We thank the referees for their comments.

Following the suggestions of R1 and R2, we will add a conclusion to the paper and modify the broader impact statement. This will clarify our contribution, its practical impact, but also the results described in Section 5. We will, of course, implement the other suggestions made by the reviewers, such as adding mathematical definitions, correcting typos, and proofreading.

Response to R1: We will indeed add a more detailed discussion on existing work in the supplementary material, this is a good idea. We will also provide an alternative graphical representation of the relu function using a different selection.

Response to R2: Our present contribution is to provide a simple theoretical framework, which allows us to describe current implementations of AD. We also explain why, *in theory*, artificial critical points are not an issue (item (2) suggested by R2). Claiming that spurious critical points are met in the practice of deep learning indeed needs developments that we have not provided (and that we cannot present in this paper). Instead, we shall illustrate their existence through toy models; we will avoid saying that the problem can be met in practice. Now let us discuss the main issue raised by R2: does the spurious behavior impact *practice*?

- Our “trap-avoidance" result suggests that, at high precision, artificial critical points are not met in practice.
- On the other hand, at low precision, “genericity results" may partly collapse. For instance, very large scale linear programs have several solutions, even in 32 bits. Yet the theory says uniqueness is generic. It raises the question: what is the impact of the spurious behavior at low precision?

These aspects will be briefly evoked, but they will be treated in a seperate work.

Line 29-30 is not a claim about the practical impact of artificial critical points. It is rather a statement about how difficult the study of the *“spurious set"* can be when dimension grows. Indeed, in larger dimensions, the composition of intermediate functions may be extremely complex, making the geometry of artificial point difficult to grasp. We will rephrase so that this appears more clearly.

Regarding the combinatorial nature of selection derivatives, the reviewer is right, combining different selection functions within a neural architecture leads to a combinatorial explosion of the number of possible selections, depending on the choice for each unit. But the user does not meet this issue because

- The selection process is implicitly described by the numerical code describing the network (Proposition 5)! In other words, practitioners already implement selection functions and selection derivatives by writing programs. Selections are mathematical objects representing programs. They are only used for proofs and theory; for implementation, one uses the corresponding programs!
- A given function has an infinity of selection representations, each inducing a potentially different selection derivative. All our results work for *any such selection derivative*. So one does not need to consider all possible selections, being able to compute a single of them is sufficient.

In other words, the scaling issue *has no impact whatsoever* and AD can be used, as is, to compute a selection derivative, which is sufficient for optimization purposes.

On the other hand, the reviewer raises a very interesting question regarding the numerical behavior of AD related to combinatorial explosion. A detailed presentation would require another complete paper, and we decided to postpone this question for future research.

Response to R3: The reviewer is absolutely right to point out rates and complexity. Contrary to the smooth setting, we are not even sure what is the correct notion of approximate stationarity that should be used. This is a relevant topic of future research, and we will add it to the (new) conclusion.

Regarding selection derivatives for discontinuous functions, the proposed framework can indeed be used. Actually, Tensorflow provides a value for the derivative of the step function (zero everywhere) so selection derivatives for discontinuous functions are already available in a sense. We believe that dropping the continuity constraint, one could prove similar results as propositions 1,3, and 4. Hence this seems innocuous from a computational perspective.

The issue with discontinuous functions is that the current “theory" does not apply directly. It is much more difficult to make a theoretically consistent interpretation of selection derivatives. In particular, Proposition 2 does not hold anymore; it is not possible to integrate along segments. This property is actually crucial to prove convergence to critical points. We plan to investigate these questions in future work.