- Looks more like equations from Bayesian inference courses rather than deep learning. Is RL a field where both lines of research work together?
  Doina: RL relies heavily on concepts from probability theory and decision theory, as well as control theory. So, a lot of the math is actually coming from these fields, and the algorithms can be explained completely without talking about deep learning. Deep learning has proven to be a useful tool for the function approximation part of RL, but there is plenty of work in RL that uses other types of approximations (linear, nearest-neighbor, trees,...)

- Hello, Doina. In general, do we prefer an agent that is learning slower than other, but it can outperform after a high number of steps?
  Doina: It really depends on the application. If we work in simulation (eg in games like Go) then we are often interested in the final performance after a lot of data. If we have a real application (eg adaptive treatment design in medicine) then we might only have a small batch of data, which is typically already gathered with some fixed policy, and in this case we want to do as well as possible given this data. And finally if we have an agent that learns continually, we are interested in its performance throughout. You might imagine chatbots as falling in this category, as they would need to adapt to the speaker.

- Is the multiple actor technique feasible in real world problems??
  Doina: It is feasible if either we have simulator, or we have multiple real systems in which we can "clone" a policy (for example, if you have 4 identical robots, you can make each of them an actor). Simulation is quite standard these days for training RL agents, and in this case, multiple actors are fairly easy to instantiate. I still hope we can find better exploration methods which allow us to learn from one stream of data, eliminating the need for multiple actors.

- What happened to previous 56 agents O_o?
  Doina: Agent-57 is named because there are 57-games in the Atari standard suite, and this Agent beats them all :) It's not a version number

- Does PER method have some problem with non stationary environments? What is the reason?
  Doina: All experience replay methods contain data from past policies, which the agent no longer uses. So, the data is "stale". This problem can be worse in non-stationary environments. It affects not just PER but all experience replay algorithms. The "fix" would be to learn online, so data is always fresh, but deep

learning works much better in mini batches and with data that looks more iid. This is why we end up using replay buffers.

- What would be the best way/s to address exploration in continuous action space domains?
  Doina: Policy optimization lets us use stochastic policies that produce continuous actions, so this is one approach. The other one is to use options/temporal abstraction, in which case we go back to discrete choices (but which have been learned from data).

- It seems that many improvements in RL come via different algorithmic procedures. Is there anything to say about the role of ANN architectures in RL?
  Doina: ANN architectures are quite crucial empirically in RL. Eg recurrent nets can make a big difference in performance for agents that need to remember stuff from the past as part of their state.

- Duration time (#iterations) seems to be an important factor, is it always considered set in advance?
  Doina: Not necessarily. It needs to be set if we have compute limitations. Otherwise, we will often do some preliminary experiments to gage how many iterations we need and the pick a number. Or, we can stop an algorithm when it is no longer making process (eg average update magnitude falls below a certain threshold)

- Could you point at the biggest or most important mathematical challenge/ open problem in the Deep RL that is crucial for further advances in the field?
  Doina: Proper generalization. Deep nets hare high capacity approximators so they are prone to overfitting. In supervised learning, we use test sets to prevalent this. In RL, on the other hand, the agent produces its own data. So the algorithms can learn spurious features, leading to good reward but not robust performance. This has been shown before empirically (eg one can change a few pixels on an Atari screen and completely confuse a well trained agent). I think we need to both understand this problem from a mathematical point of view and also fix it, perhaps by doing function approximation differently for RL than for supervised learning.

- Is UCB preferable also with the DNQ algorithm and its variants?
  Doina: Optimism in the face of uncertainty is a general strategy, which can be used with DQN too. UCB as we discussed it requires counting how many times

actions have been chosen in certain states. There is a nice paper by Bellemare et al on count-based exploration, which explains how to approximate counts using deep nets, so they can be used with DQN too.

- To what extent do what is known about RL and the exploration/exploitation dilemma from cognitive science and neuroscience inspire RL strategies?
  Doina: There is quite a bit of inspiration. For example, there is a line of work on intrinsic motivation for exploration, which means agents setting their own reward functions and pursuing these, in order to ensure exploration. This idea comes directly from cognitive science.

- Can we use algorithms of agent 57 in continuous action space?
  Doina: Yes, in principle the ideas in Agent-57 can be used with continuous actions, but some things would need to be adapted. For example, taking a max in discrete action spaces needs to be replaced with an optimization in continuous actions.

- What do you propose to tackle those exploration issues?
  Doina: The agent needs to reason about the value of information that exploratory actions bring. This has been known since the 1990s (eg the work of Mike Duff), but computing this kind of quantity is intractable. We should try to find some good approximations.

- Is there anything like a sawtooth-style move from explore to exploit until some threshold - is this what you meant my episodal?
  Doina: Episodic means the agent takes action until a terminal state is reached (this constitutes an episode) then it is reset to the start. I am not sure about the sawtooth remark (I don't quite understand it).

- Is there a RL framework for modeling the following? Explore, reach some saturation (information level) and then switch to exploitation behavior; repeat the cycle
  Doina: Yes, this is an approach called E^3 (https://www.cis.upenn.edu/~mkearns/papers/KearnsSinghE3.pdf). The idea is to exploit in states where you are certain, then switch to the "unknown" part of the environment and explore there, until states become "known"

- In DDPG, is the critic updated before the actor or vice versa?

Doina: Both are updated at the same time, but they may use different learning rates. Typically in policy gradient we update the critic “faster” ie with higher learning rate than the actor (for theoretical reasons).

- Is DRL learning in real-world a good research topic to work on?
  Doina: Practical applications of DRL are quite important, yes. However, as stated this is a very broad topic so it would have to be made a lot more precise.

- Will on-policy method work in continual learning setting?
  Doina: Yes, in continual learning we expect the policy to change all the time, and on-policy learning should work well.

- Is RL exploration similar to hyperparameter optimization in classical machine learning?
  Doina: This is a confusing question. RL exploration is what produces the data, so it is not like hyperparameter optimization. However, we may need hyperparameters optimization to pick the exploration rate epsilon in epsilon-greedy (or other parameters than control the amount of exploration)

- Is it possible that PER method for DQN requires more times than plain eps greedy to be effective?
  Doina: Yes, it is possible that PER requires more data (there is no proof in general that one algorithm is better than the other). Also, note that PER and eps greedy have different purposes. PER is about sampling data you already have in the replay buffer, whereas epsilon greedy is about producing new data

- Would it be sensible to have replay buffer that prioritizes returning to the states which are estimated to be good? (a la "go-explore" paper)
  Doina: Yes, returning to good states can be quite useful, and in principle could be achieved by the replay buffer priority.

- Can you outline applications and benefits for monitoring dynamic living systems?
  Doina: I am not sure I know of RL applications deployed in this field, but this could be q uite useful in agriculture, I think. For example, imagine you have a farm where there are sensors eg for detecting soil humidity, illumination, nutrient levels etc. You could imagine an RL agent which learns a policy for watering plants or for adding fertilizer depending on these sensor readings. I am not sure if anyone is pursuing this kind of research though.

- Could you please elaborate again why the value-based methods may lead to drastic changes of policy while policy gradients are more smooth?
  Doina: Imagine that we have two actions, whose values are Q(a1)=0.2 and Q(a2)=0.205, and we are doing epsilon-greedy, with eps=0.1. So, action a2 currently has probability 0.9, since it's the greedy action, and a1 has probability 0.1. Imagine now that we try a2 and it turns out slightly worse, because reward are stochastic. So we update its value to Q(a2)=0.199. Now all of a sudden a1 is the greedy action, and it's probability flips to 0.9, and probability of a2 is 0.1. This is a problem with epsilon-greedy specifically, and can be mitigated somewhat by other methods. For example, sotmax/Boltzmann exploration makes the probabilities a function of the action values, and there is a temperature parameter that allows the change to be more smooth. But then a very high temperature leads to a lot of randomness which slows down learning. In policy-based methods, this issue does not arise.

- Is it a baseline always recommended also for DQN methods?
  Doina: DQN learns action-value estimates, so it has no use for a baseline. However, reducing the variance in the updates can still be useful (this is what the baseline is for). We can do this eg by reward clipping or normalization.

- In the PO, are we expecting the network to learn a trade-off between exploitation and exploration as opposed to the value-based models?
  Doina: The policy that we learn will end up being stochastic for many methods, so it will continue to explore. So in that sense, it does a trade-off already. However, DDPG for example can learn a deterministic policy, in which case we still need to inject randomness to ensure exploration.

- Would it make sense to use epsilon-greedy with the PO? If not, why not?
  Doina: policy optimization parameterizes the policy, whereas epsilon-greedy maps the value function into a policy, without parameterizing the latter directly. So they are like apples and oranges, and we can't put them together. But we can add randomness into the policy learned by PO, eg by regularizing towards high entropy.

- why does value function have less variance than policy function? (and hence used as baseline)
  Doina: The policy is a mapping from states to actions, which the value function which we use as baseline is a function of states. The baseline has to be a

function only of state, so that when we take its gradient wrt the policy parameters, it is 0.

- How stable/reproducible are the results presented in the Agent57 paper? It use great amount of computational resources (X-axis in the figure is 1e11 scale).
  Doina: Yes, this is an agent that requires a lot of computation, but the algorithm is straightforward. Previous research of this type from DeepMind has been reproduced by other labs (eg AlphaGo), so I see no reason why Agent-57 would be different.

- What are best practices to improve the stability of RL algorithms (random seeds, choice of discount rate, etc)?
  Doina: One needs to optimize hyper-parameters (exploration rate, learning rate, eligibility trace parameter). I do not consider the discount in this category, as it defines the problem (it's like the loss function in supervised learning). Random seeds are useful in order to figure out how much variability there is among runs.