# Reinforcement Learning

## Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2025-05 Dozent: Karsten Flügge

# Themen des Tages

## Tag 13

**Novelty**, curiosity, surprise: **Exploration II**

**Hierarchic** RL <> HR

**LLM+RL**

**Diskussion**

# Dumb exploration

**Bisher**ige Exploration war **zufällig** und **ungerichtet**:

**ε-greedy** zufällig per Definition (aber ok wegen Gesetz der großen Zahlen)

Besser:

**softmax**, Aktion je nach W-keit (nie null!)

# Information und Entropie

**Shannon-Information := -log(p)  self Information** (≠ Fischer Information)

**p=1 => -log(1) = 0** keine 'Information' wenn ein sicheres Ereignis eintritt

**p≈0 => -log(0) ≈ ∞** maximale 'Information' wenn ein seltenes Ereignis doch eintritt

**"punktweise Überraschung"**

## Entropie:

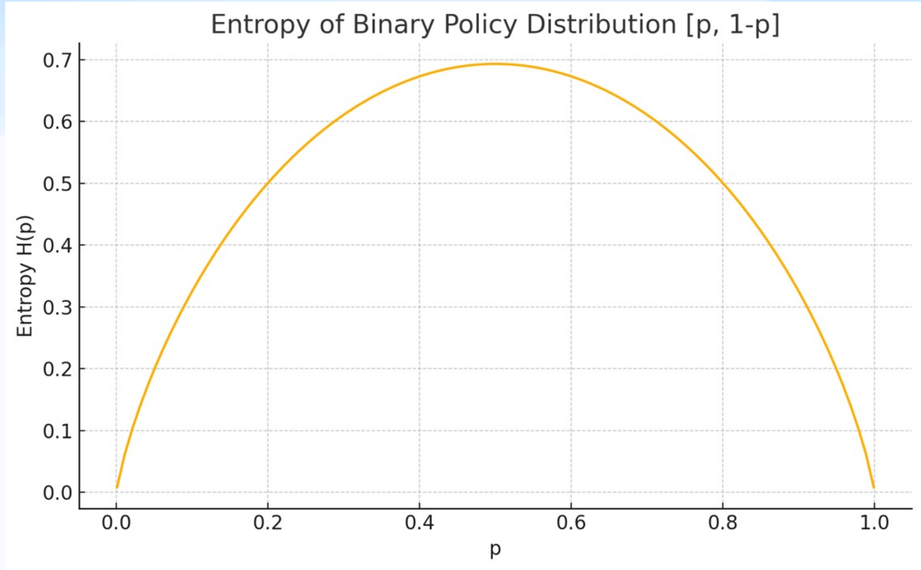
**Entropie (mittlere Shannon-Information):**

durchschnittliche Überraschung / Unordnung (einer Verteilung)

$$H(p) = -\sum p_i * \log(p_i) \quad \sum p_i = 1$$

Bei Unfairer Münze die immer Kopf/Zahl ergibt, gibt es keine Überraschungen => H=0

Bei Fairer Münze die mal Kopf/Zahl ergibt, gibt ist Ergebnis nicht vorhersehbar "Überraschungsfaktor H=.7"




Wird verwendet z.B. in **SAC, PPO, MaxEntropy IRL**

# Profi exploration

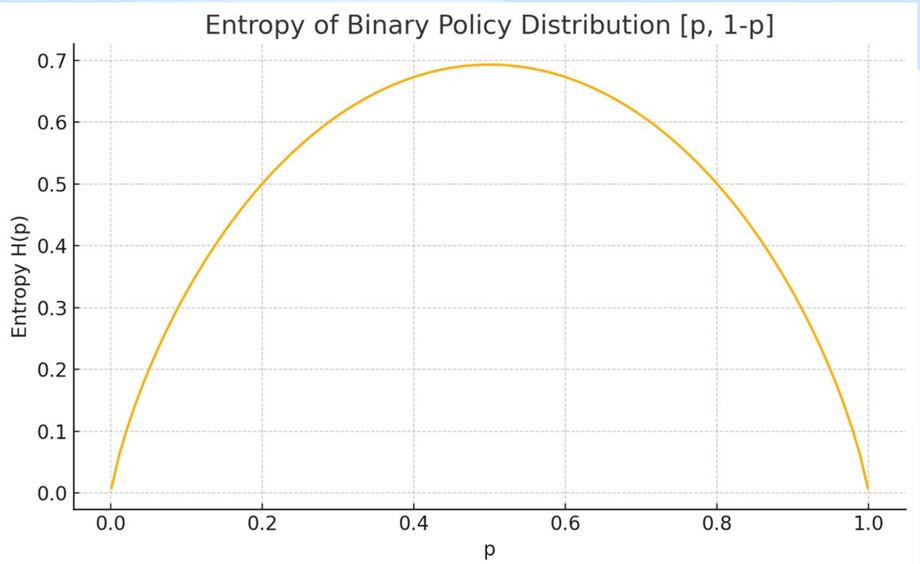
## Entropie:

$$\mathcal{L}_{\text{actor}} = \mathbb{E}[A(s,a) + \alpha \cdot \mathcal{H}(\pi(\cdot|s))]$$

- $\mathcal{H}(\pi)$: Entropy of the policy, i.e. $-\sum \pi(a|s) \log \pi(a|s)$
- Promotes **action diversity** (randomness), especially early in training

Does **not modify rewards**, but affects the gradient

- **No environment signal** involved
- **No model** involved
- **No reward** shaping




**Entropie: Maß für Chaos / Unsicherheit / Diversität**

**Kreuzentropie: -∑p*log(q) Maß für Treffsicherheit / Informationsverlust** (bei Softmax)

# Profi exploration

$$\mathcal{L}_{\text{actor}} = \mathbb{E}[A(s,a) + \alpha \cdot \mathcal{H}(\pi(\cdot|s))]$$

- $\mathcal{H}(\pi)$: Entropy of the policy, i.e. $-\sum \pi(a|s) \log \pi(a|s)$
- Promotes **action diversity** (randomness), especially early in training

**Automatische Entropie:**

Adaptives α: Viele moderne Verfahren (z. B. Soft Actor-Critic) lernen α automatisch durch eine zusätzliche Optimierung:

Dadurch wird α **erhöht**, wenn die Policy zu **deterministisch** wird, und **gesenkt**, wenn **zu viel Entropie** da ist.

# Novelty, surprise: curiosity exploration

- **Novelty** (e.g., **count-based** or hash-based methods)
- Curiosity via **surprise** (e.g., **prediction error** of *models*) <>
- Curiosity via **uncertainty** (e.g., Bayesian uncertainty or **ensemble disagreement**)

Bekannte Loss-Terme, die systematischer Exploration induzieren:

• **Exploration Bonus (Count-based / Pseudo-count):**
Zusätzlicher Reward $r' = r + \beta/\sqrt{N(s,a)}$. Das geht direkt in den Policy-Loss ein, ohne an Entropie gebunden zu sein. Implementiert als "intrinsic reward", nicht als Entropie.

• **KL-Regularisierung gegen eine Prior-Policy:**
Statt Entropie $-\sum \pi \log \pi$, minimiert man $\mathrm{KL}(\pi(\cdot|s)\|\pi_0(\cdot))$ gegen eine gewünschte Explorationsprior (z.B. uniforme Policy).
Loss-Term:
$L_{\mathrm{KL}} = \alpha \mathrm{KL}(\pi_\theta(\cdot|s) \,\|\, U(\cdot))$

→ zwingt systematisch nahe an Uniformität, bis genügend Daten vorliegen.

• **Information Gain (VIME, variational intrinsic motivation):**
Reward proportional zur Veränderung der Belief-Verteilung über das Modell durch eine Aktion. Loss-Terme enthalten dann KL zwischen posterior und prior Model-Parameterverteilungen.

• **Divergence Measures statt Entropie:**
Tsallis- oder Rényi-Entropie-Regularizer. Höhere Sensitivität gegen zu kleine Wahrscheinlichkeiten, kann systematischer unentdeckte Actions hochziehen.

# Novelty, surprise: curiosity exploration

KL  hält ja Abstände klein, wie kann es gleichzeitig Diversität erhöhen?

Das scheinbare Paradox löst sich, wenn man genau hinsieht, *welche* Referenzverteilung in der KL steht:

- **Trust Region (PPO/TRPO)**

$D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{old}})$

→ KL klein halten ⇒ Policy-Update bleibt *nah* an der alten Policy. Das sichert Stabilität, bremst aber Diversität.

- **Exploration-Regularisierung**

$D_{\mathrm{KL}}(\pi_\theta(\cdot|s) \,\|\, U(\cdot))$

wobei $U$ die uniforme Verteilung ist.

→ KL klein halten ⇒ Policy *muss* nah an Uniformität sein ⇒ jede Aktion erhält positive Wahrscheinlichkeit ⇒ Diversität ↑.

- **Information Gain (VIME u.ä.)**

$\mathbb{E}\left[D_{\mathrm{KL}}(p(\theta|D, s') \,\|\, p(\theta|D))\right]$

→ Maximierung der KL-Differenz zwischen Posterior und Prior = Aktionen suchen, die das Modell maximal verändern ⇒ führt zu Explorationsverhalten.

**Stabilität versus Diversität!**

# Novelty, surprise: curiosity exploration

## KL-Divergenz

**KL(p,p') = Kreuzentropie(p,p') - Entropie(p)**

**KL(p,p') = -∑p·log(p') + ∑p·log(p)  = ∑p· log(p)/log(p') =  ∑p · log_ratio(p,p')**

$D_{\mathrm{KL}}(p \parallel q) = H(p,q) - H(p)$ *Extra-Unsicherheit*, die nur entsteht, weil Modell **p'** nicht zur Wirklichkeit **p** passt.

**KL-Divergenz** misst, wie „**anders**“ die neue Policy **π′** im Vergleich zur alten Policy **π** ist.

KL Target ist ein Sollwert:

Wenn die gemessene KL kleiner als der Zielwert ist → **die Lernrate kann erhöht werden**

(größere Updates → schnellere Anpassung).

Wenn die gemessene KL größer ist → **die Lernrate wird reduziert**

(kleinere Updates → stabiler, weniger riskant).

Das Ziel: nicht zu große Policy-Sprünge (verhindert „Policy Collapse“),

aber auch nicht so kleine Updates, dass das Training ewig dauert.

# Novelty, surprise: curiosity exploration

- **Novelty** (e.g., **count-based** or hash-based methods)

  **zähle** wie oft ein State erreicht wurde!

  diskret oder über continuous neighborhood **estimate, distance**

- Curiosity via **surprise** (e.g., **prediction error** of forward models) <>
- Curiosity via **uncertainty** (e.g., Bayesian uncertainty or **ensemble disagreement**)

sigmoid peeks vs near uniform

Steuerung zum Beispiel in **policy direkt** oder über **reward**

$$r_t^{\text{aug}} = r_t^{\text{env}} + \beta \cdot r_t^{\text{explore}}$$

<> **Advantage** orthogonal objectives:

*advantage* optimizes **known rewards**,

*exploration* bonuses **bias** toward acquiring new knowledge

# Novelty, surprise: curiosity exploration

- Curiosity via **surprise** (e.g., **prediction error** of forward models) <>

Steuerung zum Beispiel in **policy direkt** oder über **reward** $r_t^{\text{aug}} = r_t^{\text{env}} + \beta \cdot r_t^{\text{explore}}$

Führt so eine Belohnung dazu dass absichtlich *falsche Vorhersagen* getätigt werden??

# Novelty, surprise: curiosity exploration

• Curiosity via **surprise** (e.g., **prediction error** of forward models) <>

Steuerung zum Beispiel in **policy direkt** oder über **reward**

Führt so eine Belohnung dazu dass absichtlich falsche Vorhersagen getätigt werden?

**JA! Problematisch:**

• **Exploration wird Selbstzweck**

→ Agent sucht **maximales Unwissen** *statt* **sinnvolle Information**

• **Loophole bei reward shaping**

→ Agent trickst das Lernsystem aus, um hohe intrinsische Belohnung zu erhalten

• **Non-stationäres Ziel**

→ Belohnung basiert auf einem sich ständig verändernden Modell

**Gegenmaßnahmen:**

• **Im *latent space* arbeiten**

• **Bonus nur temporär** oder nur in **frühen Phasen**

• **Decay des Intrinsic-Reward-Gewichts**

• **Maskierung chaotischer Zustände** (z. B. durch epistemische Unterscheidung)

• **Use surprise only when learning is possible** (→ predictability prior)

Belohne nur Zustände mit **hoher epistemischer Unsicherheit** (nicht bloß hoher Fehler)

**Bild 8.5** Das Problem des verrauschten Fernsehers ist ein theoretisches und praktisches Problem, bei dem ein Reinforcement-Learning-Agent mit einem einfachen Sinn für Neugierde von einem verrauschten Fernseher in den Bann gezogen wird und ihn ewig anstarrt. Das liegt daran, dass er von Natur aus durch Unvorhersehbarkeit belohnt wird, und weißes Rauschen sehr unberechenbar ist.

# Novelty, curiosity, surprise: Exploration

- **Novelty** (e.g., **count-based** or hash-based methods)
- **Curiosity** (e.g., **prediction error** of forward models) <>
- **Surprise** (e.g., Bayesian uncertainty or **ensemble disagreement**)

NICHT **explizit** in Dreamer! Aber **implizit:**

Dreamer relies purely on:

1. **Stochastic world models** (e.g., RSSM)

→ Intrinsic variability in latent imagination leads to **structured exploration**.

2. **Entropy regularization** in the actor loss:

$$\mathbb{E}_{a\sim\pi}[\log \pi(a|s)]$$

→ Encourages **policy entropy**, indirectly promoting exploration.

Dreamer avoids these by **not adding any intrinsic reward**

- Leverages **learned latent dynamics** for **long-horizon planning**

# Novelty, curiosity, surprise: Exploration

**Manuell alle Aktionen ausprobieren**

```
# funktioniert bestens, wenn Zustandsraum klein und diskret ist
def policy_try_new(x,y):
  min = 2**31
  best_action = 0
  tries = 0
  for action in ACTION_SYMBOLS.keys():
    tries = actions_tried[x, y, action]
    if tries < min:
      min = tries
      best_action = action
  return best_action

# Entropie macht so etwas ähnliches
```

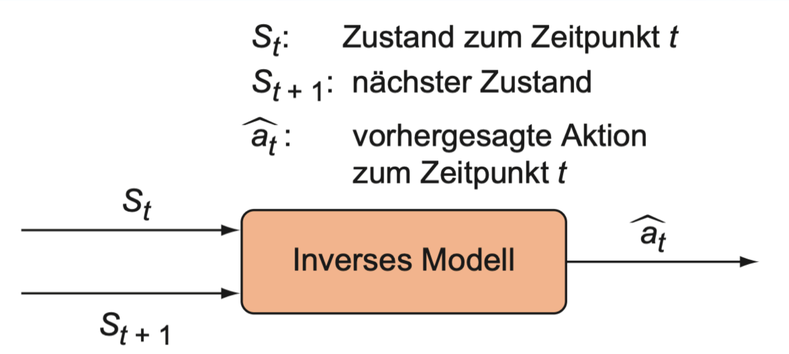
# „Curiosity-driven Exploration by Self-supervised Prediction“ (2017)

einer der am einfachsten zu implementierenden Algorithmen

Zusätzlich zur Dynamik, also zum Modell

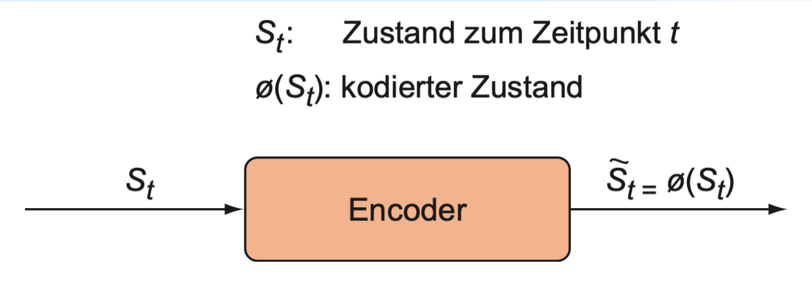
**p(s,a)=>s'** bauen wir die Rückrichtung

**f(s,s')=>a** welche versucht vorherzusagen mithilfe von **welcher Aktion zum nächsten Zustand** gefunden wurde.




Wir wollen uns nur auf '**wichtige Aspekte**' des Zustandes konzentrieren, daher

Encoder s=>**z**



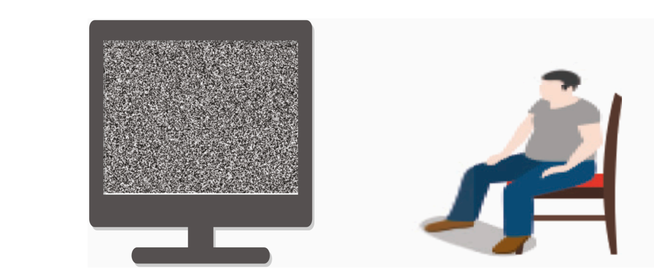



**Bild 8.5** Das Problem des verrauschten Fernsehers ist ein theoretisches und praktisches Problem, bei dem ein Reinforcement-Learning-Agent mit einem einfachen Sinn für Neugierde von einem verrauschten Fernseher in den Bann gezogen wird und ihn ewig anstarrt. Das liegt daran, dass er von Natur aus durch Unvorhersehbarkeit belohnt wird, und weißes Rauschen sehr unberechenbar ist.

# „Curiosity-driven Exploration by Self-supervised Prediction“ (2017)

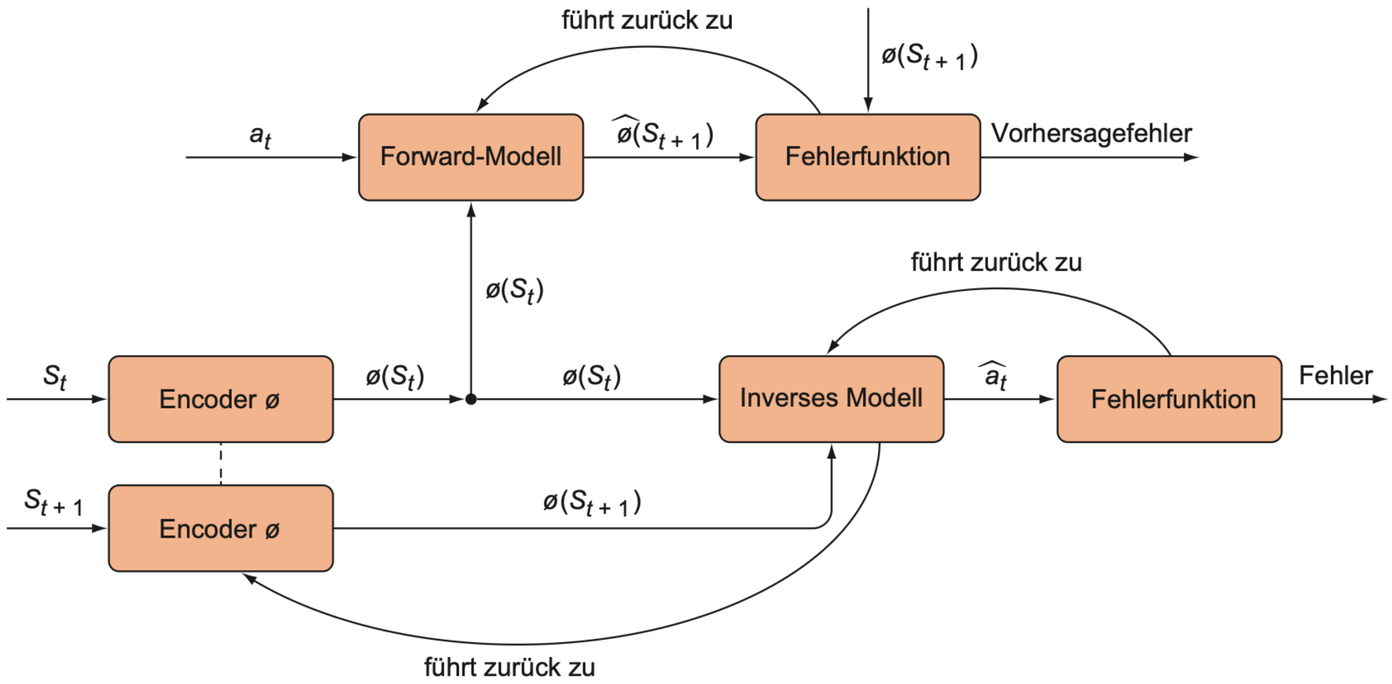



**Bild 8.10** Das Neugierde-Modul. Zuerst kodiert der Encoder die Zustände $S_t$ und $S_{t+1}$ in niedrigdimensionale Vektoren, $\phi(S_t)$ bzw. $\phi(S_{t+1})$. Diese kodierten Zustände werden an das Forward- und das inverse Modell weitergegeben. Beachten Sie, dass das inverse Modell auf das Encoder-Modell zurückgeführt wird und es somit durch seinen eigenen Fehler trainiert. Das Forward-Modell wird durch Backpropagieren aus seiner eigenen Fehlerfunktion trainiert, aber es wird nicht wie das inverse Modell zum Encoder zurückgeführt. Dadurch wird sichergestellt, dass der Encoder lernt, Zustandsdarstellungen zu erzeugen, die nur für die Vorhersage, welche Aktion durchgeführt wurde, nützlich sind. Der schwarze Kreis zeigt einen Kopiervorgang an, bei dem die Ausgabe vom Encoder kopiert und die Kopien an das Forward- und das inverse Modell weitergegeben werden.

# Pause

Later work (e.g., **Plan2Explore**, **DreamerV3 with pretraining**) does explore **intrinsic reward extensions** — but not Dreamer V1/V2 in their core forms.

**Contrasted with methods that do use explicit signals:**

| Method | Exploration Signal |
|---|---|
| ICM (Curiosity, 2017) | Prediction error of forward model |
| RND (Burda et al., 2018) | Disagreement with fixed random net |
| Count-based (MBIE-EB) | State visitation counts |
| Pseudo-counts | Density models (PixelCNN, etc.) |
| Disagreement (e.g. PETS) | Variance in model predictions |

Dreamer avoids these by **not adding any intrinsic reward**:

$$r_t^{\text{total}} = r_t^{\text{env}} \quad \text{(no exploration bonus)}$$

# Entropie und KL für Exploration

$$H(\pi) = -\sum_a \pi(a \mid s) * \log(\pi(a \mid s))$$

Hohe Entropie bedeutet: die Policy ist „breit“, verteilt Wahrscheinlichkeiten gleichmäßiger, probiert also mehr aus.

**Loss += $\alpha$*H($\pi$)**

Der Faktor $\alpha$ kontrolliert die **Balance** zwischen **Exploitation** (Belohnung maximieren) und **Exploration** (Policy unscharf halten).

KL

Exploration hier bedeutet nicht direkt „mehr zufälliges Ausprobieren“, sondern *kontrolliertes Verändern*: man bleibt nahe an der alten Policy, um stabile Lernschritte zu machen.

# Model statt Counts

In hochdimensionalen Umgebungen (z. B. Bilder) kann man nicht einfach zählen. Dort kommen **dichte Annäherungen**:

– **Pseudo-counts**: Zählwerte aus einem probabilistischen Modell der Zustände (Bellemare et al. 2016).

– **Intrinsic Curiosity Module (ICM):** misst Überraschung, also Vorhersagefehler des eigenen Weltmodells.

– **Random Network Distillation (RND):** misst, wie unbekannt ein Zustand ist, indem man ihn gegen ein zufälliges, festes Netzwerk prüft.

## Neugierde in Dreamer

1. **Lernen des Weltmodells:**

– Das **Modell** (RSSM: Recurrent State-Space Model) wird trainiert, um Zustandsdynamik und Rewards vorherzusagen.

– Fehlerhafte Vorhersagen zwingen das Modell, seine Repräsentationen zu verbessern.

– **Zustände, die neu oder selten sind, liefern große Gradienten → Modellkapazität wird dort stärker beansprucht**

– Wenn das Modell in einem Bereich unsicher ist, entstehen ***breitere Verteilungen*** im Latent Space.

– In solchen Regionen kann die Policy potentiell höhere Rewards entdecken, weil die Imagination den Suchraum „aufweitet“.

– Ergebnis: Eine Policy, die in unbekannte Regionen geht, hat langfristig bessere Performance, weil sie auf stabilere Modellvorhersagen zugreifen kann.

# Pause

# Pause

⚠️ Using an **MLP** instead of self-**attention**, we obtain better generalization on Sudoku-Extreme (improving from 74.7% to 87.4%; see Table 1).
💡 This worked well on Sudoku 9x9grids, given the small and fixed context length; however, we found this architecture to be suboptimal for tasks with large context length, such as Maze-Hard and ARC-AGI

Große Netze können schneller overfitten.

# ARC HRM

**https://arcprize.org/**

**Automatic Reasoning Competition**

**(Für Computer) schwere Probleme "Lösungen erst in 10 Jahren zu erwarten"**

**https://arcprize.org/blog/hrm-analysis**

**Hauptkritik: Zu viel augmentation.**

Adaptive computational time (ACT)  Q-learning objective "done"

Deep supervision and the 1-step gradient approximation provide a more biologically plausible and less computationally-expansive alternative to Backpropa-gation Through Time (BPTT)

**Essenz:**

**deep supervision** ist der eigentliche Ausschlaggeber:

reusing the previous latent features ($z_H$ and $z_L$) as initialization for the next forward pass.

Transformer mit durchgeschliffenem hidden state ist mächtig!

Transformer agiert wie RNN.

Each network is a 4-layer Transformers  384 layers of *effective depth* ≈ RNN rollout $_{n}\text{layers}_{(n+1)TN}\text{sup} = 4 * (2 + 1) * 2 * 16$

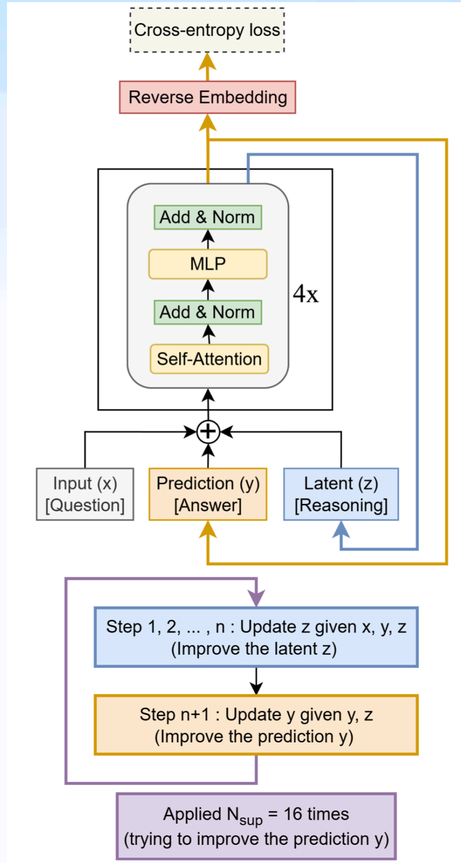
Using deep supervision doubled accuracy over single-step supervision (going from 19% to 39% accuracy), while recursive hierarchical reasoning only slightly improved accuracy over a regular model with a single forward pass (going from 35.7% to 39.0% accuracy).

# ARC HRM

**Folgepaper: Less is More: Recursive Reasoning with Tiny Networks https://arxiv.org/pdf/2510.04871v1**

Tiny Recursive Model (TRM), a much simpler recursive reasoning approach that achieves significantly higher generalization than HRM, while using a **single tiny network** with only **2 layers**.

With only 7M parameters, TRM obtains
45% test-accuracy on ARC-AGI-1 and
8% on ARC-AGI-2, higher than most LLMs
(e.g., Deepseek R1, o3-mini, Gemini 2.5 Pro)
with less than 0.01% of the parameters.
**SoTA**
**Sudoku**-Extreme from 55% to 87%,
**Maze**-Hard from 75% to 85%,
ARC-AGI-1 from 40% to 45%, and
ARC-**AGI-2** from 5% to 8%



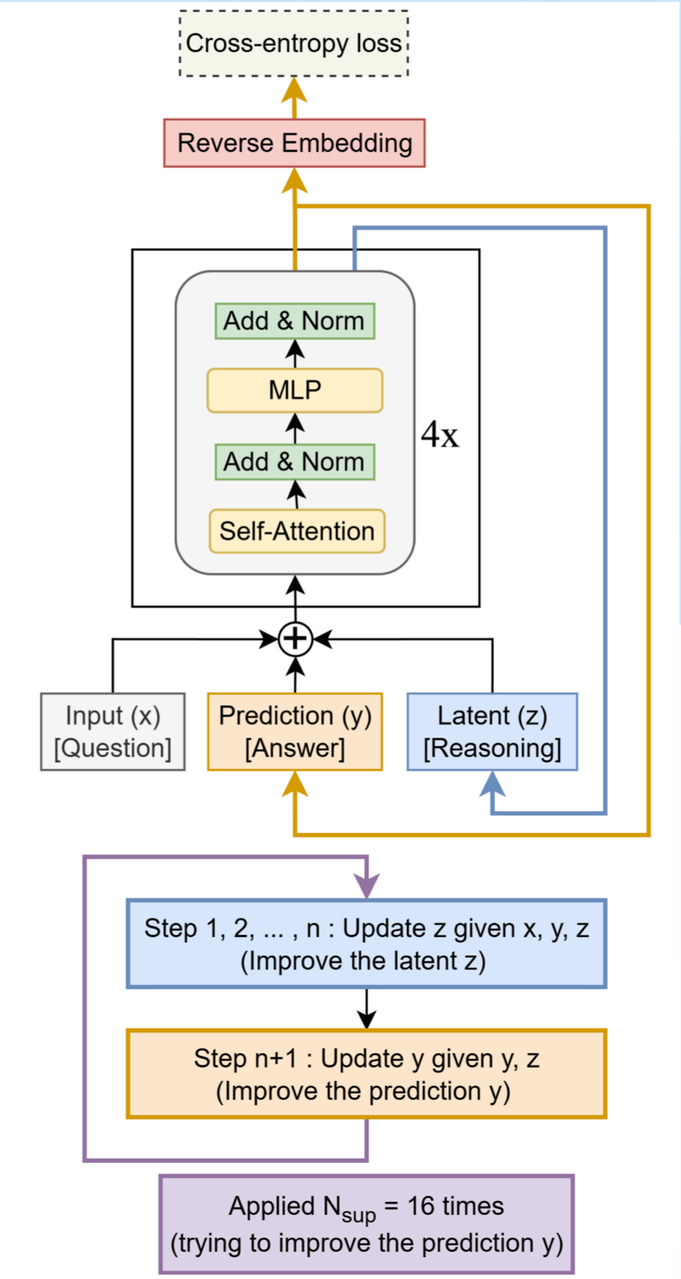
# ARC HRM

## Less is More: Recursive Reasoning with Tiny Networks

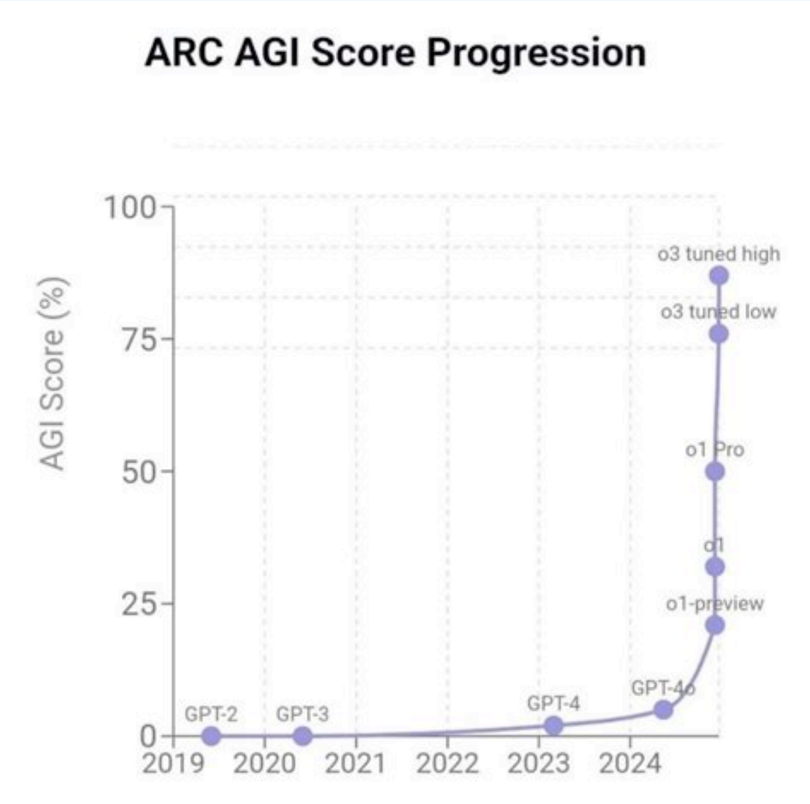
```python
def latent_recursion(x, z, n=6):
    for i in range(n+1): # latent recursion
        z = net(x, z)
    return z

def deep_recursion(x, z, n=6, T=3):
    # recursing T−1 times to improve z (no gradients needed)
    with torch.no_grad():
        for j in range(T-1):
            z = latent_recursion(x, z, n)
    # recursing once to improve z
    z = latent_recursion(x, z, n)
    return z.detach(), output_head(y), Q_head(y)

# Deep Supervision
for x_input, y_true in train_dataloader:
    z = z_init
    for step in range(N_supervision):
        x = input_embedding(x_input)
        z, y_hat, q_hat = deep_recursion(x, z)
        loss = softmax_cross_entropy(y_hat, y_true)
        loss += binary_cross_entropy(q_hat, (y_hat == y_true))
        z = z.detach()
        loss.backward()
        opt.step()
        opt.zero_grad()
        if q[0] > 0: # early−stopping
            break
```



# ARC HRM




https://arcprize.org/blog/hrm-analysis

# LLM+RL

**LLM+RL**

**RLHF RL with human feedback OpenAI finetuning**

**LLM liest sich Internet durch => Nazi**

**Supervised Fine-Tuning (SFT)**

**DeepSeek R1 'reasoning Modelle' nach GPT-4**

**LLM** liest sich Internet durch =>

Mathe Aufgaben: **<think>**…**</think><answer>**X**</answer>** extended chain-of-thought later: THINK special tokens

**reward** bei richtiger Lösung, Lösungsweg frei!

innerhalb von think: chinesisch und tokens die kein sinn machen. “aha moment” revision of previous answer state.

**extra Belohnung** wenn CoT text in **<think>**…**</think>** 'leserlich' ist.

Group Relative Policy Optimization (**GRPO**) Vereinfachung von PPO

Vergleiche Gruppe von **Kandidaten policies π k** via **group advantage, statt critic!**

https://www.datacamp.com/blog/what-is-grpo-group-relative-policy-optimization

https://verl.readthedocs.io/en/latest/algo/grpo.html

$$A_i = \frac{r_i - \mathrm{mean}(\{r_1, r_2, \cdots, r_G\})}{std(\{r_1, r_2, \cdots, r_G\})}$$

GRPO is defined as follows:

$$\max_{\pi} \mathbb{E}_{y \sim \pi_k(\cdot|x)} \min \left( \frac{\pi(y|x)}{\pi_k(y|x)} A_{\pi_k}(x,y),\ \mathrm{clip}\left( \frac{\pi(y|x)}{\pi_k(y|x)}, 1-\epsilon, 1+\epsilon \right) A_{\pi_k}(x,y) \right) - \beta \mathrm{KL}(\pi || \pi_{\mathrm{ref}}),$$

**Liefert das RL hier wirklich Mehrwert? Wird aktuell diskutiert/debattiert/gestritten.**

**Hierarchical Reasoning Model**

## LLM+RL

**Muscle Memory?**

Jailbreak verhindern via special tokens.

Z.B. <system> </system>  SYSTEM_MESSAGE_TOKEN

Wie viel kann System Instruktion überschrieben werden?

Nie endgültig!

PROMPT injection

# LLM+RL

Chain-of-thoughts (CoT)

# Übersicht

**Unsere Algorithmen und Formeln und Loss**

**SARSA** basic loop

**TD Temporal Difference** (one step)

**Deep Q-Learning  DQL / DQN**

**Policy Gradient-Methoden (PG)**  REINFORCE

**Actor-Critic-Methoden (AC)**

**Monte Carlo (MC)** (whole episode)

**PPO**

# Übersicht

**Unsere Algorithmen und Formeln und Loss**

| Methode | Typ | Update-Regel / Loss | Bemerkung |
|---|---|---|---|
| SARSA | On-Policy **TD** | $\mathbf{Q(s,a)} \leftarrow Q(s,a) + \alpha\,[r + \gamma\, Q(s',a') - Q(s,a)]$ | Focus auf **Loop** |
| Temporal Difference TD(0) | On-Policy **TD** | $V(s) \leftarrow V(s) + \alpha\,[r + \mathbf{\gamma\, V(s') - V(s)}]$ | ein **Step**, V / Q |
| Monte Carlo (MC) | episodisch | $V(s) \leftarrow V(s) + \alpha\,[\mathbf{G_t} - V(s)] \quad G_t = \sum_k \gamma^k\, r_{t+k}$ | **G**anze **Episode**, V / Q |
| DQN / DQL | Off-Policy TD + NN | Loss $L = [r + \mathbf{\gamma\, \max_{a'} Q_{t-}(s',a') - Q_t(s,a)}]^2$ | **PG** Loss via Adam |
| REINFORCE | On-Policy PG | Loss $L = \sum \log \pi(a \mid s) \cdot \mathbf{G_t}$ | $G_t$ kann durch A / δ ersetzt werden: |
| Actor-Critic (AC) | Hybrid PG + T**D** | Actor Loss $L = \log \pi(a \mid s) \cdot \mathbf{\delta}$; $\delta = r + \gamma\, V(s') - V(s)$ Critic L= δ | auch mit MC! |
| PPO | On-Policy PG | $L = E[\min(r_t \hat{A}_t, \mathbf{clip}(r_t, 1-\varepsilon, 1+\varepsilon)\hat{A}_t)]$ $r_t = \pi(a\|s)/\pi_t(a\|s)$ ratio | |