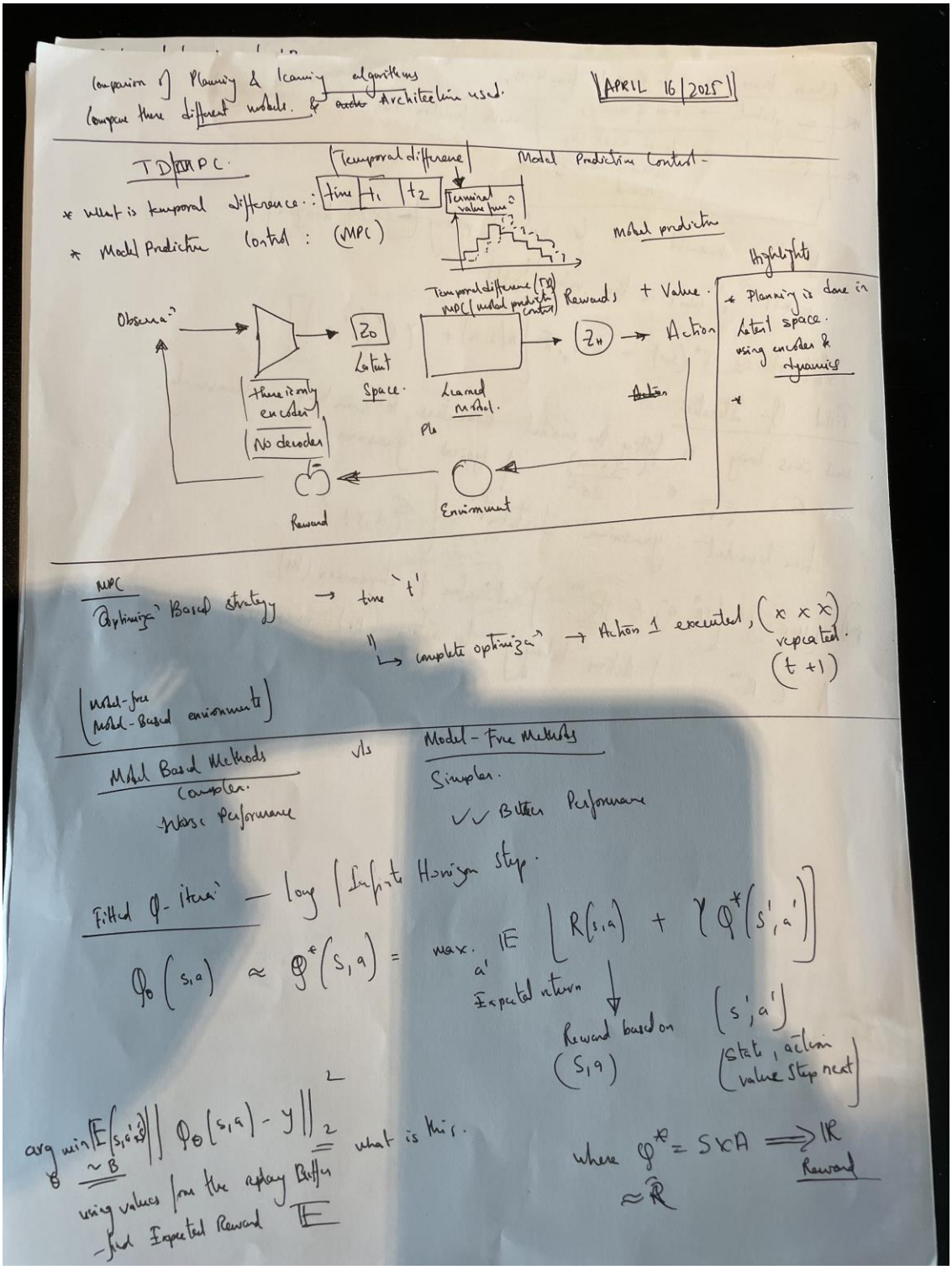
Notes:


Comparison of Planning & learning algorithms
Compare three different models & encoder Architecture used.
APRIL 16 2025
TD MPC.
Temporal difference
Model Prediction Control
* what is temporal difference :
time t1 t2
Terminal value func?
* Model Prediction Control : (MPC)
Model prediction
Highlights
* Planning is done in latent space. using encoder & dynamics
Temporal difference (TD) MPC (model prediction control)
Rewards + Value
Observation
Zo
Latent Space.
Zt
Action
There is only encoder
No decoder
Learned model.
Reward
Environment
MPC
Optimization Based strategy → time 't'
→ complete optimization → Action 1 executed, (x x x) repeated. (t+1)
Model-free
Model-Based environments
Model Based Methods
Complex.
Worse Performance
vs
Model-Free Methods
Simpler.
✓✓ Better Performance
Fitted Q-iteration — long Infinite Horizon step.
Qo(s,a) ≈ Q*(s,a) = max_a' E [R(s,a) + γ Q*(s',a')]
Expected return
Reward based on (s,a)
(s',a') (state, action value step next)
avg min E_{s,a,s'~B} || Qθ(s,a) − y ||_2^2 = what is this.
using values from the replay Buffer
Find Expected Reward E
where Q* = SKA ⟹ R Reward
≈ R̂

APRIL 17/2025

* contrastive models
vs generative models

Model of env.
↓ sample seq.
sample action

TD-learning
$Q_\theta(s,a)$.

1. (Learn model of environment) → (2. Sample Actions) | Evaluates trajectories using the learnt model.

Sample, Observ, Rewards, Actions ↔ Replay BUFFER

sample

enc

Latent Space → $\hat{a}$, $\hat{r}$ ground truth reward | single step reward., q, value TD learning.

q. function: Value func^n → for all episodes
$q(s_t, a_t) = q_{t+1}$ future return | Σ all state
→ dynamics $(s_t, a_t) = s_{t+1}$
→ reward $(s_t, a_t) = r_{t+1}$ | reward at t+1
not States but latent representation
→ $\mathcal{E}(s_t) = z_t$
→ $T(s_t) \rightarrow a_{t+1}$

Latent model vs ground truth.

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}} \Big[\sum_{t=0}^{H} \lambda^t \Big(\| z_t' - sg(h(s_t^+)) \|_2^2 + CE(\hat{r}_t, r_t) + CE(\hat{q}_t, q_t) \Big) \Big]$$

predictor - ground truth reward
predict. one step td target
q. target.

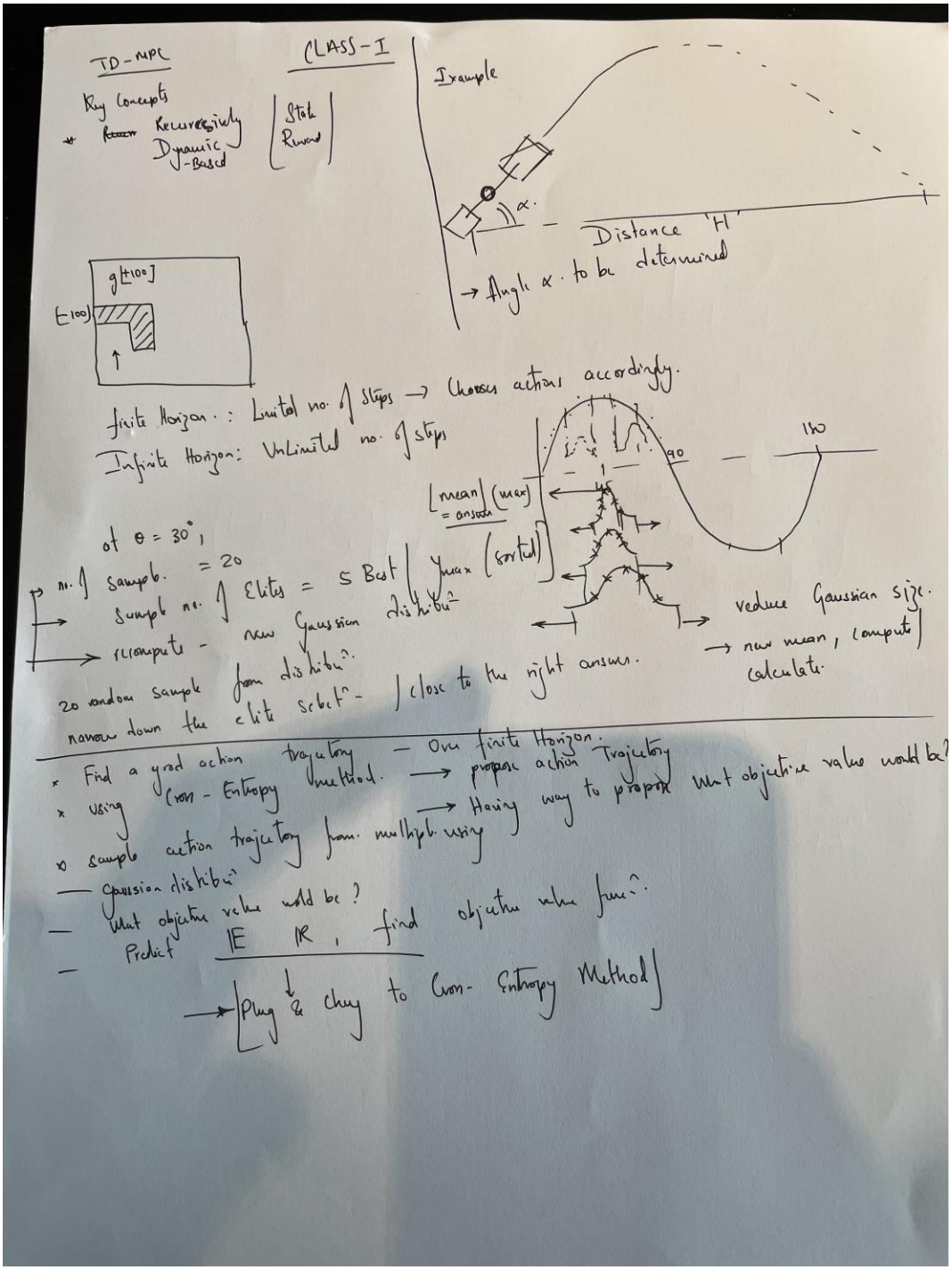
No decoder

Online KL | Offline KL
High dimensional prob.
data → policy — fast learning.
TDMPC-2 (100 different tasks) [100+ tasks]

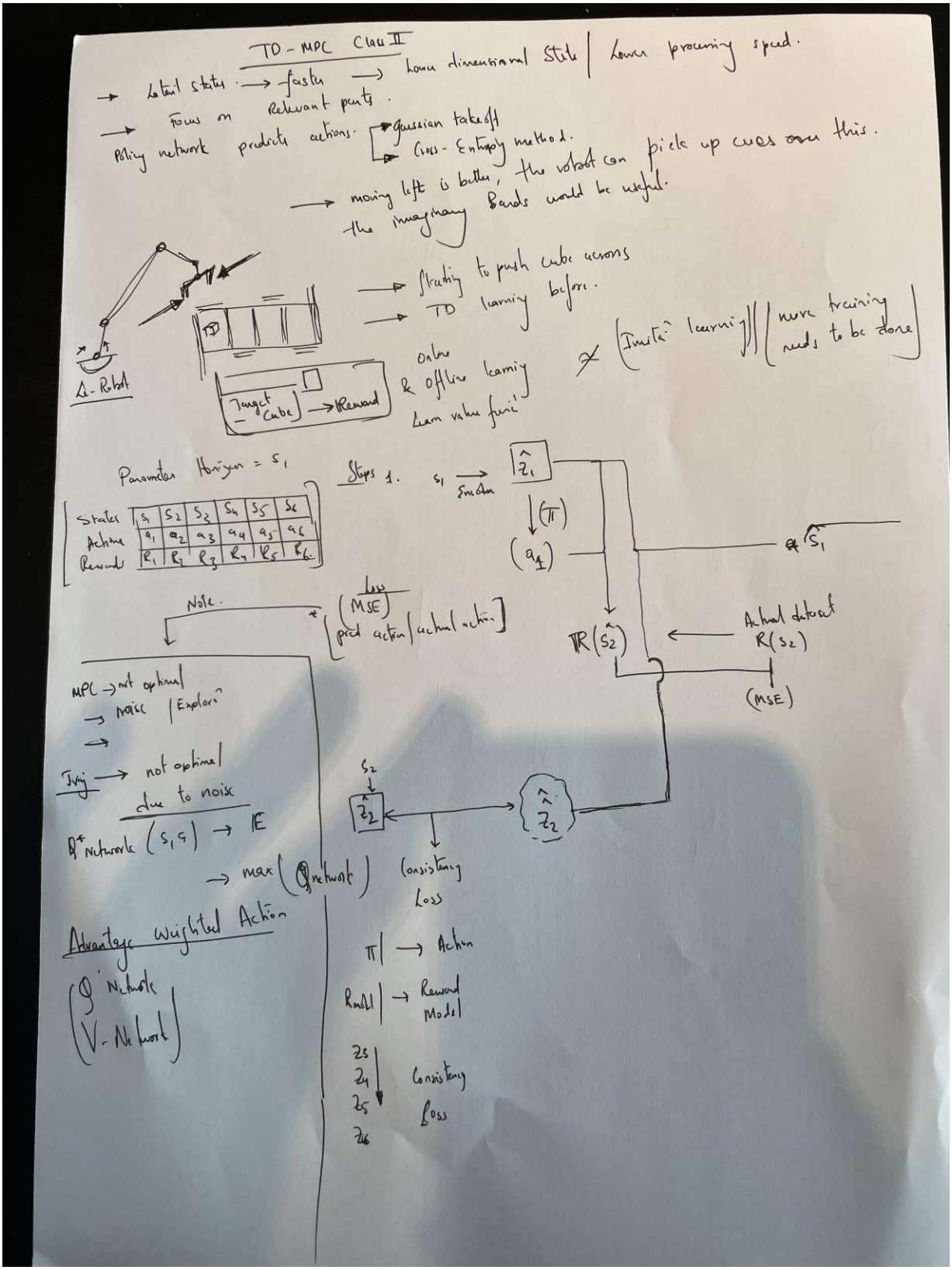
Algorithm
Abl^n → removing parts / specific
Observe → how parts affect the system performance.

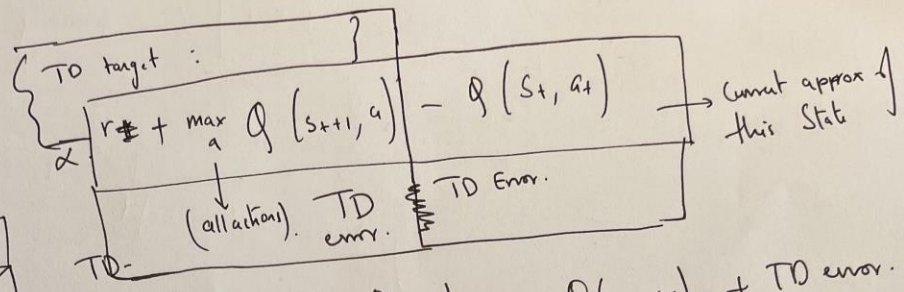
→ Policy
→ Planning

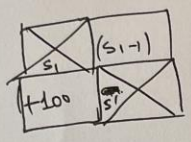
Class 1:


TD-MPC
CLASS-I
Key concepts
Recursively
Dynamic
Model-Based
State
Reward
Example
Distance 'H'
→ Angle α. to be determined
g[+100]
E[100]
finite Horizon: Limited no. of steps → Chooses actions accordingly.
Infinite Horizon: Unlimited no. of steps
mean
(max)
= answer
90
180
of θ = 30°,
no. of samples = 20
Sample no. of Elites = 5 Best
recompute - new Gaussian distribn
20 random sample from distribn
narrow down the elite selectn - close to the right answer.
reduce Gaussian size.
→ new mean, compute
calculate.
Find a good action trajectory
using Cross-Entropy Method.
sample action trajectory from multiple using
Gaussian distribn
What objective value would be?
Predict
Over finite Horizon.
→ propose action Trajectory
→ Having way to propose what objective value would be?
IE IR, find objective value funcn.
→ Plug & chug to Cross-Entropy Method

Class II


TD - MPC Class II
Latent states → faster → Lower dimensional State | Lower processing speed.
Focus on Relevant parts.
Policy network predicts actions. Gaussian take off
Cross - Entropy method.
moving left is better, the robot can pick up cues over this.
the imaginary Bands would be useful.
Starting to push cube across
TD learning before.
Online & offline learning
Learn value func
Robot
Target Cube
Reward
Parameter Horizon = s_1
Steps 1.
States
Actions
Rewards
Encoder
Policy
Noise
Loss (MSE)
pred action | actual action
MPC → not optimal
→ noise | Explori
Traj → not optimal due to noise
π_θ Network $(s, a) \rightarrow \mathbb{E}$
→ max ($\pi_{network}$)
Advantage weighted Action
Q Network
V - Network
Actual dataset
$R(s_2)$
(MSE)
Consistency Loss
π → Action
RwdM → Reward Model
Consistency Loss



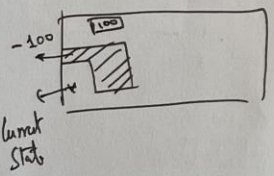


Take $Q(s_t, a_t)$ → $Q(s_t, a_t)$ + TD error.

Current Estimate

$+ \alpha$ [TD error]

(+100)
− 1
(+99)
$(s_1' + 100)$
$(s_1 - 1)$



(−) for Extra time Step

target

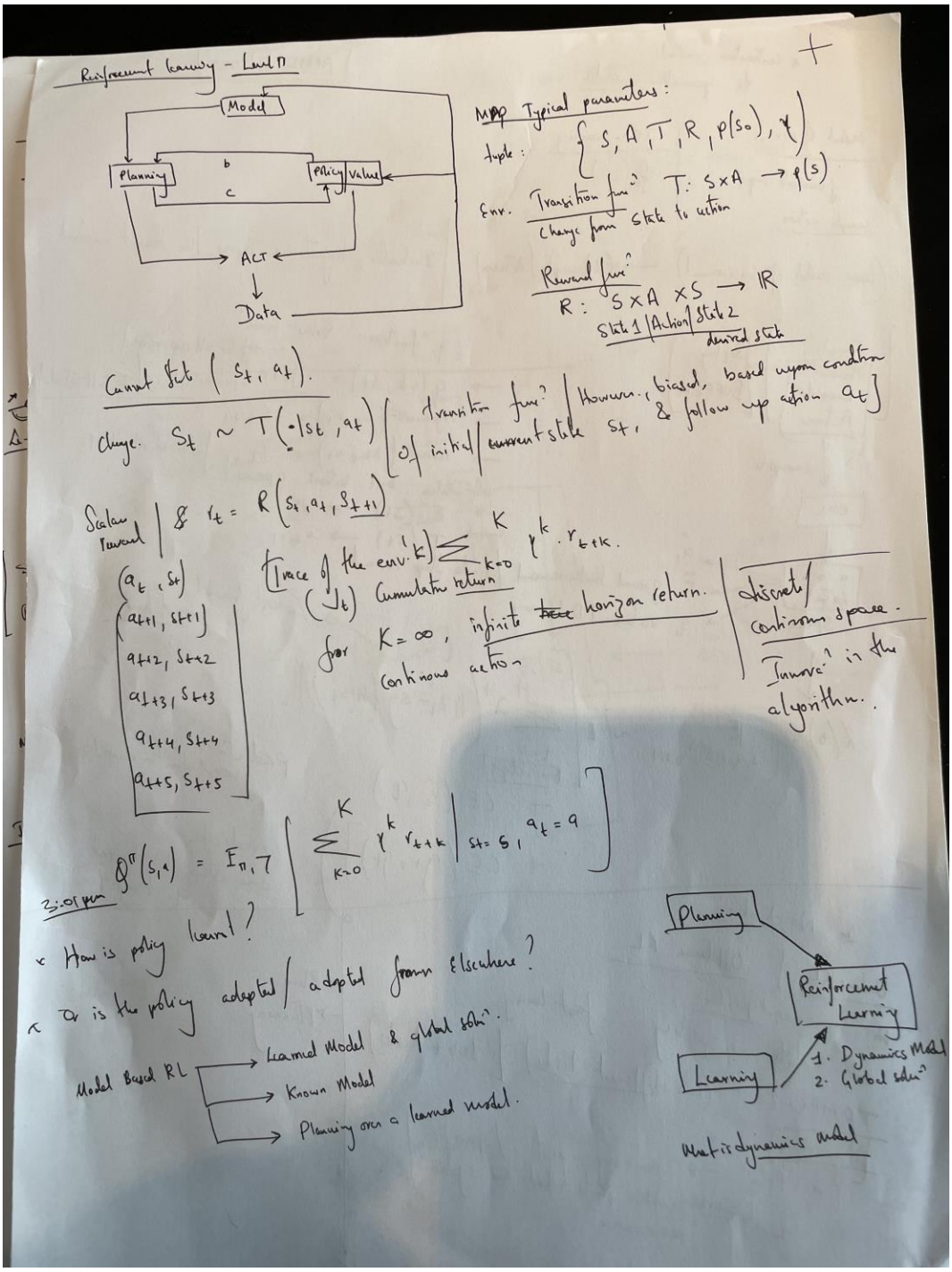
$$MSE\left[q(\hat{z}_1, a_1), r_2 + q_{\text{network}}\left(\hat{z}_2, \underbrace{\pi(\hat{z}_2)}_{\text{policy model}}\right)\right]$$

Best possible action in any state

Stop Grad here

not actual weights
actual exponential moving avg. → stable target.

Reinforcement learning Overview:



APRIL 16 | 2025

- Markov Decision problems (MDP)
probability of trajectory (states, actions & Rewards) of length 'T':

$p(\tau) = P_0(s_0)$

Length 'T': T_1 T_2 T_3 T_4 .. $(T-1)$

Trajectory - (T) (T-1)

States (example) Reward

- States | s_0 start, s_1 neutral, s_2 goal

Actions : a_{left}, a_{right}

Rewards: s_2 = | $r = +10$; others | $r = 0$

$P_0(s_0) = 1.0$ Initial

$\pi(a_{left} | s_0) = 0.4$ Policy

Policy action (π): Deterministic | always $\begin{matrix}1.0\\0.0\end{matrix}$ or $\begin{matrix}0.0\\1.0\end{matrix}$

: Stochastic | 0 . . . . 1.0 (left ... right)

π |

→ Product Symbol (π) → trajectory probability | Chain sequentially

$$p(\tau) = P_0(s_0) \prod_{t=0}^{T-1} \pi(a_t | s_t) P_T(s_{t+1} | s_t, a_t) P_R(r_t | s_t, a_t, s_{t+1})$$

In a MDP

- States (total T+1 states) | $s_0 s_1 s_2 \dots s_T$
- Actions (total T actions) | $a_0, a_1, a \dots a_{T-1}$
- Rewards (total T Rewards) | $r_0, r_1, \dots r_{T-1}$

$p(\tau) = P_0(s_0) \cdot \pi(a_0 | s_0) P_T(s_1 | s_0, a_0) P_R(r_0 | s_0, a_0 s_1) \cdot \pi(a_1 | s_1) \cdot$

$* \pi(a_1 s_1) \cdot P_T(s_2 | s_1, a_1) P_R \cdot (r_1 | s_1, a_1, s_2)$

$* \pi(a_2 | s_2) P_T(s_3 | s_2, a_2) P_R(r_2 | s_2, a_2, s_3)$

$= 1.0 * [s_0, a_{right}, r_1 = 0, s_1, a_{right}, r_1 = 0, s_2, a_{right}, r_2 = 10, s_3]$

Initial
$P_0(s_0) = 1.0$

Policy
$\pi(a_{right} | s_0) = 0.6$
$\pi(a_{left} | s_0) = 0.4$

Transition
$P_T(s_1 | s_0, a_{right}) = 0.8$
$P_T(s_2 | s_1, a_{right}) = 0.9$

$r=0$, $r=10$

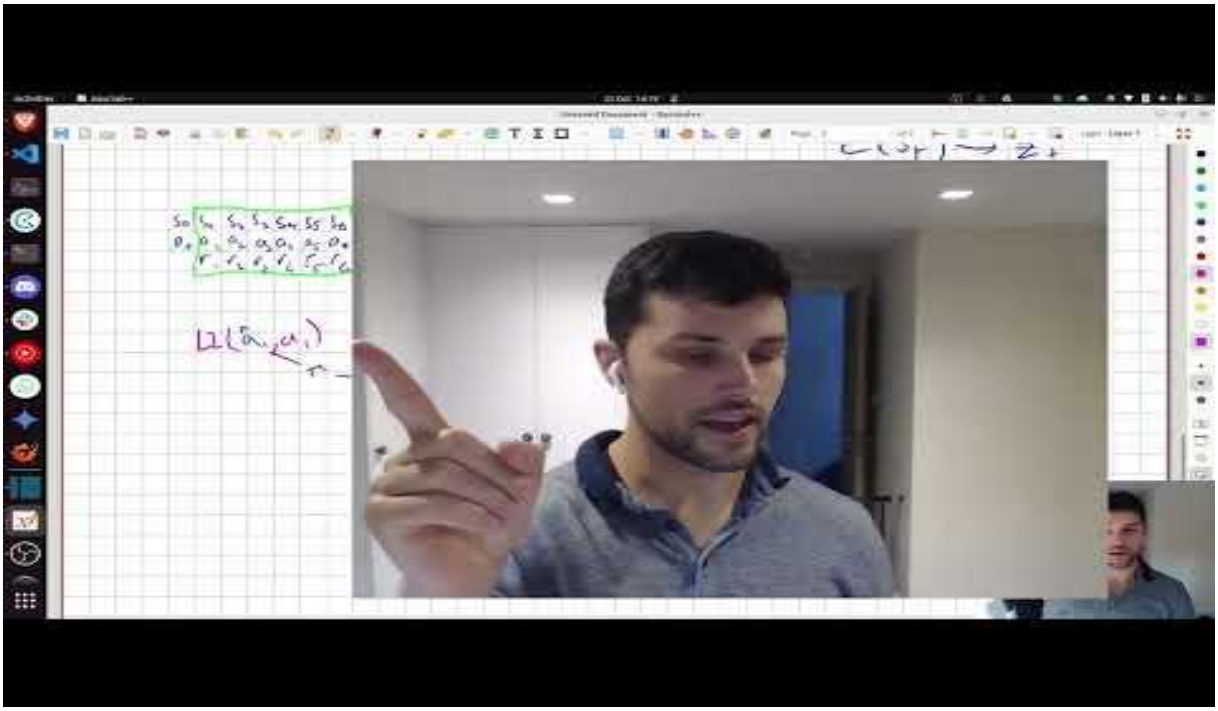
Need to reach s_3

Probability transition state $s_1 \rightarrow s_0$, a_{right}

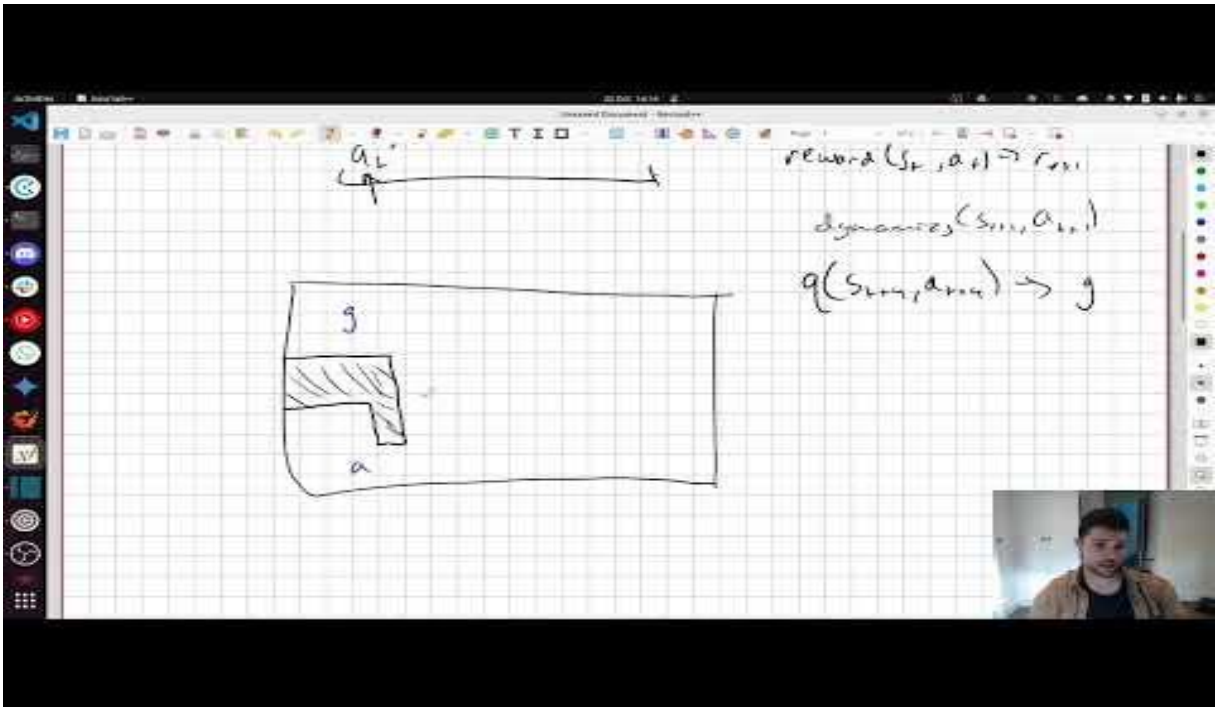
Sources:

Original Paper: https://github.com/nicklashansen/tdmpc

Reference learning: https://www.youtube.com/watch?v=_CKJJRAlvKI



https://www.youtube.com/watch?v=--hDN4LLmPI


reward(s_t, a_t) → r_{t+1}
dynamics(s_{t+1}, a_{t+1})
q(s_{t+1}, a_{t+1}) → g
g
a