CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Using Neural Tangent Kernel metrics to measure Intrinsic Motivation in Reinforcement Learning

Master's thesis in Computer science and engineering

VILGOT JANSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Using Neural Tangent Kernel metrics to measure Intrinsic Motivation in Reinforcement Learning

VILGOT JANSSON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Using Neural Tangent Kernel metrics to measure Intrinsic Motivation in Reinforcement Learning

VILGOT JANSSON

Supervisor: Devdatt Dubhashi, Department of Computer Science and Engineering
Examiner: Moa Johansson, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Using Neural Tangent Kernel metrics to measure Intrinsic Motivation in Reinforcement Learning

VILGOT JANSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

We aim to investigate if the Neural Tangent Kernel (NTK) is a useful perspective to explain why intrinsic motivation works, in an effort to try to apply theories from supervised learning to the reinforcement learning domain. We find some inconclusive evidence that suggests that an intrinsic motivation, Intrinsic Curiosity Module (ICM), does in fact increase NTK trace as a mechanism to improve performance, and show that NTK trace and other metrics based on the NTK, can be used to artificially select better training sets that decreases test loss.

Keywords: Computer, science, computer science, engineering, project, thesis.

Acknowledgements

Thank you Devdatt, for aiding me with this and thank you Moa, for inspiring me and letting me choose my own idea.

Vilgot Jansson, Gothenburg, 2025-08-26

Contents

List of Figures

List of Tables

1

Introduction

Intrinsic motivation is an important tool in solving the ubiquitous problem of exploring an environment seen in Reinforcement learning (RL). But how can we know that it is working well, and by what mechanism is intrinsic motivation useful? The simplest method of comparing different types of intrinsic motivation is obviously to measure the final reward of the agent. But this only measures the outcome and fails to identify by which mechanism intrinsic motivation works. This is why in this work we will analyze the effect of intrinsic motivation using methods taken from the supervised learning domain.

There are several metrics that we wish to use. First is the method of analyzing the trace of the neural tangent kernel. Using this as a metric has been done previously with good results. [1] saw that decreasing this metric during training increased regularization, and [2] showed that starting with a higher value led to better generalization. This suggests that we could find better generalization results in reinforcement learning if we could find a policy that explores the environment in a way so that the training data increases this metric before training. This sounds very similar to the concept of intrinsic motivation, which is already a way of creating policies aimed at exploring the environment in a way to increase the final rewards. So it is also reasonable to assume that existing intrinsic motivation policies already work by increasing this metric, which is something we want to investigate if it is true or not. Beyond the NTK trace, there are also other NTK metrics that could be used, like NTK entropy and kernel alignment.

But, the hypothesis that existing intrinsic motivation policies increase performance by selecting data points with higher NTK trace presupposes certain facts. It assumes that selecting data points with higher NTK trace increases performance somehow. This idea came from [2] where they showed both starting NTK trace and NTK entropy were correlated with test loss. But only looking at a correlation has 2 problems. First of all, this correlation has two sources of variance, the random training set *and* the randomly initialized model. We are interested in the idea that selecting a training set with higher NTK trace is correlated with lower test loss, but this could be useless if it instead is so that selecting a model with higher NTK trace is correlated with lower test loss.

The second problem, is that a correlation is not necessarily causation. The NTK metrics are measured at the start while the test loss is measured at the end. The final test loss can obviously not cause the NTK metric at the start, but there can

be some sort of underlying condition at the start that impacts both measurements, which could make it useless as a way to lower test loss. We will test this by seeing if selecting training sets with specific NTK metrics has an impact on final test loss to directly see if selecting data points with higher NTK trace could be a mechanism to improve performance.

If it is true that intrinsic motivation in general works by increasing NTK trace, this discovery could lead to advancements in our understanding of both RL and the NTK. The NTK trace could be used as a tool to more accurately measure the effectiveness of intrinsic motivation policies, leading to better intrinsic motivation designs. It would also establish NTK metrics as an effective way of selecting training sets that could be used in other areas like bootstrapping methods.

2

Theory

2.1 A review of Neural Networks

A neural network is a learned function $\hat{f}$ that maps an input x and parameters θ to an output $\hat{f}_\theta(x)$. The purpose of a neural network is to try to approximate a target function f, so $\hat{f}_\theta(x) \approx f(x)$. For simplicity's sake, we will consider value output neural networks, networks with only one output neuron $\hat{f}_\theta : X \to \mathbb{R}$ where X is the set of all possible inputs.

But we will view the NN as a function on ALL inputs at the same time. Call A the training set and B the test set. A and B are subsets of X, so therefore A and $B \in \mathcal{X}$, the power set of X. With this notation, we can see that $f(A) \in \mathbb{R}^n$, where n is the size of the training set, $n = |A|$ and $f(B) \in \mathbb{R}^m, m = |B|$.

2.2 The Neural Tangent Kernel (NTK)

The neural tangent kernel $K_\theta(A, A)$ is a matrix containing the necessary information to describe how a neural networks outputs changes after a training step of gradient decent. Let $K_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. $K_\theta(A, A)$ is therefore the matrix with elements $K_\theta(A, A)_{ij} = K_\theta(x_i, x_j)$ where

$$K_\theta(x_i, x_j) = \nabla_\theta \hat{f}_\theta(x_i) \cdot \nabla_\theta \hat{f}_\theta(x_j) \tag{2.1}$$

and x_i, x_j are the i:th and j:th elements of A [2]. So $K_\theta(A, A)$ is $(n \times n)$ matrix. Notice that $K_\theta(x_i, x_j)$ is symmetric, which means that the NTK is a symmetric matrix.

2.2.1 Gradient Decent

Gradient decent is the simplest way a neural network is trained and it also reveals why the NTK looks like this. During gradient decent, we update the parameters θ with

$$\theta \to \theta + \delta\theta, \text{ with } \delta\theta = -\eta \nabla_\theta L(\theta) \tag{2.2}$$

where $\eta > 0$ is some learning rate hyperparameter and $L(\theta)$ is the empirical loss $L(\theta) = \mathcal{L}(\hat{f}_\theta(A), f(A))$ for some loss function $\mathcal{L}$. We will use the loss function MSE so $\mathcal{L}(x_1, x_2) = (x_1 - x_2)^2$.

The goal of gradient decent is to change the outputs of $\hat{f}_\theta(x)$ to liken $f(x)$. Consider the situation of evaluating the new output $\hat{f}_{\theta+\delta\theta}(x_i)$ of input x_i after training with gradient decent on input x_j. By linearizing $\hat{f}_\theta$ we get

$$\hat{f}_{\theta+\delta\theta}(x_i) = \hat{f}_\theta(x_i) + \nabla_\theta \hat{f}_\theta(x_i) \cdot \delta\theta + \mathcal{O}(\delta\theta^2) \tag{2.3}$$

and with

$$\delta\theta = -\eta \nabla_\theta L(\theta) = -2\eta \nabla_\theta \hat{f}_\theta(x_j)(\hat{f}_\theta(x_j) - f(x_j)) \tag{2.4}$$

the equation becomes

$$\hat{f}_{\theta+\delta\theta}(x_i) = \hat{f}_\theta(x_i) - 2\eta \nabla_\theta \hat{f}_\theta(x_i) \cdot \nabla_\theta \hat{f}_\theta(x_j)(\hat{f}_\theta(x_j) - f(x_j)) + \mathcal{O}(\delta\theta^2). \tag{2.5}$$

For small updates, we can neglect the $\mathcal{O}(\delta\theta^2)$ and see that the change of output at x_i is

$$\begin{aligned} \hat{f}_{\theta+\delta\theta}(x_i) - \hat{f}_\theta(x_i) &\approx -2\eta \nabla_\theta \hat{f}_\theta(x_i) \cdot \nabla_\theta \hat{f}_\theta(x_j)(\hat{f}_\theta(x_j) - f(x_j)) && (2.6) \\ &= -2\eta K_\theta(x_i, x_j)(\hat{f}_\theta(x_j) - f(x_j)) && (2.7) \end{aligned}$$

and in fact, we can approximate the change of output in all of A after training with gradient decent on all of A with

$$\hat{f}_{\theta+\delta\theta}(A) - \hat{f}_\theta(A) \approx -2\eta K_\theta(A, A)(\hat{f}_\theta(A) - f(A)). \tag{2.8}$$

This means that the NTK is a matrix that converts "desired updates" into an approximation of the actual updates. If we want to update the outputs of $\hat{f}_\theta$ by $(\hat{f}_\theta(A) - f(A))$, then $K_\theta(A, A)(\hat{f}_\theta(A) - f(A))$ is what the actual update will be, due to how training on one data point will also affect the others.

2.3 NTK trace

Simply taking the trace (the sum of the diagonal elements) of $K(A, A)$ can be a useful metric. [2] found in their results that it, $Tr(K)$ measured before training, had a correlation coefficient of -0.5 with test loss. Meaning datasets that incurred a higher $Tr(K)$ was more likely to perform better and have a lower test loss.

This result is also substantiated by some theory. In practice, the NTK trace is a good approximation of the largest eigenvalue [3], which is commonly seen as corresponding with better generalization. Though formally, generalization capabilities are related to the full eigenspectrum. [4]

2.4 NTK entropy

A second metric from the state of the NTK, what I will call the NTK entropy, has its origin in physics and information theory. The NTK entropy is simply the von Neumann entropy of the NTK [2]. The easiest way of the NTK entropy, S is as the Shannon entropy of the eigenspectrum $\{\lambda_i\}_{i=0}^n$,

$$S = -\sum_{i=0}^{n} \lambda_i \ln \lambda_i. \tag{2.9}$$

2.5 Kernel regression

One way of creating this learned function $\hat{f}$ is by using kernel regression. Letting the neural tangent kernel be our kernel, we get

$$\hat{f}(x) = K(x, A)K(A, A)^{-1}f(A). \tag{2.10}$$

Notably, this is also the learned function created from an infinite-width neural network that is trained with gradient decent until 0 training loss. This marks a stark connection between kernel regression and neural network learning.

2.6 Kernel alignment

The standard kernel alignment[5] is

$$D(\theta, A) = \frac{f(A)^T K(A, A) f(A)}{||K(A, A)||_F ||f(A)||^2}. \tag{2.11}$$

In addition, this work also uses a kernel test alignment

$$D'(\theta, A, B) = \frac{f(B)^T K(B, A) f(A)}{||K(B, A)||_F \cdot ||f(A)|| \cdot ||f(B)||}. \tag{2.12}$$

The reasoning behind this is that if both kernel alignments are maximized by $K(A, A) = f(A)f(A)^T$ and $K(B, A) = f(B)f(A)^T$ then the kernel regression from 2.10 becomes

$$\hat{f}(B) = \Big(f(B)f(A)^T\Big)\Big(f(A)^{T,-1}f(A)^{-1}\Big)f(A) = f(B) \tag{2.13}$$

which is a perfect approximation. And since kernel regression is linked to wide neural networks there is reason to believe that together, these alignments are linked to a models capability to generalize to a test set.

2.7 Reinforcement learning

We will provide a slight review of RL as an introduction to the reader. RL is a field of how to optimize the behavior (a policy) of an agent so it gets the maximum cumulative reward. The mathematical model to describe this is typically a Markov decision process.

A Markov decision process consists of:

- S, the set of possible states,
- A, the set of possible actions,
- the transition function $P_a(s, s') = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$,
- the reward function $R_a(s, s')$.

An agent starts with a state s_0 and is then asked to choose an action a_t for each state s_t it is in. The logic used to choose one action is typically a function called a policy, $a_t = \pi(s_t)$. Choosing an action a_t determines what the next state will be using the transition function, s_{t+1} is sampled from $P_{a_t}(s_t, s')$. Which in turn decides the immediate reward from this action, $r_t = R_{a_t}(s_t, s_{t+1})$. The goal of RL is then to optimize $\pi(s)$ to maximize the return $g = \sum_{t=0}^{T} r_t \gamma^t$ for some discount factor $\gamma \in (0, 1]$.

2.7.1 Deep Q Network

A deep Q network is an algorithm to produce an agent (policy) that can maximize its reward in an environment (Markov decision process). It it used in this work due to its simplicity which makes it natural to apply concepts like the NTK.

It consists of a neural network $\hat{f}_\theta(s, a)$ called the Q function. The goal of the Q function is to for each pair of state and action, $\hat{f}_\theta(s, a)$ approximates the expected maximum return for using that action in that state. It basically evaluates each possible action you could do by how much reward you might get in the future, and that is why it can also sometimes be called a value function.

To train the Q function, data is needed. The agent is placed to navigate the environment using some sort of policy. A typical policy is the ϵ-greedy policy, which has probability $1 - \epsilon$ that $\pi_\epsilon(s_t)$, the selected action, is the legal action that maximizes $\hat{f}_\theta(s, a)$, but probability ϵ that $\pi_\epsilon(s_t)$ is a randomly selected action for some typically small value ϵ. These experiences are recorded and stored in a replay buffer as a collection (s, a, r, s'), the current state, selected action, received reward and received state. $\hat{f}_\theta(s, a)$ is then updated using the loss

$$L(\theta) = (r + \gamma \max_{a'} f_\theta(s', a') - f_\theta(s, a))^2. \tag{2.14}$$

2.7.2 Replay buffer

To train a neural network we need a lot training points. If the model only train once on each data point once as soon as we leave that environment state, the training will be severely bottlenecked by how fast the agent can step through the environment. Also, training on single data points will not let us make use of optimizations due to parallelization in network training. That is why the experiences are recorded like this and then trained on multiple times instead of instantly discarding this data.

A naive way would be to store every single experienced state, but this runs into the problem that that the amount of states quickly increases and soon the training will bottleneck the stepping. The simple solution to this is the replay buffer, storage for a constant amount of datapoints. Training for the network is done on the examples stored in the replay buffer, and when the buffer is full, old data points are ejected and forgotten to make place for the new observations.

This replay buffer is therefore the "training set" for $\hat{f}_\theta(s, a)$. Which is why calculating the NTK for a deep Q network is done by calculating the Q networks NTK in relation to the replay buffer.

2.7.3 Intrinsic motivation

Intrinsic motivation is a tool to help design a policy that properly explores the possible states. To illustrate this, it is easier to explain what isn't intrinsic motivation. A full greedy policy that always selects the action that maximizes a return is not intrinsic, it is an extrinsic motivation. Note that it is not just one greedy policy, there is a family of different greedy policies differentiated by if they focus on short-term or long-term investments. But they are all fully extrinsic since they seek to maximize a return.

Intrinsic motivation is a concept to create policies that don't only try to maximize a return. The simplest intrinsic motivation is to select actions randomly, so ϵ-greedy policies has a degree of intrinsic motivation. This work uses a more sophisticated method called an Intrinsic Curiosity Module (ICM) [6] as implemented by the library RLLTE eXplore.

The strength of a intrinsic motivation can usually be set as a parameter. With ICM, the strength is decided by a parameter β. $\beta = 0$ indicates that the ICM has no effect, while higher values of β means the agent is more likely to perform actions that are not intended to give rewards.

3

Methods

There are four main experiments. The first experiment $\mathcal{A}$ is to see if the evolution of NTK trace is related to the prevalence of intrinsic motivation during RL training. The second experiment $\mathcal{B}$ aims to answer if there is a meaningful correlation between any of the NTK metrics and final test loss. This is also done while fixing either the data or the model, this is to attribute this correlation to either a change in data set or change in model, since otherwise both of these can influence the NTK metrics. The third experiment $\mathcal{C}$ is the next step after the results of experiment two. It tests if these correlations is actually a causal relation, by trying to select training sets with specific NTK metrics in order to lower the final test loss. The last, fourth experiment $\mathcal{D}$ is to see if choosing what data points are in the replay buffer to increase the NTK trace can increase rewards. Choosing what actions to take to do this would be difficult, the simpler method that $\mathcal{D}$ will use is to instead expel the data points with the lowest NTK trace out of the replay buffer.

Some methods in this chapter are relevant for the execution of multiple experiments, so to make this relationship clear, each method sub-section is accompanied by a set of letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ indicating which experiments this explanation pertains.

3.1 RL training setup ($\mathcal{A}, \mathcal{D}$)

The RL experiments is done on a setup using the python library "Stable baselines 3". The environment agents are trained on the environment "Acrobot-v1" from Gymnasium which is a collection of reference RL environments. In this environment, the agent controls a driven double pendulum and the goal is to swing the end up to a desired height in as few steps as possible by applying either clockwise or counter clockwise torque. Every step that does not reach this receives a reward of -1, while if the height is reached, the simulation terminates and the agent receives 0 reward.

This environment was chosen due to simplicity, it has a low dimensional continuous input and a discrete output which is necessary when using a Deep Q network. Without specifying any additional parameters the Q network is a dense feedforward neural network with two hidden layers of width 64.

Variable Name	Role	Type
displacement	Feature	Continuous
mpg	Target	Continuous
cylinders	Feature	Integer
horsepower	Feature	Continuous
weight	Feature	Continuous
acceleration	Feature	Continuous
model_year	Feature	Integer
origin	Feature	Integer
car_name	ID	Categorical

Figure 3.1: Description of the variables for the fuel dataset.

3.2 Data set for supervised learning ($\mathcal{B}, \mathcal{C}$)

The non-RL training experiments in this work is done on the Auto MPG dataset (referred to as the fuel dataset) from the UC Irvine machine learning repository. It is a fairly small dataset of 398 data points. It is a single value regression problem with mixed input of both integers and continuous. It also has a categorical input (car_name), but that is discarded in this work for the sake of simplicity. A full description of features of the fuel data set is shown in Figure 3.1.

3.3 Measuring NTK metrics during training ($\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$)

The idea is to measure how NTK metrics evolve during training. The naive approach would be to calculate the NTK metric for the whole dataset after each update to the model, or after each epoch. With n training points, the NTK becomes $n \times n$, and the NTK entropy requires calculating the eigenvalues of the NTK, which has a time complexity of $\mathcal{O}(n^3)$. This runs into problems for large datasets, for example in RL where there can be lots of states.

A solution is to instead regard a mini-batch of m samples as the entire training set to calculate a smaller $m \times m$ NTK. Using this NTK to calculate the metrics can be used as an approximation of the real metric. As long as the NTK trace is normalized to compensate for the different sizes of the NTKs. In this work, when using the fuel dataset the NTKs are calculated on the full training since the fuel dataset is small enough to do that.

The NTK metric calculation during the RL training can be done using the mini-batch approach, but due to a limitation of the RL training library used in this work, it is difficult to the get access to the exact batch used for updating the model, so

instead, in this work the NTK calculations on the RL training is done by randomly sampling an arbitrary number (1000) of data points from the replay buffer.

3.4 Variance in data vs. model ($\mathcal{B}$)

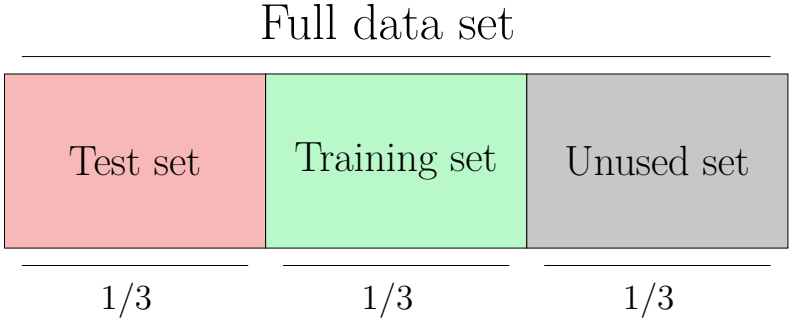
The second test aims to verify if NTK trace is an indicator of lower final test loss and more importantly, the role of the training data set. Before training a model there are two sources of variance. One is the model, its architecture and the randomly initialized weights. The second source of variance is the datasets, specifically what data is chosen to be used for training. Both of these will affect what the NTK trace initially is, and then also affect the training and what the final test loss will be. We are specifically interested in selecting a dataset to increase the NTK trace as a way of decreasing the test loss. Even if initial NTK trace is negatively correlated with final test loss, if that correlation is only due to changes in the model, selecting data won't impact the test loss.

To test this, we train a model many times, and measure the correlation of NTK trace before training with test loss after training, first with randomly sampled training points and randomly initialized model. Secondly with randomly sampled training data and a fixed model. Thirdly with a fixed set of training data and randomly initialized model.

But note, that when experimenting with a fixed object, like a fixed model, we can't run the experiments with only a single model. That would only calculate the correlations when using that single model. Instead, the experiments is ran several times with different, fixed models giving us a statistical description of what the correlations are when the model is fixed.

3.4.1 De-correlating training and test sets ($\mathcal{B}, \mathcal{C}$)

Whenever training sets and test sets are sampled from the fuel datasets for experiments (the metrics correlation tests and the artificially selected training set tests) the training set is 1/3 of the fuel data set, the test set is 1/3 of the data set, and 1/3 of the data set is not in the training set or the test set. Also, the training set and test set are disjoint. This is done to de-correlate the selection of the training set to the selection of the test set. If the training set and test set form a perfect partition of the full dataset, selecting a specific training set also indirectly selects a specific test set. That means that if the "easy" data points are in the training set, the "difficult" data points will be left in the test set, leading to an unproductive scenario where minimizing one of the losses will increase the other. This argument is supported by results. When the fixed model experiment was run with training and test sets that partitioned the fuel data set, training loss and test loss had a correlation of -0.28. While with the de-correlated sets the same correlation was -0.02.



3.5 Stronger indicators - NTK entropy and Kernel alignment ($\mathcal{B}$)

Seeing that NTK trace is very weakly correlated with final test loss, the solution might lie in a NTK metric with a stronger correlation. Two candidates we will test here are NTK entropy and Kernel alignment.

3.6 Artificially selecting datasets ($\mathcal{C}$)

Seeing that datasets with higher NTK trace/entropy/kernel alignment all tend to have lower test loss, an obvious test is to see if artificially selecting these datasets to maximize one of these values leads to a lower test loss. To test this, methods were developed to select datasets and then trained on.

It is important to note that the training and test set was de-correlated as previously explained. So for each experiment, a test set consisting of 1/3 of the data set was randomly sampled. A training set of the same size was then selected from the remaining 2/3 of the data set.

3.6.1 Selecting training set with maximal NTK trace ($\mathcal{C}, \mathcal{D}$)

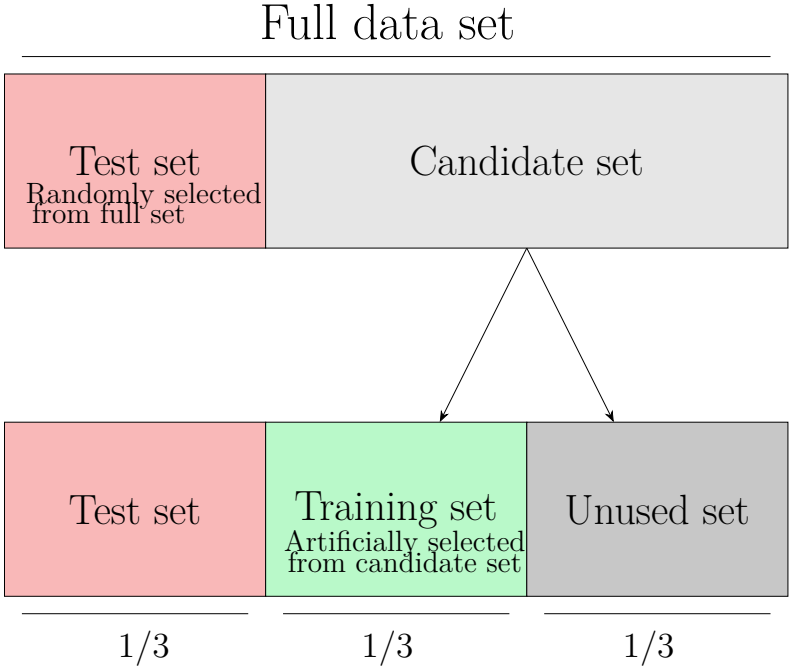
Selecting the training set with the highest NTK trace is quite easy since the contributions from each training point is independent from the others. Expressing it mathematically, the NTK trace T,

$$T = \sum_{x \in A} K(x, x). \tag{3.1}$$

T is maximized if $x \in A, y \notin A \Rightarrow K(x, x) > K(y, y)$. This is achieved by calculating $K(x, x)$ for all $x \in B$ and then selecting the n x's with the highest values of $K(x, x)$ to be in A.

3.6.2 Selecting training set with higher NTK metric ($\mathcal{C}$)

Selecting a training set to maximize for example kernel alignment is more difficult since there is no trivial way to measure the contribution of each data point in a



vacuum. The method used does not guarantee that the optimal configuration is found, it only aims to find a configuration that is better than random.

A candidate training dataset was sampled randomly and then augmented by randomly choosing one data point in the training set and one data point outside of the set. It was then tested to see: if the training data point is exchanged for the outside point, would the kernel alignment increase? If so, the training point is exchanged for the outside point and the resulting training set has a slightly higher kernel alignment. This augmentation is then tried for an arbitrary amount of times (100).

This method is very basic and metric agnostic, so it can also be used to select for NTK trace, kernel test alignment and NTK entropy.

3.7 Expelling low NTK trace data from replay buffer ($\mathcal{D}$)

For the fourth experiment $\mathcal{D}$, the replay buffer is set to a smaller value of 4000 to increase the effect of choosing these points wisely. The expelling is done on every step, so after each step in the environment, $K(x, x)$ is calculated for each x in the replay buffer and the point with the lowest value is marked for expulsion. This means that its data will be removed to make room for storing the next data point received after taking the next step in the environment. Meaning that the replay buffer is never larger than 4000 and only removes the data point contributing the least to the NTK trace.

Setting the replay buffer to only store 4000 values also has a dramatic effect so to measure the effect of selecting what data points remain, it is compared to a control group that trains with the same parameters and a replay buffer of 4000 points, but

doesn't select specific points for its replay buffer. The control group's replay buffer operates with a typical first-in first-out principle instead.

4

Results and Discussion

4.1 NTK Trace during RL ($\mathcal{A}$)

The results from the first experiment, seeing if NTK trace is increased by intrinsic motivation, are presented in Figure 4.1 and Figure 4.2.

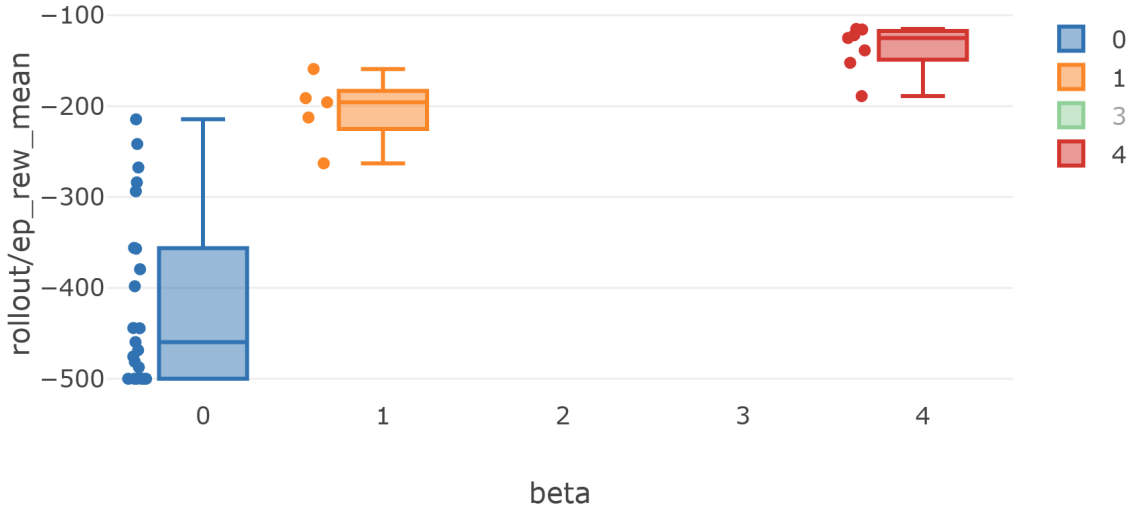


Figure 4.1: Average rewards during RL training with different levels of intrinsic motivation strength (beta). 0 beta is equivalent to no intrinsic motivation, 1 beta is default strength and 4 beta is extra strong intrinsic motivation. It is clear that intrinsic motivation leads to a higher return.

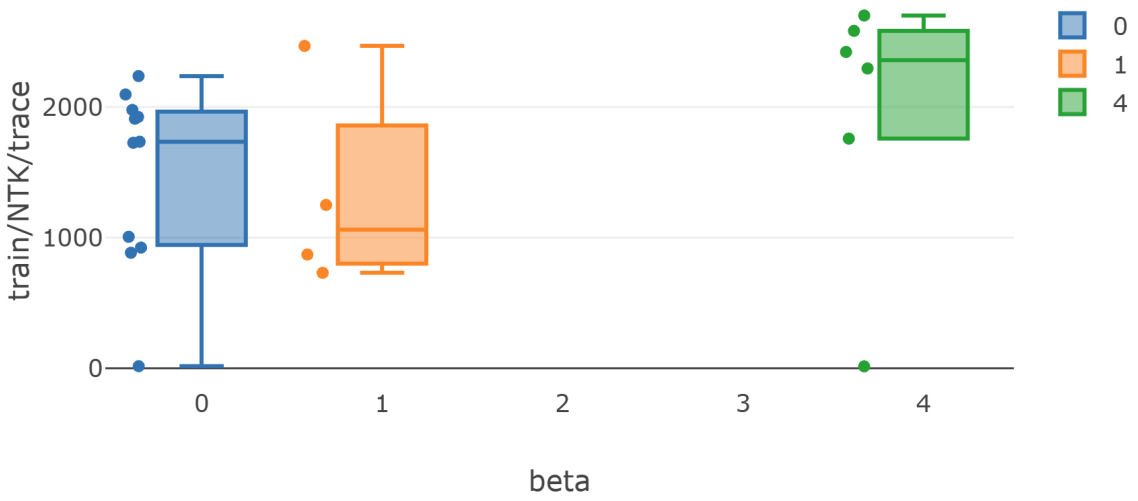


Figure 4.2: Final NTK trace after training RL with different levels of intrinsic motivation strength (beta). The variance is very large and it is unclear if intrinsic motivation has any effect. And even if there is an effect, it could be indirect.

Looking at Figures 4.1 we can see that intrinsic motivation does lead to more rewards even if it does not maximize reward in the short term. But from Figure 4.2 it is difficult to tell if intrinsic motivation directly increases NTK trace. First of all, NTK trace naturally has a large variance so it is difficult to tell if intrinsic motivation had an impact. But it does look like there is some positive correlation between the two.

Even if they are positively correlated, that doesn't imply that intrinsic motivation *directly* causes NTK trace to rise. It is still possible that intrinsic motivation causes the model to achieve higher rewards, and higher rewards might increase NTK trace. This would mean intrinsic motivation indirectly increases NTK trace, which does not answer the main question.

To answer the main question, i.e., if intrinsic motivation increases NTK trace to increase rewards, a more meticulous method is needed. For example, to record every single change in the NTK trace, and attribute these changes to either training or recording a new experience. This could isolate intrinsic motivations impact on the NTK trace from the training process that has a very large and unpredictable impact on the NTK trace. Which would make the impact more measurable.

4.2 NTK metric correlations ($\mathcal{B}$)

The results from the second experiment, finding correlations between NTK metrics and test loss, are presented in Table 4.1

	Baseline	Fixed model	Fixed data
Initial NTK trace	0.142	-0.036 (± 0.16)	-0.11 (± 0.20)
Initial Kernel alignment	-0.015	-0.008 (± 0.15)	0.17 (± 0.21)
Initial Kernel test alignment	-0.002	-0.041 (± 0.17)	0.21 (± 0.27)
Initial NTK entropy	-0.021	-0.087 (± 0.15)	-0.03 (± 0.20)

Table 4.1: Means and standard deviations of correlations against final test loss for different experiments. Fixed model has randomly sampled data, fixed data has randomly initialized model and baseline has randomly sampled data and randomly initialized model. The strongest influences over the test loss when the model is fixed, is initial NTK trace, kernel test alignment and entropy.

We see from table 4.1 that these NTK metric have some level of negative correlation with test loss when we only vary the data. Suggesting that selecting training sets with high metrics would in general lead to lower test loss and better generalization. We can interpret this statistics as if we randomly initialize a model, the correlation between the initial NTK trace and the final test loss will be sampled from a normal distribution with mean -0.036 and standard deviation of 0.16. This means that for most models, increasing the NTK trace would lower test loss, but for some models, increasing the NTK trace tends to increase test loss. So in general, it is good to increase NTK trace of the training set.

A data point that needs to be discussed is the baseline correlation for the initial NTK trace at 0.142. This is unusual since most baseline correlations are somewhere in between the fixed model mean and the fixed data mean. The most likely explanation is that the Fixed data-initial NTK trace correlation mean is inaccurate since the fixed data experiments were done with a more limited sample size. Considering that the other correlations seem to be more positive the data is fixed, it is possible that NTK trace correlation *should* also be more positive.

4.3 Selecting training set ($\mathcal{C}$)

The results from the third experiment, Selecting training sets, are presented in Table 4.2, Table 4.3, Table 4.4, Table 4.5, and Table 4.6.

	Baseline	Selected data	Selected data and switched model
Final test loss	0.727	0.647	0.831
Final training loss	0.201	**0.167**	**0.165**
Initial NTK trace	3.246	3.193	3.246
Initial Kernel alignment	0.751	0.886	0.858
Initial Kernel test alignment	0.705	0.852	0.822
Initial NTK entropy	1.549	1.537	1.531

Table 4.3: Average metrics for training on a dataset selected to artificially increase initial Kernel alignment.

	Baseline	Selected data	Selected data and switched model
Final test loss	0.727	0.661	0.666
Final training loss	0.201	0.186	0.208
Initial NTK trace	3.246	3.179	3.204
Initial Kernel alignment	0.751	0.823	0.799
Initial Kernel test alignment	0.705	0.803	0.768
Initial NTK entropy	1.549	1.549	1.542

Table 4.4: Average metrics for training on a dataset selected to artificially increase initial Kernel test alignment.

	Baseline	Selected data
Final test loss	0.727	0.668
Final training loss	0.201	0.208
Initial NTK trace	3.246	3.618
Initial Kernel alignment	0.751	0.773
Initial Kernel test alignment	0.705	0.722
Initial NTK entropy	1.549	1.552

Table 4.2: Average metrics for training on an dataset selected to artificially increase initial NTK trace. Baseline is trained on uniformly randomly sampled training set. Selected data is trained on training set with increased initial NTK trace. Selected data and switched model is trained on a training set with increased NTK trace, but the model is randomly initialized again after the training set was selected.

Regarding selecting better training sets, even though NTK metrics evaluates the data given a specific model, most of the improvements remained even after swapping the original model for a new randomly initialized one. This means that the selected training set is good in a pretty robust way. Perhaps this could be used to used to evaluate data points agnostic to the choice of model.

The most useful NTK metric was surprisingly, the NTK trace. Despite being the absolute simplest, it had a impact roughly equal to the other NTK metrics when the training sets were iteratively selected. What really made it stand out was the ability to select the training set with the maximal NTK trace, which was faster than

	Baseline	Selected data	Selected data and switched model
Final test loss	0.727	0.658	0.706
Final training loss	0.201	0.210	0.206
Initial NTK trace	3.246	3.265	3.352
Initial Kernel alignment	0.751	0.750	0.770
Initial Kernel test alignment	0.705	0.748	0.776
Initial NTK entropy	1.549	1.628	1.621

Table 4.5: Average metrics for training on a dataset selected to artificially increase initial NTK entropy.

	Baseline	Selected data	Selected data and switched model
Final test loss	0.727	**0.461**	**0.457**
Final training loss	0.201	0.183	0.193
Initial NTK trace	3.246	4.097	4.052
Initial Kernel alignment	0.751	0.768	0.772
Initial Kernel test alignment	0.705	0.571	0.609
Initial NTK entropy	1.549	1.555	1.546

Table 4.6: Average metrics for training on a dataset selected with the maximal initial NTK Trace.

the iterative process and magnitudes better, like seen in Table 4.6.

This result is partially surprising. The idea that a good training set can be assembled by selecting data points independently without looking at the other data points is frankly ridiculous. To illustrate, if we had 100 copies of each training point, maximizing the NTK trace would require selecting 100 copies of the same data point with the maximal $K(x, x)$, which is obviously a very bad design for a training set.

But there is also theory that could explain why this works. High NTK trace can be seen as an indicator of good generalization, but this only works if the test data is sampled from the same distribution as the training data. And while both training set and test set will come from the fuel data set, it is not completely clear if the choosing the high NTK trace data can effectively be seen as randomly sampled. Because seeing that the generalization gain from selecting a high NTK trace training set was not affected if the model is reinitialized tells us that some data have an intrinsic property of being high $K(x, x)$ (at least from the perspective of this model architecture) so selecting them would not be random. Though, if the $K(x, x)$ if high frequency enough, we could see it as a pseudo-random number generator which would make the high NTK trace selection effectively random. This idea that the value of $K(x, x)$ can be seen as random is also echoed in some studies[7].

We had a hypothesis that this performance boost could come from high NTK trace artificially "accelerating" training in a similar way to increasing the learning rate does. So if we trained the two models for longer, their differences would disappear. But no, the models were well and truly finished training. Giving them more time

made no difference. Selecting for high NTK trace truly did find training sets that generalized better.

Initial kernel alignment also deserves to be mentioned. Selecting training sets with higher kernel alignment had the lowest training loss, and the test loss even increased after switching out the model. It is likely that this method effectively selects the best training set to overfit on.

4.4 Expelling low NTK trace points from Replay buffer ($\mathcal{D}$)

The results from the fourth experiment is shown in Figure 4.3.

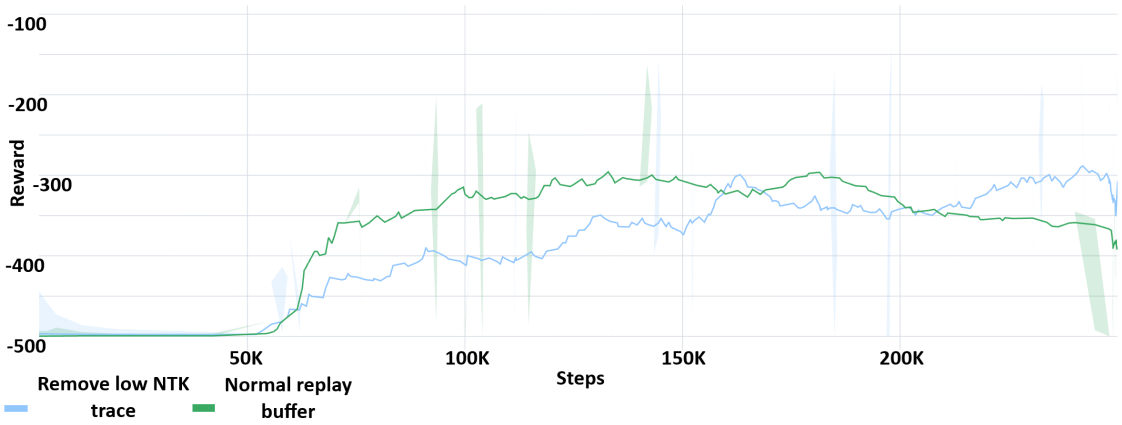


Figure 4.3: Average reward during training for different types of replay buffers. The normal replay buffer gain rewards much quicker, but then drops. The modified replay buffer learns slower but doesn't lose rewards. They both achieve the same peak of -300 reward.

We can see from Figure 4.3 that removing the lowest NTK trace point from the replay buffer has a big impact to how the agent learns. It is also important to note that the control group is also behaving in an unsual manner. Since the control group also has a very small replay buffer of only 4000 points, after the initial bump in reward of learning a strategy, the agent tends to "forget" how to achieve that reward as the early data points are removed from the replay buffer. A typical agent would probably have a much larger replay buffer and therefore not forget this much.

Going back to the modified method, we see that it learns it environment much slower, but with the upside that it didn't forget over time. This could mean that this method actually does a good job of selecting a set of training points that leads to a robust understanding of the environment without forgetting. But, it could also just mean that the modified version changes slower, and that if they were trained for longer, even the modified version would forget. Since currently, the modified version seems to reach its peak right at the end of training, but there is no way of knowing if it drops after the peak or not without rerunning the experiments for a longer time.

Either way, any possible positive effect seems to be overshadowed by the negatives. It might lead to more robust learning, but probably only during the unusual case of having an unnaturally restrictive replay buffer, and at a huge cost of speed since it both requires more training steps and slows down each training step.

But even if this method is not useful, just the result that it might have some positive effect is enough to corroborate the idea that the NTK trace of the training set is

linked with some important concept of machine learning. Since selecting a replay buffer from some arbitrary metric would likely be catastrophic for training.

4.5 Limitations and future work

As previously noted, the obvious next step is categorize the contributions to the NTK during training. Additionally, the RL tests were only done on a single RL environment. There is a wide arrangement of possible RL environments so it is possible to have different results on another environment. It would for example be interesting to see the behavior of the NTK for environments where intrinsic motivation does not help. Also, this work only examined one version of intrinsic motivation, ICM. But other intrinsic motivation policies might work in a different way and it would be interesting to some day make intrinsic motivation that explicitly tries to increase NTK trace.

Regarding selecting training set with specific NTK metrics, the tests were only done with one data set and with one model architecture. It would be interesting when doing the model swap to swap to a model with an entirely different architecture, like a CNN, and see if the good training data selections really are architecture-agnostic. And also to test the selection method on larger data sets to see if it scales properly. Though note that due to the derivation from the kernel regression, kernel alignment and kernel test alignment might be useless for non-regression tasks.

5

Conclusion

We have shown that selecting a training set with higher NTK trace does improve generalization which supports the idea that NTK trace might be related to mechanism to how intrinsic motivation improves performance. We have also found light evidence that might suggest that usage of intrinsic motivation is correlated with increased NTK trace. But additional research will be needed to prove a causal relationship. In light of this, we can guess that NTK trace is probably not a core component of what makes modern intrinsic motivation work. But we have seen that selecting data point with high NTK trace worked shockingly well for small data sets. It remains to see if this scales to larger models and if it is possible to create intrinsic motivation that purely selects for high NTK trace.

Bibliography

[1] V. Szolnoky, V. Andersson, B. Kulcsár, and R. Jörnsten, "On the interpretability of regularisation for neural networks through model gradient similarity," *Advances in Neural Information Processing Systems*, vol. 35, pp. 16 319–16 330, 2022.

[2] S. Tovey, S. Krippendorf, K. Nikolaou, and C. Holm, "Towards a phenomenological understanding of neural networks: Data," *Machine Learning: Science and Technology*, vol. 4, no. 3, p. 035 040, 2023.

[3] M. Murray, H. Jin, B. Bowman, and G. Montufar, "Characterizing the spectrum of the ntk via a power series expansion," *arXiv preprint arXiv:2211.07844*, 2022.

[4] J. B. Simon, M. Dickens, and M. R. DeWeese, "Neural tangent kernel eigenvalues accurately predict generalization," 2021.

[5] H. Shan and B. Bordelon, "A theory of neural tangent kernel alignment and its influence on training," *arXiv preprint arXiv:2105.14301*, 2021.

[6] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*, PMLR, 2017, pp. 2778–2787.

[7] B. Hanin and M. Nica, "Finite depth and width corrections to the neural tangent kernel," in *International Conference on Learning Representations*, 2020. [Online]. Available: `https://openreview.net/forum?id=SJgndT4KwB`.