We thank the reviewers for their feedback. We reply to the main points below:

Usefulness of the method (R1, R3) and applicability of assumptions (R4): Actually, this work started precisely to unblock the adoption of a real-world Computer Vision AutoML system: Users fine-tune models selected from a large model zoo testing hundreds of combinations of different architectures, pre-training sets and hyper-parameters, but are reluctant to do so without visibility of the expected ROM cost of the training. We focus on fine-tuning (R3) since it is faster and typically performs better than training from scratch in vision problems. For these reasons, it is generally the choice in AutoML systems, where users pay by the hour. While our work may not directly impact academic researchers, it is an enabler of large-scale AutoML, which we expect will further foster academic research in the years ahead. In this sense, we would say our work is useful: it enables a cost estimate that allows reducing a large search space to fit a user's budget, a small step toward better accessibility, and democratization of ML. We are happy to expand the discussion in this direction if it is of interest to the readers.

Note the technical hypotheses that the weights remain close to initialization (R4) is only used to derive the approximation, which is then verified empirically to hold (albeit with a larger error margin) even when the task is quite different from pre-training (e.g., fine-grained airplane classification using ImageNet pre-training, see also Fig. 10 and Section B). In real-world AutoML this concern is largely moot since model selection techniques selects for further fine-tuning only models pre-trained on close data.

Make contributions over related literature more clear (R2): Our main contribution is to introduce the problem of predicting training time in realistic use cases (see previous point), in particular how this depends on the hyper-parameters (for which some previous literature exists) and on the interaction between target task and pre-training (which, to the best of our knowledge, is new). *NTK theory* [15] studies randomly initialized DNNs in the limit of infinite width and batch size. We replace it with an off-the-shelf pre-trained network fine-tuned with SGD and show that its predictions can be effective on practical networks. We also show how to approximate the NTK matrix efficiently while maintaining a good accuracy for our end-tasks. *Gradients as feature* [25] uses a linearization-based analysis of fine-tuning which is similar to ours, but it focuses on efficiently using the gradients of the network as features for a linear classifier, and is unrelated to our work in both scope and methods. *Mean field approximation* [13] focuses on describing a special initialization technique. However, we build on their SDE approximation in weight space and show that it can be translated to function space, making a numerical solution possible for real sized DNNs.

Connection to learning curve prediction methods (R3): We thank the reviewer for pointing out this relevant area of research, which we will discuss in the paper. We should note that these papers focus on predicting the effect of different hyper-parameters for fixed task and architecture. However, we found the relation between target task and pre-training is more difficult to model. We initially employed a similar black-box regressions techniques to predict training time, but they turned out to be data-hungry and less likely to generalize on different models. This prompted us to develop a more interpretable and less data expensive approach presented here.

Experiment with different definition of training time (R3), different architectures (R2): We further tested our method with the suggested definition (first time training error decreases by less than ϵ in one epoch), and also increased the number of iterations to 600 to ensure convergence and top accuracy. In this (more challenging) case, we achieve 13% avg. relative prediction error and 0.94 Pearson's correlation between predicted time and ground-truth. The method also generalizes to several deeper architectures (ResNet-50, Densenet-121). However, we note that predictions for older architectures (AlexNet) is less accurate (40% avg. error, 0.46 Pearson's correlation). This is expected as their loss landscape is known to be very rough, so gradients at initialization are less informative.

Clarifications: We have amended the text to make all requested clarifications. More in detail:

- **Table 1 (R2, R4):** We report the absolute error on training-time prediction as a function of the selected threshold. We show the error both using a small learning rate (our approximation holds better) and high learning rate. Different thresholds reflect training time predictions at different regimes: initial (high ϵ) and final (low ϵ) convergence.
- **Random projections (R2):** We do not need the gradients to be sparse. We use the property of random projections to preserve the expected L2 distances (and hence inner product) when applied to high dimensional vectors (see [1,5]).
- **Figure 1 (R4):** For each point we sample a different batch size, learning rate and a different dataset size.
- **Dataset details (R4):** We added a table describing for each dataset the number of images and classes.
- **Overestimation or underestimation? (R3):** We slightly overestimate the training time. This is due to the non-linearity and high capacity of the network which is not entirely captured by the linearization approximation (see Section B).
- **Infinite batch size assumption (R2):** Please note that this is used only in eq. (4) to give an analytical interpretation. We make no such assumption in eq. (1), which is what we use for the actual training time prediction.