If someone claimed that modern machines were intelligent, how would we disprove them? We could think of tasks that humans can perform that machines cannot, but this is a two-way street: in some tasks, machines vastly outperform humans. Besides, examples alone can never get to the heart of the matter. We could use terms like "generalized" intelligence, but what exactly does that mean?

To be clear, my interest is not merely philosophical. I am interested in *mechanisms* that make complex, unstructured cognition possible. I believe that developments in machine learning bring us closer to answering these questions. Specifically, there is a development common to many recent innovations that I think touches on some of the most essential elements of intelligence: in contrast to traditional machine learning models, which take in data and output predictions, many of the most successful recent models incorporate mechanisms that take in ***state*** and output ***state***. This is actually the defining characteristic of a deep network: inner layers of a network operate on state, not directly on data or predictions. I think that a model's ability to manipulate, mutate, and store state is one of the main determiners of intelligence. Discussion of a specific example will hopefully strengthen this claim:

When I ride a bicycle, my brain seems to upload what we might call a bicycle-riding "program" from memory. This "uploading" mechanism seems to take cues not from external sensory inputs but from some kind of internal signal (maybe something we could call *volition*?). Additionally, my brain draws on complex memories, such as the meaning of traffic signs, that are relevant to the activity. It also attends carefully to certain sensory signals like the movements of cars, while ignoring other signals that might be important in other circumstances, like expressions on faces. Given this complex system of conditioning mechanisms, my behavior on the bicycle from moment to moment requires little attention

or effort.

In general, it seems to be true that in the moment of activity while performing a certain task, behavior is often (always?) automatic. Indeed, the progress of machine learning research suggests that current architectures are capable of mastering many moment-to-moment inference tasks. What distinguishes human intelligence is *not* the specific architecture for performing inference, but the *supporting* architectures that set the stage for inference. In computational terms, the significant element is not the CPU, but the programs that populate the computer memory and establish the state of the system before the computation is even performed.

The power of this meta capability becomes evident in applications that require rapid learning, for example, one-shot learning. In these applications, a model does not have time to slowly learn the inference function using gradient descent. Instead, models have to learn a more abstract mapping: from inputs to hidden states that then facilitate the rapid learning of the actual function. For example, when a person learns a new word, the brain's rapid assimilation of the word into vocabulary clearly suggests the presence of more a powerful *learned* program specifically responsible for vocabulary acquisition.

What happens when this idea is taken to its logical extreme? LSTMs operate on their own parameters, but imagine a model that operates on programs themselves. Humans seem to exhibit this capability: most ordinary adults with a sense of balance and a basic familiarity with wheeled vehicles can almost immediately learn to ride a scooter. Somehow, the mind identifies related, learned programs and combines them to create new ones. Though slow, Hebbian learning might gradually improve the abilities of the scooter rider, the initial encounter with the scooter only benefits from learning related to other tasks. For most tasks,

manipulation and recombination of existing programs seems to account for the bulk of overall learning.

Dr. Tenenbaum's recent paper "blah blah blah" is some of the most interesting work on this topic. The paper describes an algorithm that learns to break a task down into subgoals. One model, the "meta-controller" learns to choose goals based on the current state in order to maximize cumulative extrinsic reward while a second model, the "controller," learns to accomplish the goal specified by the meta-controller using conventional reinforcement learning. The paper demonstrates the performance of the algorithm in two domains: a simple state machine with probabilistic transitions and the classic Atari game, Montezuma's Revenge (in the latter, the algorithm uses convolutional neural networks to approximate the Q-function).

This paper offers one of the most compelling solutions to reinforcement learning problems with sparse feedback. The division of labor between controller and meta-controller simplifies the task for both: the controller doesn't have to learn how to choose goals and the meta-controller doesn't have to learn how to perform them. However, the paper has some limitations that it readily concedes. In particular, in Montezuma's revenge, goals are specified as "objects" (e.g. key, door, etc.) in the environment, picked out by a custom object detector. Of course, this approach would generalize better if methods existed for unsupervised object detection, but one can even imagine situations in which visual objects were not a good specification of goals (for example a natural language task).

One naive solution is for the meta-controller to learn an embedding of goals---instead of outputting a selection among pre-selected goals as in the paper, the meta-controller could output an encoded representation of the goal that the controller would then learn to interpret. In order for

the controller to learn to interpret goals, it would need some kind of feedback indicating when the goal was actually accomplished. Therefore, some network---either the meta-controller or perhaps a third 'critic' network---would be responsible for outputting rewards to the controller for accomplishing goals.

Unfortunately, this simpler formulation is unlikely to work: the critic network will not learn how to reward the actions of the controller until the actual task is accomplished and some feedback is received from the environment. Until then, exploration will be random and the model will suffer from the same difficulties as an ordinary deep Q network---never reaching any objectives, never receiving any feedback, and therefore never learning.

In fact, forcing the model to associate goals with visual objects tells the model a great deal about its environment, things that would otherwise never learn: that rewards are associated with specific locations in the environment that tend to be diffuse.

This difficulty has no easy resolution. In fact, what we have is something of a chicken-and-the-egg dilemma: complex tasks must be broken down into subgoals to be accomplished but it is (seemingly) impossible to learn a dynamic breakdown of a task into subgoals without accomplishing the task at least once and getting some feedback.

This is the problem that I would like to research: how does a model learn to dynamically plan a series of goals without first accomplishing the global task? I believe that this process is recursive and hierarchical: in the base case, the model learns simple tasks through ordinary reinforcement learning; then more complex tasks are mastered by composing previously learned simpler tasks. I would like to explore this idea in using Tenenbaum's controller/meta-controller architecture.