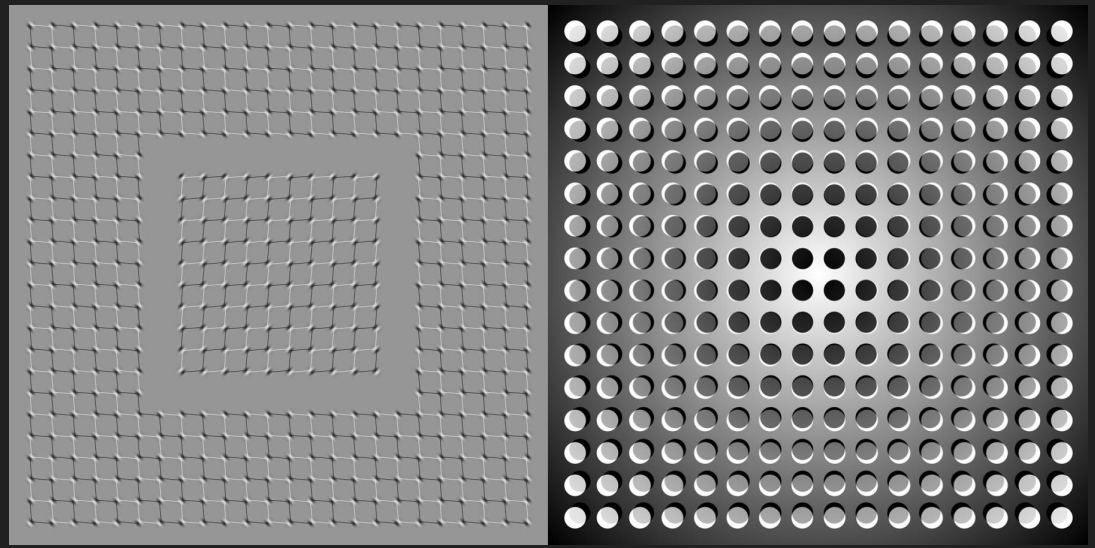
Modelos del mundo - ¿Un programa puede aprender a partir de sus sueños?

https://twitter.com/osanseviero
www.linkedin.com/in/omarsanseviero/

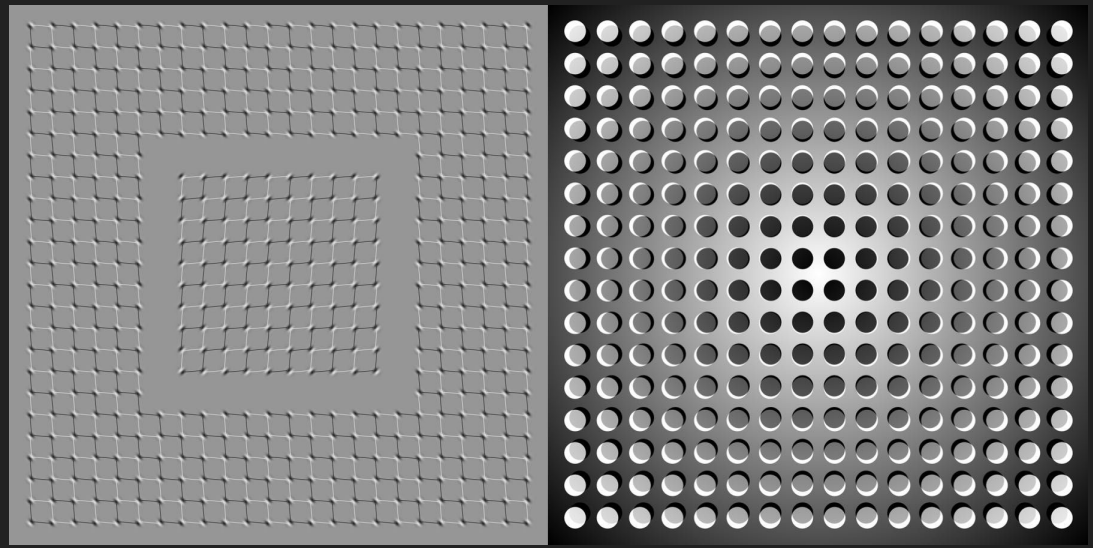
Modelos del mundo

- Los humanos tenemos un modelo mental del mundo
  - Basado en lo que hemos vivido
  - Conceptos y relaciones entre ellos



Modelos del mundo

- Para manejar tanta información, nuestro cerebro aprende representaciones abstractas de la información espacial y temporal.
- Lo que percibimos se basa en lo que nuestro cerebro predice para el futuro.



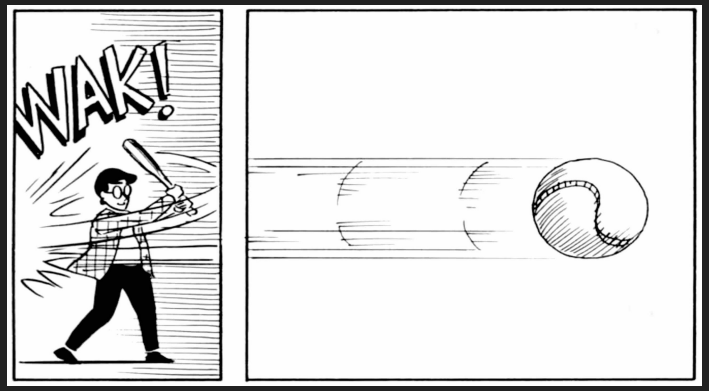
Modelo del cerebro

- No predice el futuro en general, predice información sensorial basándose en el input motor actual.

  - ¿Cómo bateamos en milisegundos?

Modelo del cerebro

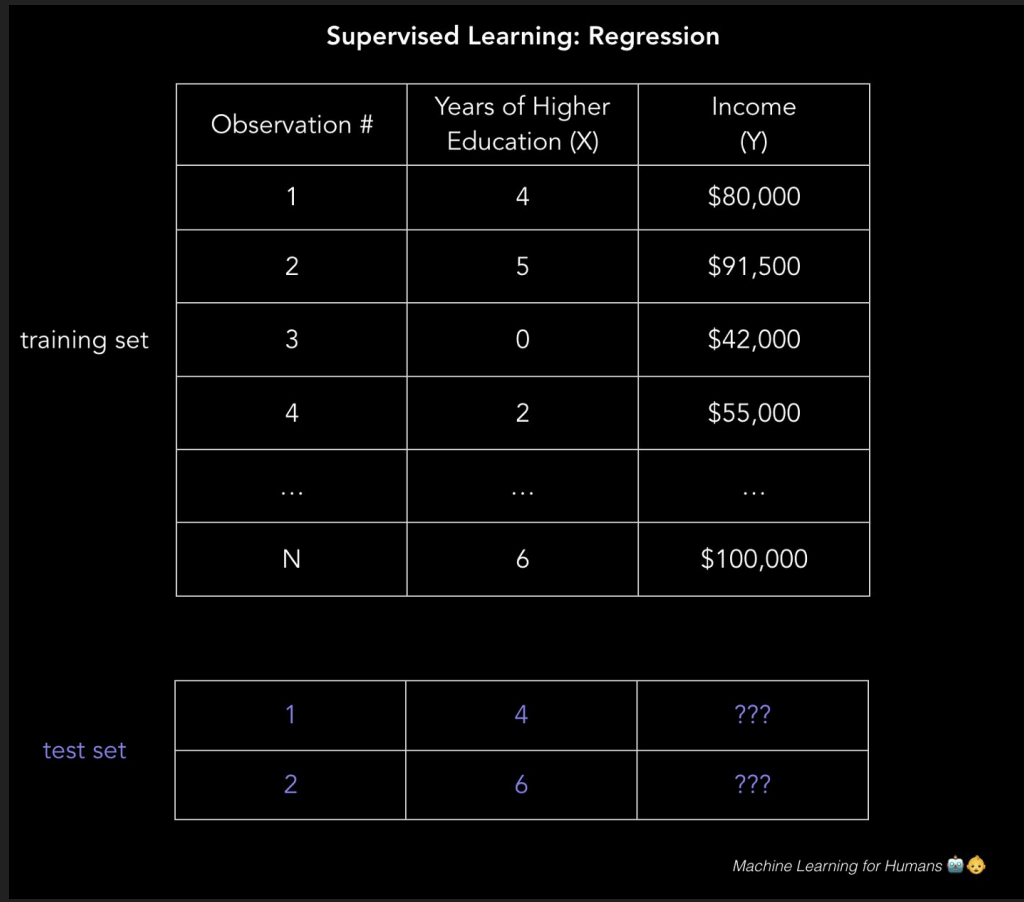
- No predice el futuro en general, predice información sensorial basándose en el input motor actual.



- Predecimos con instinto dónde y cuándo irá la pelota
- Pasa subconscientemente para profesionales
- Reaccionan gracias a las predicciones internas del modelo

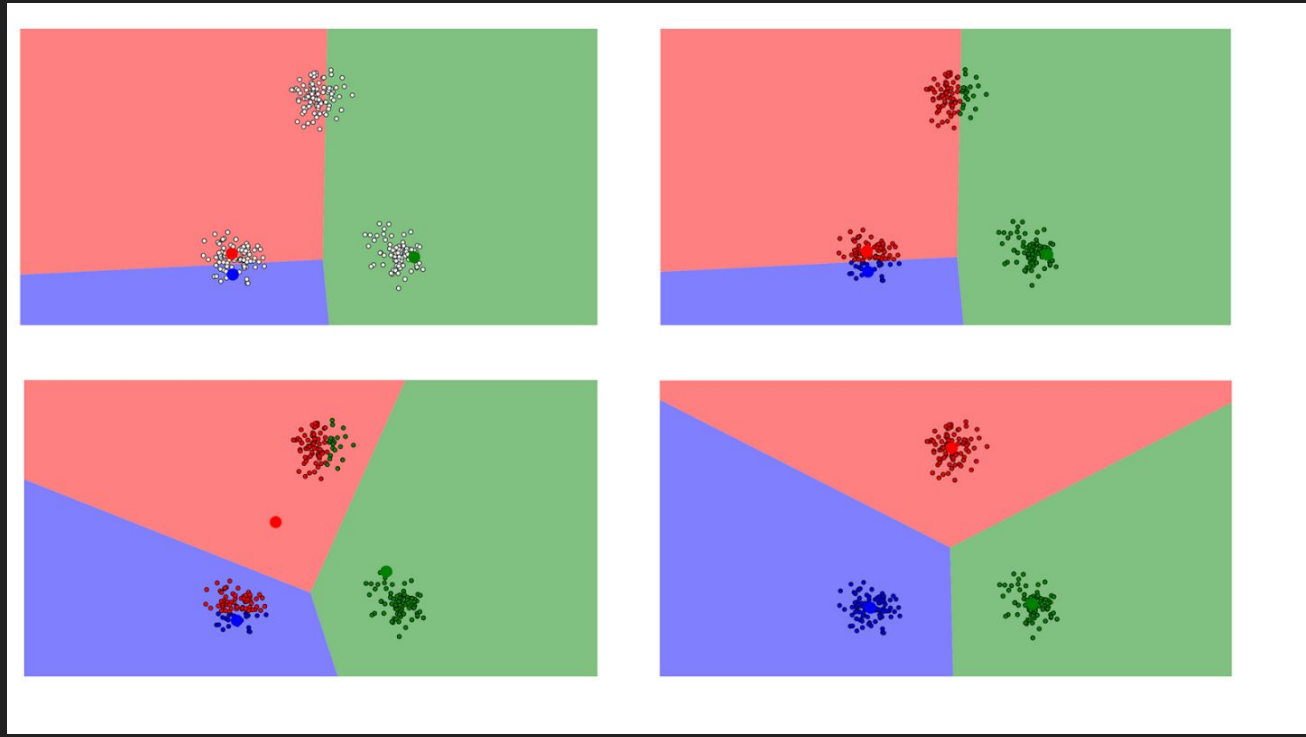
Tipos de Aprendizaje

- Supervisado



Tipos de Aprendizaje

- Supervisado
- No Supervisado



Tipos de Aprendizaje

- Supervisado
- No Supervisado
- Aprendizaje por Refuerzo

Reinforcement Learning

- Aprendizaje a partir de interacciones
- Utilizamos ambientes con reglas definidas
    - Agente aprende a interactuar en el ambiente

- Mucha relación con psicología y neurociencia
- El más parecido a aprendizaje humano

Aplicaciones

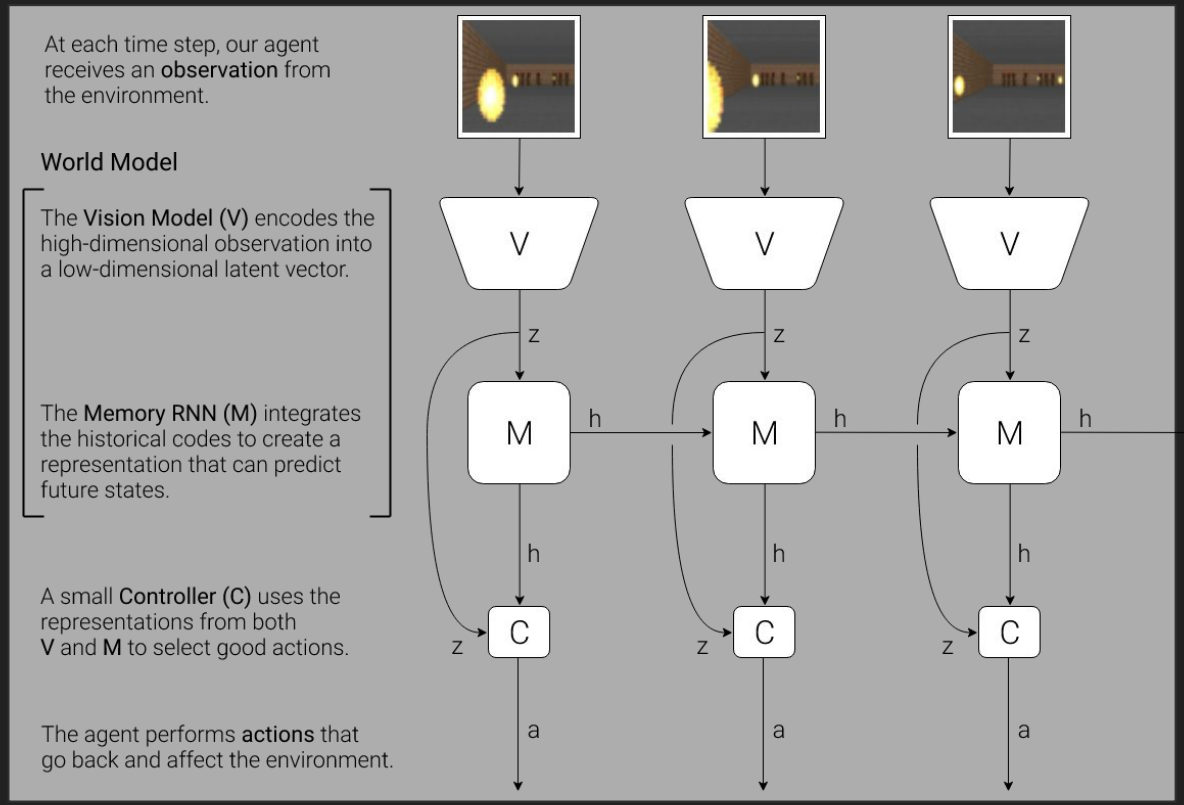
- Coches autónomos (https://selfdrivingcars.mit.edu/)
- Juegos
  - Backgammon (90s)
    - Se descubrieron nuevas estrategias
    - 10^{20} estados posibles
  - AlphaGo
  - Atari (Deep Reinforcement Learning)
  - Dota
- Robótica
  - Enseñar a caminar
  - Aviones
  - Barcos
- Muchos otros campos

RL para un robot humanoide

Reinforcement Learning

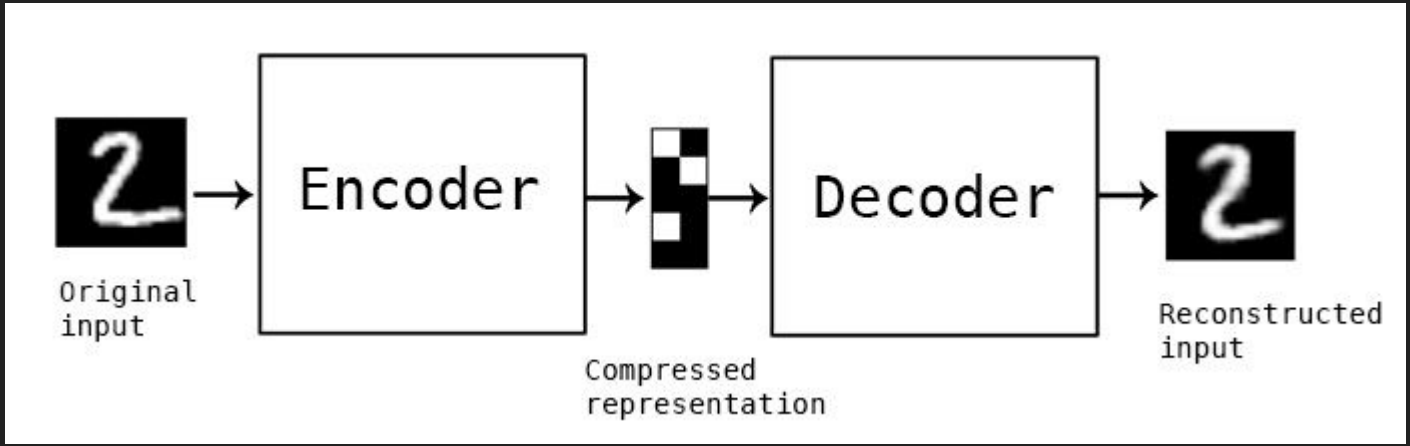
- Un agente se beneficia de tener una buena representación de estados del pasado y del presente.
- Necesitamos un modelo poderoso que pueda predecir en el tiempo
- Podemos utilizar Recurrent Neural Networks (RNNs)
  - Modelos muy expresivos de información temporal
  - No se usa tanto en RL - se prefieren redes neuronales sencillas para entrenar rápidamente
    - Costo en algoritmos tradicionales es muy alto
- Objetivo: Crear un agente de RL con RNNs

Modelo del agente



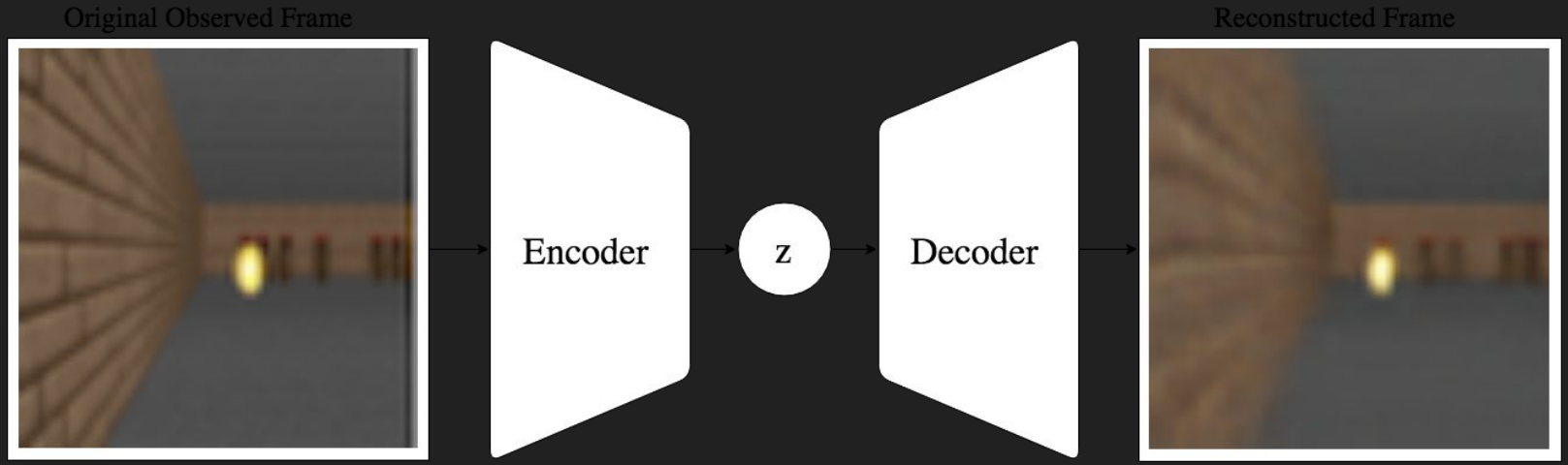
Autoencoder

- Modelos usados para comprimir la información
- Se crea una modelo fino que permite representar la información de manera más sencilla.

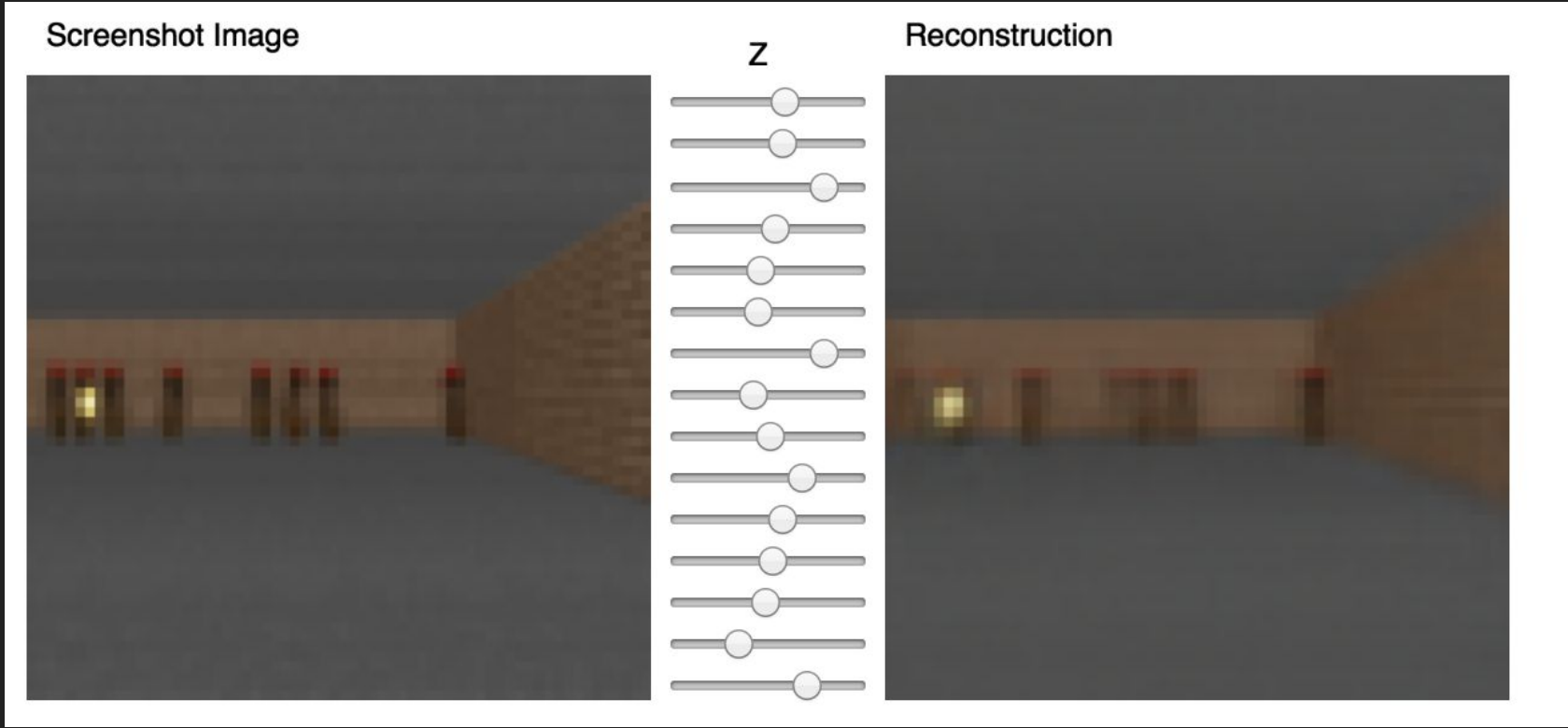


Modelo VAE (Visión)

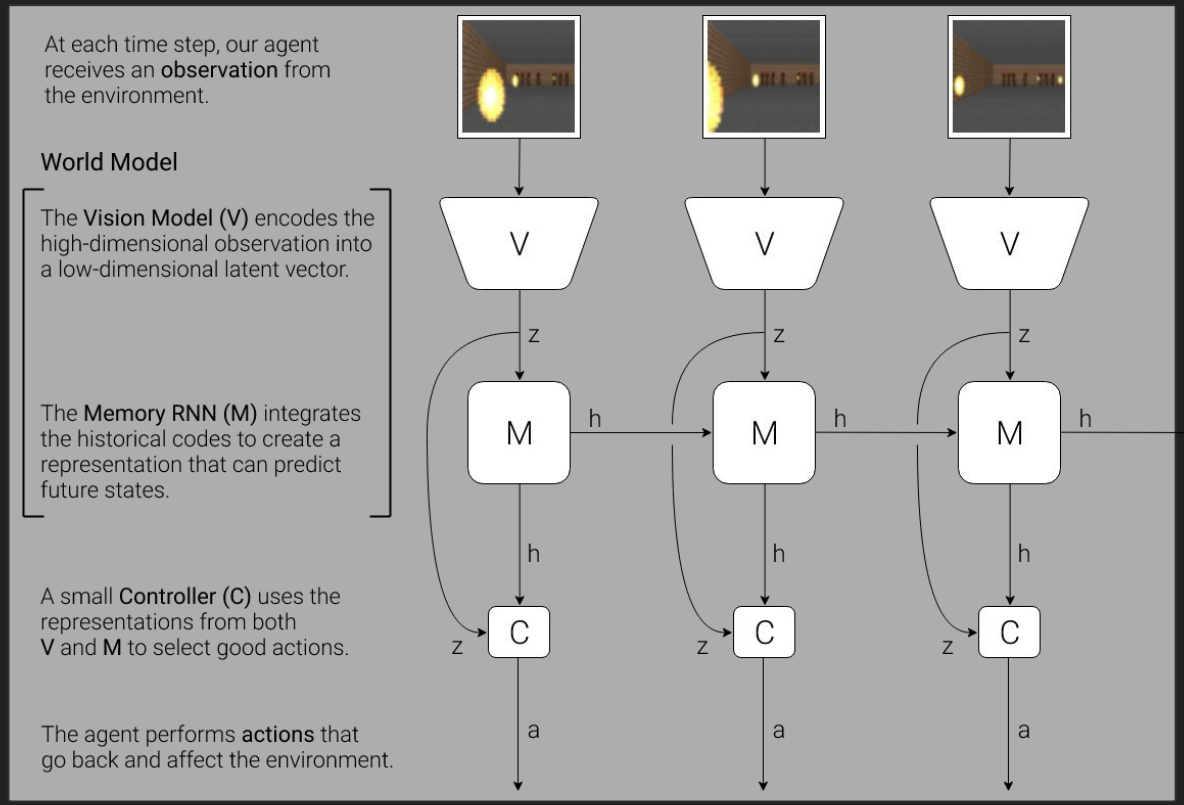
- El ambiente da al agente un input de muchas dimensiones
  - Ejemplo: Un frame 2D de un video, pero en secuencia
- Objetivo: Aprender representación abstracta de los frames observados
- Usamos autoencoders (Variational Autoencoders - 2013, 2014)



Modelo VAE (Visión)

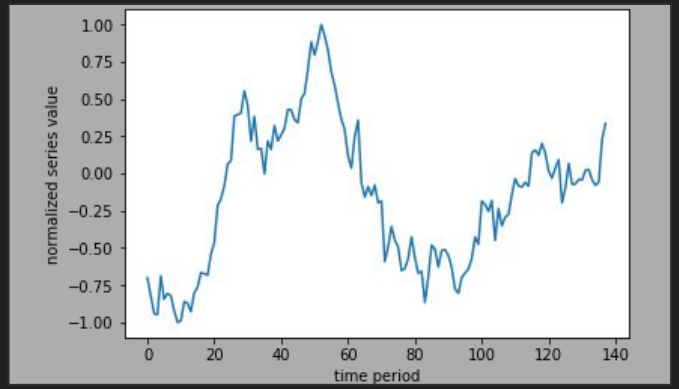


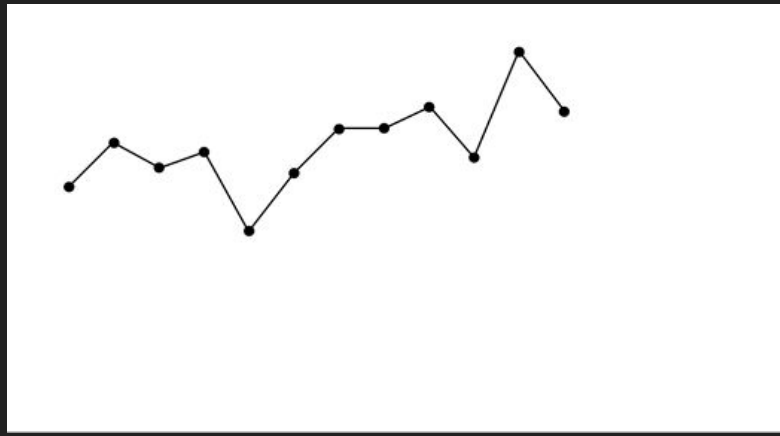
Modelo del agente



RNN

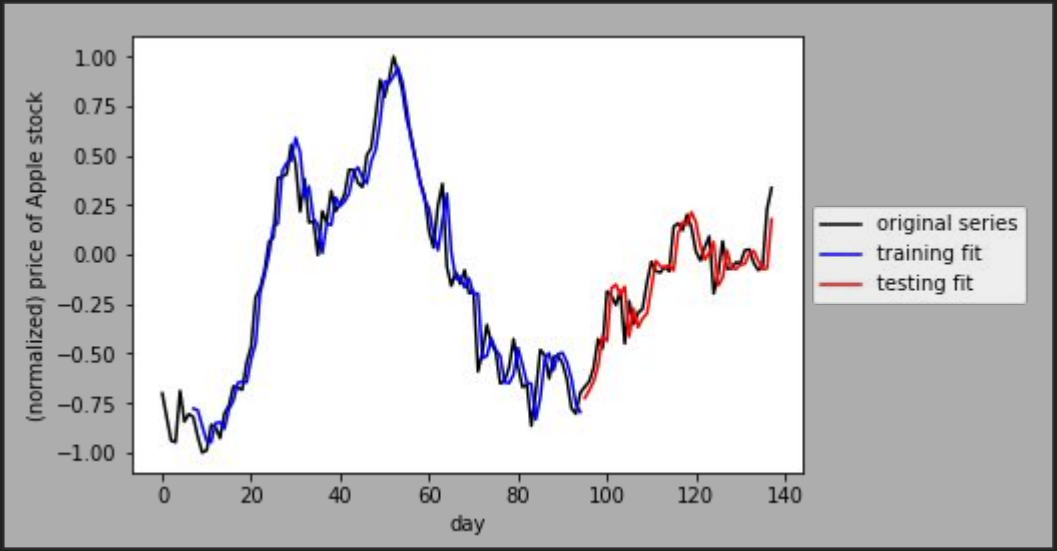
- Tienen memoria
- Dependencia sobre el tiempo
- No sólo usamos el input actual, sino el input previo





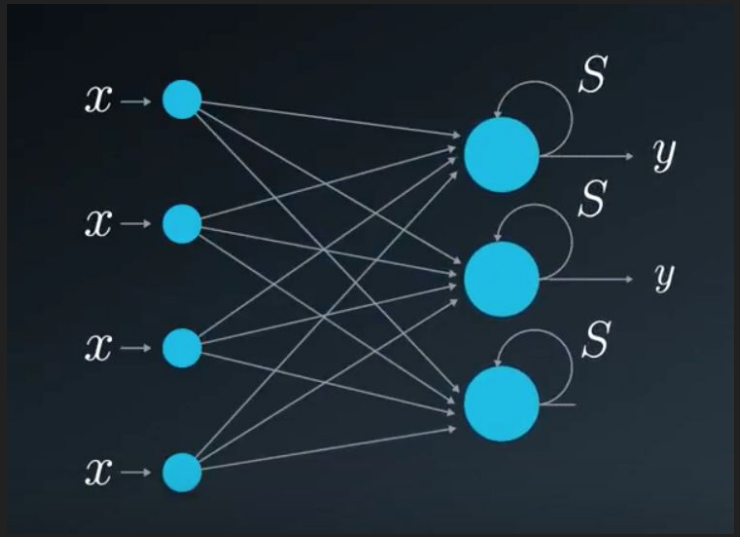
- Ejemplos:
  - Completar secuencias
  - Distinguir acciones de videos

RNN

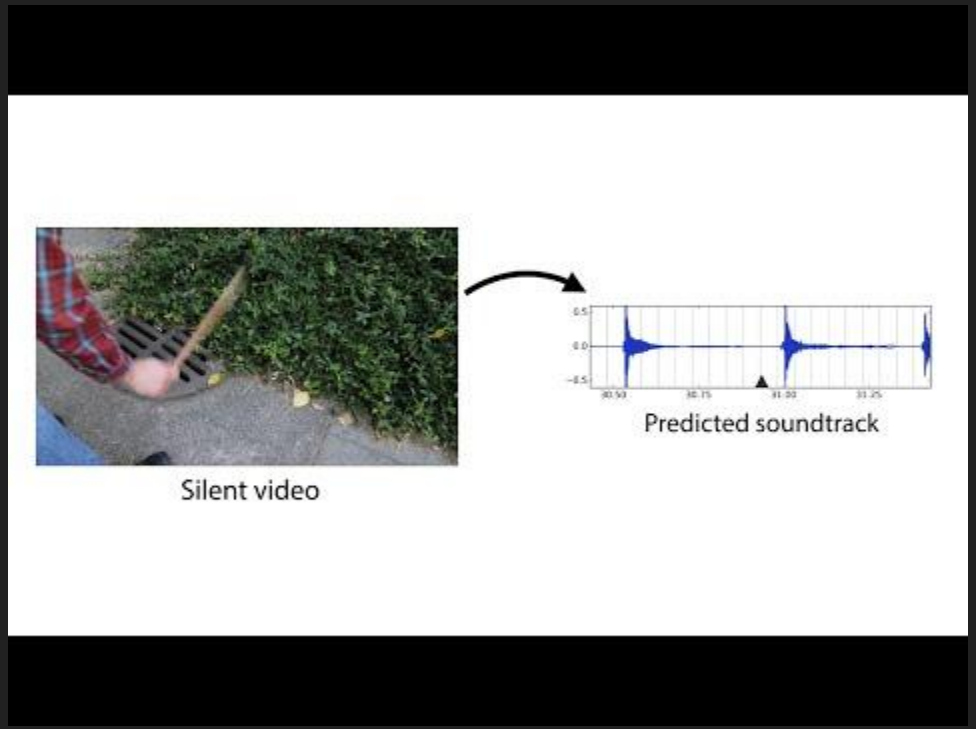

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
(normalized) price of Apple stock
0
20
40
60
80
100
120
140
day
original series
training fit
testing fit

RNN

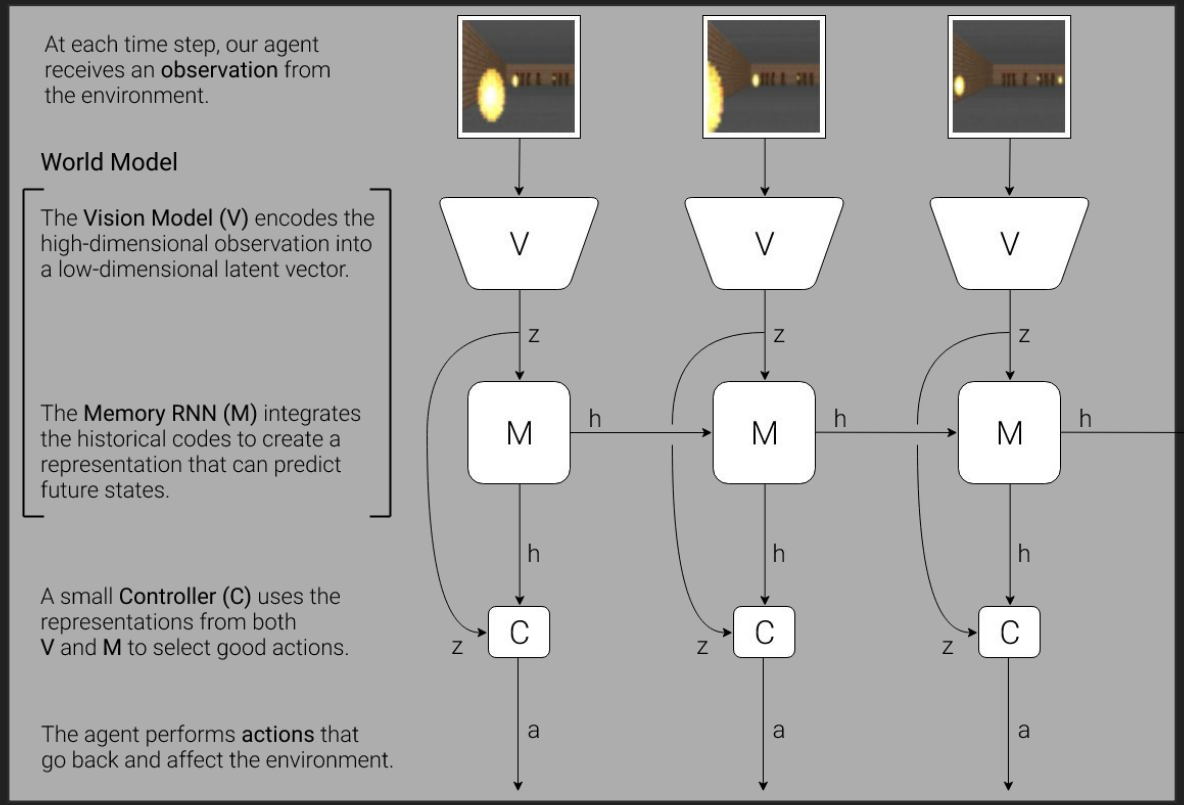
- Recurrent: Ocurre repetidamente
- Hacemos la misma tarea en la secuencia de entrada múltiples veces



Aplicaciones - Agregando Sonido

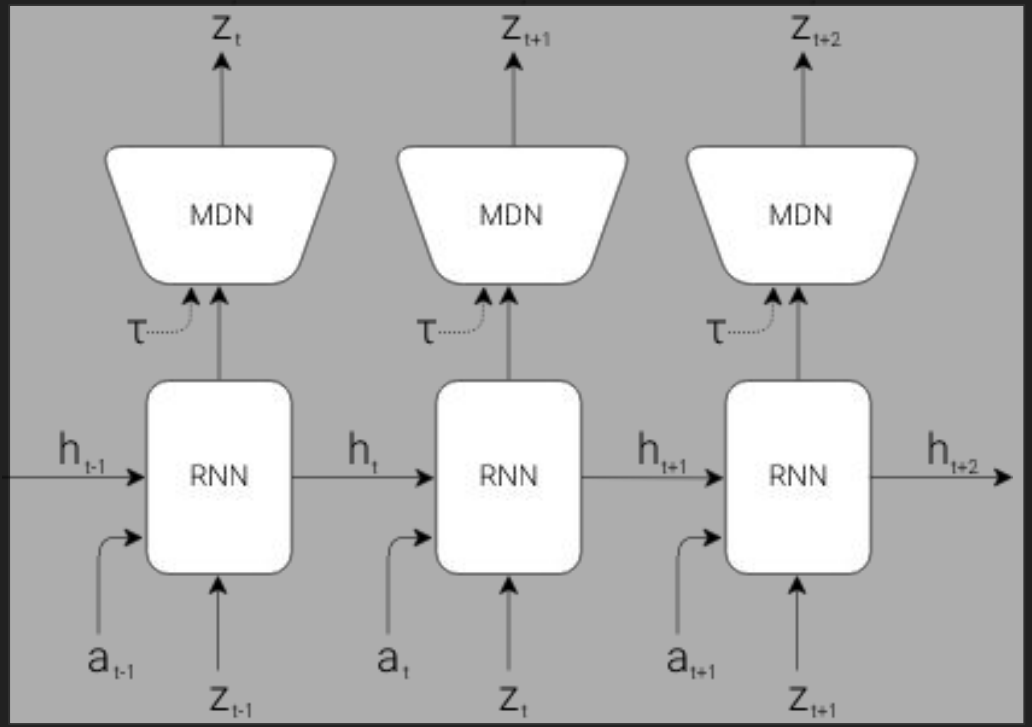


Modelo del agente



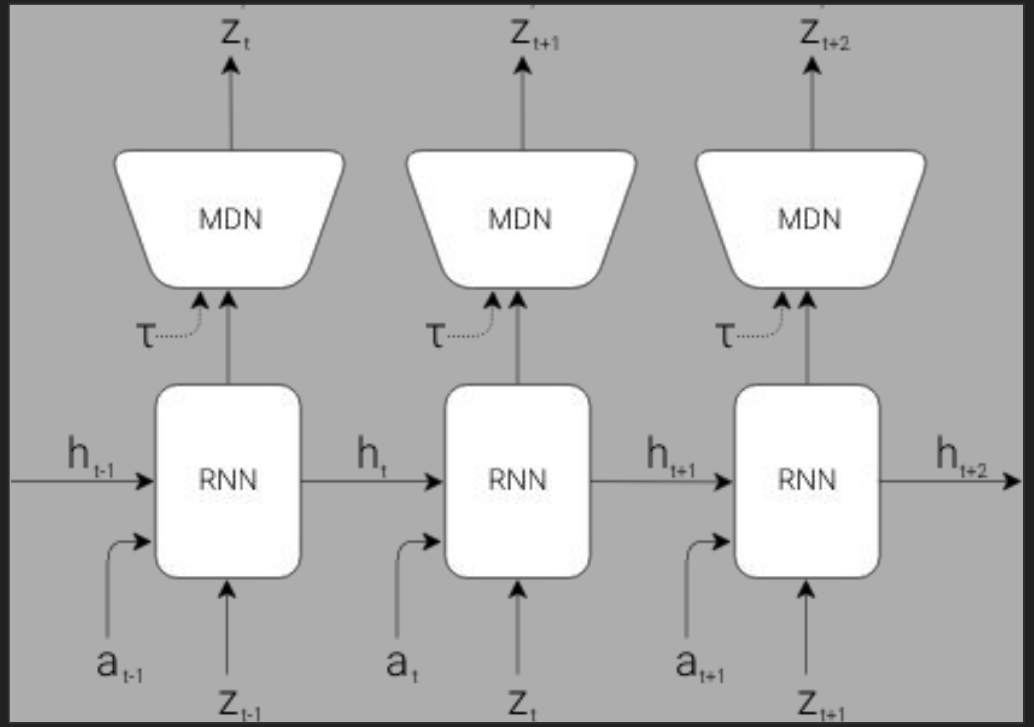
Modelo MDN-RNN (Memoria)

- Comprime lo que ocurre en el tiempo
- Modelo que predice el futuro
- Predice vectores z que espera que V produzca



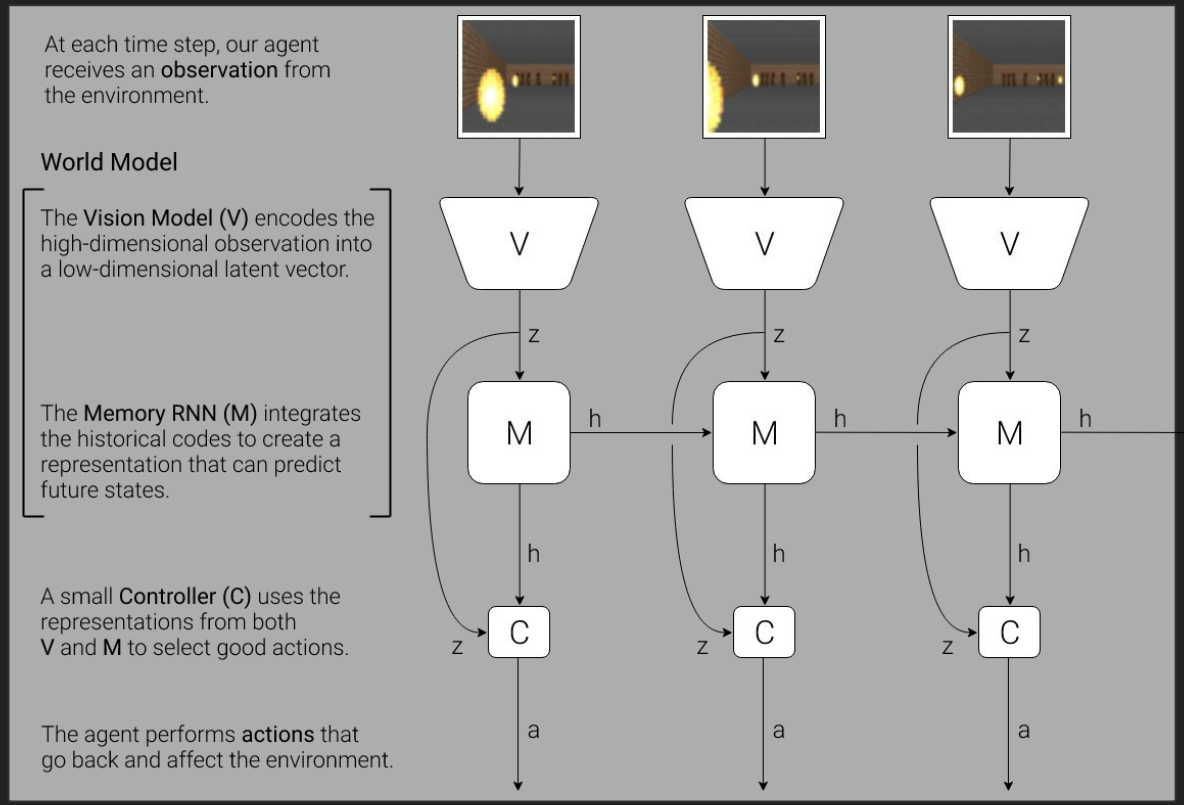
Modelo MDN-RNN (Memoria)

- Lo manejamos como una densidad de probabilidades en vez de un modelo determinístico
- Parámetro t es la temperatura - se ajusta para cambiar incertidumbre del modelo



MDN-RNN - Mixture Density Network RNN

Modelo del agente

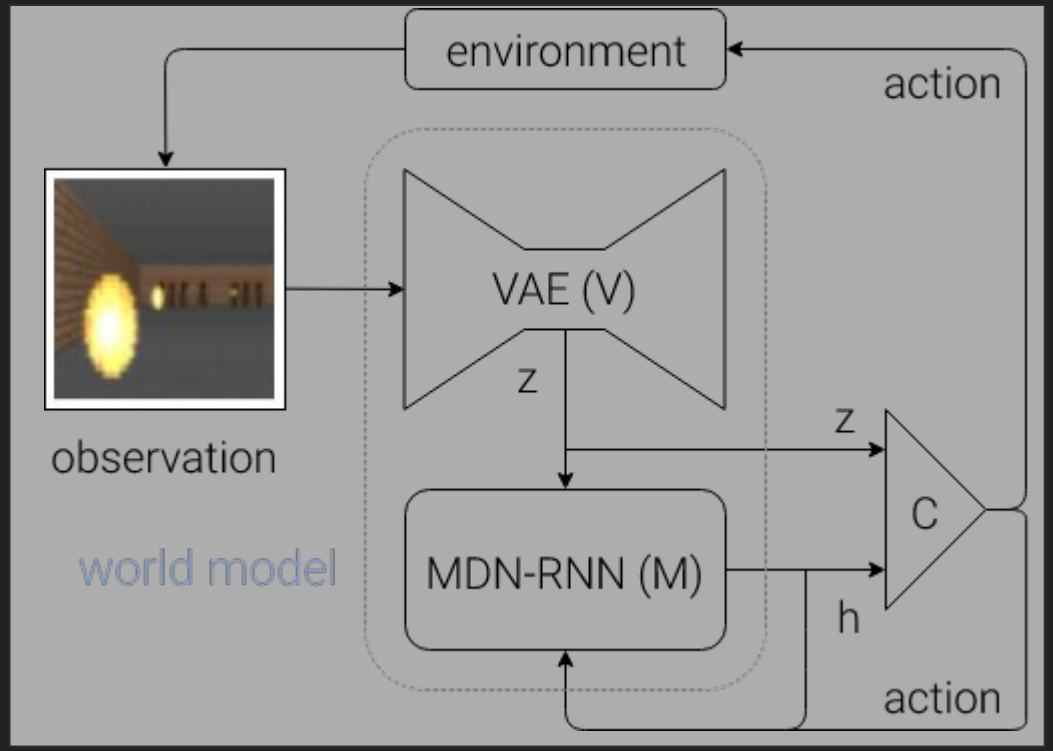


Modelo Controlador (C)

- Determina acciones para maximizar recompensa
- Simple y pequeño

- Pocos parámetros - la complejidad está en V y M
  - V y M están entrenados con backpropagation
  - C se puede entrenar de otras maneras, como algoritmos evolutivos
    - Convariance-Matrix Adaptation Evolution Strategy (CMA-ES)
      - Funciona bien en espacios con pocos parámetros

Combinando todo...




```
def rollout(controller):
  ''' env, rnn, vae are '''
  ''' global variables  '''
  obs = env.reset()
  h = rnn.initial_state()
  done = False
  cumulative_reward = 0
  while not done:
    z = vae.encode(obs)
    a = controller.action([z, h])
    obs, reward, done = env.step(a)
    cumulative_reward += reward
    h = rnn.forward([a, z, h])
  return cumulative_reward
```

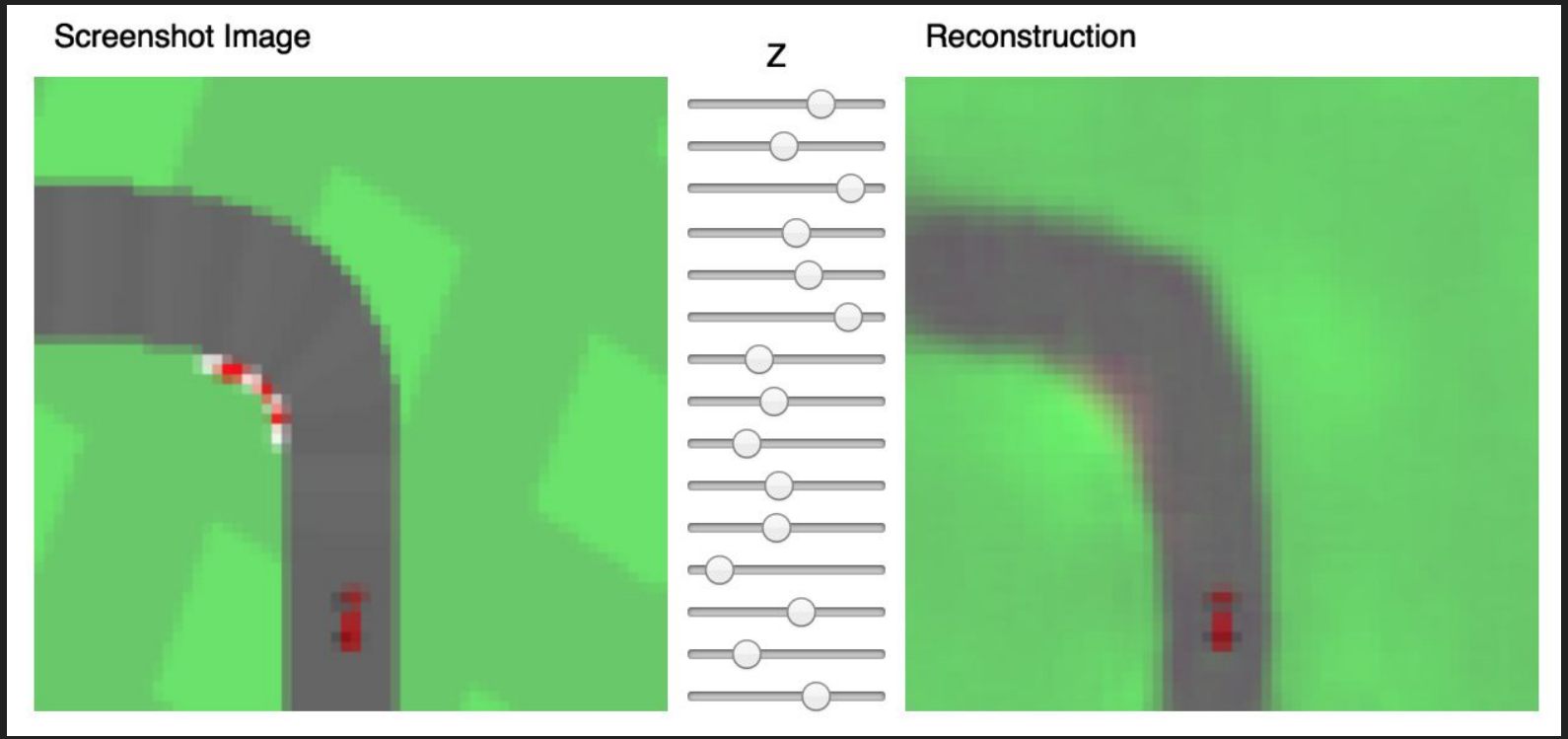
Experimento 1. Carrera de coches

Experimento 1

- Crear un modelo predictivo del mundo
  - Representar tiempo y espacio
- Tarea de control continua

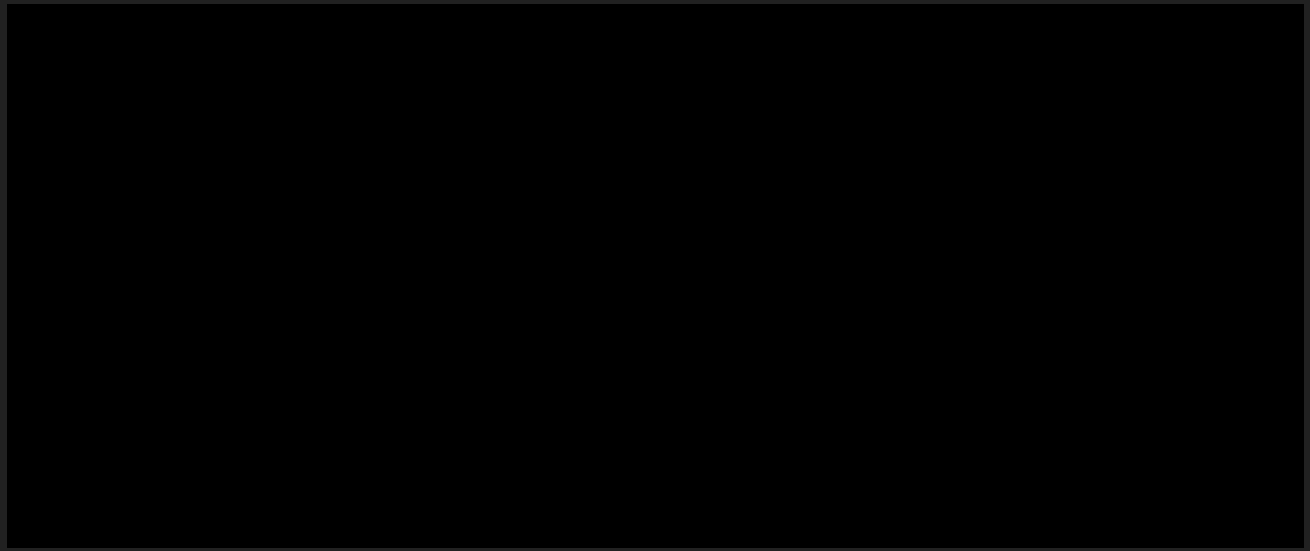
- Ambiente genera mapa aleatoriamente
- Recompensa es número de lugares visitado
- Acciones: girar, acelerar, frenar

E1. Entrenamos V



E1. Resultado

- MDN-RNN se entrena con resultado de V
- Sólo C sabe la recompensa
  - Sólo hay 867 parámetros

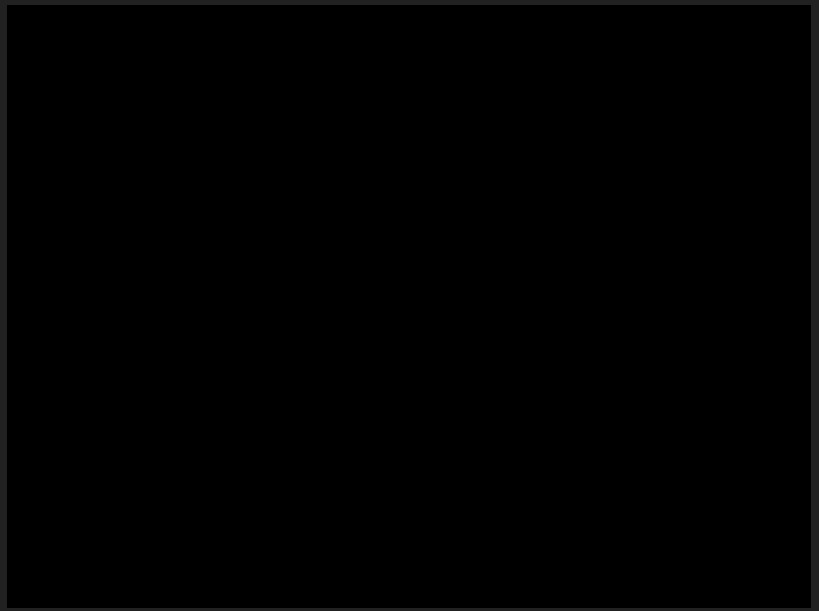


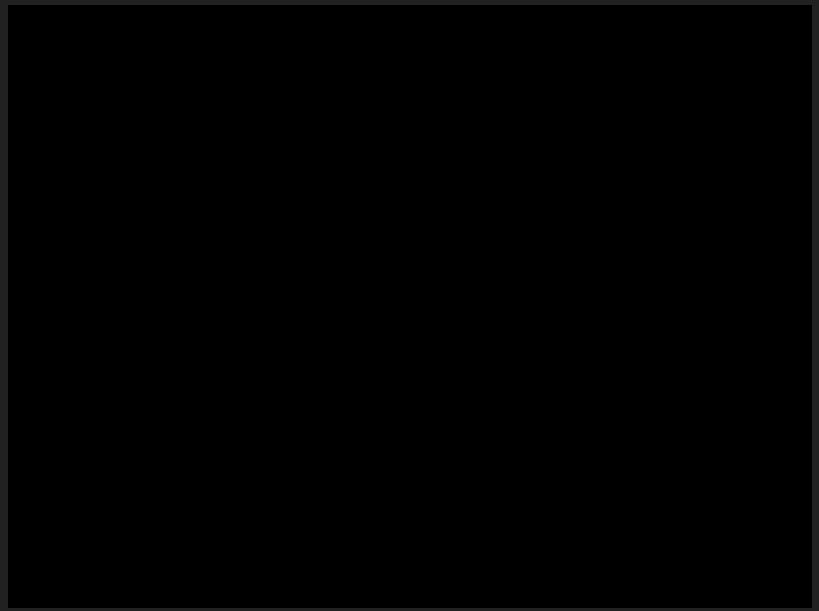
E1. Procedimiento

1. Se generan 10mil corridas con política aleatoria
2. Se entrena VAE para codificar cada frame
3. Se entrena MDN-RNN para dar distribución de probabilidades
4. Definimos un controlador
5. Usamos CMA-ES para maximizar recompensa

Model	Parameter Count
VAE	4,348,547
MDN-RNN	422,368
Controller	867

V vs V con M



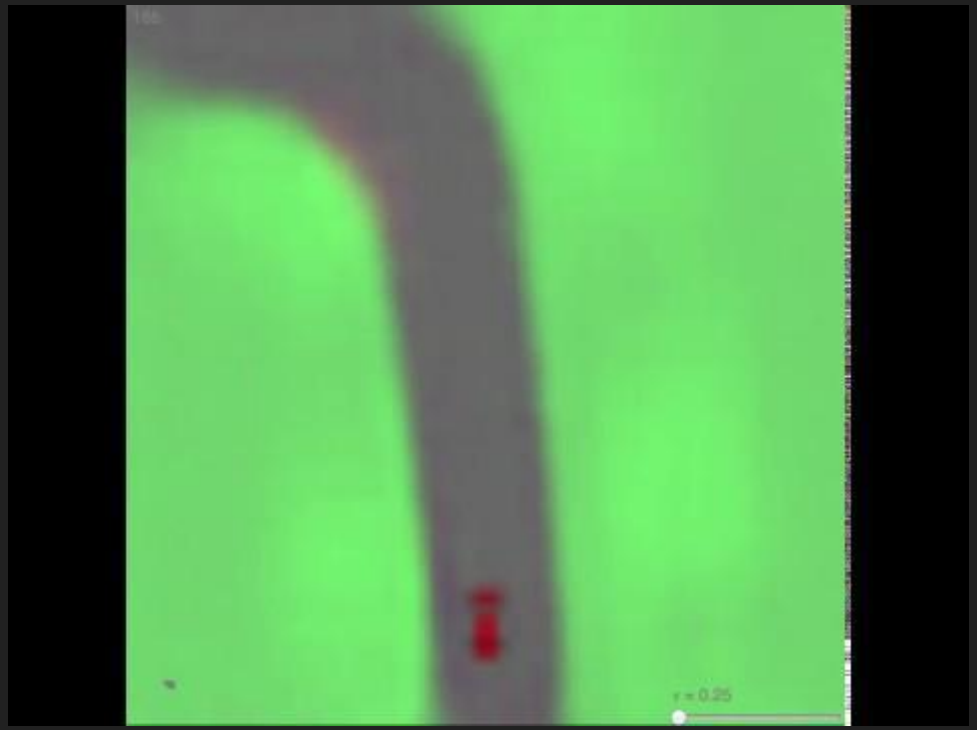


Comparando

Method	Average Score over 100 Random Tracks
DQN [54]	343 ± 18
A3C (continuous) [53]	591 ± 45
A3C (discrete) [52]	652 ± 10
ceobillionaire's algorithm (unpublished) [48]	838 ± 11
V model only, z input	632 ± 251
V model only, z input with a hidden layer	788 ± 141
Full World Model, z and h	$\mathbf{906 \pm 21}$

Soñando carreras de coches

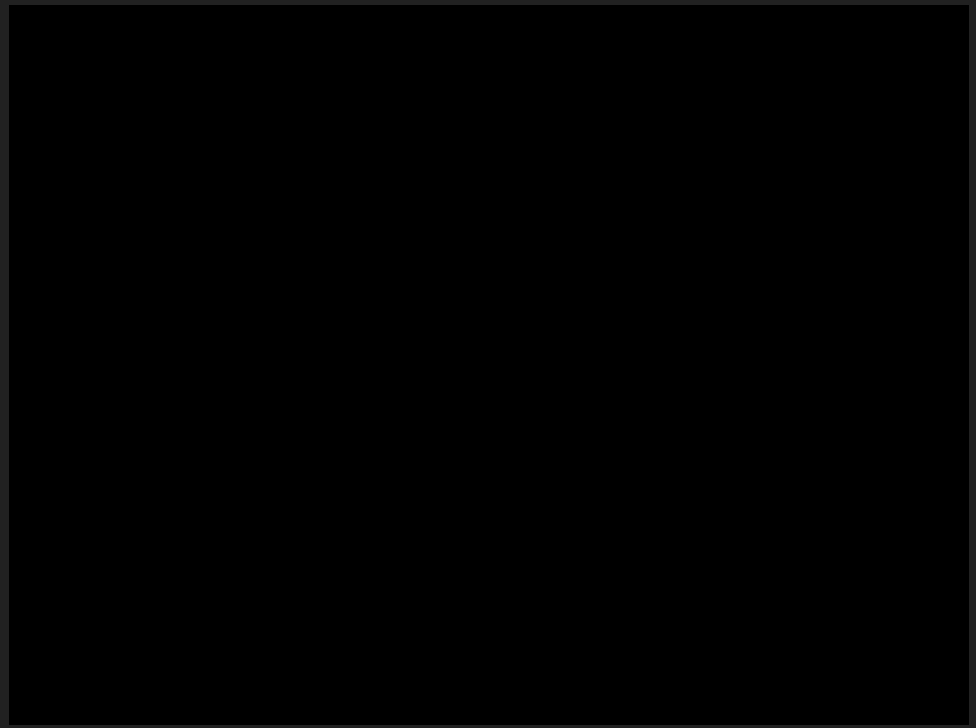
- Como el modelo puede predecir el futuro, puede halucinar carreras hipotéticas.
- Podemos pedir al MDN-RNN que genere z.
- Ponemos nuestro controlador entrenado en el sueño.



Soñando carreras de coches

- Podemos aplicar lo que hemos aprendido en el mundo real en el sueño.
- ¿Podemos aprender en el sueño y usar eso en el mundo real?
  - Transferir la política aprendida en el sueño al mundo real

Experimento 2. VizDoom

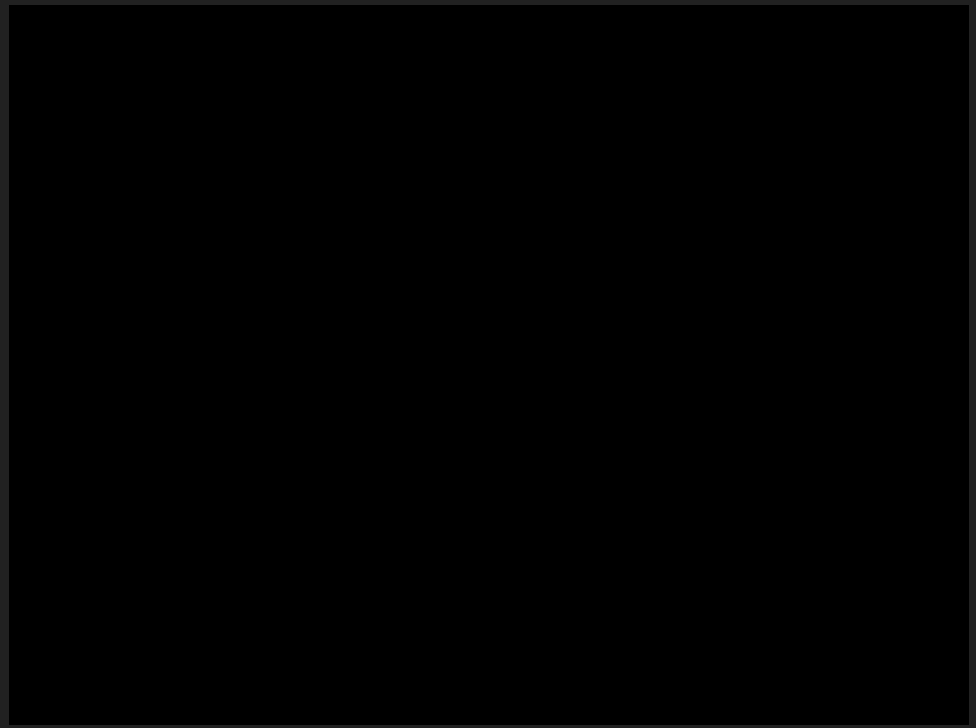


Entrenando en el sueño

¿Qué pasa?

- VAE es entrenado con información aleatoria
- Luego, podemos ir al sueño, donde el RNN va generando (o prediciendo) el mundo
  - Aprendió a simular aspectos del juego - lógica, comportamiento de enemigos, física y el rendering en 3D
  - Si el agente envía acción a la izquierda, M sabe cómo cambiar los pixeles. También aprende a que no pase las paredes
- A diferencia del ambiente verdadero, podemos agregar incertidumbre, haciendo el juego más difícil en el sueño
  - Si el agente se desempeña bien con alta temperatura, generalmente desempeñará bien en el ambiente real
  - Aumentar temperatura previene que agente tome ventaja de errores del mundo real (exploits del juego)

Transferimos la política (policy transfer)



Agente aprende a hacer trampa

Cómo evitamos eso

- Temperatura da incertidumbre. El modelo M es sólo una representación imperfecta abstracta del mundo real.
- Por lo tanto, en M, las bolas de fuego no siempre seguirán las trayectorias correctas. Esto hace que se generen exploits en M que no se aplican en el mundo real.
- Es esto que ha hecho que muchas técnicas de RL no se apliquen en la vida real.
- MDN-RNN, como es distribución de probabilidades, previene esto.
    - Cambiar la temperatura nos permite cambiar entre explotabilidad y realismo.

Temperature	Score in Virtual Environment	Score in Actual Environment
0.10	2086 ± 140	193 ± 58
0.50	2060 ± 277	196 ± 50
1.00	1145 ± 690	868 ± 511
1.15	918 ± 546	1092 ± 556
1.30	732 ± 269	753 ± 139
Random Policy Baseline	N/A	210 ± 108
Gym Leaderboard [34]	N/A	820 ± 58

¡Gracias!

https://twitter.com/osanseviero
www.linkedin.com/in/omarsanseviero/