**Motivation**

Predicting the next raw pixel output $y$ from a raw pixel input $x$ is hard. If we were to do that for a system of bouncing balls, knowing that an algorithm underlies this task should help us in some way. With a suitably abstractified data $\tilde{x}$, predicting the future abstract state $\tilde{y}$ could be as easy as running a force calculation algorithm.

Though this abstraction seems to simplify the path from $x$ to $y$, it convolutes our efforts as now we need to take care of a bigger pipeline $x \rightarrow \tilde{x} \rightarrow \tilde{y} \rightarrow y$ where $x \rightarrow \tilde{x}$ requires the knowledge of right abstraction or a massive paired dataset to learn the mapping $\tilde{x} \rightarrow \tilde{y}$ implies a perfect algorithm, which in reality we might not have.$\tilde{y} \rightarrow y$ calls for a differentiable renderer or a massive paired dataset to learn the mapping.

---

**Architecture**

Two-stage encode-process-decode

1st stage: train the $x \rightarrow \tilde{y}$ pathway with the encode-process-decode architecture $x \xrightarrow{f} z \xrightarrow{\tilde{P}} z' \xrightarrow{\tilde{g}} \tilde{y}$

- encoder $f$: learns to map abstract inputs $\tilde{x}$ into a high-dimensional latent $z$
- processor $\tilde{P}$: learns a “neural executor” in a high-dimensional space
- decoder $\tilde{g}$: learns to map the high-dimensional latent $z'$ into the abstract output $\tilde{y}$

$P$ is now a differentiable module that learned to simulate $x \rightarrow \tilde{y}$ in a high-dimensional space.

2nd stage: train the $x \rightarrow y$ pathway with the encode-process-decode architecture $x \xrightarrow{f} z \xrightarrow{P} z' \xrightarrow{g} y$

where we swapped out abstract encoders and decoders for natural ones

- encoder $f$: learns to map pixel inputs $x$ into the high-dimensional latent $z$
- processor $P$: frozen from the previous step to retain the semantics of its mapping
- decoder $g$: learns to map the high-dimensional latent $z'$ into the pixel output $y$

---

**Results**

- Significantly better than C-SWM on 12 / 19 games
- Indistinguishable on 7 / 19