Answered questions (offline)

- What do you think about model-based vs model-free RL?
  Diana: Personally, a big fan of model-based RL and still holding hope that we will find the right usage for them. Currently, in most SoTA algorithms we use model-free RL, in part because we are relying on a simulator and able to generate vast amounts of data. I would expect applications and scenarios that have stricter restrictions on the sample complexity, to be the ones that could benefit from a more model-based approach. If you do have access to the simulator, it's hard to argue against more (real) data -- in a sense that the best non-parameteric model you can get.

  There's a nice study on where a model might help and careful study: https://arxiv.org/pdf/1906.05243.pdf (Neurips 2019)
  The paper was a response to: https://arxiv.org/abs/1903.00374
  (I would recommend reading both as a more balanced argument)

- Can we use RL to model/optimize limited-resources distribution problems? for example electricity distribution for houses? and if yes how can I formulate it?
  Diana: This is a partial answer as I don't know enough about the problem (the limited-resources distribution problem). But, as a general principle:
  - Can it be casted as a cumulative reward maximisation problem? (it might not be trivial to come up with a reward fct here, but you already have an optimisation criteria you can shape the reward based on that).
  - What is the action space? What are the controls and/or intervention available to you (the 'things' that you can do and change to optimise the objective)
  - What is the state space? Can I make this Markov? (This is might be different from our pure observation state)

    Maybe an interesting case study that might be similar (again limited understanding of the problem), might be the work we've done in the cooling data centers: https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40

- Are there situations in which a combination of RL and evolutionary computing may be beneficial?
  Diana: There has been some work along those lines. In particular, cross-entropy optimisation has been shown to be quite effective (some say embarrassingly effective for such a simple method) for direct policy optimisation in continuous control problems.

  See, for instance:
  https://www.aaai.org/Papers/ICML/2003/ICML03-068.pdf
  https://ieeexplore.ieee.org/document/6961536

  More recent work:
  https://arxiv.org/abs/1805.07917
  https://openreview.net/forum?id=BkeU5j0ctQ

- May you please suggest few papers where RL is used for image classification tasks!
  Diana: The example was trying to make a point about the generality of the framework. I don't think there are advantages to treating as such: actually with the right reward function I think training a classification loss would be what an RL (Monte Carlo based, for instance) would do as well -- aka the RL problem would recover the supervised learning problem, up to exploration. Again, it speaks to the generality of the framework and maybe points to the fact that, if we were to desire to train complex agents that have to do many (types of) tasks during its lifetime or as subroutine, we can rely on the same formalism to capture all of these.

- When you say that MDP assumptions is often too rigid, this explains the use of recurrent networks for parametrizing policies in DRL algoritms?
  Diana: Indeed - in a sense we are hoping that the recurrent model would learn to reconstruct something close to the Markov state. That it learns to include and 'remember' all the necessary information and history to make the current prediction (usually this is a value function).

- In the discounted infinte horizon return function, G(t) represents the value function at time t based on future rewards?
  Diana: In all formulations, G_t is the (potentially discounted, depending on formulation) cumulative reward obtained by the agent after time t.

- Is it correct that Bellman equation is simply dynamic programming for the expected reward calculation?
  Diana: The Bellman eq. indeed enables us to do (efficient) dynamic programming, but moreover enables us to look at sample-based algorithms (like TD) when dynamic programming is not an option (maybe the model unknown and/or really big state-action space).

- what if we have MDP, but the state space is intractable. What algorithm would allow us to benefit from the knowledge of the environment model but computable?
  Diana: Not sure what 'intractable' here means. But I would remind you of the AlphaGo example, where the number of states are more than there are atoms in the universe and we are, to some extent, able to deal with these kinds of situations, via sampling and generalisation (like most ML) - we never need to see everything.

- For TD learning, can it help to start with higher value of n and decrease it over iterations?
  Diana: You could consider that. I would maybe go the opposite direction - credit assignment wise. Starting with a small n would mean the first iteration of policy evaluation might finish sooner and with lower variance and we get to improve on this. On the other hand a large n might introduce a lot of variance that might be quite damaging at the beginning of learning where policies are not very competent and would take most random actions.

- Why usually SOTA DRL algorithms are applied by using TD one-step methods instead of n-steps? Wouldn't trade off bias for variance be good for such methods?
  Diana: I think this depends largely on the environment. I would actually say that n-step methods are more prevalent in a lot of DRL algorithms.
  For instance: Atari struggles with 1-step (can make it work), but n=3 or 4 speeds up learning and performance considerably. More complex environments, like DM Lab(https://deepmind.com/blog/article/open-sourcing-deepmind-lab) typically require n=4 or n=8 to just get off the ground.
  In general, this is a tunable parameter, but we rarely see n=1 to be the best one.

- Can RL be seen/formulated as a unifying approach to other machine learning problems (classification/regression/so-called unsupervised learning/etc)?
  Diana: We can definitely cast supervised learning problems as an RL problems + one that actively cares and optimises for the data collection process (called exploration in RL terms). Whether this insight is useful or actionable remains to be seen. In my personal perspective, it's a more of sanity check: when I'm thinking a design an algorithm that needs to perform a fairly complex task that might requires as subtasks or subroutines, pure prediction (mini)problems, classification (mini)problems and control (mini)problems, I have framework and learning rule that can capture all of the above + the data collection that would cater to all of these subproblems.

- Offpolicy vs Onpolicy learning which to use and why?
  Diana: This will depend on the problem - which of them are better. In general, on-policy learning and on-policy data is better for learning, because we are sampling exactly from the trajorities and states that we are trying to evaluate. At the same time, off-policy learning enables us to use multiple sources of data and not need to re-generate data for every policy evaluation problem we might encounter in a control optimisation pb. Although this is a **big** advantage in terms of reusability of data, coverage of this data might still be insufficient to learn about the value function we are trying to fit right now. If the data that we are using for learning, does not cover sufficiently the stationary state-action distribution for the policy that we are trying to evaluate, the learning problem is bound to suffer and its generalisation across this target stationary state distribution might be very poor. And as we might rely on this estimate to improve our policy, poor generalisation might lead to severe delusions of what the payoff in this part of the space might be. The only way to correct this delusion would be to go back to the environment, **on-policy,** to collect more data (on-policy) to contradict this poor generalisation.

- Temporal difference learning: it is not clear for me, how can we look only to one state. Don't we need to unfold the full trajectory?
  Diana: This relies on the Markov properties and you can 'read it off' the Bellman eq. What is going to happen in the future is captured entirely, in terms of expected value, by the value function at the next state and action Q(S_{t+1}. A_{t+1}). Thus from my current state S_t and having taken action A_t all of the information about the expect value of my return in this state

is captured by the expectation over the next state of the world/agent (S_{t+1}, A_{t+1}) and the reward I received as part of this transition.

- Why we need (1-lambda) in lambda-Return?
  Diana: This is done to make the sum be a convex combination of the n-step returns. The individual weights for each n-step return would be w_n = (1-\lamba)\lambda^{n-1}. Thus the sum over all weights w_n =1 (as n tends to \infty).

- Hello, Diana. In lambda return - the parameter lambda is a hyperparameter and it can be tuned?
  Diana: Indeed as with n (in the nstep) this becomes a tunable parameter, servicing the whole spectrum from 1-step/few steps to close to Monte Carlo. I don't think I mentioned this in the lecture, but this for any lambda in (0,1). The end points in this spectrum are defined separately as: TD(0) = 1-step TD and TD(1) = Monte Carlo return.

- How to overcome the problem of non iid data in Deep Q learning? Is the Replay Buffer the only method to overcome the same?
  Diana: Indeed the replay helps with the regime in which deep nets have been shown to be effective. Note that TD doesn't need this per se and with other FAs (like linear approximations on a given basis) we don't typically need to do this.

- Can off-policy methods be interpreted as transfer learning so that I am transferring the knowledge of the behavioral policy to the policy I am searching?
  Diana: Interesting view. I guess the answer is yes, where what we are transferring is data. Usually, in most transfer scenarios, we have other elements, maybe (part of the) parameters, maybe (sufficient) statistics, maybe part of the models that are being 'transferred'. In the same sense, as we view replay as a non-parametric model, one can argue that we are doing a transfer or distill of knowledge from a behavior policy to the target one that we are optimising.

- Does the Bellman equation still holds when we use an n-step update?
  Diana: It does indeed. The Bellman eq. relies on the Markov assumptions and if a process is 1-step Markov (as per the MDP assumptions), it is n-step Markov as well.

- What is your take on probabilistic/Bayesian views in RL? Some generic thoughts and pointers on this would be great!
  Diana: As Bayesian, I'm a fan and I think it's a beautiful formulation and way to think about the problem, especially when thinking about information gain about the environment, reasoning about uncertainty, exploration and even more guided planning under model uncertainty. I would really want to see more work in this subfield and progress in making this formulation tractable. It has been shown to be very efficient in small domains, (maybe) at the expense of (a lot more) computation. In most environments we looked at, most methods that rely on something like history in the observations, become intractable.

  Some pointers:
  Here is a nice survey: https://arxiv.org/abs/1609.04436 from a former colleague.

  Another nice work in this space: https://arxiv.org/abs/1205.3109 (Arthur has a couple of papers on this and I'm linking also his PhD thesis that I think does a great job at explaining the paradigm and its potential benefits and limitations for efficient planning