Testing Components of the Attention Schema Theory in Artificial Neural Networks

東京大学松尾研究室
DLHacks
Kazuyuki Yoshikawa
「意識理論のAIへの応用」

自己紹介

- 兵庫県立大学工学部電気電子情報工学科4年
- 来年から兵庫県立大学大学院情報科学研究科

研究内容(人工知能、脳、量子コンピュータ、ロボティクス)

- 東京大学松尾研究室LLMATCH研究員で世界モデル
- 大学で量子コンピュータ、半導体の物理シミュレーション
- 大学院でEEG脳波解析×AI研究予定
- 東大発の神経科学×AIスタートアップで業務委託としてロボティクス、侵襲型ブレインマシンインターフェース開発

興味:人工意識

AIへの実装という観点から特に重要な4つの主要理論

- 「現象的意識（phenomenal consciousness）」...クオリア
- 「機能的意識（functional consciousness）」...注意制御、情報統合、自己モニタリングといった意識に関連する特定の認知能力を実装する
- グローバル・ワークスペース理論（GWT）
- 統合情報理論（IIT）
- 予測符号化（PP）
- 注意スキーマ理論（AST）←今回の論文

注意スキーマ理論（AST）

- 脳は身体の状態を表現する「ボディスキーマ」と同様に、自身の注意の状態を表現するための「注意スキーマ」を構築する
- この内部モデルによって、脳は自身の注意が今どこに向けられ、どのように変化しそうかを監視・予測し、内的に（トップダウンで）制御することが可能になる。
- 私たちが「意識している」と感じるものは、この注意スキーマが表象する情報

わかりやすく言うと

- 私たちが誰かと会話するとき、相手の視線や表情から「あ、今この話に興味を持っているな」と、相手の注意がどこに向いているかを無意識に推測
- 注意スキーマ理論とは、人間の脳がこのような高度な社会的やり取りを可能にするために、「注意」そのものについての単純化されたモデル（＝注意スキーマ）を持っているとする理論
- 自分自身の注意をコントロールするため:「今、集中できていないな」と客観視して、注意を向け直す。
- 他者の注意をモデル化するため:相手の注意状態を推測し、行動を予測することで、円滑なコミュニケーションや共同作業を可能にする。

Testing Components of the Attention Schema Theory in Artificial Neural Networks

- 本研究では、注意スキーマのようなモデルを人工エージェント（トランスフォーマーベースのニューラルネットワーク）に導入し、以下の点を検証

①他者の注意状態を判断・分類する精度が向上するか

②エージェント自身の注意パターンが、他者にとって分類しやすくなるか

③共同タスク（2体のエージェントがシーンを描く共同作業）において、相互予測・協調の性能が向上するか

(性能向上は単なるネットワーク複雑性の増加によるものではなく、「他者の注意を判断・分類・予測する」特定のタスクに関連していることが明らかになった。)

結果

- ①注意スキーマを持つ受信側ネットワークは送信側の注意データの分類精度が向上し、
- ②注意スキーマを持つ送信側ネットワークの注意は他者にとって判別しやすくなり、
- ③2体協調の描画タスクでも性能が向上した。
- これらの改善はネットワーク複雑性の単純増加では説明できず、注意関連タスクに特異的であった。

Methods（方法）

- **実験1**（他者注意のカテゴリ判断）
- **実験2**（注意スキーマ効果が単なる複雑性増加かの検証）
- **実験3**（2体協調の塗り絵タスク）

実験1
AIは他AIの「集中度」を見抜けるか？

- 目的: 注意スキーマを持つことで、「他AIの注意状態を判断する能力」と「他AIから判断されやすくなる能力」が向上するかを検証

- エージェントは最初に視覚的な分類タスク(たとえば、写真をゴルフボールやゴミ収集車を示すものとして分類する)でトレーニング
- 次に、最後のブロックを除くすべての層の重みを凍結し、新しいタスクである「注意判断」タスクでエージェントをトレーニング

- このタスクにより、あるエージェントが別のエージェントの注意状態について決定を下す能力をテストする

- 方法:2種類のAI（送信側・受信側）を用意し、「注意の伝言ゲーム」タスクを行わせた。

- 送信側AI (Sender): 画像分類タスク（例：「ゴルフボール」か「ゴミ収集車」かを見分ける）を行う。その際、画像のどこに注意を向けていたかのデータ（注意テンソル）を記録。
- 注意データの加工: 送信側AIが生成した注意データには、「本物（Veridical）」と、データをランダムにシャッフルした「偽物（Scrambled）」の2種類を用意。

- 受信側AI (Receiver): 送信側から送られてきた注意データが「本物」か「偽物」かを見分ける訓練をする。これができれば、そのAIは「正常な注意のパターン」を理解していることになる。

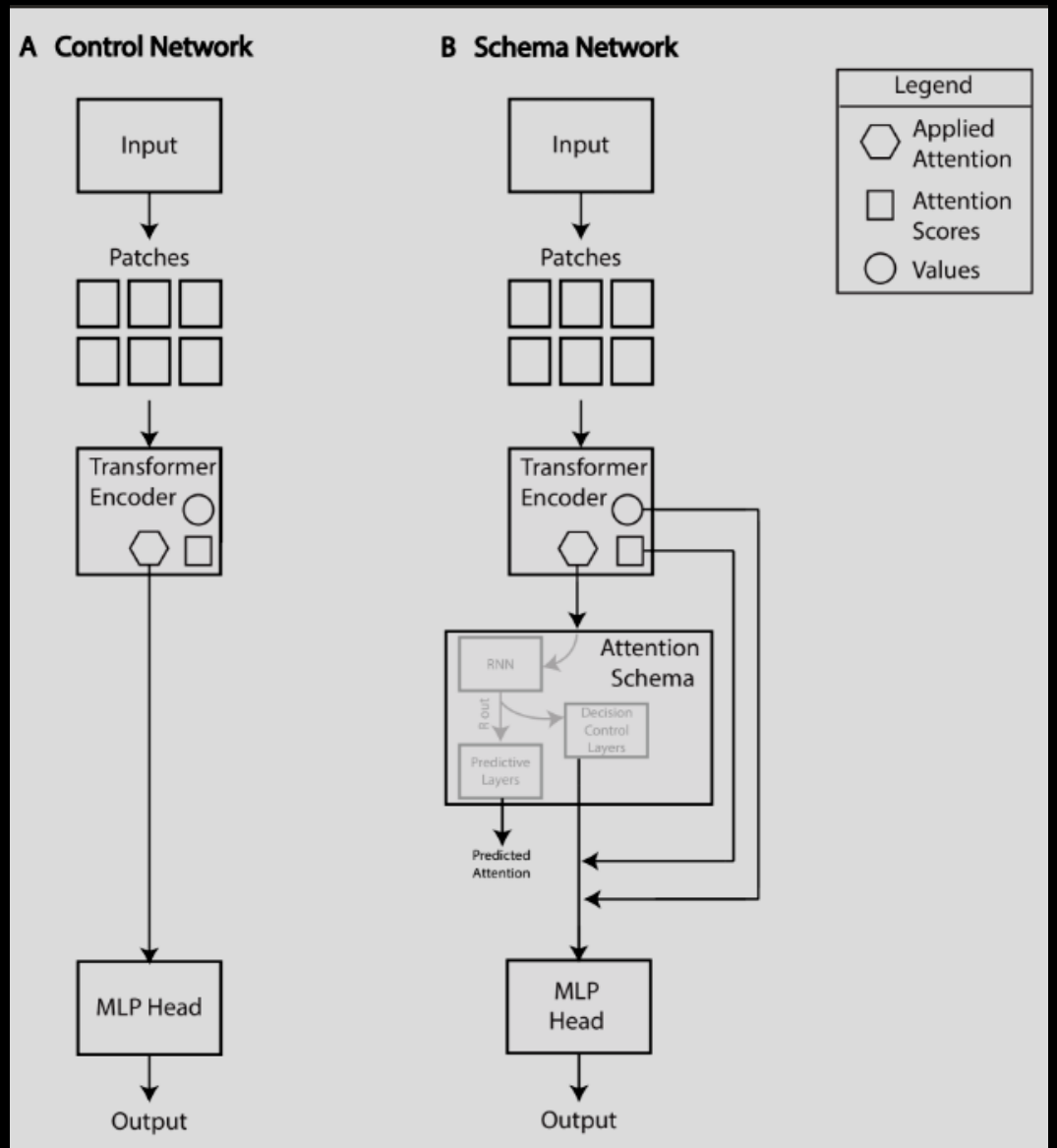
- 送信側・受信側のそれぞれに「注意スキーマあり/なし」の全4パターンで比較。A: スキーマあり → スキーマありB: スキーマあり → スキーマなしC: スキーマなし → スキーマありD: スキーマなし → スキーマなし

2つの仮説

- 自分の注意をモデル化する訓練をしたAIは、その能力を応用して、他AIの注意状態を理解し、予測する能力も向上するのではないか？

- 自分の注意をモデル化する過程で、AIは自分自身の内部状態をよりシンプルで規則正しい（＝予測しやすい）ものに再構築するのではないか？その結果、他AIにとって「分かりやすく、意図が読み取りやすい」存在になるのではないか？

1A...制御アーキテクチャ(制御ネットワーク:アテンションはあるがアテンションスキーマはない)
1B...アテンションスキーマを追加したアーキテクチャ(アテンションスキーマネットワーク)
「Attention Schema」ブロックは、AI自身の「注意」がどこに向いているかを一度観測し、「次にどこに注意が向きそうか」を予測します（Predicted Attention）。
その予測を使って注意をより効率的な状態に修正してから最終的な判断を下します。
一言でいうとスキーマありのAIは、自分自身の注意状態を客観視して改善する「自己分析」機能を持っている

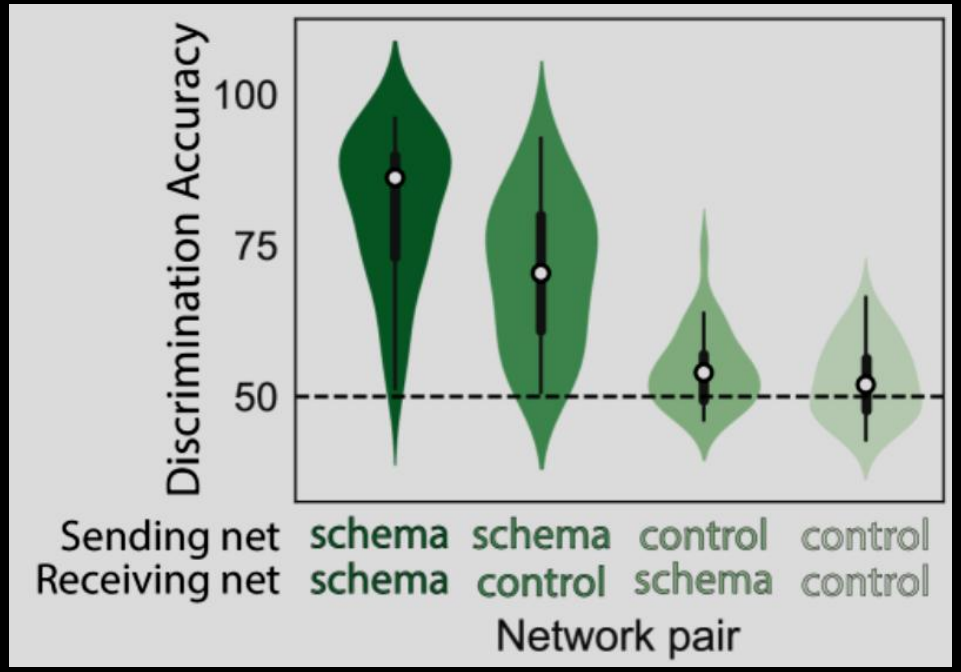


（実験1の前処理）

- 画像入力：256×256×3。
- 画像二値分類タスクを3種（A: パラシュート vs ベンチ、B: フレンチホルン vs カセットプレーヤー、C: ゴルフボール vs ごみ収集車）
- いずれもImagenetteからサブサンプル。
- 学習データ 1,912枚・テスト 730枚、20エポック学習。
- 注意テンソルは、6ヘッド中ランダムに3ヘッドを抽出し、257×257×3（16×16パッチ＋CLSトークンで257）に積層。
- 真値（veridical）は256×256×3に縮約、スクランブルは最終次元をランダム置換して256×256×3に縮約。
- 送信側1ネットあたり1,460テンソル（真値/スクランブル半々）を作成。
- 受信側は画像分類で事前学習後、最終MLP以外を凍結し、注意テンソル1,314枚（半々）で800エポック学習、146枚で評価。
- 受信側は元の画像タスクと異なるタスク由来のテンソルで学習（交差設定）。9回反復×3タスクで27点の精度を得る。
- 注意スキーマ有りモデルは主タスク:注意予測の損失重みを0.95:0.05に設定。

結果

- 送信・受信の両方がスキーマ有り：80.87%（SEM 2.38）。
- 送信のみスキーマ有り・受信なし：70.61%（SEM 2.37）。
- 送信なし・受信のみスキーマ有り：54.58%（SEM 1.25）。
- 両方なし：52.78%（SEM 1.21）。
- ANOVA：受信側主効果 F=10.17, p=0.0018（検出力 0.77）、送信側主効果 F=136.18, p≈1.28×10^-20、交互作用 F=4.99, p=0.027。最大効果は送信側のスキーマ有無。
- **F値が大きいほど**: その効果が偶然ではなく、**本物である可能性が高い**
- **p値が小さいほど**:「こんなことは偶然ではまず起こらない！」と言えるため、**結果の信頼度が高い**



仮説について

- 受信側: 注意スキーマを持つと、持たない場合に比べて正答率が上がった（左端 vs 左から2番目、右から2番目 vs 右端）。これは仮説1（他者理解力の向上）を支持する
- 送信側: 注意スキーマを持つと、その注意データは受信側にとって格段に判断しやすくなった（左2つ vs 右2つ）。この効果は非常に大きく、仮説2（自己の透明化）を強く支持する 。

- 分かったこと: 注意スキーマは、他者を理解する能力を向上させるだけでなく、自分自身を他者にとって「分かりやすい存在」に変える効果がある

実験2：性能向上は「ただ複雑になっただけ」ではないか？

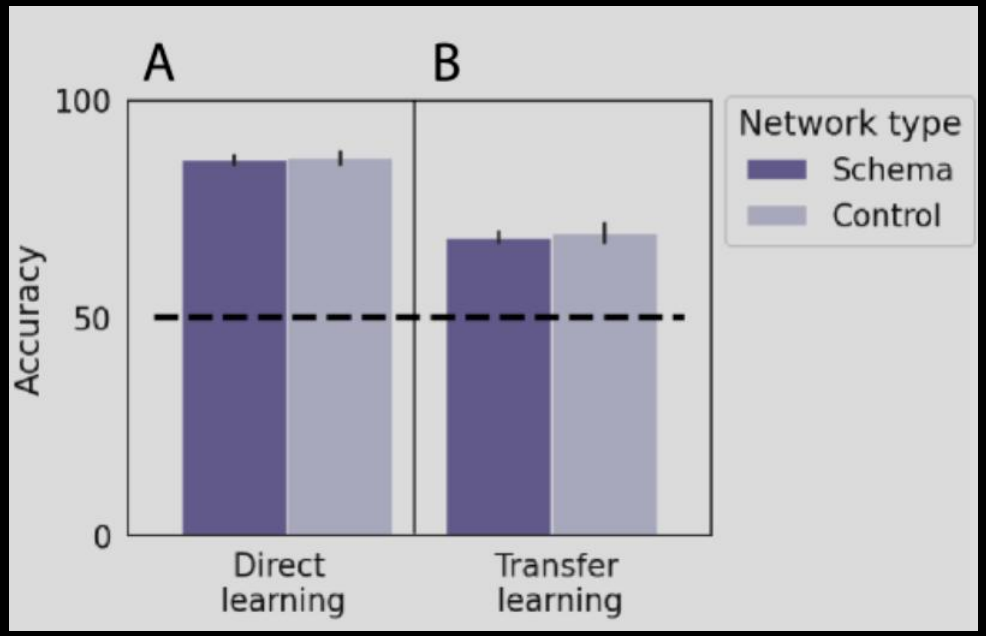
- 目的: 実験1の結果は、単に注意スキーマという部品を追加したことでAIが複雑になり、全般的に賢くなっただけではないか、という疑問を解消する。

- 方法:AIに、実験1とは異なる別の画像分類タスクを学習させる（転移学習）。
- もし注意スキーマが汎用的な性能向上をもたらすなら、このタスクでも「スキーマあり」の方が良い成績を収めるはず

結果

- 注意スキーマの有無によって、転移学習の成績に有意な差は見られなかった
- 注意スキーマによる性能向上は、注意パターンの判断のような「社会的」なタスクに特化したものであり、単にネットワークが複雑になったことによる副次的効果ではない
- 画像分類→別画像分類への転移でも有意差なし：**t=-0.74, p=0.46, d=0.20**（事後検出力 **0.111**）。
- → 改善は「注意関連」特異効果であり、単なる複雑性増大の一般効果ではない。

- A. 実験１におけるスキーマ対制御ネットワークの画像分類タスクの平均精度。有意差は見られなかった。
- B. 実験結果 2の転移学習(1つの画像分類タスクで事前学習し、その後2番目の画像分類タスクに転移学習)におけるスキーマと制御ネットワークの平均精度。有意差は見られなかった。
- A・Bのどちらのグラフを見ても、スキーマあり（濃い紫）とスキーマなし（薄い紫）の成績にほとんど差がない



実験3

注意スキーマ理論の中心的な仮説...人々において、注意スキーマの存在により、社会的認知が向上し、協力的なタスクでのパフォーマンスが向上する

その理由は、単純で基本的なレベルでは、注意が行動に影響を与えるため、注意状態を予測することで行動をより適切に予測できるため

実験3：共同作業「ペアでお絵描き」はうまくいくか？

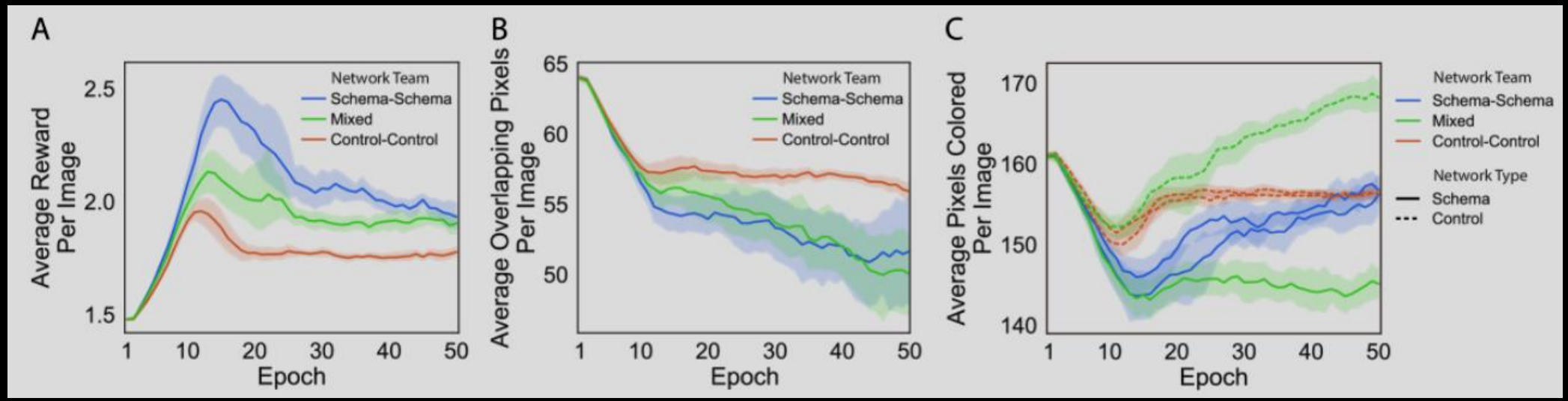
- 方法:2体のAIに、1枚の画像を共同で塗り絵するタスクを与えた。
- ルール:新しく塗れたピクセルが多いほど、チームの得点が増える。しかし、パートナーと同じ場所を塗ってしまうと、ペナルティで得点が減る。各ターン、相手がどこを塗るかは分からないため、相手の次の行動を予測することが高得点の鍵となる。

- このタスクを、以下の3つのチーム構成で比較
- スキーマ・スキーマチーム: 両方のAIが注意スキーマを持つ。
- 混合チーム: 片方だけが注意スキーマを持つ。
- コントロール・コントロールチーム: 両方とも注意スキーマを持たない。

詳細

- 画像ベースMARL環境で「塗り絵」タスクを設計。
- 10カテゴリからの2,000枚画像プール。
- 各ターンで両ネットがピクセル群を着色。過去の塗り履歴とどちらが塗ったかは観察可能だが相手の当ターン選択は非公開。
- 報酬：新規ピクセル数を係数0.5で加点、重複ピクセル数を係数1.7で減点。
- エピソード終了：全ピクセル塗り終え or 画像ピクセル数と同数のターンに達した時点。
- チーム構成：スキーマ×スキーマ、混成、コントロール×コントロールの3種。
- 学習：各構成を50エポック、各エポック6,000ターン、テストは15回繰り返し。

結果(シェーディングは95%信頼区間)
A. 各画像のカラーリング中に発生する平均報酬。高いほど良い
B. 各画像の重なり合うピクセルの平均量。低いほど協調性が高い
C. ペア内の各ネットワークについて、画像ごとに色付けされた平均ピクセル数。各AIが個々に塗ったピクセル数で、個々の戦略を示す



- チームの得点 (図4A): スキーマ・スキーマチームが最高得点を記録し、次いで混合チーム、コントロール・コントロールチームが最も低い結果。

- 協調性（オーバーラップの少なさ, 図4B）:
- スキーマを持つAIがいるチーム（スキーマ・スキーマ、混合）は、コントロールのみのチームに比べてピクセルの重複が少なく、うまく連携できていることが分かった

- 個々の行動 (図4C):
- 混合チーム(緑)の非対称性: 学習が進むと、スキーマを持たないAI（破線）が積極的に多くのピクセルを塗り、スキーマを持つAI（実線）は塗る量を抑えるという役割分担が見られた。
- スキーマを持たないAIがスキーマを持つAIの行動が「予測しやすい」ことを学習し、それをうまく利用して効率的に得点を稼いでいたことを示唆している。

まとめ

注意スキーマは、

- (i) 自己注意の予測学習を通じて他者注意のモデル化を促進し、
- (ii) 自分自身の注意パターンを単純化・規則化して他者から解釈されやすい表現に再構成される
- という2つの経路で「社会的」パフォーマンスを高める可能性がある。

Limitations（限界）

- 協調タスクはシンプルな**塗り絵**であり、人間の高度な社会認知を直接一般化できるとは限らない。
- 画像分類や転移では有意差なしであり、広範な一般化には更なる検証が必要。

再現実装URL

- kathrynfarrell/attention_schemas_in_anns

- 実行結果
- https://github.com/K0909github/consciousnessai-attention-schema-theory-

関数(1番目のセル)

- ## 1. schematrain
- 役割: 「スキーマありAI」専用
- 何をするか: データのごく一部（1バッチ）を使って、スキーマありAIに2つのことを同時に学習させる
- 画像分類: 画像がどのカテゴリに属するかを正しく当てる練習。
- 自己注意の予測: 自分の「注意」がどこに向いているかを予測する練習。
- プロセス: AIの間違い（損失）を「画像分類の間違い」と「注意予測の間違い」の2種類で計算し、その合計スコアを基にAIのパラメータを更新（賢く）する

- ## 2. schematrain_policy
- 役割: 「スキーマありAI」の転移学習（応用問題）専用
- 何をするか: 画像分類の正解を当てる練習だけに集中。
- なぜ必要か: 転移学習の際は、AIの大部分の知識は固定（凍結）されており、応用力だけを鍛えるため、学習内容をシンプルにする
- ## 3. controltrain
- 役割: 「スキーマなしAI」専用
- 何をするか: データの一部を使って画像分類を正しく当てる練習だけをさせる
- schematrainとの違い: 自己注意を予測する機能がないため、学習内容が画像分類の一つだけ

- ## 4. fitschema
- 役割: 「スキーマありAI」の学習全体
- 何をするか: 指定された回数（エポック数）、以下のサイクルを回して学習全体を管理。
- 訓練データ全体をいくつかの小さなバッチに分ける。バッチごとに（schematrain関数）を呼び出してAIを少しずつ賢くさせる。データ全体を1周見終えたら、検証用データで「理解度テスト」を行い、成績を記録する。最も成績が良かった時点のAIの状態を記憶しておき、最終的にその一番賢い状態をモデルとして採用する。
- ## 5. fitcontrol
- 役割: 「スキーマなしAI」の学習全体
- 何をするか: fitschemaと全く同じだが、呼び出す関数がcontroltrain関数
- ## 6. evaluate
- 役割: 学習が完了したAIの最終成績を採点する
- 何をするか:学習済みのAIに、未知のテストデータ（valloader）を次々と見せる。計算した正答率をテキストファイル（accuracy_...txt）に保存
- ## 7. freeze_models
- 役割: 転移学習の準備をする
- 何をするか:学習済みAIのパラメータ大部分を「凍結」させ、これ以上変化しないようにロック。ただし、最終的な判断を下す部分（policy）だけは凍結せず、学習可能な状態のままにする。
- なぜ必要か: これにより、AIはこれまでに得た豊富な知識を維持したまま、その知識を応用して新しいタスクに素早く適応（転移学習）できるようになる

実験の実行 (2番目のセル)

- ### for trial in range(0, 20):
- 結果が偶然でないことを確認するため、同じ実験を20回繰り返す。
- ### ① 通常学習 (Direct learning)# A, # B, # C:
- まず、まっさらなAIモデルを6つ（A, B, Cの各タスクに対し、スキーマあり/なしの2種類）用意。それぞれを各タスク（AのモデルはAのデータセット、BはBのデータセット…）で20エポック訓練し、性能を評価。ここでの結果が論文の図3Aに相当し、AIの基本的な分類性能を示す
- ### ② モデルの凍結 (FREEZE THE MODELS)
- 通常学習が終わった全モデルの大部分をfreeze_models関数で凍結。これにより、AIの基本的な知識は固定され、最後の分類部分だけが再学習できる
- ### ③ 転移学習 (TRANSFER LEARNING)
- ここで実験2を行う。例えば、タスクA（パラシュート vs ベンチ）を学習したAIに、全く新しいタスクB（フレンチホルン vs カセット）を学習させる。この結果が論文の図3Bに相当

結果
(1枚目の画像は239バッチ中の学習、数値はそこまでの平均混乱スコア)

```
0: 0.679358959197998 / 239: 0.002842506105430954
1: 0.8153344988822937 / 239: 0.006253947523348501
2: 0.659233570098877 / 239: 0.009012246979829157
3: 0.7260761260986328 / 239: 0.01205022240283599
4: 0.6982764005661011 / 239: 0.014971880982610472
5: 0.7147909998893738 / 239: 0.017962638304323333
6: 0.6804316639900208 / 239: 0.02080963271432342
7: 0.7297563552856445 / 239: 0.023863006585811475
8: 0.6901165843009949 / 239: 0.026750523674936976
9: 0.6755895614624023 / 239: 0.02957725824172527
10: 0.6857888102531433 / 239: 0.03244666748964637
11: 0.6495785117149353 / 239: 0.035164569212303
12: 0.6728949546813965 / 239: 0.03798002927373144
13: 0.664060652256012 / 239: 0.040758525726685466
14: 0.6826902627944946 / 239: 0.04361497034088837
15: 0.6372897624969482 / 239: 0.04628145470280029
16: 0.6490436792373657 / 239: 0.04899711863266375
17: 0.6892343163490295 / 239: 0.05188094422408228
18: 0.5775551795959473 / 239: 0.05429749309268457
19: 0.6542345285415649 / 239: 0.057034876057293626
20: 0.5902770161628723 / 239: 0.05950465436759853
21: 0.6247652769088745 / 239: 0.06211873502412102
```

```
237: 0.22905240952968597 / 239: 0.436318578705129
238: 0.44832080602645874 / 239: 0.4381943979772063

Epoch : 1, train loss : 0.4381943979772063
Epoch : 1, val loss : 0.31440070725005603

Epoch : 2, train loss : 0.2627824199226172
Epoch : 2, val loss : 0.2257671982616834

Epoch : 3, train loss : 0.20634215690632
Epoch : 3, val loss : 0.26970603701699036

Epoch : 4, train loss : 0.18193008441859893
Epoch : 4, val loss : 0.2474006537510001

Epoch : 5, train loss : 0.1537442439777394
Epoch : 5, val loss : 0.21158040699589506
```