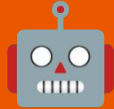




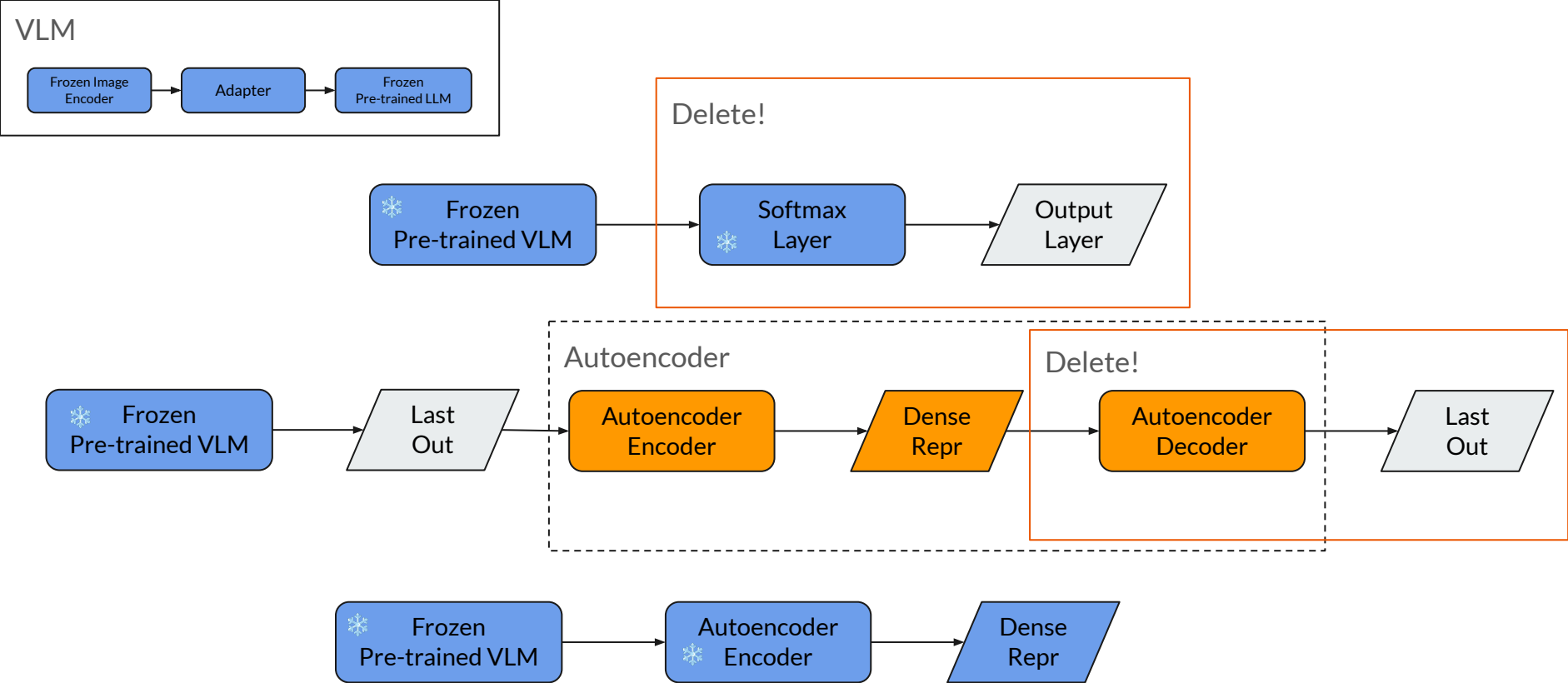
# Environment Encoder

A proposal for having agents interact with world models meaningfully

# Example Model Architecture Diagrams

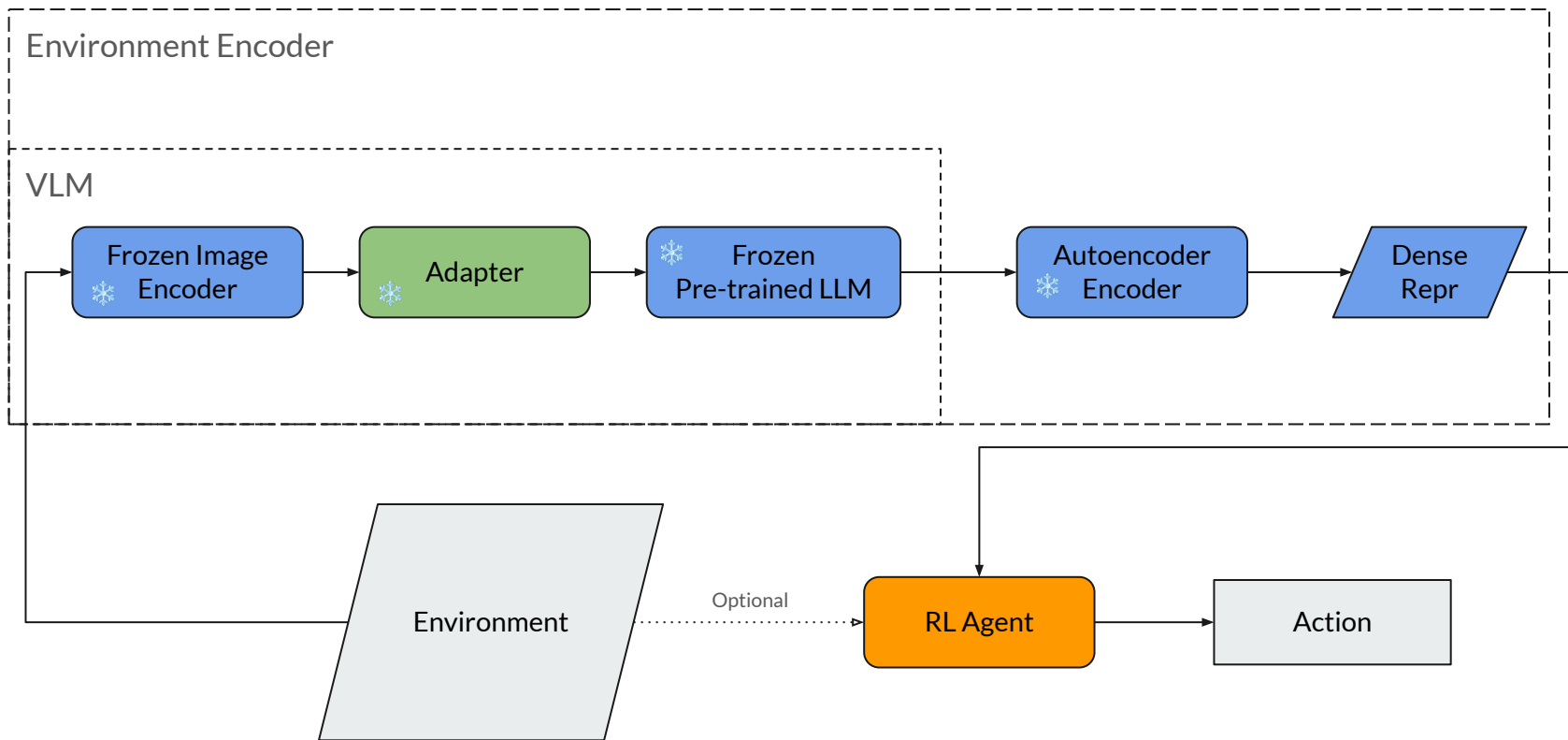


---



Stage 1 Training Architecture

\*The deletion happens in the final LLM section of the VLM



Stage 2 Training Architecture

# Concerns 🦷

---



## Two main questions come to mind

- **Why not use [INSERT VLM]?** You could! Any VLM would work, Llava seems like it would be a good choice with just its linear adapter layer, but we'll need to investigate further.
- **Won't the latent space look very similar (if not identical) from frame-to-frame?** Actually the autoencoder helps fix this, since the compression (should) disregard these regularly repeating pieces of information. It would help to train our autoencoder of sequence by sequence frames of games we expect to play.

The Gist? 🤔

—



## The training procedure consists of six key points

1. We chop off a VLM's unembedding and output layers.
2. An autoencoder is attached to the end of the VLM.
3. The autoencoder is trained alongside a frozen VLM to compress the vectors into a dense representation. (Training should be multiple sequential frames fed to our VLM).
4. We chop off the decoder section of the autoencoder and preserve the dense representation.
5. A VLM is finetuned on a set of image-advice pairs that provide advice and short explanations for a given game's frame.
6. The whole thing is put in sequence and we feed the dense representation to an RL agent alongside its environment during training and inference.

*We could optionally train a VLM's linear layer / Qformer alongside RL agent.*



# The Cool Stuff 🙄

---



## There's three interesting things we might pull off 🙄🙄

1. We might not need to feed the environment to the RL agent! Maybe the compressed latent space is enough.
  - a. That could mean it would generalize over multiple environments!
2. Leveraging the VLM as the world model, we should converge faster or reach a better solution. (or at least reduce sensitivity to hyperparameters).
3. This might solve the idea of tuning for reward functions if we include a transformer as a 'reward model' (see interesting ideas section)

# Interesting Ideas

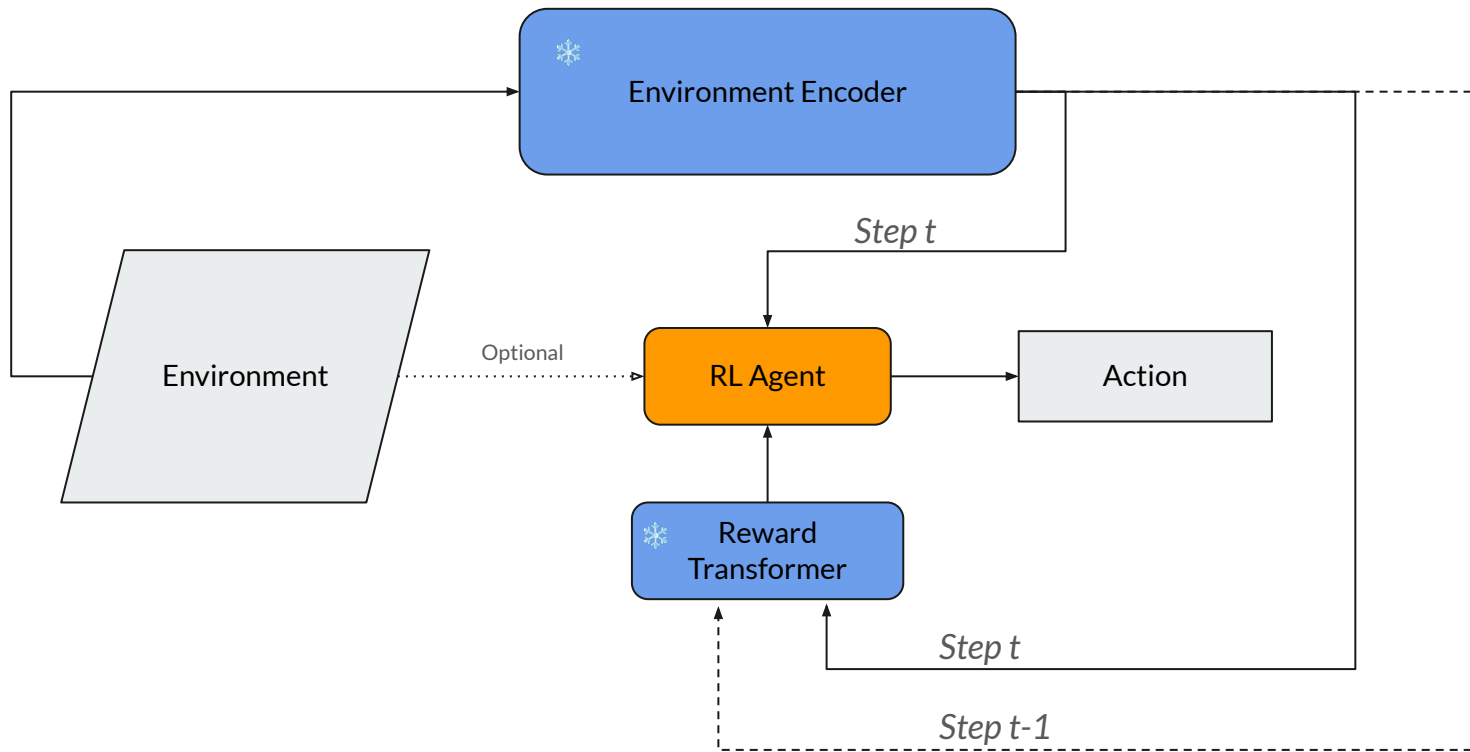
(They might not be the best)



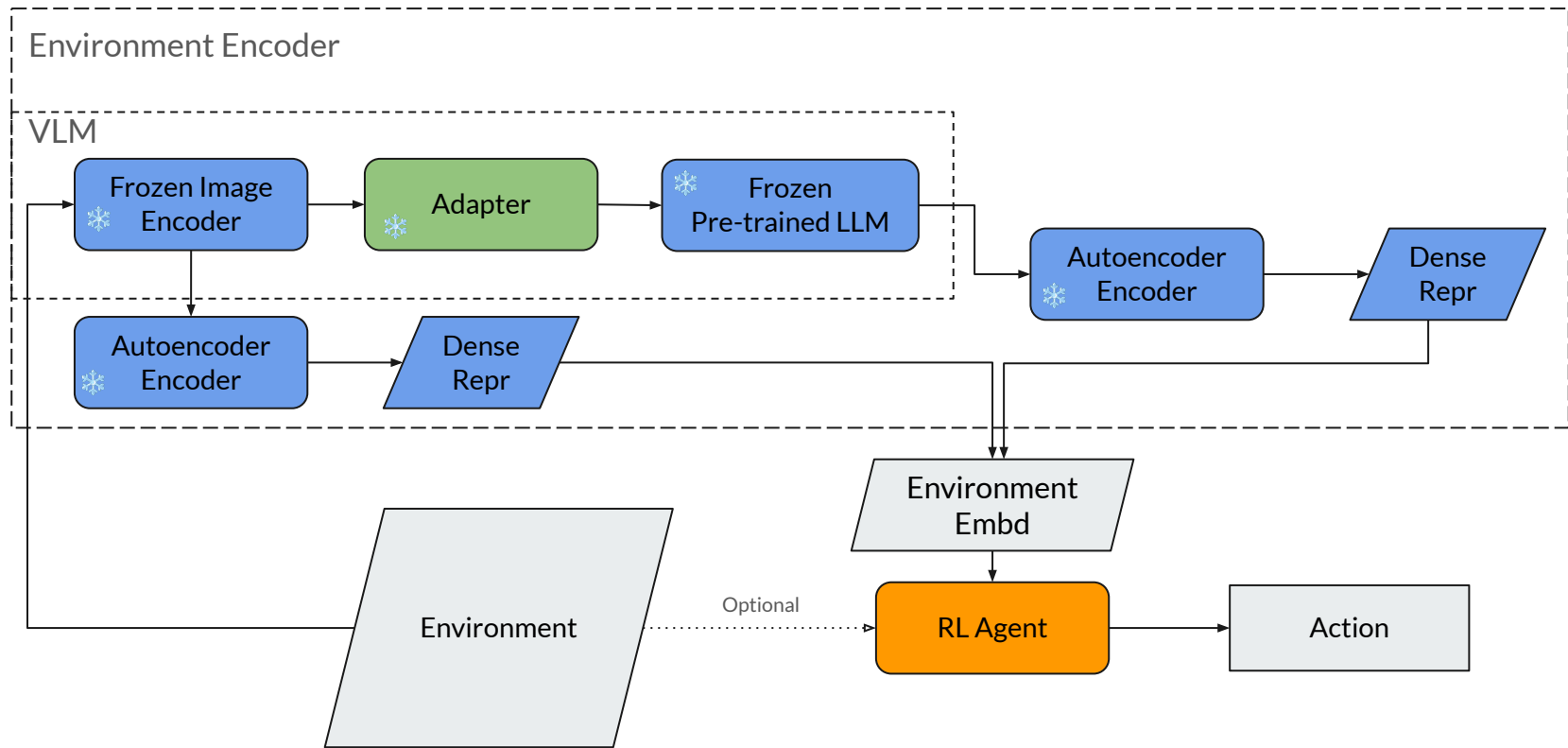


## A few ideas could be possible

1. Leveraging transformers as reward models.
2. Apply an autoencoder to compress the output of the image encoder and feed that alongside our current latent space to the RL model as one huge matrix.
3. Linear layer might be enough instead of an autoencoder compressing everything.
  - a. However, this is unlikely because the learning objectives are vastly different.



Using an LLM as a reward model? We could train it on a preference dataset generated as {input frame, loser frame, winner frame}. During inference it will take *step t-1* as input and *step t* to provide a reward to, so it can determine the action taken.



Adding an autoencoder to our image encoder? Then we'd have a holistic environment embedding? But also maybe this is just redundant.



## Related Works

1. <https://arxiv.org/pdf/2212.08860>
  - a. They didn't utilize any LLMs.
2. <https://arxiv.org/pdf/2310.17722>
  - a. They use it for embodied tasks, they have the idea of embedding stuff. We can benchmark against it!

# Implementation Details: Possible Stage 2 Training Methodologies.

Very subject to change.

---





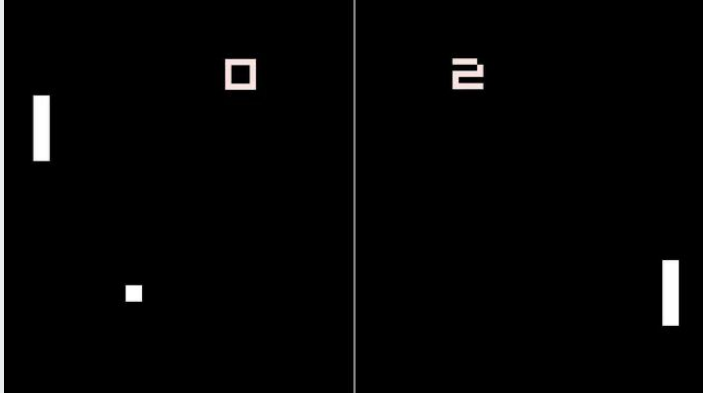
## 1. Use Pretrained VLM Without Further Finetune

- We could just use a pretrained VLM directly and it might just work.
- Directly train RL agent on latent space extracted as environment.



## 2. Finetune Pretrained VLM

- Supervised training set of images and appropriate best moves and explanations to each image as labels.
- We can generate a synthetic PoC version of the data using GPT-4V or Llava-Llama3-8b.
- Directly train RL agent on latent space extracted as environment.

Image	Label
	<p>A game of pong, the score is 2-0, the left player should move down</p>

2: Example Dataset



### 3. Train VLM Alongside RL Agent

- Train during RL simulation.
- Apply loss to VLM adapter alongside agent.
- Could happen after finetune or without.
- Directly train RL agent on latent space extracted as environment.

Least ideal approach?



# Acknowledgements

- Thank you Faisal for criticising me really hard and making this better.
- Thank you Hisham for helping out with the related works.

If you read this, please be really harsh and send me any criticism!

I still have not done the rigorous research to back up all these claims perfectly, I still need to do an in-depth literature review.