Economically-Optimal Compute Allocation

Anton Troynikov

April 2025

https://github.com/atroyn/ComputeAllocation

1 Introduction

Frontier AI labs must determine how to allocate their available compute between training new models and serving existing models to paying customers.

Previous work [1] has sought to determine how to optimally allocate between training and inference for model performance, but how should frontier labs allocate compute if they wish to maximize the net present value (NPV) of all revenue (possibly the light cone of all future value in the universe [2])?

2 Optimization

We define the following variables:

- $C(t)$: Total available compute at time t
- $\alpha(t) \in [0, 1]$: Fraction of compute allocated to training at time t
- $M(t)$: Model capability at time t, which increases with training
- $R(M)$: Revenue as a function of model capability

The model trains as a function of total compute allocated to training so far, with diminishing returns:

$$\frac{dM}{dt} = \beta_T \cdot (\alpha(t) \cdot C(t))^{\epsilon}, \quad \epsilon \in (0, 1) \tag{1}$$

where β_T is a scaling constant.

Revenue is a function of model capability. Let γ be the return to increasing model capability:

$$R(t) = \beta_R M(t)^{\gamma} \cdot (1 - \alpha(t)) \cdot C(t), \quad \gamma > 0 \tag{2}$$

where β_R is another, different scaling constant.

Assuming a discount rate of ρ, over a finite time horizon T, we want to optimize:

$$\max_{\alpha(t) \in [0,1]} \int_0^T e^{-\rho t} \cdot R(t)\, dt \tag{3}$$

This can be cast as an optimal control problem, with α as our control and M our state.

Following the Pontryagin maximum principle [3, pp. 216], we construct the Hamiltonian:

$$\mathcal{H}(M, \alpha, \lambda, t) = e^{-\rho t} \cdot \beta_R M^{\gamma} \cdot (1 - \alpha) \cdot C(t) + \lambda \cdot \beta_T \cdot (\alpha \cdot C(t))^{\epsilon} \tag{4}$$

with λ subject to (co-state):

$$\frac{d\lambda}{dt} = -\frac{\partial \mathcal{H}}{\partial M} = -e^{-\rho t} \cdot \gamma \beta_R M^{\gamma - 1} \cdot (1 - \alpha) \cdot C(t) \tag{5}$$

Differentiating (4) with respect to α:

$$\frac{\partial \mathcal{H}}{\partial \alpha} = -e^{-\rho t} \cdot \beta_R M^{\gamma} \cdot C(t) + \lambda \cdot \beta_T \cdot \epsilon (\alpha \cdot C(t))^{\epsilon - 1} \cdot C(t) \tag{6}$$

Setting (6) to zero and solving for α gives the optimality condition:

$$\alpha^* = \left(\frac{e^{-\rho t} \cdot \beta_R M^{\gamma}}{\lambda \cdot \beta_T \cdot \epsilon} \right)^{\frac{1}{\epsilon - 1}} \cdot C(t)^{-1} \tag{7}$$

We'll have to clip (project) α^* to be in the range $[0, 1]$ when this isn't the case.

3 Dynamics simulation

Unfortunately, as is so often the case in this cursed, fallen universe, we do not obtain an analytical form for the resulting trajectory and so must simulate numerically.

We choose relatively conservative parameters:

- Revenue and performance scaling factors are normalized ($\beta_T = 1.0$, $\beta_R = 1.0$)
- Revenue is convex in model capability, assuming tokens from smarter models are worth more, and smarter models are easier to integrate ($\gamma = 2.0$)
- Returns to performance from additional compute following empirical results from scaling laws ($\epsilon = 0.5$)
- Exponential growth in compute, with $C(t) = e^{0.1 \cdot t}$, initially available compute set to 100 'units' ($C(0) = 100$).
- We use a standard discount rate ($\rho = 0.05$).

We simulate over a finite time horizon with $T = 10$, and transversality condition $\lambda(T) = 0$, i.e. no more returns from model capability at the end of the time horizon. Note that in practice, this does not limit the generality of our result. The result is shown in Fig. 1.

We observe that the initial 'training ramp-up' where a frontier lab spends all of its compute on training until a model becomes useful enough to deploy at all is captured, followed by all resources eventually being allocated to inference.

Conservatively, we may regard these results as relevant to the training and deployment of a given 'generation' of model, rather than of all training conducted over the lifetime of the lab.

3.1 Sensitivity analysis

We perform a simple parameter sweep to conduct sensitivity analysis for each of ϵ, γ, ρ, and the compute growth rate. We evaluate the variation in final NPV, model capability, and

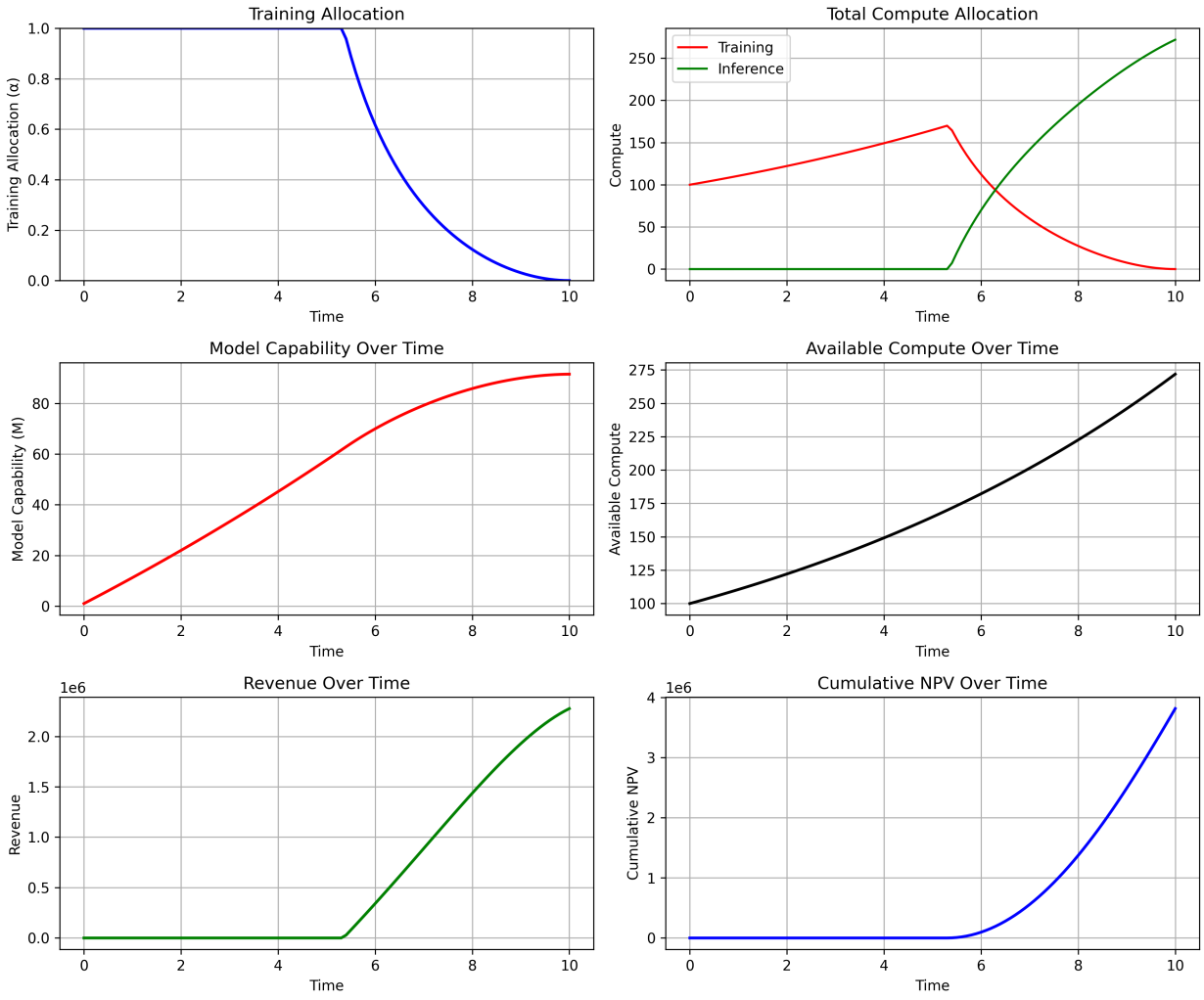

Figure 1: Simulated optimal compute allocation over time. $\beta_T = 1.0$, $\beta_R = 1.0$, $\epsilon = 0.5$, $\gamma = 2.0$, $\rho = 0.05$, $C(t) = e^{0.1 \cdot t}$, $C(0) = 100$, finite time horizon with $T = 10$.

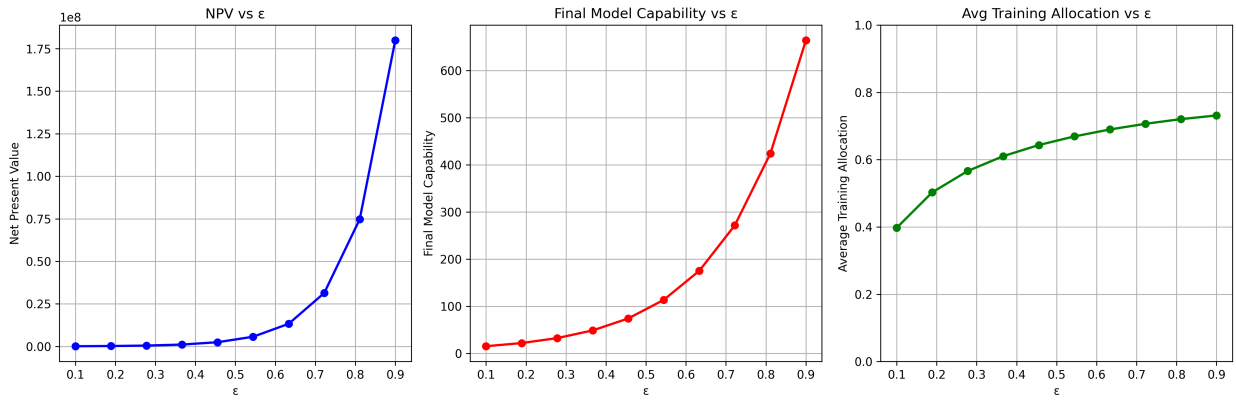

Figure 2: Sensitivity to ϵ

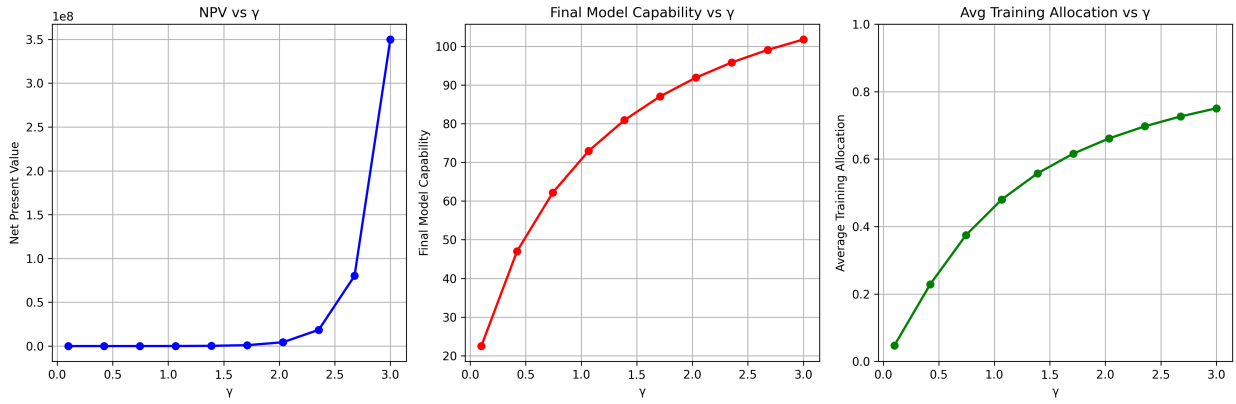

Figure 3: Sensitivity to γ

average training allocation over the time horizon. The results are given in Figs. 2, 3, 4, 5.

We note that the most important determinant of the proportion of compute optimally allocated to training is not the rate of growth in compute, but the return on model capability γ. This suggests that not only model capability, but market demand and penetration should strongly influence how frontier labs allocate compute.

4 Steady state analysis

We are also interested in the optimal allocation in steady state, i.e. with infinite time horizon, and a constant control signal α^*. This is akin to 'smoothing out' the S-curves of multiple sequential model generations, over the (infinite) lifetime of the lab.

We assume that in steady state, since the allocated training fraction α is constant, model

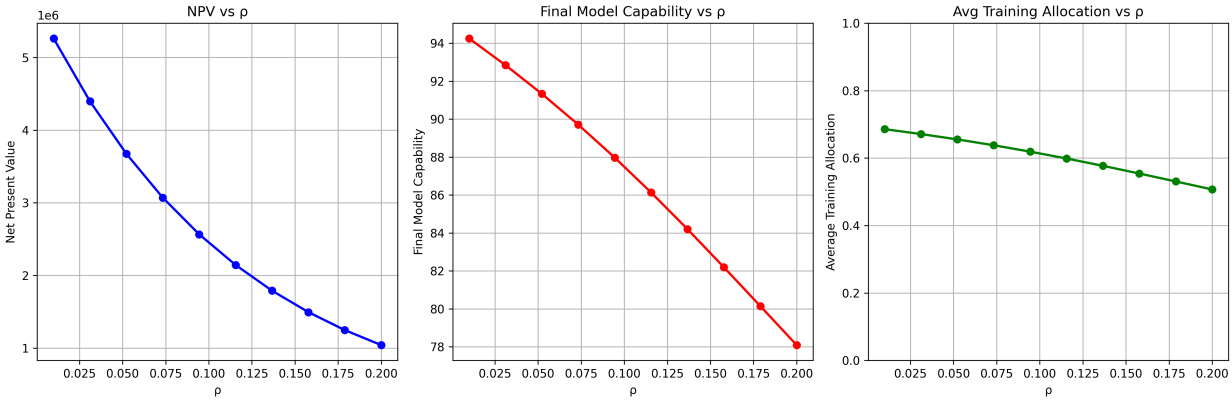

Figure 4: Sensitivity to ρ

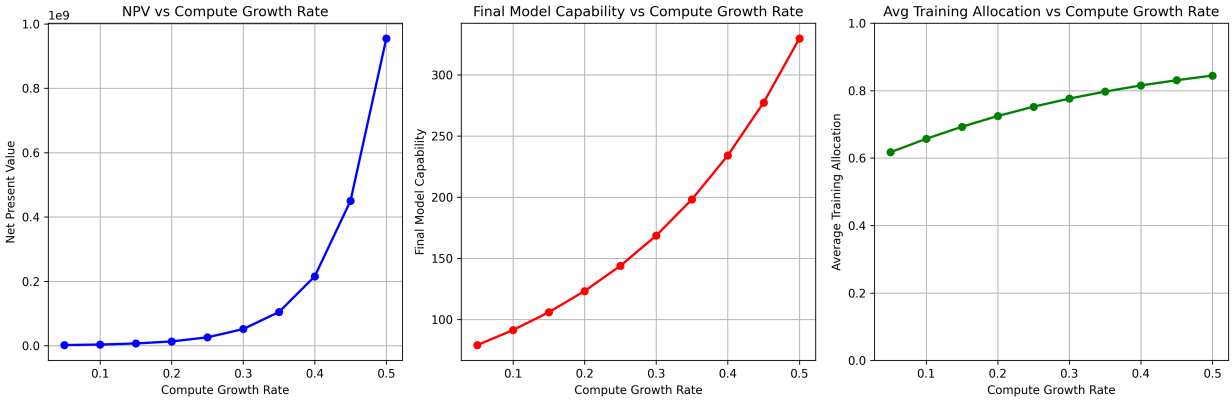

Figure 5: Sensitivity to compute growth rate

capability M grows exponentially at a constant rate k, $M(t) = M_0 e^{kt}$, and the proportional growth rate $\frac{dM}{dt} \cdot \frac{1}{M} = k$.

Assuming total compute also grows exponentially with rate g as $C(t) = C_0 e^{gt}$, and with α constant, by (1) we have:

$$k = \epsilon \cdot g \tag{8}$$

Which gives $M = k \cdot C^{\epsilon}$. We note that for convergence, we must assume discounting is greater than the rate of compute growth, i.e. $\rho > g$. If compute growth outpaces discounting forever, then almost all value is in the future and all allocation must be to training forever.

The steady state condition for optimal control on an infinite time horizon [3] is given by:

$$\frac{d\lambda}{dt} = \rho\lambda - \frac{\partial H}{\partial M} = 0 \tag{9}$$

By substituting the optimality condition (Eq. 6) we have:

$$\lambda = \frac{\beta_R M^{\gamma}}{\beta_T \cdot \epsilon(\alpha \cdot C(t))^{\epsilon-1}} \tag{10}$$

Combining (5), (8), (9) and (10), and solving for α gives:

$$\alpha^* = \frac{\epsilon}{(\epsilon + (\gamma - 1) \cdot (1 - g/\rho))} \tag{11}$$

4.1 Simulation

To validate the assumptions of the analytical result, we simulate a variety of steady state allocations numerically. We adopt the same parameter values as in the dynamics simulation. The results are shown in Fig. 6.

We observe that the numerically optimal allocation (0.5) is close to that derived by the analytical solution (0.6). Nice.

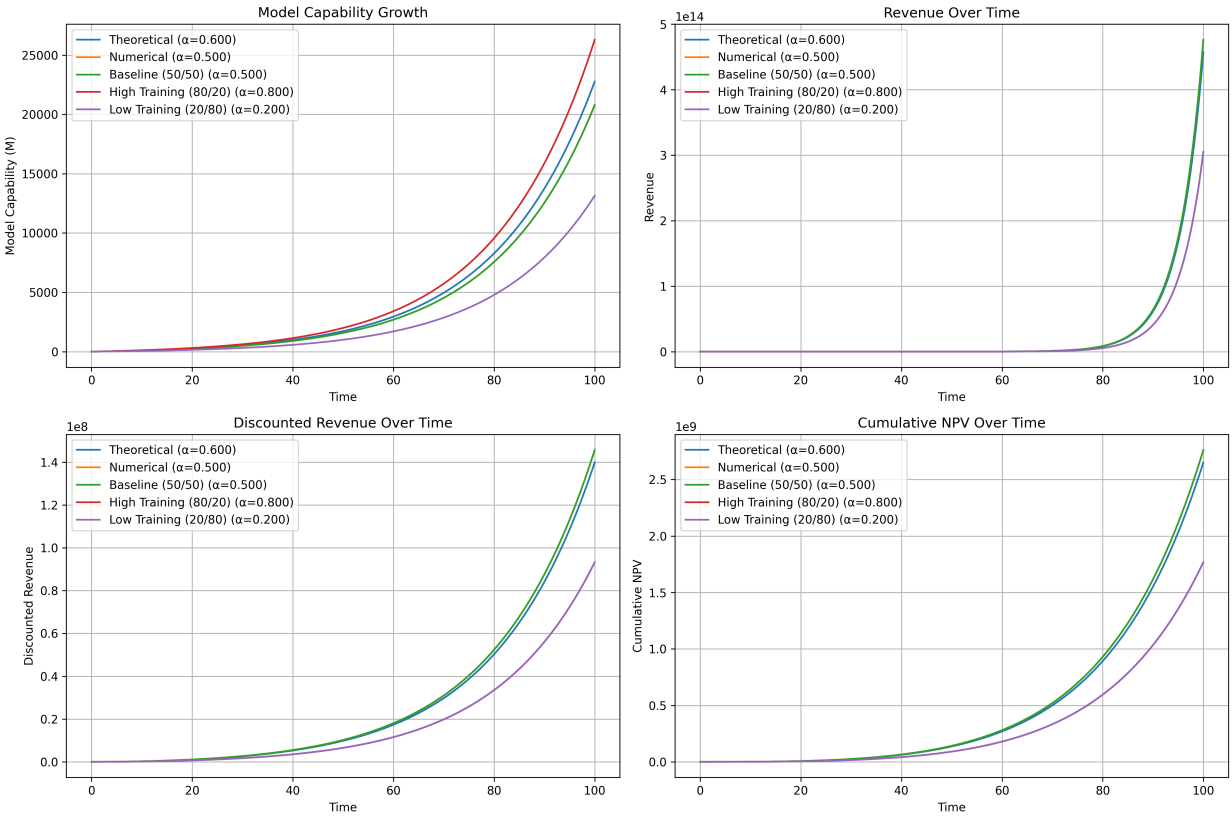

Figure 6: Steady state simulation for varying α

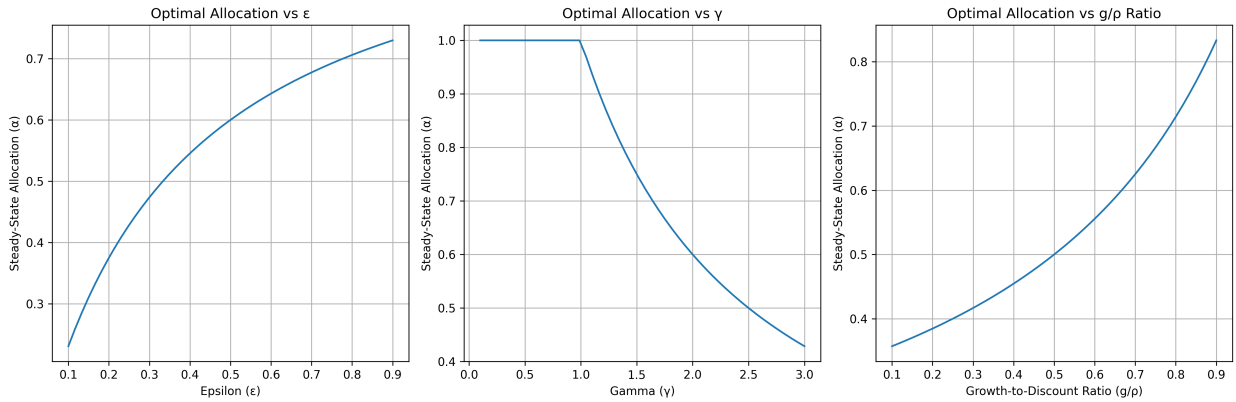

Figure 7: Steady state sensitivity of α for varying ϵ, γ, and g/ρ

4.2 Sensitivity analysis

We perform sensitivity analysis for the optimal allocation for ϵ, γ, and the ratio of compute growth to discounting g/ρ. The result is shown in Fig. 7.

We observe that unless revenue is convex ($\gamma > 1$) in model capability, we always allocate all resources to training, and never release a model. Bold strategy.

5 Conclusion, limitations, and future work

Casting the optimal compute allocation problem as an optimal control problem turns out to work pretty nicely. While we did not encounter any real surprises in our analysis, we hope that this model is of use as a starting point for planners at frontier labs, industry analysts, and others with an interest in the possible future development of AI technology.

Our analysis is limited by the relatively simplistic set of parameters we've chosen to model. Notably, we assume that labs always have access to an exponentially increasing amount of compute – in practice, one has to be really good at fundraising to get enough money to get enough GPUs before you've shown any RoI. However, we note that many prominent frontier lab leaders are really good at that, and so our assumption is not altogether unrealistic.

We are also limited by not having internal knowledge of the real proportion of compute allocation at the frontier labs, and so we don't know how well our model reflects reality. We will try to hang out at more SF house parties and get this information.

Future work should fit real values to our parameters, and observer whether model matches

reality. Additionally, it might be entertaining to run monte-carlo simulations to add stochasticity to our model, though we would have to extend our notion of optimality to the stochastic setting using e.g. the Hamilton-Jacobi-Bellman equation [3].

6 Acknowledgements

Thanks to Casey Handmer for checking my math, Kamile Lukosiute for poking at some of my assumptions, and Mónica Belevan for encouragement. Thanks to Claude for help with cleaning up and annotating the code, and writing the README.

References

[1] Ege Erdil. Optimally allocating compute between inference and training, 2024. Accessed: 2025-04-23.

[2] Connie Loizos. Sam altman's leap of faith. *TechCrunch*, 2019. Accessed: 2025-04-21.

[3] Robert F Stengel. *Optimal control and estimation*. Courier Corporation, 1994.