



Comparative Analysis of Curriculum Strategies in training Meta-Learning
Curriculum Strategies for Faster Meta-Learning

Maria Mihai¹

Supervisors: Matthijs Spaan¹, Joery de Vries¹

¹EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Maria Mihai
Final project course: CSE3000 Research Project
Thesis committee: Matthijs Spaan, Joery de Vries, Pradeep Murukannaiah

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Abstract

Meta-Learning is an emerging field where the main challenge is to develop models capable of distilling previous experiences to efficiently learn new tasks. Curriculum Learning, a group of optimization strategies, structures data in a meaningful order which aids learning. However, the extent to which curriculum strategies can optimize the performance of meta-learners remains unclear. Here we study the separate and joint effects of a model-based (ScreenerNet) and a statistics-based (Active Bias) curriculum strategy on the training of a meta-learning model (Neural Processes) which solves 1-D function regression tasks. The findings show that ScreenerNet increases in-task accuracy and speeds convergence, but decreases the generalisation performance. Active Bias achieves mixed generalisation results and significantly decreases training efficiency when trained on noisy data-sets. Combining them partially mitigates ScreenerNet’s overfitting and stabilises Active Bias’ susceptibility to noise, but further research is necessary in order to achieve consistent improvements to the baseline.

1 Introduction

A great part of the evolution of Machine Learning systems can be closely traced to inspirations drawn from the human brain. In 1993, it was proposed to extend this link by mirroring the learning strategy of humans (Elman, 1993). Few students are taught concepts in random order; instead, significant efforts and considerations are paid by their teachers to order the concepts, usually by increasing difficulty, in *curricula*. Applying a similar process in the training of ML models has been repeatedly shown to increase learning performance (Soviany et al., 2022; Wang et al., 2021).

Another emerging topic in computer science is Meta-Learning, or “learning-to-learn”. Hospedales et al define Meta-Learning as “the process of distilling the experience of multiple learning episodes ... to improve future learning performance” (Hospedales et al., 2021, p. 5149). In other words, meta-learners do not rely on dedicated human tuning in order to achieve satisfactory performance for a specific task; they can generalize insights acquired in solving previous tasks in order to more efficiently learn similar, yet new, tasks. Contemporary deep learning is often criticized for requiring large data volumes and extracting superficial patterns (Marcus, 2018). Meta-learning aims to address these issues but faces challenges such as susceptibility to label noise (Liang et al., 2022; Lu et al., 2020) and high computational costs (Hospedales et al., 2021). Optimization techniques have been applied with the purpose of addressing some of these limitations (Killamsetty et al., 2022; Mazumder et al., 2021). However, few studies comprehensively analyze the effects of Curriculum Learning techniques in the training of meta-learners (Que and Yu, 2024). This can be partly attributed to the fact that many of these techniques rely on a domain-dependent - and often human-assigned - definition of difficulty (Jiang

et al., 2015; Graves et al., 2017), which is not trivial in the case of meta-learning.

A first requirement for a Curriculum Learning strategy is wide, task-agnostic applicability, for which the subset of Self-Paced Learning methods is well-suited (Soviany et al., 2022). Within this family, the general idea is to construct curricula by ordering samples from easy to difficult according to a difficulty measure obtained from the model’s performance. A large and diverse pool of strategies exist for distilling this measure of difficulty, with earlier versions using simple performance statistics (Kumar et al., 2010; Lee and Grauman, 2011). Later, a new family of strategies emerged, which replaced these statistics with model-based approaches aimed at discovering more complex and abstract patterns (Kim and Choi, 2018; Jiang et al., 2018). In the statistics-based group, the ‘easy-to-difficult’ ordering was redefined through more nuanced techniques (Chang et al., 2017; Tang and Huang, 2019).

In this research, we will analyse near state-of-the-art representatives from both the model-based and the statistics-based approaches. Within the first group, ScreenerNet is an attached neural network which performs training on all samples, but assigns higher importance to tasks predicted to be more difficult (Kim and Choi, 2018). Within the second group, Active Bias aims to find a general solution to the target sample difficulty on which to train, using uncertainty as a measure to construct curricula at the ‘edge’ of a model’s ability (Chang et al., 2017). Our contribution is comparing these approaches in the extent to which they can improve the performance of a meta-learning model (Neural Processes) which is trained for solving 1-D function regression tasks; we additionally study the effects of combining the two approaches.

We will compare the strategies from three perspectives: noise robustness, generalization, and efficiency. In the first category, experiments will assess each strategy’s impact on model performance when trained on noisy data. In the second, we will evaluate the model’s ability to generalize to new task types. In the third, we will analyze the training behavior of each strategy.

2 Background & Related Work

This section contains a deeper description of the problem being optimized (meta-learning), a general description of the family of methods which constitute the optimization approach (curriculum strategies), and a description of the specific strategies which constitute the focus of the research (ScreenerNet and Active Bias). For these strategies, their main properties will be presented along with the motivation for their study. A more detailed description of how they were adapted and implemented can be found in section 3.

2.1 Meta-Learning & Neural Processes

In classical machine learning, models are trained on large amounts of data in order to achieve high performance on a specific problem. However, if a change occurs in the underlying assumptions of the problem, the model needs to be re-trained using an amount of time and data resources which is

comparable to the one used in the initial training. The challenge of meta-learning is to extract knowledge from previous experiences such that it can be applied in the learning of new tasks, thus reducing the learning effort (Vanschoren, 2018). Meta-Learning techniques have achieved promising results in various fields (from increasing accuracy in Language Modelling (Vinyals et al., 2016) to adapting more quickly in 2D Navigation RL tasks (Finn et al., 2017)). Because they are trained to reuse insights from past experience, they are particularly suited to cases where there are multiple, similar tasks to learn, but few examples for each (Garnelo, 2018; Dubois et al., 2020).

Neural Processes are “meta-learning algorithms for few-shot function regression” (Garnelo et al., 2018, p. 5). Their name stems from their architecture, which combines properties of Neural Networks and Gaussian Processes (stochastic processes which are fully specified by a mean function and a covariance function (Williams and Rasmussen, 1995)). The architecture of Neural Processes consists of three main components: an encoder, an aggregator, and a conditional decoder. The encoder is parameterised as a neural network; it accepts pairs $(x, y)_i$ of values from the context set, and outputs a representation $r_i = h((x, y)_i)$. The aggregator then outputs a global representation r (often computed as a mean of all representations r_i). Finally, the decoder (also parameterised as a neural network) accepts the representation r and the target input x_T and predicts a mean and variance of the predictive distribution of the target output y_T . This architecture is illustrated in Figure 1. The testing and application of Neural Processes can be described as follows: the NP model is presented with a context set for a task it has not encountered before; using only a forward pass (and no gradient updates), it adapts to the current context and uses information gained during the training process in order to make predictions for unseen samples (the target set) of the current task. The sample loss function used for the training of Neural Processes is the negative of the Evidence Lower Bound (ELBO), which represents the likelihood of the observed data (Y) given the current prediction (μ, σ) for the value of $f(X)$.

$$ELBO(f) \leq \ln(N(Y|\mu, \sigma)) \quad (1)$$

2.2 Curriculum Learning

The family of curriculum strategies is diverse in applications, methods, and levels at which they can be applied. Soviany et al propose seven categories of curriculum learning (CL) methods: vanilla CL, self-paced learning, balanced CL, self-paced CL, teacher-student CL, implicit CL, and progressive CL (Soviany et al., 2022). As noted by the authors, this categorization still presents overlaps, and is not exhaustive. However, it reflects the main trends and evolution in the field.

Vanilla CL methods - introduced by Bengio et al (Bengio et al., 2009) - use pre-defined rules to assemble fixed curricula. At the other end of the spectrum, self-paced learning methods - introduced by Kumar et al (Kumar et al., 2010) - create curricula dynamically, based only on the learning model’s performance. This approach is more suitable for optimizing meta-learning, where comprehensive heuristics

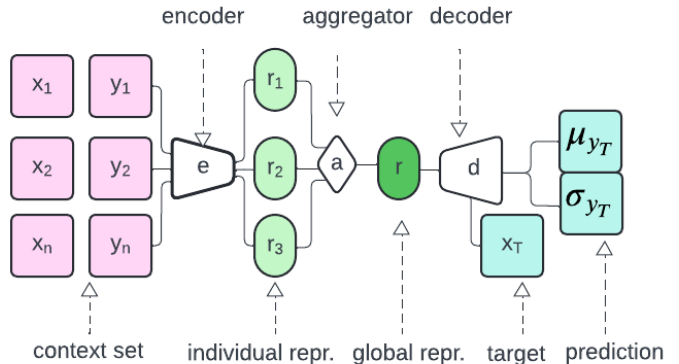


Figure 1: Architecture of Neural Processes, consisting of an encoder (e), an aggregator (a), and a decoder (d). An input consisting of a context set of x, y pairs is provided to the encoder, which outputs an individual representation to each pair of points. These representations are then aggregated into a global representation, which, along with the x coordinate of a point from the target set, is used by the decoder to make a prediction defined by a mean and variance. *Note.* Adapted from (Garnelo, 2018).

for defining task difficulty are not yet available or intuitive. The other categories of CL include variations or combinations of Vanilla CL and SPL. Relevant to the current research are Teacher-Student Curriculum Learning (CL) and Active Learning.

Under Teacher-Student strategies, learning is split into the main task (for which the *student* is being trained) and determining the optimal parameters for learning (for which the *teacher* is being trained). Therefore, Teacher-Student strategies are similar to self-paced CL in that the learning progress is used for predicting certain components of a curriculum, but for Teacher-Student systems, the prediction is more advanced and delegated to another learning system. An example of a Teacher-Student system is ScreenerNet (Kim and Choi, 2018), which is described in subsection 2.3.

Active Learning strategies place emphasis on uncertainty, rather than learning models. In other words, they note that focusing on the ‘easy’ samples is more effective in some scenarios, while focusing on the ‘difficult’ samples can yield better results in others. The proposed general solution is to take into account the uncertainty, or how consistent is the model’s performance on a given sample. An example of an Active Learning strategy is Active Bias, which is described in subsection 2.4.

2.3 ScreenerNet

ScreenerNet is an attachable neural network, trained at the same time as the main neural network, which aims to learn the significance of each learning sample (Kim and Choi, 2018). The significance score is highly influenced by the error which ScreenerNet predicts the main neural network will have on a given sample (the higher the estimated error, the higher the

weight). In the training procedure outlined by the authors, ScreenerNet does not select only certain tasks which will be passed on to the main neural network; rather, all tasks are selected, but some of them have higher weight on the loss of the main neural network. Moreover, it has been shown to increase convergence speed and accuracy for image classification and deep reinforcement learning, as well as out-perform sampling-based curriculum learning methods (with which it can be combined for further improvement). On the other hand, the authors do not discuss training conditions which include noisy datasets. In such cases, given the fact that higher error samples have higher weights, the noise robustness can potentially have lower performance.

Objective function ScreenerNet minimizes the following objective function:

$$L^{SN} = \sum_{x \in X} ((1 - w_x)^2 l_x + w_x^2 \max(M - l_x, 0)), \quad (2)$$

where X is the set of samples (or tasks), w_x is the weight assigned to a task x , l_x is the main neural network’s loss for the given task, and M is a constant used for upwards-clipping the value of the losses.

Expected effect We expect that ScreenerNet’s main improvement will be increasing the model’s accuracy for the tasks with higher predicted difficulty. However, a limitation of the current research is that we will not investigate the extent to which ScreenerNet can model difficulty. At the same time, we expect that ScreenerNet will improve the convergence speed of Neural Processes because of its significance-weighted parameter update.

2.4 Active Bias

Active Bias is a sampling strategy which emphasises uncertain samples Chang et al. (2017); it is part of the Active Learning category of curriculum strategies, and the authors describe multiple implementations, of which one (SGD-PV) is more applicable for regression tasks. The main insight is that the samples on which the model consistently gets low errors might be too easy to further improve the model, while the samples on which the model consistently gets high errors may be too difficult and thus not bring significant improvements to the model. Therefore, the prioritized samples are those for which the model sometimes gives low errors, but where performance can still be improved. This strategy is introduced in response to the observation that Self-Paced Learning (emphasising easy samples) and Hard Example Mining (emphasising difficult samples) both have demonstrated performance improvements in different situations (Shrivastava et al., 2016; Jiang et al., 2015). However, when approaching a new dataset, it is not always clear which method will be more effective. The proposed strategy, Active Bias, aims to automatically discover the target difficulty for a given scenario. Figure 2 illustrates an insight into the difference between easy, uncertain and difficult samples.

Expected effect We expect that Active Bias increases the generalisation performance, because the training is performed on samples which are expected to have stronger learning signals. However, because not all samples are presented




sample	error history		type	method
	0.05	0.08	easy	SPL
	0.4	0.15	uncertain	Active Bias
	0.77	0.74	hard	HEM

Figure 2: Simplified example of easy, uncertain and hard samples. The first and third functions both have low error variance, even though their errors significantly vary in magnitude. The first example would be sampled with higher probability in Self Paced Learning, while the third example would be emphasised in Hard Example Mining. However, the second function has larger error variance. The intuition behind Active Learning is that a model can have the largest learning model when training on such a task, and will therefore assign it higher sampling probability. *Note.* Adapted from (Chang et al., 2017).

to the model with the same frequency, we expect that the model will have lower accuracy on some sample types. As a result, there could be large differences in the loss values for different sample types. Finally, we expect that the noisy samples will be considered more ‘difficult’ and therefore be sampled with lower probability, therefore increasing the noise robustness.

3 Approach

This section contains a description of the procedures by which the three techniques (ScreenerNet, Active Bias and their combination) were applied in the context of training Neural Processes for 1-D function regression tasks. For each of these, we will describe mainly what has been added to the baseline training procedure illustrated in Figure 3.

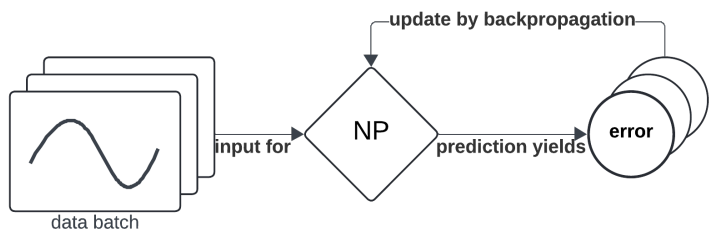


Figure 3: Simplified overview of the training procedure of Neural Processes, where their architecture is treated as a black-box model. This structure, which uses mini-batch gradient descent, is seen as a baseline against which the other learning variants will be compared.

3.1 ScreenerNet

Training The joint training procedure for ScreenerNet and Neural Processes is outlined in the following pseudo-code:

Algorithm 1 Training Neural Processes with ScreenerNet

```

1: Initialize ScreenerNet
2: Initialize NP
3: for iteration = 1, 2, ... do
4:   for Every mini-batch of tasks do
5:     for Every sample f in do
6:        $w_f \leftarrow \text{normalised}(S(f))$ 
7:        $e_f \leftarrow \text{ELBO}(f)$ 
8:        $l_f \leftarrow L(f)$  (Eq. 3)
9:     end for
10:     $e_b^{NP} \leftarrow -\frac{1}{|b|} \sum_{f \in b} (w_f \cdot e_f)$ 
11:     $e_b^{SN} \leftarrow \frac{1}{|b|} \sum_{f \in b} L^{SN}(l_f, w_f)$  (Eq. 2)
12:    UpdateNetwork(NP,  $e_b^{NP}$ )
13:    UpdateNetwork(SN,  $e_b^{SN}$ )
14:   end for
15: end for
  
```

Every training iteration consists of the sampling of batches of tasks (or functions). For each of them, ScreenerNet predicts a weight, and a loss is computed as an average of losses over a target set of points on the function. Then, the loss by which ScreenerNet is updated is its objective function applied to the current batch, and the loss by which NP is updated is the negative of the weighted mean of all losses in the current batch. This procedure is illustrated in Figure 4.

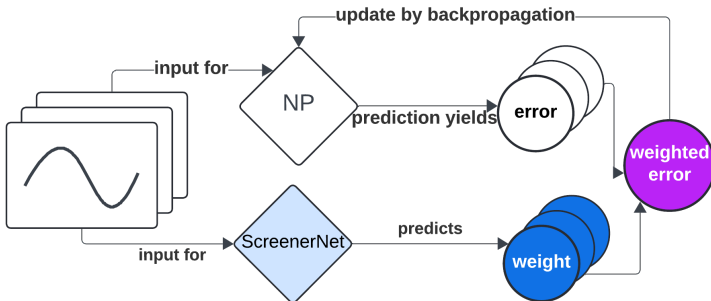


Figure 4: Overview of training Neural Processes using weights predicted by ScreenerNet. At each training step, a batch of data is sampled stochastically, and provided as input for both ScreenerNet and NP. The errors of NP are weighted using ScreenerNet’s output and averaged in order to obtain the error by which NP will be updated. For visual simplicity, the updating of ScreenerNet using its objective loss is not displayed.

Changes The original pseudo-code was slightly adapted for the specific context of the training procedure of Neural Processes. The most significant way in which the procedure was adapted is that the loss on which the NP model is trained differs from the loss which is inputted to ScreenerNet. This is because of two reasons: firstly, it was empirically seen that ScreenerNet is not able to ‘predict’ the Evidence Lower Bound, which reflects the internal KL-complexity. Secondly, the objective loss of ScreenerNet requires positive losses in order for its predicted weights to be well-defined. Therefore, after experimenting with different formulas, the follow-

ing sample loss functions has been identified:

$$L(f) = \frac{1}{|T|} \sum_{i \in T} \left(\sigma_i + \frac{\|y_i - \mu_i\|}{\sigma_i} \right), \quad (3)$$

where $T = (x, y)_i$ is the target set of the sample function x , and μ_i, σ_i are the predicted mean and variance of the model for each value y_i . The purpose of this formulation of the loss function is to penalize both low accuracy (reflected in the term $\frac{\|y_i - \mu_i\|}{\sigma_i}$) and low precision (σ_i). This formula is similar to the Gaussian log-likelihood, but it does not apply logarithm to the variance in order to ensure positive sample losses.

The other changes which were made to the original training algorithm are:

- Regularization was not applied to the ScreenerNet parameters because the regularizing term did not improve accuracy or performance;
- Weights were not scaled to the $[0, 1]$ range because this reduced the weighted loss below the initial loss, artificially lowering the learning rate. Instead, weights are normalized to sum to the batch size.
- The first 10% of epochs were used as ‘burn-in’ epochs to ensure that ScreenerNet’s initial sub-optimal performance does not affect the early stages of training while it is being improved.
- The authors found that ScreenerNet performs best when its architecture is similar to the main neural network. However, due to the deep architecture of Neural Processes, a simpler architecture with fewer layers was chosen to reduce computational efforts.

3.2 Active Bias

Sampling Distribution By substituting the predicted class with prediction loss in the sampling distribution by which the SGD-PV Active Bias method is defined, the resulting sampling distribution can be obtained:

$$P(i|(l^i)_1^c) \propto p((l^i)_1^c) + \epsilon, \text{ where} \quad (4)$$

$$p((l^i)_1^c) = \sqrt{\widehat{\text{var}}((l^i)_1^c) + \frac{\widehat{\text{var}}((l^i)_1^c)^2}{c-1}} \quad (5)$$

In Equation 4, the probability of selecting sample i , given the entire set of prediction losses of the model previously obtained on that sample $(l^i)_1^c$, is proportional to its corresponding sampling score plus a smoothing constant ϵ (which prevents low-variance samples from not being sampled again). In Equation 5, the sampling score $p((l^i)_1^c)$ is computed based on the estimated variance of the loss ($\widehat{\text{var}}((l^i)_1^c)$) and its confidence interval. The loss metric is the Evidence Lower Bound (described in subsection 2.1), and their variance is computed using a rolling-variance algorithm with numerical stability properties (Cook, 2014).

Training The training procedure is similar to that of the baseline and is divided into two phases based on the authors’ recommendation to use Active Bias after several burn-in epochs. In the first phase, tasks are sampled equally, and their training errors update stored utilities to compute running variance. In the second phase, variances continue to be updated, but tasks are sampled from a categorical distribution using probability scores from Equation 5. The authors suggest experimenting with different burn-in percentages; in the experiments, we used 50%.

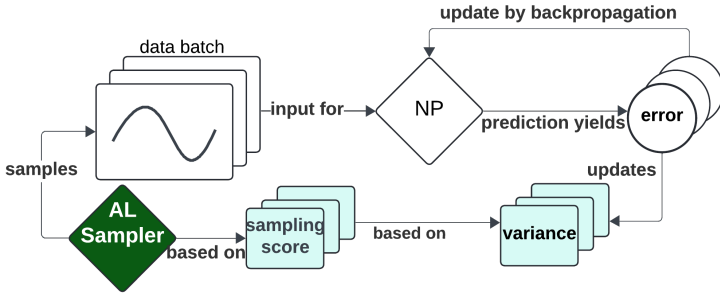


Figure 5: Overview of training Neural Processes using batches sampled according to the Active Bias distribution. At each training step, Neural Processes are trained on a batch sampled by the AL Sampler. The error history for each sample in the batch is updated, which is followed by a per-sample update in variance and sampling scores for the next iteration.

3.3 Combined approach

Changes The authors of ScreenerNet state that their model can be used to extend stochastic sampling methods for increased performance (Kim and Choi, 2018). This can be done by using the Active Bias sampling distribution in the sampling of batches in line 4 of Algorithm 1. Because ScreenerNet and Active Bias target different elements of the training procedure (the losses and the samples, respectively), no changes were applied to either of the techniques. The procedure is illustrated in Figure 6.

Expected effect We expect that combining these techniques will mitigate some of their possible negative effects. For instance, as elaborated in subsection 2.4, Active Bias could insufficiently train the model on high loss samples; ScreenerNet’s significance loss could lead to the parameters being updated by a larger step after such samples. At the same time, assuming that ScreenerNet is more susceptible to highly noisy data, Active Bias could decrease the probability that such samples are presented to the model.

4 Experimental Setup and Results

The purpose of this research is to determine whether the described strategies improve the performance of Neural Processes with regard to three aspects: Training Performance, Generalization and Robustness.

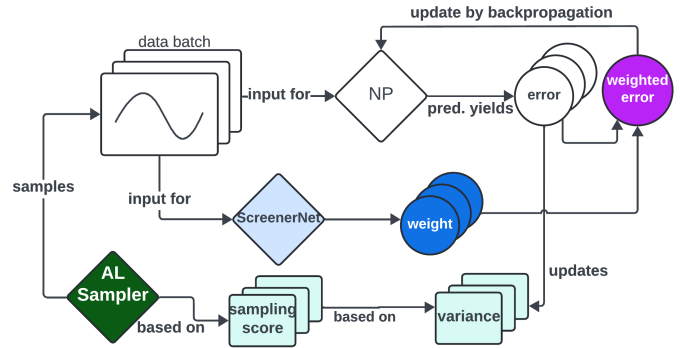


Figure 6: Complete training procedure for Neural Processes. Both Active Learning sampling and ScreenerNet error re-weighting are performed. For visual simplicity, the updating of ScreenerNet using its objective loss is not displayed.

This section contains a description, motivation and discussion of results for the experiments performed in comparing ScreenerNet, Active Bias and their combination in the training of Neural Processes. The task on which the evaluation is performed is presented in subsection 4.1; the metrics chosen for the evaluation are described in subsection 4.3, and the structure of the experiments is specified in subsection 4.2. Finally, the results, along with their interpretation, are presented in subsection 4.4.

4.1 Dataset description

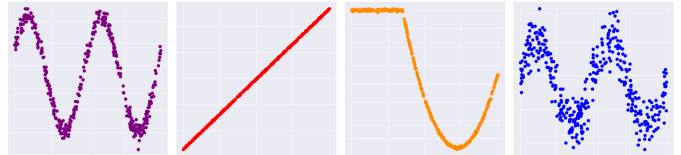


Figure 7: Examples of tasks on which the variants are trained and tested. The first three examples showcase different function types (from left to right: Sinusoidal, Slope, and clipped Polynomial of rank 2), with a low level of noise (0.01) applied. The final example depicts a Sinusoidal function with a higher level of noise (0.2) applied.

In **1-D function** regression, each task involves learning a one-dimensional function. Initially, the NP model receives a (possibly noisy) context set consisting of (x_i^C, y_i^C) points from the function. For evaluation, a target set of previously unseen points (x_i^T, y_i^T) is chosen. The NP model outputs a pair μ and σ , which represent the predicted value and the associated uncertainty for the corresponding $y_i^T = f(x_i^T | C)$ value of the query input x_i^T .

In the experiments, three family functions were considered (Sinusoidal, Slope, and second order Polynomials), as well as multiple levels of noise - examples of such functions are illustrated in Figure 7. Each function sample is represented by a context set (having 128 (x, y) pairs) and a target set (having 64 (x, y) pairs), both sampled from the function’s graph, such that for every pair, $x \in [-1, 1]$; the data-set is obtained

by randomly sampling functions within the specified families. Additional transformations applied to the function samples are: vertically shifting (adding a constant value to all values), masking (not including any samples whose x coordinate falls within a randomly-chosen sub-interval of the domain) and widening (sampling points from a larger domain, e.g. $[-3, 3]$).

4.2 Setup

Preliminary Details The experiments concern 4 models, which shall be abbreviated and referred to as NP (unaugmented Neural Processes - the baseline), NP+SN (Neural Processes weighted by ScreenerNet), NP+AB (Neural Processes with SGD-PV sampling), and NP+SN+AB (Neural Processes with ScreenerNet-weighted loss and SGD-PV sampling). These models were trained and evaluated on the same data-sets. The training and evaluation data-set distributions are specified in Appendix A. All training hyperparameters are specified in Appendix B. The implementation of the variants and experiments was performed in Python and JAX(Bradbury et al., 2018) (a library which leverages high-performance numerical computing); Pytorch was used for some data handling utilities(Paszke et al., 2017). Most of the experiments were run on the DelftBlue super-computer(Delft High Performance Computing Centre , DHPC).

Note on training set-up Initially, the models were trained on both set-ups (one consisting of different function families, and one consisting of different noise levels). However, the results indicated that the Active Bias variants trained on noisy datasets performed significantly worse. In 3 out of 5 runs, there were no improvements in loss values after 150 epochs. It can therefore be concluded that, under Active Bias sampling with a noisy dataset, **the NP+AB model did not evolve past the initial state, and therefore did not demonstrate sufficient improvements to be comparable to the NP and NP+SN variants.** For this reason, the quantitative experiments were performed on the variants trained under the set-up with different function families. The exact values for both set-ups are presented in Appendix B.

Structure The evaluation of the four variants contains a quantitative and a qualitative set of experiments. The qualitative experiments are targeted at the training efficiency of the 4 variants. Each quantitative experiment is performed by evaluating all variants on a single function type (this is in contrast to their training, which has been performed with three different function types). Here, a function type is used to refer to a fixed distribution consisting of function family along with its defining parameters, noise levels, and (optionally) parameters such as shift values or gap interval. In selecting the functions for the quantitative experiments, absolute differences between hyper-parameter values were taken into consideration as an informal measure of distance. We loosely refer to Setlur et al’s categorisation of In-Distribution (ID) and Out-Of-Distribution (OOD) evaluation settings, although most of their work concerns meta-learners for classification tasks (Setlur et al., 2021). To this end, we split the quantitative experiments into two categories: ID (evaluating on identical and very similar function types as in the training

distribution), and OOD (evaluating on function types with a higher degree of changes applied in the test function type parameters).

Hypotheses The experiments are set-up to reflect the following claims, which are motivated in the Expected effect paragraphs from subsection 2.3 and subsection 2.4.

1. ScreenerNet increases ID accuracy, achieving lower losses on task types seen during training.
2. Active Bias increases generalisation performance, achieving lower error than the baseline on test functions which are highly different from the training distribution.
3. Combining ScreenerNet with Active Bias achieves higher generalisation performance than ScreenerNet.
4. Combining ScreenerNet with Active Bias leads to lower loss variance than Active Bias.

4.3 Metrics

Various metrics can objectively assess the accuracy of a model in a regression context. When selecting evaluation metrics for our experiments, we considered the following aspects. First, we chose a widely used metric, Mean Squared Error, to ensure consistency with other experiments. Second, we selected a metric that incorporates uncertainty: Negative Log Likelihood. The exact formulas used to compute these metrics are specified below.

$$MSE(y_T, \mu_T, \sigma_T) = \frac{1}{|T|} \sum_{i \in T} (y_i - \mu_i)^2 \quad (6)$$

$$NLL(y_T, \mu_T, \sigma_T) = \sum_{i \in T} \log(\max(\sigma_i^2, \epsilon)) + \frac{(y_i - \mu_i)^2}{\max(\sigma_i^2, \epsilon)} \quad (7)$$

Under the first loss function (Equation 6), only inaccuracy ($(y_i - \mu_i)^2$ is penalised. Under the second loss function (Equation 7), both inaccurate predicted means ($\frac{(y_i - \mu_i)^2}{\max(\sigma_i^2, \epsilon)}$) and imprecise variances ($\log(\max(\sigma_i^2, \epsilon))$) are penalised (here, the ϵ constant ensures numerical stability in the case of small variances, and it is set at 10^{-6}). These loss functions will be used to assess the predictions of each variant against the evaluation tasks described in the next section.

4.4 Results

Qualitative: Convergence Figure 8 depicts the training loss evolution for the four variants trained in the set-up with different function families. It can be seen that NP+SN converges around the 50th epoch and there is less variance between the evolutions of the 5 runs. The NP+AB model stabilizes approximately around the 75th epoch. However, when trained on the noisy set-up, the NP+AB variant did not show any improvements in the evaluation results after 150 epochs for 3 out of the 5 runs. This phenomenon was also observed for the NP+SN+AB, but for 1 out of the 5 runs. The mean loss values and 95% confidence intervals are represented in Figure 9. After an attempt to identify the runs which had no evolution and train them for an additional 50 epochs, no

improvement was seen. Because of time constraints, it was decided to not run the evolution for more epochs. Future research could investigate the training behaviour of Active Bias under noisy conditions further, as well as perform the quantitative comparisons.

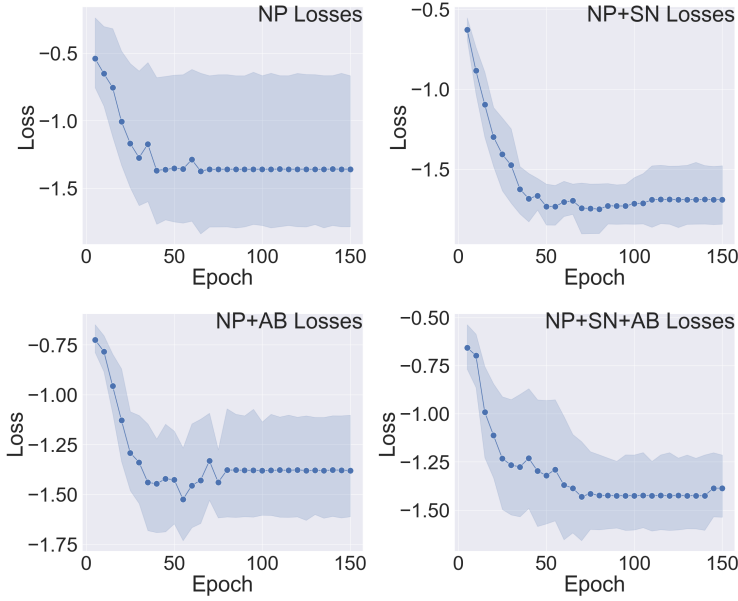


Figure 8: Evolution of NLL loss evaluated during the training process of the four variants on the clean data-set, represented as mean value and 95% confidence interval. The losses were evaluated after every fifth epoch.

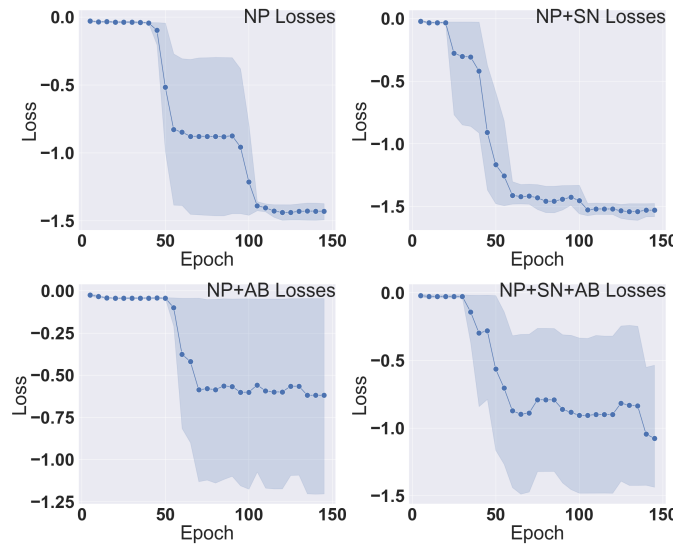


Figure 9: Evolution of NLL loss evaluated during the training process of the four variants on the noisy data-set, represented as mean value and 95% confidence interval. The losses were evaluated after every fifth epoch.

Quantitative: In-Distribution Figure 10 shows RMSE and NLL values for the functions which the variants trained

on. Although the differences in precision (RMSE loss) are negligible, ScreenerNet appears to have a slight improvement for the NLL loss values for the polynomial and slope functions, **confirming Claim 1**. The NP+AB variant achieves similar results to the baseline, for the sinus and slope functions, but slightly higher losses with a large outlier for the polynomial functions. This could be attributed to preferential sampling. The NP+SN+AB variant does not present improvements to the either of the other variants.

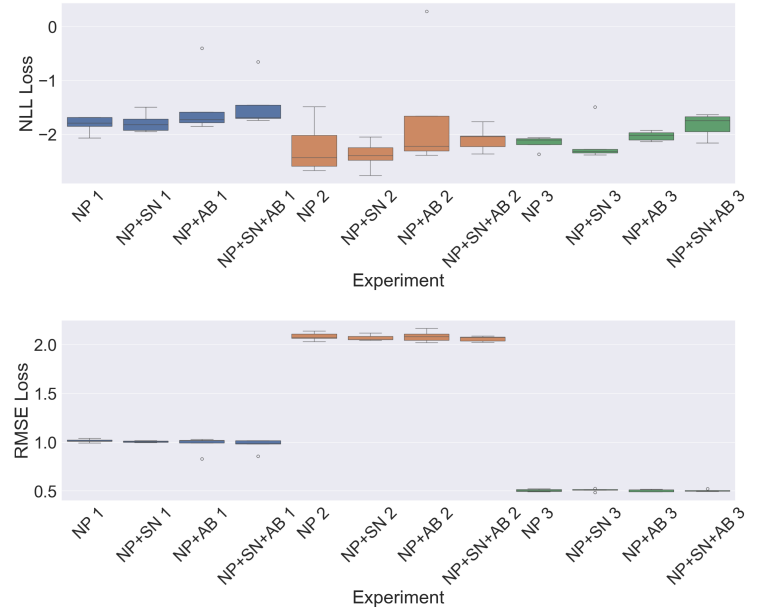


Figure 10: Box-plots of NLL and RMSE losses aggregated over 5 distinct runs. The box-plots are grouped according to experiments (the first four plots, in blue, are results for the sinusoidal functions, the next four plots, in orange, are results for the polynomial functions, and the final four plots, in green, are result for the slope functions).

Quantitative: Out-of-Distribution Figure 11 shows experiment results for different but similar functions to those seen during training (sinusoidal functions with different amplitudes and periods - the exact parameters are specified in Appendix A). ScreenerNet has worse performance than the baseline for most of the experiments, while Active Bias and the combined approach have similar or slightly better losses. Figure 12 depicts the NLL and RMSE losses for 4 categories of experiments, where the same type of sinusoidal function is inputted to the following transformations: large noise volumes, vertical shift, masking, and larger domain than in the training set. The first observation is that the baseline achieves the best results in most of the experiments. The second observation is that, for some of the experiments, Active Bias and the combined method have outliers with extremely large losses. **Claim 2 is thus neither accepted nor rejected**, as NP+AB achieves mixed generalisation results when compared to the baseline. The third observation is that the combined method achieves slightly lower NLL and RMSE losses for the generalisation experiments, but this result is off-set by

the presence of high-loss outliers. Therefore, **the third claim is neither accepted nor rejected**. Finally, by analysing both ID and OOD experiments, we can observe that the combined approach has lower loss variance than Active Bias in nearly all experiments, thus **confirming Claim 4**.

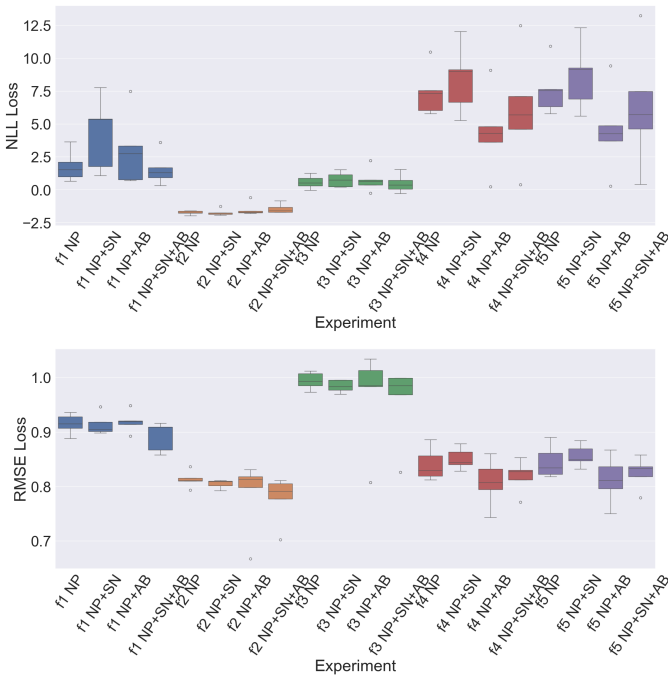


Figure 11: Box-plots of NLL losses aggregated over 5 distinct runs. The box-plots are grouped according to the 5 different experiments on function types which are similar to those on which the training was performed.

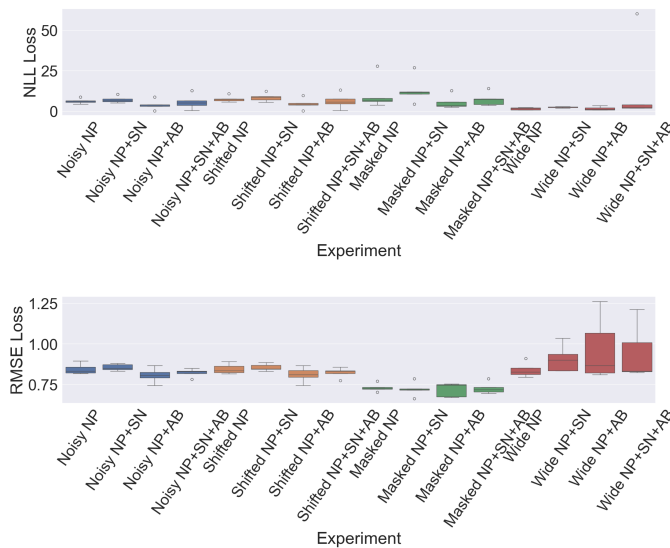


Figure 12: Box-plots of NLL and RMSE losses aggregated over 5 distinct runs. The box-plots are grouped according to 4 experiment types, corresponding to the following function types (highly noisy, vertically shifted, masked and wider domain of definition).

5 Conclusions and Future Work

The purpose of this research has been to investigate the extent to which two Curriculum Strategies can improve the efficiency, robustness and generalisation of training Meta-Learning system. The chosen strategies were ScreenerNet (Kim and Choi, 2018) and Active Bias (Chang et al., 2017), and they were assessed in the context of training Neural Processes (Garnelo et al., 2018) for 1-D function regression.

ScreenerNet is a model-based approach to automatic curriculum detection. Our hypothesis was that it would increase accuracy on data-sets with the same distribution as the training set, and that it would accelerate convergence. The findings confirmed this hypothesis, although the improvements are limited. However, ScreenerNet appears to decrease the accuracy on out-of-distribution tasks, presenting signs of overfitting. This can be partially attributed to an inherent trade-off between architecture complexity and training efficiency: increasing the complexity of ScreenerNet’s architecture can also improve the accuracy of the weights it assigns, but requires more training epochs.

Active Bias, a statistics-based approach to automatic curriculum detection, was hypothesized to improve noise robustness and generalization. Results showed mixed accuracy improvements for out-of-distribution tasks, with no consistent pattern being identified. We were unable to either accept or reject the noise robustness hypothesis, as the model failed to evolve past the initial state after 200 epochs in 3 out of the 5 training runs under a noisy training set configuration. Future research could investigate the training behaviour with different degrees of noise, and whether a trade-off between noise robustness and training efficiency is necessary when applying Active Bias.

Finally, we analysed the combined effects on training of ScreenerNet and Active Bias. The combined approach partially mitigates ScreenerNet’s tendency to overfit on the difficult tasks, achieving better results on out-of-distribution tasks, and partially mitigates Active Bias’ susceptibility to noise, by achieving more consistent behaviour when trained on noisy data. Although the current implementation of this approach did not show reliable or consistent improvements to the baseline performance, in the future an ablation study could investigate different ways in which the two strategies can be applied at the same time in order to mutually mitigate some of their shortcomings and amplify their benefits.

6 Responsible Research

A very important part of conducting research is reflecting on its potential impact and actively using the result of these reflections to shape and drive the research. This section will schematically describe the research process under three perspectives: Integrity, Reproducibility and Ethics. Reproducibility will be presented with regard to the FAIR principles of data-management: Findable, Accessible, Interoperable, and Reusable (Wilkinson et al., 2016).

Findable This principle states that data and meta-data should be assigned unique and persistent identifiers. In our case, data-sets are not persisted, but generated at every run. However, we provide the implementation of the data generation procedure, and clearly state which hyper-parameters were used to generate each training and testing data-set. Moreover, for each training run, we store the final parameters in log-files which are clearly named (containing method name, training set-up and the run identifier).

Accessible This principle states that the authentication and authorisation processes for retrieving data should be clear. In our case, the research is openly accessible at repo-link.

Interoperable This principle states that the data should be compatible with broadly-used languages for knowledge representation. In our case, data is persisted using Pickle, a Python object serialisation module ¹. Although the data-sets are not persisted, the loss and model parameters are JAX and Flax objects, respectively.

Reusable The data-sets are reusable because the data-set generation procedure is implemented at <https://github.com/maria-mihai/screenernet-activebias>. Furthermore, the experiments are reusable because we store the final optimal parameters for each variant and each of the 5 runs.

Integrity While researchers may have some bias towards their proposed solutions, it is crucial that this does not affect the objectivity of their findings. The analysis presented here does not include corrupted or altered results. Comparisons between different variants were conducted under the same conditions, thoroughly described in the report. Additionally, experiments were repeated with multiple random seeds to minimize the influence of random initialization. However, the research context imposed some limitations. Due to time constraints, each model was run only five times; ideally, this number would be higher. Limited time and computational resources also restricted the optimization of model hyperparameters. More extensive experimentation with different architectures, loss functions, and burn-in epochs would likely impact the final results, but this was not feasible within the project’s timeline.

Ethical considerations At a low-level, the current body of research does not propose a new technique, but rather adapt existing optimization strategies to augment an already existing model. Therefore, one could claim that it does not introduce a new system which could have direct impact on the external world. A more indirect consideration, however, is that

this research constitutes a contribution to the field of meta-learning. This is worth reflecting on because meta-learning enables foundation models. As presented by Bommasani et al, foundation models are defined by two properties: emergence - they derive the ability to satisfactorily solve tasks for which they were not explicitly trained; and homogenization - they can be applied in a wide range of scenarios, which leverages previously inaccessible tasks, but also leads to single points of failure (Bommasani et al., 2022). We acknowledge that emergence and homogenization were indirect goals of this current research, and recognise the necessity of research into AI safety principles.

¹<https://docs.python.org/3/library/pickle.html>

References

- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., ..., and Liang, P. (2022). On the opportunities and risks of foundation models.
- Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.
- Chang, H.-S., Learned-Miller, E., and McCallum, A. (2017). Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30.
- Cook, J. D. (2014). Accurately computing running variance. https://www.johndcook.com/blog/standard_deviation/. Accessed: 2024-06-03.
- Delft High Performance Computing Centre (DHPC) (2024). DelftBlue Supercomputer (Phase 2). <https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2>.
- Dubois, Y., Gordon, J., and Foong, A. Y. (2020). Neural process family. <http://yanndubs.github.io/Neural-Process-Family/>.
- Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.
- Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.
- Garnelo, M. (2018). Meta-learning and neural processes. Presentation held as part of the DeepMind ELLIS UCL CSML Seminar Series.
- Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. (2018). Neural processes. *arXiv preprint arXiv:1807.01622*.
- Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. Pmlr.
- Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2021). Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169.
- Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. (2015). Self-paced curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.
- Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. (2018). Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR.
- Killamsetty, K., Li, C., Zhao, C., Chen, F., and Iyer, R. (2022). A nested bi-level optimization framework for robust few shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7176–7184.
- Kim, T.-H. and Choi, J. (2018). Screenernet: Learning self-paced curriculum for deep neural networks. *arXiv preprint arXiv:1801.00904*.
- Kumar, M., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23.
- Lee, Y. J. and Grauman, K. (2011). Learning the easy things first: Self-paced visual category discovery. In *CVPR 2011*, pages 1721–1728. IEEE.
- Liang, K. J., Rangrej, S. B., Petrovic, V., and Hassner, T. (2022). Few-shot learning with noisy labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9089–9098.
- Lu, J., Jin, S., Liang, J., and Zhang, C. (2020). Robust few-shot learning for user-provided data. *IEEE transactions on neural networks and learning systems*, 32(4):1433–1447.
- Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.
- Mazumder, P., Singh, P., and Namboodiri, V. P. (2021). Rnnp: A robust few-shot learning approach. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2664–2673.
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.
- Que, X. and Yu, Q. (2024). Dual-level curriculum meta-learning for noisy few-shot learning tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14740–14748.
- Setlur, A., Li, O., and Smith, V. (2021). Two sides of meta-learning evaluation: In vs. out of distribution. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3770–3783. Curran Associates, Inc.
- Shrivastava, A., Gupta, A., and Girshick, R. (2016). Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769.
- Soviany, P., Ionescu, R. T., Rota, P., and Sebe, N. (2022). Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565.
- Tang, Y.-P. and Huang, S.-J. (2019). Self-paced active learning: Query the right thing at the right time. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5117–5124.

- Vanschoren, J. (2018). Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*.
- Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. *Advances in neural information processing systems*, 29.
- Wang, X., Chen, Y., and Zhu, W. (2021). A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.
- Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., et al. (2016). The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9.
- Williams, C. and Rasmussen, C. (1995). Gaussian processes for regression. *Advances in neural information processing systems*, 8.

A Training and Evaluation Set-ups

Table 1: Training and Evaluation set-up 1 (Figure 8, Figure 10)

Experiment number	Function type	Noise level
1	sinus, amplitude=0.5, period=2.0	0.01
2	polynomial, second degree	0.01
3	slope	0.01

Table 2: Training set-up 2 (Figure 9)

Experiment number	Function type	Noise level
1	sinus, amplitude=0.5, period=2.0	0.0
2	sinus, amplitude=0.5, period=2.0	0.1
3	sinus, amplitude=0.5, period=2.0	0.2

Table 3: Evaluation table used in Figure 11

Experiment number	Function type	Noise level
1	sinus, amplitude=1.0, period=2.0	0.15
2	sinus, amplitude=0.75, period=1.0	0.05
3	sinus, amplitude=0.5, period=1.2	0.25
4	sinus, amplitude=1.0, period=0.75	0.05
5	sinus, amplitude=1.0, period=0.75, vertically shifted by 1.0	0.0

Table 4: Evaluation set-up used in Figure 12

Experiment number	Function type	Noise level
1	sinus, amplitude=1.0, period=0.75	0.45
2	sinus, amplitude=1.0, period=0.75, shifted by 3.0	0.1
3	sinus, amplitude=1.0, period=0.75, masked	0.1
4	sinus, amplitude=1.0, period=0.75, domain=(-3, 3)	0.1

B Hyperparameters

Hyper-parameter	Value	Comments
Batch size	128	
Context set size	64	
Target set size	32	
Number of epochs	150	
Dataset size	12800	
ScreeenerNet burn-in epochs	15	Number of epochs for which ScreeenerNet was trained without its output being applied in training NP.
Active Bias burn-in epochs	75	Number of epochs for which Active Bias was trained without its output being applied in training NP.
M	1.5	ScreeenerNet clipping constant
Evaluation set size	1280	
Evaluation context set size	512	
Evaluation target set size	256	