Adversarial Robustness Through the Lens of Singular Learning Theory

Introduction

Singular learning theory contextualises statistical learning theory to singular models, of which neural networks are an example (1). The theory introduces the real log canonical threshold, which can be interpreted as a measure of model complexity. Subsequent work has developed the local learning coefficient as a computable metric for model complexity (2) that scales to transformer architectures (3).

It will be the subject of this work to explore the relationship between the local learning coefficient and the adversarial robustness of the corresponding model. We are operating with the intuition that simpler models are more robust. Thus we are expecting to observe that a model trained through adversarial training exhibits a lower local learning coefficient than a model trained through standard techniques.

We can justify this intuition in the following way. Suppose an image classifier operates at the levels of pixels. Namely, it analyses individual pixel values to construct its prediction of the image's class. The relationship between pixels and output class is likely to be complex, and thus the corresponding model is also likely to be complex. Moreover, it is probably the case that the model is more susceptible to adversarial attacks since there are many fine details that influence the resulting output. On the other hand, if the model has abstracted high-level features such as edges, then the function encoded by the model is likely to be simpler. Moreover, the model is more robust to adversarial attacks since attacks would have to modify high-level features of the input.

In this work we utilise the DevInterp Python package (4) to study the relationship between the local learning coefficient and the adversarial robustness of a model. In particular, we aim to answer questions at the model level and at the sample level.
Model level:

- How does the local learning coefficient and the robustness of the model change through standard training and adversarial changes? In particular, can we identify different phases during training using these metrics?
- How does post-hoc finetuning a model to be robust compare to training a model from scratch with adversarial training?
- What effect does model architecture and training hyper-parameters have on the above mentioned relationships?
- Where in the parameter space is majority of the local learning coefficient captured? More precisely, how does the local learning coefficient at different layers of the model evolve through training and finetuning?

Sample level:

- Can we investigate the local learning coefficient on neighbourhoods of samples to identify a relationship between the complexity of the model around a sample and its corresponding robustness?

Experimentation

To explore the previously mentioned questions we set up the following pipeline. We consider an *image classifier* on a low-dimensional dataset to enable the use of relatively small models to ensure the tractability of repeatedly computing local learning coefficient estimates. Specifically, we generate 3-by-3 images delineated equally into three classes. We train feed-forward fully-connected neural networks.

- Standard training refers to using stochastic gradient descent to perform this optimisation.
- Robust training refers to augmenting half the samples of each batch with adversarial perturbations of a maximum size of 2 in the L_∞-norm. These perturbations are determined using projected gradient descent.

A finetuned models is obtained by taking a model trained standardly, and then continuing its training with batches full of adversarial examples.

To measure robustness we perform a projected gradient descent attack on each point in a held-out test set. For each point we record the L_∞ perturbation necessary to cause the corresponding model to reclassify the point. For more robust model we should expect these perturbation amplitudes to be larger. Therefore, for a given ϵ-perturbation, we quantify the attack success rate as the proportion of samples in the held-out test set with perturbations less than this ϵ threshold. Thus, a more robust model has a lower attack success rate for a given ϵ.

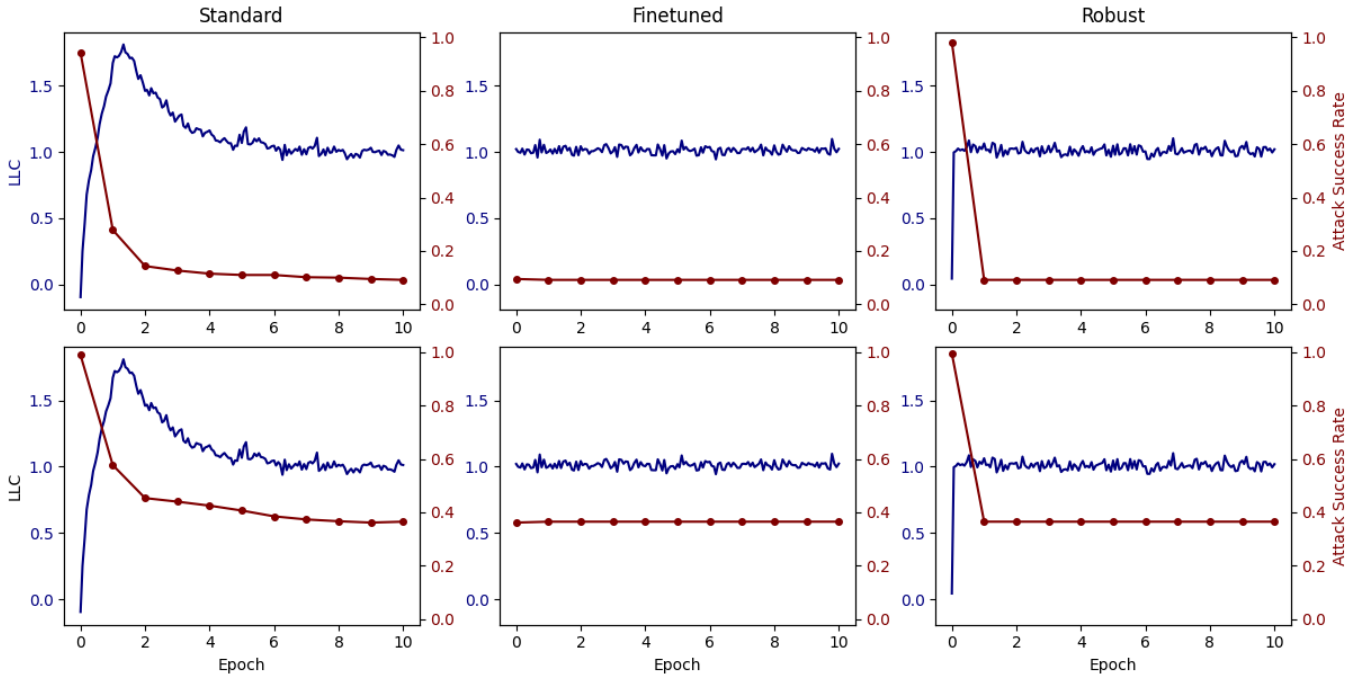
Model-Level

Effects Through Training

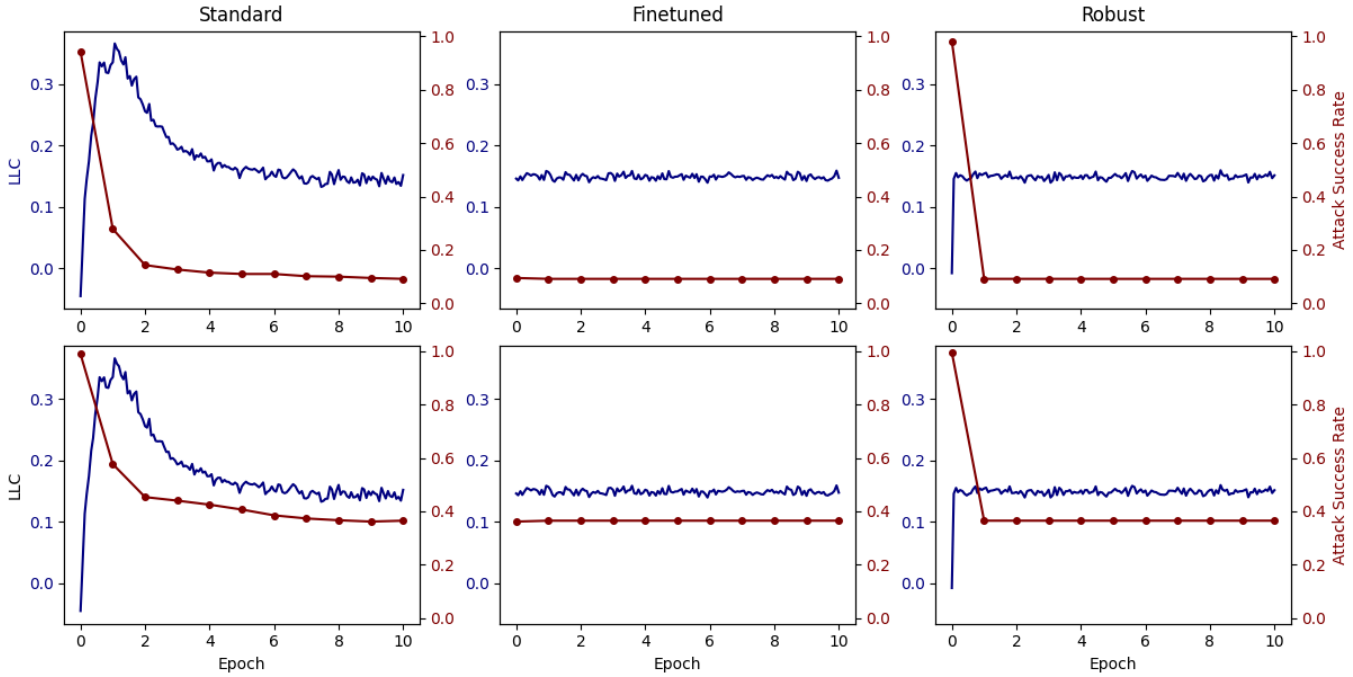
To investigate the relationship between robustness and the local learning coefficient though training, we trained a neural network with 3 hidden layers of size 16. In each stage of training we used stochastic gradient descent with a learning rate of 0.01.

Initial Epochs

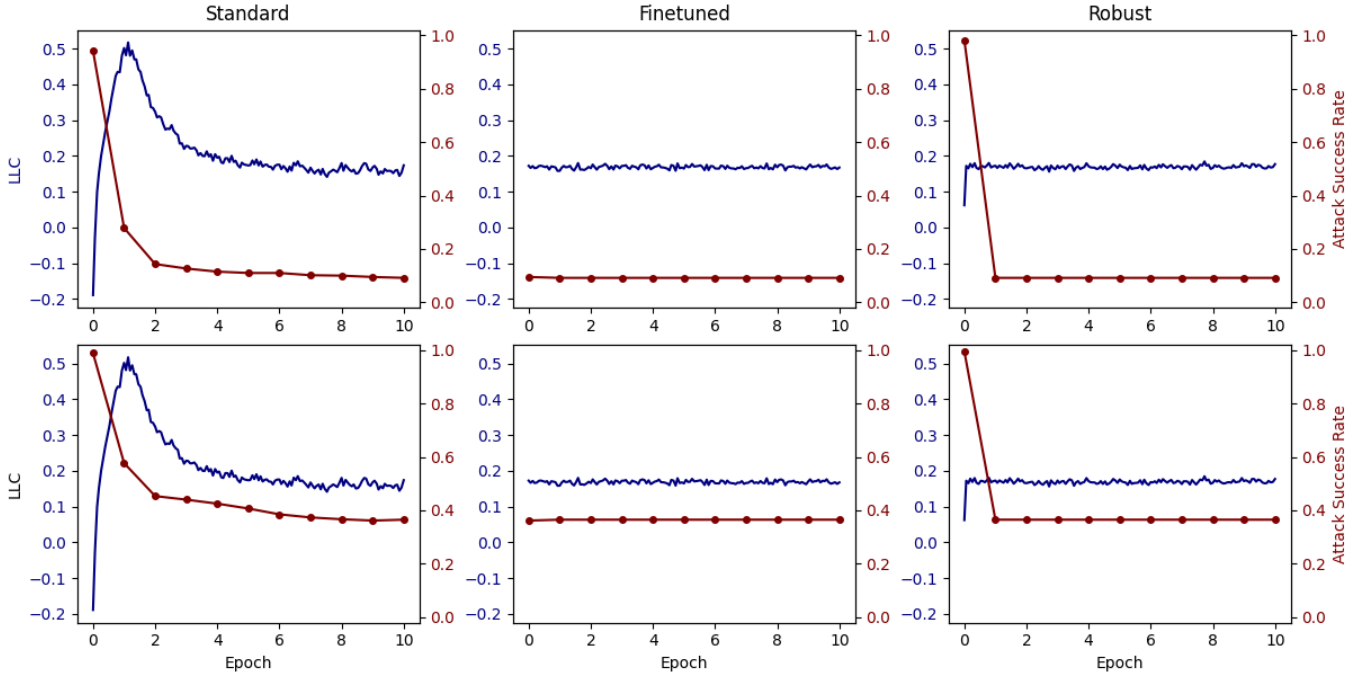
Firstly, we focus on the first few epochs of training, done with a batch size of 32.



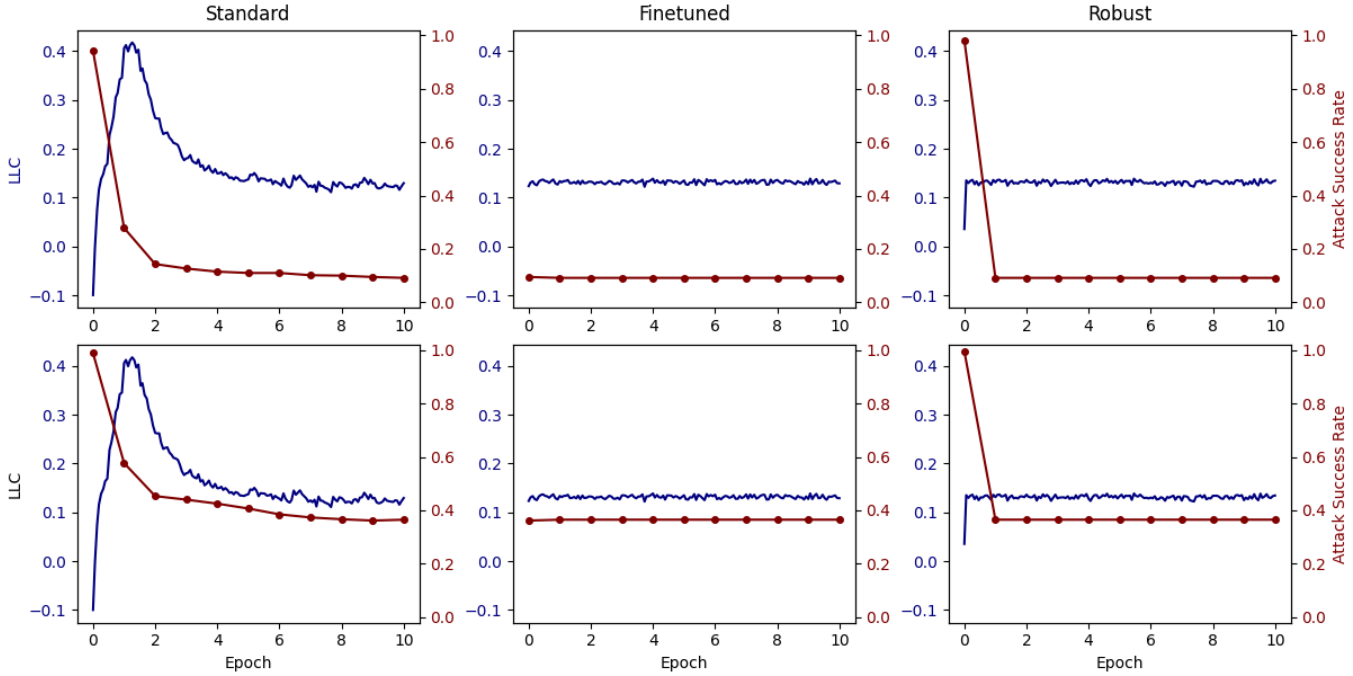
All layers



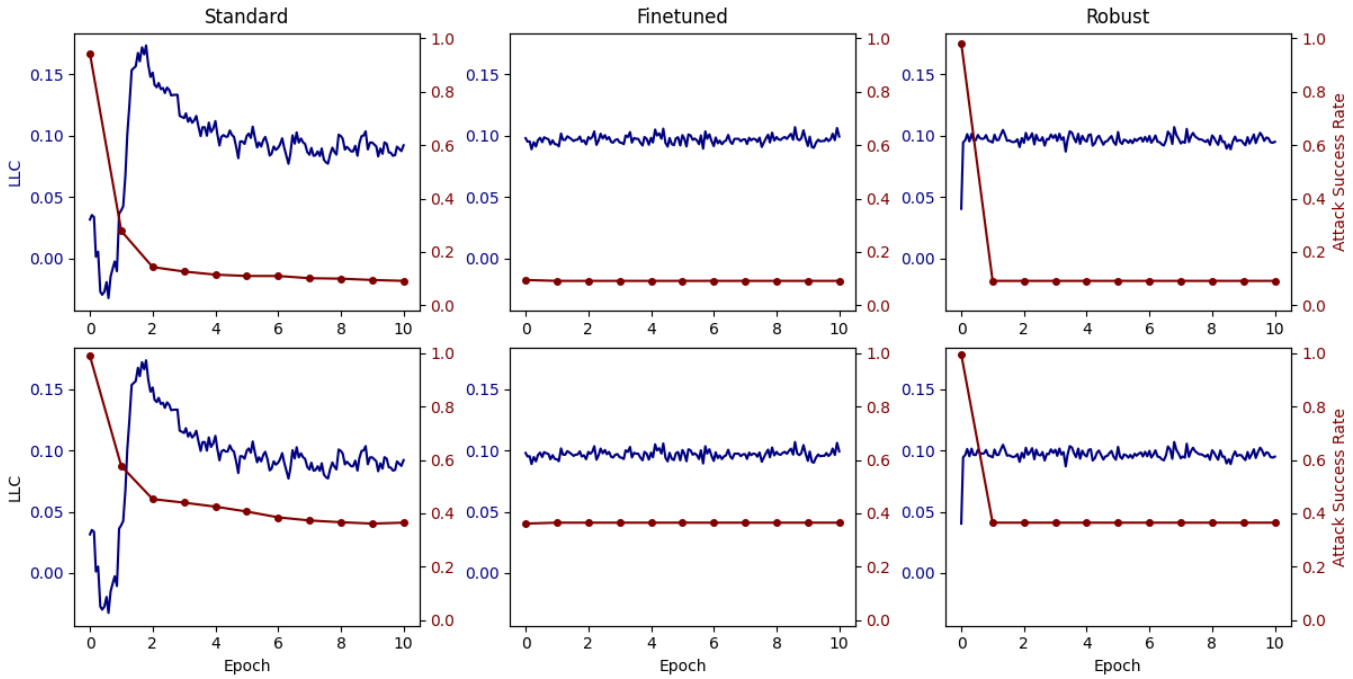
Input layers



First hidden layer



Second hidden layer

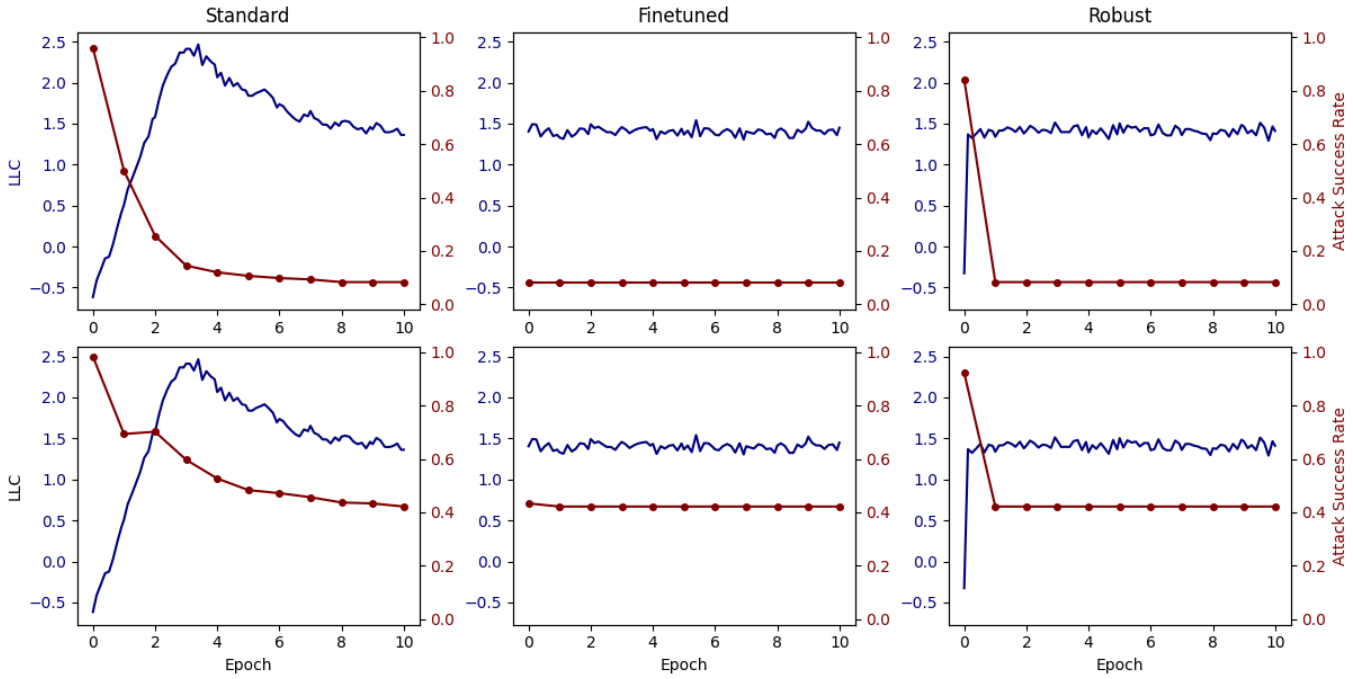


Output layers

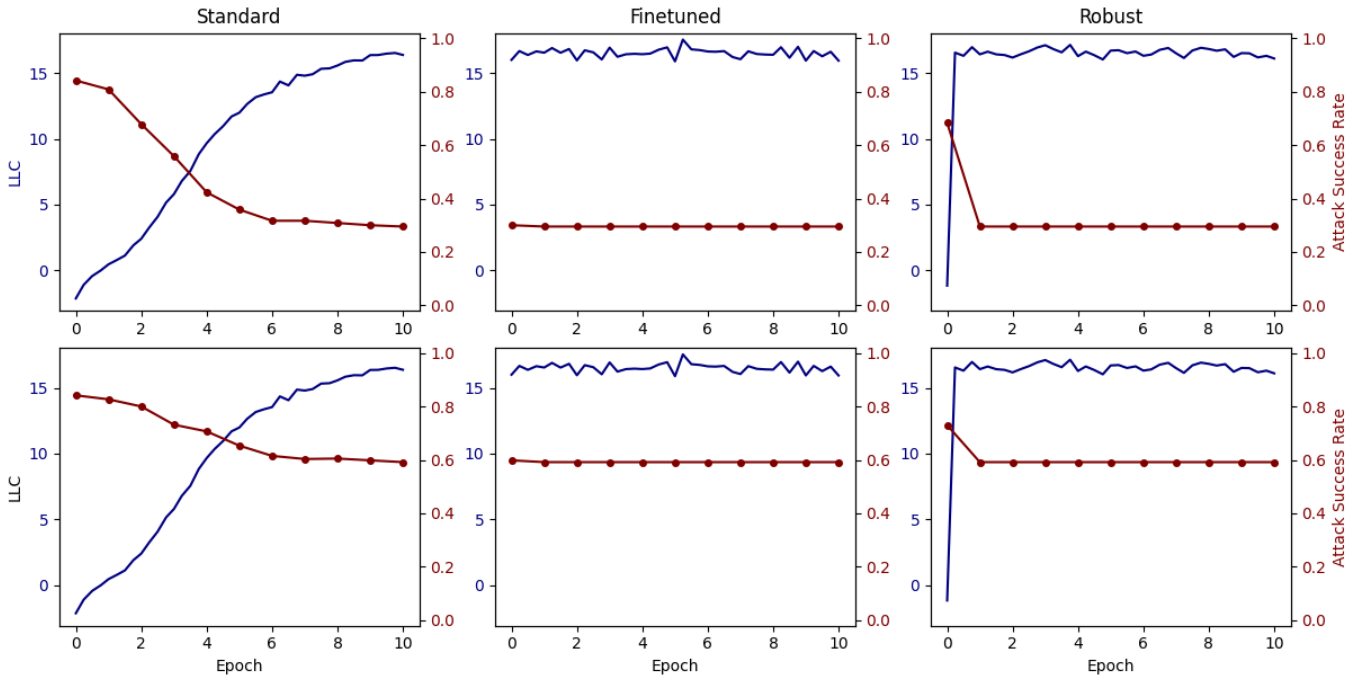
We see that the local learning coefficient is most significantly altered under standard training. In the first few epochs it experiences a spike, which is observed across all layers, and then it settles to the same value as that of the robustly trained model. Finetuning does not seem to effect the local learning coefficient, which is perhaps because the robustness has already been established by this point. It seems as though robustly training the model stabilises the local learning coefficient faster, eliminating its initial spike.

Interesting behaviour is seen in the output layer where the local learning coefficient dips before spiking and then settling. I do not have a clear intuition of why this would be the case.

Increasing the batch size with which these models are trained can alter these observed patterns.



Batch size 64

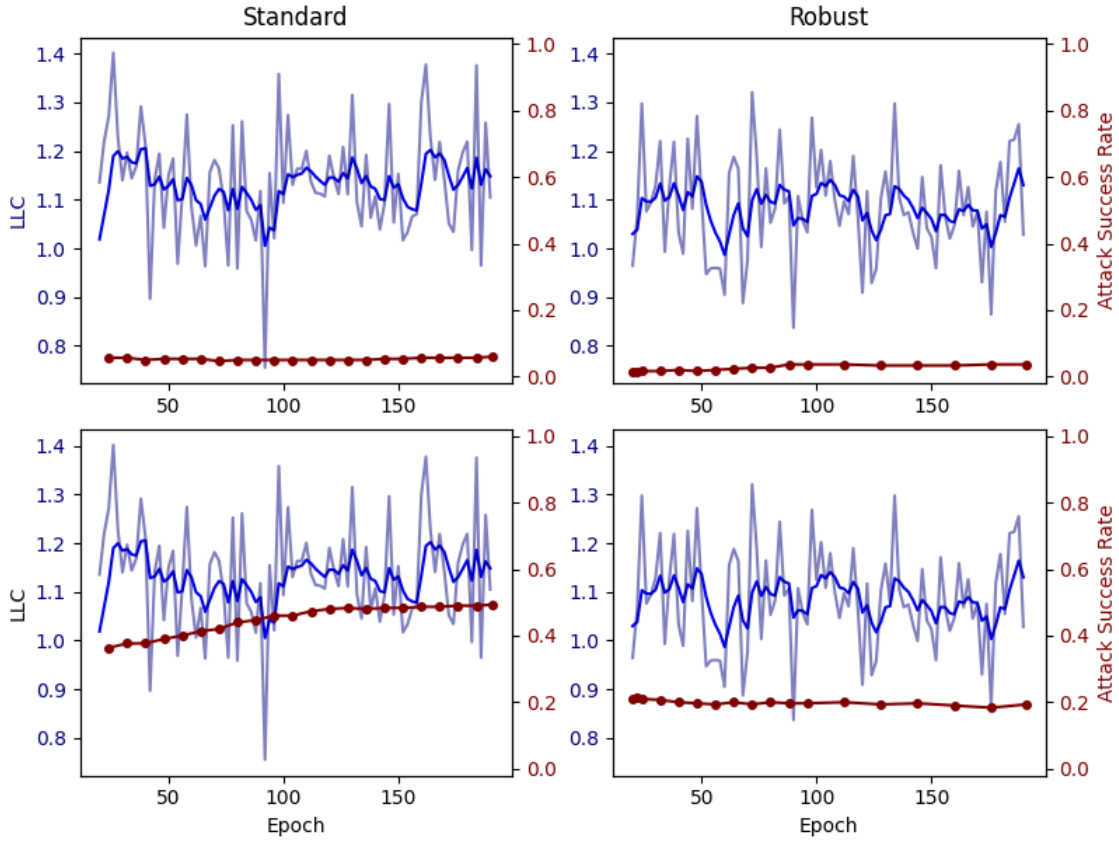


Batch size 128

We observe again that robustly training the model stabilises the local learning coefficient and achieves robustness faster than the standardly trained module. However, we see that as the batch size increases the width of the peak increases, and the model settles at a high local learning coefficient and lower robustness.

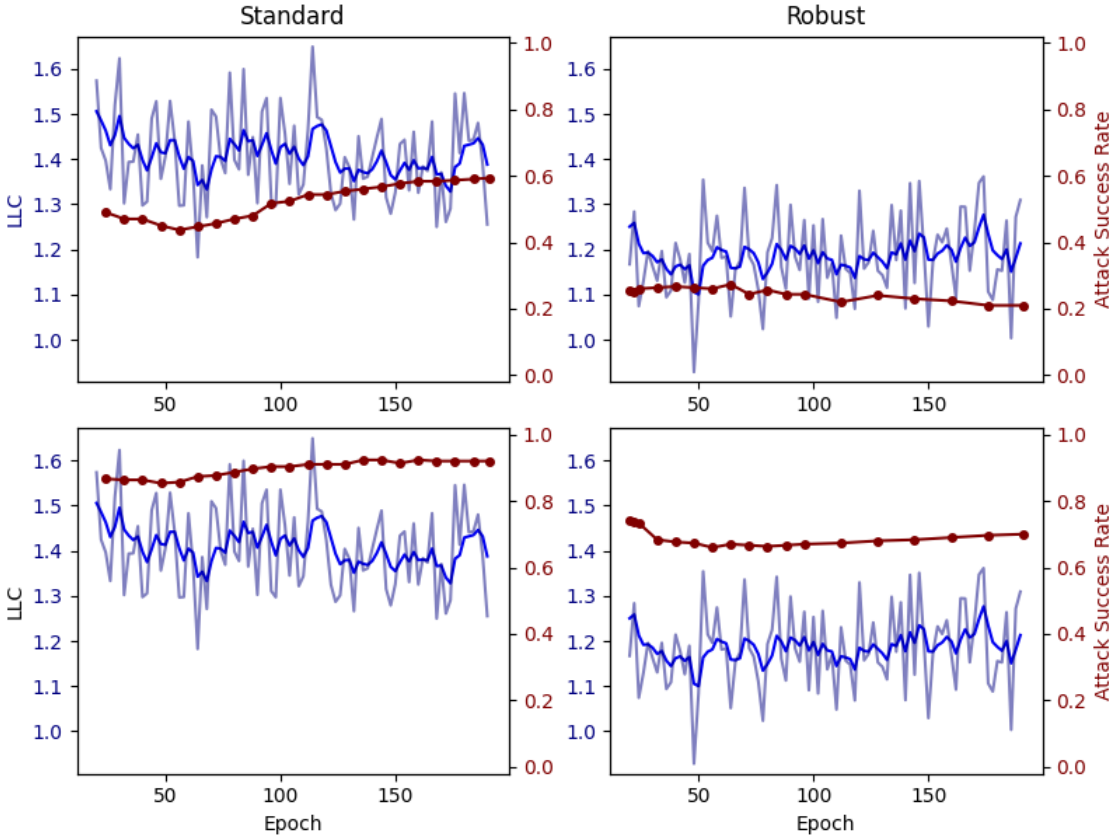
Extended Epochs

Now we explore what happens when we extend the training for many more epochs. In particular, we take the same model architecture and training parameters as above, with a batch size of 64, and we continue to train the model up to 192 epochs.

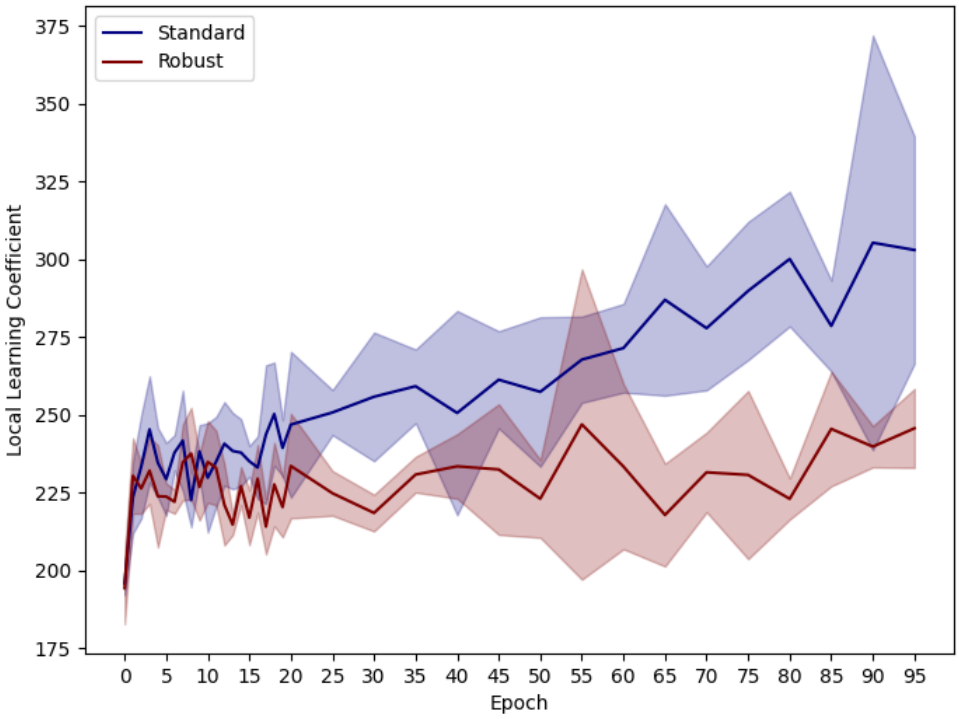


We observe that robustness seems to decrease overtime, however, there is relatively little change in the local learning coefficient.

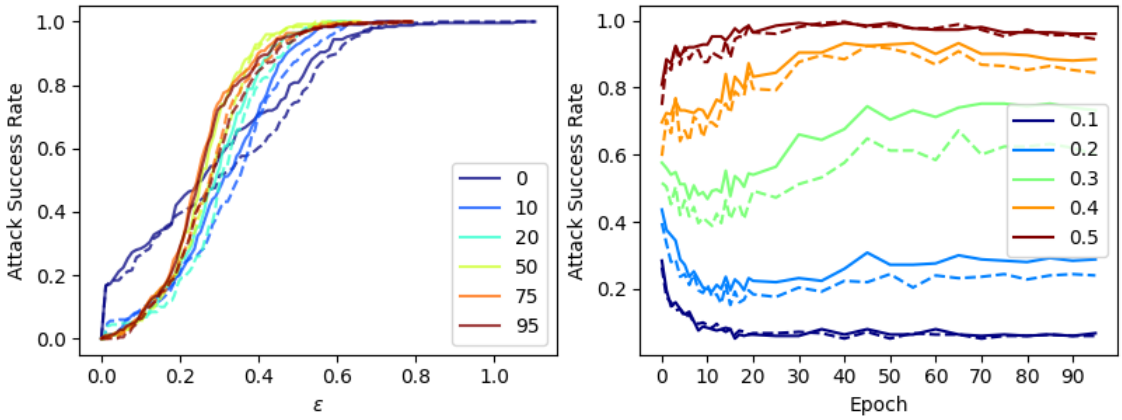
This effect is more pronounced on a smaller model.



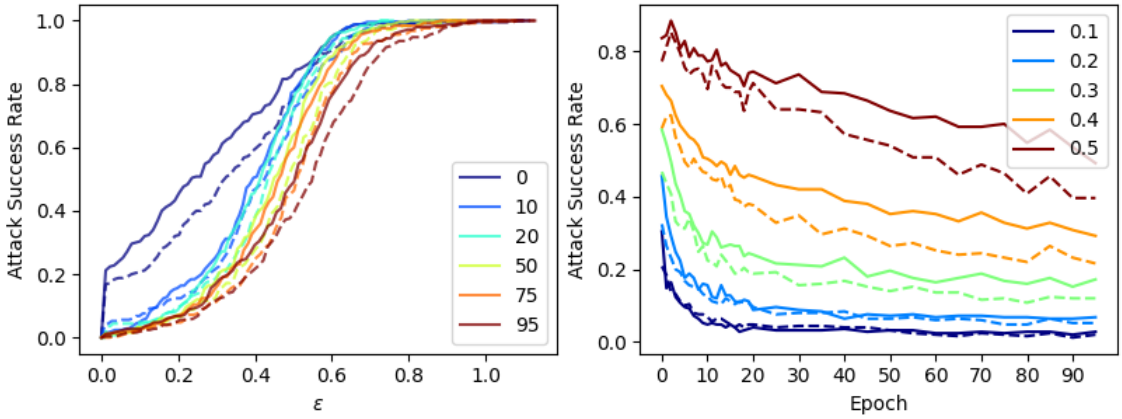
In this case, we are dealing with a low-dimensional synthetic dataset. When we repeat this investigation on the MNIST dataset we obtain the following.



We observe that the local learning coefficient increases over time for the standardly trained model, which correlates with its decrease in robustness.



On the other hand, the robustness of the robustly trained model remains fairly consistent.

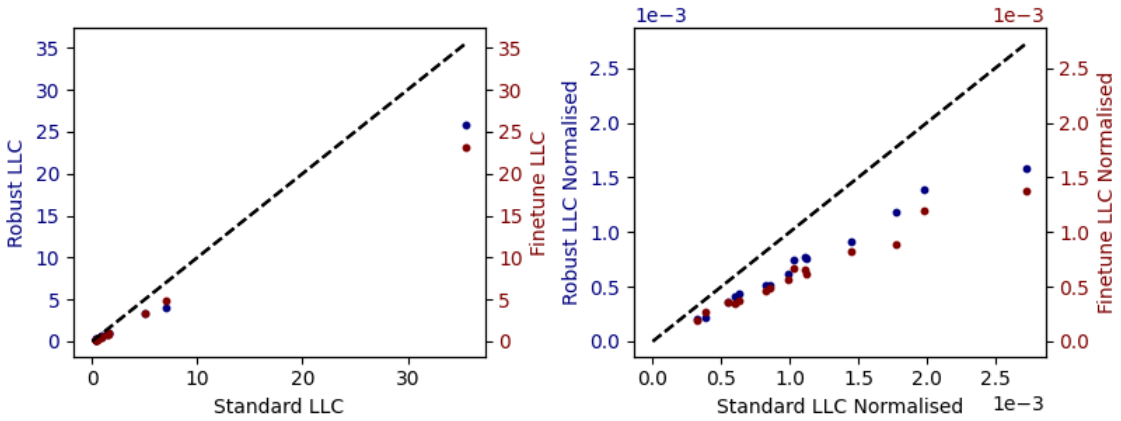


Effects of Finetuning

Here we analyse the effect that finetuning compared to robust training using the local learning coefficient.

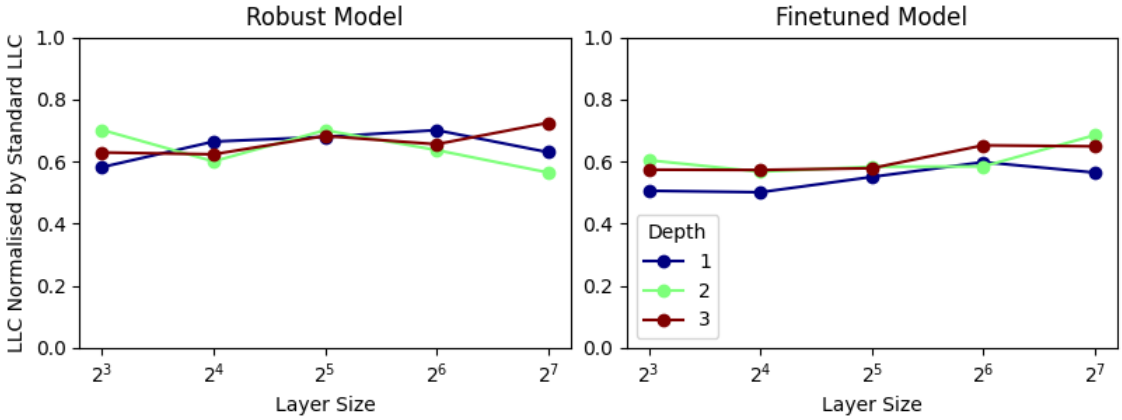
In this instance, we consider models with varying numbers of hidden layers and varying sizes of hidden layers. Standard training and robust training occur for 100 epochs, with finetuning occurring for 50 epochs.

We observe that the local learning coefficient for finetuned models are lower than those of robust models, which are in turn lower than those of standard models. The latter difference is expected from our intuition, whereas the former difference is not obvious.

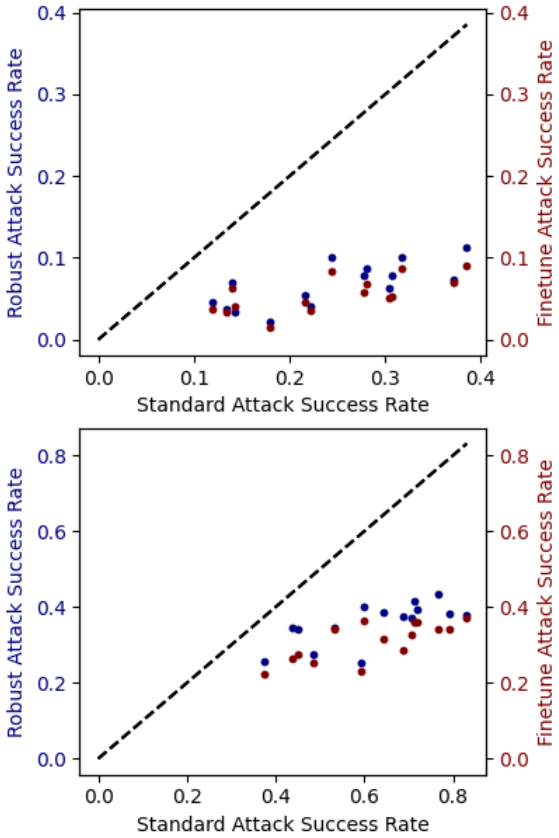


Note that we normalise the local learning coefficients by the number of parameters. One could argue that the robust model and finetune model are trained for a different number of epochs, which is what leads to the lower local learning coefficient. However, we observed previously that the local learning coefficient does stabilise during training. Supposing that this difference is not attributed to the difference in training epochs, one may argue the difference is due to standard training more effectively extracting the features of the dataset compared to the robust training. Since the robust training incentivises robust features from the beginning, it may limit the capacity of the model to sufficiently explore the landscape to determine representative features, which it can then later solidify.

In our experiments we trained models with hidden layers of size $\{8, 16, 32, 64, 128\}$, and depths $\{1, 2, 3\}$. Interestingly, we observe that with each architecture the relative reduction in local learning coefficient provided by the robust and finetuned models remains fairly consistent.



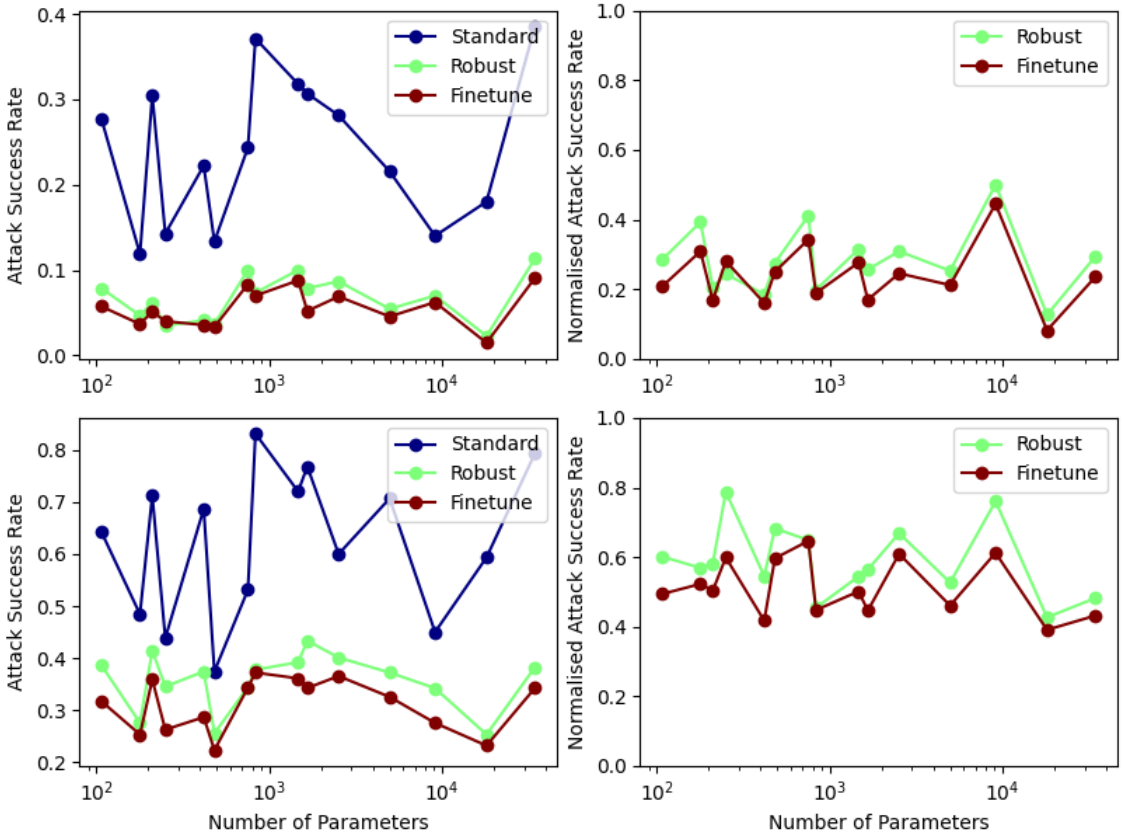
Either this is because the models are overparameterized in every case, so the changes in architectures are insignificant. Or the robustness of the model is dependent on other factors.



The above figure considers the attack success rates on the robust and finetune models against the attack success rate on the standard model. The top row sets the perturbation threshold at 2 and the second row sets it at 3. Recall that during adversarial training the models only encounter perturbations of size at most 2.

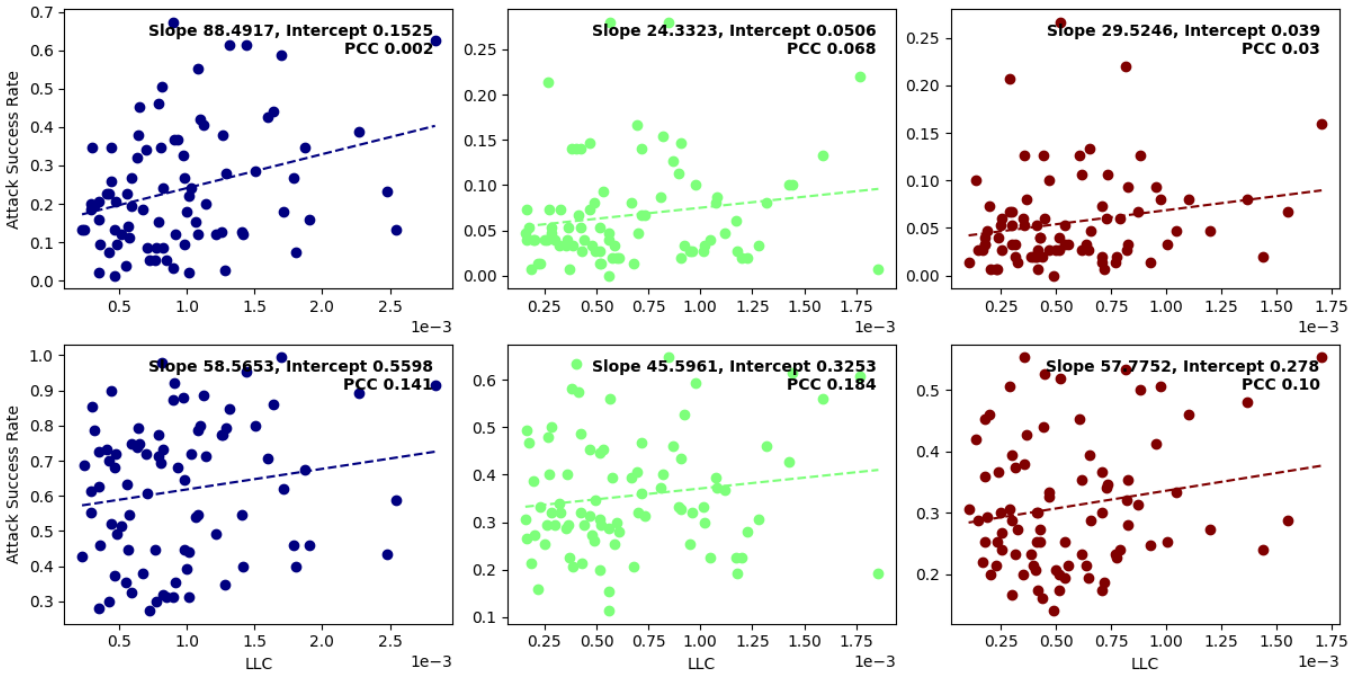
The attack success rates in all cases increases as the perturbation threshold increases, which is to be expected. Furthermore, we observe that the robust and finetune models are significantly more robust compared to the standard models. In particular, the improvement in robustness degrades as the perturbation threshold is increase, which is also to be expected. Moreover, the finetuned models are more robust compared to the robust models as expected.

As before, we relate the attack success rate to the number of parameters in the model.



Each row corresponds to the same perturbation thresholds of the previous figure. Just as we observed a consistent improvement in local learning coefficient across the architecture of the networks when we normalised it against parameter count, we observe here that the robustness improvement provided by the robust and finetuned models is consistent across parameter count. Reinforcing the connection between the local learning coefficient and robustness.

Now we directly compare the local learning coefficient, normalised by parameter count, and the attack success rate. Recall that from out intuition we expect there to be a positive correlation.

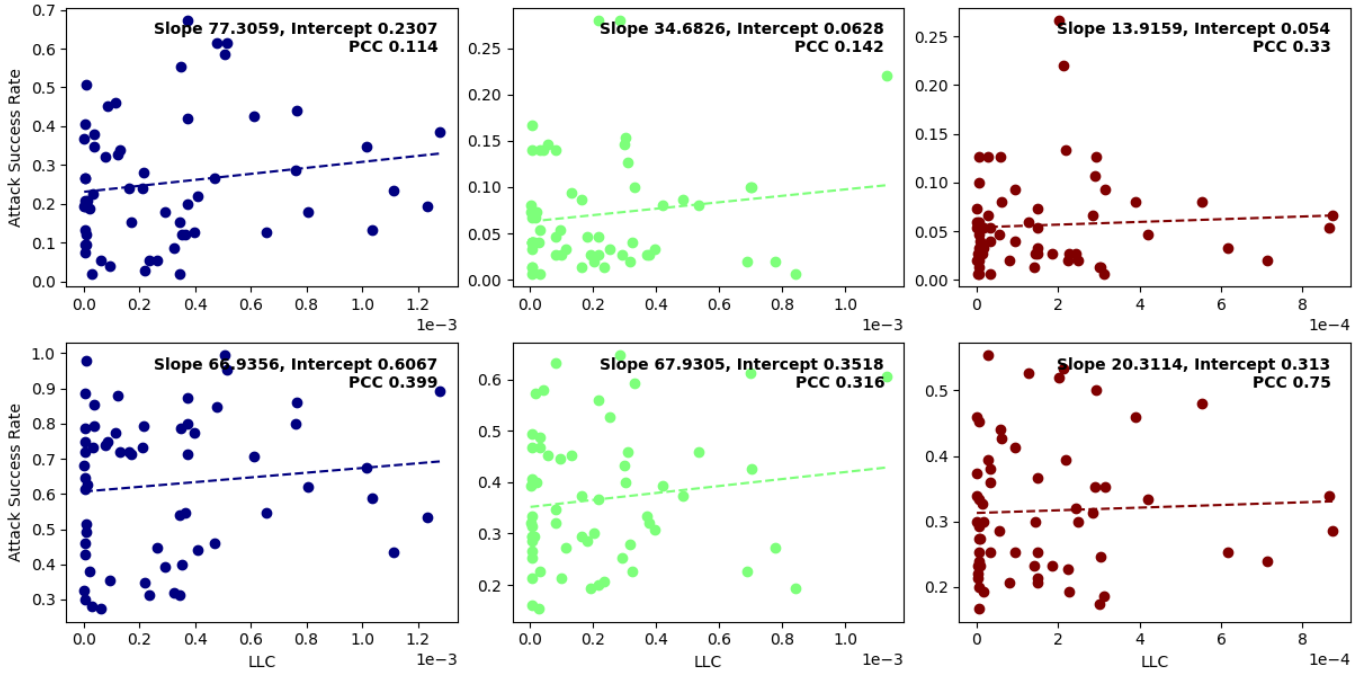


Indeed we observe that there is, in most case, a statistically significant positive correlation between the attack success rate and the local learning coefficient. What's more, these correlations tend to be more significant in the case of the standard and finetuned model.

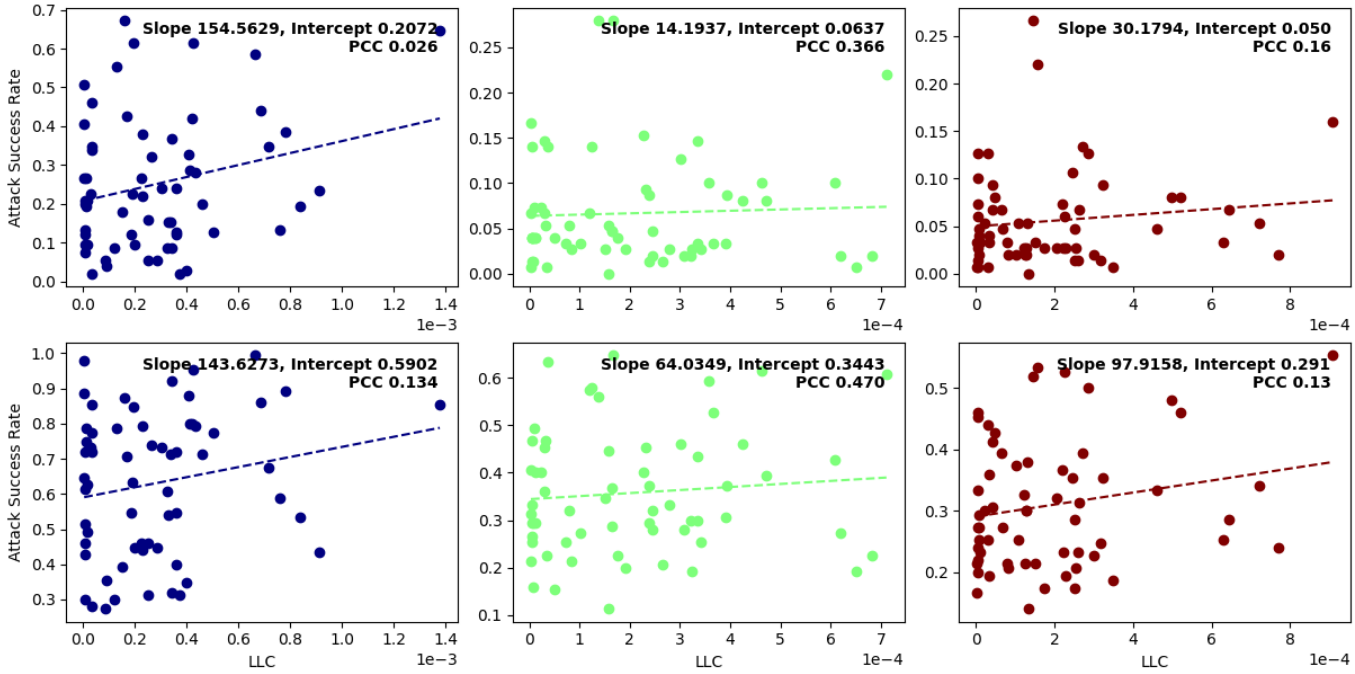
Effects on Layers

Where previous local learning coefficient estimations were conducted on the entire parameter space, here we focus on subspaces of the parameter space corresponding to different layers.

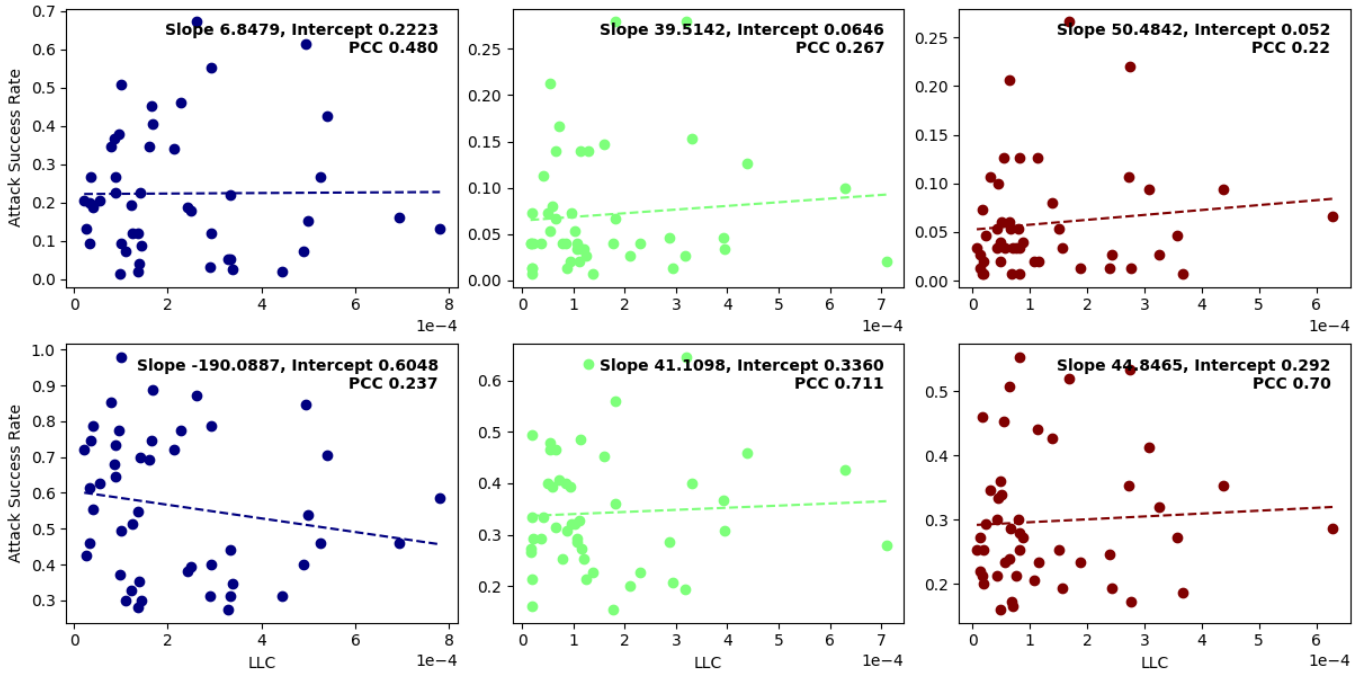
We consider the input layer, the output layer, the hidden layers, and where applicable the first of the hidden layers and the second of the hidden layers. In each case we observe that the local learning coefficient follows our outlined intuition, and the patterns observed when we computed the estimate on the entire parameter space. However, we find the that positive correlation between the local learning coefficient and attack success rate is only present in the input and output layers. In the hidden layers there is no correlation between the local learning coefficient and the attack success rate. This is perhaps because the hidden layers are becoming robust to higher-level features, which cause the local learning coefficient to reduce, however this does not show up in the attack success rate which is focused on the robustness against small perturbations. On the other hand, since the input layer directly observes the perturbations and the output layer is responsible for the model's decisions, it make sense that the correlations are observed in these layers.



Input layer.



Output layer.



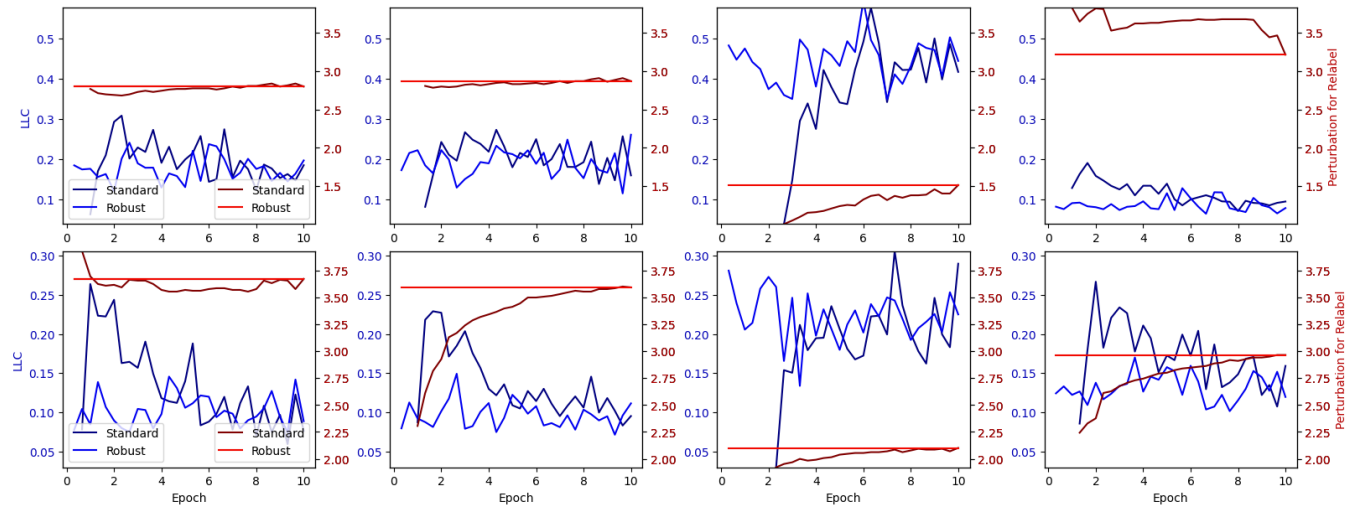
Hidden Layers

Sample-Level

To investigate questions at the sample level, we take an individual sample from a test set, sample random datapoints within a small neighbour, provide these samples with the same label as the test set sample, and then estimate the local learning coefficient using this data.

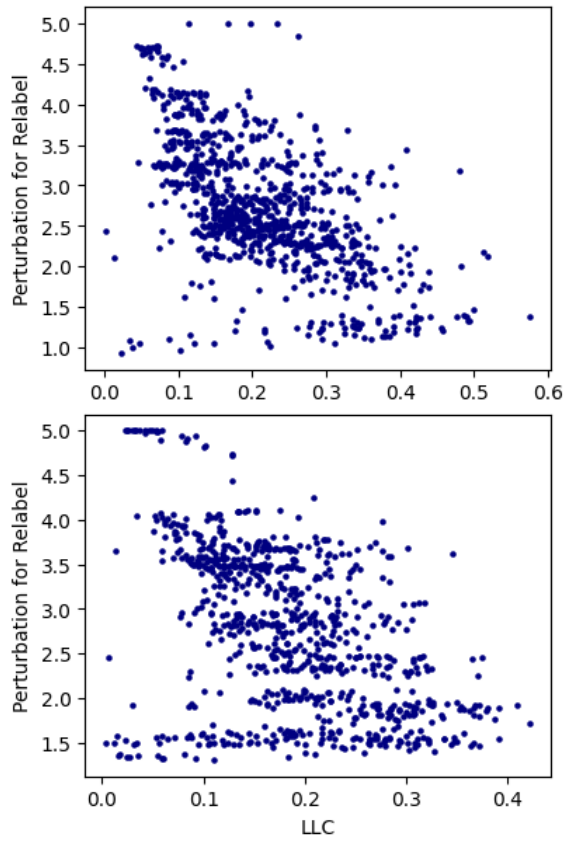
At these test samples we also investigate the perturbation necessary, under a PGD attack, to cause the model to reclassify the data point. In every case, the size of the neighbourhood from which the samples is taken is much smaller than this perturbation amplitude, justifying our step in labelling these neighbourhood samples with the same label as the test file.

In the following figure we display the local learning coefficient and perturbations for four test samples across 10 epochs of training. The top row of the figure corresponds to one model, and the bottom row to another. Each column represents a different test sample, with each row of the column also corresponding to a different test sample. Note that we have also removed the local learning coefficient estimates that are negative, as to not skew the diagrams.

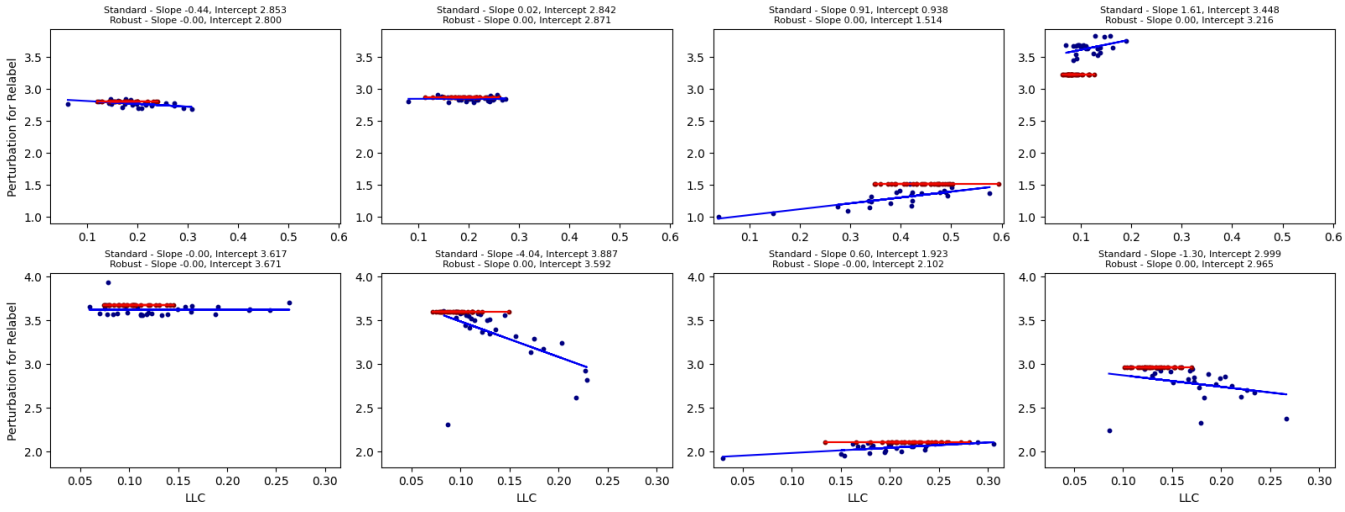


What we observe is that a higher local learning coefficient correlates with a lower perturbation to reclassify the test sample. Similarly, a lower local learning coefficient correlates with a higher perturbation to reclassify the test sample. Reinforcing the intuition that a lower local learning coefficient corresponds to a a simpler model solution to classify the test sample, meaning that the test sample is more robust to perturbations. Moreover, we observe that robustly training the model quickly saturates the robustness possible at a particular sample.

For each of the models considered above, we obtained this data for 32 separate test samples. Considering the perturbation for relabelling against the local learning coefficient for all these samples further highlights the negative correlation we just detailed.



In the figure one can observe striations in the points, highlighting the locality of this analysis. Even though the absolute value of the local learning coefficient changes, there seems to be universal trends in how the perturbation for relabelling correlates with the local learning coefficient.



Indeed we see this with the above figure, where we consider the correlations between the local learning coefficient and robustness of individual test samples. Most notably, we observe that the test samples at which this negative trend fails ,correspond to the test samples that are not-robust points, as identified by an earlier figure.

It may be difficult to utilise this analysis for adversarial sample detection, since often adversarial perturbations can be fairly close to the original sample. However, it may useful as a heuristic for identifying which samples are most susceptible to adversarial attack.

References

(1) Murfet, D. *et al.* (2023) 'Deep Learning is Singular, and That's Good', *IEEE Transactions on Neural Networks and Learning Systems*, 34(12), pp. 10473–10486. Available at: https://doi.org/10.1109/TNNLS.2022.3167409.

(2) Lau, E., Murfet, D. and Wei, S. (2023) 'Quantifying degeneracy in singular models via the learning coefficient'. arXiv. Available at: https://doi.org/10.48550/arXiv.2308.12108.

(3) Hoogland, J. *et al.* (2024) ‘The Developmental Landscape of In-Context Learning’. arXiv. Available at: https://doi.org/10.48550/arXiv.2402.02364.

(4) Stan van Wingerden, Jesse Hoogland, and George Wang (2024) ‘DevInterp’. Available at: https://github.com/timaeus-research/devinterp/tree/main (Accessed: 31 August 2024).