# MM

Frederik Schwarz
*Informatik*
*Hof-University*

Eugen
*Informatik*
*Hof-University*

***Abstract*—TRAINING**

## I. Project Architecture

Im Rahmen des Projekts wurde eine modulare und reproduzierbare Forschungs- und Entwicklungsinfrastruktur für das Training und die Evaluation spielender Agenten entworfen und umgesetzt. Die Gesamtarchitektur ist bewusst in klar abgegrenzte Subprojekte unterteilt, wobei jedes Subprojekt eine wohldefinierte Aufgabe innerhalb der Pipeline übernimmt. Diese Trennung ermöglicht sowohl eine isolierte Weiterentwicklung einzelner Komponenten als auch eine flexible Kombination im Gesamtsystem. Aufgrund der geringen Modellkomplexität des zur Aktionsgenerierung eingesetzten Agents können mehrere Bots parallel betrieben werden, was eine effiziente gleichzeitige Simulation einer großen Anzahl von Agenten ermöglicht.

Die gesamte Infrastruktur ist CLI-basiert aufgebaut. Sämtliche Funktionalitäten von der Datensatzerzeugung über das Training bis hin zur Evaluation werden über CLI-Tools angesteuert. Dadurch wird eine hohe Automatisierbarkeit sowie eine einfache Integration in Skripte und Batch-Prozesse erreicht, was insbesondere für experimentelle Workflows von zentraler Bedeutung ist.

Zur Konfiguration der einzelnen Subsysteme werden TOML-Konfigurationsdateien und CLI-Argumente verwendet.

Ein zentraler Bestandteil des Projekts ist die Implementierung von spielenden Agenten, die gegeneinander antreten. Als Agent wurde die lib `gobang` verwendet, welche gezielt korrigiert wurde, um identifizierte Fehler zu beheben. Die Spiellogik selbst wurde eigenständig implementiert und basiert auf effizienten NumPy-Operationen, um eine performante und minimale Simulation großer Spielmengen zu ermöglichen. Zusätzlich werden die Agenten parallel in einem Thread-Pool ausgeführt, um die verfügbare Rechenleistung optimal auszunutzen und die Simulationsgeschwindigkeit weiter zu erhöhen.

Ergänzend dazu wurden mehrere spezialisierte Module entwickelt, darunter ein Render-Modul, welches vollständig headless arbeitet und somit ohne grafische Ausgabe auskommt, ein Modul zur Datensatzerzeugung durch automatisierte Spielsimulationen, ein Trainingsmodul für das Anlernen der Modelle, ein Evaluations- bzw. Testmodul zur quantitativen Bewertung des Agenten sowie dedizierte Upload- und Download-Skripte zur Verwaltung experimenteller Artefakte.

Im Verlauf des Projekts hat sich die Infrastruktur iterativ weiterentwickelt. Während zu Beginn viele Komponenten lokal und pfadbasiert organisiert waren, wurde dieses Vorgehen schrittweise durch eine zentrale Upload- und Download-Struktur über Hugging Face ersetzt. Diese Umstellung ermöglichte ortsunabhängigen Zugriff auf experimentelle Ergebnisse.

Darüber hinaus wurden im Entwicklungsprozess zahlreiche Variablen und Konfigurationen explorativ variiert. Da die erforderlichen GPU-Ressourcen ausschließlich auf dem Hochschulserver zur Verfügung standen, erfolgten diese Parameteranpassungen direkt dort. Dieses Vorgehen ermöglichte eine effiziente Exploration des Parameterraums, bevor stabile Konfigurationen finalisiert und zentral versioniert wurden.

## II. Training

Die Datensätze, die für das Training verwendet wurden, sind das egan scha/gomoku_vlm_ds Dataset, die Repository für den Trainingscode und Dataset generation befindet sich unter github.com/frederik-uni/robo-muehle, und als Basis-Modell diente Qwen/Qwen3-VL-2B-Instruct.

### A. Überarbeitung des Trainingsskripts

Das ursprüngliche Trainingsskript wies mehrere Fehler und Inkonsistenzen auf, die zu instabilen Trainingsläufen und unzuverlässigen Evaluierungen führten. Insbesondere waren Teile des Datensatz preprocessing fehlerhaft implementiert. Aus diesem Grund wurde das Skript gänzlich überarbeitet.

### B. Trainingskonfiguration und Hyperparameterwahl

Das Fine-Tuning des Modells erfolgte mittels Supervised Fine-Tuning (SFT) unter Verwendung von Low-Rank Adaptation (LoRA), um eine parameter-effiziente Anpassung großer Sprach- und Multimodalmodelle zu ermöglichen. Ziel war es, sowohl visuelle als auch logische Fähigkeiten schrittweise zu verbessern, ohne das Basismodell vollständig neu zu trainieren.

### C. Optimierungsstrategie

Als Optimierer kam AdamW (adamw_torch) mit einer Gewichtsnormierung (weight decay) von 0.01 zum Einsatz. Die Lernrate wurde mittels eines Cosine Learning Rate Schedulers mit einer Warmup-Phase von 10 % der gesamten Trainingsschritte gesteuert. Diese Kombination erwies sich als stabil und reduzierte Oszillationen im Trainingsverlauf.

Die maximale Gradienten-Norm wurde auf 1.0 begrenzt, um exploding Gradient zu vermeiden. Das Training wurde im bfloat16-Modus durchgeführt, was eine höhere numerische Stabilität gegenüber fp16 bei vergleichbarem Speicherverbrauch bietet.

### D. Epochen, Batch-Größe und Akkumulation

Das Training wurde für 3 Epochen durchgeführt. Die effektive Batch-Größe ergab sich aus

der Kombination von per_device_train_batch_size und gradient_accumulation_steps, wodurch auch bei begrenztem GPU-Speicher eine stabile Optimierung ermöglicht wurde. Evaluierungen und Checkpoint-Speicherungen erfolgten schrittbasiert, angepasst an die jeweilige Datensatzgröße und Iterationsgeschwindigkeit.

### E. Wahl der Lernrate

Für den visuellen Trainingsabschnitt wurde eine Lernrate von $2 \times 10^{-4}$ verwendet. Für den nachgelagerten Strategy-Trainingsschritt (Step 2) wurde die Lernrate auf $1 \times 10^{-5}$ reduziert, um feinere Anpassungen auf höherer Abstraktionsebene zu ermöglichen.

Vorversuche mit einer durchgängig niedrigen Lernrate im Bereich von $10^{-5}$ zeigten jedoch, dass zu Beginn kaum messbare Fortschritte erzielt wurden. Insbesondere in Kombination mit niedrigen LoRA-Rank, wie sie in vielen bestehenden Arbeiten empfohlen werden, erwies sich die Anpassung als zu restriktiv. Diese empirischen Beobachtungen motivierten die Wahl einer höheren initialen Lernrate in den frühen Trainingsphasen.

### F. LoRA-Konfiguration und Rank-Wahl

Die LoRA-Adapter wurden mit einem Rank r = 32 konfiguriert. Dieser Wert liegt deutlich über den häufig in der Literatur genannten Bereichen (r = 8 oder r = 16), erwies sich jedoch in der Praxis als notwendig, um eine ausreichende Modellkapazität für die Zielaufgaben bereitzustellen.

Insbesondere bei multimodalen und strategischen Aufgaben zeigte sich, dass niedrigere Ränge die Anpassungsfähigkeit des Modells stark einschränkten. Erst mit r = 32 konnten konsistente Leistungsgewinne beobachtet werden. Die Skalierung der LoRA-Gewichte erfolgte mit lora_alpha = 1.5 × r, was die Stabilität des Trainings weiter verbesserte.

Je nach Trainingsmodus wurden unterschiedliche Zielmodule angepasst:

- Visueller Modus: Projektionen der Selbstaufmerksamkeit (q_proj, k_proj, v_proj, o_proj) sowie der multimodale Projektor.
- Logik-/Strategiemodus: Zusätzlich die Feedforward-Komponenten (gate_proj, up_proj, down_proj), um komplexere transformationsbasierte Anpassungen zu ermöglichen.

### G. Curriculum Learning und sequentielles Adapter-Merging

Das Training folgte einem Curriculum-Learning-Ansatz, bei dem das Modell schrittweise an zunehmend abstrakte Aufgaben herangeführt wurde. Zunächst lag der Fokus auf visuell dominierten Aufgaben, bevor in späteren Phasen strategische und logisch anspruchsvollere Inhalte trainiert wurden.

Zur technischen Umsetzung dieses Curriculums wurden die LoRA-Adapter sequenziell trainiert und anschließend vollständig in das Hauptmodell gemerged. Nach Abschluss eines Trainingsabschnitts wurden die Adaptergewichte in das Basismodell integriert und der Adapter anschließend entladen. Dieser Ansatz hatte zwei wesentliche Vorteile:

1) Stabilität: Das sequentielle Mergen verhinderte Warnungen und potenzielle Inkonsistenzen beim gleichzeitigen Laden mehrerer Adapter.
2) Explizite Wissensakkumulation: Jede Trainingsphase baute direkt auf den zuvor integrierten Gewichten auf, wodurch das Curriculum explizit im Modellzustand verankert wurde.

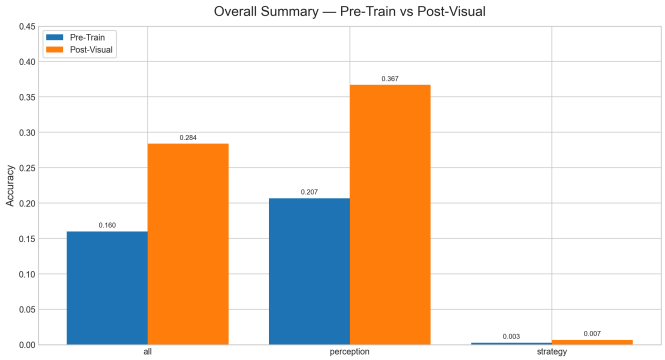
Durch dieses Vorgehen entstand eine klare Abfolge von Lernschritten, bei der jede Phase das Ergebnis der vorherigen konsolidierte, anstatt konkurrierende Adapter parallel zu halten.

### H. Setup und Vergleichslogik

Zur Quantifizierung des Trainingseffekts des LoRA-Adapters vergleichen wir die in [baseline-paper] berichtete Pre-Training-Baseline mit dem Modellzustand nach dem Fine-Tuning. Als Leistungsmaß wird die Accuracy herangezogen, wobei die Bewertung durch einen LLM-as-a-Judge-Ansatz (LISA) erfolgt. Die Ergebnisse werden aggregiert über (i) alle Aufgaben (`all`), (ii) den Wahrnehmungsanteil (`perception`) und (iii) den Strategieanteil (`strategy`) sowie zusätzlich getrennt nach Fragefokus und Fragevariation (`Q...`) ausgewiesen. Pro Fragevariation umfasst der Trainingsdatensatz insgesamt 250 Samples, der Testdatensatz enthält jeweils 25 pro Fragevariation.

## III. Ergebnisse

### A. Ergebnisse des Visual Fine-Tunings




### B. Evaluation des Visual Fine-Tunings

Nach dem Fine-Tuning der visuellen Komponente zeigt sich ein deutlicher Performancegewinn auf den Wahrnehmungsaufgaben: Die Gesamt-Accuracy steigt von 0.16 auf 0.284, während `perception` von 0.207 auf 0.367 zunimmt. Demgegenüber bleibt `strategy` erwartungsgemäß nahezu unverändert von 0.003 auf 0.007, da in diesem Trainingsschritt ausschließlich visuelle Fragefoki optimiert wurden.

a) *Ergebnisse nach Fragefoki:*

Die deutlichsten Verbesserungen zeigen sich bei den kernvisuellen Fragefoki. So steigt die Accuracy für `color_at_position` von 0.48 auf 0.72. Über mehrere Fragefoki hinweg fällt zudem auf, dass insbesondere Varianten mit frühen Spielzuständen hohe Genauigkeiten

erreichen. So erhöht sich beispielsweise Q1 innerhalb von `color_at_position` von 0.64 auf 0.96.

b) *Zählaufgaben:*

Auch die Zählaufgaben profitieren deutlich: `count_black_stones` verbessert sich von 0.01 auf 0.24 und `count_white_stones` von 0.04 auf 0.27, wobei die Varianten Q101 und Q201 jeweils um +0.56 zulegen. Abweichend hiervon verhält sich `count_empty_intersections`: Hier deutet sich an, dass die Aufgabe für das Modell tendenziell leichter wird, je später der Spielzustand gesampelt wird. Dies ist plausibel, da mit zunehmender Rundenzahl mehr Felder belegt sind und folglich weniger freie Schnittpunkte zu zählen bleiben. Im Unterschied zu den übrigen Zählfoki weist dieser Fokus damit eine inverse Schwierigkeit über den Spielverlauf auf.

Gleichzeitig erreichen auch sehr frühe Spielzustände eine gewisse Genauigkeit. Eine mögliche Erklärung ist, dass das Modell in diesen Fällen nicht ausschließlich durch explizites Zählen zum Ergebnis gelangt, sondern die bekannte Größe eines unbespielten Spielbrettes ausnutzt. Sobald jedoch sehr viele freie Schnittpunkte explizit zu zählen sind, fällt die Performance auf 0.00, was darauf hindeutet, dass das Modell bei hoher Anzahl freier Positionen auf einem $15 \times 15$-Brett an seine Grenzen stößt.

c) *Reihenerkennung und Folgerelationen:*

Schließlich verbessern sich auch Aufgaben zur Erkennung von Spielsteinreihen signifikant: `three_in_a_row` steigt von 0.07 auf 0.27 und `four_in_a_row` von 0.02 auf 0.24. Den größten relativen Sprung zeigt dabei `three_in_a_row` Q401 von 0.00 auf 0.68.

Die gestärkte Fähigkeit, Reihenstrukturen zuverlässig zu erfassen, spiegelt sich zudem in Fragefoki wider, deren Kern darin besteht, nahezu geschlossene Reihen zu erkennen und daraus unmittelbare Konsequenzen abzuleiten. So steigt `can_you_win` von 0.54 auf 0.79 und `can_you_lose` von 0.51 auf 0.63. Beide Foki weisen damit ein Ergebnis auf, welches deutlich besser ist als einfaches Raten.

d) *Fokus `print_board_matrix`:*

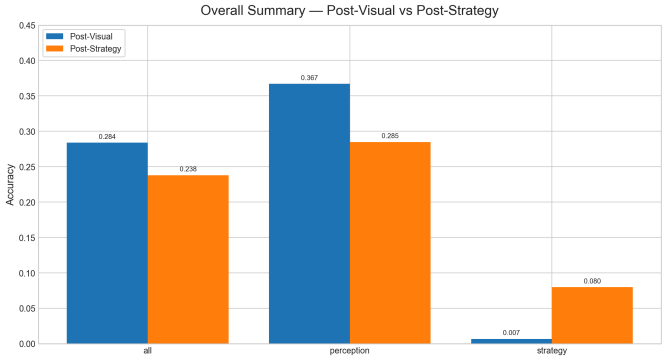
Für den Fokus `print_board_matrix` bleibt die Accuracy trotz Visual-Fine-Tunings bei 0.00. Dieses Ergebnis ist erwartbar, da die Aufgabe faktisch eine stark verschärfte Variante von `color_at_position` darstellt: Statt den Feldzustand an einer einzelnen Koordinate zu bestimmen, muss das Modell den vollständigen $15 \times 15$-Spielzustand (also 225 Felder) korrekt als Matrix ausgeben. Bereits kleine Abweichungen (ein einzelnes falsches Feld, ein Formatfehler oder eine vertauschte Zeile) führen zur Gesamtfehlklassifikation.

Der Fokus wurde dennoch bewusst im Frageset belassen, da er als Baustein für die späteren `reason_next_move`-Aufgaben vorgesehen ist: Ziel ist, dass das Modell den aktuellen Spielzustand explizit rekonstruiert und darauf aufbauend mittels (**chain-of-thought**) eine konsistente Zugentscheidung ableitet. Dabei ist nicht zwingend erforderlich, dass die Matrixrekonstruktion in jeder Zelle fehlerfrei ist, vielmehr war die Annahme, dass eine weitgehend korrekte, explizit ausformulierte Zustandsrepräsentation dem Modell helfen kann, strategisch relevante Strukturen zuverlässiger zu identifizieren und dadurch die finale Zugwahl näher am Entscheidungsverhalten des regelbasierten Bots zu treffen.

### C. Setup des Strategy Fine-Tunings

Zur Stabilisierung des Trainings bei hoher Lernrate (bedingt durch die Schwierigkeit der Fragestellungen und den vergleichsweise kleinen Datensatz) setzen wir neben einem eingefrorenen visuellen Adapter zusätzlich Warmup, Early-Stopping auf Basis des Eval-Datensatzes sowie Rehearsal bereits gelernter Fragen in einem Anteil von 10% ein. Eine geringere Lernrate ($2 \times 10^{\{-5\}}$ statt $2 \times 10^{\{-4\}}$) wurde ebenfalls probiert, führte jedoch erwartungsgemäß zu einem deutlich reduzierten Trainingserfolg der strategischen Komponente und wurde daher verworfen.

### D. Ergebnisse des Strategy-Fine-Tunings




### E. Evaluation des Strategy Fine-Tunings

Auffällig ist insbesondere, dass das Strategy-Fine-Tuning nicht nur die strategischen Fragefoki beeinflusst, sondern auch spürbare Nebenwirkungen auf mehrere visuelle Foki hat. Während `determine_who_won` im Mittel relativ stabil bleibt, zeigen vor allem die Zählaufgaben (`count_*_stones` sowie `count_empty_intersections`) deutliche Einbußen gegenüber dem Post-Visual-Zustand. Dieses Muster deutet auf einen leichten **catastrophic forgetting**-Effekt hin: Obwohl während des Strategy-Fine-Tunings Rehearsal und Stabilisierungstechniken (u. a. Warmup und Early-Stopping) eingesetzt wurden, deuten die rückläufigen Werte inbesenondere in einzelnen, besonders schwierigen visuellen Foki (den Zählfoki) darauf hin, dass die Optimierung auf die neu eingeführten strategischen Fragen bei gleichzeitig hoher Lernrate die zuvor gelernten visuellen Teilfähigkeiten teilweise überlagert hat.

Gleichzeitig zeigt sich beim neu eingeführten strategischen Fokus `win_next_turn` ein klarer Leistungsgewinn (im Vergleich zum Post-Visual-Zustand). Dies ist erwartbar, da dieser Fokus mit einem eigenen Trainingssignal (250 Samples pro Fragevariation) gezielt optimiert wurde und zudem eine binäre Antwortstruktur („yes/no“) besitzt, die für das Modell im Training vergleichsweise leicht auszunutzen ist. In der Summe steigt dadurch der aggregierte Strategieanteil deutlich an, während die Gesamtleistung (`all`) und insbesondere `perception` leicht zurückgehen.

Demgegenüber verbleiben die Strategieaufgaben `best_next_move` und `reason_next_move`, welche die Bot-Policy nachspielen sollten in der Evaluation weiterhin bei 0.00.

Der Fokus `reason_next_move` ist als zusammengesetzte „Kür“ mehrerer zuvor eingeführter Teilfähigkeiten zu verstehen: Neben der vollständigen $15 \times 15$-Matrixausgabe umfasst er eine Gewinnprüfung (`CAN_WIN_NEXT_TURN`), eine Verlust-/Blockierprüfung (`CAN_LOSE_NEXT_TURN`) sowie anschließend die Wahl des nächsten Zugs im Format `row col`, analog zum Fokus `best_next_move`. Ziel dieser Konstruktion war es, einen direkten Vergleich zu ermöglichen, ob ein **chain-of-thought**-Reasoning die Imitation der Bot-Policy im Gegensatz zur einfachen Ausgabe der besten `row col` verbessert. Trotz dieser strukturierten Anleitung gelang es dem Modell jedoch nicht, das Entscheidungsverhalten des regelbasierten Bots zuverlässig nachzuahmen, dies zeigt sich gleichermaßen für `best_next_move` und `reason_next_move`.

Beide Fragefoki konnten vom Modell nicht erlernt werden. Damit liefert die explizite **chain-of-thought**-Struktur keinen messbaren Zusatznutzen gegenüber einer direkten Zugvorhersage ohne Zwischenrepräsentation.

Eine plausible Erklärung ist die unzureichende Abdeckung des Zustandsraums durch die Trainingsdaten: Angesichts der extrem großen Menge möglicher Brettkonfigurationen erscheint ein Trainingsumfang von lediglich $\approx 1\{,\}000$ Beispielen pro strategischem Fokus nicht ausreichend, um eine konkrete Bot-Policy robust zu approximieren, selbst wenn das Modell zuvor grundlegende visuelle Teilfähigkeiten (z. B. Zellklassifikation und lokale Mustererkennung) verbessert hat.

Darüber hinaus ist die Interpretation dieser Ergebnisse inhaltlich eingeschränkt: In `best_next_move` und `reason_next_move` wird das Verhalten des Bots als Ground Truth behandelt und damit implizit als „best“ angenommen. Da Gomoku jedoch kein gelöstes Spiel ist, ist „best“ hier nur relativ zum verwendeten Bot definiert. Folglich erlaubt die Evaluation primär Aussagen über die Imitationsgüte gegenüber der Bot-Policy, nicht jedoch darüber, ob die tatsächlich gespielten Züge des Modells durch die **chain-of-thought**-Struktur objektiv stärker oder schwächer geworden sind.

### *F. Curriculum-Learning*

Als abschließendes Experiment wurde ein Curriculum-Learning-Ansatz umgesetzt, um dem Modell visuelle Teilfähigkeiten schrittweise (von einfachen zu komplexeren Spielzuständen) zu vermitteln. Aufgrund zeitlicher Restriktionen sowie der Beobachtung, dass ein Großteil der für Gomoku relevanten Teilaufgaben primär visuell geprägt ist, wurde das Curriculum in der vorliegenden Umsetzung ausschließlich auf die Visual-Questions angewandt. Zudem erschien das Erlernen der Bot-Policy auf Basis von lediglich 1,000 Samples pro Fragefokus angesichts des großen Zustandsraums möglicher Spielpositionen als ohnehin nicht erreichbar. Die resultierenden Evaluationsergebnisse sind dennoch hinreichend, um zumindest die Erfolgsaussichten dieses Ansatzes unter den gegebenen Randbedingungen zu beurteilen.

a) *Grundidee und Progression über den Spielverlauf:*
Der Curriculum-Learning-Ansatz nutzt eine zentrale Eigenschaft der verwendeten Fragetyp-Architektur: Viele Fragefoki liegen in mehreren Variationen vor (z. B. Q1–Q4), die nicht nur unterschiedlich formuliert sind, sondern auch aus unterschiedlichen Rundenintervallen des Spielverlaufs gesampelt werden. Niedrige Varianten (Q1*) repräsentieren frühe Spielphasen mit geringer Steindichte, während höhere Varianten (Q4*) aus späteren, dichter belegten Spielphasen stammen. Für die mit Stern markierten Fragetypen entspricht die Question-ID einem festen Sampling-Intervall über den Spielverlauf: Q1* ($t \in [0, 30]$), Q2* ($t \in [31, 75]$), Q3* ($t \in [76, 150]$) und Q4* ($t \in [151, 224]$). Dadurch ergibt sich eine natürliche Progression entlang der Spielphasen.

b) *Kumulatives Sampling je Curriculum-Stufe:*
Auf dieser Grundlage definieren wir vier visuelle Curriculum-Stufen. Eine Stufe bedeutet dabei nicht, dass alle visuellen Fragefoki gleichzeitig trainiert werden, sondern dass pro Stufe nur ein definierter Teil der Visual-Questions verwendet wird. Innerhalb dieser Auswahl werden die Varianten-IDs kumulativ bis zur aktuellen Stufe berücksichtigt: In Stufe 1 werden Q1*-Samples genutzt, in Stufe 2 Q1*+Q2*, in Stufe 3 Q1*–Q3* und in Stufe 4 Q1*–Q4*. Dieses Prinzip gilt auch für Fragefoki, die erst in einer späteren Stufe neu hinzukommen: Ab ihrer Einführung werden unmittelbar alle bis dahin freigeschalteten Varianten (z. B. in Stufe 3 direkt Q1*–Q3*) mitgesampelt, um neue Fragetypen nicht ausschließlich in späten, besonders dichten Spielzuständen zu trainieren. So überwiegen in frühen Stufen frühe Spielphasen und einfachere Aufgaben, während in späteren Stufen zunehmend komplexere, stärker belegte Spielphasen und perzeptiv anspruchsvollere Aufgaben dominieren. Damit erfolgt das Training schrittweise entlang der Spielphasen und gleichzeitig entlang der inhaltlichen Komplexität der visuellen Aufgaben.

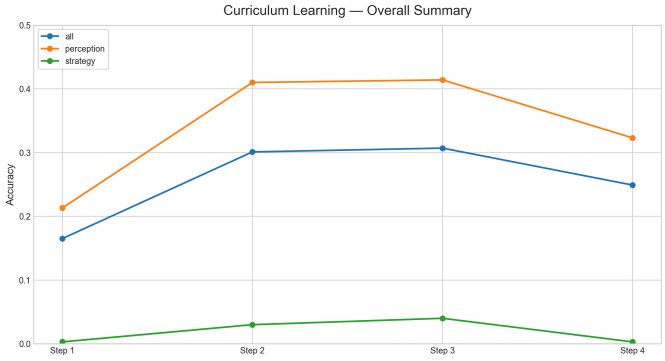
c) *Ausnahme: inverser Schwierigkeitsverlauf bei `count_empty_intersections`:*
Ein Fokus wird bewusst abweichend behandelt: `count_empty_intersections` weist eine inverse Schwierigkeitsprogression auf, da mit zunehmender Rundenzahl mehr Felder belegt sind und folglich weniger freie Schnittpunkte zu zählen bleiben. Dieser Fokus wird daher in umgekehrter Questionreihenfolge (Q4 $\rightarrow$ Q1) trainiert.

d) *Curriculum-Stufen und Rehearsal:*
- Stufe 1 (Early Game; Q1*). Visuelle Basisaufgaben in frühen Spielzuständen: `color_at_position` sowie grundlegende Zählaufgaben (`count_black_stones`, `count_white_stones`, `count_empty_intersections`).
- Stufe 2 (Mid Game; Q2*). Ergänzung um `three_in_a_row`, `four_in_a_row` sowie `print_board_matrix`, um Reihenstrukturen und die explizite Zustandsrekonstruktion frühzeitig zu adressieren, jedoch erst nachdem das Modell in Stufe 1 die elementare Zellklassifikation (`color_at_position`) erlernt hat.
- Stufe 3 (Mid/Late Game; Q3*). Ergänzung um `determine_who_won` zur Erkennung beendeter Spielzustände.

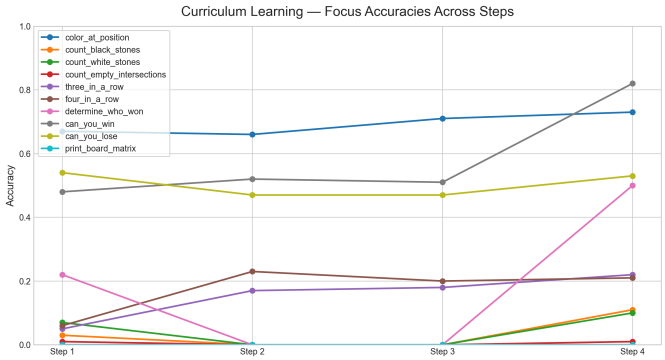
- Stufe 4 (Late Game; Q4*). Integration der visuell-taktischen Fragefoki `can_you_win` und `can_you_lose`, die zusätzlich zur Perzeption eine Bewertung von Threat-Strukturen erfordern.

Zur Reduktion von **catastrophic forgetting** wird in jeder Curriculum-Stufe ein kleiner Rehearsal-Anteil aus zuvor trainierten Stufen beibehalten. Konkret werden pro bereits behandelter Fragevariation jeweils 25 Rehearsal-Samples in den Trainingsmix aufgenommen, um erlernte visuelle Teilfähigkeiten während der Optimierung auf spätere Stufen zu stabilisieren. Mit dieser schrittweisen Progression untersuchen wir, ob sich die Modellperformance über unterschiedliche Spielzustände hinweg konsistent steigern lässt.

### G. Ergebnisse des Curriculum Learnings




### H. Evaluation des Curriculum Learnings




a) *Gesamtbild über die vier Stufen:*

Die Ergebnisse des Curriculum-Learnings zeigen keinen monotonen Leistungsanstieg über die vier Stufen, sondern ein dynamisches Bild: In mehreren Fällen steigen insbesondere jene Foki deutlich an, die in der jeweiligen Stufe neu eingeführt oder erstmals in größerer Breite trainiert werden, während einzelne zuvor erlernte Teilfähigkeiten gleichzeitig stagnieren oder temporär rückläufig sind.

b) *Stufe 2: Zugewinne bei Reihenaufgaben bei gleichzeitigem Einbruch der Zählfoki:*

Besonders gut illustriert dies Stufe 2: Hier verbessert sich das Modell spürbar in den neu hinzugekommenen Reihenaufgaben (`three_in_a_row`, `four_in_a_row`), während die Zählfoki (`count_*_stones`, `count_empty_intersections`) gleichzeitig deutlich einbrechen. Einfachere perzeptive Basisfähigkeiten wie `color_at_position` bleiben hingegen weitgehend stabil und sinken nur marginal von 0.670 auf 0.660. Zwar erholen sich einzelne Zählaufgaben in Stufe 4 teilweise, das Gesamtmuster ist jedoch konsistent mit einem **catastrophic forgetting**-Effekt, bei dem die Optimierung auf neu eingeführte Inhalte zuvor gelernte, vergleichsweise fragile Teilfähigkeiten überlagert. Dies deutet darauf hin, dass der gleichmäßige Rehearsal-Anteil von 10 % in den späteren Stufen für die Zählfoki nicht hinreichend war und einzelne „Basisfähigkeiten“ in der Trainingsverteilung gezielt stärker hätten gewichtet werden müssen (z. B. durch fokussiertes Oversampling der Zählaufgaben und/oder eine höhere Rehearsal-Quote speziell für diese Foki).

c) *Stufe 4: visuell-taktische Gewinne, aber kein proportionaler Effekt auf alle visuellen Foki:*

Ein weiteres, besonders klares Beispiel liefert hierfür Stufe 4: Mit der Einführung der visuell-taktischen Foki `can_you_win` und `can_you_lose` steigt `can_you_win` deutlich an (bis 0.82), während andere visuelle Aufgaben im selben Trainingsschritt nicht im gleichen Maß profitieren. So zeigen `three_in_a_row` und `four_in_a_row`, obwohl diese in Stufe 4 ebenfalls mit 250 Samples auf der höchsten Variantenstufe (Q4*) trainiert wurden gar keinen Zugewinn. Dies spricht dafür, dass die Reihenaufgaben in späten, dicht belegten Spielzuständen so viel schwieriger werden, dass das bereitgestellte Trainingssignal (250 Samples) im Verhältnis zur Aufgabenkomplexität nicht ausreicht, um die Leistung in diesem Schritt weiter anzuheben. Plausibel ist daher, dass diese Fragetypen in Stufe 4 überproportional hätten gewichtet werden müssen (z. B. durch gezieltes Oversampling der Q4*-Varianten und eine höhere Rehearsal-Quote für Reihenfoki in den vorherigen Schritten), um den Schwierigkeitssprung in den Late-Game-Zuständen auszugleichen.

### I. Einordnung des Curriculum-Learnings

a) *Aggregierter Vergleich zum Visual Fine-Tuning:*

In der vorliegenden Konfiguration liefert das Curriculum-Learning aggregiert über alle Fragetypen hinweg keinen klaren Vorteil gegenüber dem einfachen Visual Fine-Tuning (`all` von 0.284 nach Post-Visual Training gegenüber `all` von 0.249 nach Curriculum Step 4). Gleichzeitig zeichnen die Einzelergebnisse ein differenzierteres, teilweise vielversprechendes Bild: Sie deuten darauf hin, dass Curriculum Learning prinzipiell lernwirksam sein kann, in dieser Umsetzung jedoch eine deutlich vorsichtigere Konfiguration erfordert, wenn zugleich über Spielzustände und Fragefoki curriculisiert wird.

b) *Begründung der Einschätzung (lokale Gewinne vs. Forgetting):*

Diese Einschätzung stützt sich darauf, dass neu eingeführte bzw. in einer Stufe erstmals breit trainierte Fragetypen häufig unmittelbar stark zulegen, während parallel einzelne zuvor gelernte Teilfähigkeiten messbar zurückgehen (insbesondere in fragilen Basisfoki), was auf ein ungünstiges Verhältnis aus neuem Trainingssignal gegen Vergessen hindeutet.

c) *Implikationen für die Konfiguration:*

Insbesondere spricht das Muster dafür, dass (i) fragilere Basisfoki gezielt stärker abgesichert werden müssen, etwa durch fokussiertes Oversampling und/oder höhere Rehearsal-Quoten, und (ii) konservativere Lernraten notwendig sind, um lokale Zugewinne zu stabilisieren und die Netto-Performance über alle Foki hinweg konsistent zu verbessern.

d) *Limitierungen und Bedarf an Iterationen:*

Letztendlich ist festzuhalten, dass eine belastbare Bewertung deutlich mehr Iterationen erfordert: Die relative Schwierigkeit muss pro Fokus und pro Rundenintervall systematisch bestimmt werden. Zudem sind zentrale Hyperparameter, insbesondere die Lernrate, sowie der Trainingsdatamix (z. B. gezieltes Oversampling einzelner Foki und unterschiedliche Rehearsal-Konfigurationen) in mehreren Experimenten konsistent zu variieren und miteinander zu vergleichen, bevor abschließende Aussagen über Netto-Effekte möglich sind.

## References