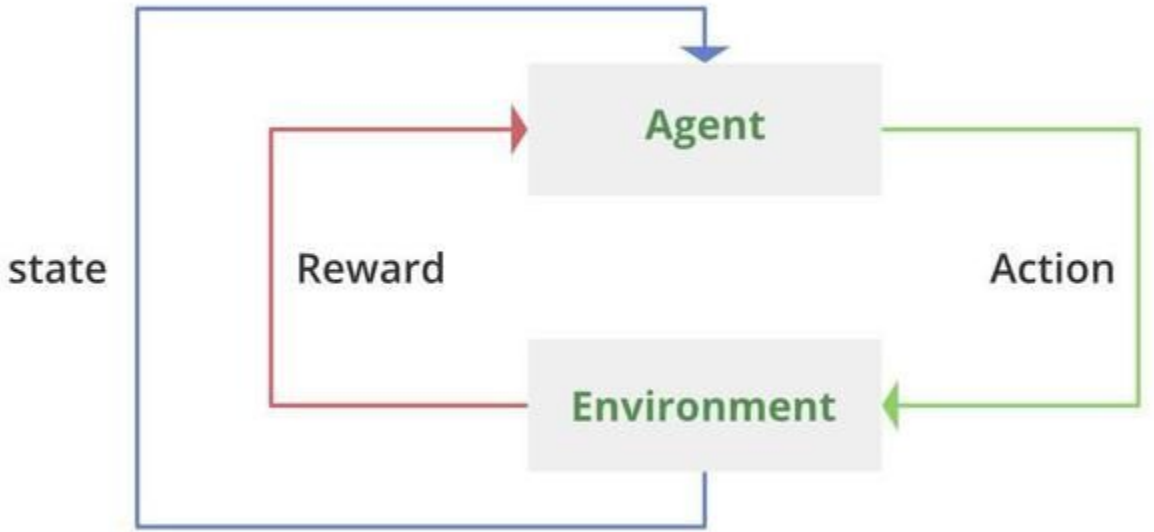
Efficient Pipeline Parallelism for Reinforcement Learning

Faiz Ahmed

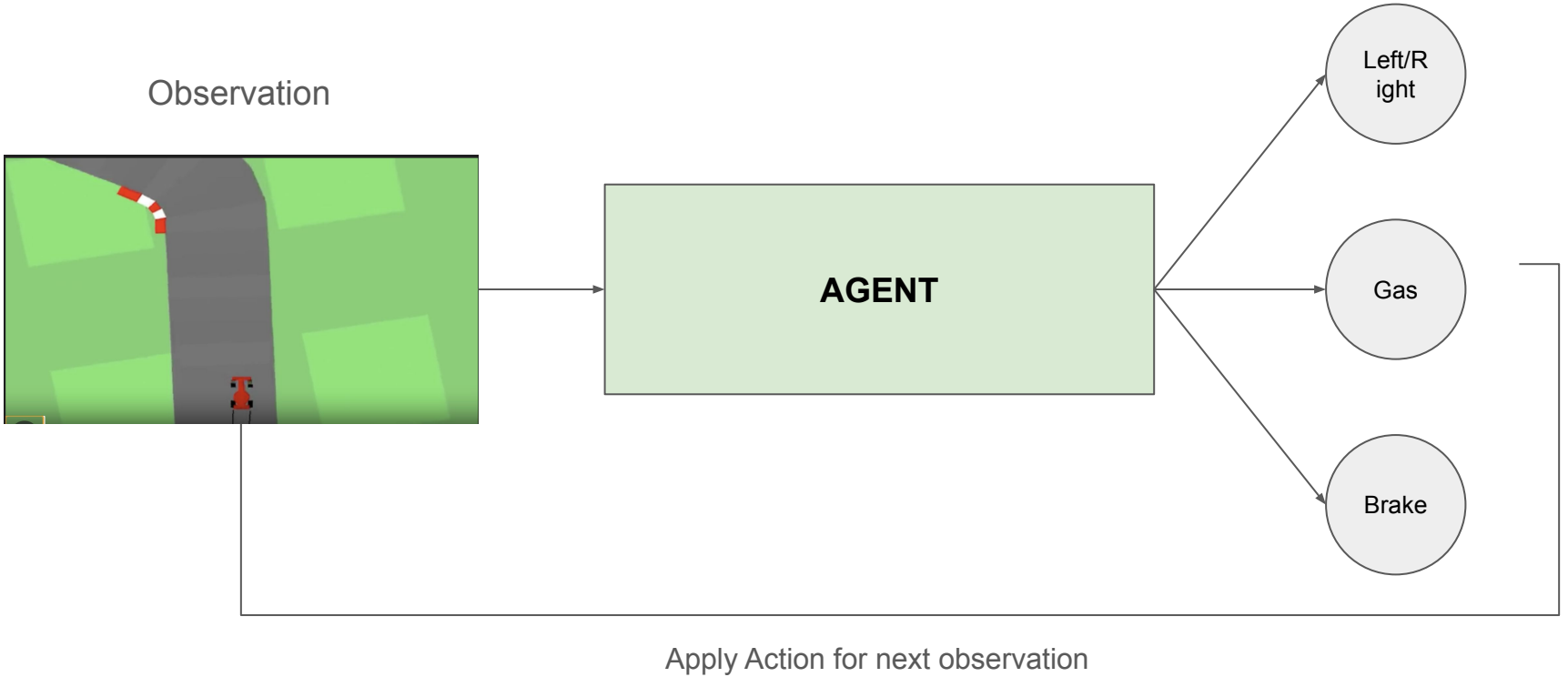
Reinforcement Learning (RL) is seeing growing adoption for various tasks

What is Reinforcement Learning?

- RL is a family of algorithms that allow **learning an objective without training data.**



Zooming into the agent

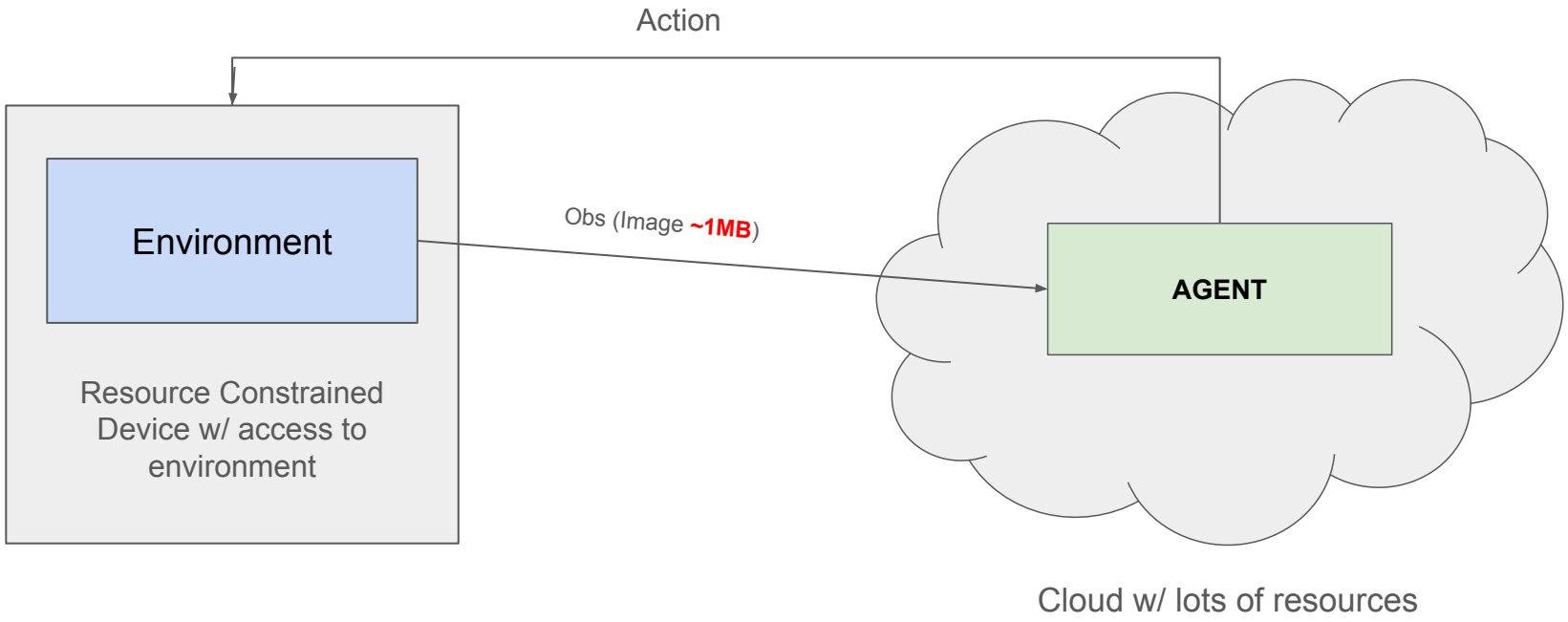


- RL Models require a lot of training before they become good (converge)

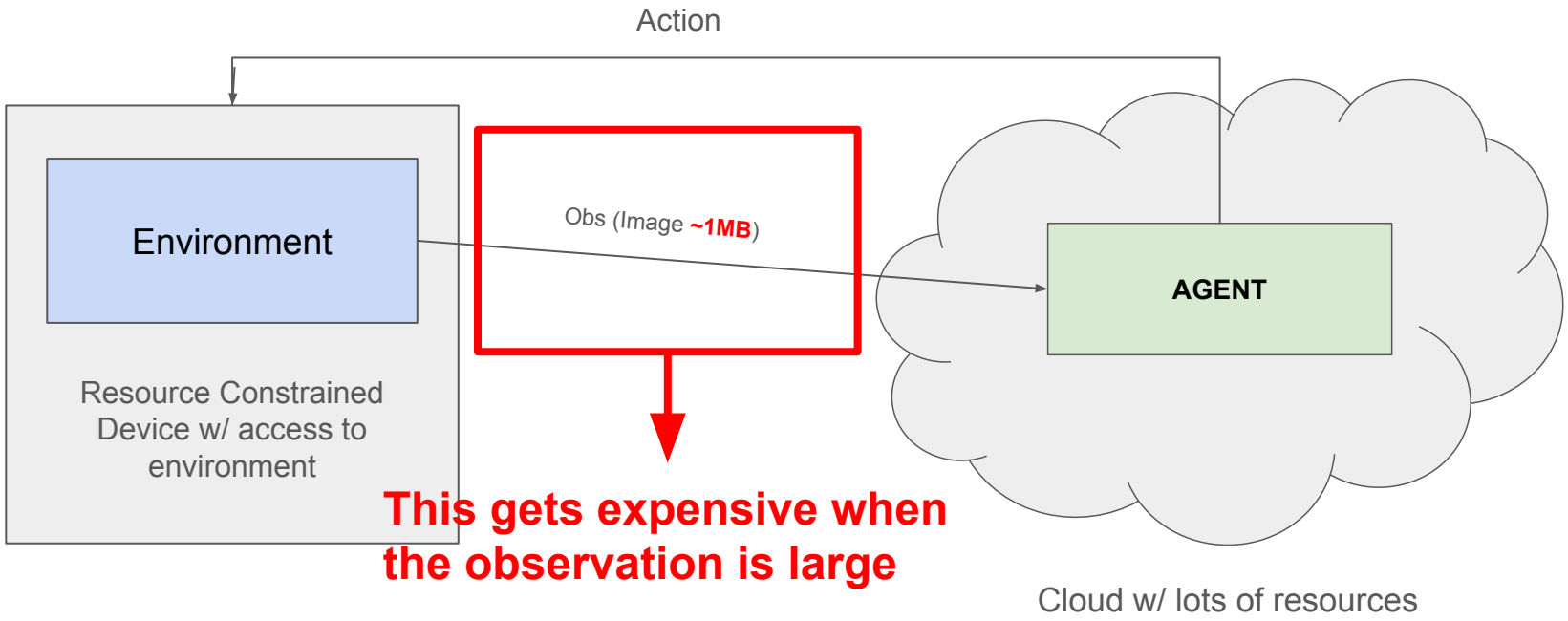
- Complex environments require large resources to train the “Agent”

- So how do we train RL in scenarios where the environment is resource constrained?

Run the Agent Separately on the Cloud



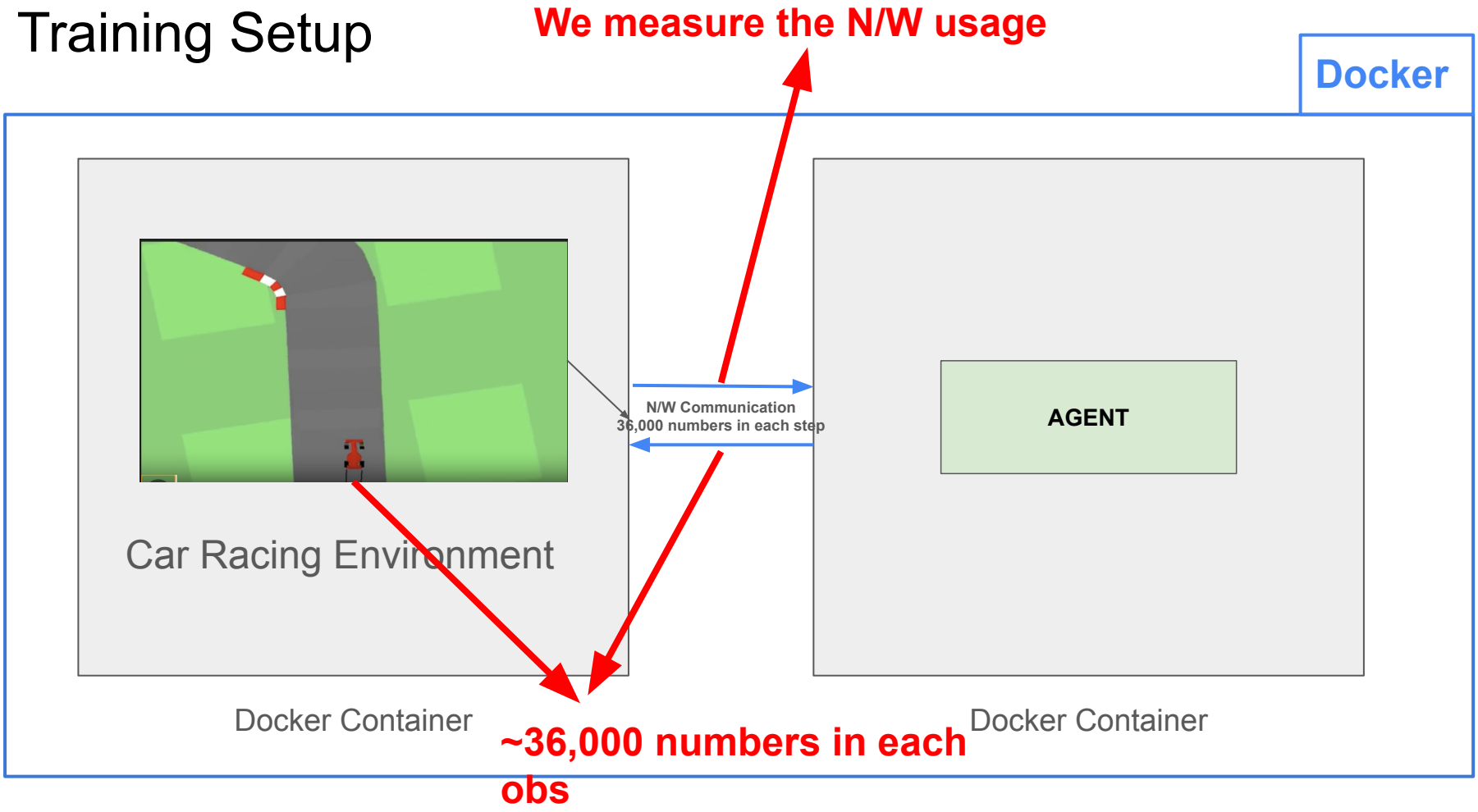
Run the Agent Separately on the Cloud



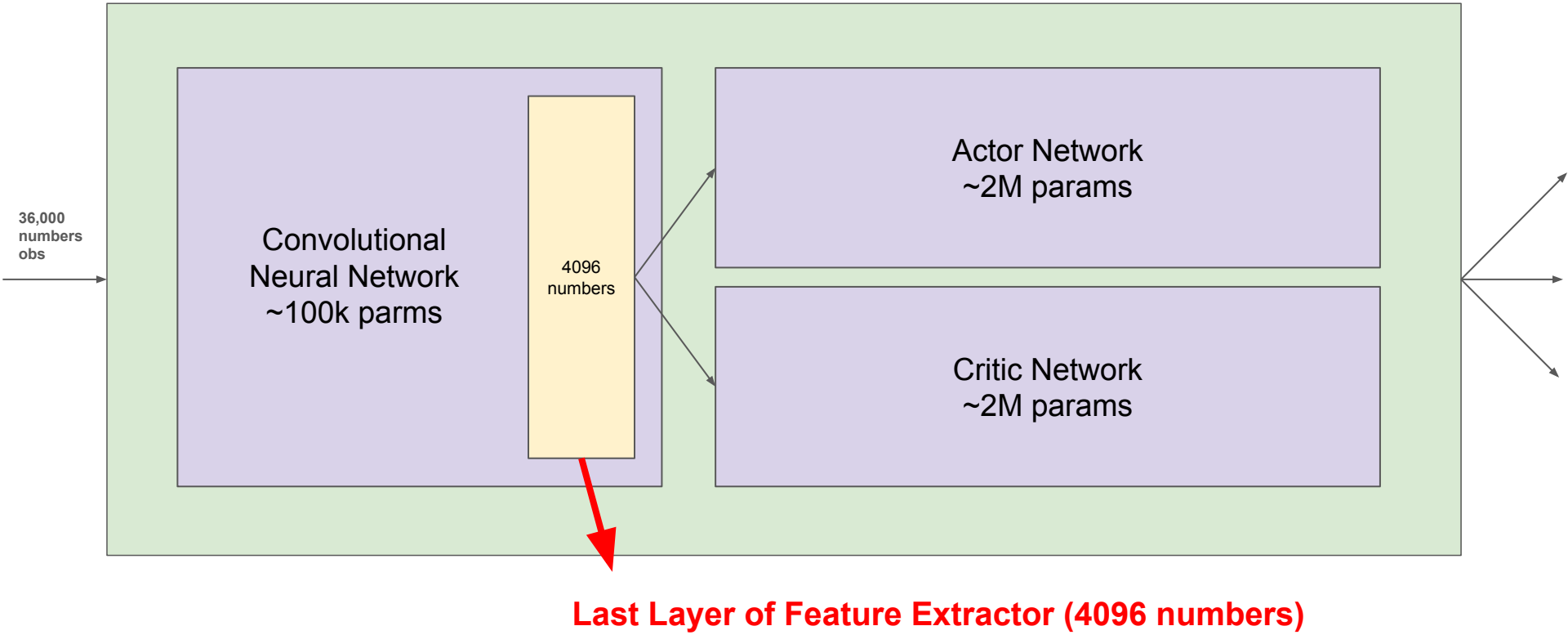
Research Goal

Reduce the amount of data we have to send over the network while training an RL algorithm

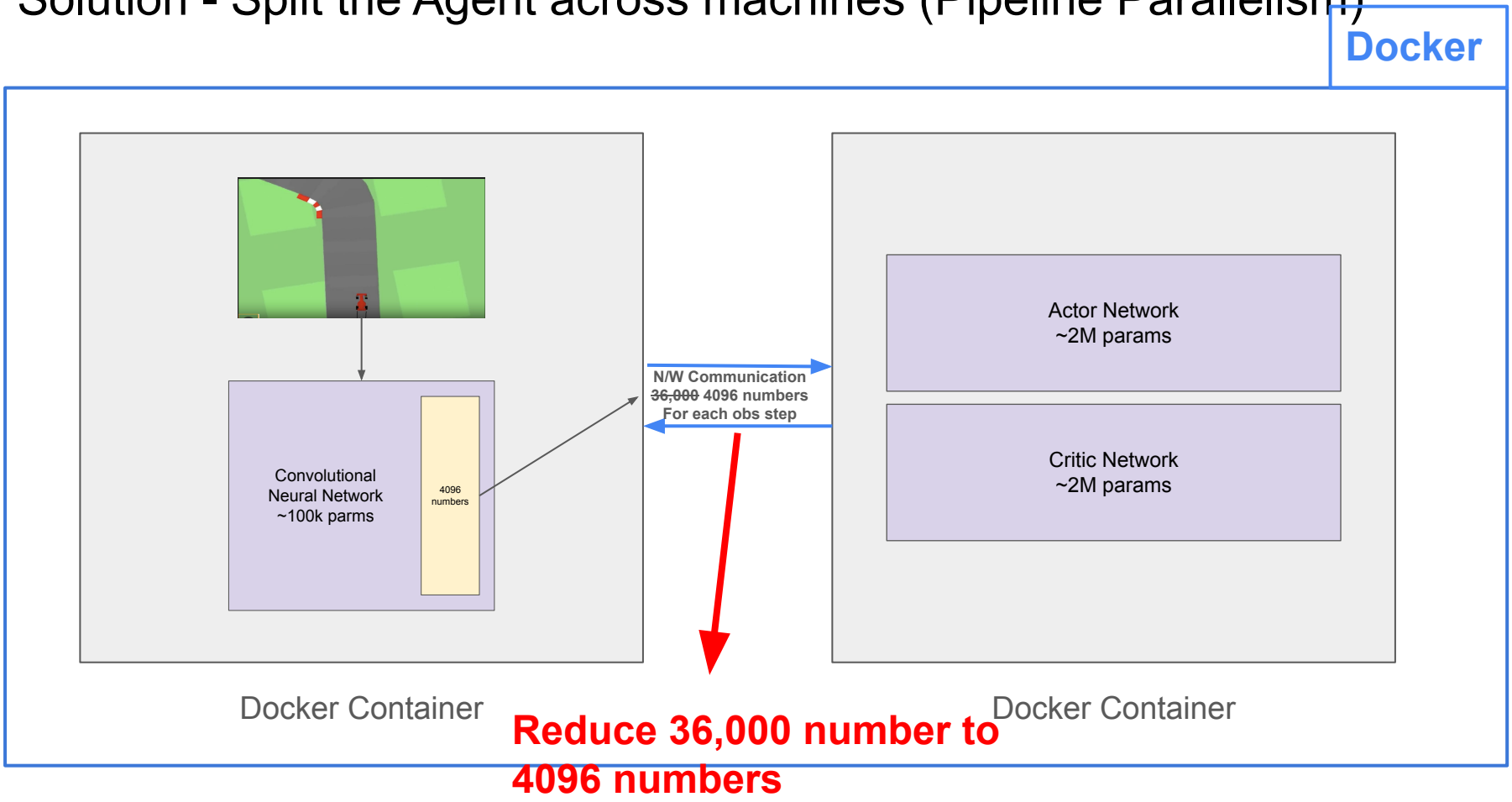
Training Setup



Agent - Is a Neural Network with Multiple Layers



Solution - Split the Agent across machines (Pipeline Parallelism)



But there is a problem!

- We saw that the data transfer **increased by ~2x** after we split the network

Why?

RL has 2 phases:

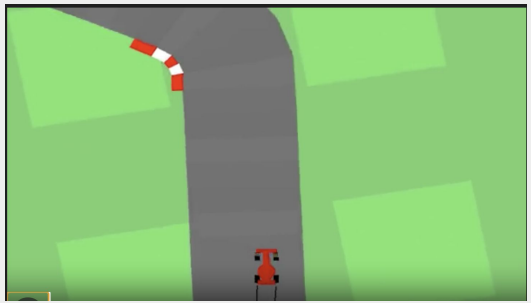
1. Policy Rollout - Interact with env and collect data
2. Training Phase - Used the collected data to update the agent

We do get a **10x reduction** in network usage during the **Policy Rollout Phase**

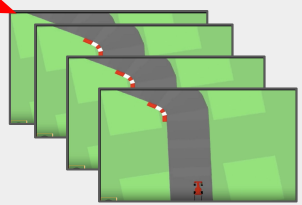
Setup	Tensors Transferred	Data Transferred (bfloat16)
Cloud Setup	14.7B tensors	~28 GB
Naive Split Learning	~1.6B tensors	3.2 GB

But in the training phase…

In the cloud setup, we do not need to use the network during training as the observations are stored on the cloud

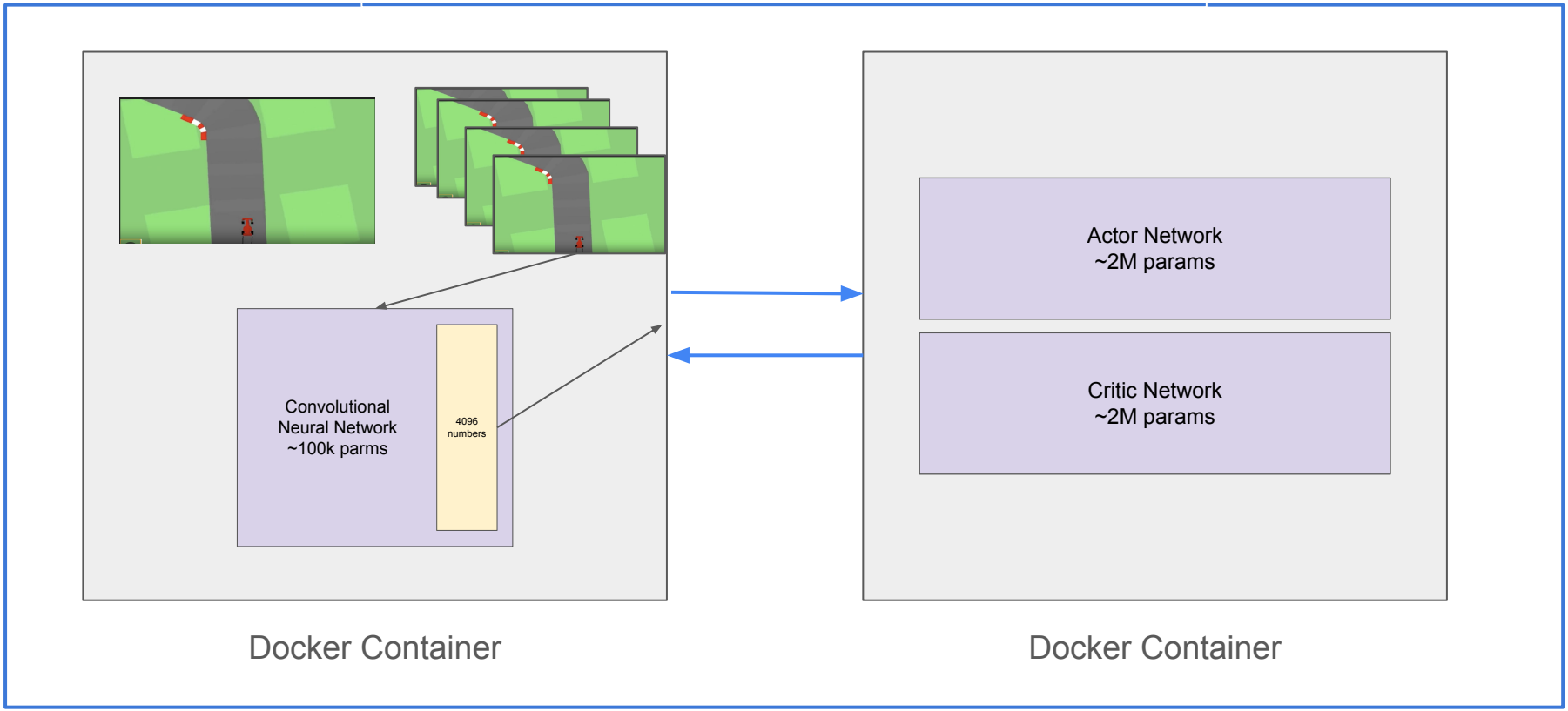


Car Racing Environment



AGENT

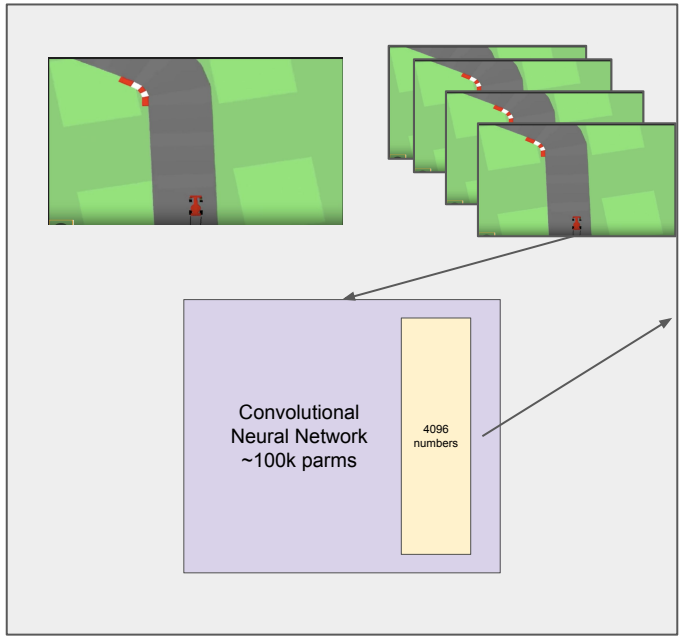
In Split Setup, since the model is split across machines, we need to send activations <u>on every training step!</u>

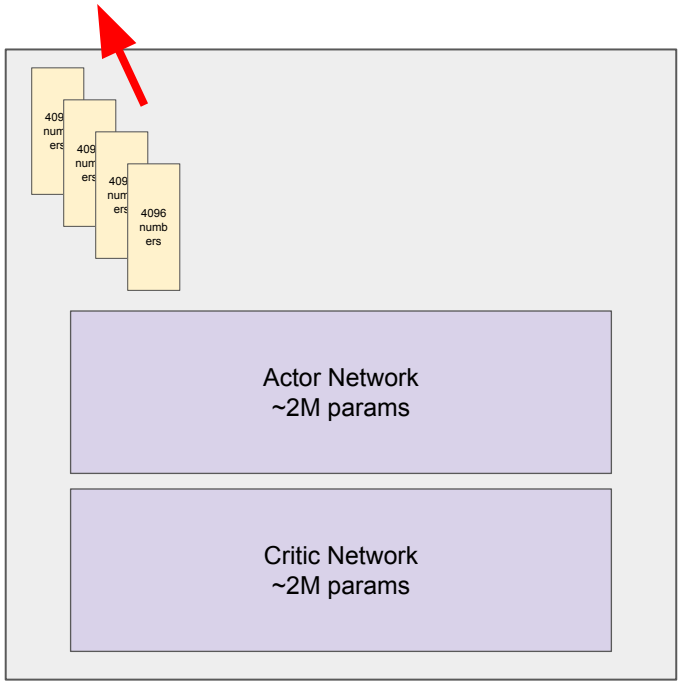


In our split setup, this was an additional 66 GB of data transfer vs 0GB for cloud setup

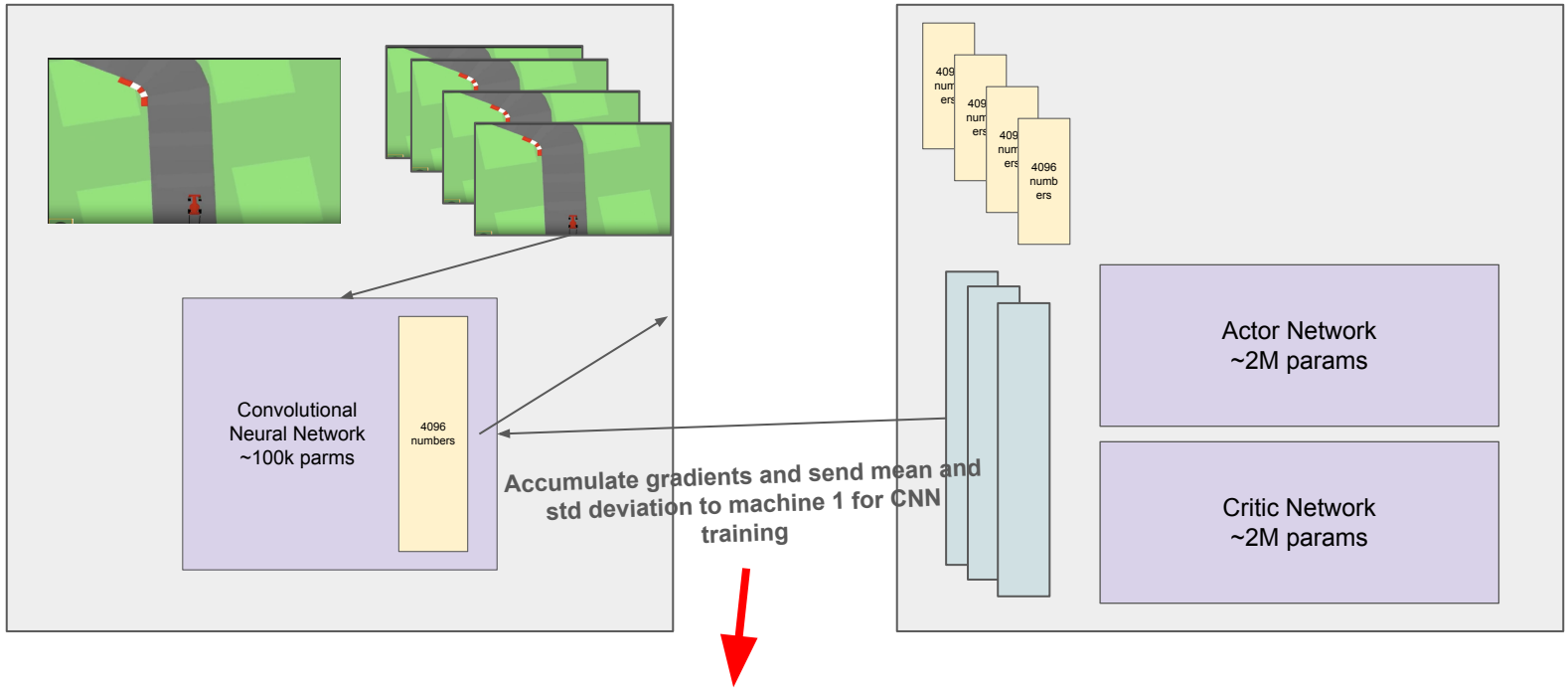
How we plan on solving this (WIP)…
Activation Accumulation and Parallel Training

Accumulate activations and first only train the actor critic network for 10 steps





How we plan on solving this (WIP)…
Activation Accumulation and Parallel Training



The CNN will sample from this to train for n steps