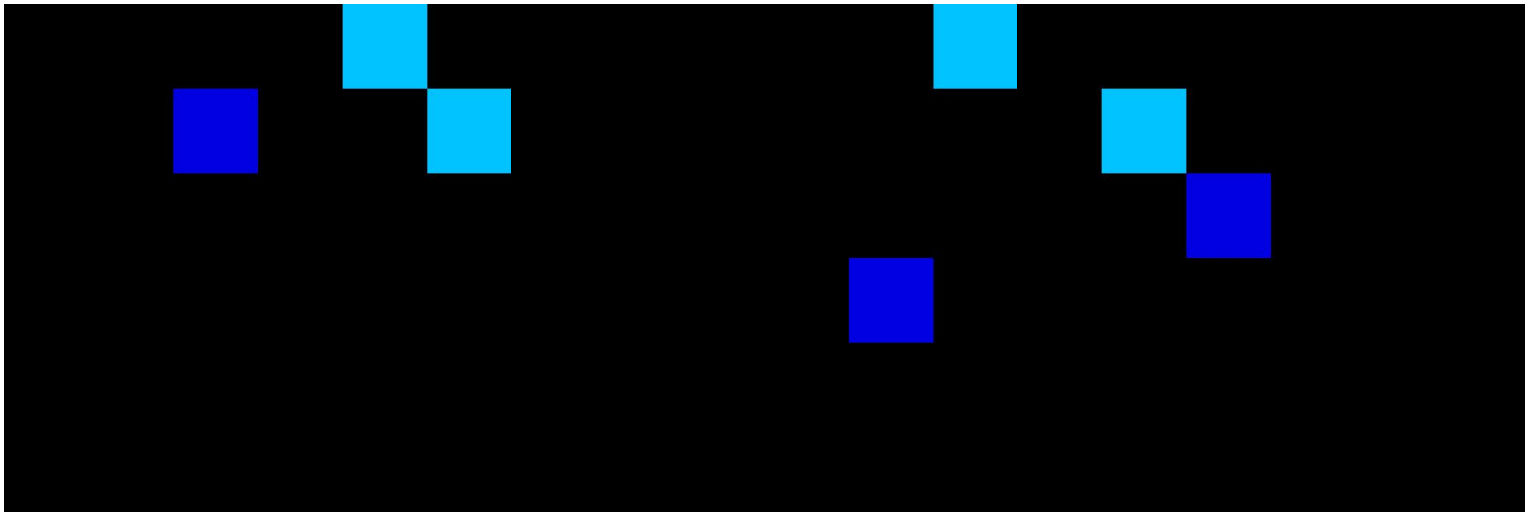
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



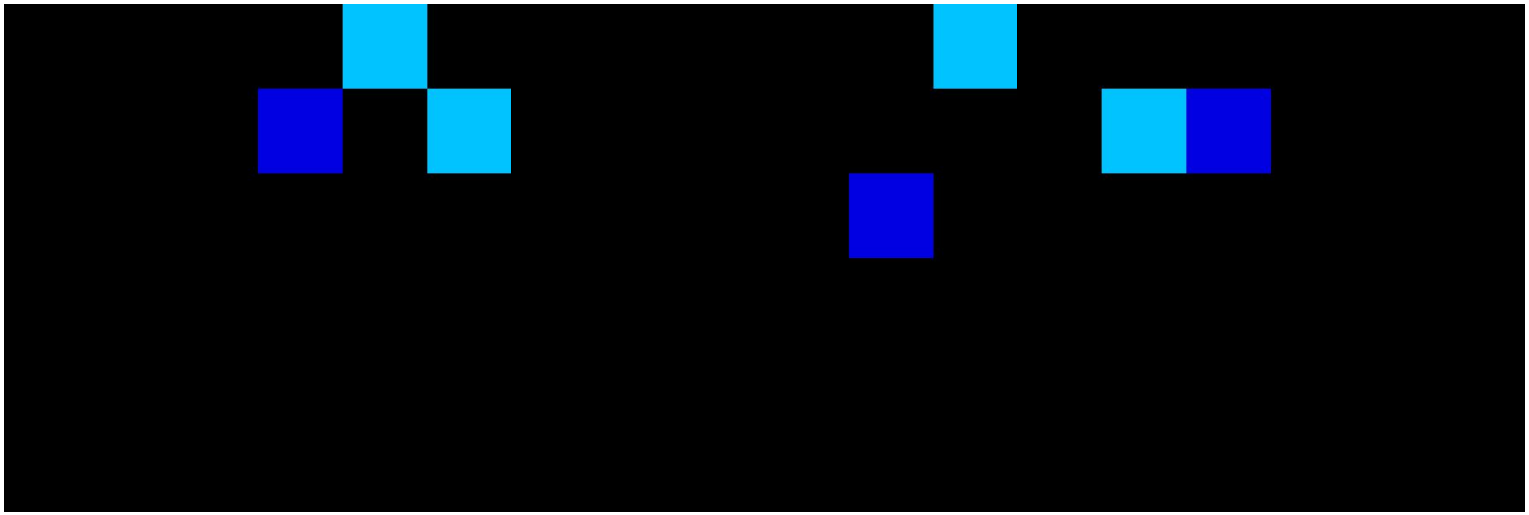
Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).



Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).

Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).

Resultados

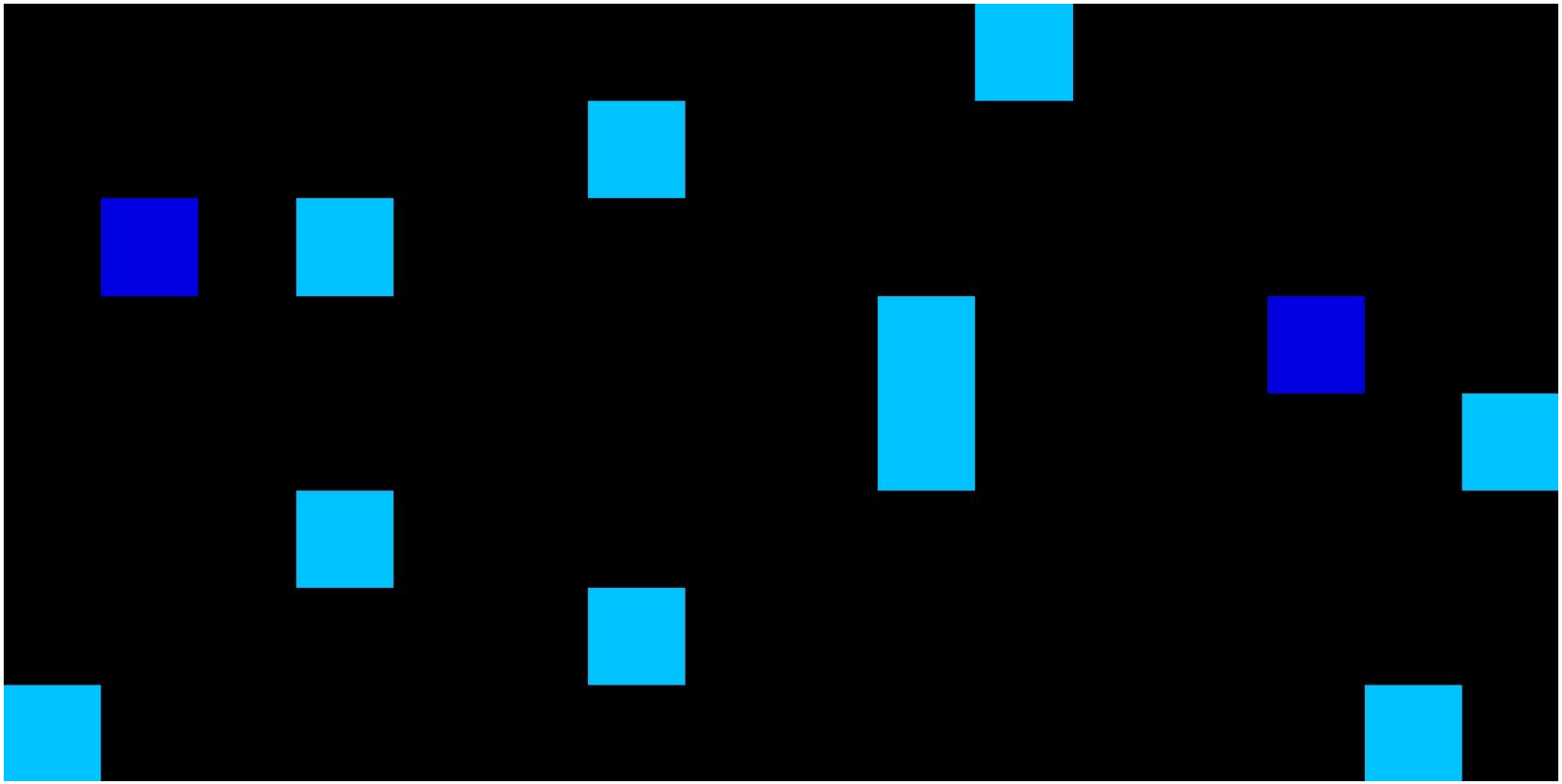
- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).

Resultados

- 6×6. Dois milhões de épocas (2 dias), Acurácia de 11.92 (limite superior: $4 \times 3 = 12$).

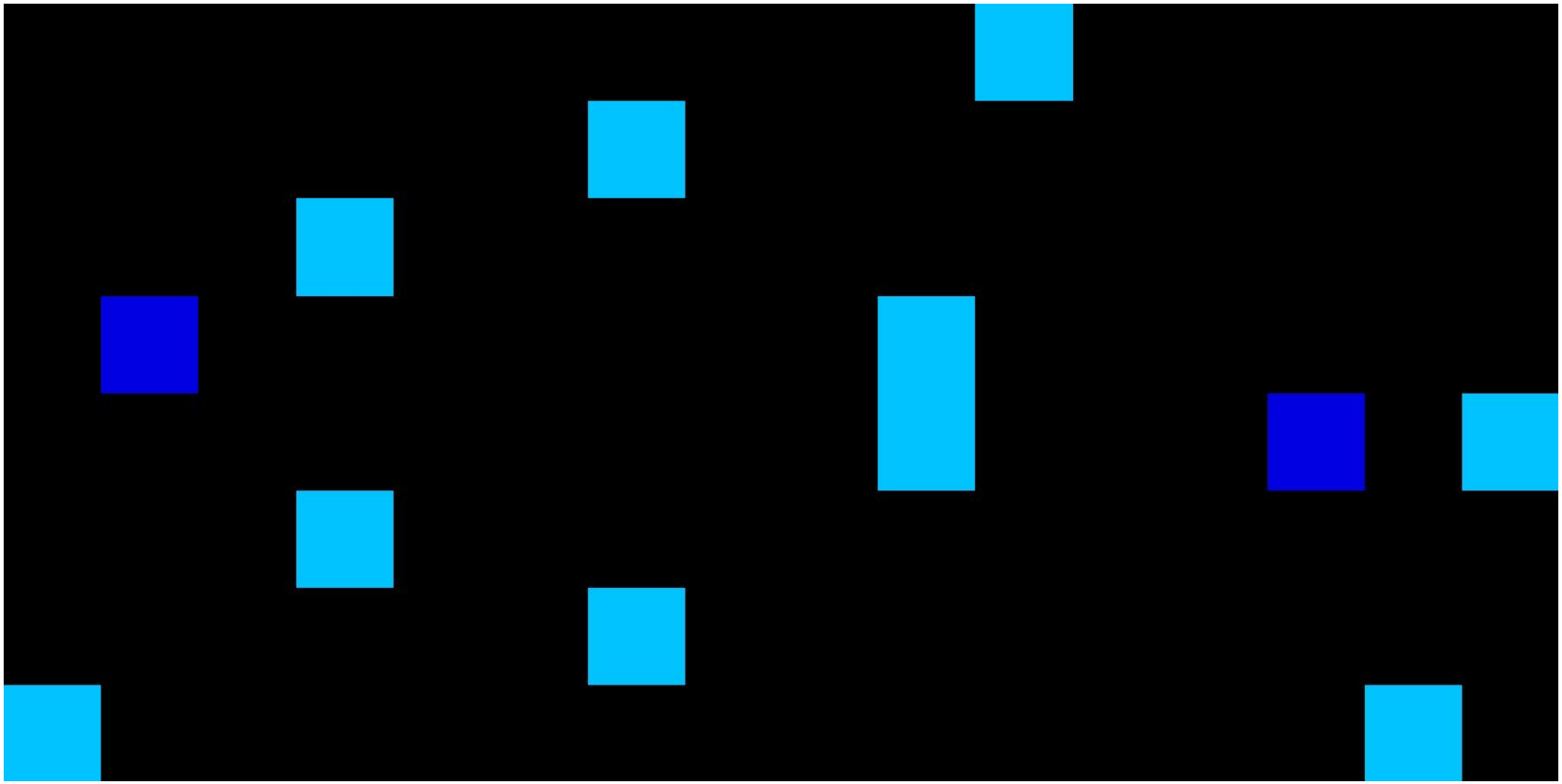
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



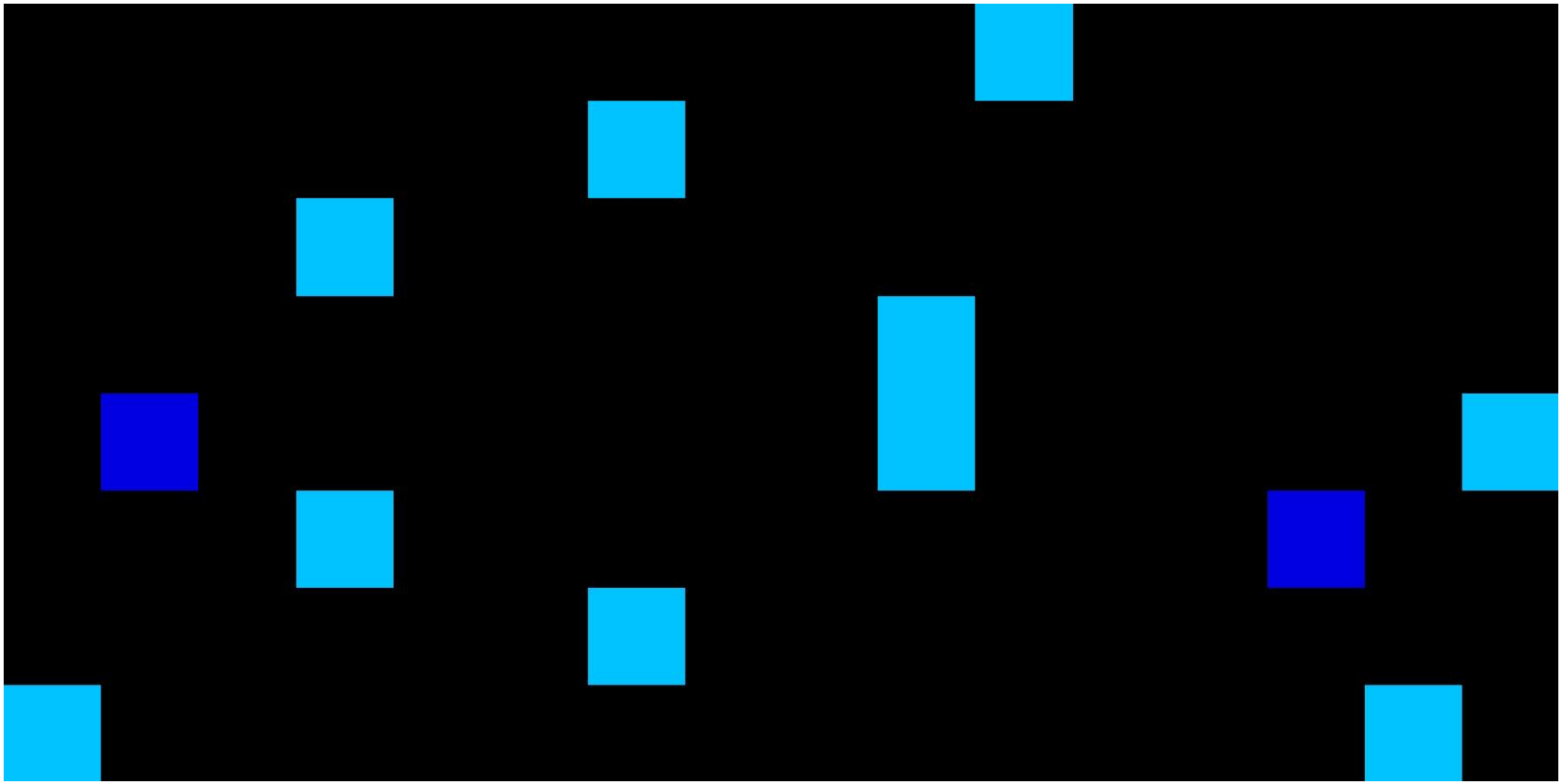
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



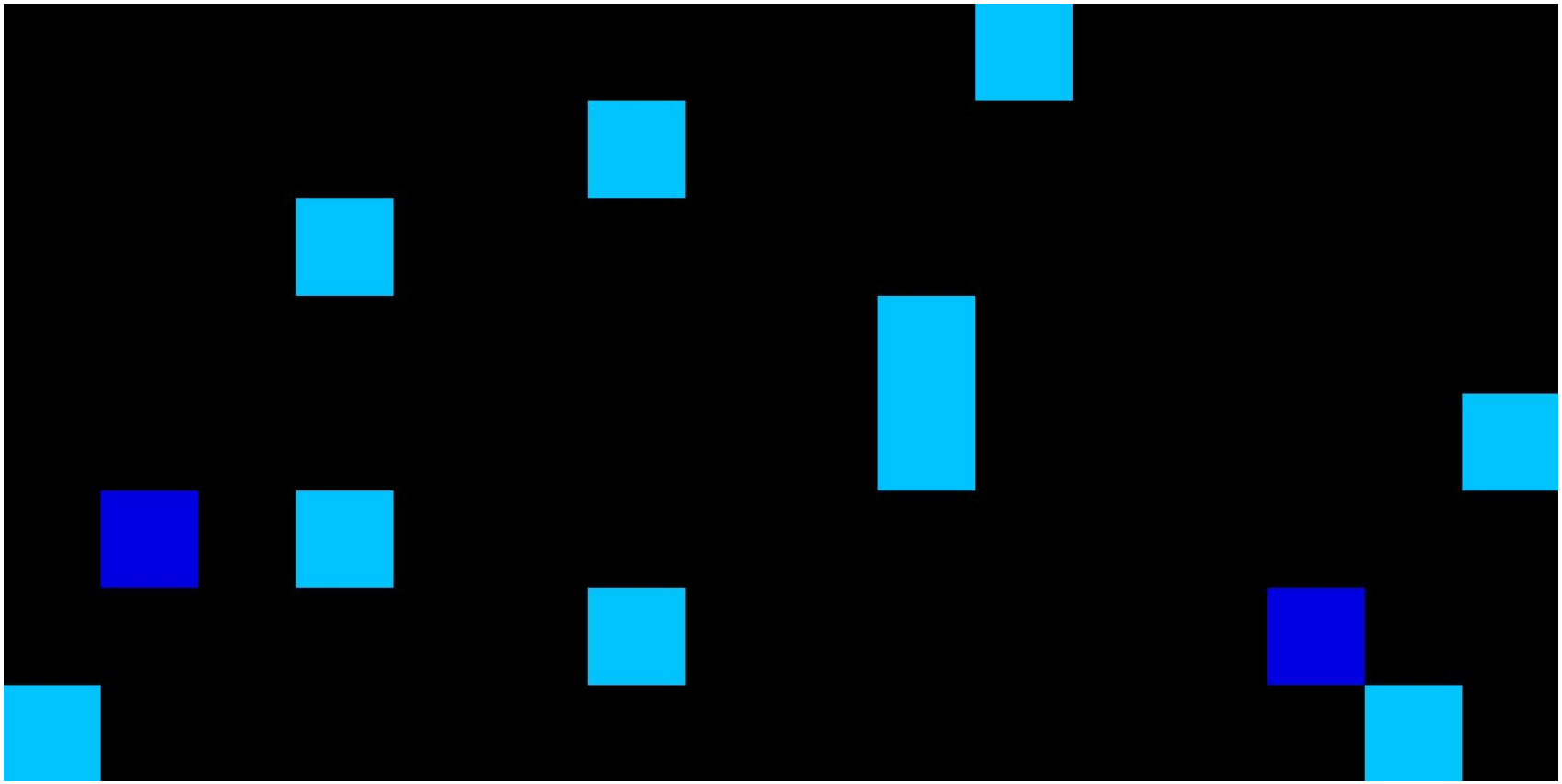
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



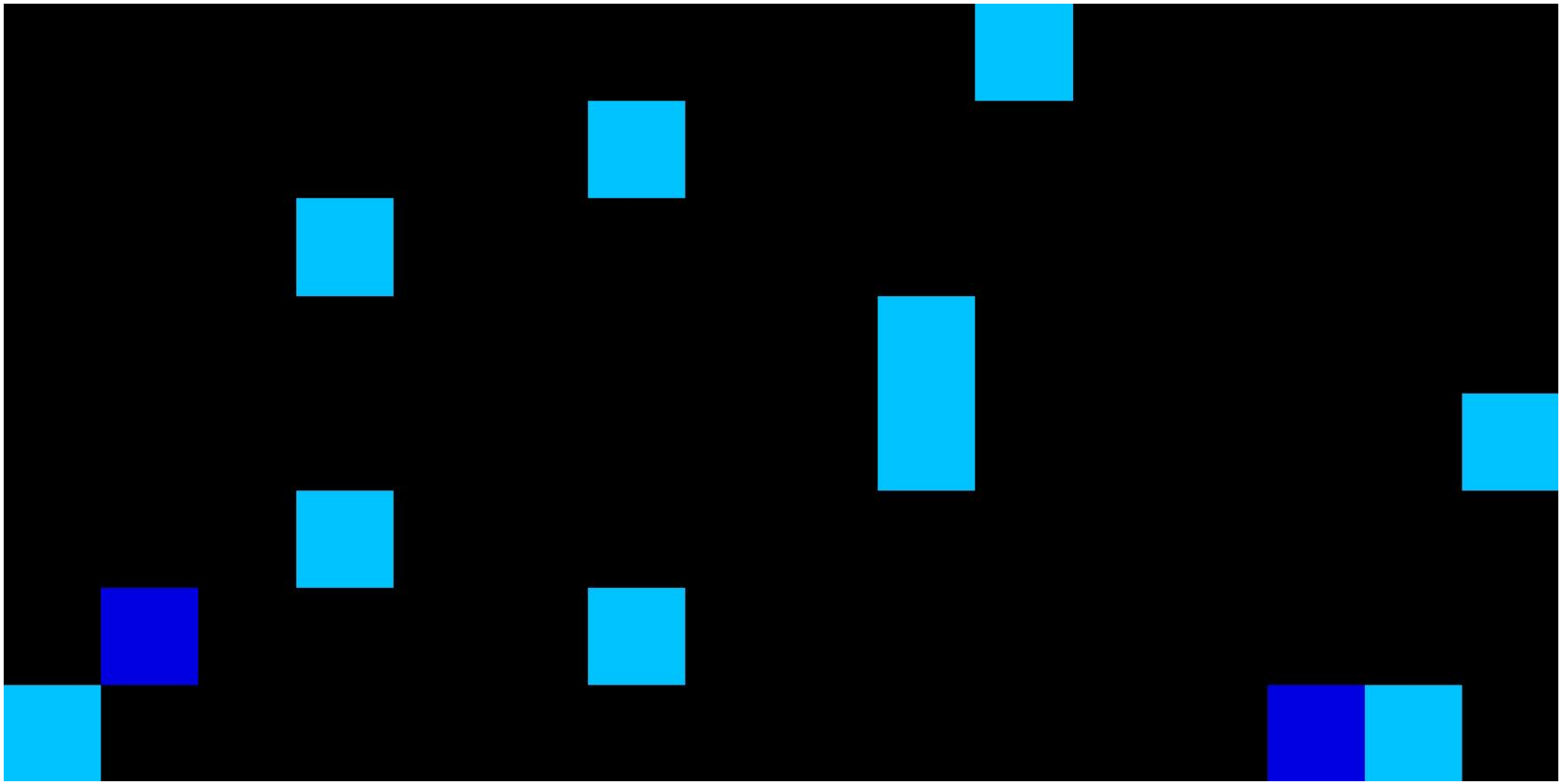
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



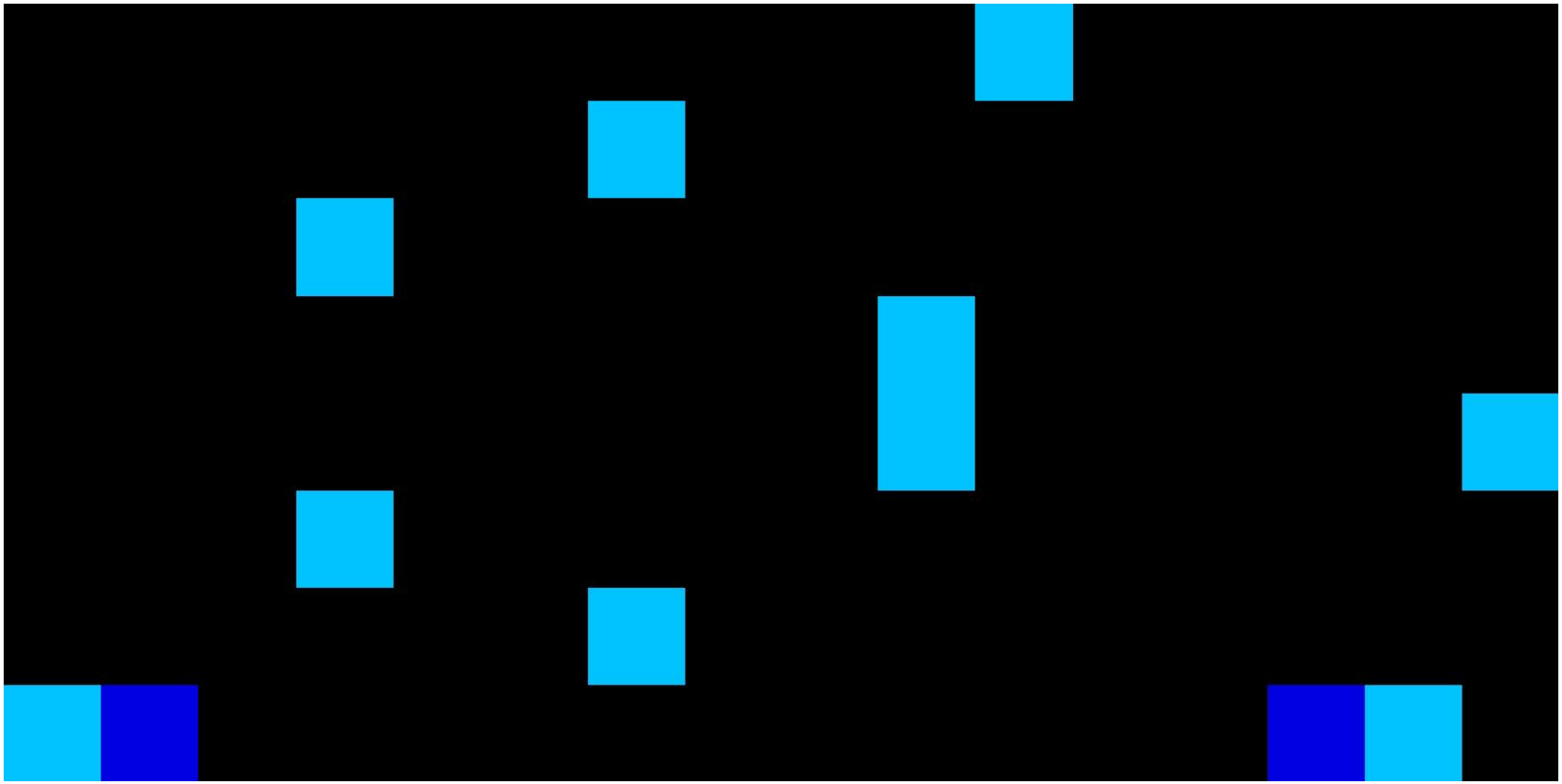
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



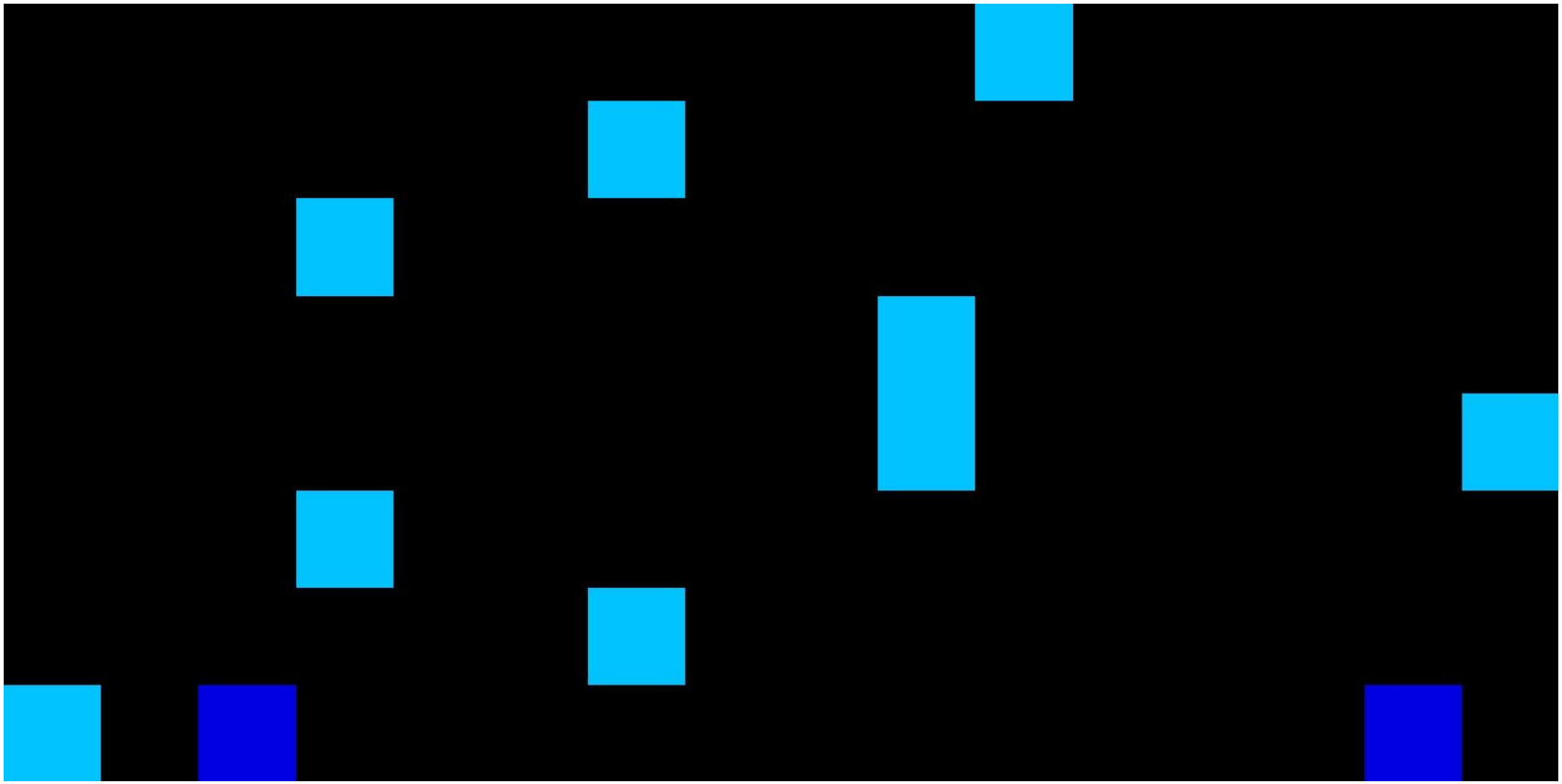
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



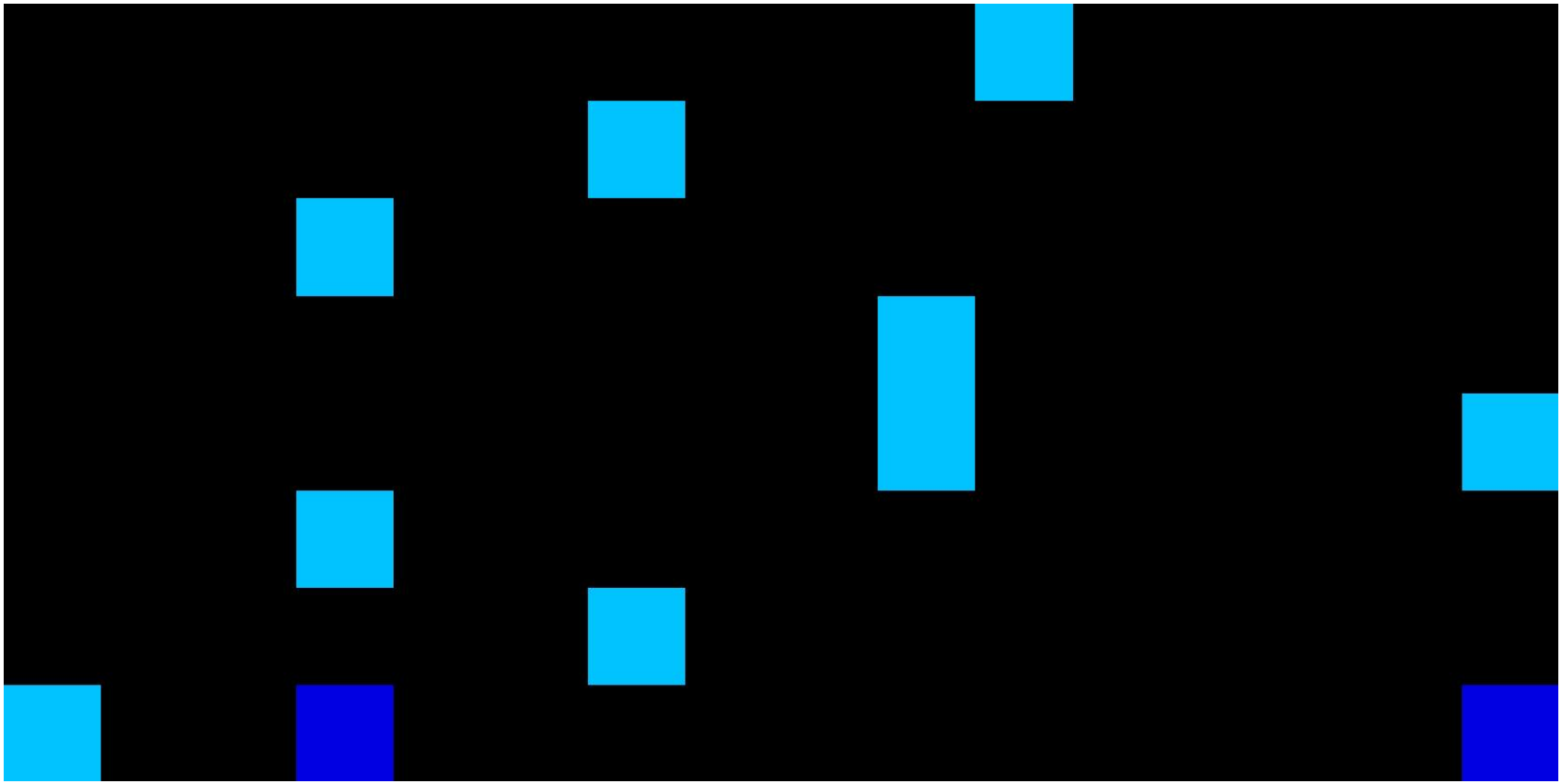
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



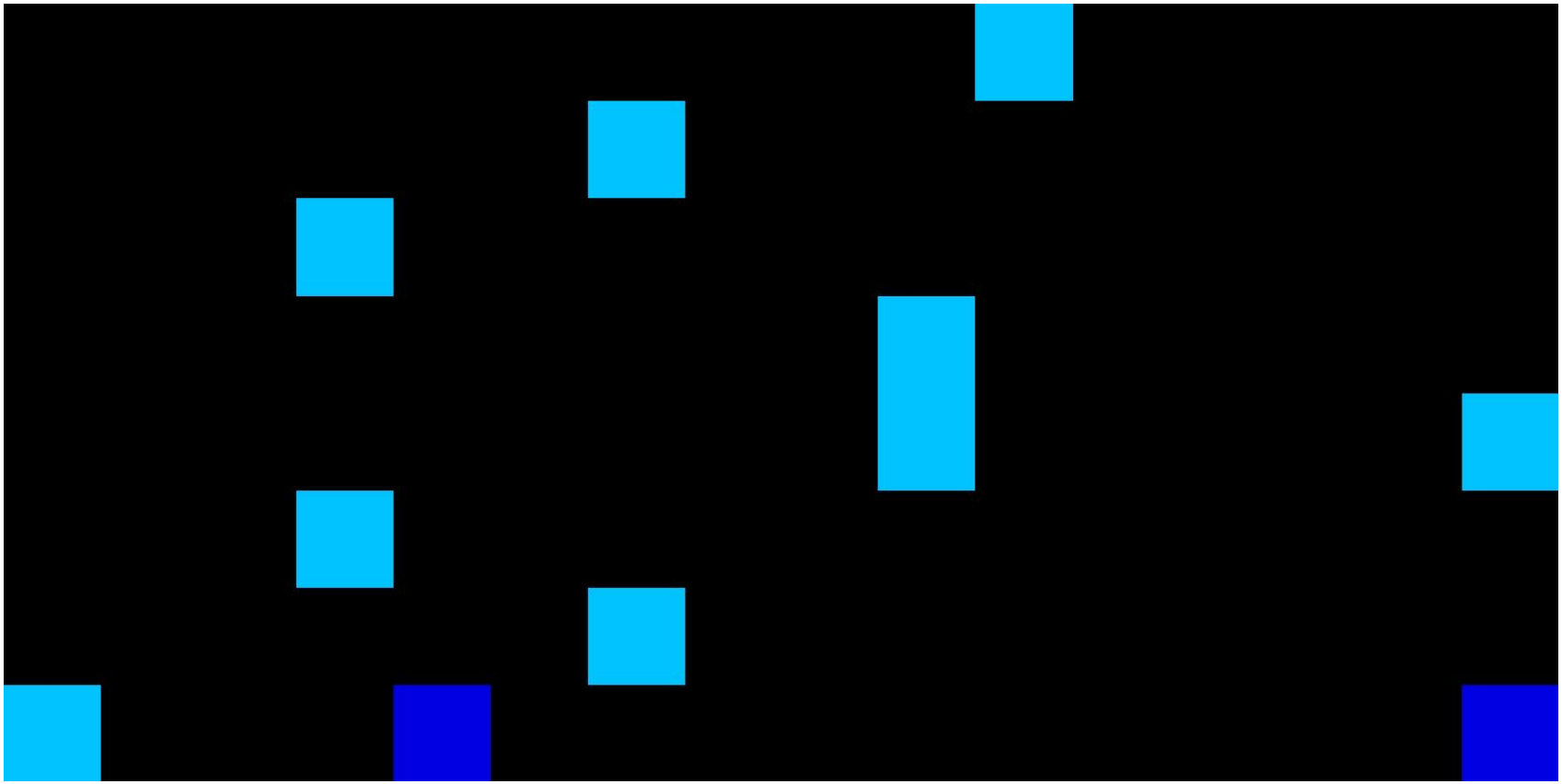
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



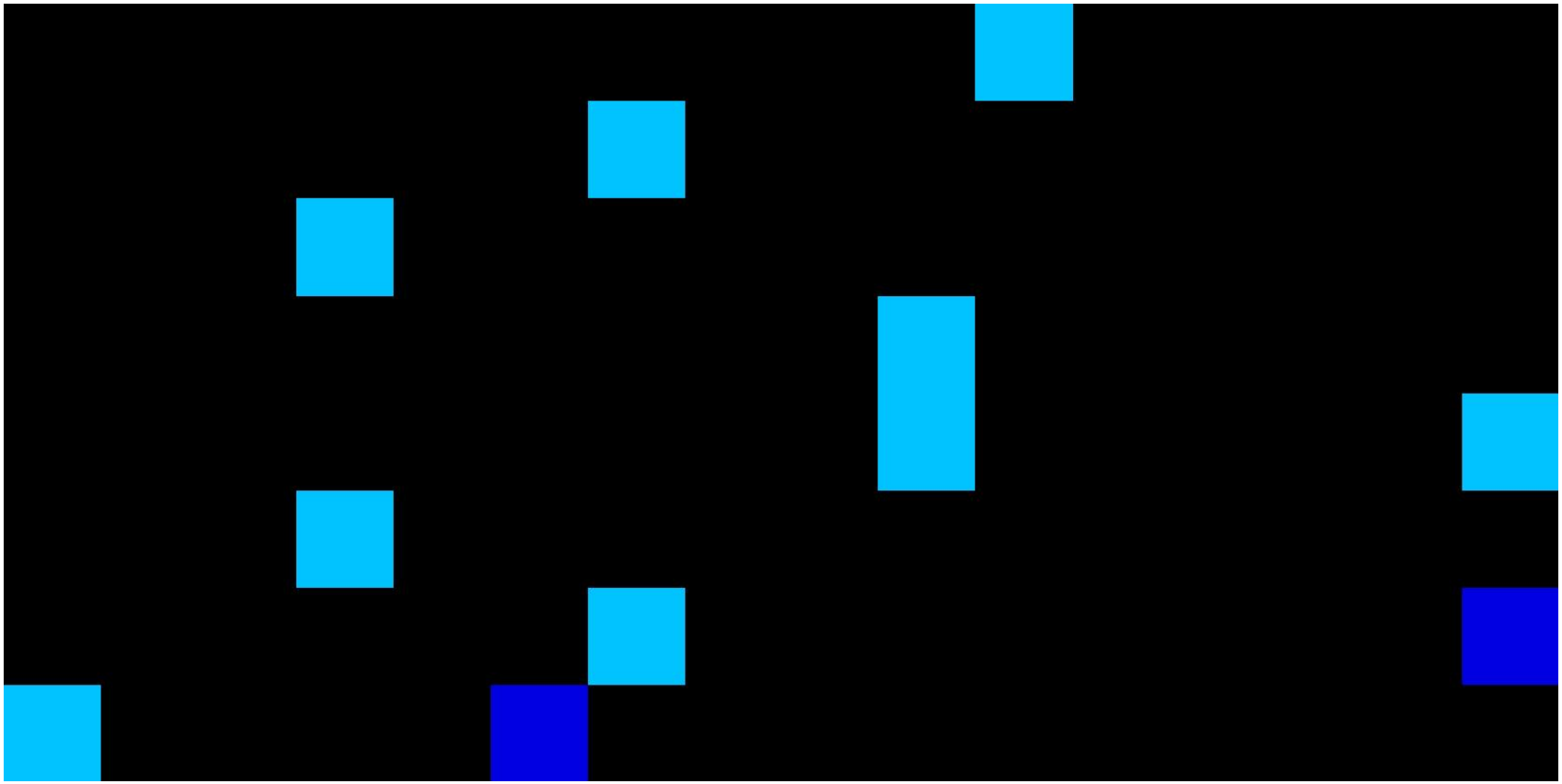
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



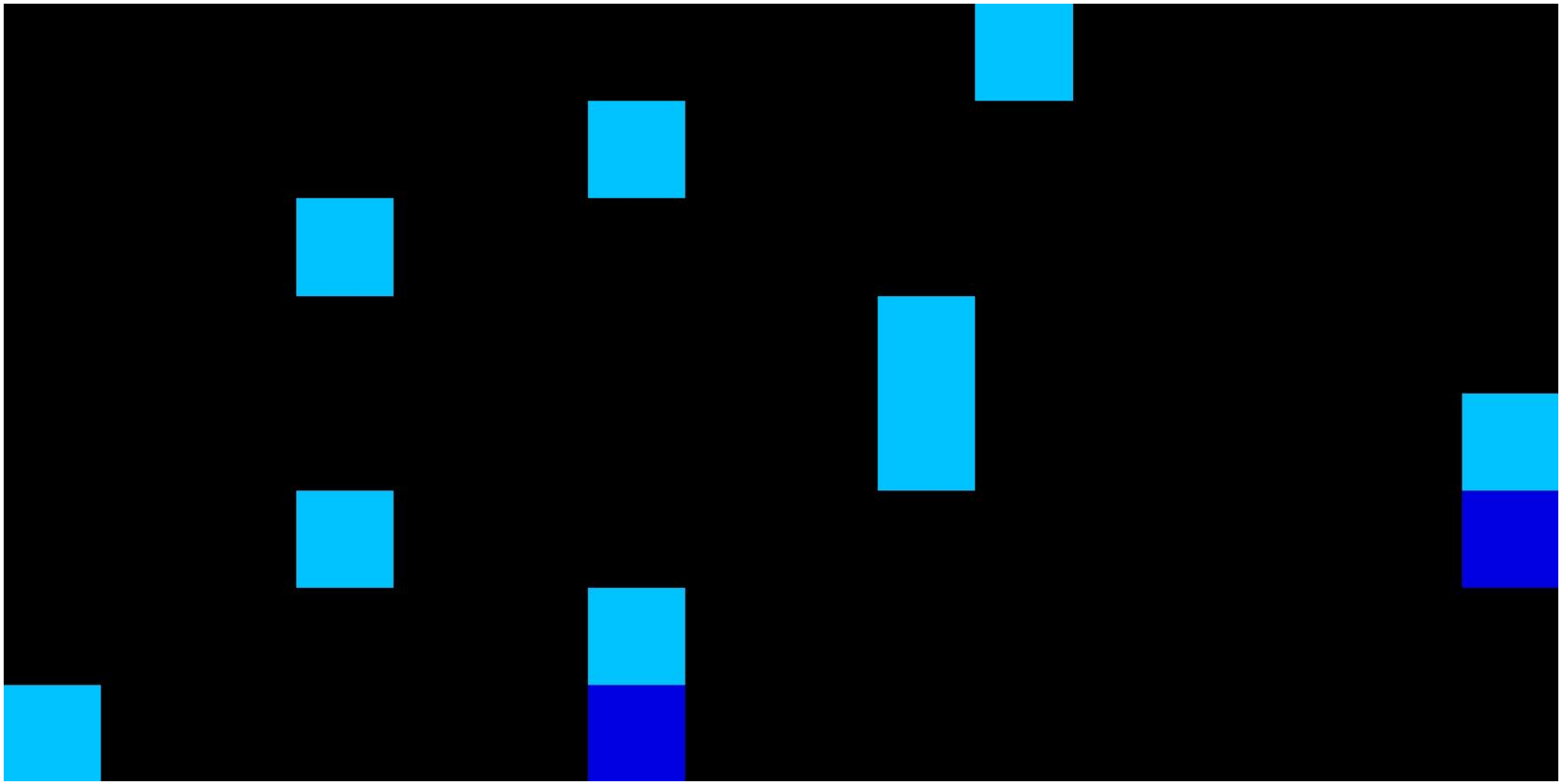
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



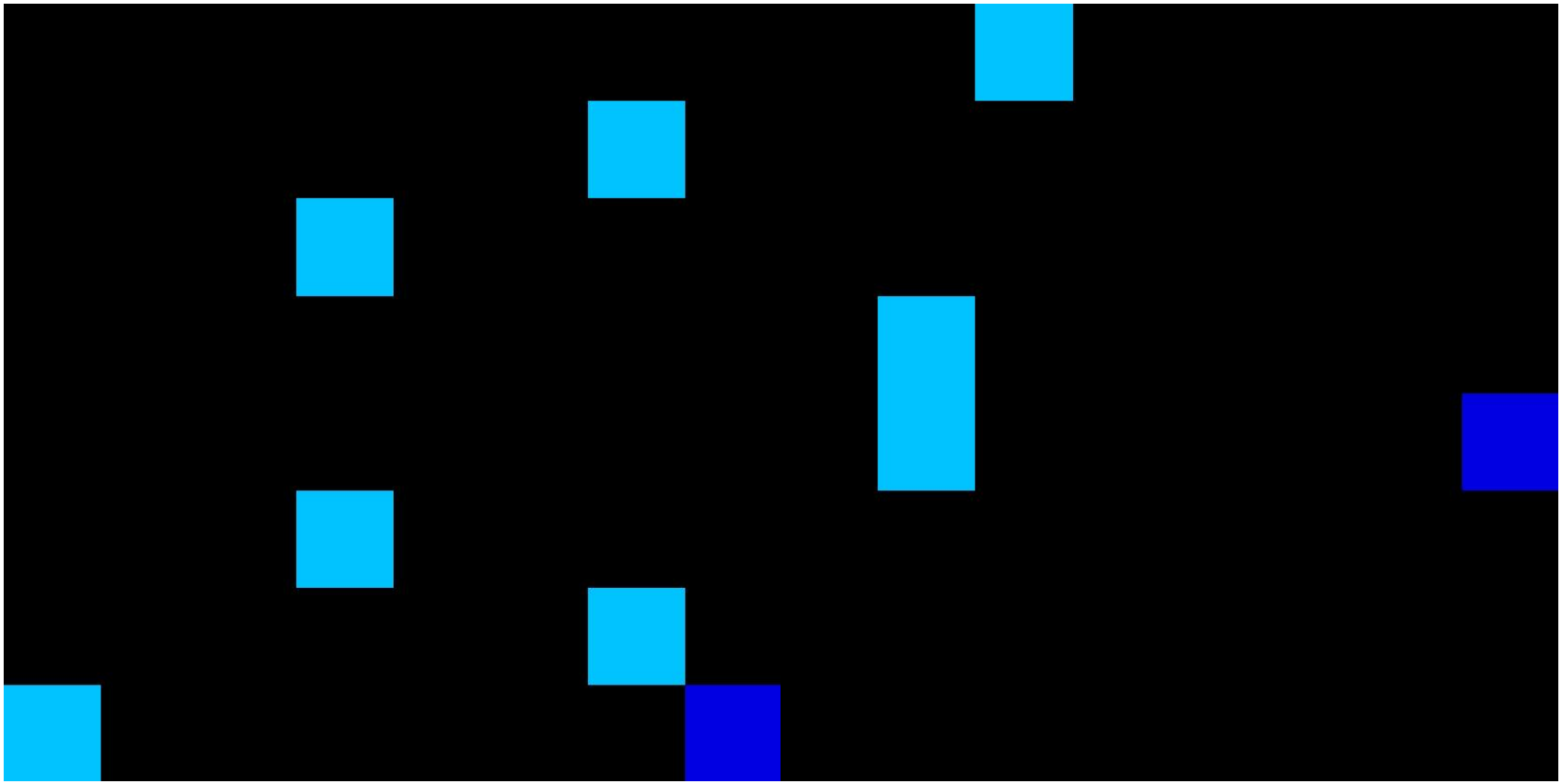
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



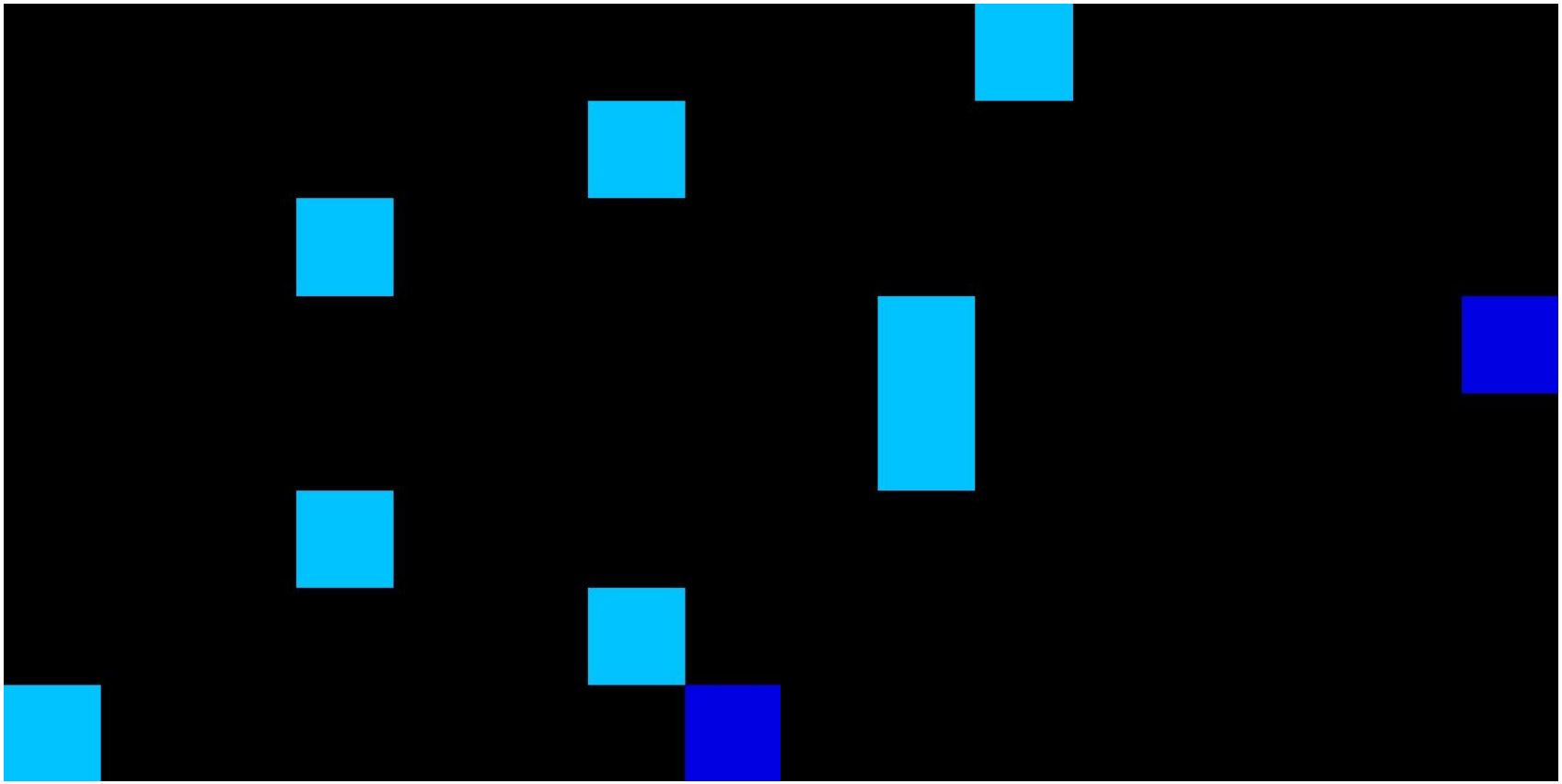
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



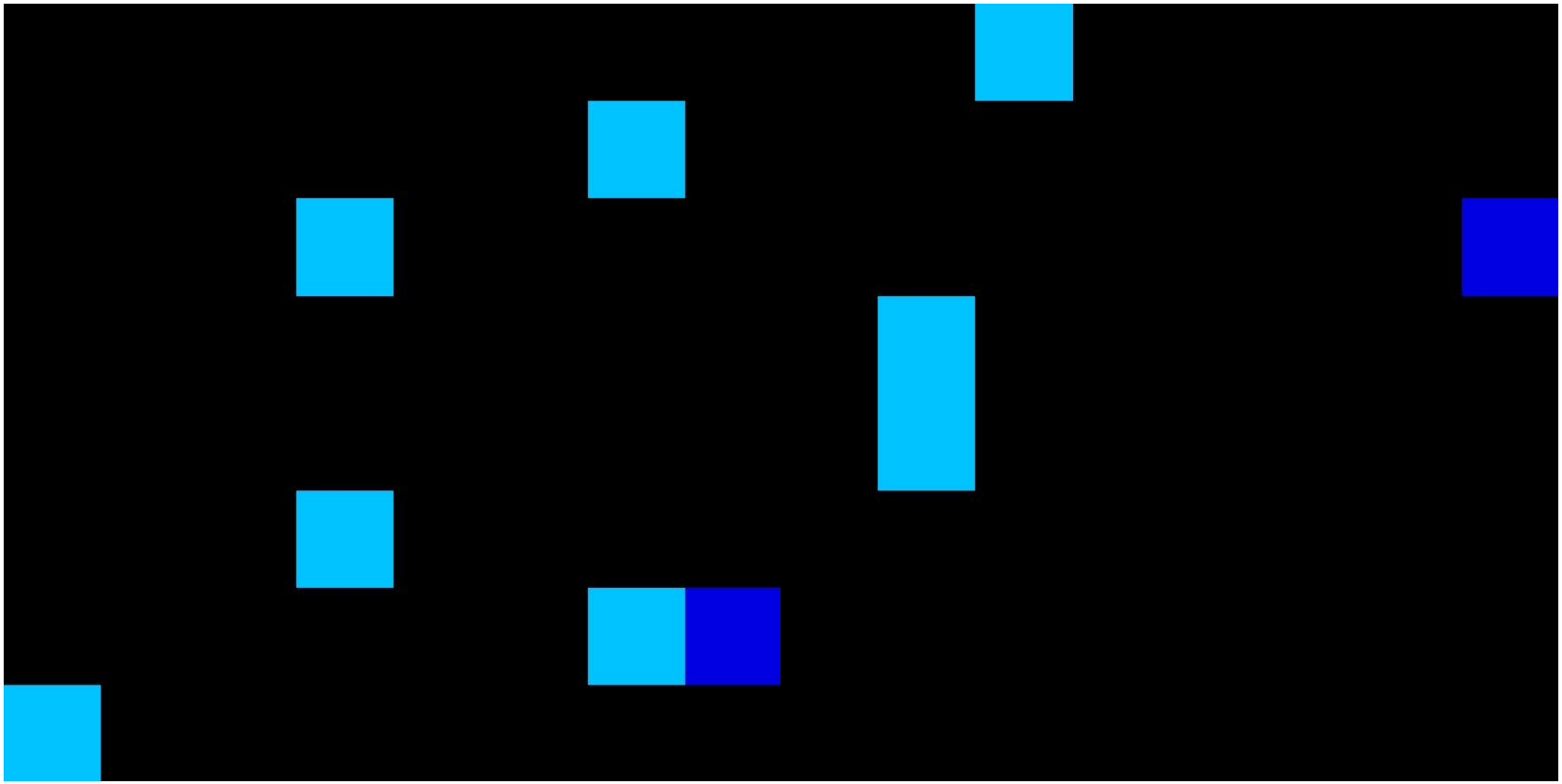
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



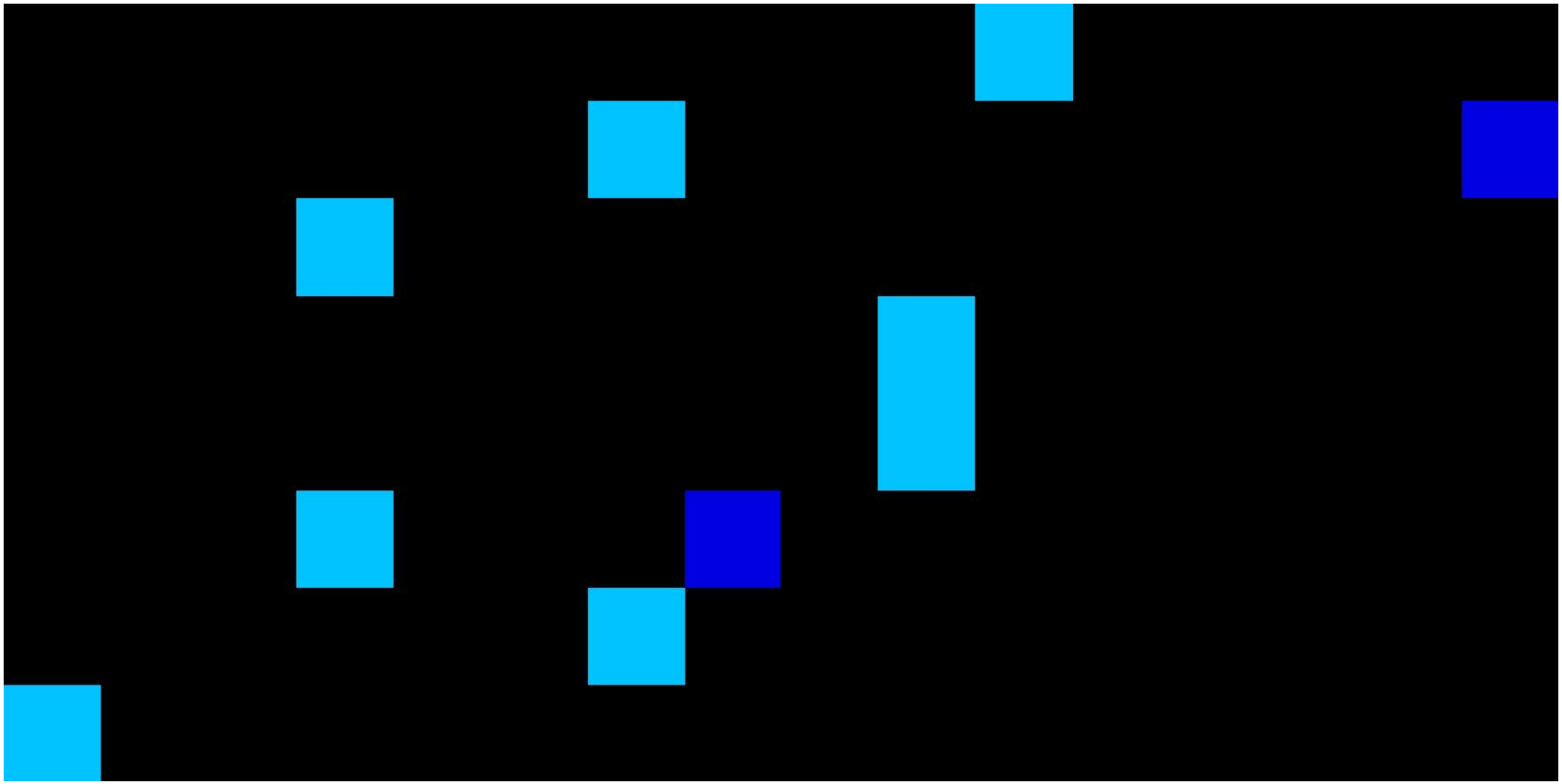
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



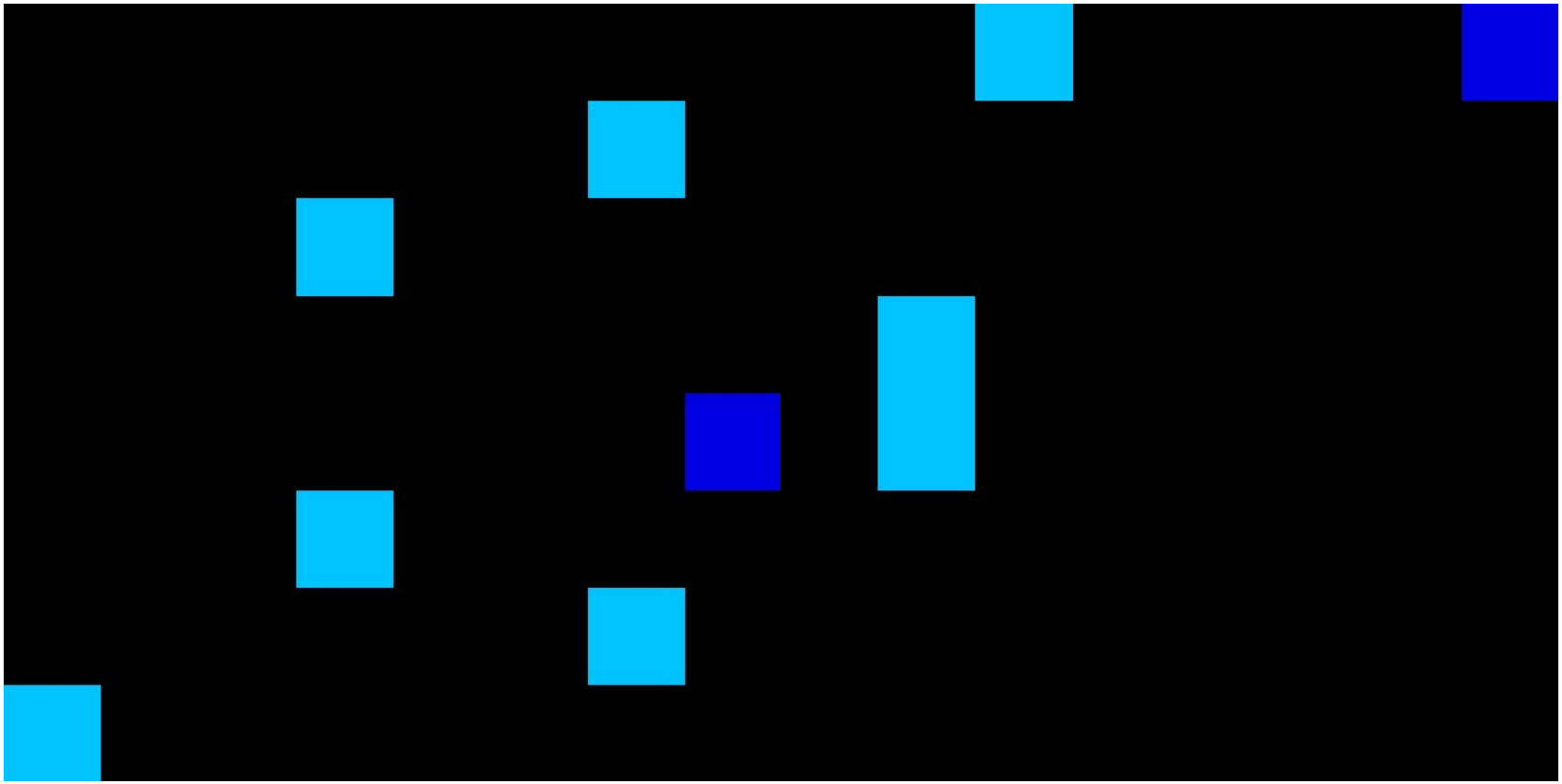
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



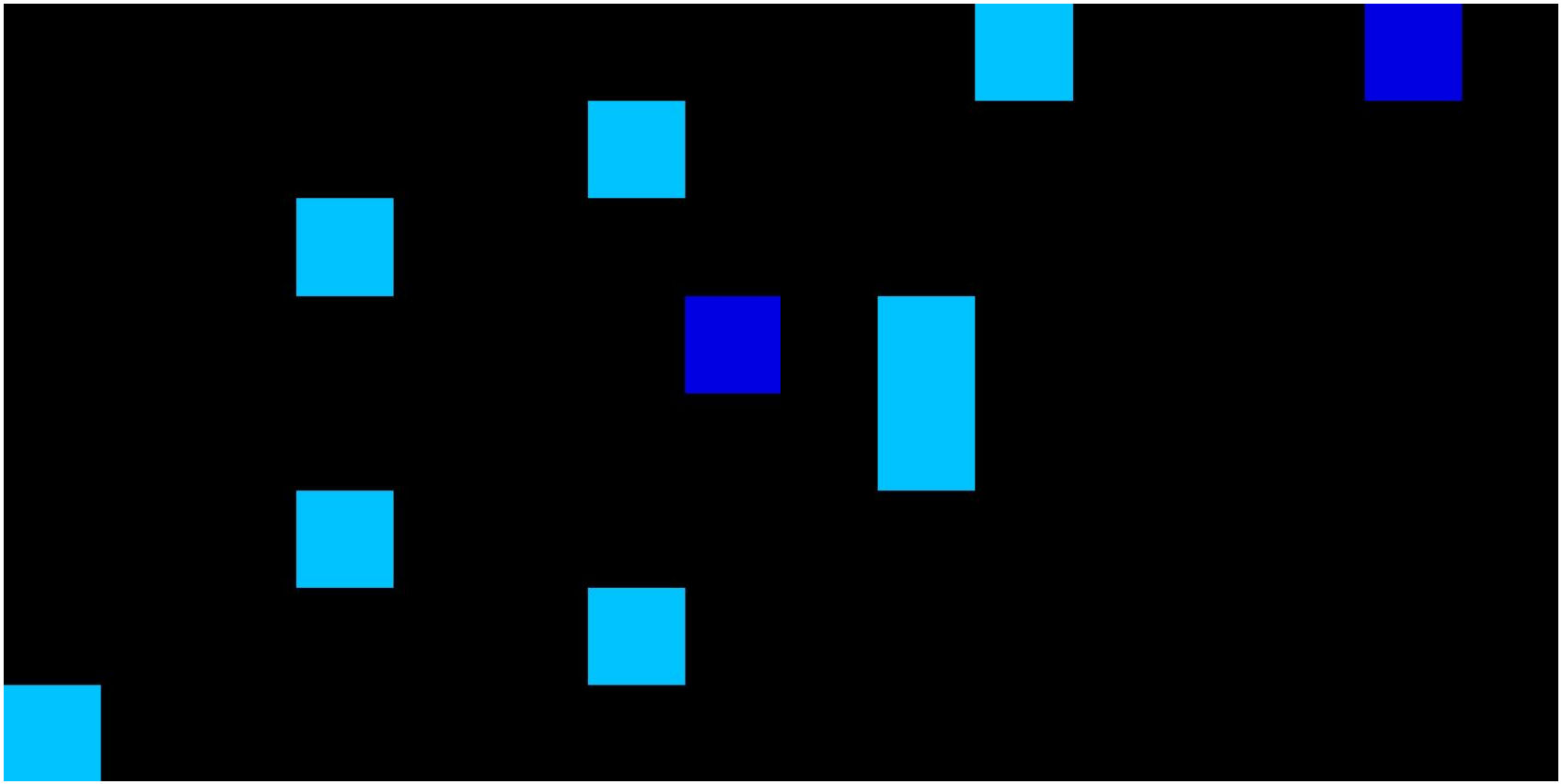
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



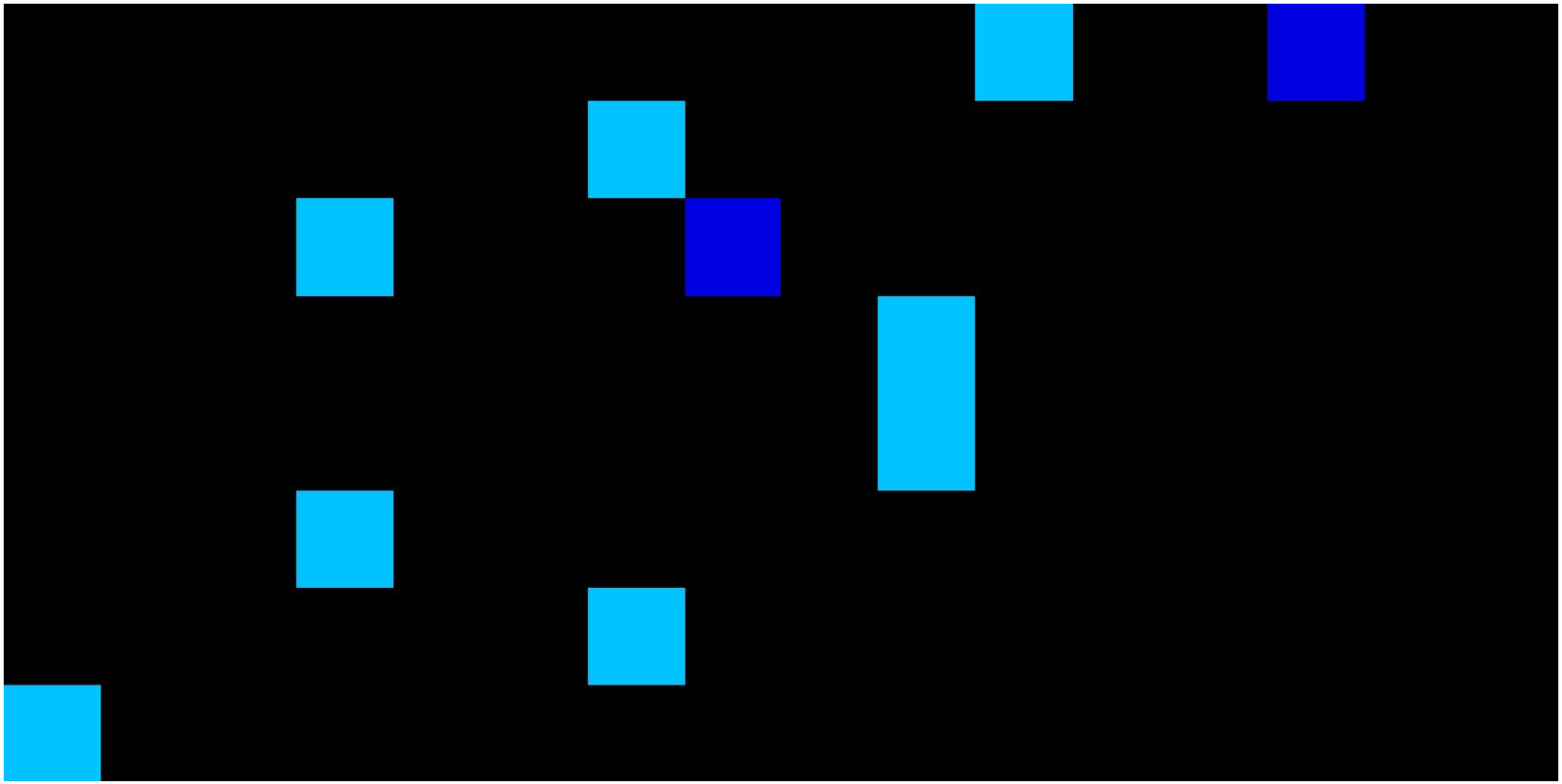
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



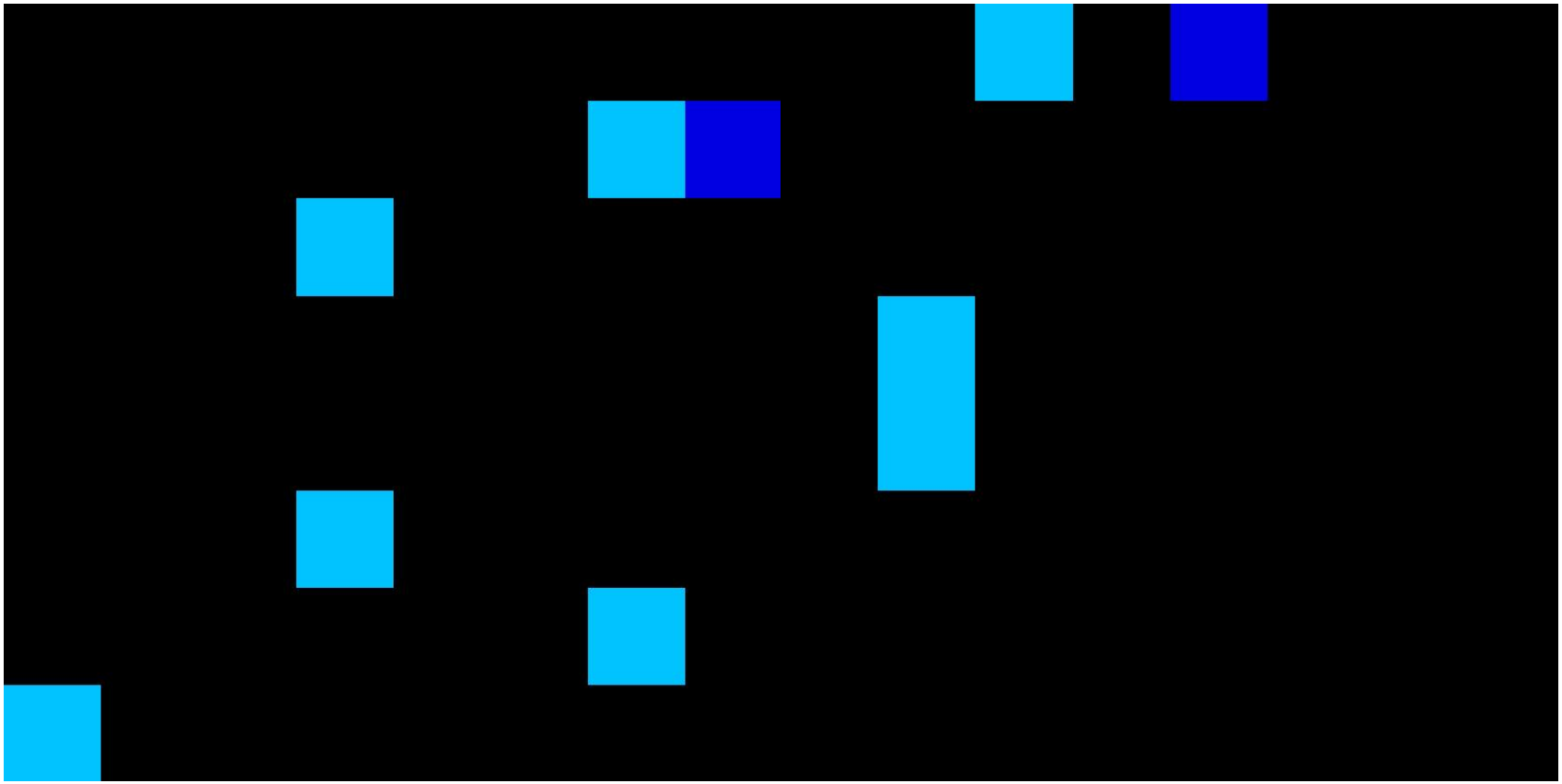
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



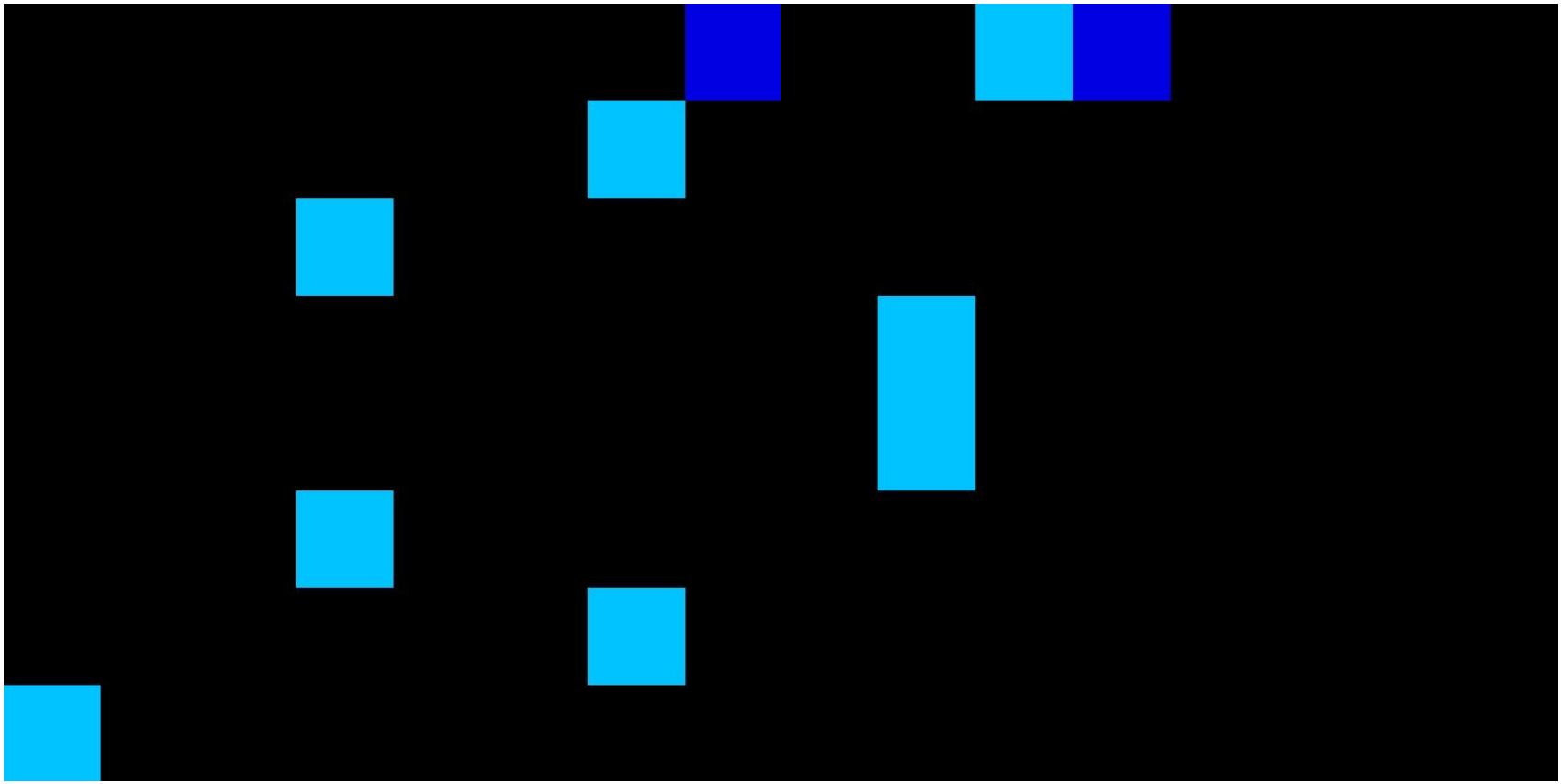
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



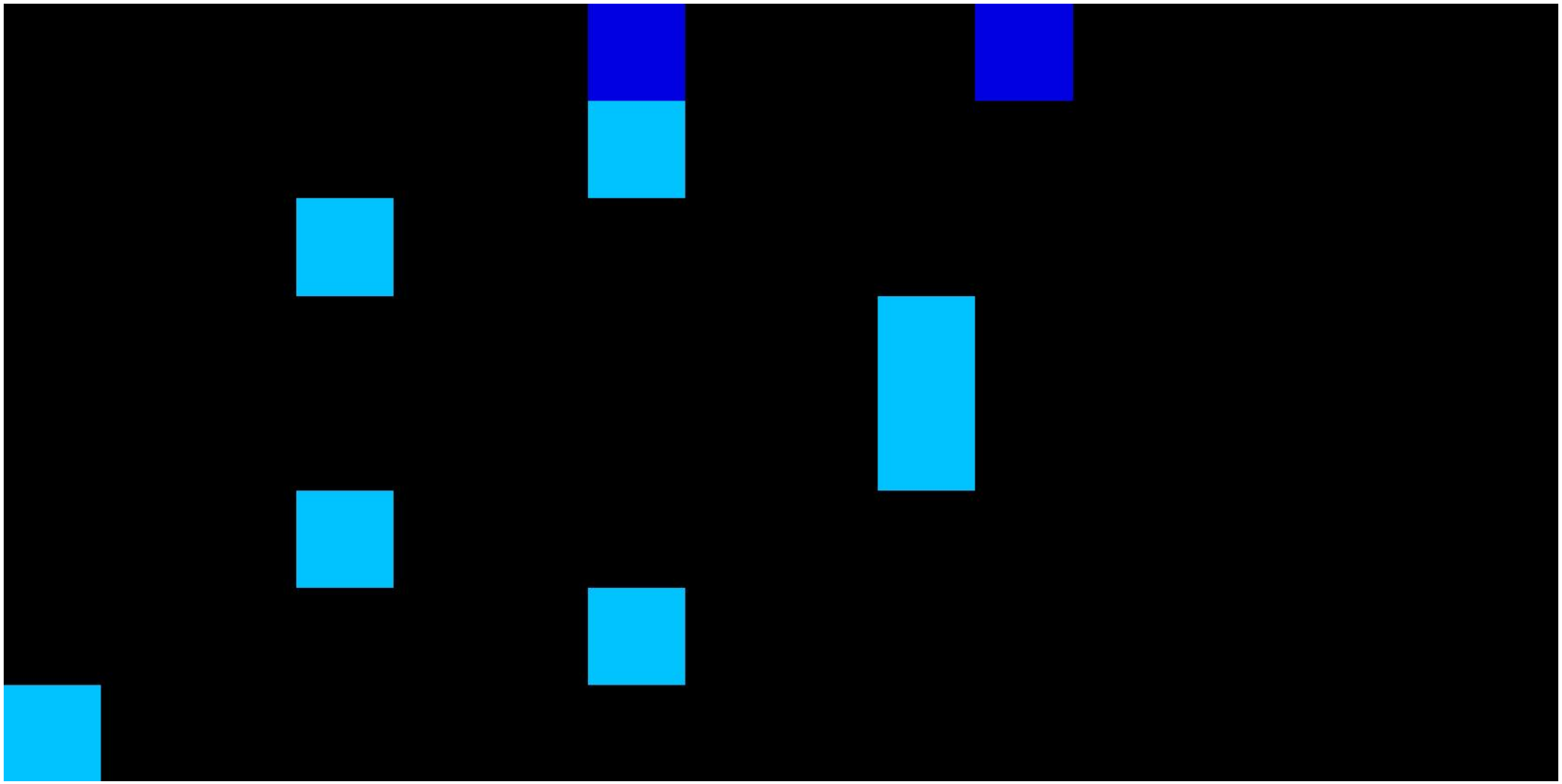
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



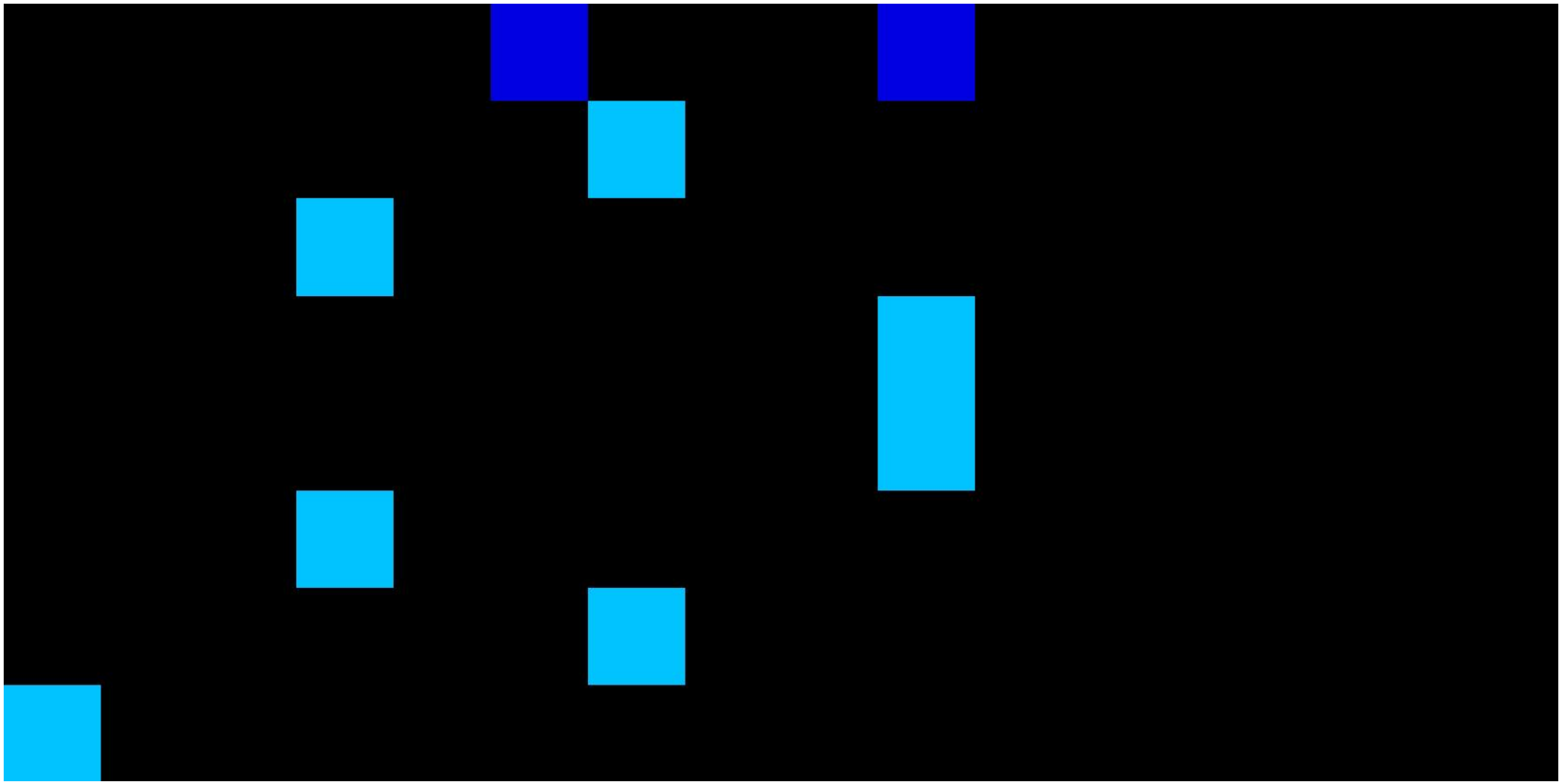
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



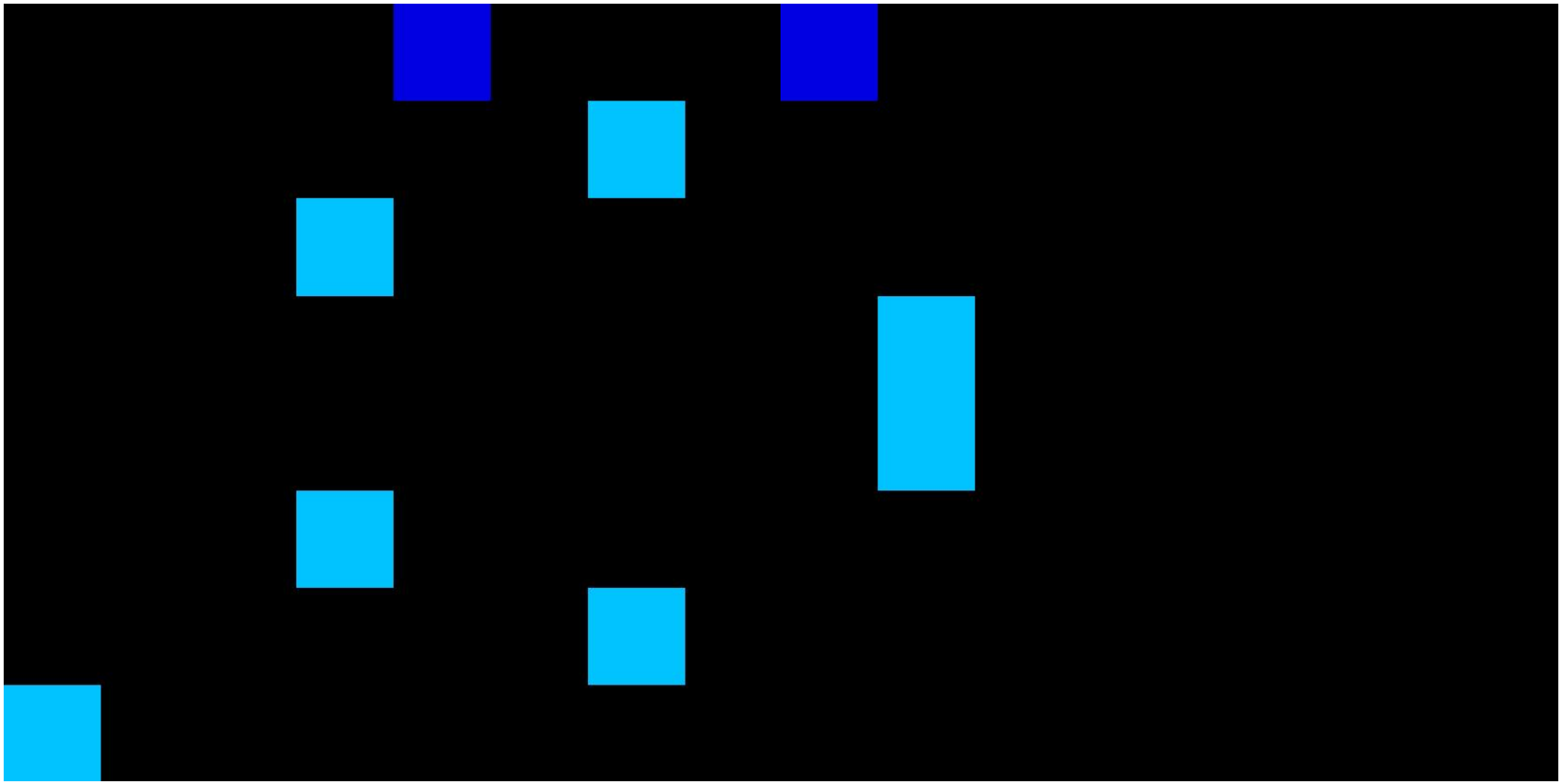
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



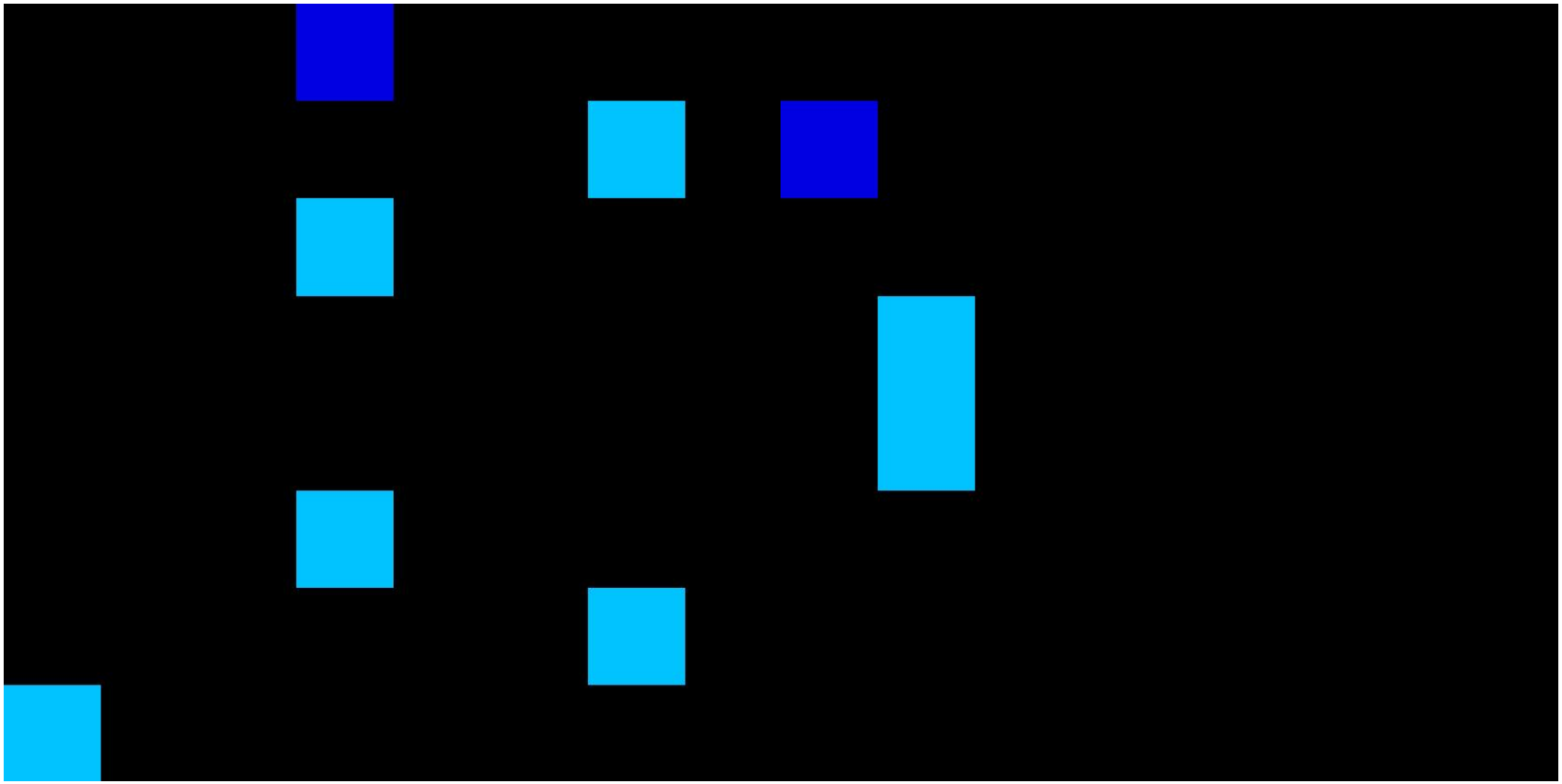
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



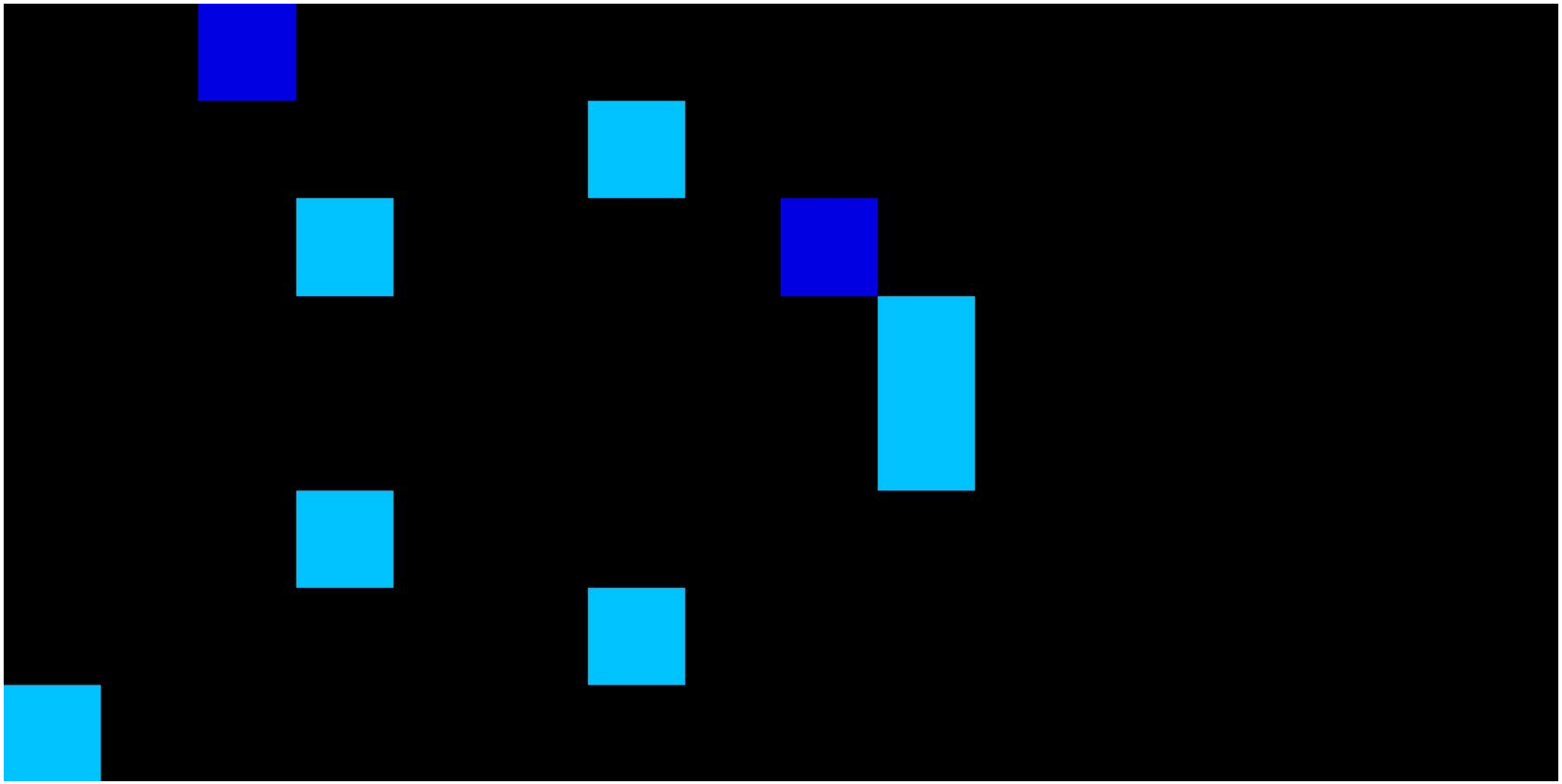
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



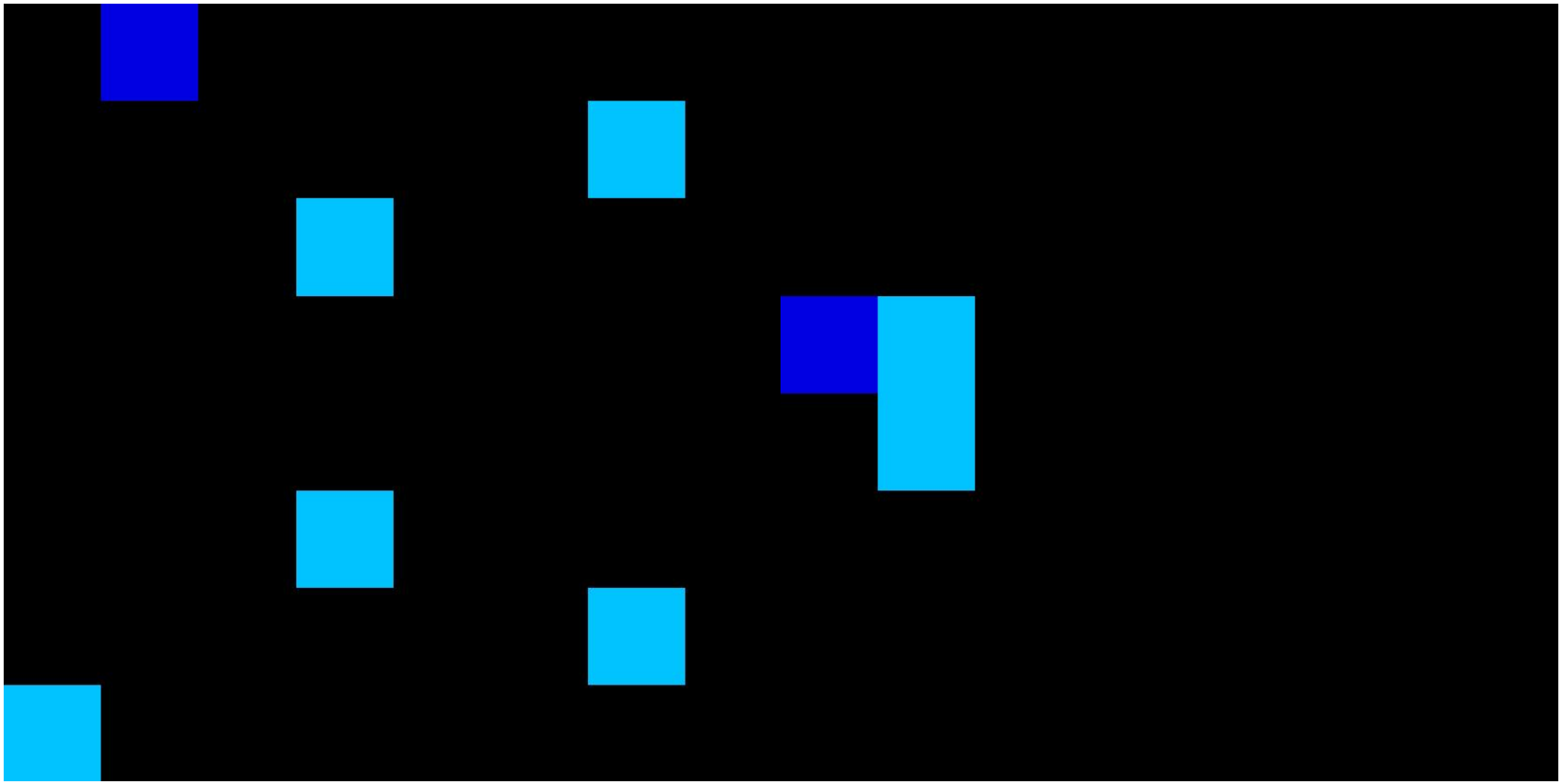
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



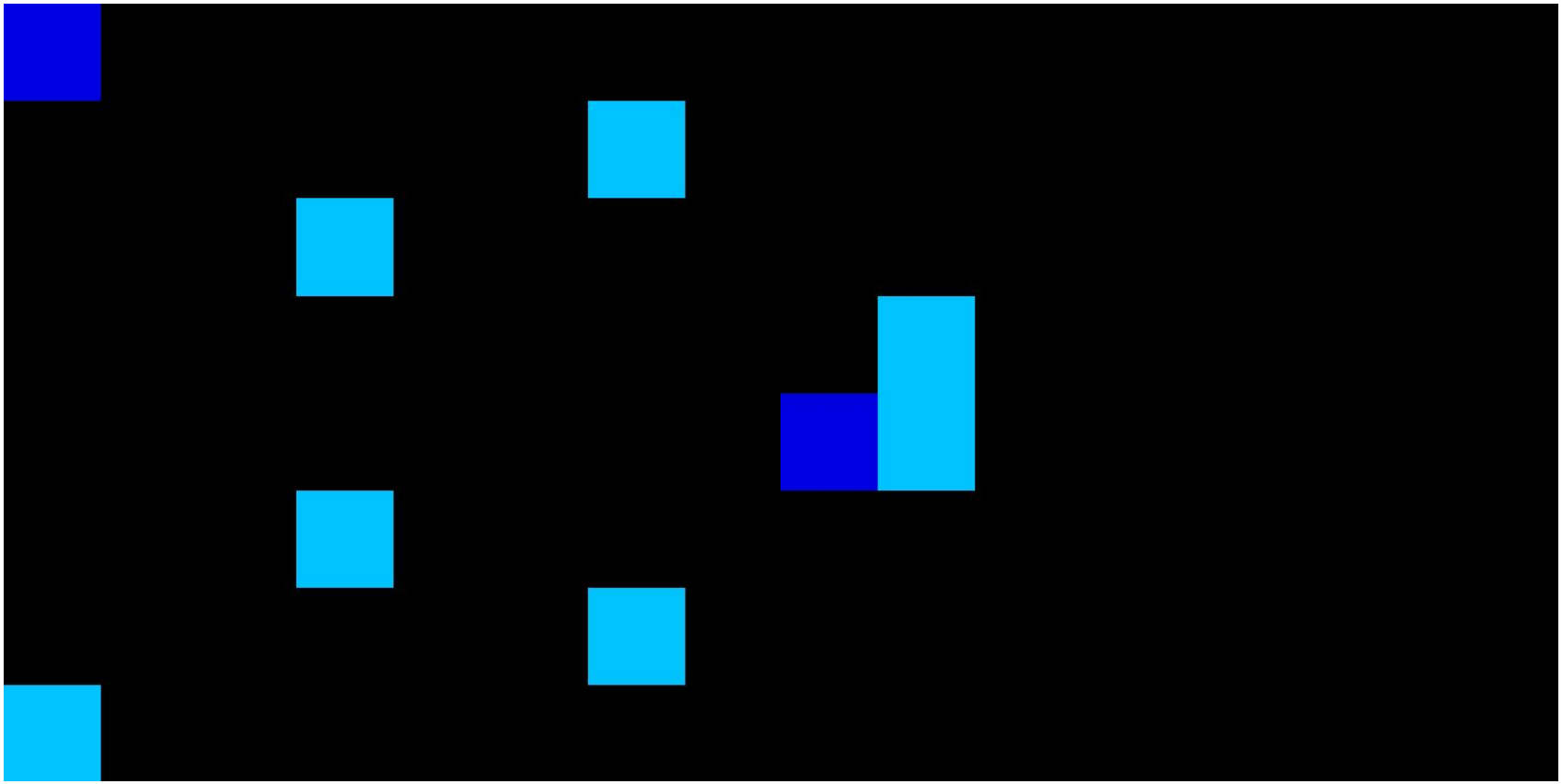
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



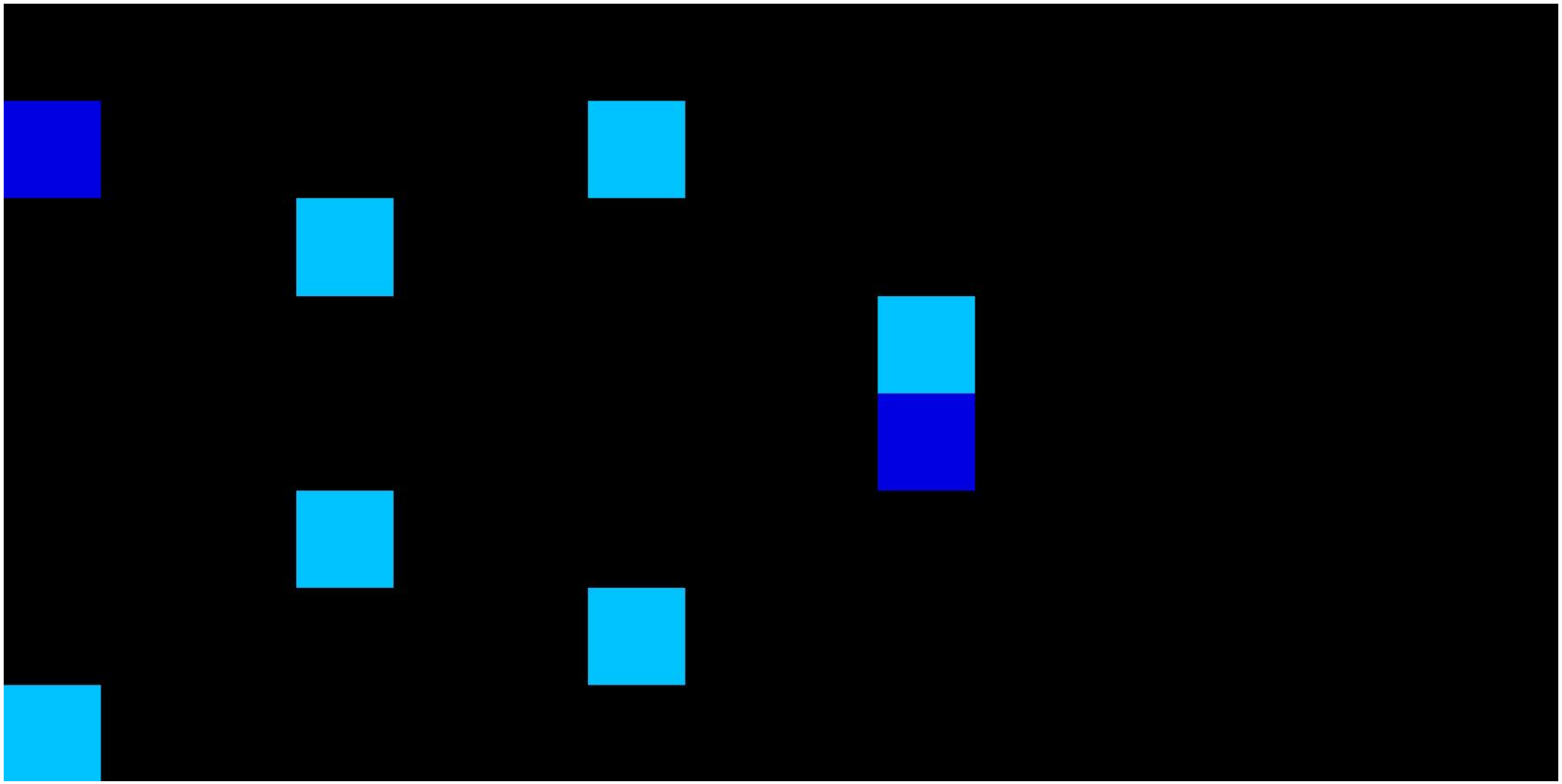
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



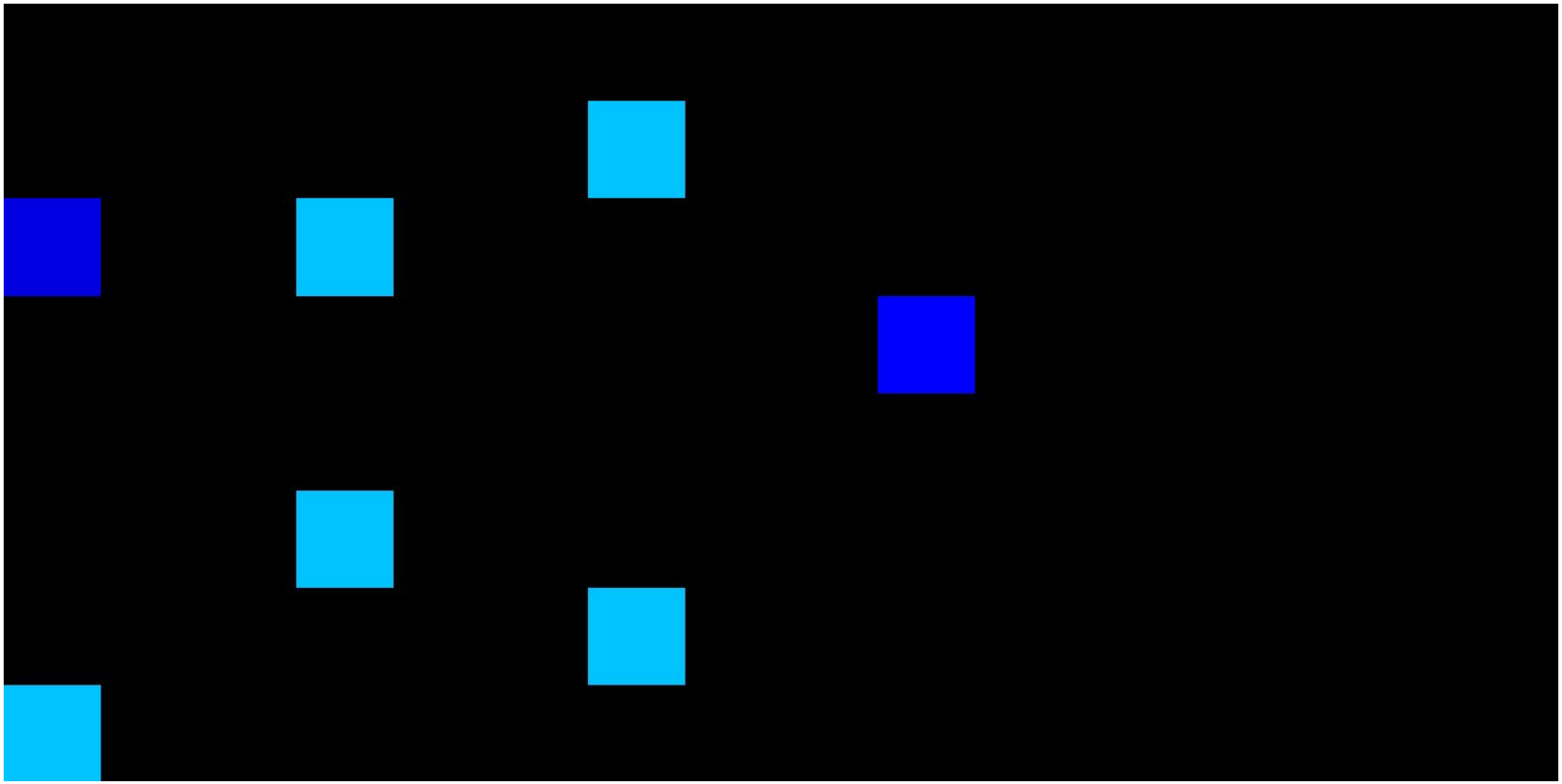
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



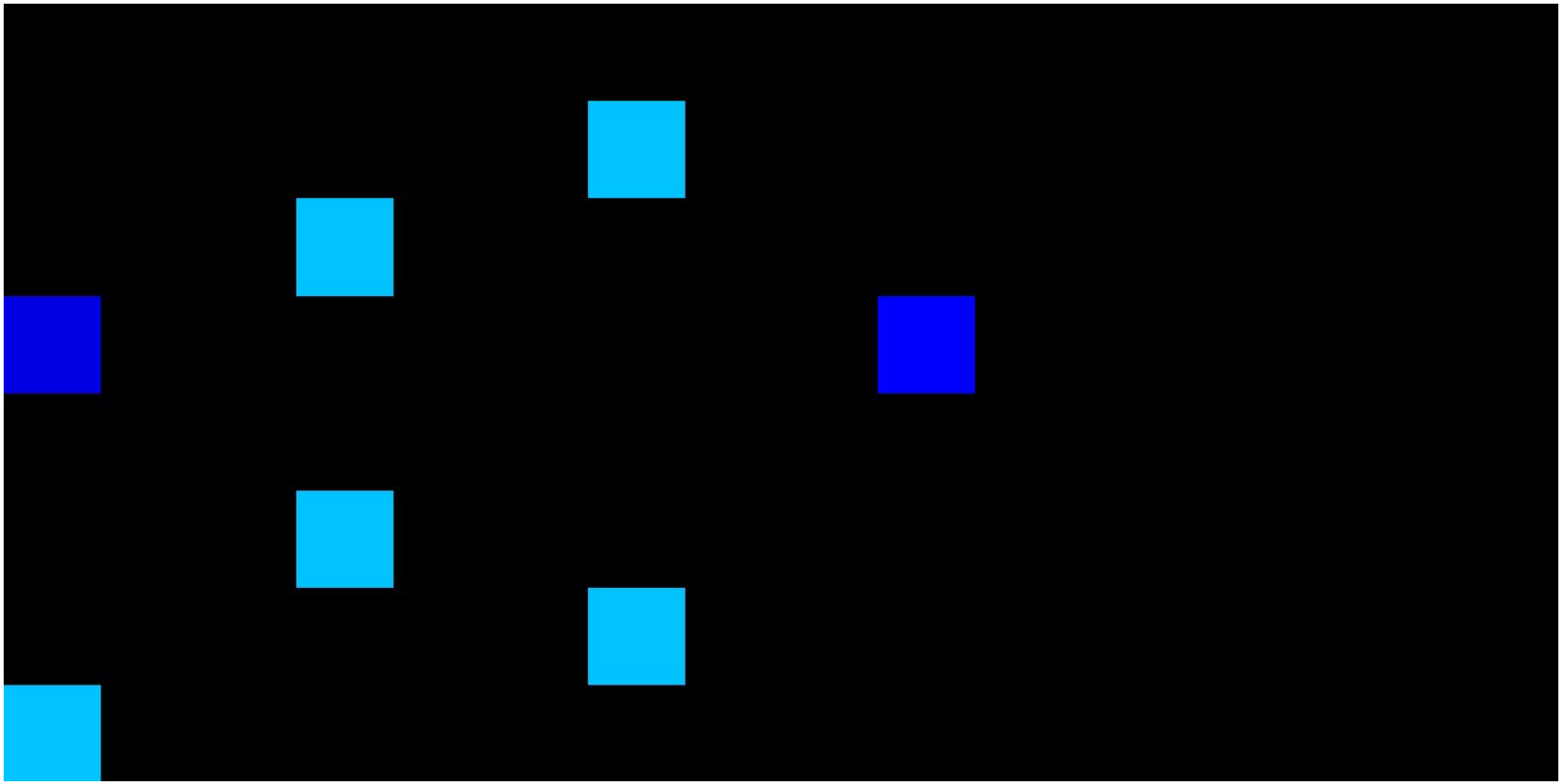
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



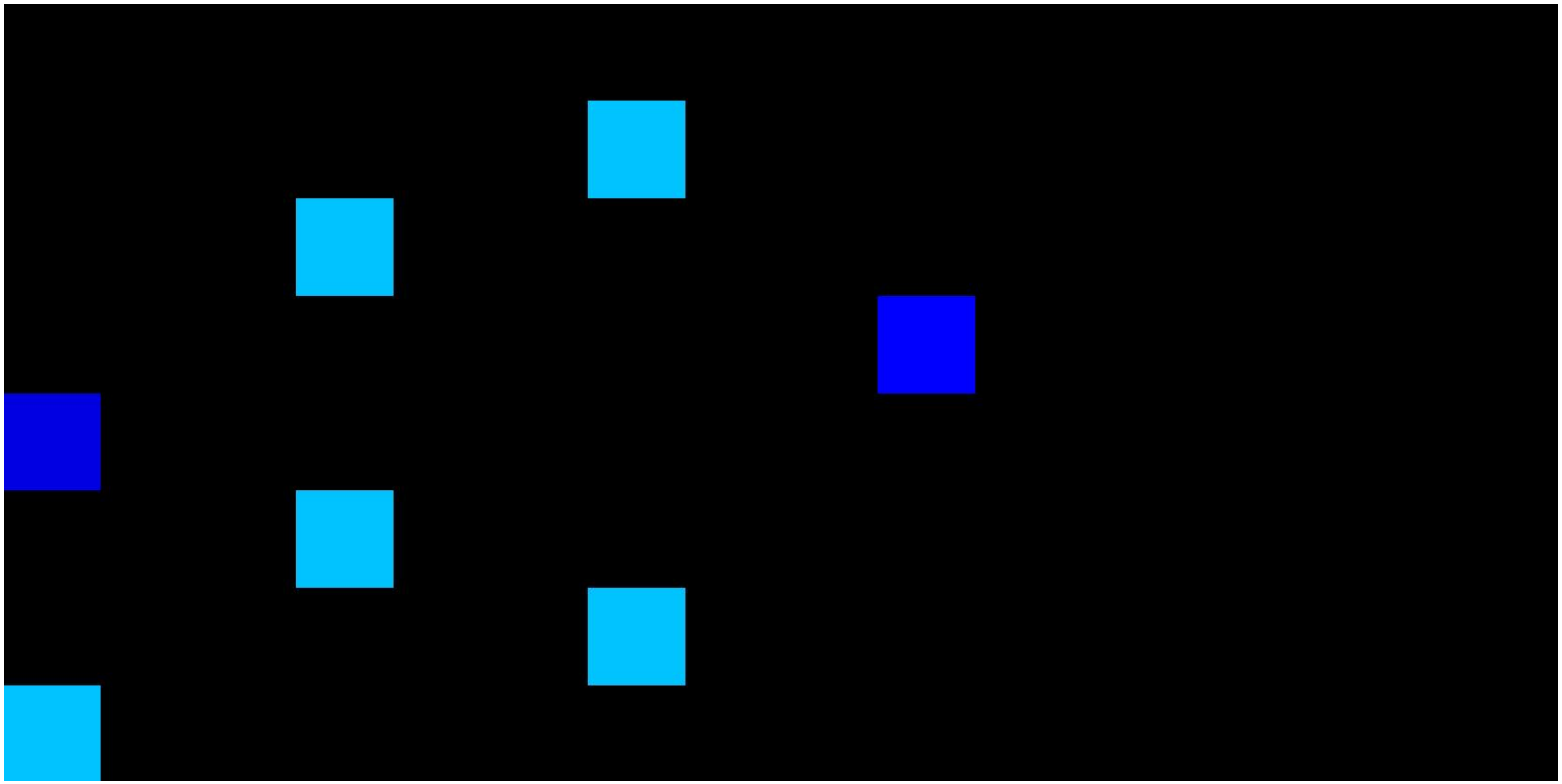
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



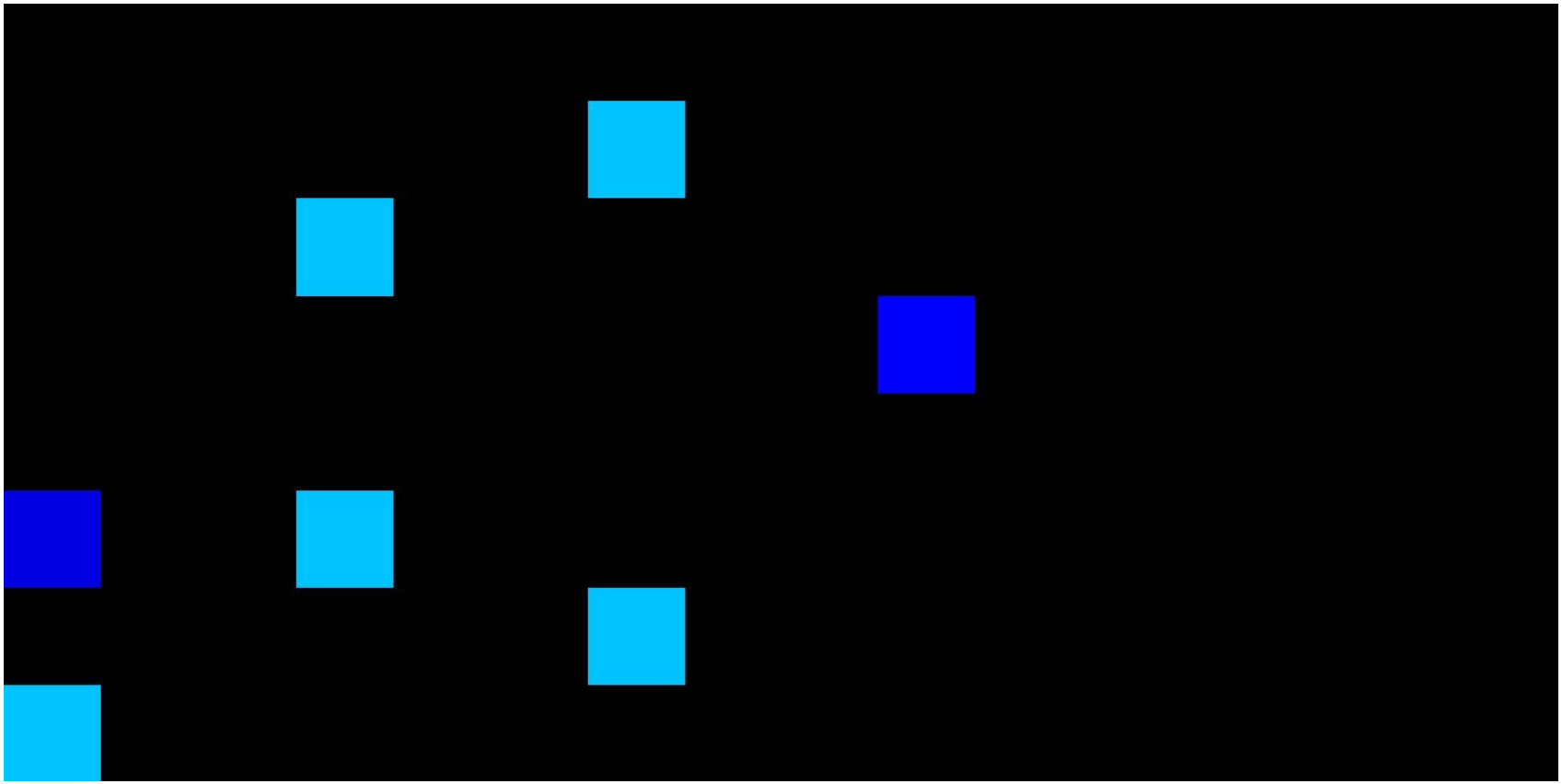
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



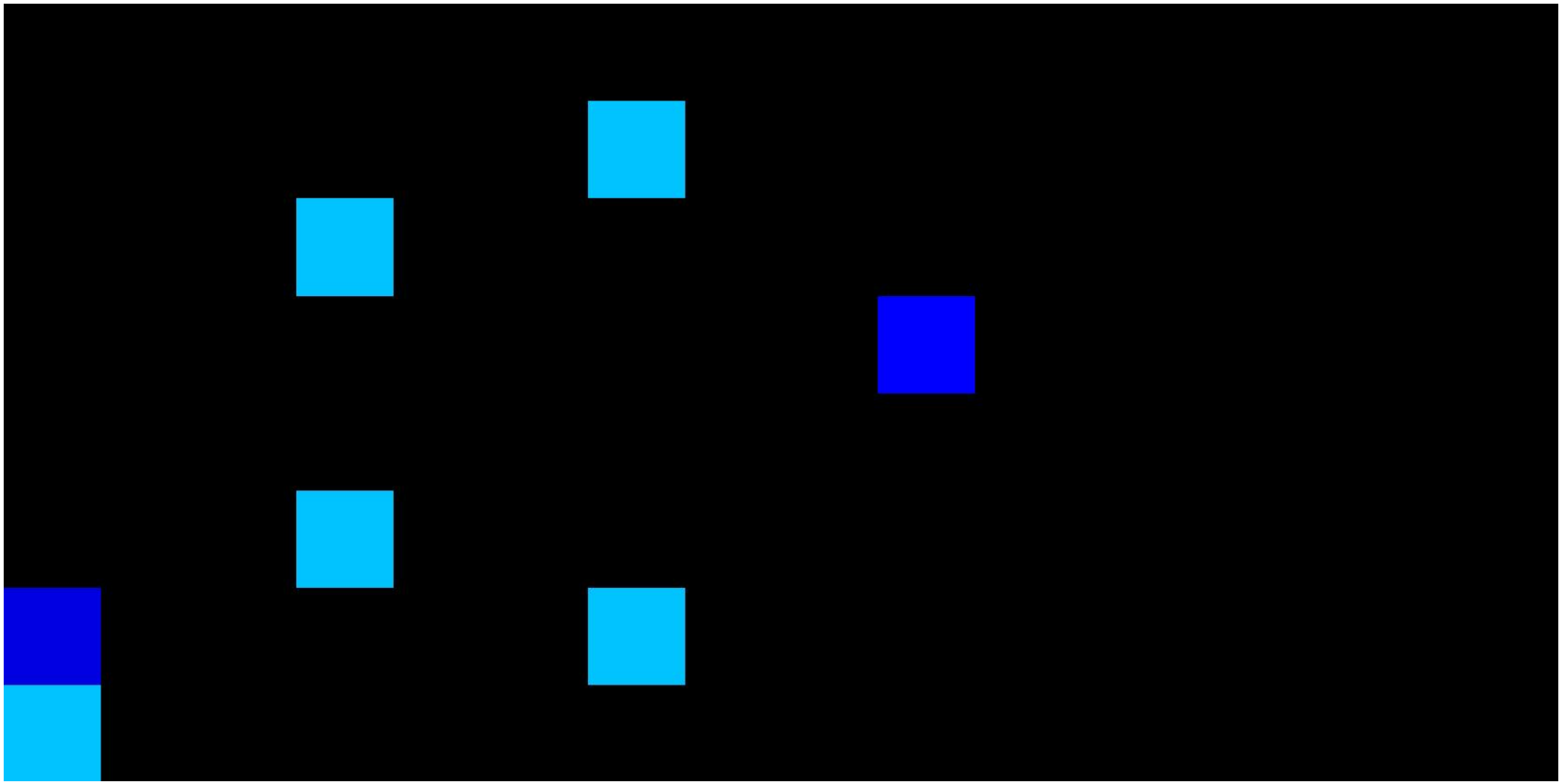
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



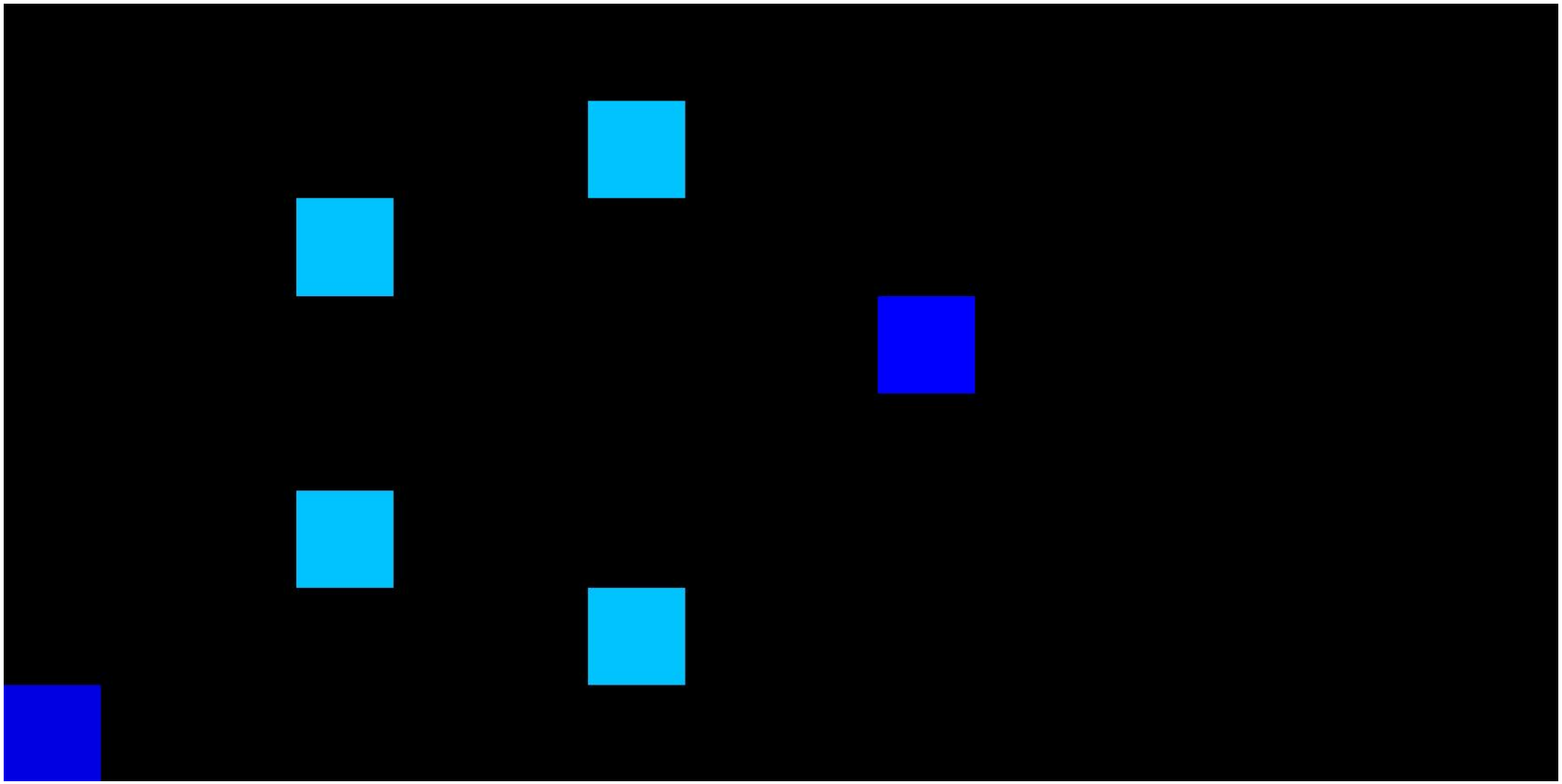
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



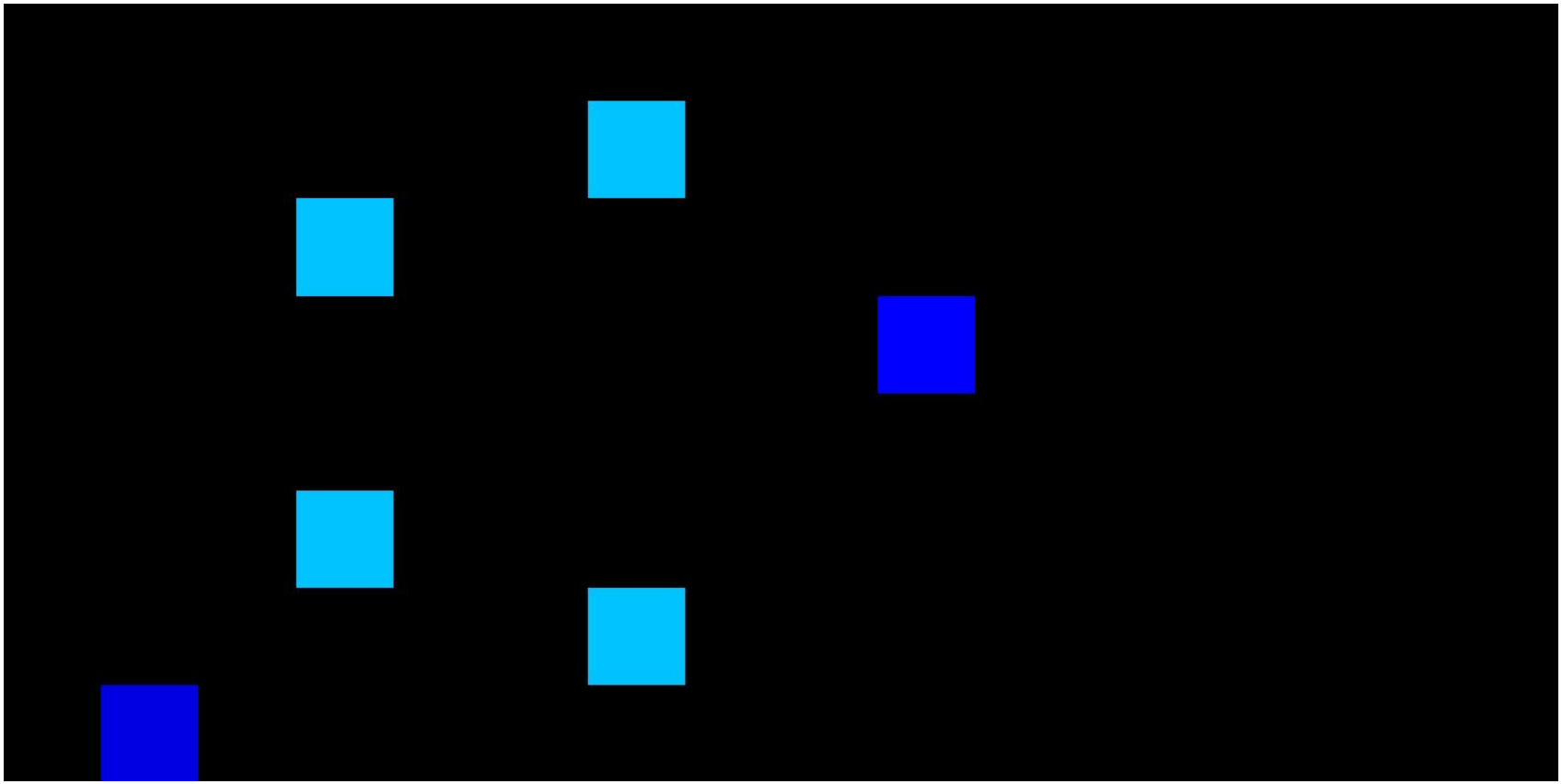
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



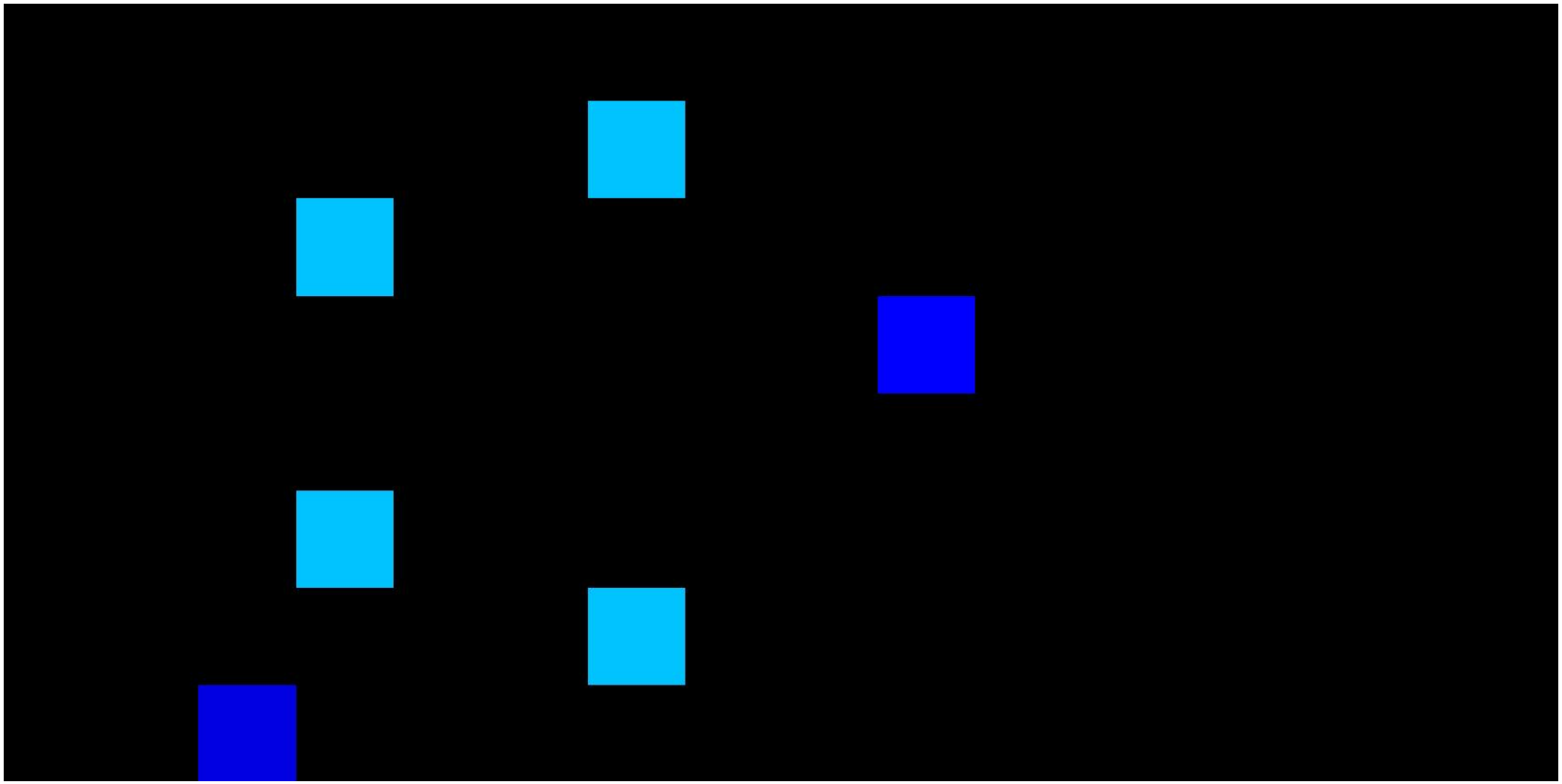
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



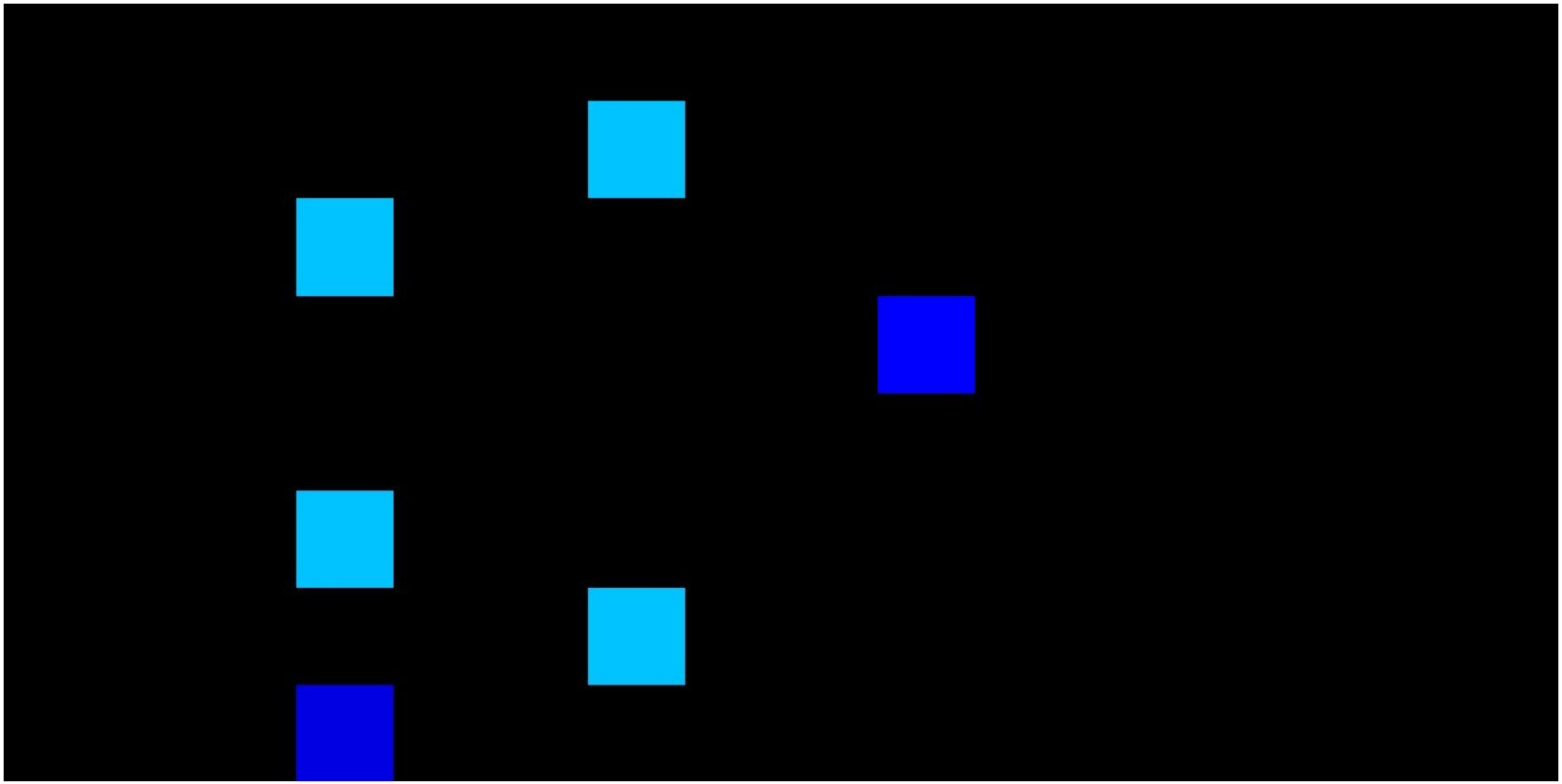
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



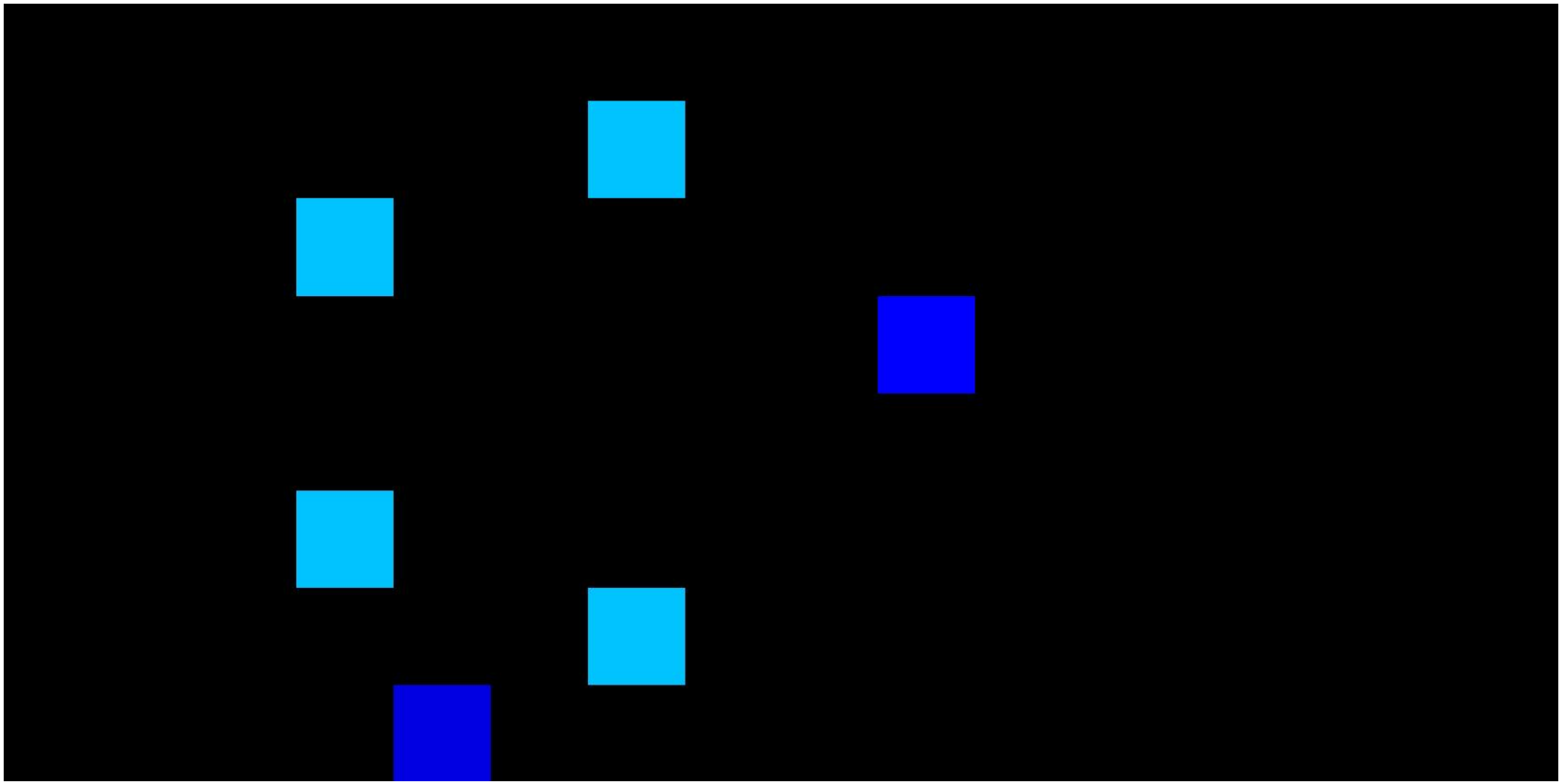
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



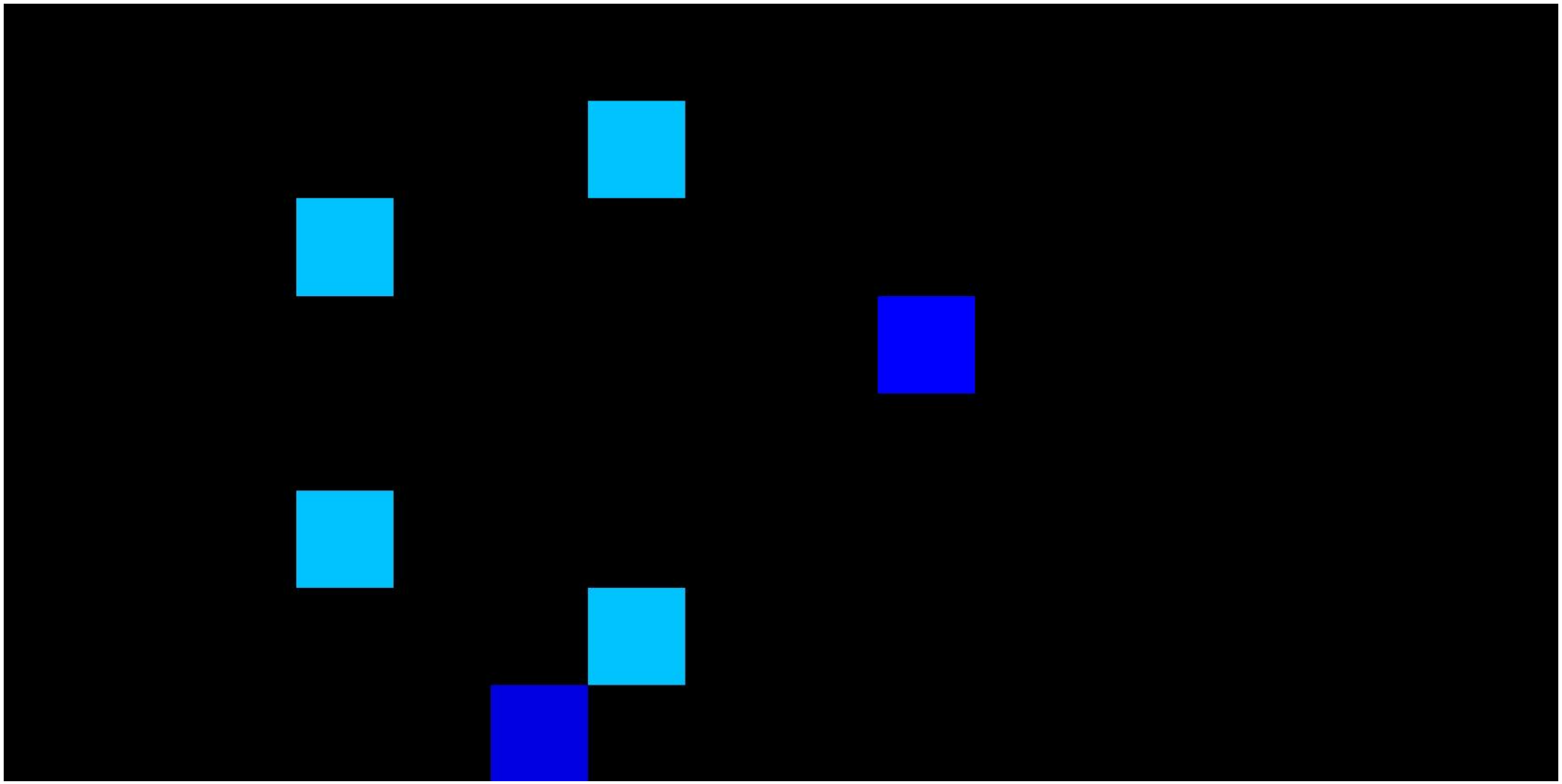
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



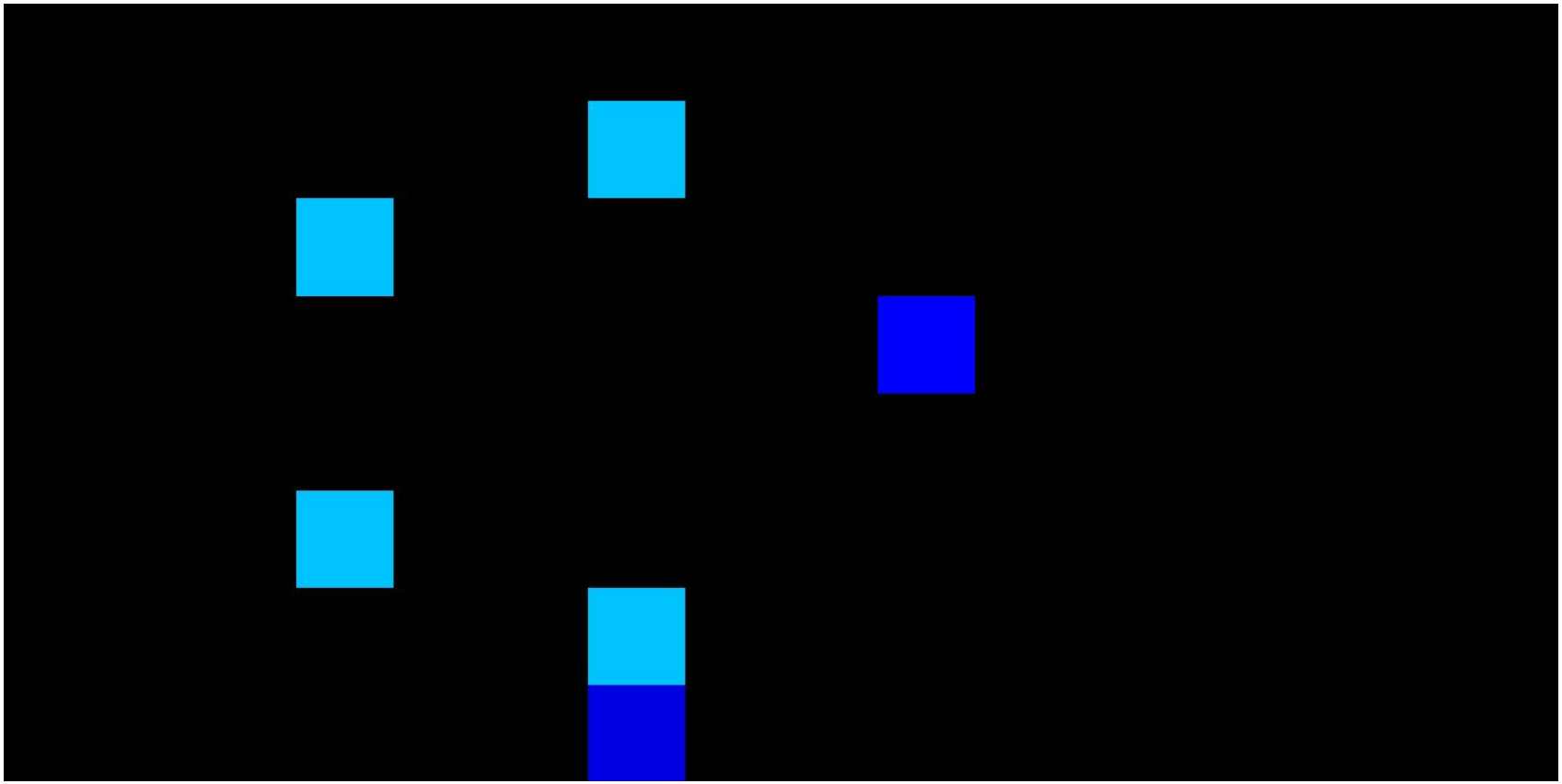
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



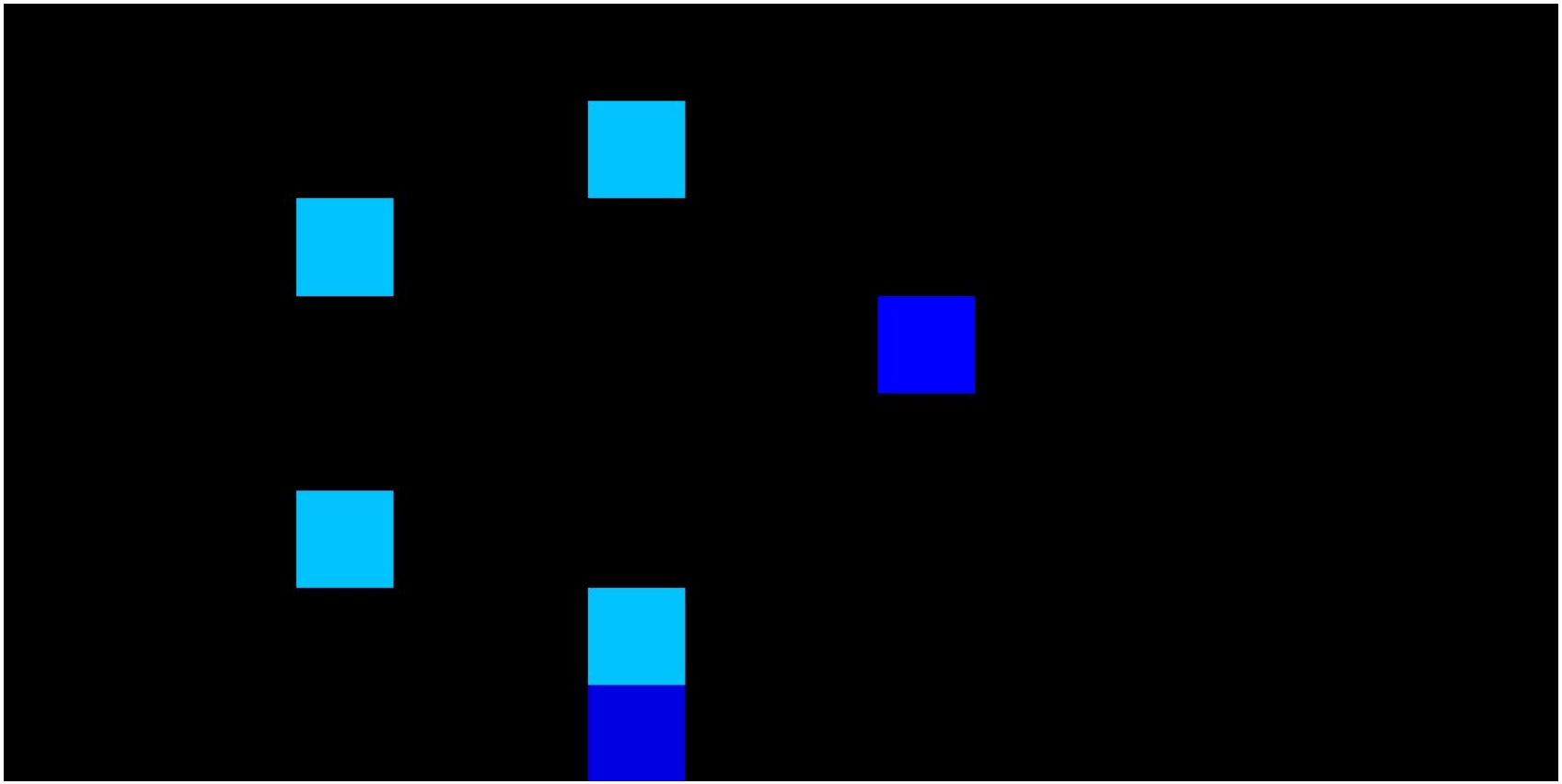
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



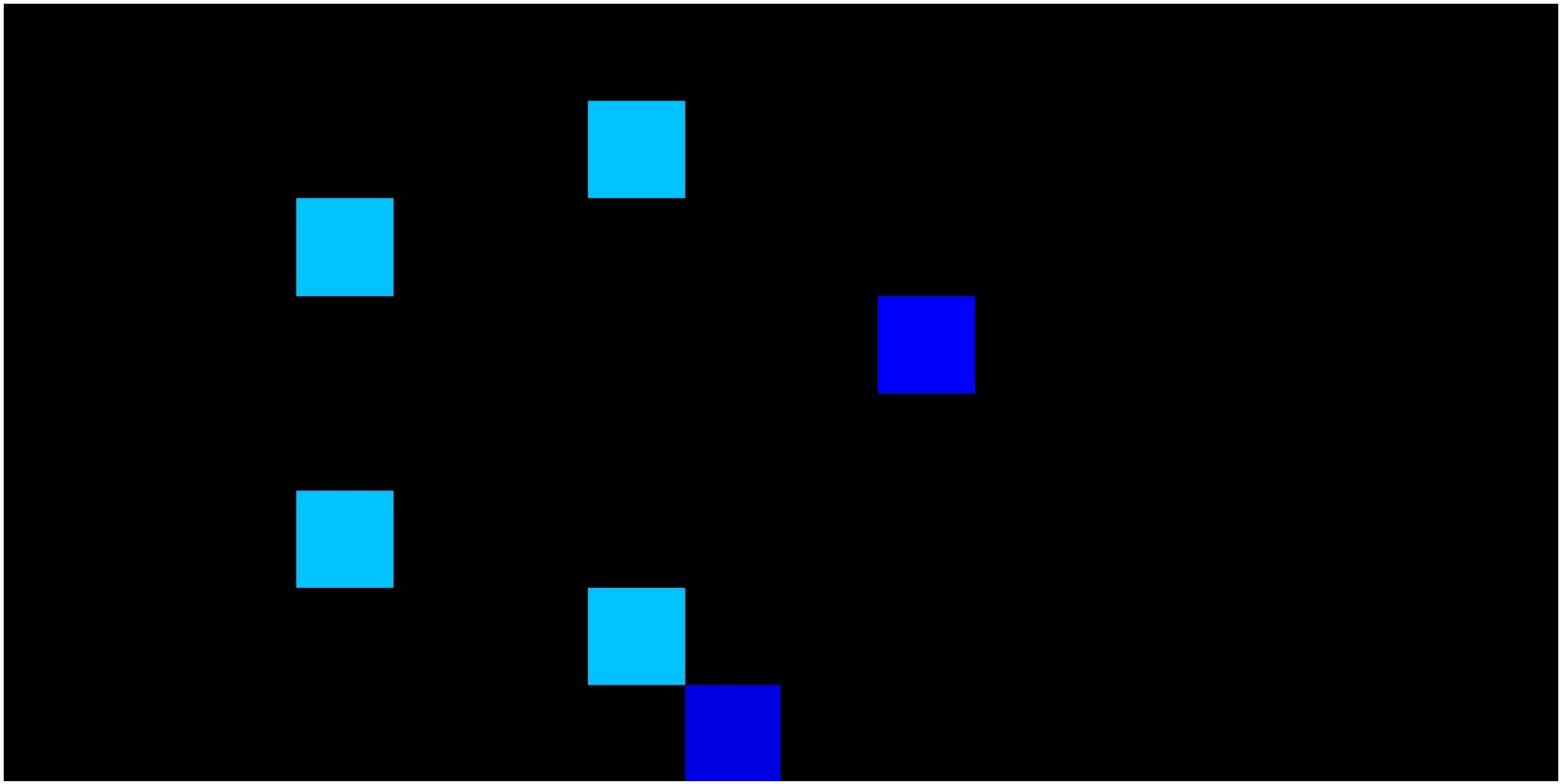
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



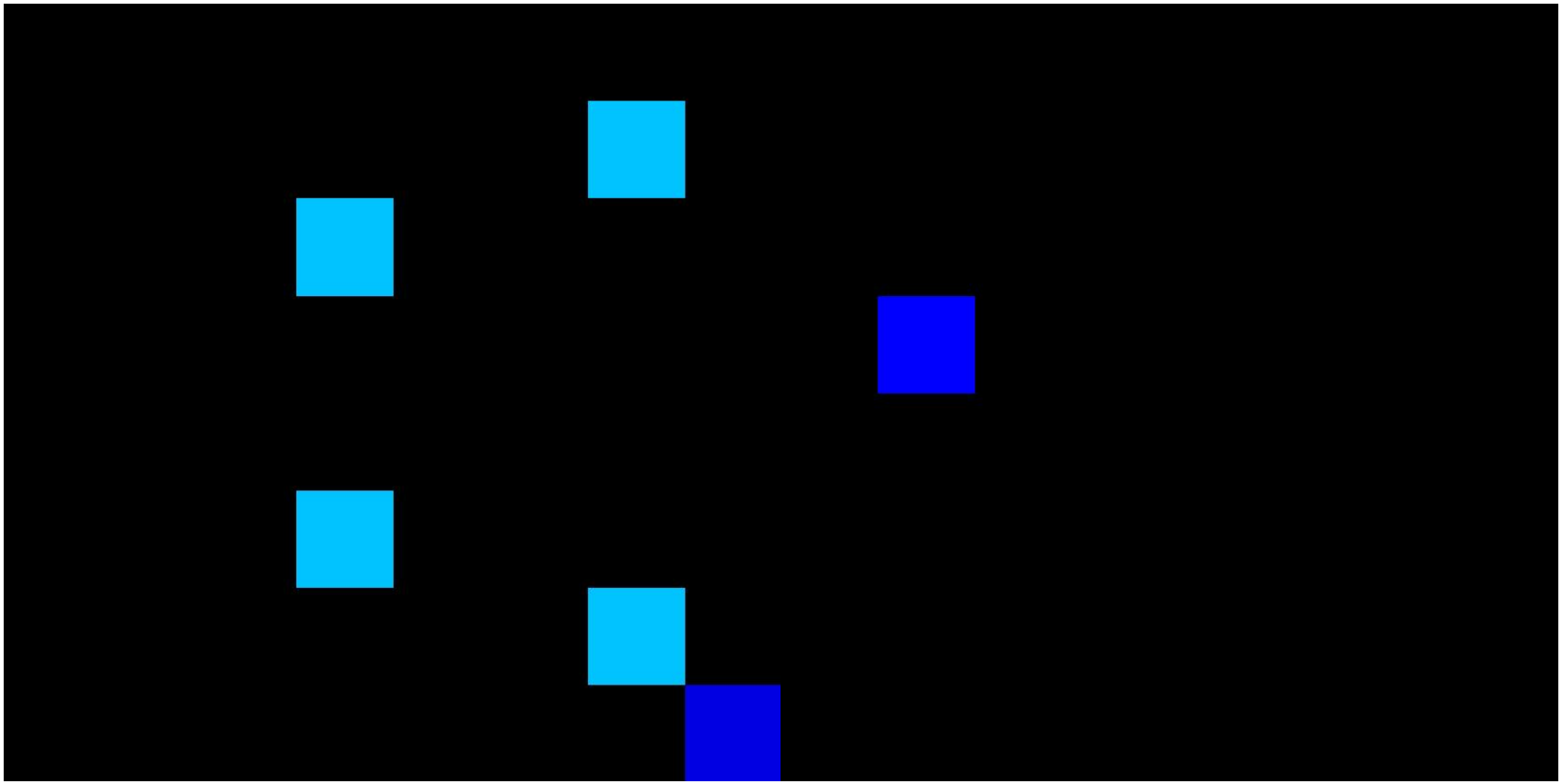
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



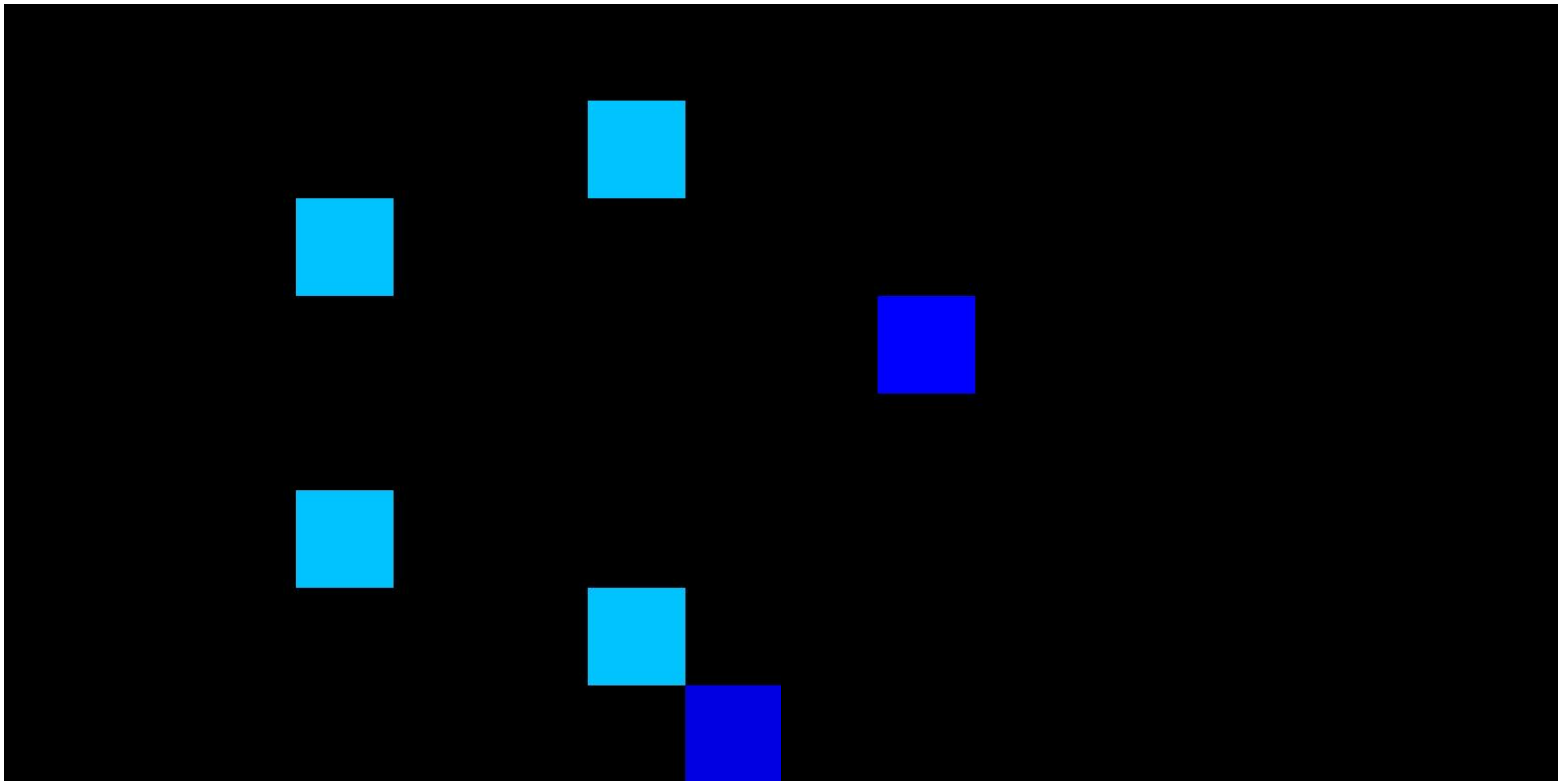
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



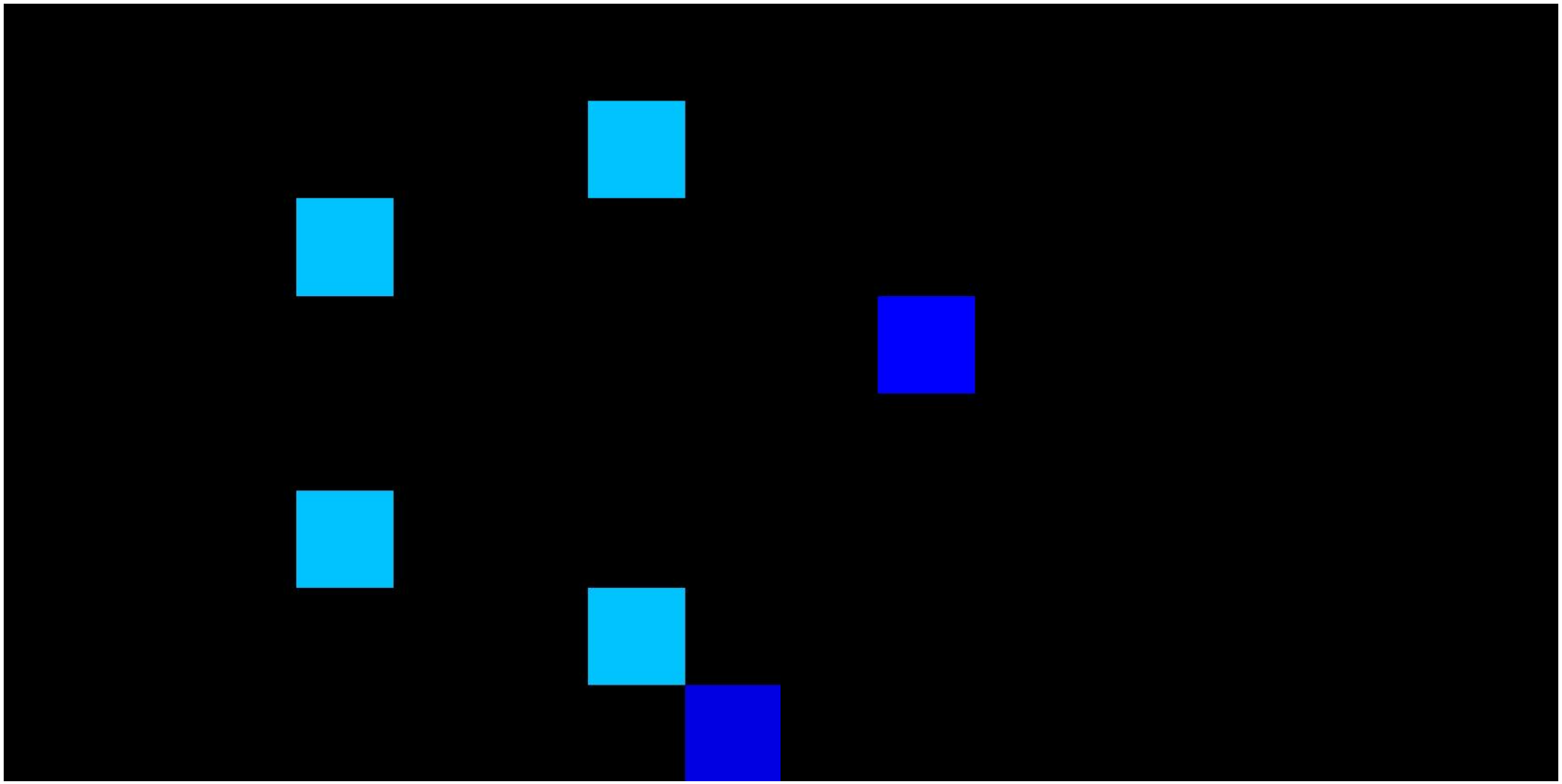
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



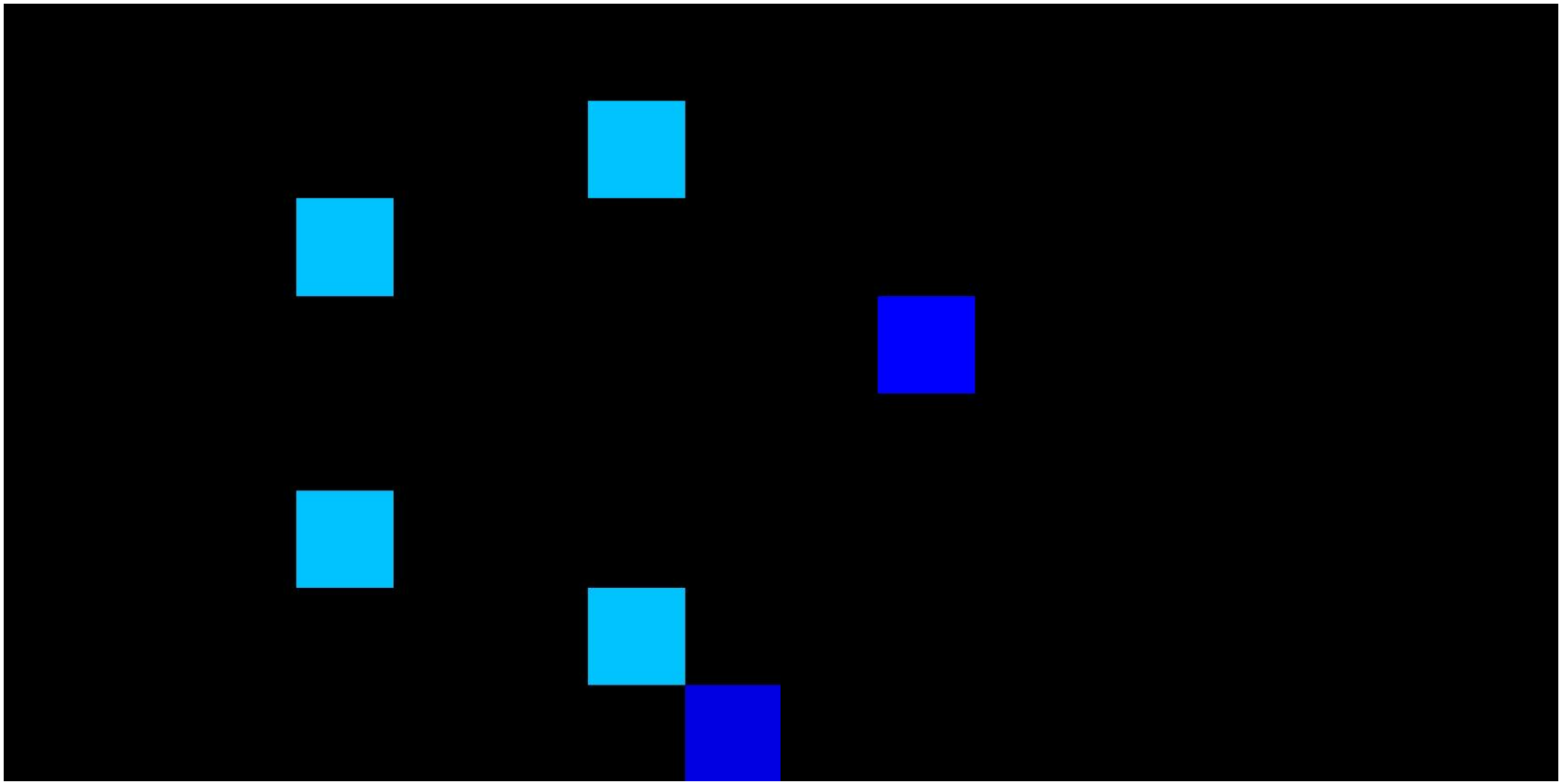
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



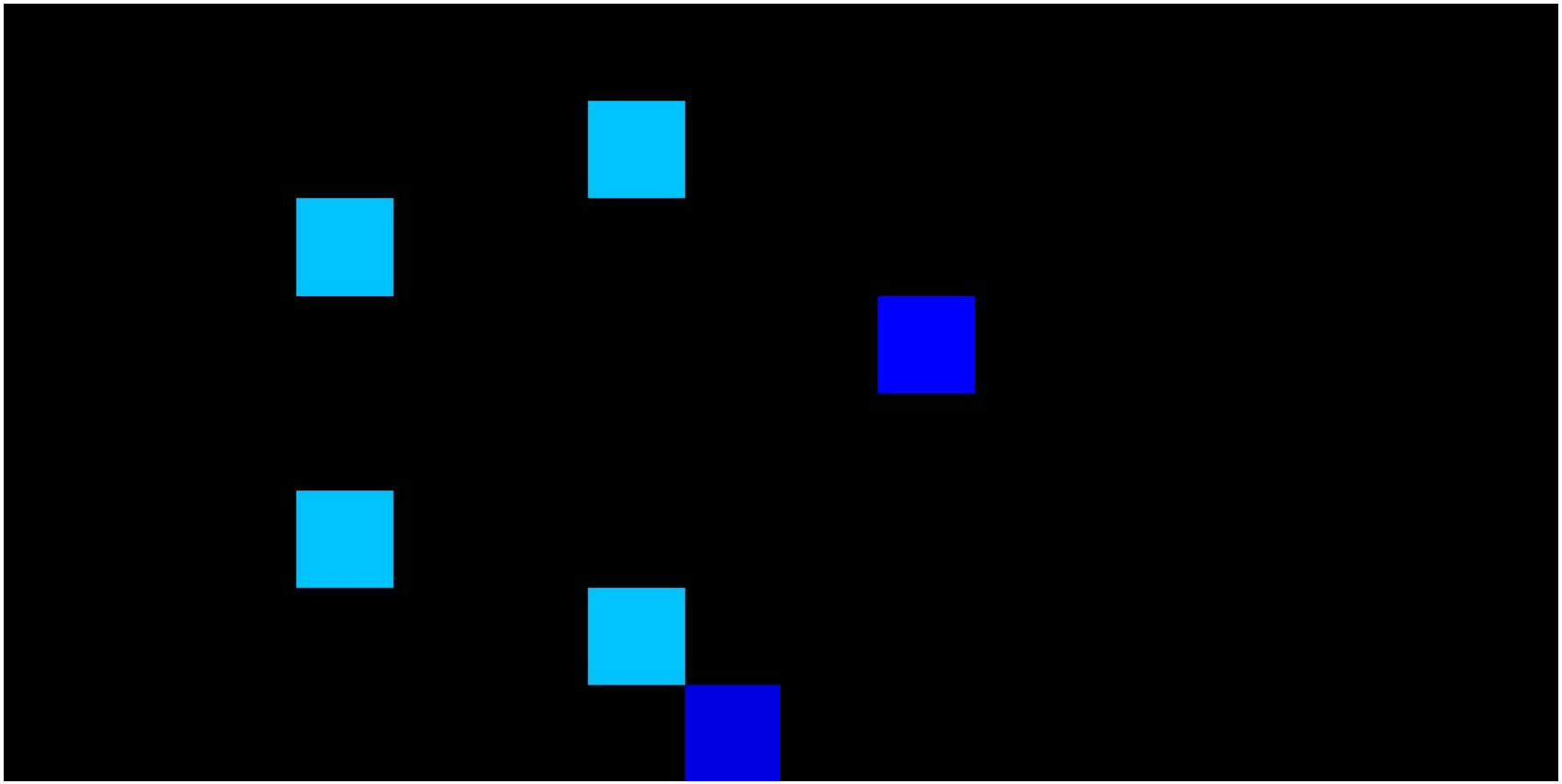
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



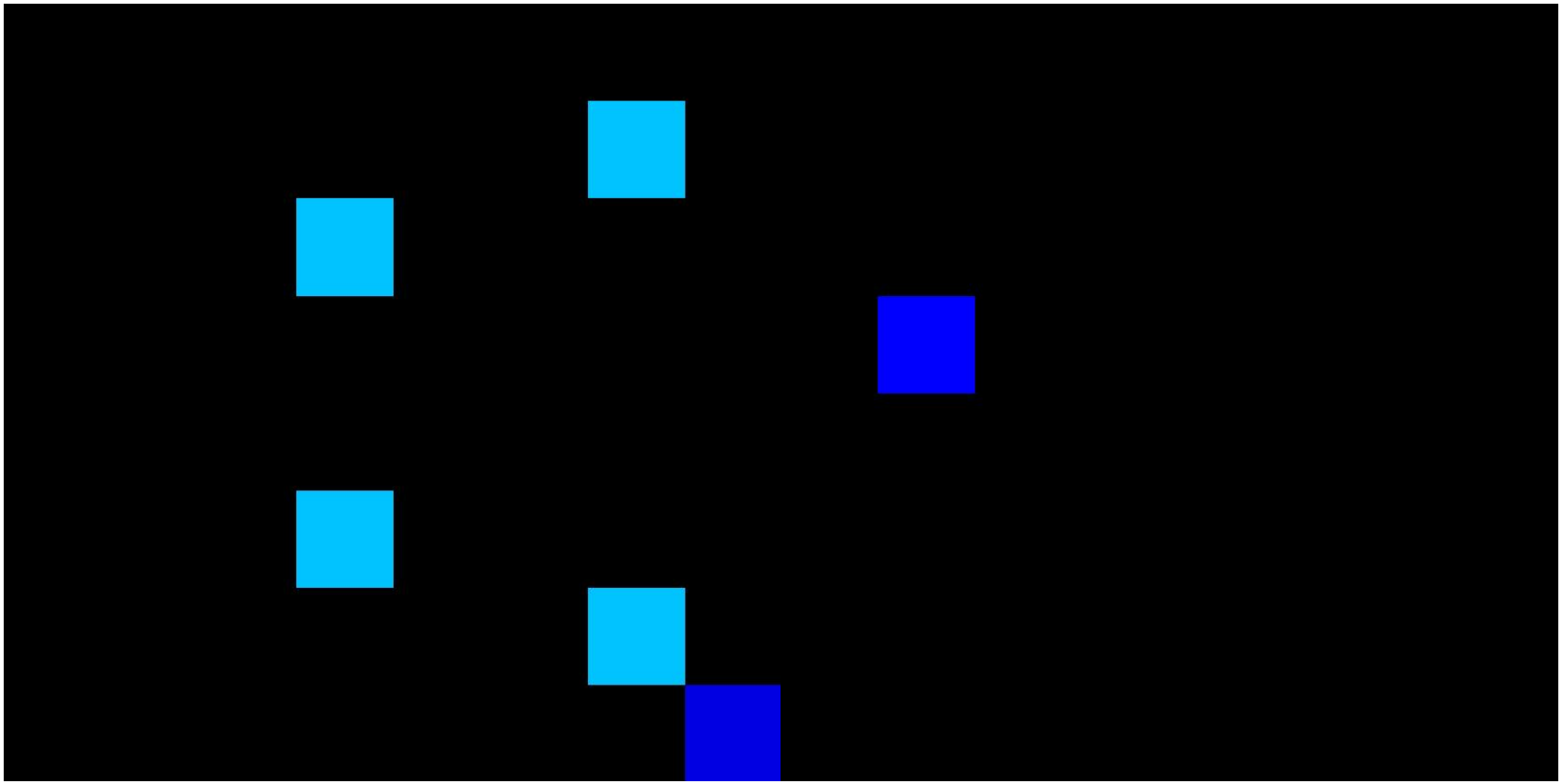
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



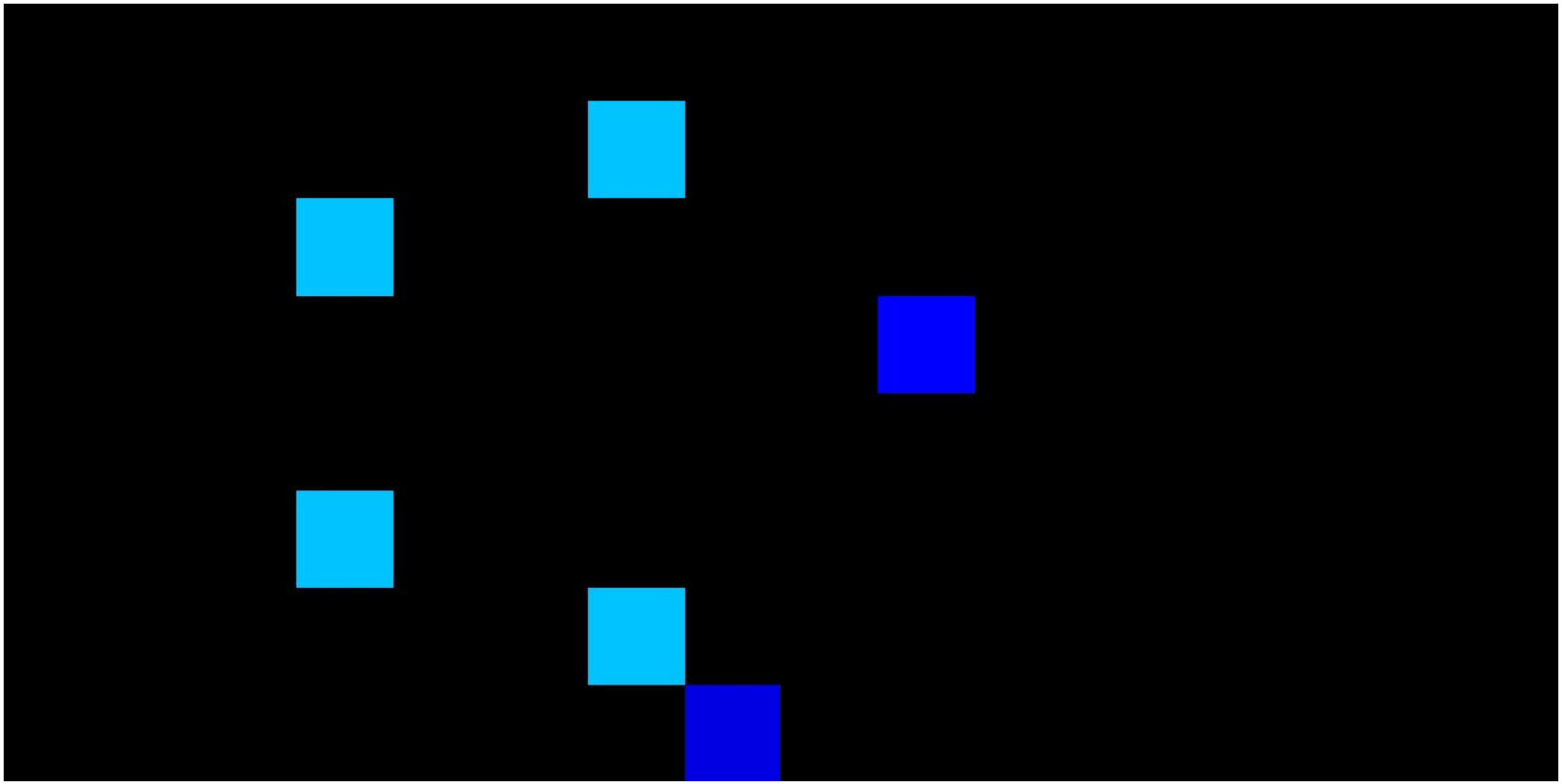
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



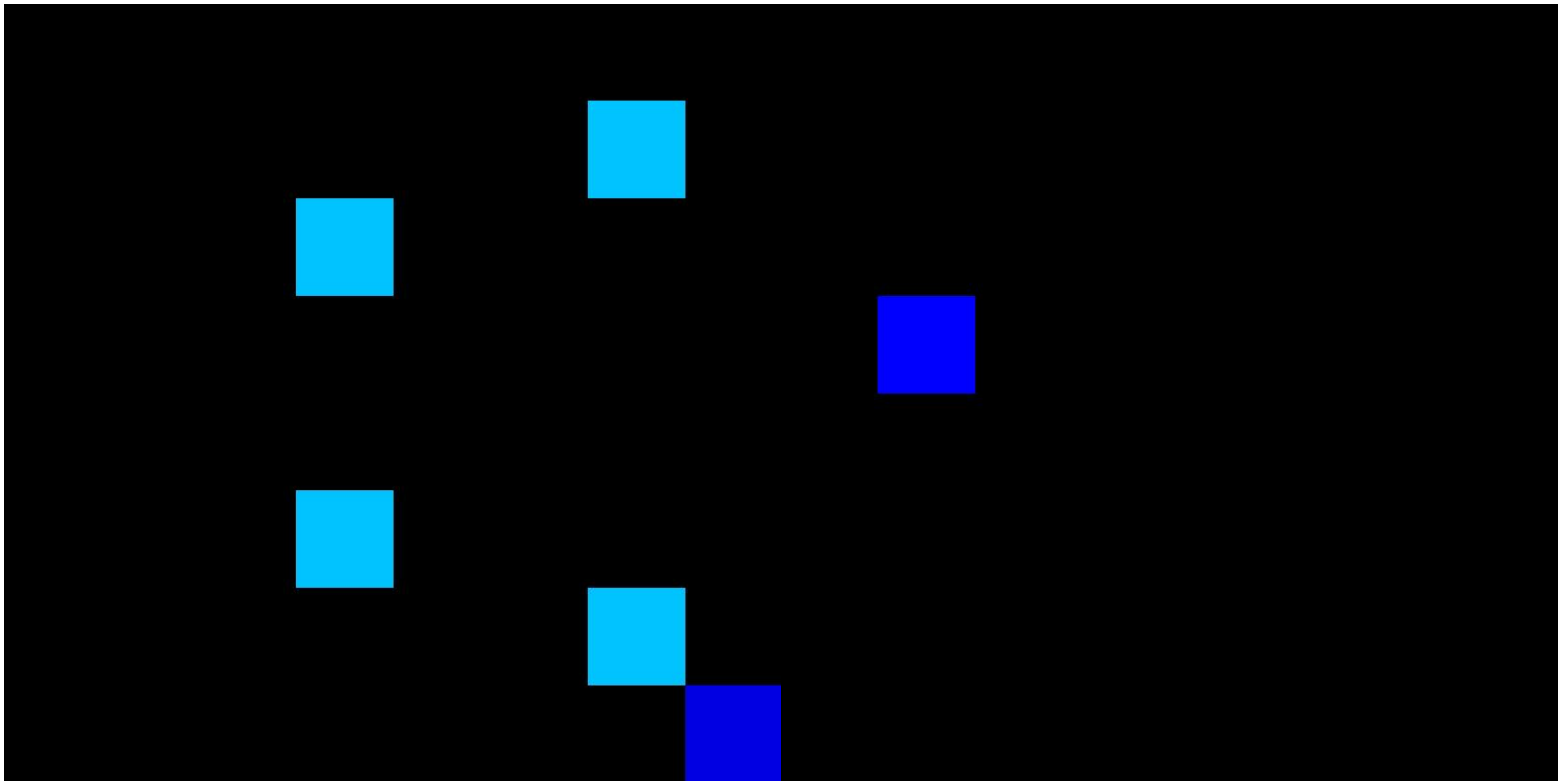
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



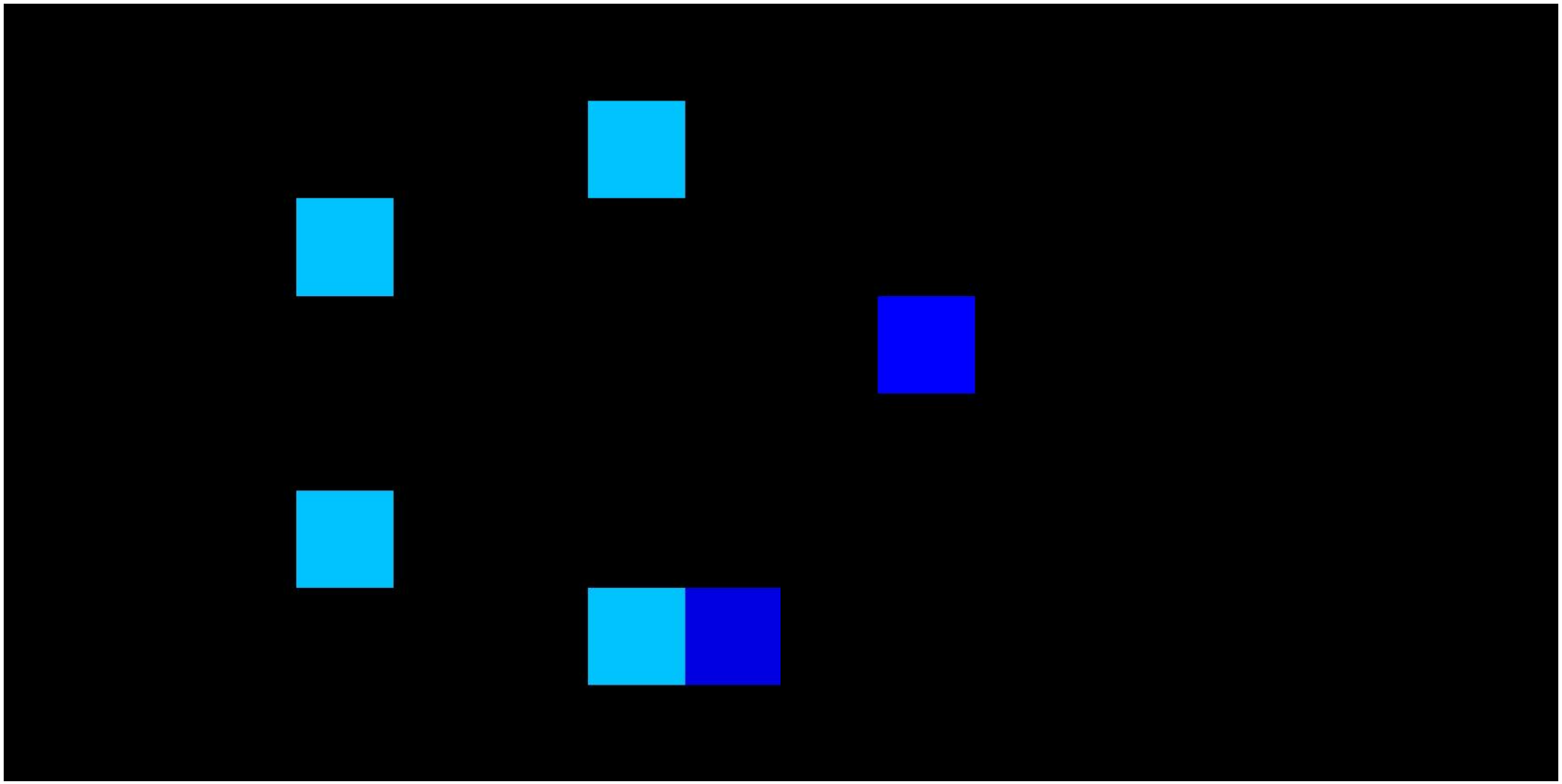
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



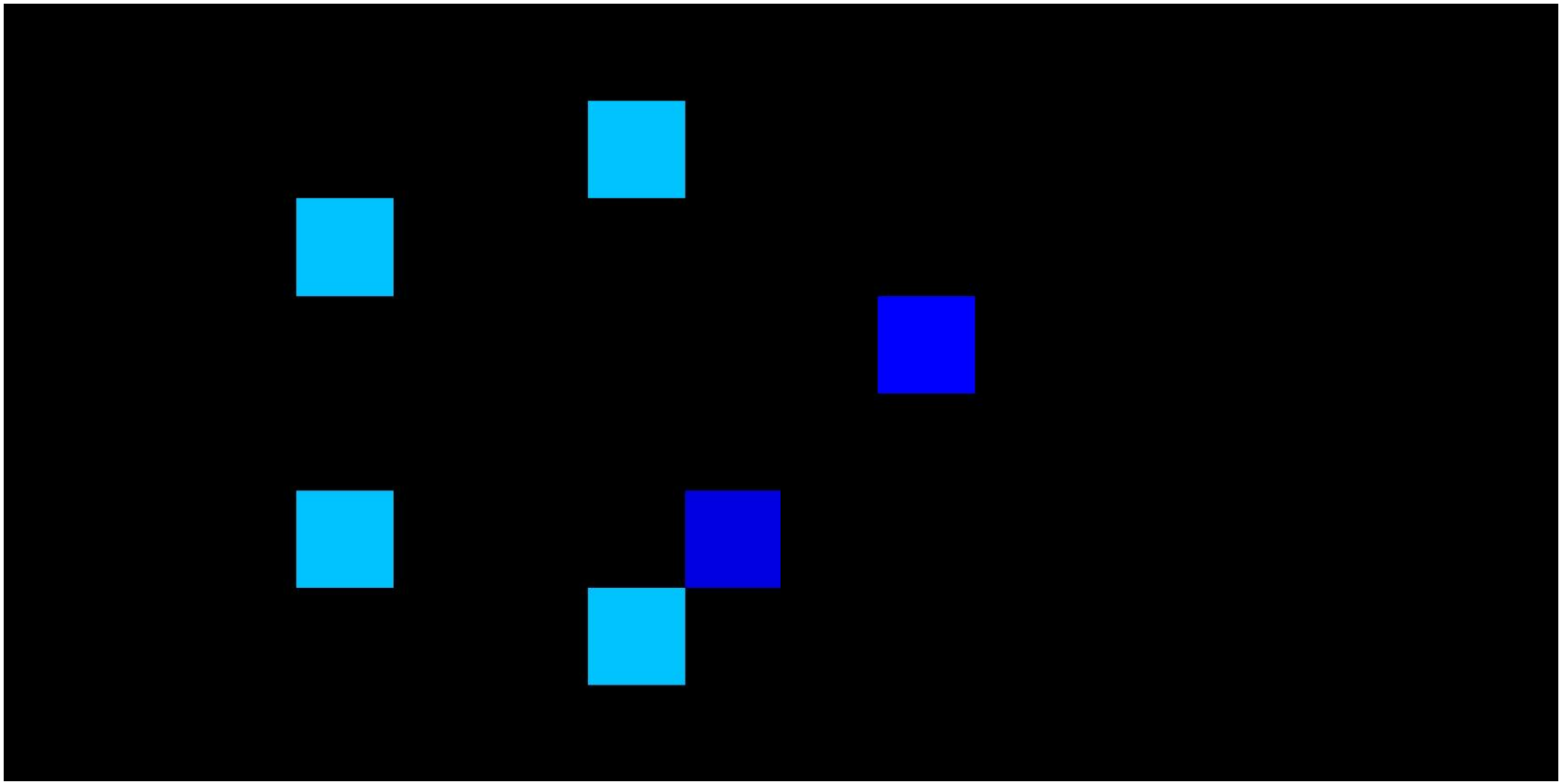
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



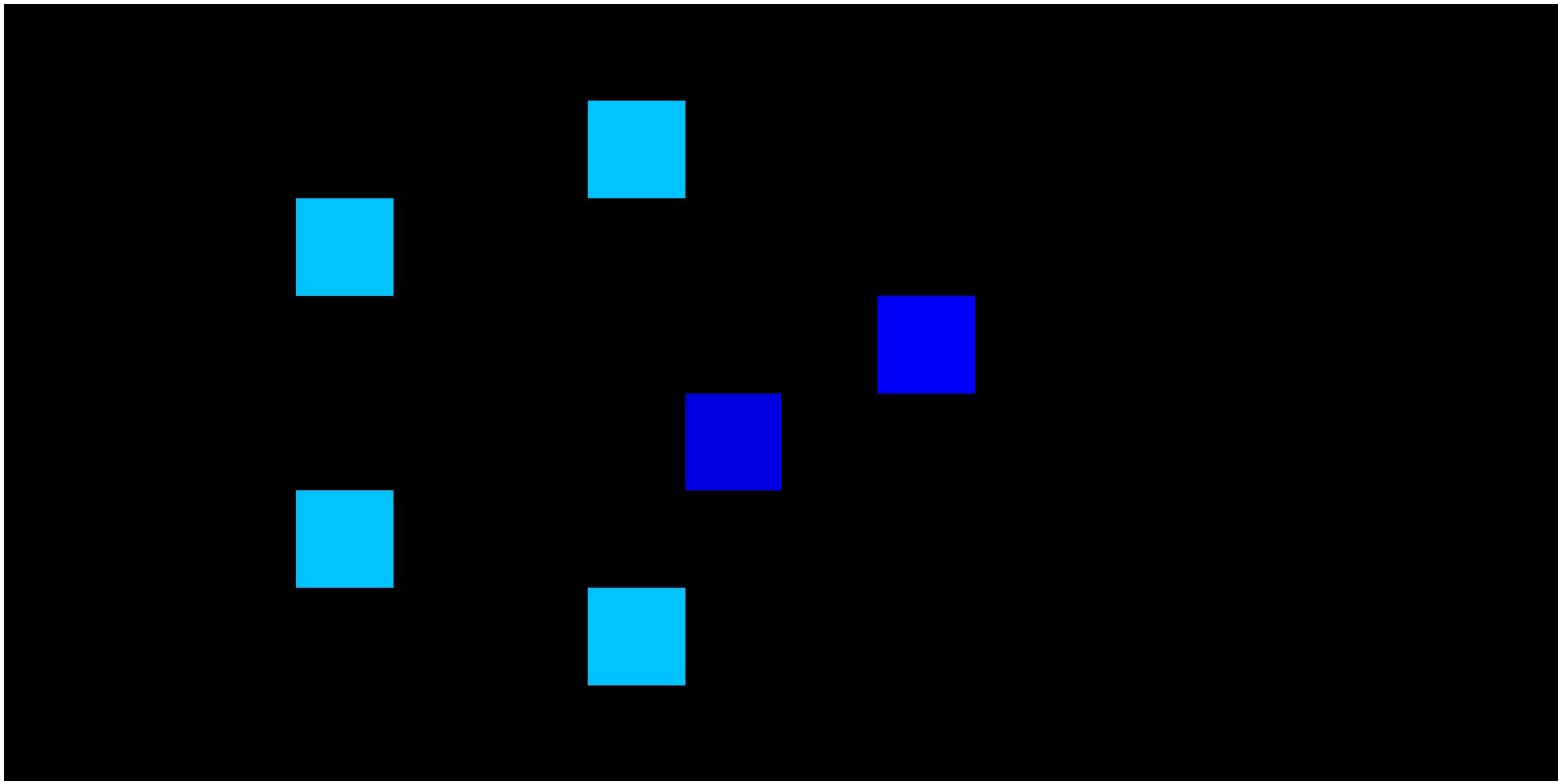
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



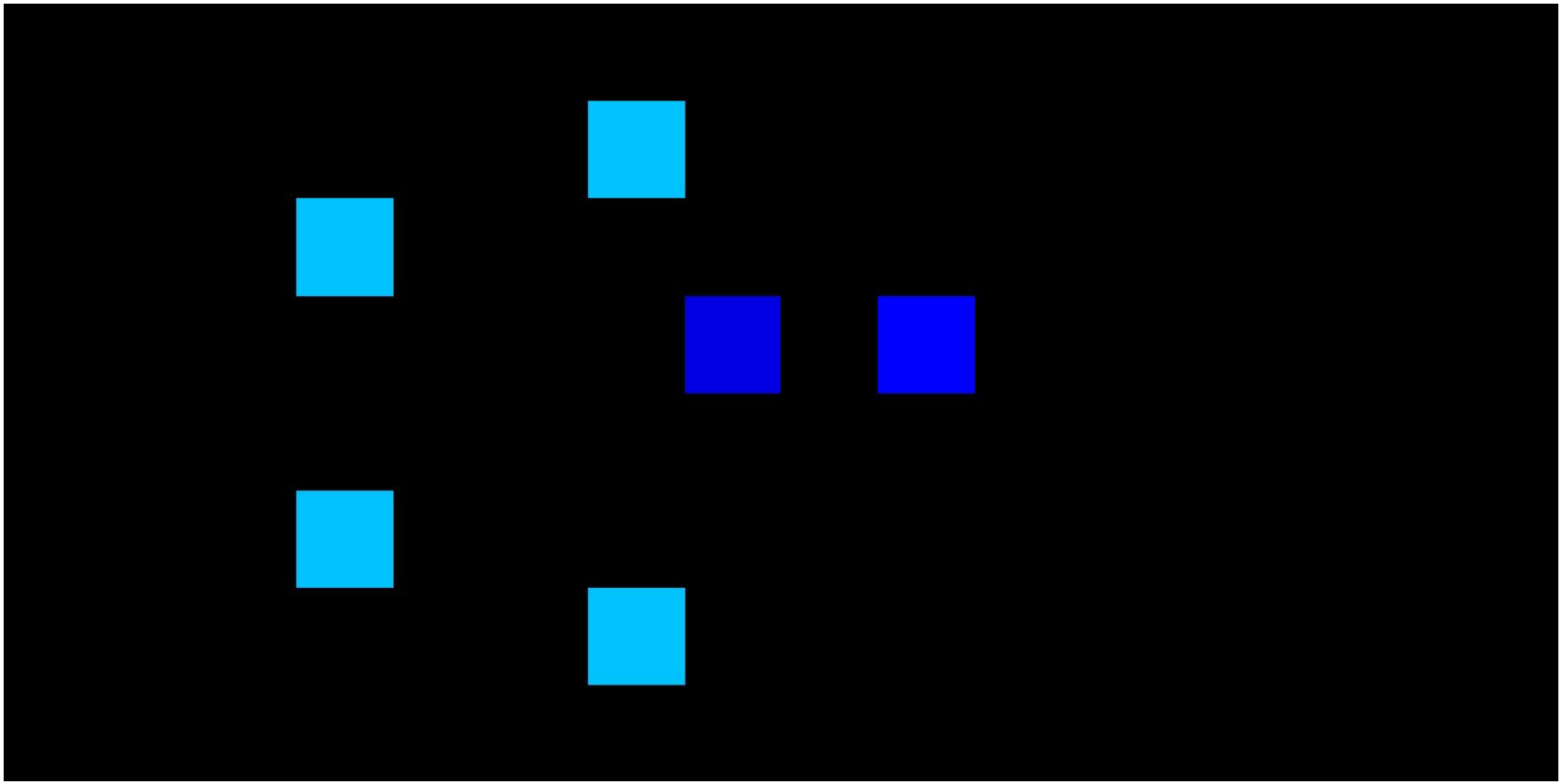
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



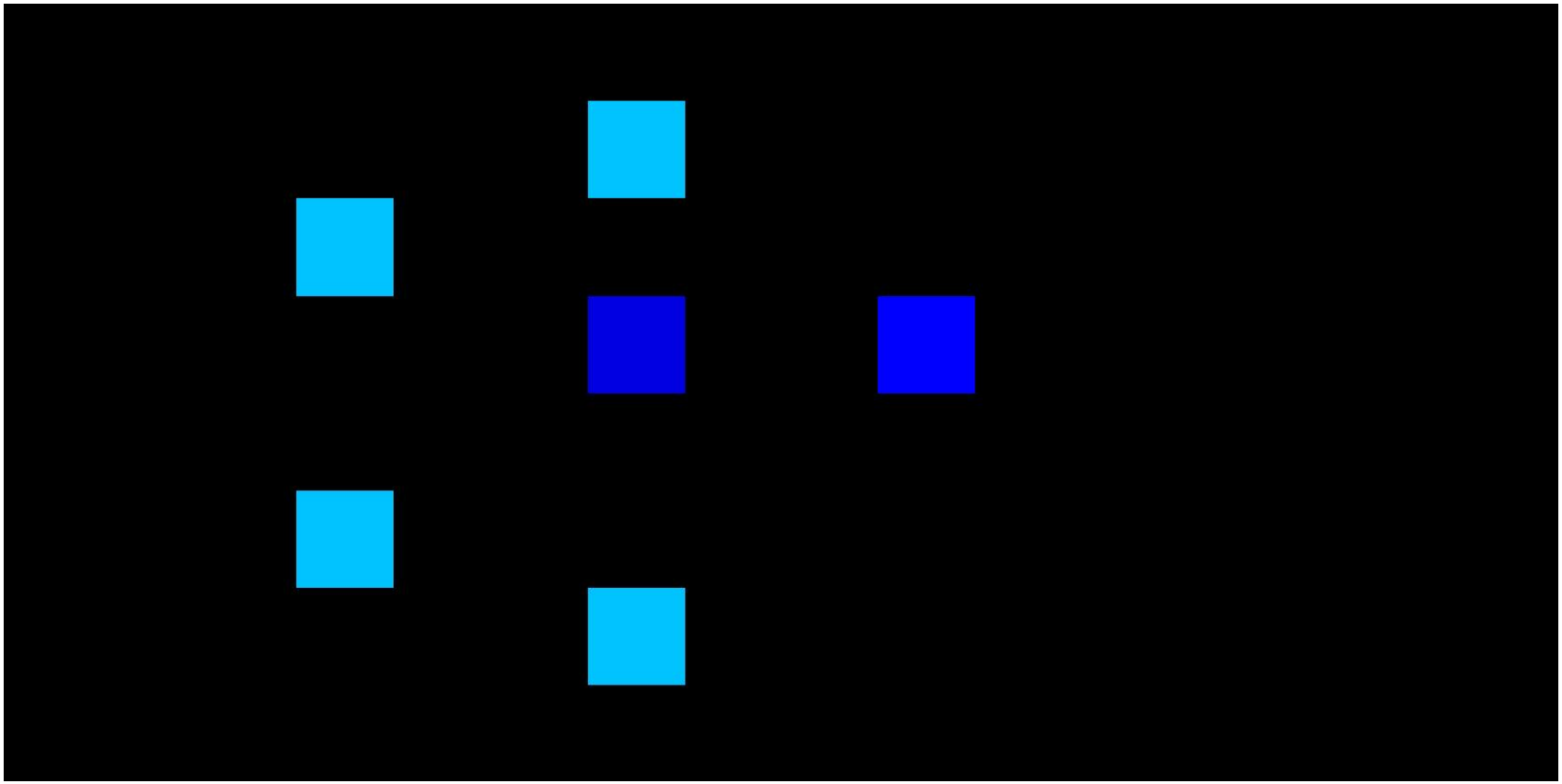
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



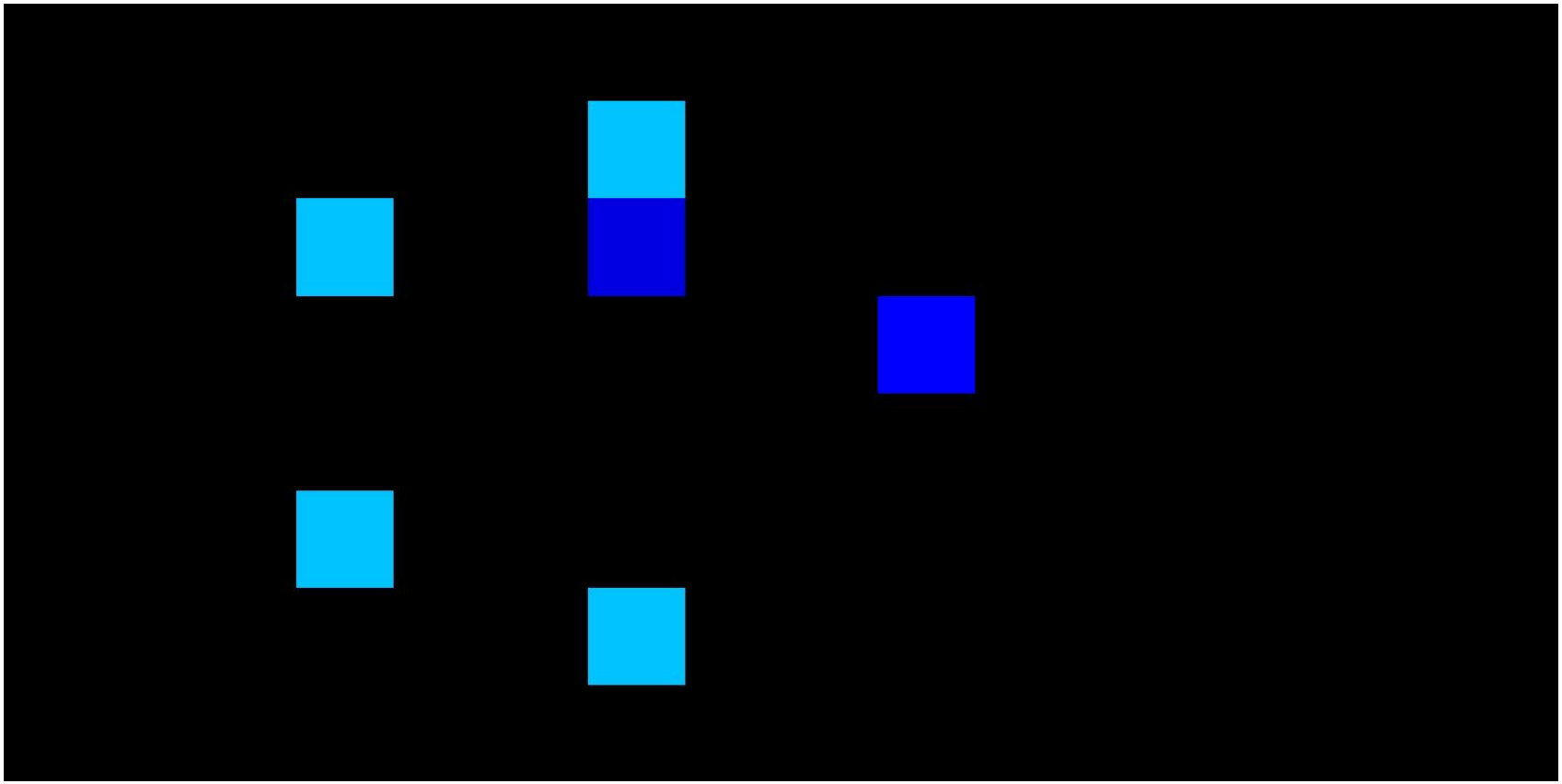
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



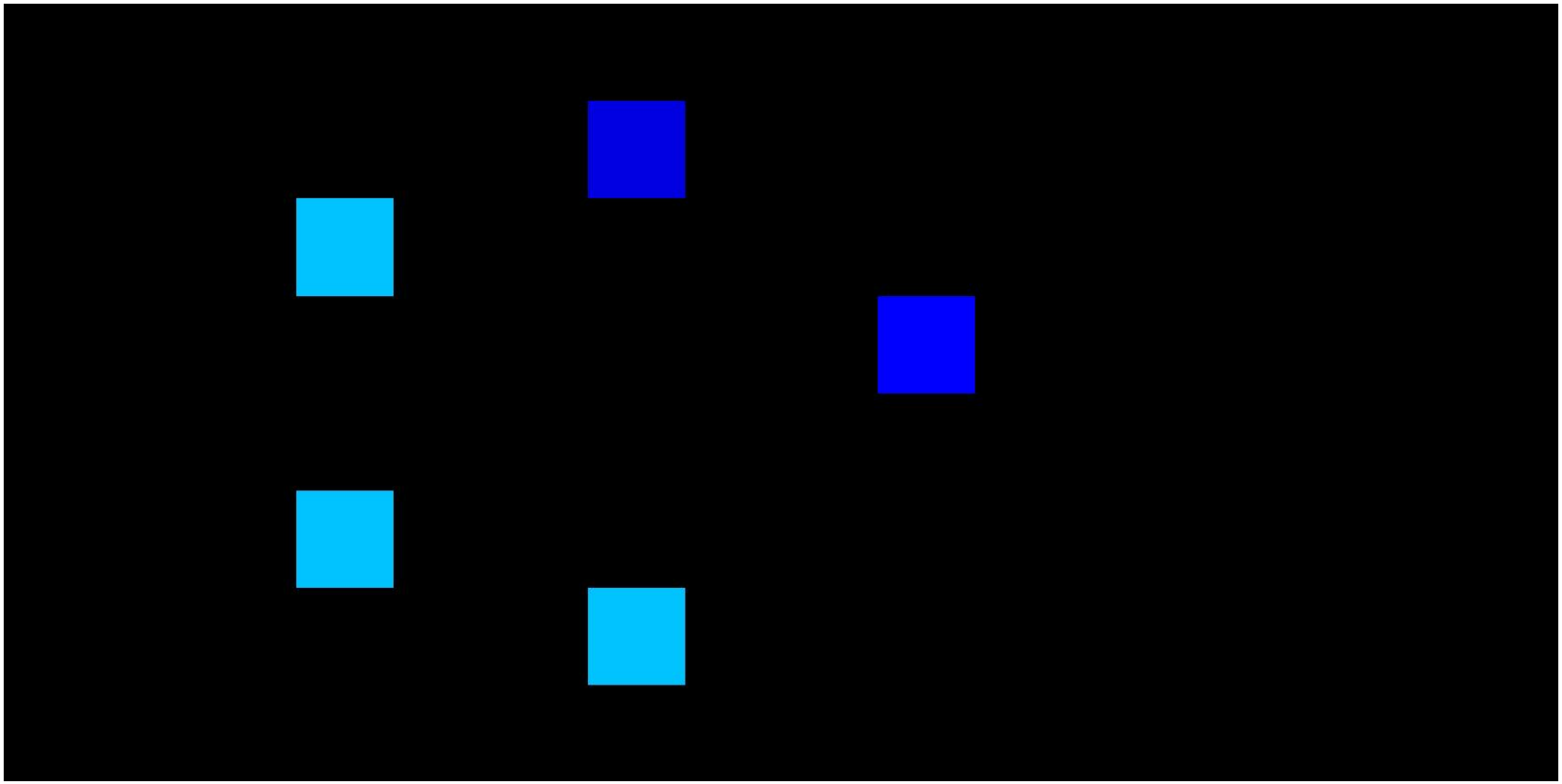
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



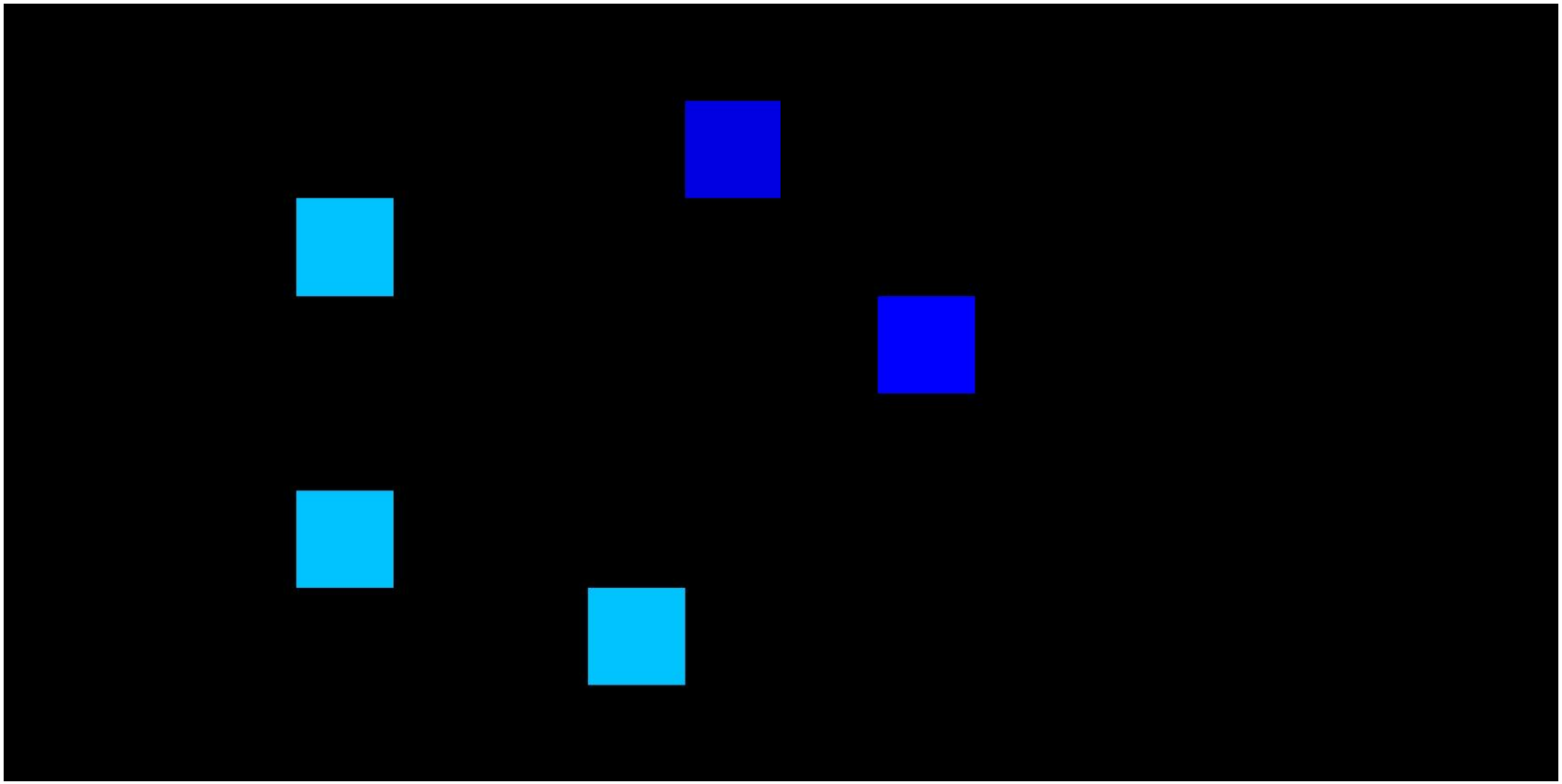
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



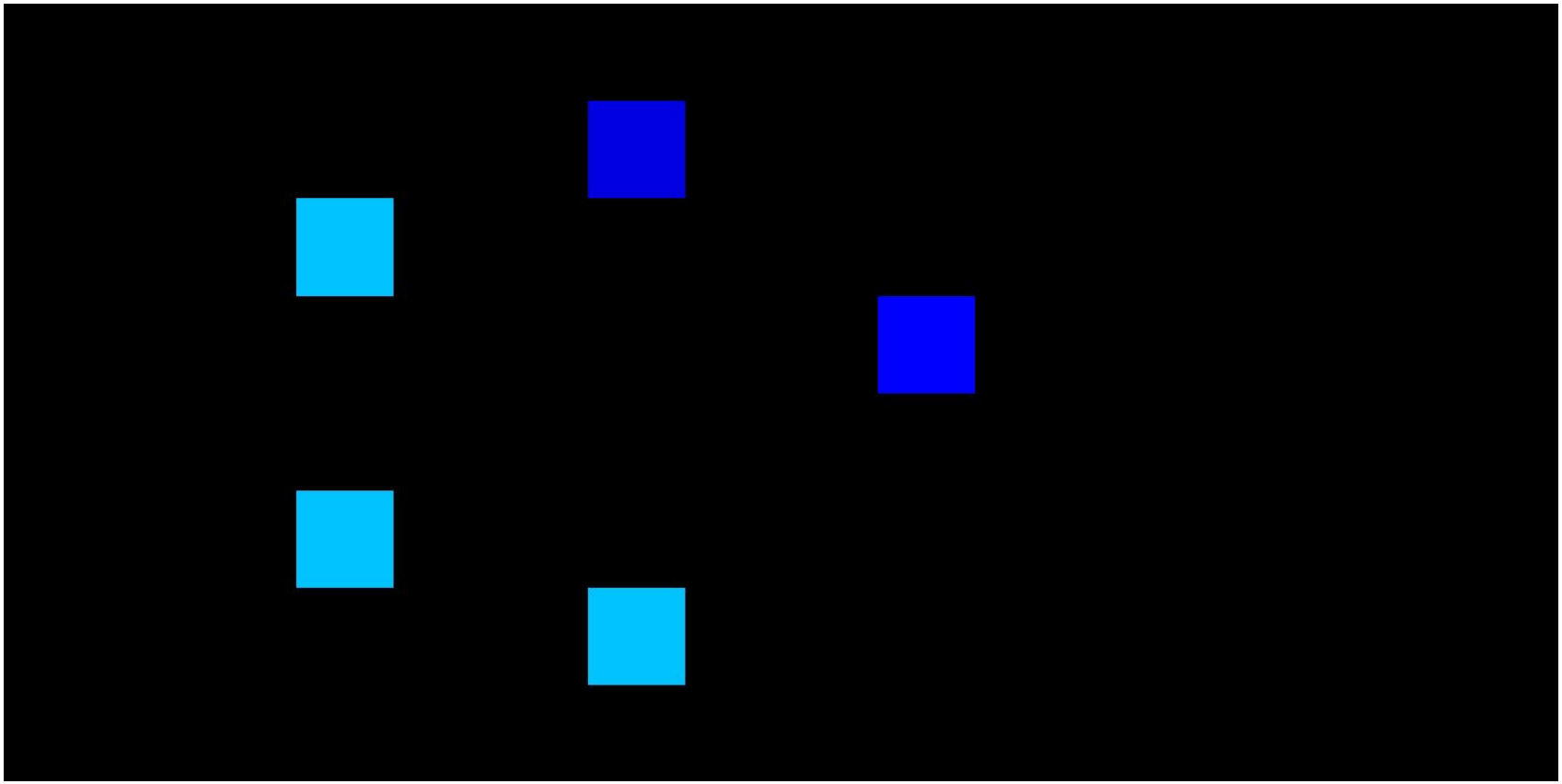
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



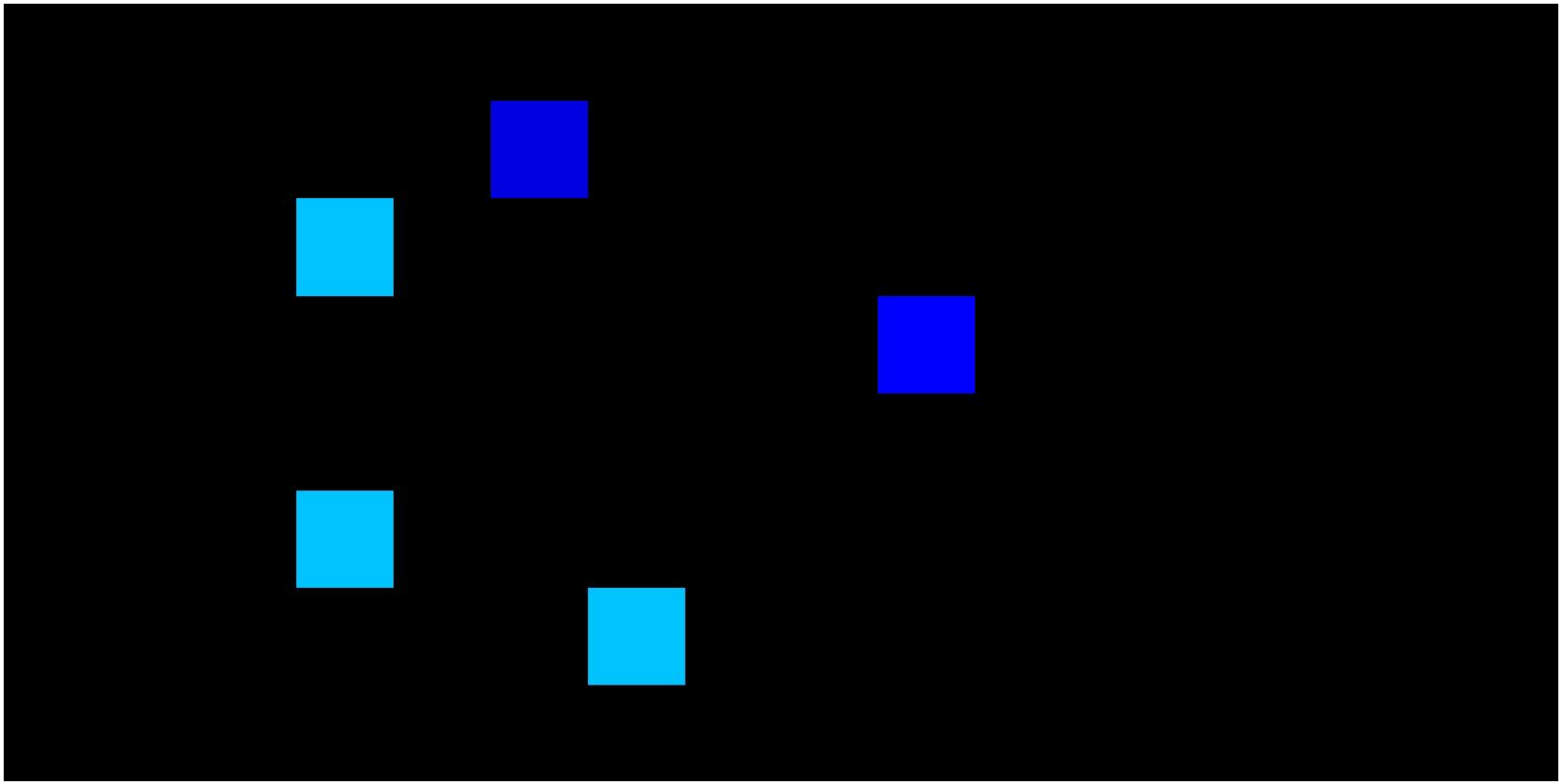
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



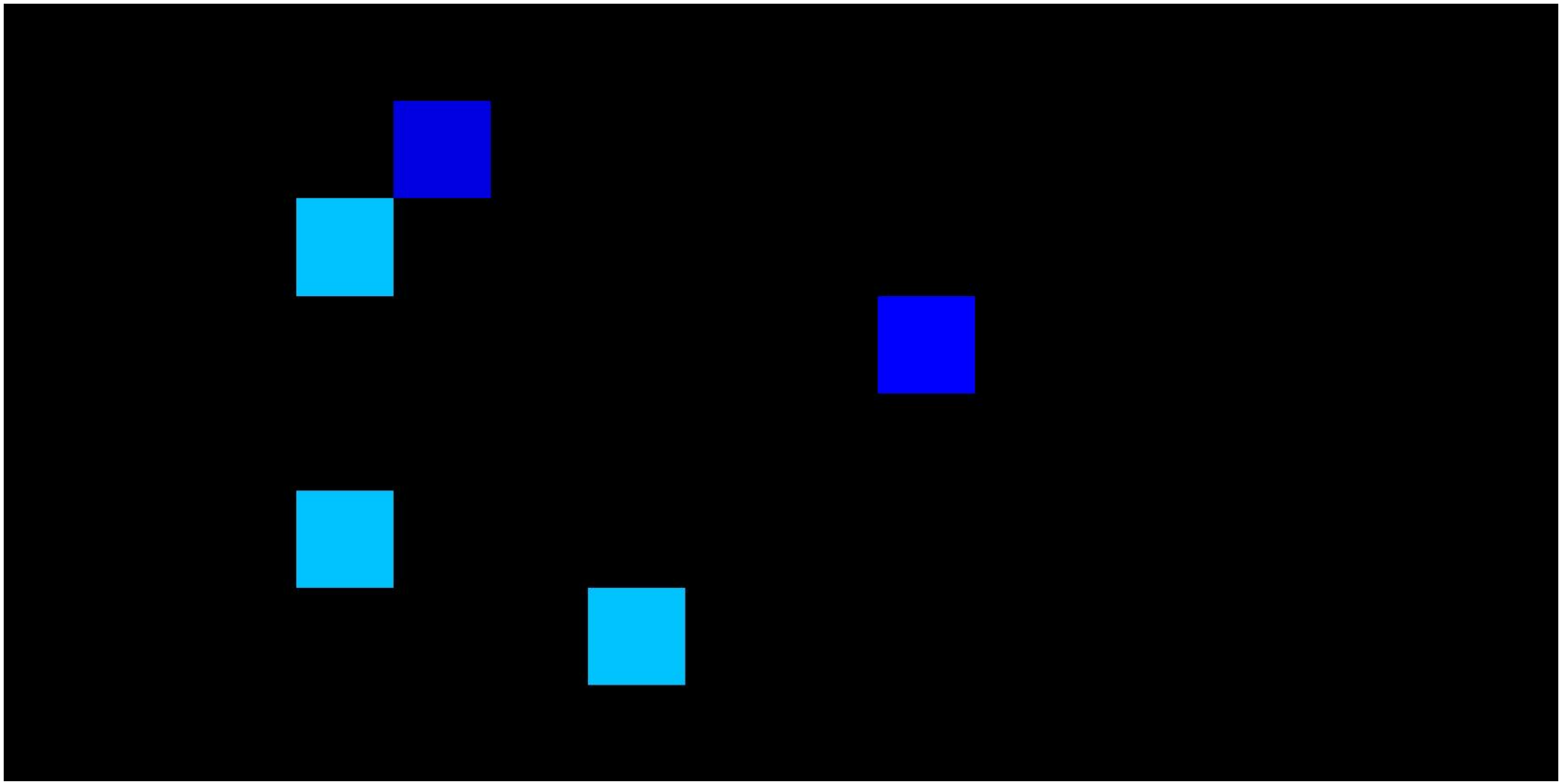
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



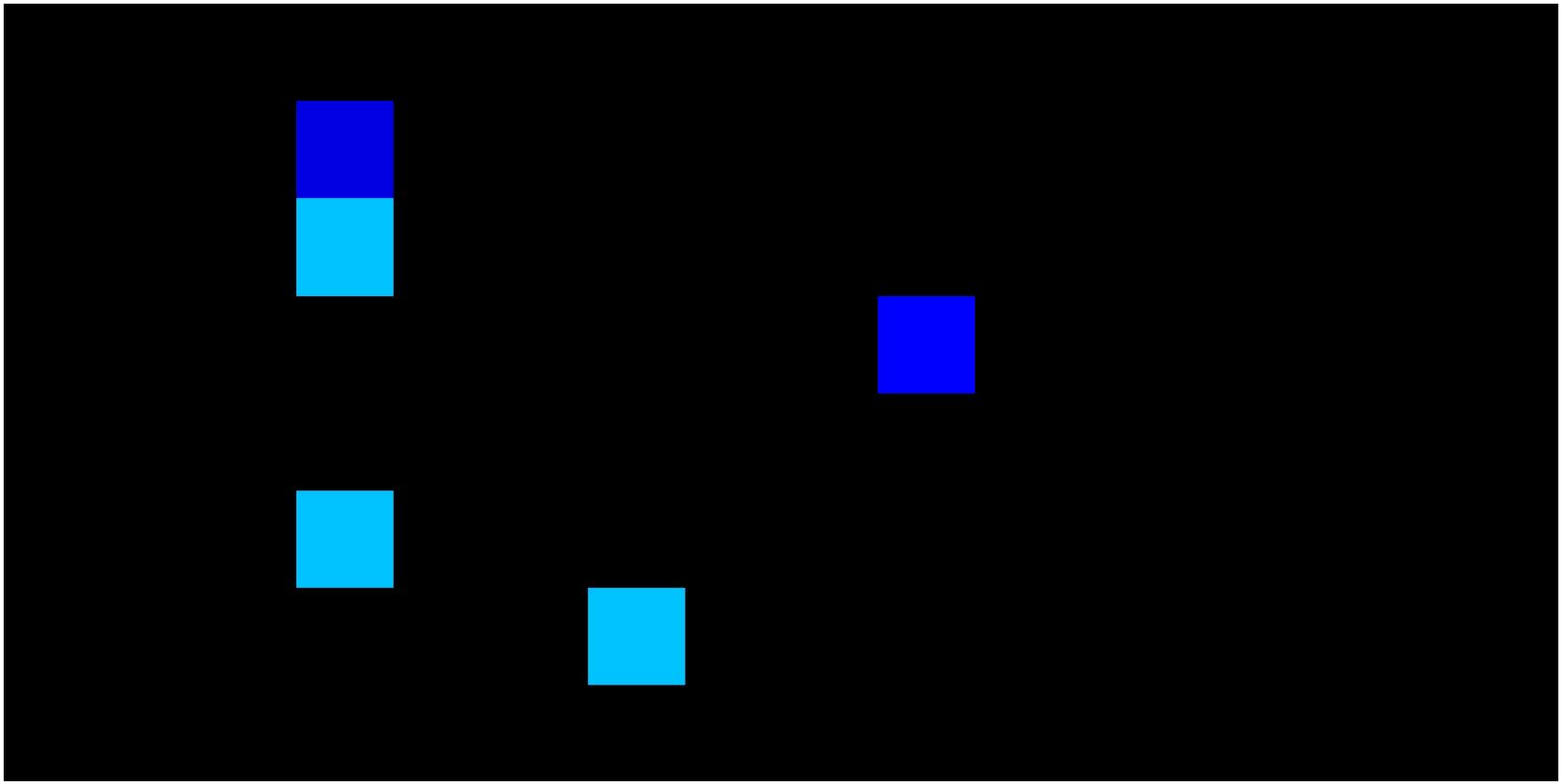
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



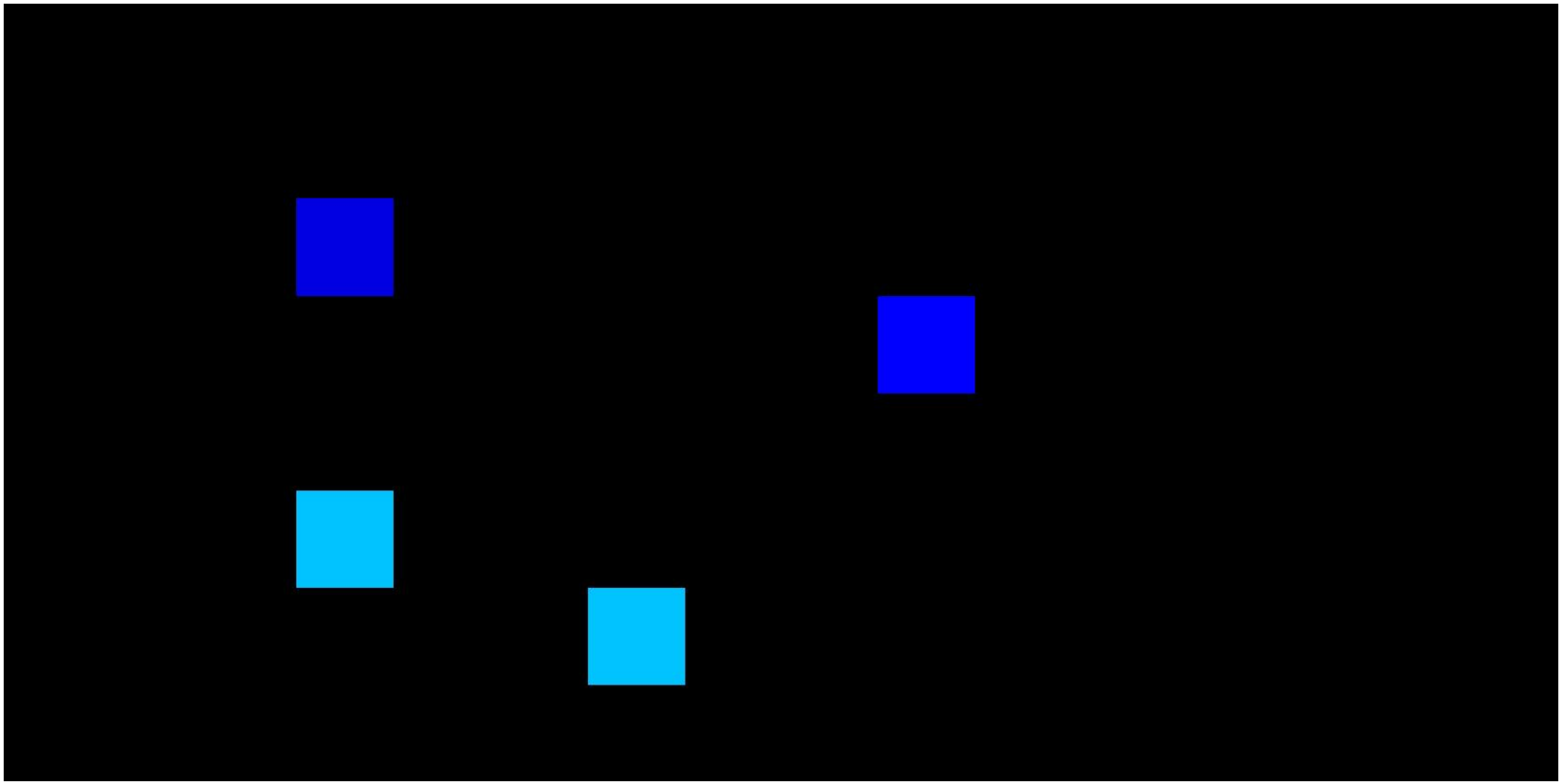
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



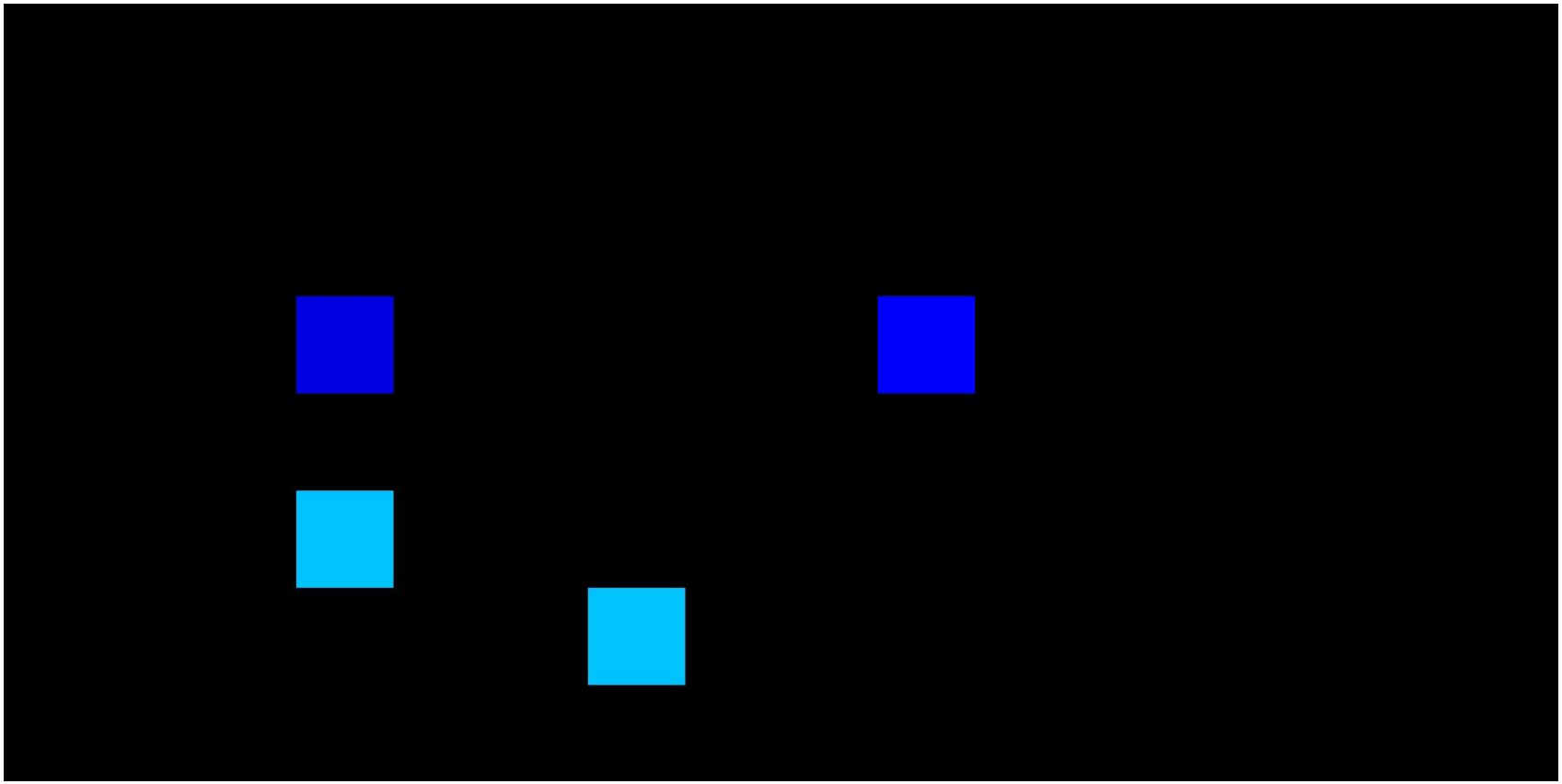
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



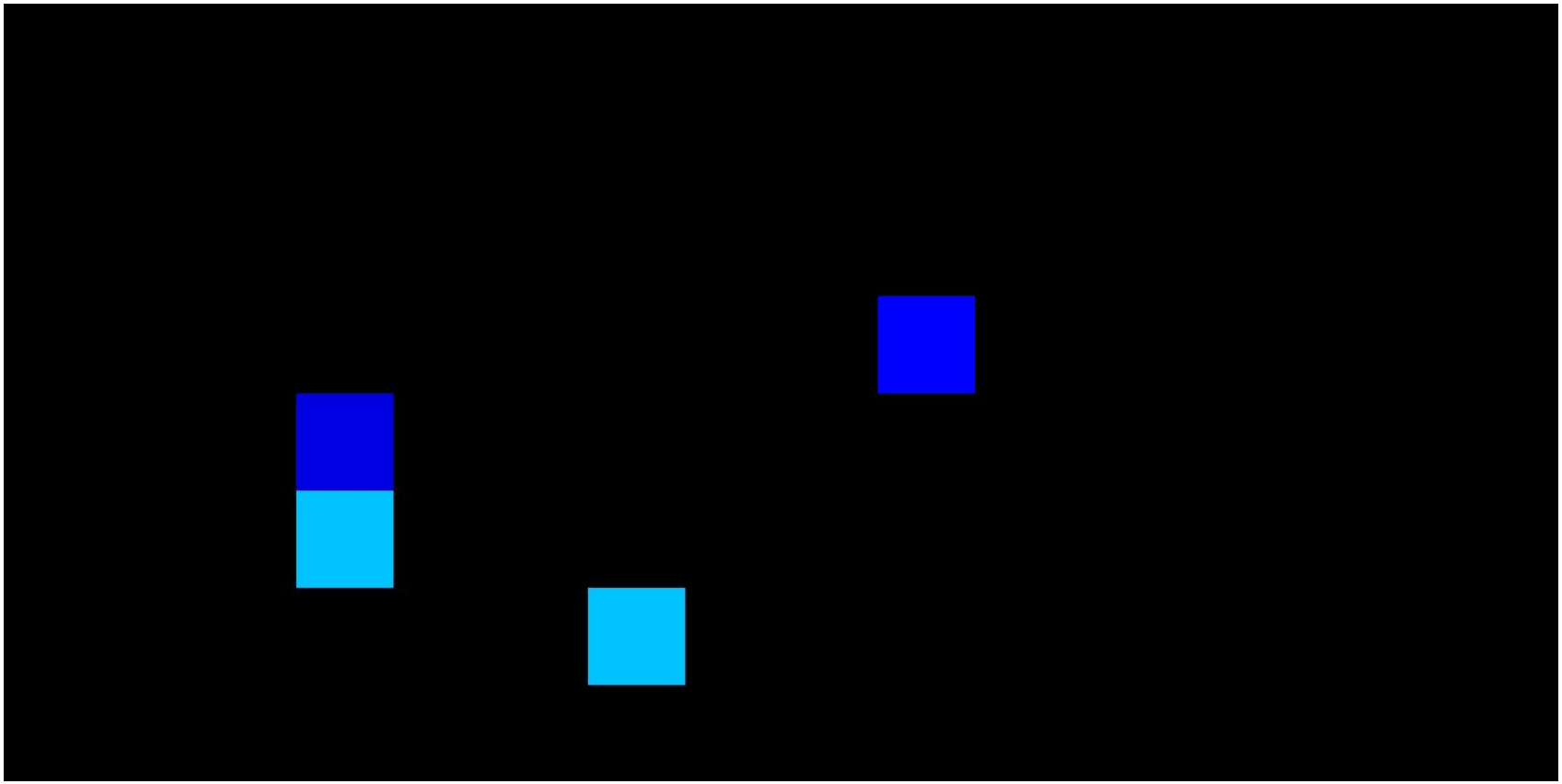
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



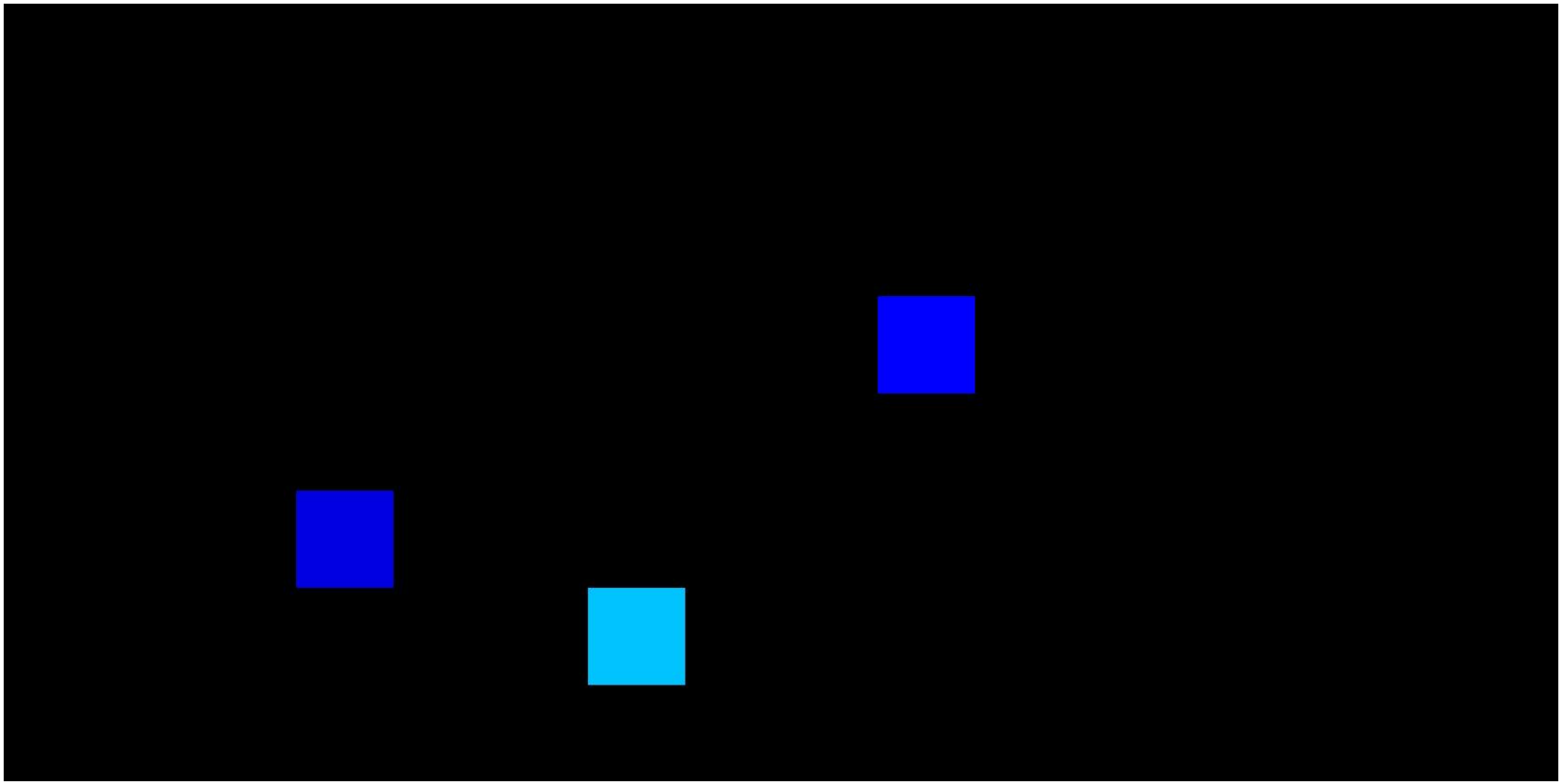
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



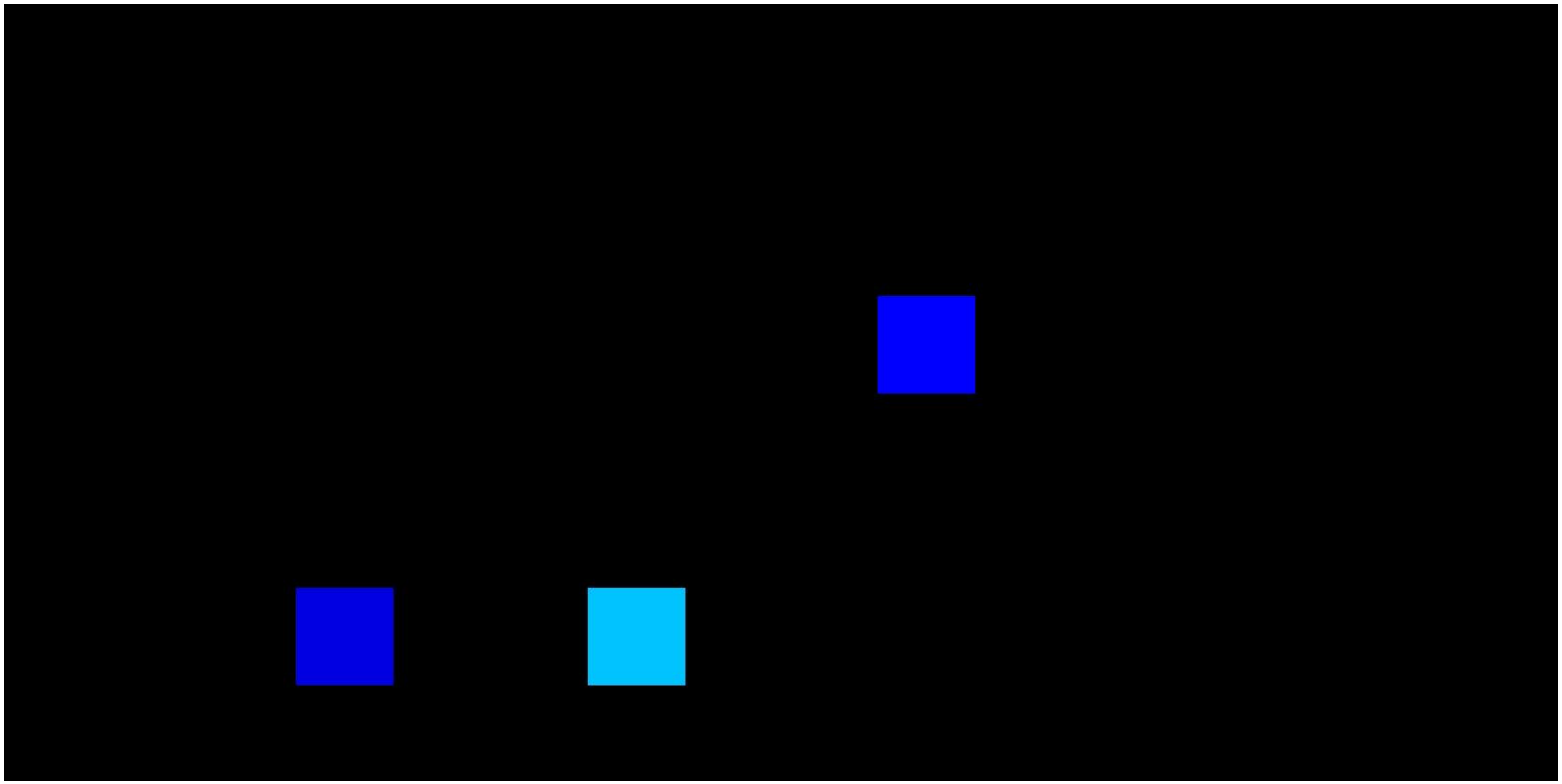
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



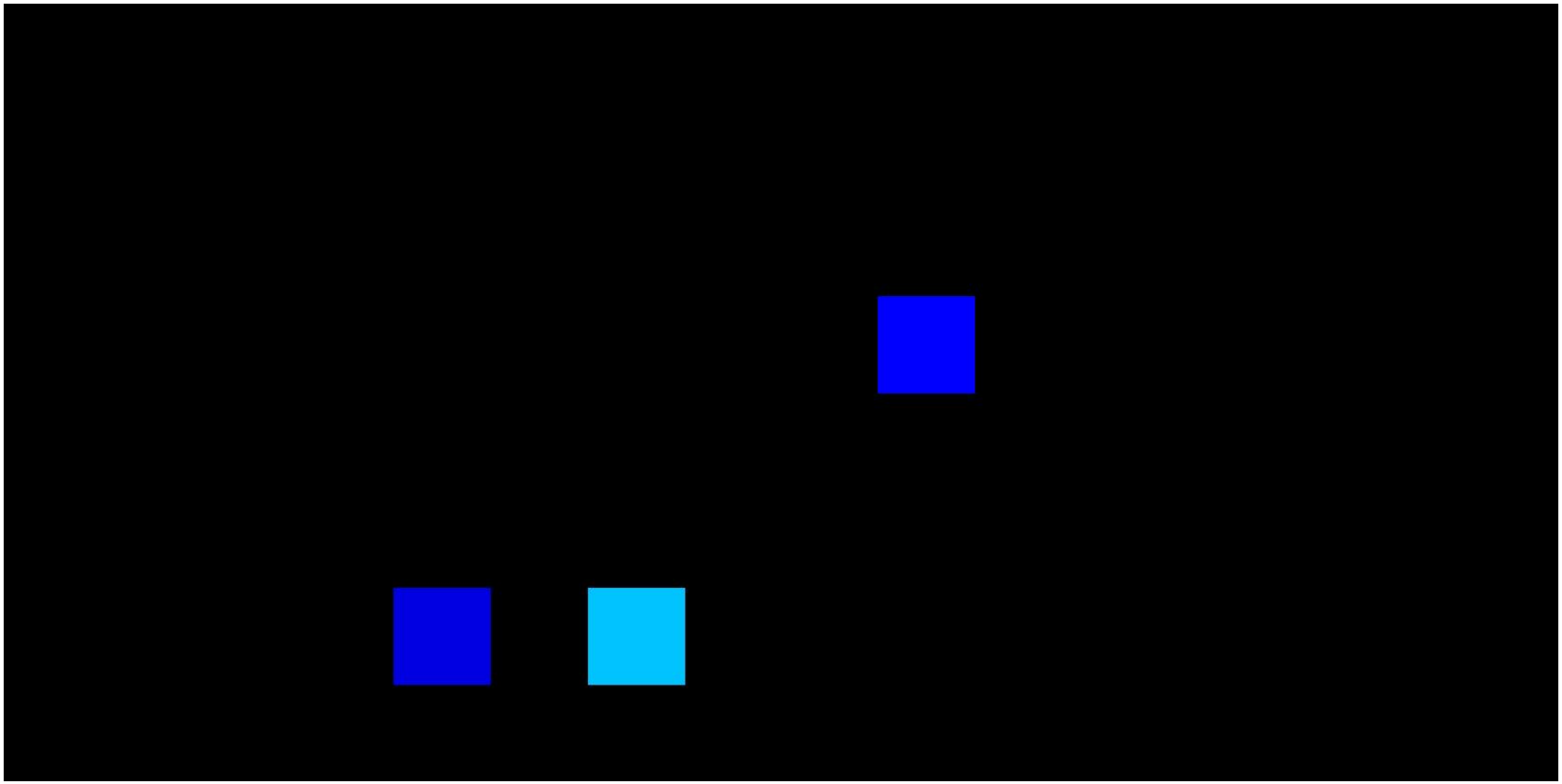
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



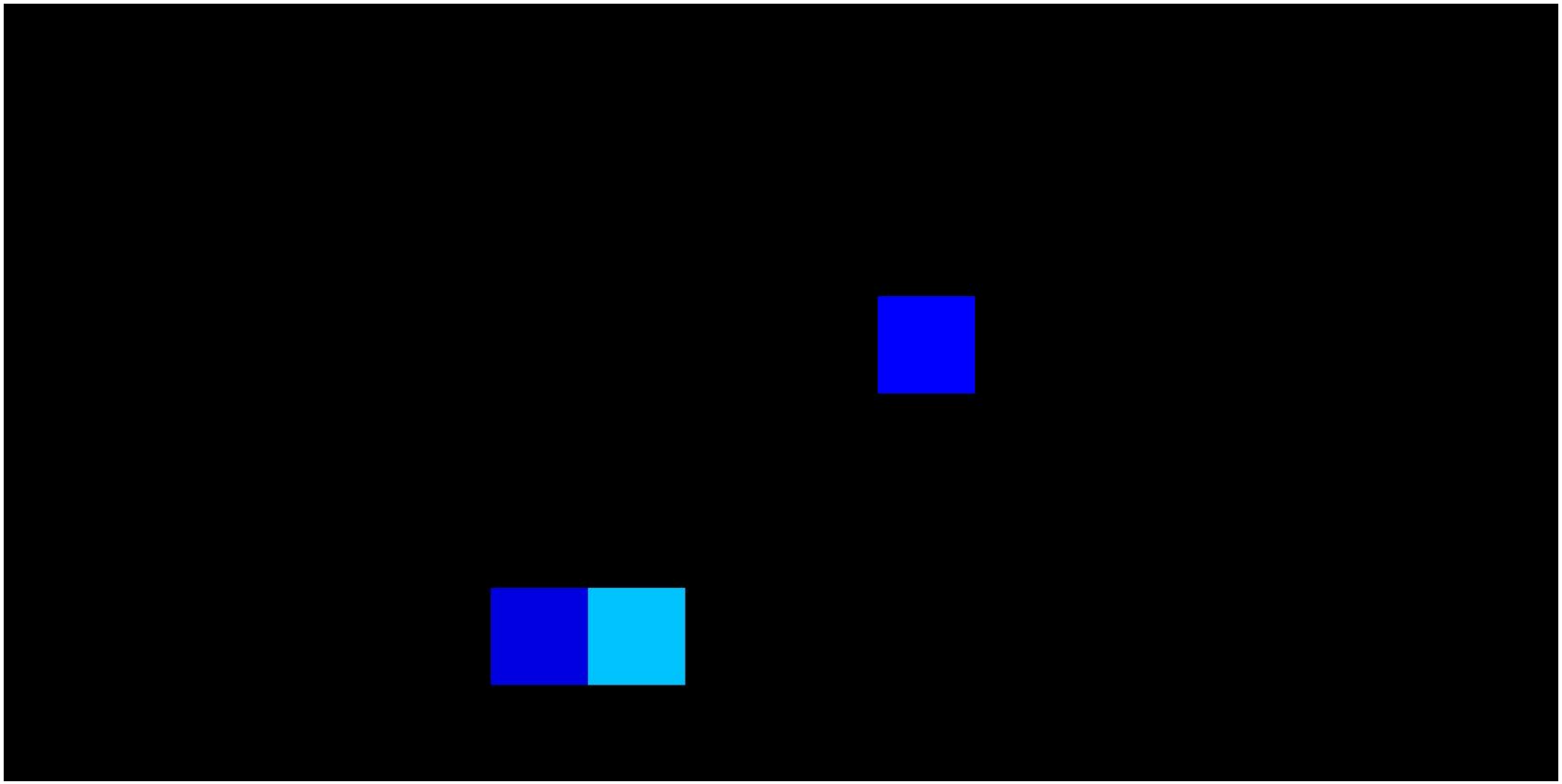
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



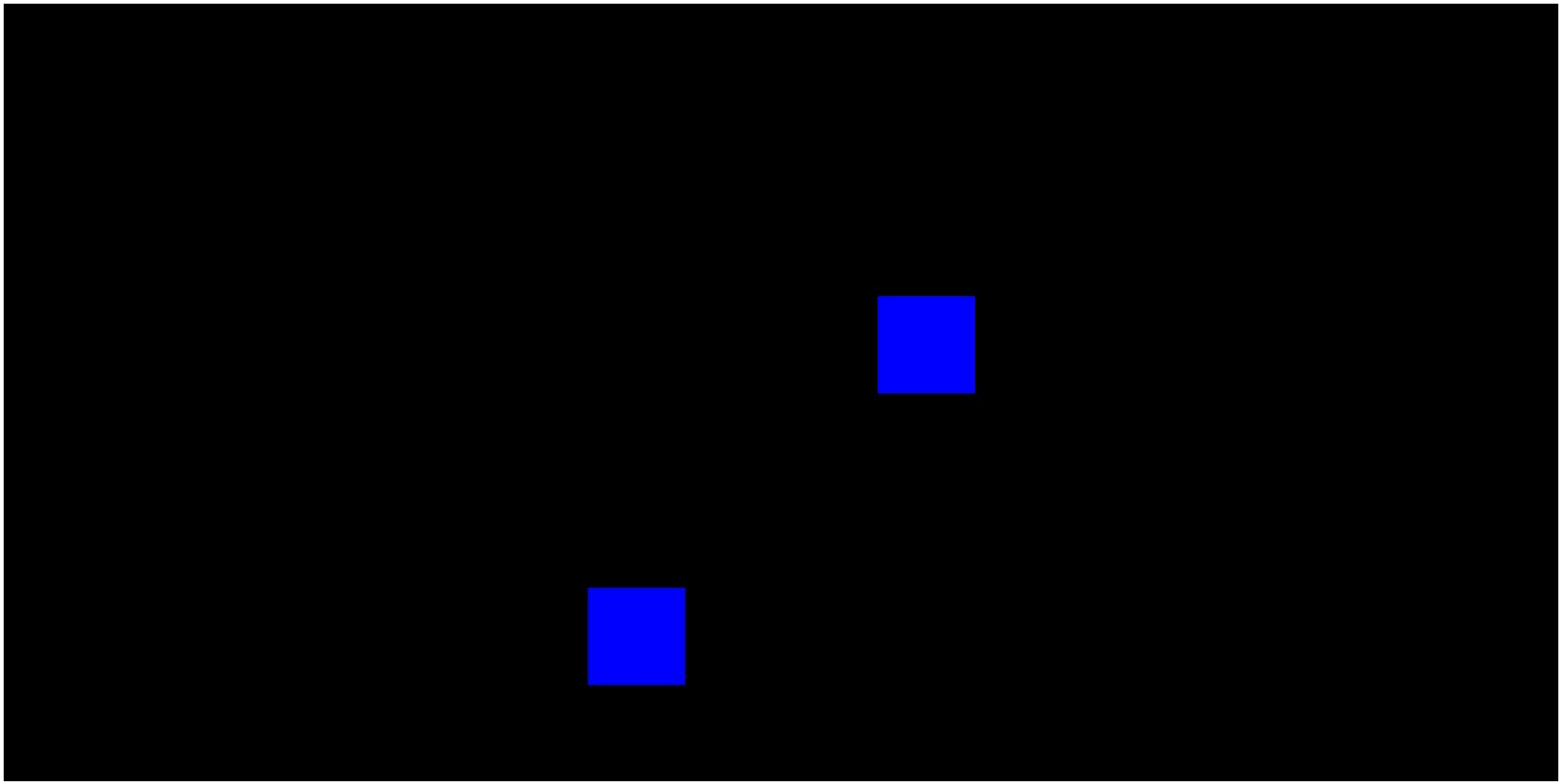
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



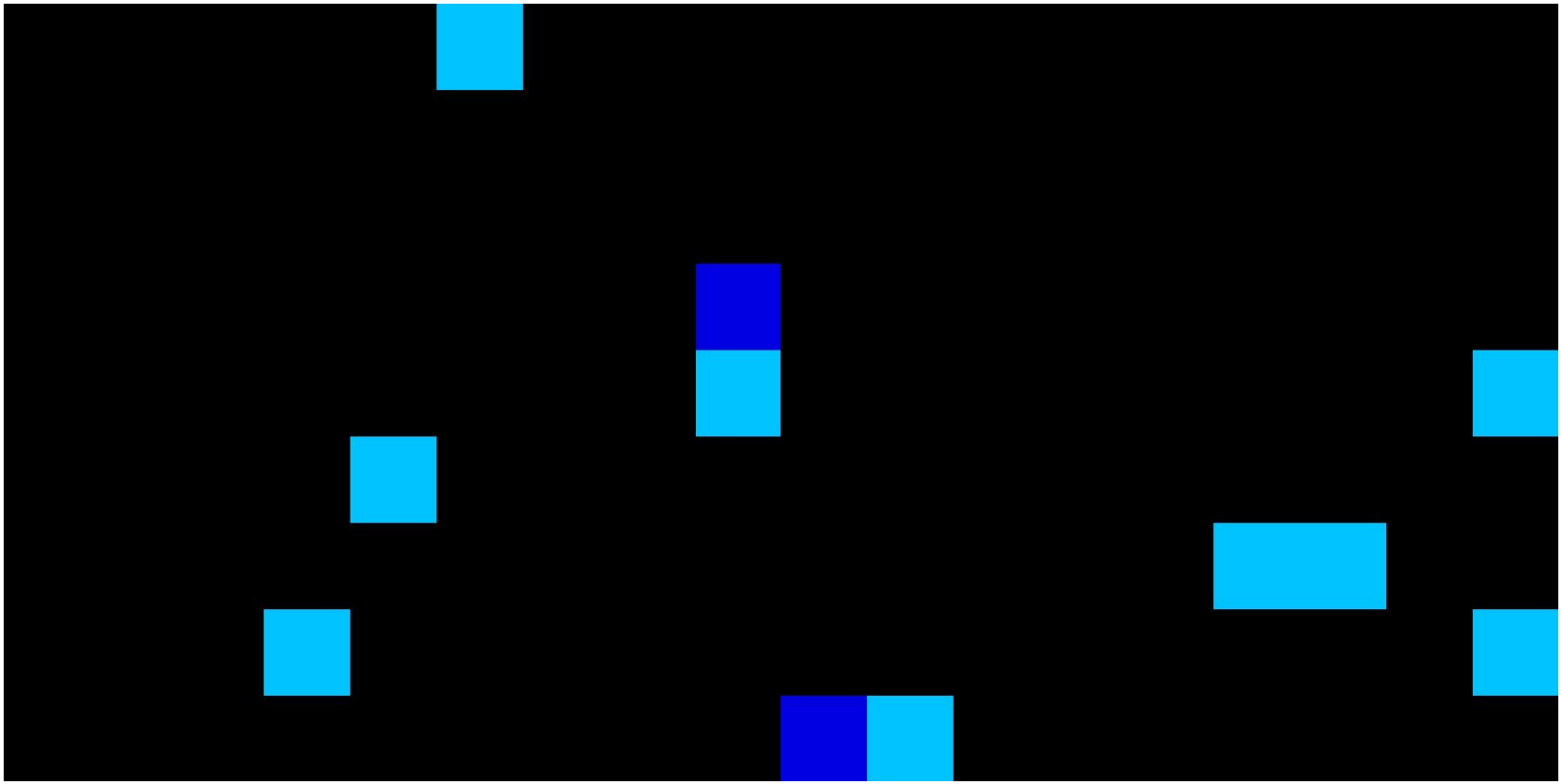
Resultados

- 8×8. Um milhão de épocas (Dois dias e meio). Acurácia 23.77 (Limite superior: 25)



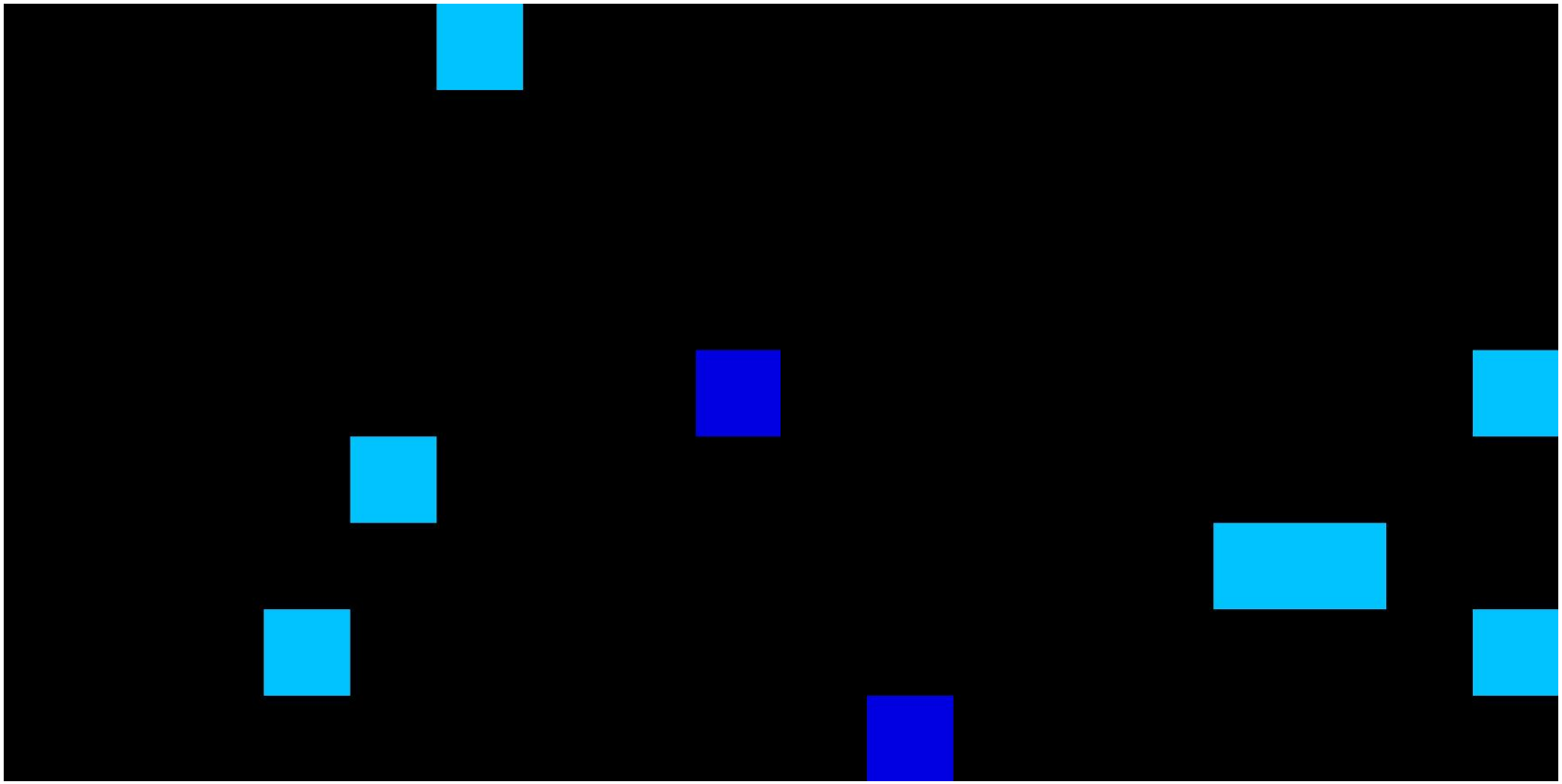
Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



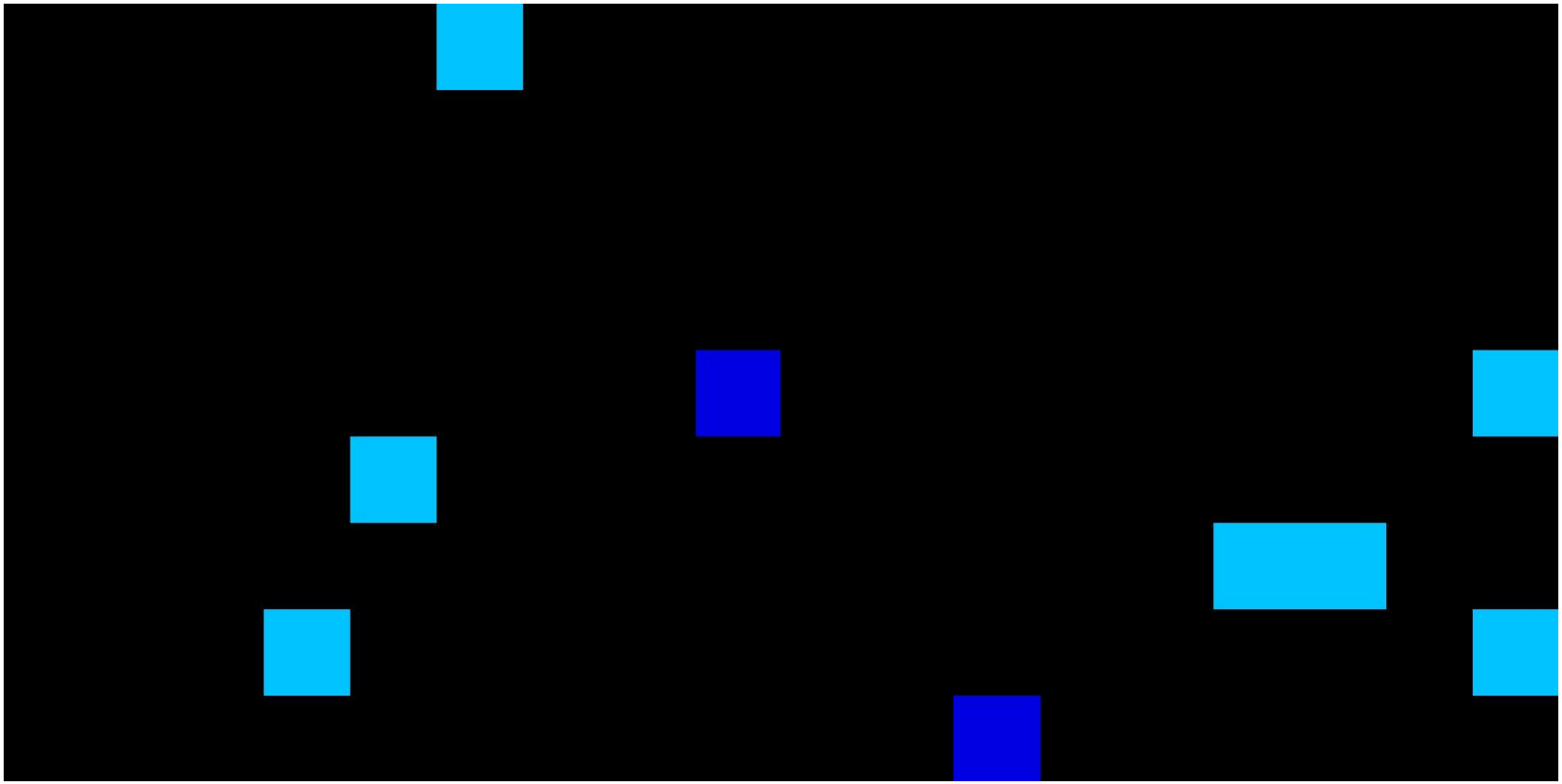
Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



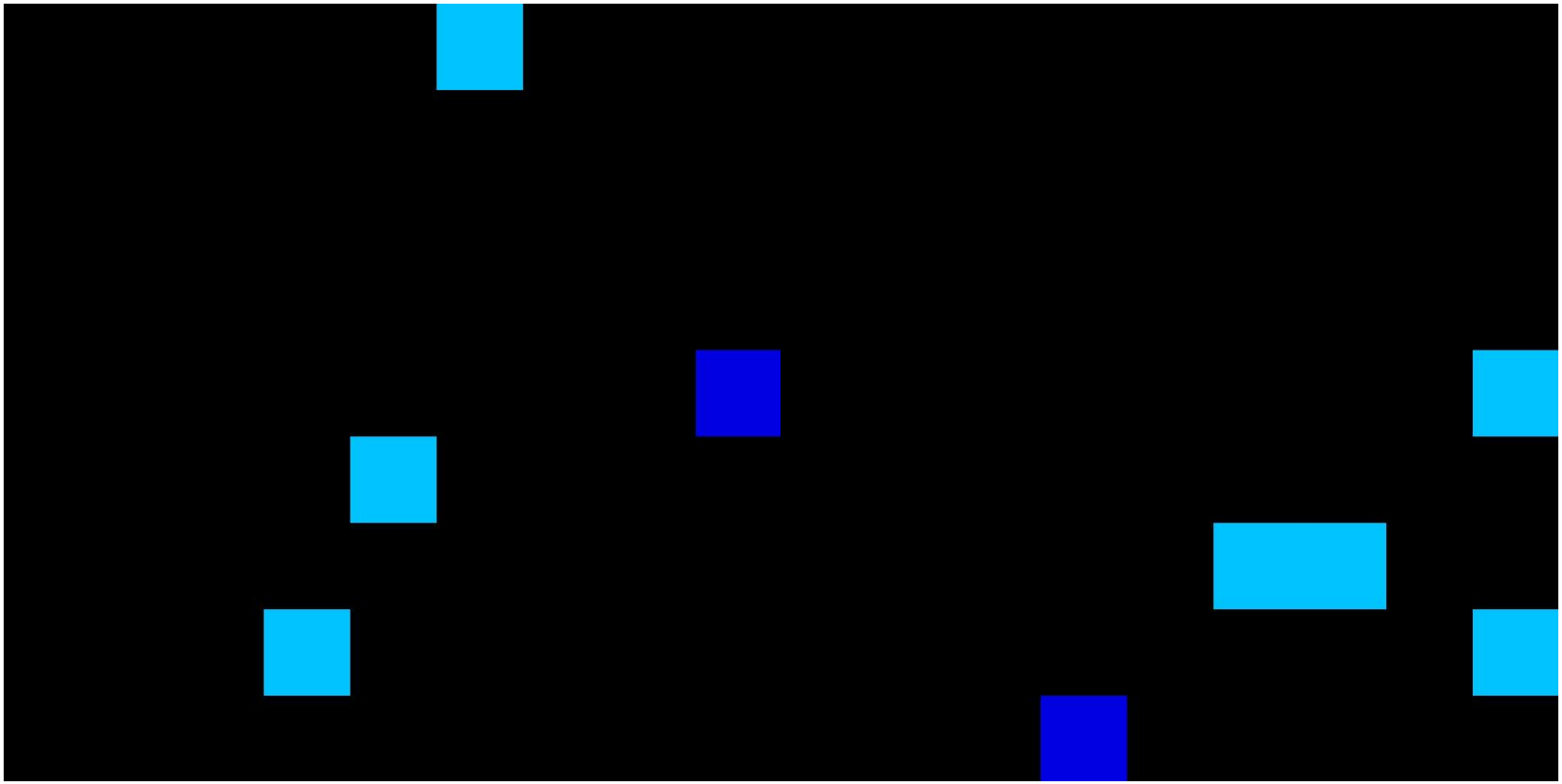
Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



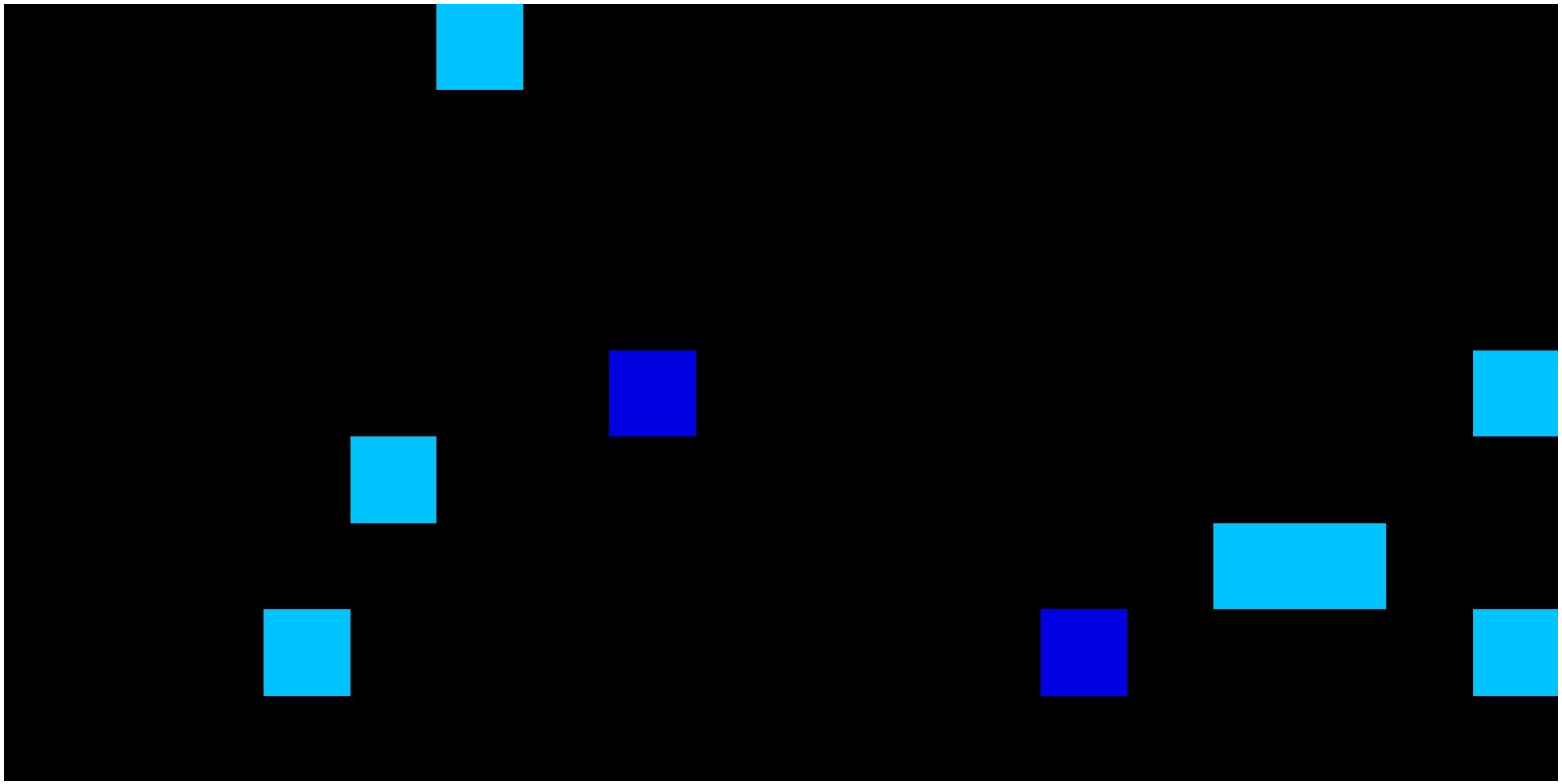
Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



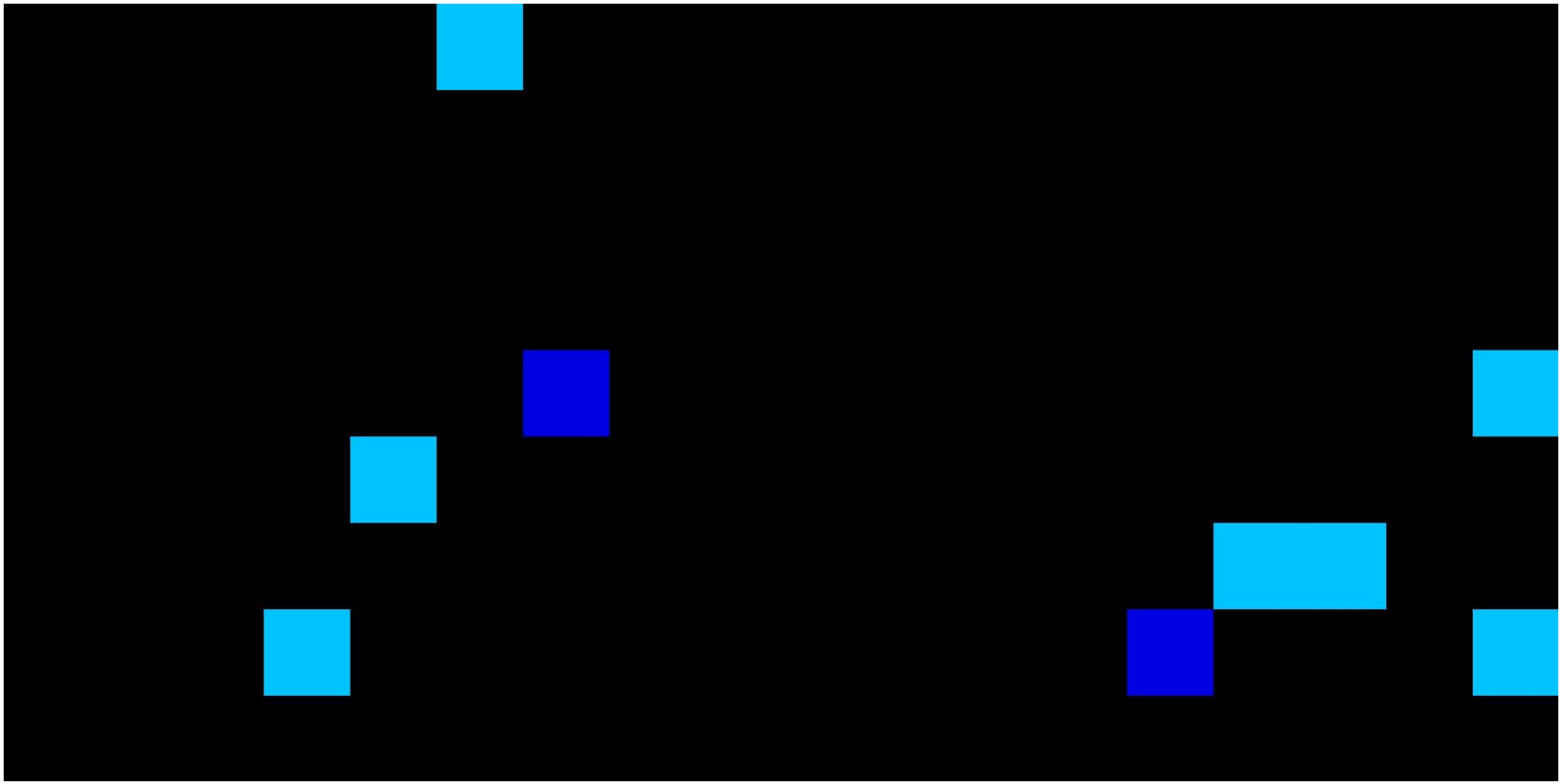
Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



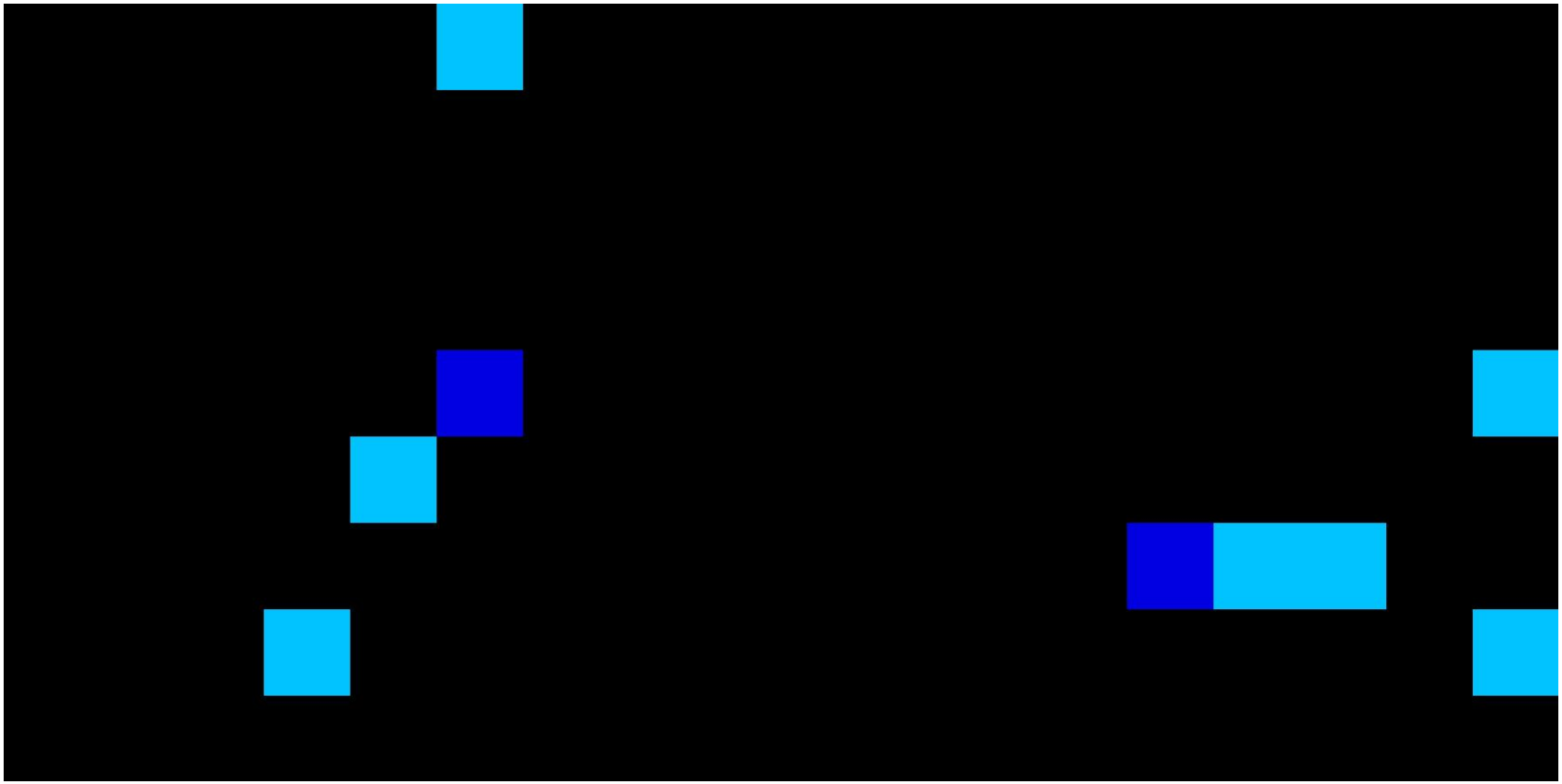
Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

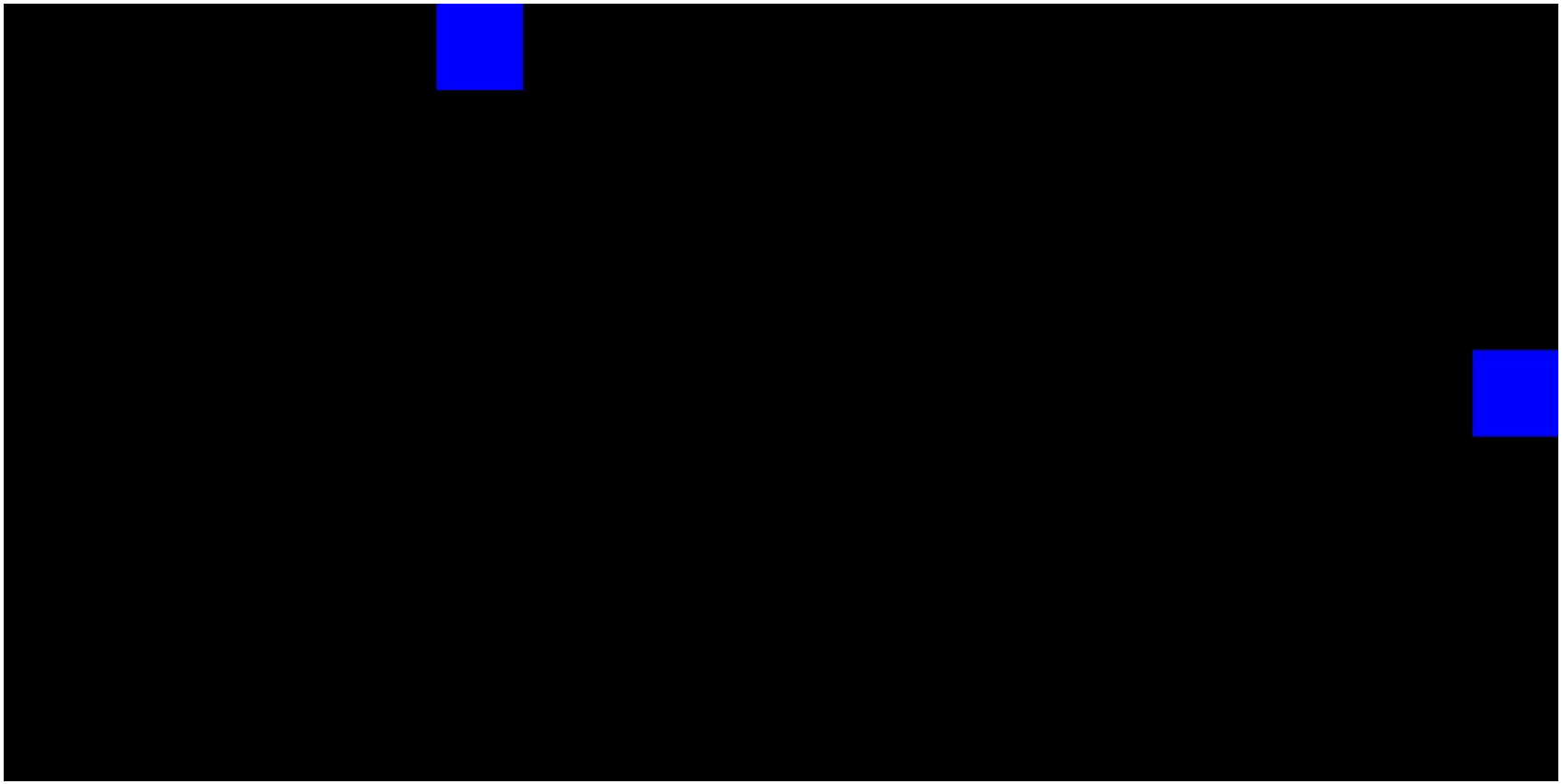
- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)

Rede DQN

- Rede DQN: 9×9. 15 mil épocas. (Cerca de 17h)



Conclusões

- O agente aprendeu a pegar os itens, apesar de estar preso em um **máximo local**.
  - Não procura um caminho pela **diagonal** (8×8).
- Por consequência **não obteve** melhor desempenho que o Deep Q-Learning.
- Para a entrada de 8×8 **necessita de muito mais treino** do que na entrada de 6×6
- Possívelmente a configuração de *rewards* e o número de épocas de ajuste da rede **levaram ao máximo local**.
- Também há possibilidade de que o algoritmo **não foi treinado** por tempo suficiente.

Conclusões

- O agente aprendeu a pegar os itens, apesar de estar preso em um **máximo local**.
  - Não procura um caminho pela **diagonal** (8×8).
- Por consequência **não obteve** melhor desempenho que o Deep Q-Learning.
- Para a entrada de 8×8 **necessita de muito mais treino** do que na entrada de 6×6
- Possívelmente a configuração de *rewards* e o número de épocas de ajuste da rede **levaram ao máximo local**.
- Também há possibilidade de que o algoritmo **não foi treinado** por tempo suficiente.

Conclusões

- O agente aprendeu a pegar os itens, apesar de estar preso em um **máximo local**.
  - Não procura um caminho pela **diagonal** (8×8).
- Por consequência **não obteve** melhor desempenho que o Deep Q-Learning.
- Para a entrada de 8×8 **necessita de muito mais treino** do que na entrada de 6×6
- Possívelmente a configuração de *rewards* e o número de épocas de ajuste da rede **levaram ao máximo local**.
- Também há possibilidade de que o algoritmo **não foi treinado** por tempo suficiente.

Conclusões

- O agente aprendeu a pegar os itens, apesar de estar preso em um **máximo local**.
  - Não procura um caminho pela **diagonal** (8×8).
- Por consequência **não obteve** melhor desempenho que o Deep Q-Learning.
- Para a entrada de 8×8 **necessita de muito mais trein**o do que na entrada de 6×6
- Possívelmente a configuração de *rewards* e o número de épocas de ajuste da rede **levaram ao máximo local**.
- Também há possibilidade de que o algoritmo **não foi treinado** por tempo suficiente.

Conclusões

- O agente aprendeu a pegar os itens, apesar de estar preso em um **máximo local**.
  - Não procura um caminho pela **diagonal** (8×8).
- Por consequência **não obteve** melhor desempenho que o Deep Q-Learning.
- Para a entrada de 8×8 **necessita de muito mais trein**o do que na entrada de 6×6
- Possívelmente a configuração de *rewards* e o número de épocas de ajuste da rede **levaram ao máximo local**.
- Também há possibilidade de que o algoritmo **não foi treinado** por tempo suficiente.

Conclusões

- O agente aprendeu a pegar os itens, apesar de estar preso em um **máximo local**.
  - Não procura um caminho pela **diagonal** (8×8).
- Por consequência **não obteve** melhor desempenho que o Deep Q-Learning.
- Para a entrada de 8×8 **necessita de muito mais trein**o do que na entrada de 6×6
- Possívelmente a configuração de *rewards* e o número de épocas de ajuste da rede **levaram ao máximo local**.
- Também há possibilidade de que o algoritmo **não foi treinado** por tempo suficiente.

Conclusões

- Possíveis opções de melhoria de desempenho:
  - **diminuir** o número de iterações de *fitting* da rede para menor que $N\_TRAJECTORIES/2$.
  - Utilizar uma rede **convolucional** de uma única camada e treinar por **muito mais tempo**.
  - Observar **melhor a presença** de um máximo local utilizando o limite superior ($N\_ITEMS * 5$).
  - Modificar os rewards de forma em que o número de passos tenham **maior penalização**. $->$ Diminuir a tendência em máximo locais.

Conclusões

- Possíveis opções de melhoria de desempenho:
  - **diminuir** o número de iterações de *fitting* da rede para menor que $N\_TRAJECTORIES/2$.
  - Utilizar uma rede **convolucional** de uma única camada e treinar por **muito mais tempo**.
  - Observar **melhor a presença** de um máximo local utilizando o limite superior ($N\_ITEMS * 5$).
  - Modificar os rewards de forma em que o número de passos tenham **maior penalização**. $->$ Diminuir a tendência em máximo locais.

Conclusões

- Possíveis opções de melhoria de desempenho:
  - **diminuir** o número de iterações de *fitting* da rede para menor que $N\_TRAJECTORIES/2$.
  - Utilizar uma rede **convolucional** de uma única camada e treinar por **muito mais tempo**.
  - Observar **melhor a presença** de um máximo local utilizando o limite superior ($N\_ITEMS * 5$).
  - Modificar os rewards de forma em que o número de passos tenham **maior penalização**. $->$ Diminuir a tendência em máximo locais.

Conclusões

- Possíveis opções de melhoria de desempenho:
  - **diminuir** o número de iterações de *fitting* da rede para menor que $N\_TRAJECTORIES/2$.
  - Utilizar uma rede **convolucional** de uma única camada e treinar por **muito mais tempo**.
  - Observar **melhor a presença** de um máximo local utilizando o limite superior ($N\_ITEMS * 5$).
  - Modificar os rewards de forma em que o número de passos tenham **maior penalização**. $->$ Diminuir a tendência em máximo locais.

Conclusões

- Possíveis opções de melhoria de desempenho:
  - **diminuir** o número de iterações de *fitting* da rede para menor que $N\_TRAJECTORIES/2$.
  - Utilizar uma rede **convolucional** de uma única camada e treinar por **muito mais tempo**.
  - Observar **melhor a presença** de um máximo local utilizando o limite superior ($N\_ITEMS * 5$).
  - Modificar os rewards de forma em que o número de passos tenham **maior penalização**. $->$ Diminuir a tendência em máximo locais.

Conclusões

- Possíveis opções de melhoria de desempenho:
  - **diminuir** o número de iterações de *fitting* da rede para menor que $N\_TRAJECTORIES/2$.
  - Utilizar uma rede **convolucional** de uma única camada e treinar por **muito mais tempo**.
  - Observar **melhor a presença** de um máximo local utilizando o limite superior ($N\_ITEMS * 5$).
  - Modificar os rewards de forma em que o número de passos tenham **maior penalização**. $->$ Diminuir a tendência em máximo locais.

Tentativas de Otimização

- Mudar RGB para Monocromático $->$ **Não** houve aprendizado.
- Utilizar redes muito profundas. $->$ Aumentou **muito** o tempo de treino e a **variância**.
- Tentou-se utilizar uma única época de *fitting* para treinar a rede. $->$ Não houve aprendizado.
- Acúmulo de experiências em trajetória $->$ **Máximo local**.

Tentativas de Otimização

- Mudar RGB para Monocromático $->$ **Não** houve aprendizado.
- Utilizar redes muito profundas. $->$ Aumentou **muito** o tempo de treino e a **variância**.
- Tentou-se utilizar uma única época de *fitting* para treinar a rede. $->$ Não houve aprendizado.
- Acúmulo de experiências em trajetória $->$ **Máximo local**.

Tentativas de Otimização

- Mudar RGB para Monocromático $->$ **Não** houve aprendizado.
- Utilizar redes muito profundas. $->$ Aumentou **muito** o tempo de treino e a **variância**.
- Tentou-se utilizar uma única época de *fitting* para treinar a rede. $->$ Não houve aprendizado.
- Acúmulo de experiências em trajetória $->$ **Máximo local**.

Tentativas de Otimização

- Mudar RGB para Monocromático $->$ **Não** houve aprendizado.
- Utilizar redes muito profundas. $->$ Aumentou **muito** o tempo de treino e a **variânci**a.
- Tentou-se utilizar uma única época de *fitting* para treinar a rede. $->$ Não houve aprendizado.
- Acúmulo de experiências em trajetória $->$ **Máximo local**.

Tentativas de Otimização

- Mudar RGB para Monocromático $->$ **Não** houve aprendizado.
- Utilizar redes muito profundas. $->$ Aumentou **muito** o tempo de treino e a **variânci**a.
- Tentou-se utilizar uma única época de *fitting* para treinar a rede. $->$ Não houve aprendizado.
- Acúmulo de experiências em trajetória $->$ **Máximo local**.

Referências

Anjun Dai, Elias B. Khalil, Y. Z. B. D. L. S. (2017).

Learning Combinatorial Optimization Algorithms over Graphs.

ICLR 2017.

Clark, J. (2016).

The skynet salesman.

http://multithreaded.stitchfix.com/blog/2016/07/21/skynet-salesman/.

Dirko Coetsee (2017).

A simple policy gradient implementation with keras.

http://dirko.github.io/Keras-policy-gradient/.

Referências

Irwan Bello, Hieu Pham, Q. V. L. M. N. S. B. (2017).

Neural Combinatorial Optimization with Reinforcement Learning.

ICLR 2017.

Karpathy, A. (2016).

Deep reinforcement learning: Pong from pixels.

`http://karpathy.github.io/2016/05/31/rl/`.