# Application of Self-Paced learning for noisy meta-learning

**Árpád Aszalós** [1]

**Supervisor(s): Matthijs Spaan**[1], **Joery de Vries**[1]

[1]**EEMCS, 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: Árpád Aszalós
Final project course: CSE3000 Research Project
Thesis committee: Matthijs Spaan, Joery de Vries, Pradeep Murukannaiah

## Abstract

Meta-learning is an important emerging paradigm in machine learning, aimed at improving data-efficiency and generalization performance across learning tasks. Challenges caused by noisy data has been extensively researched in traditional learning settings. However, its impact in the context of meta-learning, especially concerning label noise in meta-training, remains under-explored. Curriculum Learning (CL), is an approach where training data is ordered from easy to complex, and models learn from easier to harder samples. A type of CL , Self-Paced learning (SPL) offers adaptive data curriculum, where ordering is based on per sample model performance during training. Self-Paced Learning (SPL) has proven effective in enhancing model robustness and convergence under noisy data scenarios. However, its application in meta-learning under these conditions remains limited. In this paper, we use a Neural Process model on 1D sinusoidal function regression tasks, with different ratio of clean / noisy training data scenarios to empirically observe the same benefits SPL can potentially offer for noisy meta-learning. In line with findings in traditional learning settings, SPL improved overall training convergence, also lead to increase in generalization thus noise robustness. Furthermore SPL lead the model to be more robust to increasing scale of noise for tasks within the training data distribution.


## 1 Introduction

Approaches in machine learning and artificial intelligence , in majority, often drawn inspiration from cognitive science and human-like biological and conceptual principles. With the main aim, to leverage what allows humans and animals to be versatile and efficient at carrying out , translating prior knowledge to, and learning new tasks (J. X. Wang, 2020). Current deep learning approaches, although achieve excellent results in lieu of large amounts of data, have room for improvement in terms of data efficiency and generalization capabilities. Emerging approaches, such as " meta-learning " aims to solve and alleviate some of these problems (Hospedales et al., 2021). In meta-learning, the goal is to learn representations across individual tasks (eg. a supervised learning task), in a way that they are transferable to learning or solving other tasks (Vanschoren, 2018).

Noisy data presents a significant challenge in machine learning, as it concerns real-world datasets and therefore real-world applicability of models. Substantial amount of research has been done in training models to be noise robust with varying techniques (Song et al., 2023) in traditional learning settings. Noise robustness in meta-learning, however, is yet to receive the same level of attention. The applicability of meta-learners to real-world use cases emphasizes the issue of noise robustness, as real-world data comprising meta-datasets would contain the same amount of noise as typical datasets, leading to meta-learners over-fitting on the noise, degrading performance.

In addition, majority of research considering noisy meta-learning has been in connection with managing label noise in the meta-test set through various methods, like adaptive support set task scheduling (Yao et al., 2021), or attentive support set profiling (Lu et al., 2021). As also mentioned in Galjaard (2023), only Chen et al. (2022) has looked at implications of noisy labeled meta-training , however their method is specific to Reptile (Nichol et al., 2018).

A long studied approach that has been previously utilized as a solution to tackling problems with noisy data is Curriculum Learning (CL). The work of Bengio et al. (2009), considered seminal in Curriculum Learning (CL), introduced the notion of curricula, being the introduction of examples from easy to hard throughout training, and saw improvements in performance on language modelling and shape classification tasks. Since then, Curriculum Learning has been applied and extended in several domains, eg. supervised classification, object detection, neural machine translation (Soviany et al., 2022; X. Wang et al., 2021) with empirical evidence for improved performance in particular data scenarios. Curriculum strategies generally consist of a difficulty measure and a scheduler (X. Wang et al., 2021), which together determine the order and timing of ordered data introduction for training.

Earlier work has seen challenges by relying on expert defined heuristics for curricula ordering and lacking generalization capability across domains and architectures. Later work includes different automatic approaches, and difficulty measures based on model performance to make the CL more transferable across different task and models. One such approach is Self-Paced learning (SPL), where the difficulty measure is determined by the model's performance on a per sample basis (Kumar et al., 2010). However, it has also been shown that curricula is highly dependent on certain training data conditions and environments. Mainly showing empirical improvements in noisy, imbalanced data as well as restricted learning resource scenarios (Wu et al., 2020). Self-paced learning has further been shown to be mostly applicable to the same scenarios, also with theoretical arguments for its validity and usefulness (Gong et al., 2015; Meng & Zhao, 2015).

Although meta-learning and curriculum learning saw some research on their joint application (Shu et al., 2019; Sun et al., 2019) and SPL have also been studied in a meta-learning context (Zhang et al., 2022) , there remains a gap in understanding of how different curricula could apply to noisy meta-learning scenarios. Recent work has further highlighted this gap, including X. Wang et al. (2021) for CL in general, and Zhang et al. (2022) for meta-learning and SPL specifically. Despite these related works, applicability of Self-Paced learning for the noisy data setting is however, yet still under-explored. Further empirical investigation is necessary to determine how SPL could be leveraged to improve meta-learning in noisy scenarios, and understand potential nature of improvements that SPL could offer for subsequent performance.

Given the demonstrated benefits of Self-Paced Learning (SPL) in traditional noisy learning settings, this paper aims

to answer: **How can Self-Paced learning aid meta-learning in presence of noisy training data**. The research seeks to address the following questions:

- How does SPL affect the meta-training trajectory in noisy training environments?
- How does incorporating SPL influence generalization performance for within and out-of training task distributions, under clean and noisy conditions, considering prior noisy training environments?

## 2 Background

### 2.1 Meta-Learning

Meta-learning is a paradigm in machine learning, that aims to allow training of models, that are capable of adapting quickly to unseen, or less seen examples of data. As opposed to normal learning, where the aim is to create a model that does well on a given machine learning task, in meta-learning, the objective is trying to learn across sets of tasks, by leveraging "meta-features" or "meta-representations" in the data that can be used to perform better across new tasks more quickly and with minimal data (Vanschoren, 2018).

To formalize this for a supervised learning problem, in a traditional learning setting the aim is to learn a predictor function $f(x)$ based on our dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, predicting the accompanying target and label pairs $(x_T, y_T)$ by $f(x_T) = y_T$ , so the learning algorithm $A_t$ is $A_t : D \rightarrow f(x)$ . We refer to a learning task $\mathcal{D}_{task}$ of learning to predict new data based on the our dataset in the form of $\mathcal{D}_{task}$ = ( $\mathcal{D}$ , $x_T, y_T$). In a meta-learning setup, training is carried out over a dataset of tasks (meta-dataset) $M_t = \{D_{task}^{(i)}\}_{i=1}^{N_{task}}$ (Dubois et al., 2020) . The objective compared to traditional supervised learning, then shifts from returning a predictor function solely based on $x_T$, to learning a function that returns predictions $y_T$ based on not only the target inputs $x_T$ , but also leveraging the $\mathcal{D}$ of that task, named context set. The meta-learning algorithm $A_m$ can be formulated as a mapping $A_m : M_t \rightarrow (\mathcal{D} \rightarrow f(x; \mathcal{D}))$ , where f is capable of providing a prediction function on a per-task basis, taking into account both $x_T$ and the context set $\mathcal{D}$ (Dubois et al., 2020). This reformulation encapsulates the paradigm of "learning to learn", or more explicitly, learning how to adapt the predictor function to new tasks based on the provided context $\mathcal{D}$. It also allows to return predictor functions that can leverage information across tasks using the context sets of new unseen tasks.

### 2.2 Neural Process model

The Latent Neural Process model (NP) , first introduced by Garnelo et al., 2018 , combines Neural Networks computational efficiency and flexibility in fitting to data , and Gaussian Processes (GPs) capability to represent distribution over functions and give uncertainty estimates of predictions. Since then, a wider range of architectural solutions have been derived to address initial shortcomings of NP , these are collectively termed as the Neural Process Family (Jha et al., 2023) , ranging from convolution and attention mechanism extensions.

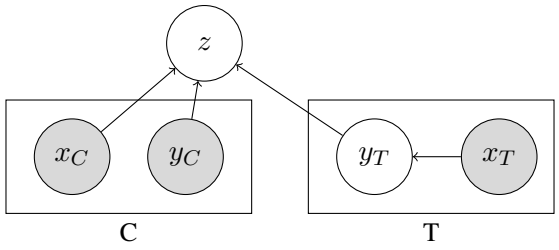


Figure 1: Graphical Model of the Neural Process , taken from (Garnelo et al., 2018)

In this section, the three core components (**encoder**, **aggregator**, **decoder**) and training of a neural process model in context of 1D regression will be laid out, to provide preliminary background knowledge. For 1D regression task learning with the NP model, we define a meta-dataset of regression tasks as $D_{mr} = \{D_{task}^{(i)}\}_{i=1}^{N_{\text{task}}}$ where a task $\mathcal{D}_{\text{task}} = ((X_c, Y_c), X_t, Y_t)$ consists of a set of $x, y$ values of a function to be regressed using the context set $\mathcal{D}_c = (X_c, Y_c)$. Succintly, we organise a sampled regression task into two sets, with sizes $n, m$.:

$$\mathcal{D}_c = \{(x_i, y_i)\}_{i=1}^{n}, \quad \mathcal{D}_t = \{(x_j, y_j)\}_{j=1}^{m}$$

The problem the NP model needs to solve is: given $\mathcal{D}_c$ and target function inputs $X_t$, predict the corresponding $Y_t$ values. However, given the nature of the Neural Process model , which is capable of predicting distributions around predictions, the output for each target $x_t$ is a Gaussian distribution of the form $\mathcal{N}(\mu_{y_t}, \sigma_{y_t})$ , that offers a probability distribution over the to be predicted value $y_t$.

Firstly, the **encoder** component $h$ takes in the context set of $(X_c, Y_c)$, or in earlier terms $\mathcal{D}_c$. These context points are encoded into a representation $r_i = h((x_{c_i}, y_{c_i}))$ per context pair. The encoder is parameterised with an Multilayer Perceptron (MLP).

These representations are **aggregated** together by their mean into $r$ , this global representation is used to parameterise the latent distribution , for global latent variable $z \sim \mathcal{N}(\mu(r), \sigma(r))$, as can also been seen from Figure 1, that the context set infers the latent variable. The importance of $z$ and related distribution, is to capture a global function representation over the set of $X, Y$ points the model uses for calculations, such that it can learn the underlying regression function $f_d$ for a given task where $f_d(x_i) = y_i$, rather than having to learn an explicit mapping to each $y_i$.

The **decoder** receives the sampled global latent variable $z$ and combines it with the target points $X_t$ to predict $p(Y_t \mid X_t; D_c)$ . As mentioned earlier, the encoder takes $\mathcal{D}_c$, importantly, during training however, the target to be predicted is formed as $\mathcal{D}_c \cup \mathcal{D}_t$. Meaning that the decoder will combine $z$ and $\mathcal{D}_c \cup \mathcal{D}_t$, during testing only $\mathcal{D}_t$ is used.

### 2.3 Curriculum and Self-Paced Learning

**Curriculum learning** , first explored for machine learning by Bengio et al., 2009 , is a learning strategy that draws upon the way human's get to learn and get better overtime, by building up from smaller easier tasks to larger more complex tasks.

The main setup of a Curriculum Strategy is a difficulty measure, that determines the easier and harder tasks, and a training scheduler, that determines the nature of introduction of the ordered tasks for the model to learn from (X. Wang et al., 2021).

**Self-Paced Learning** is an adaptive curriculum strategy, where the difficulty measure is determined by the model's capability, and is encoded through the loss the model produces on a per training example basis. This can be thought of as synonymous to a student "self-learning" and setting up a curricula based on their current performance and accrued understanding. Therefore Self-Paced Learning offers an adaptive curriculum, that changes as the model trains over time. The SPL objective was first formulated into a weighted loss optimization objective by Kumar et al. (2010) , where the usual loss minimzation scheme was extended to include the per sample weights and accompanying weighted loss. The objective is then given by:

$$\min_{\mathbf{w},\theta} \sum_{i=1}^{N} w_i \ell(f(\mathbf{x}_i;\theta), y_i) - \lambda \sum_{i=1}^{N} w_i$$

Where:

- $\mathbf{w} = [w_1, w_2, \ldots, w_N], w_i \in [0, 1]$ are the weights assigned to the training samples,
- $\theta$ are the model parameters,
- $\ell(f(\mathbf{x}_i;\theta), y_i)$ is the loss for sample $i$,
- $\lambda$ is the thresholding parameter (often called *age parameter*)

The above objective is often solved by an alternating optimization strategy over the curriculum weights and model parameters. Firstly , with model parameters kept constant the optimal weight parameters can be found by:

$$w_i^* = \begin{cases} 1, & \text{if } \ell_i < \lambda \\ 0, & \text{otherwise} \end{cases}$$

Secondly, by fixing the weights, a normal gradient descent algorithm can then optimise the model parameters. Overtime the threshold $\lambda$ (*age parameter*) is increased with each epoch, to include harder samples, as determined by the model's loss on the samples. The nature of this increase can be understood as the training scheduler. In practice , the samples for which $w_i = 1$ are considered "easy" or low loss , and will be used to form the current epoch's training subset , in our meta-learning notation $M_{ts}$. The *age parameter* updates determine the pace at which $M_{ts}$ approaches the full dataset $M_t$, $|M_{ts}| \sim |M_t|$ over the epochs, an example visualization of how the subset gets to include more of the samples based on losses can be seen in Figure 2. So, over-time the pacing leads to the introduction of training examples with higher losses, acting as a filter on the admissible examples, until the model starts training on the whole training dataset $|M_t|$.

This original version of Self-Paced Learning has since been extended , to include more informative and task specific metrics, for example approaches that include a querying aspect for relevant data selection (Tang & Huang, 2019), include more diverse training data to improve imbalanced data scenarios (Jiang et al., 2014), or to incorporate prior knowledge in form of a predetermined curricula restricting the weight space of samples (Jiang et al., 2015). Further natural extension in terms of a soft-weighing scheme based formulation (Zhao et al., 2015) has also been explored, in order to increase flexibility of the difficulty encoding that is binary in the original version. Recent work has also explored SPL within a meta-learning context with added cross-query reguralization for few shot image classification tasks, named SepMeta by Zhang et al. (2022).

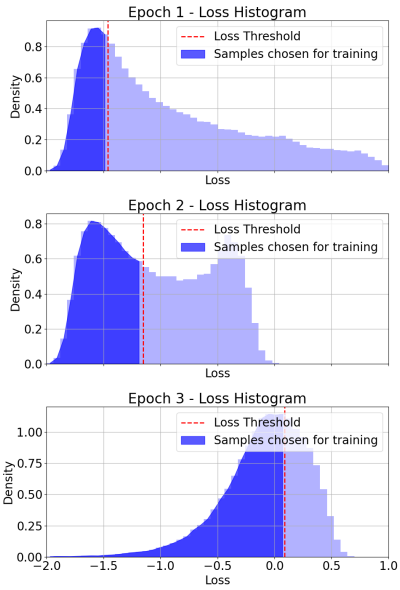


Figure 2: Example visualization of how the Self-Paced Learning threshold selects a subset $M_{ts}$ over 3 epochs. The red dashed line shows the threshold up to which examples are chosen based on the current distribution of losses

## 3 Experimental Setup

As mentioned in the introduction, prior research has shown SPL to be an effective technique against noisy data in traditional learning settings. However, applicability with similar outcomes in meta-learning has not been explored. In order to investigate SPL offering the same benefits, an NP model was trained with different ratio of clean/noisy training data splits. Training loss and performances were then used to answer how SPL can aid in a noisy meta-learning setting.

**Dataset:** The training meta-dataset consists of **128,000** regression tasks (see Figure 3 for an example). Each task is synthetically created with 96 uniformly sampled $(X, Y)$ points in the range $[-1, 1]$ from families of sinusoidal functions. Each task is synthetically created with uniformly sampled $(X, Y)$ points in the range $[-1, 1]$ from families of sinusoidal functions. These points were further split into context and target sets randomly during training. Each sample consists of 64 points for the context set $\mathcal{C}$ and 32 points for the target set $\mathcal{T}$.

$$\mathcal{C} = \{(x_i, y_i)\}_{i=1}^{64}, \quad \mathcal{T} = \{(x_j, y_j)\}_{j=1}^{32}$$

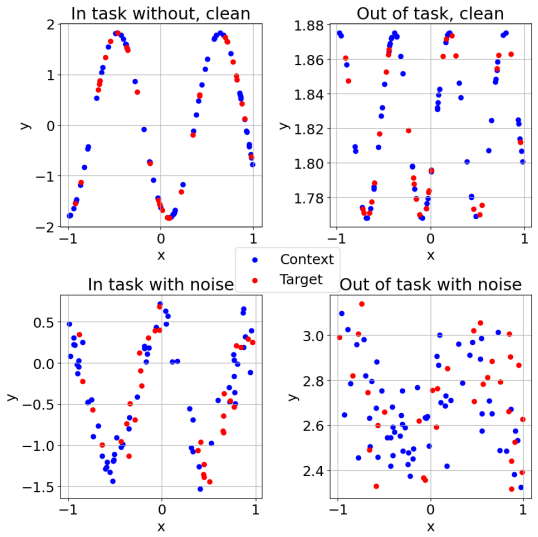


Figure 3: Examples of the training dataset (in task) , and the intra training evaluation out of task distribution, with and without noise.

The sine regression tasks were created using a randomizing Fourier series function, that allowed to parameterise the number of sine components $n$, the range for $amplitude$ and $period$ , and returned random sinusoidal functions from the thus defined distribution of possible regression tasks. A detailed explanation on the data generation and parameters can be found in Appendix A.1. The models were trained on three training data setups with varying proportions of noisy and clean training data:

- 0% Noisy / 100% Clean training data in the meta-dataset
- 30% Noisy / 70% Clean training data in the meta-dataset
- 60% Noisy / 40% Clean training data in the meta-dataset

These noisy splits allow evaluation of SPL as a technique for noise robustness for meta-learning in presence of noisy data, as well as whether the effects are specific to certain ratios of noisy data present in the training data.

**Noise:** Noise was introduced within the regression tasks by perturbing the $Y$ values of the sampled points by a Standard Gaussian $\epsilon \sim \mathcal{N}(0, 1)$ applied to all $\hat{y_i} = y_i + \epsilon_i * s$ , where $s = 0.2$ (referred to as 0.2 Noise in the paper, and $s$ as noise level) points within a task if that task was made noisy, within a given noisy / clean training split setup. Noisy training examples can be seen from figure 3.

**Evaluation metric:** For the evaluation metric, **Empirical Cross Entropy (ECE)** was used. In a normal regression , the output of the model would entail a single scalar value $y = f(x)$ for the function value at a given point $x$ . However in order to capture the uncertainty estimate of the NP model in the accuracy measure, normally employed regression metrics will not suffice. Empirical Cross Entropy can be defined as:

$$\text{ECE} = -\frac{1}{N}\sum_{i=1}^{N} \log P(y_i|x_i)$$

where $P(y_i|x_i)$ is the probability of the target value $y_i$ given the predicted distribution at $x_i$. This metric determines how likely the target value is, given our predicted distribution. In evaluation, the ECE value over a single task was computed first, according to the aforementioned formula. To determine the performance on a test set, the individual ECE values from the tasks within that test set were further averaged.

Furthermore, despite the availability of the uncertainty estimates, **Root Mean Squared Error (RMSE)** was also used for evaluation the same way, in order to evaluate the predicted means only as well.

**Training:** The training setup is available from the code [1] , a single model training script has been setup for both the baseline NP , and NP with SPL. The script allows providing a manual dataset and data-loading seed. For each clean / noisy split training data setup the models were trained on 10 different dataset and data-loading seeds (consistent across the 2 models) to ensure the results are not influenced by specific seed initalizations, altogether 60 models were trained. An epoch is defined as a whole pass over the current training data subset $M_{ts}$. In case of the Base NP model, it always holds that $M_{ts} = M_t$, but the SPL curriculum leads to epochs where $|M_{ts}| < |M_t|$. A training step is a gradient descent step and update for a single batch , for this experiment a **batch size of 128** was used. To further guarantee consistent training conditions a **training step restriction of 6000** has also been introduced to ensure both models train on consistent amount of available information. During training at every **500 training steps** , an evaluation was run on four test datasets sampled on the fly (so different dataset for each 500 step evaluation period, but consistent across models within training runs through manual seeding), all **intra training evaluation test sets of size 12800**. Two of these test set was comprised of tasks that are drawn from the same training task distribution (in-task distribution), clean and noisy (0.2 Noise). The other two evaluation sets were drawn from an out of training task distribution (different from training data distribution, further details in Appendix A.1) , clean and noisy respectively. For these four test sets metrics mentioned above were calculated. The intra training evaluation setups allows analysis with regards to the effects of SPL on both the training trajectory and generalization capabilities, and their dynamics of change throughout the learning process. The training also saved the accrued SPL curriculum weights, that is, which samples are included in $M_{ts}$ in each epoch. The final best model parameters and intra training metrics over the different in task and out of task distributions were saved for later visualisations across the seeds.

**Evaluation:** Post training evaluation was carried out with in and out of task distributions with **sizes 12800**. Increasingly more out of task distributions were used for evaluation runs with 0.2 Noise and without noise to observe the extent of generalization capability of the model and possible improvements offered by the curricula in presence of noisy test as well. Regression tasks were created while changing $Fourier$ function parameters one at a time. All the out of task distribution setups can be seen in Table 1. The in task generalization performance evaluation was carried out with varying

[1] https://github.com/aszi09/SPL-NoisyMetaLearning

noise level $s \in \{0.0, 0.2, 0.3, 0.4, 0.5, 0.6\}$, in order to assess and observe the nature of possible noise robustness against increasing levels of noise that SPL potentially offers.

| Parameter(s) | Values |
|---|---|
| $n$ | $\{3, 4\}$ |
| Amplitude | $\{1.0, 2.0\}$ |
| Period, Period Range | $\{(1.0, 0.5)$ , $(1.5, 0.5)\}$ |
| Each setup is evaluated with noise levels: 0.0 , 0.2, 0.4 | |

Table 1: Evaluation Setups for Out-of-Task Distributions, for further clarification on the meaning of the parameters and values refer to Appendix A.1

**SPL setup:** The SPL implementation of CurML(Zhou et al., 2022) has been adapted to JAX, for defining the curricula a start_rate of 10 % have been used, and the subset $M_{ts}$ increases to the full training set size within 5 epochs. Although as previously discussed , the training step size restriction is required to allow equal learning for both baseline and curriculum setups, we still require the epochs to infer the curriculum training schedule growth as well.

**Tools and Technical Setup:** For model setup and dataset generation JAX (Bradbury et al., 2018), for dataloading Pytorch was used, as there was an available codebase for Neural Process model coded in JAX, both frameworks also offer reproducibility through manual seeding. The training was carried out on DelftBlue ((DHPC), 2024) on a single NVidia Tesla V100S 32GB, with 64G RAM. However the experimental setup allows for commercially available GPUs to run the experiment, as the code have been written to adapt chunking computations for VRAM bottlenecks.

## 4 Results and Discussion

Figure 4. shows the training trajectories for both Base and SPL-based Neural Process models across different noisy training scenarios. SPL led to faster loss convergence at around the 1500 training step mark , as also supported by the ECE intra-training performances in the Sub-figures 6 , 7. Also, to more stable learning, as shown by less variance in per training step training loss, except for a slight increase in the 30% noisy setup. Interestingly, there were distinct sudden increases in training loss aligning with the 28% and 82% curriculum subset marks.

These results demonstrate that SPL affects meta-training by accelerating loss convergence and reducing variance across runs, pointing to a more robust learning process. Improved convergence points to usefulness of CL in restricted learning time scenarios as pointed out by Wu et al. (2020), that is shown to hold for meta-learning as well. The potential distribution shifts causing the loss jumps could suggest a connection to transitioning from different higher confidence regions of the underlying training data, as mentioned by Gong et al. (2015).

From sub-figures of Figure 8. the intra training evaluations ran every 500 training steps can be observed, showing average performances of Base and SPL models over the 10 runs and 95% CI can be seen. The noisy training split Figures 6 and 7 show highly similar performance trends, meaning that the ratio of noisy training data is less of an important factor as much as the presence of noise. SPL model shows slightly worse in-task performance and significantly worse noisy in task performance in terms of ECE. Out of task performance however clearly points to the SPL based model outperforming the Base model in both RMSE and ECE, also showing no significant difference due to the added level of noise.

These results point to SPL preventing the model from overfitting to noise, meanwhile improving overall generalization capability as indicated by performance on out of task regression tasks being invariant to introduced noise. This generalization capability is likely due to the model being exposed to fundamental features early in training as a result of SPL. Then, this exposure allows the model to later leverage key features improving generalization across different tasks.

From sub-figure 5 the clean training split intra training performances can be seen. The results show SPL to have highly degraded performance in presence of noise for in-task distribution test cases. Also , performance on overall out of task test cases aligns with the Base model, having progressively worse performance due to overfitting on the training distribution.

The above results show that SPL lead to increased generalization capability for out of task distributions despite added noise level. It aided meta-learning by preventing the model from overfitting to noise, leading to robustness. Furthermore , as demonstrated by previous work by Wu et al. (2020) SPL, as a CL, even in meta-learning only seems to offer benefit in noisy scenarios . The showcased overfitting in clean scenario is a problem of SPL due to repeated learning on same easy examples, as also explored by Jiang et al. (2014).

Post training, in task distribution performances can be seen plotted for ECE in figure 9 (RMSE in appendix 14) with increasing level of noise introduced. The usage of SPL curriculum greatly improved in task generalization performance for increasing levels of noise. Meanwhile Base model also shows faster degradation of performance compared to SPL based model as the regression tasks get more noisy. In the clean training setup case, performances are similar as suggested by also the intra training metrics, although with higher uncertainty for the SPL based model.

The results above show SPL helping meta-learning by improving in task generalization performance in presence of varying noise levels, as well as on par performance for clean test cases. These findings further showcase how SPL can aid meta learning in presence of noisy training data.

Post training, out of task distribution performance measurements based on the setup mentioned in Table 1 can be seen from Table 2, with the most important collective visualization for ECE visible from Figure 10 (and further visualizations from Figures 11, 12, 13), for all the different added noise and noisy training setups. RMSE performances across models and setups tend to be similar with negligible degree of differences suggesting that the model performance differences can mostly be observed through the uncertainties in predictions of the model and not the overall raw values. Figure 10. shows that in noisy training setups, the SPL model significantly outperforms the Baseline model in terms of al-

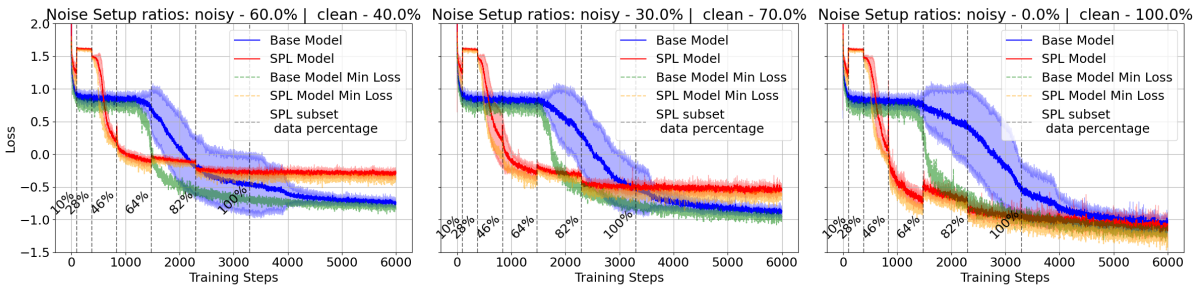


Figure 4: Training losses averaged over 10 seeds per training step, showing the standard deviation as a fill. The dashed line show the curriculum subset progression over the training with aforementioned $start\_rate = 0.1$ , $growth\_epochs = 5$.

most all types of out of task distributions for ECE. The 30% noisy |70% clean based SPL model however shows very high uncertainty in the $amp = 2.0$ case. In clean training setups the inclusion of SPL degrades out of task distribution performance compared to Base model. Furthermore, the ratio of training data made noisy, only leads to marginal performance increase for the 60% noisy setup in other distributions.

These results further showcase the findings of improved generalization even across more varied out of task distributions. These indicate that SPL helps noisy meta-learning by preventing the model from overfitting to the noise, as well as allowing it to leverage key features from the data, leading to a robust model. Also, again showing that the ratio of noise during training is less of a factor, than overall presence of noise in the training data.

### 4.1 Limitations

The study's findings on how SPL can benefit noisy meta-learning solely focuses on sinusoidal regression task, which has limited potential real-world applicability. The level of diversity found within real world datasets is much greater than the regression task datasets that have been used for training. To enhance the relevance of the findings, and ensure they generalise to real world use cases, a more diverse set of regression tasks could have been devised. In addition, leveraing commonly used meta-learning datasets, for example Omniglot (Lake et al., 2015) or a larger collection of datasets like Meta-Dataset (Triantafillou et al., 2020), could also take steps towards real world applicability, as well as allow for a more standardised basis of comparison and validation of findings with other studies.

The experimental design in the study includes only a few specific clean / noisy training splits , noise levels and types of noise, which may not comprehensively capture the noisy data challenges faced in real world datasets. Exploring a more extensive set of noisy data ratios, levels and types of noise could provide deeper insights into potential shortcomings regarding the applicability of SPL in such scenarios. Moreover, using more varied hyperparameter initialization could provide a more nuanced understanding of the specific noisy data characteristics where SPL is applicable. For example, ablation studies on the sensitivity of hyperparameters to noisy training data ratios, noise levels and types, a well as mixture of these, could reveal more general or specific benefits of SPL. Differences in the start_rate might be important for convergence improvements in different noisy training data ratios as that has not been observed greatly. Meanwhile, the growth_epochs could potentially influence the overall generalization improvements that the study showed.

Lastly, the study's investigation into SPL's applicability could further benefit from including other methods for noise robustness, to provide possible comparison and place the findings in a wider scientific context. For example, inclusion of other CL techniques like meta-weight net (Shu et al., 2019) or more general noise robustness techniques like dropout could offer comparative insights into SPL's noise robustness benefits.

## 5 Conclusions and Future Work

This paper set out to empirically investigate the potential applications of Self-Paced Learning for meta-learning in presence of noisy training data , and analyse its effects on the training process and consequent in and out of task generalization performance. The study has found that applicability of SPL for noisy meta-learning, aligns with the observed benefits in traditional learning settings, therefore showcasing SPL as a promising technique for noisy meta-learning as well.

The SPL based model has been found to significantly improve training loss convergence over the Base model overall. Furthermore SPL lead to stabilization of per training step losses over the different runs, when compared to the high standard deviation of losses throughout training of the Base model. Overall leading to more robust meta-learning, as also showcased by the improved generalization performances in the intra-training evaluations.

In line with previous studies on CL and SPL, the study has shown that application of SPL in clean data scenarios is highly limited, despite the training loss convergence. This is attributable to SPL overfitting in clean data training setups. Out of task distribution performance is crucial for the future of meta-learning, as improvements in this area could lead to greater efficiency in training data to be leveraged across tasks, leading to overall decrease in training times and more capable models.

The study has found that SPL improves the out of task generalization capability of the model compared to Base model, hence also increases noise robustness. In task distribution performance differences in less noisy scenarios remained

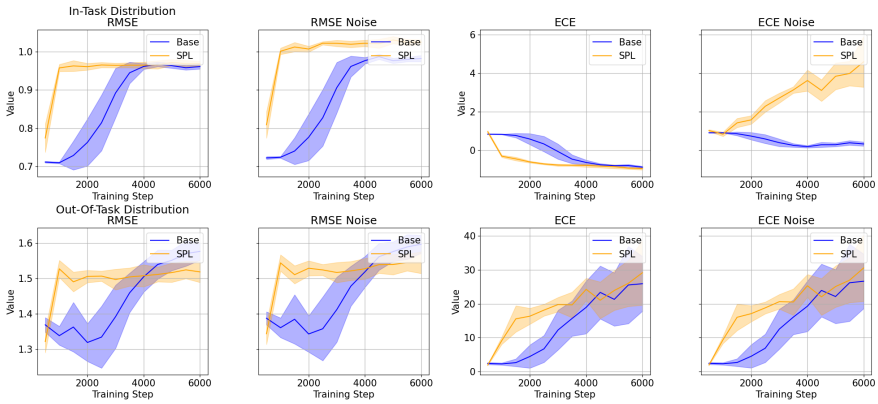


Figure 5: Intra training metrics for the Noise setup ratio: noisy - 0.0% |clean - 100.0%. As we can be seen, SPL does not offer any added performance benefit on with clean data. It also leads to overfitting, degrading performance on In-Task ECE Noise tests cases.

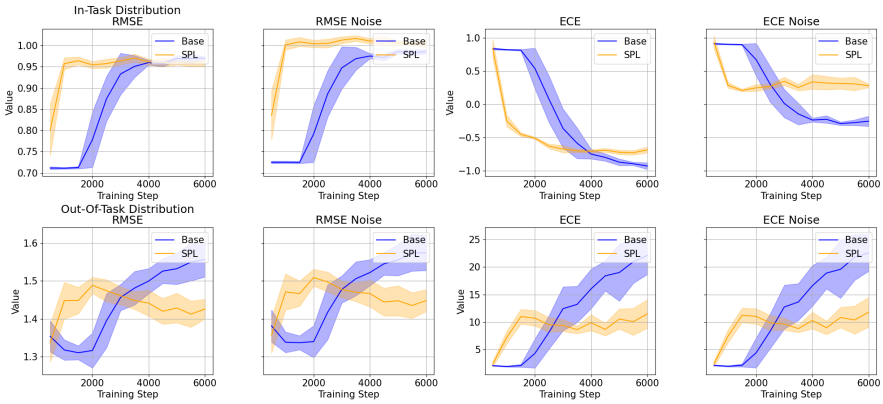


Figure 6: Intra training metrics for the Noise setup ratio: noisy - 30.0% |clean - 70.0%. SPL also offers robustness in meta-learning as suggested by efficacy for noisy learning in traditional learning settings. Increased out of task generalization capability can be seen. SPL further prevented the model from overfitting on the noise found in the training dataset.

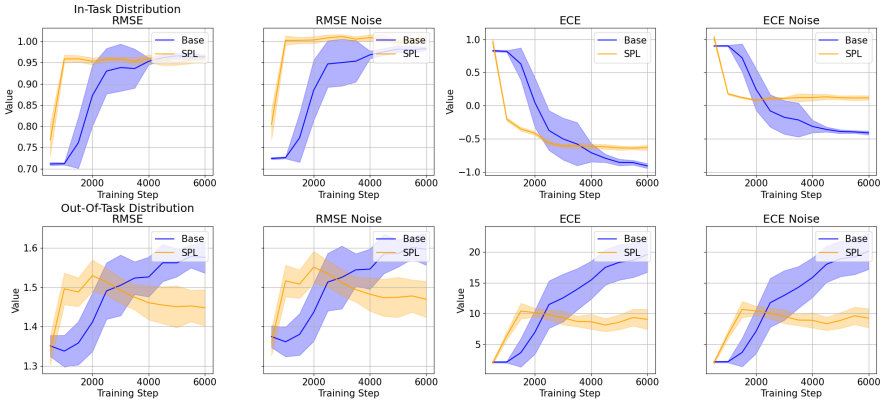


Figure 7: Intra training metrics for the Noise setup ratio: noisy - 60.0% |clean - 40.0%. SPL also offers robustness in meta-learning as suggested by efficacy for noisy learning in traditional learning settings. Increased out of task generalization capability can be seen. SPL further prevented the model from overfitting on the noise found in the training dataset. Even in the presence of higher amount of noisy data in the training dataset.

Figure 8: Sub-figures for the Intra training RMSE and ECE performance metrics showcasing the per 500 training step evaluation runs with 12800 test datasets of in and out of task distributions with clean and 0.2 noise = (_noise) added to the test tasks. The lines are the aggregated means and the fill is the 95% CI from the means of the 10 runs.

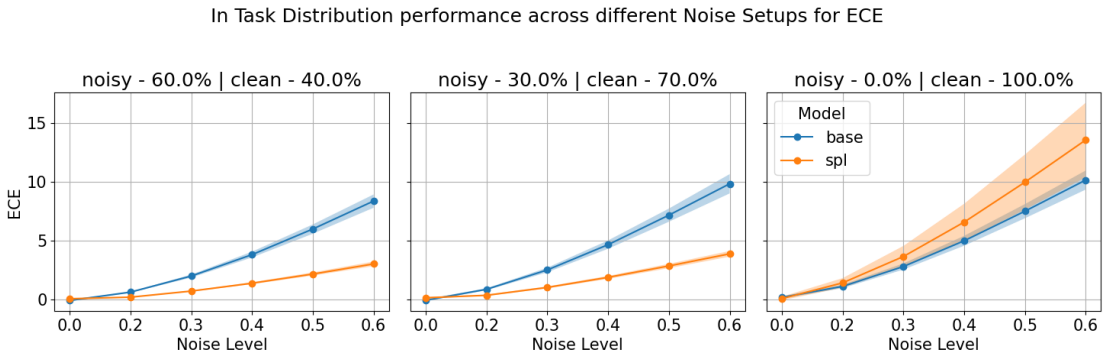


Figure 9: Post training in task distribution ECE performances in different noisy training setups, and with increasing level of added noise, averaged over 10 runs, fill showing 95% CI. Sub-figures showcase the benefit of SPL for meta-learning along the same lines as has been found in traditional learning settings.

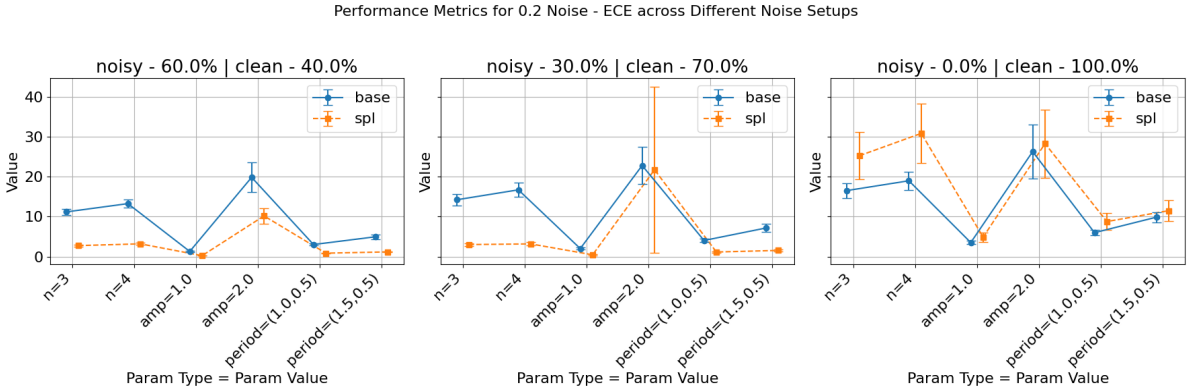


Figure 10: Out of task distribution ECE performances averaged over the 10 runs with 95%CI , with 0.2 Noise. The graph shows the improved generalizatoin capability of the model due to SPL in the noisy training splits. Although with very high uncertainty in the 30% noisy $amp = 2.0$ case. Overall clear improvement for out of task distribution can be observed in noisy trainging setups, and degradation due to overfitting in the full clean training setup

only marginal, however SPL has increased noise robustness with regards to increasing level of noise compared to Base model. Furthermore RMSE measures have shown that the improvements offered by SPL can mostly be attributed to a decrease in uncertainty of the Neural Process model predictions as shown by improved ECE but not always RMSE.

Although the findings of the study showcase some promising results, as the limitations point out, there are shortcomings that need to be further explored more deeply to make sure that SPL is a technique worth incorporating in real world meta-learning scenarios. Several directions of exciting future work can be discussed not only in connection with SPL but with regards to Curriculum Strategies in general.

Next step in researching SPL and Meta-learning could look at different pacing functions, as well as already established and used SPL versions, such as soft weighing (Zhao et al., 2015) or diversity (Jiang et al., 2014), in meta-learning scenarios. Research on more diverse meta-learning scenarios are still yet to be explored, one highly influential avenue of work could include exploring imbalanced training data settings in meta-learning , and how Curriculum Strategies could help mitigate such scenarios. As one of the main goals of meta-learning is out of task distribution performance with only few data, such imbalanced training data setups could be explored with different Curriculum Strategy setups to observe its effects in mitigating imbalance of certain task types.

Another important and interesting approach to future work could include focus on deriving new CL techniques or build upon previous ones that solve more meta-learning specific problems. For example, one major challenge that has been identified in multi-task learning scenarios and speculated to also appear in meta-learning is conflicting gradients, high positive curvature and large gradient differences (Yu et al., 2020). Exploring how these characteristics apply to meta-learning and developing new Curriculum Strategies to mitigate these issues would be a valuable direction. In addition, recent paper has already explored the use Variance of Gradients (Agarwal et al., 2022) for estimating example difficulty, which could be further explored for inferring curriculum for meta-learning.

## 6 Responsible Research

Reflecting on the ethical, societal impact and values provided by a given research is crucial for furthering science with integrity and reproducibility in mind. Reflection on the conducted research based on principles from the Netherlands Code of Conduct for Research Integrity is found in this section.

In terms of reproducibility of the research, the experiment has been setup in several aspects in order to improve reproducibility. The data used in the experiments are synthetically created, using random seeding, therefore reproducing the required data to reproduce the same results for the post training evaluations is freely available in the appendix. Manual random seeding has further been used to ensure reproducibility of results, both random initilizations of parameters and dataloading. The experimental setup section details the experiment in depth with added details observable from Appendix A.1 as well the code [2], further improving the reproducibility of the research. In addition, the experiment has been setup to allow for VRAM chunking, making the training accessible on commercially available GPUs as well.

In terms of Honesty, the paper included uncertainty measures over the shown performance metrics, as well as showed detailed statistical information for the training losses. Furthermore, data used for the graphs can also be found in the appendix in the form of tables for further clarity and honesty declaring the results.

The principle of scrupulousness is reflected in the research, by carefully detailing each aspect of the experimental design, making it available in the appendix, and providing available code for result reproduction.

The research is aligned with principles of transparency, by providing clear explanations of methods, code and data creation used for the experiments. The code has been made public, and there are no third party or related stakeholders connected to the research that are undisclosed. Furthermore, export of surveyed papers and relevant notes are also available from the code repository, to offer further transparency of the research process.

Independence of the research is ensured, as no commercial or other non-scholarly parties were involved or were invested in the formulation, making or implications of the conducted research. The study concerns solely the student, supervisor and responsible professor, all of whom are impartial and dedicated to research with integrity.

In reflection on ethical and social impact and related principle of responsibility, the data usage and real-world applicability is concerned. The experiment has used synthetic data, therefore there is no possible privacy concerns. As the limitations of the study has also pointed out , real-world applicability of the findings is constrained. Although this is the case, the general ethical and social impact of machine learning model robustness, is of great importance. In society, more and more key infrastructure systems are incorporating models that must provide reliable information, as their impact could affect people immensely. By researching noise robustness, the research aimed at fulfilling the principle of responsibility by striving for scientific and social relevance.

[2] https://github.com/aszi09/SPL-NoisyMetaLearning

## References


Agarwal, C., D'souza, D., & Hooker, S. (2022, June 21). Estimating example difficulty using variance of gradients. https://doi.org/10.48550/arXiv.2008.11600

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *60*, 6. https://doi.org/10.1145/1553374.1553380

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.3.13). http://github.com/google/jax

Chen, D., Wu, L., Tang, S., Yun, X., Long, B., & Zhuang, Y. (2022, June 4). Robust meta-learning with sampling noise and label noise via eigen-reptile. https://doi.org/10.48550/arXiv.2206.01944

(DHPC), D. H. P. C. C. (2024). DelftBlue Supercomputer (Phase 2).

Dubois, Y., Gordon, J., & Foong, A. Y. (2020, September). Neural process family.

Galjaard, J. (2023, May). *Meta-Learning with Label Noise: A Step Towards Label Few-Shot Meta-Learning with Label Noise* [Master's thesis]. Delft University of Technology. http : / / resolver . tudelft . nl / uuid : 65aa4a0c-d2d6-44f1-bda8-6dd093488f40

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., & Teh, Y. W. (2018). Neural processes.

Gong, T., Zhao, Q., Meng, D., & Xu, Z. (2015). Why curriculum learning & self-paced learning work in big/noisy data: A theoretical perspective [Publisher: Big Data & Information Analytics]. *Big Data & Information Analytics*, *1*(1), 111–127. https://doi.org/10.3934/bdia.2016.1.111

Hospedales, T. M., Antoniou, A., Micaelli, P., & Storkey, A. J. (2021). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2021.3079209

Jha, S., Gong, D., Wang, X., Turner, R. E., & Yao, L. (2023). The neural process family: Survey, applications and perspectives.

Jiang, L., Meng, D., Yu, S.-I., Lan, Z., Shan, S., & Hauptmann, A. (2014). Self-paced learning with diversity. Retrieved April 24, 2024, from https : / / www . semanticscholar . org / paper / Self - Paced - Learning - with - Diversity - Jiang - Meng / 44606e1209a47d1fcf88b90e306db9e4b84fa2c5

Jiang, L., Meng, D., Zhao, Q., Shan, S., & Hauptmann, A. (2015). Self-paced curriculum learning. *AAAI Conference on Artificial Intelligence*. Retrieved April 22, 2024, from https://www.semanticscholar.org/paper/Self - Paced - Curriculum - Learning - Jiang - Meng / 21d255246cd7ddba24a651fd716950f893ea8eb2

Kumar, M. P., Packer, B., & Koller, D. (2010). Self-paced learning for latent variable models. Retrieved April 24, 2024, from https : / / www . semanticscholar . org / paper / Self - Paced - Learning - for - Latent - Variable - Models - Kumar - Packer / a049555721f17ed79a97fd492c8fc9a3f8f8aa17

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338. https://doi.org/10.1126/science.aab3050

Lu, J., Jin, S., Liang, J., & Zhang, C. (2021). Robust few-shot learning for user-provided data. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(4), 1433–1447. https://doi.org/10.1109/TNNLS.2020.2984710

Meng, D., & Zhao, Q. (2015). What objective does self-paced learning indeed optimize? *ArXiv*. Retrieved April 28, 2024, from https : / / www . semanticscholar.org/paper/What - Objective - Does - Self - paced - Learning - Indeed - Meng - Zhao / a37873860f279bfda39add3bc0caf69e2f9ffbff

Nichol, A., Achiam, J., & Schulman, J. (2018). On first-order meta-learning algorithms.

Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., & Meng, D. (2019, September 26). Meta-weight-net: Learning an explicit mapping for sample weighting. https://doi.org/10.48550/arXiv.1902.07379

Song, H., Kim, M., Park, D., Shin, Y., & Lee, J.-G. (2023). Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(11), 8135–8153. https://doi.org/10.1109/TNNLS.2022.3152527

Soviany, P., Ionescu, R. T., Rota, P., & Sebe, N. (2022). Curriculum learning: A survey. *International Journal of Computer Vision*, *130*(6), 1526–1565. https://doi.org/10.1007/s11263-022-01611-x

Sun, Q., Liu, Y., Chua, T.-S., & Schiele, B. (2019). Meta-transfer learning for few-shot learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 403–412. https://doi.org/10.1109/CVPR.2019.00049

Tang, Y.-P., & Huang, S.-J. (2019). Self-paced active learning: Query the right thing at the right time [ISSN: 2374-3468, 2159-5399 Issue: 01 Journal Abbreviation: AAAI]. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 5117–5124. https://doi.org/10.1609/aaai.v33i01.33015117

Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P.-A., & Larochelle, H. (2020). Meta-dataset: A dataset of datasets for learning to learn from few examples.

Vanschoren, J. (2018). Meta-learning: A survey.

Wang, J. X. (2020). Meta-learning in natural and artificial intelligence.

Wang, X., Chen, Y., & Zhu, W. (2021). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. https://doi.org/10.1109/TPAMI.2021.3069908

Wu, X., Dyer, E., & Neyshabur, B. (2020). When do curricula work? *ArXiv*. Retrieved April 22, 2024, from https : / / www . semanticscholar . org / paper / 9d2c96574019305a8c86cc5b84cb9f616ccf0eb3

Yao, H., Wang, Y., Wei, Y., Zhao, P., Mahdavi, M., Lian, D., & Finn, C. (2021). Meta-learning with an adaptive task scheduler.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., & Finn, C. (2020). Gradient surgery for multi-task learning.

Zhang, J., Song, J., Gao, L., Liu, Y., & Shen, H. T. (2022). Progressive meta-learning with curriculum [Conference Name: IEEE Transactions on Circuits and Systems for Video Technology]. *IEEE Transactions on Circuits and Systems for Video Technology*, *32*(9), 5916–5930. https://doi.org/10.1109/TCSVT.2022.3164190

Zhao, Q., Meng, D., Jiang, L., Xie, Q., Xu, Z., & Hauptmann, A. (2015). Self-paced learning for matrix factorization [Number: 1]. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9584

Zhou, Y., Chen, H., Pan, Z., Yan, C., Lin, F., Wang, X., & Zhu, W. (2022). CurML: A curriculum machine learning library. *Proceedings of the 30th ACM International Conference on Multimedia*, 7359–7363. https://doi.org/10.1145/3503161.3548549

# A Further details on the experimental setup

## A.1 Explanation of functions used for data generation

The Fourier series function that is used in the study to create different sinusoidal regression task distributions allows changing the nature of the regression tasks using the number of sine components, amplitude , period and period_range. Subsequent explanation of each parameter follows, to improve clarity for the reader in understanding the task distributions used in the study.

Parameter **n** allows changing the number of sine components the resulting fourier series will have, with the components being n-1, so n=2 will lead to normal sine functions.

Parameter **amplitude**, determines the uniform distribution of possible amplitude values per sine component. So the resulting amplitude value for a given sine component $a_i$ is defined as : $a_i = a - 1, \quad a \sim U(amplitude * 2)$

Parameters **period, period_range** define the distribution of periods the individual sine components can have. The period is a scalar supplied parameter. The period_range a shift in the supplied period parameter, $period\_shift = p_s - period\_range, \quad p_s \sim U(period\_range * 2)$. Then for each sine component the period is $p_i = period - period\_shift$

Furthermore **shift** is sampled $shift \sim U(\pi)$ , affecting shift of $x$ values.

Another function that is used to make the regression tasks more complex is **Shift**, which determines further possible x and y value shifts of the sampled regression tasks. Both allow a base parameter for raw scalar shifts $x\_shift, y\_shift$ or parameters $x\_shift\_range, y\_shift\_range$ that determine uniform distributions to sample $x\_shift, y\_shift$ from. Then the final $x\_shift, y\_shift$ values of a regression task is given by $x\_shift = x\_s - x\_shift\_range, \quad x\_s \sim U(x\_shift\_range * 2)$ and $y\_shift = y\_s - y\_shift\_range, \quad y\_s \sim U(y\_shift\_range * 2)$ respectively.

- The training data distribution (in task distribution) has the following insantiation of Fourier: $Fourier(n = 2, amplitude = 0.5, period = 1.0, period\_range = 0.2)$.
- The out of task distribution for the intra training evaluations has the following insantiation of Fourier: $Shift(Fourier(n = 2, amplitude = 1, period = 1, period\_range = 0.5), x\_shift = 0.0, x\_shift\_range = 1.5, y\_shift = 0.0, y\_shift\_range = 3.0)$

## A.2 More detail on training params

Training is initialized with the Adam optimizer from the Optax (Bradbury et al., 2018) library, with $learning\_rate = 0.001$ and $weight\_decay = 0.000001$. Furthermore $clip = 0.1$ and $clip\_by\_global\_norm = 1.0$ has been used for all models trained.

# B Graphs of Out of Task performances based on values in Table 2.

# C Varied out of task evaluations across different noise setups

# D In task performances table for each noise setup

# E RMSE in task distribution performance

# F Use of LLM

LLM has been used for occasional sentence rephrasing, conciseness, grammar checking and the tikz created image in Figure 1.

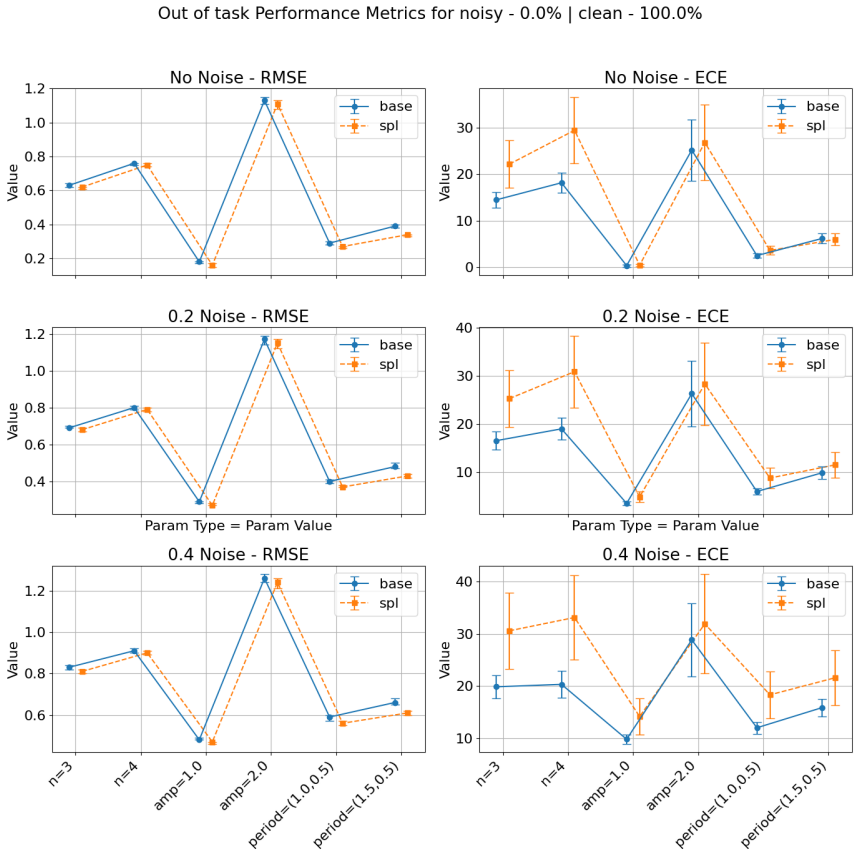


Figure 11: Post Training RMSE and ECE performance metrics for differing out of task distributions, showing mean of 10 runs and 95% CI

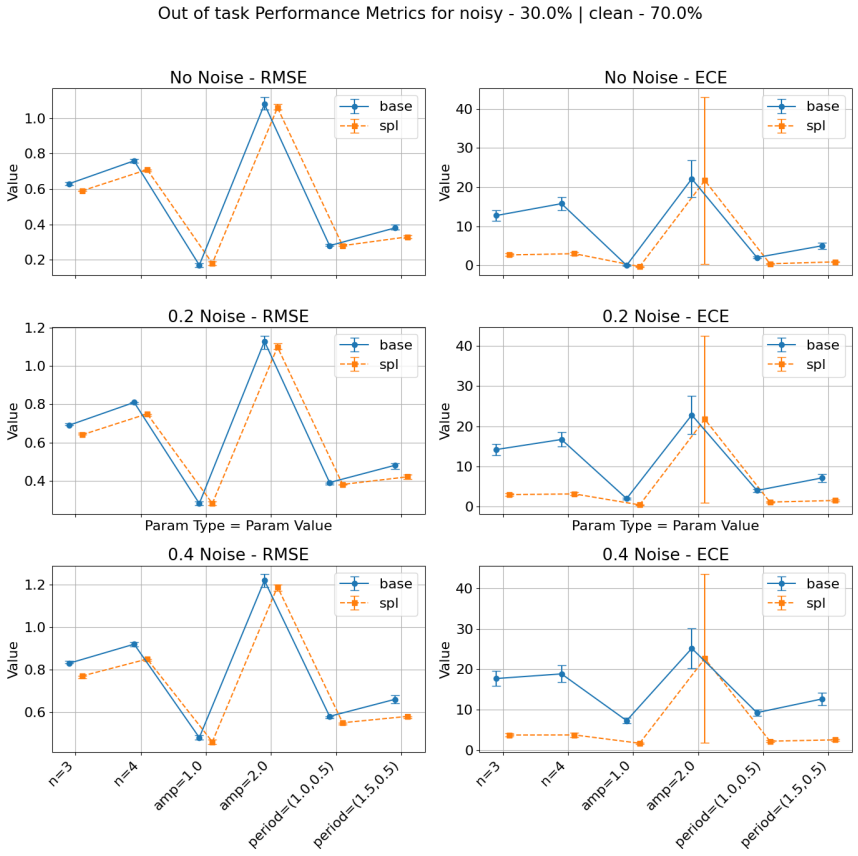


Figure 12: Post Training RMSE and ECE performance metrics for differing out of task distributions, showing mean of 10 runs and 95% CI

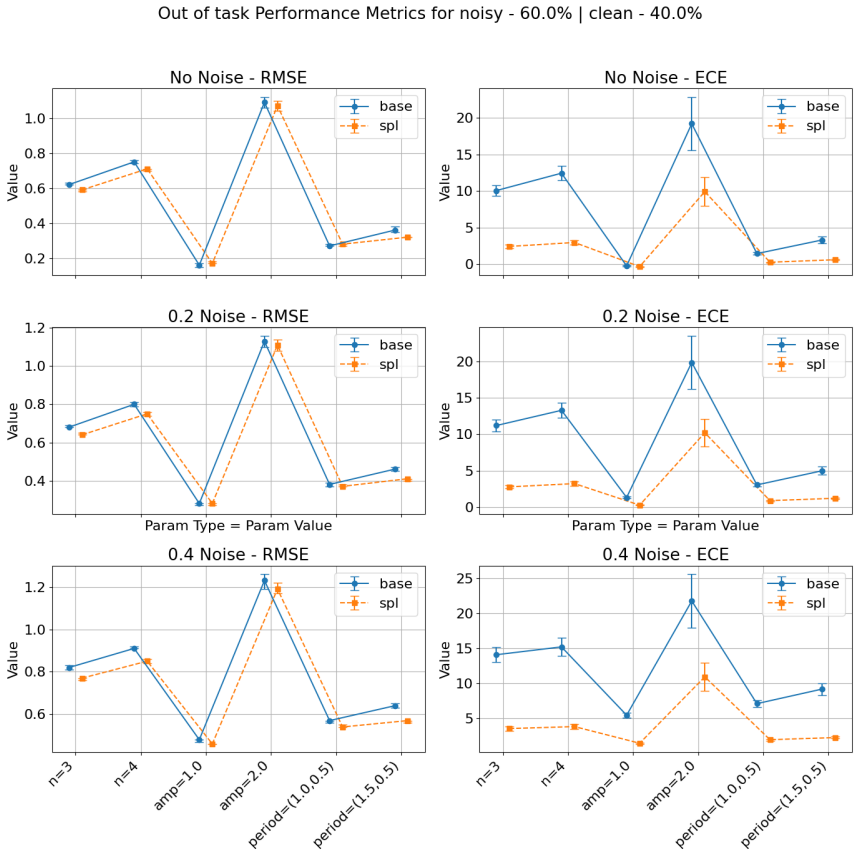


Figure 13: Post Training RMSE and ECE performance metrics for differing out of task distributions, showing mean of 10 runs and 95% CI.

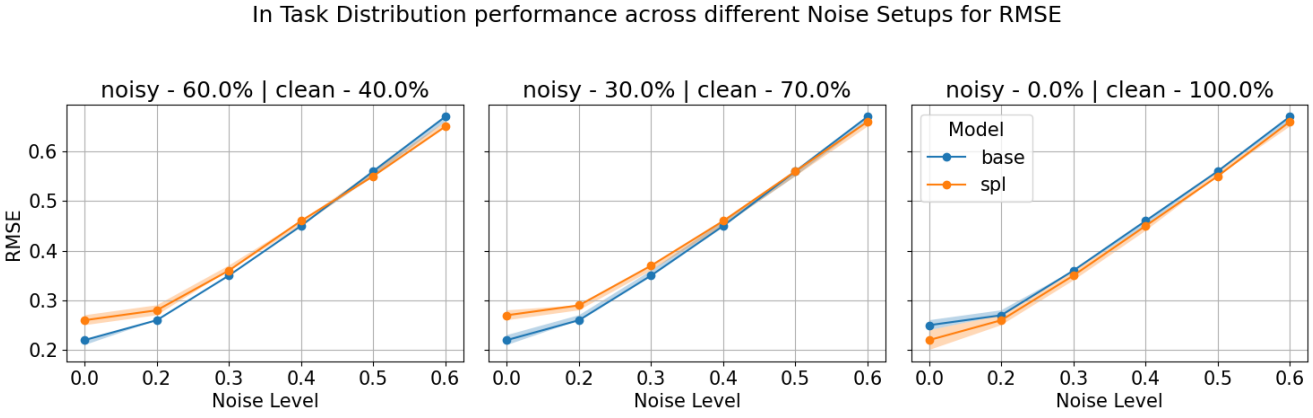


Figure 14: Post training in task distribution RMSE performances in different noisy training setups, and with increasing level of added noise, averaged over 10 runs, fill showing 95% CI. Meaningful RMSE differences for the in task generalization performance cannot be discerned, when it comes to noise robustness.

| | | | Param Type | n | | amp | | period | |
|---|---|---|---|---|---|---|---|---|---|
| Noise setup | Noise level | Model | Param Value<br>Metric | 3 | 4 | 1.0 | 2.0 | (1.0,0.5) | (1.5,0.5) |
| Noise setup 0.6-0.4 | No Noise | base | rmse | 0.62 (0.62, 0.63) | 0.75 (0.74, 0.76) | 0.16 (0.15, 0.17) | 1.09 (1.06, 1.12) | 0.27 (0.27, 0.28) | 0.36 (0.35, 0.38) |
| | | | ece | 10.08 (9.34, 10.82) | 12.46 (11.46, 13.46) | -0.23 (-0.34, -0.12) | 19.23 (15.59, 22.87) | 1.46 (1.30, 1.62) | 3.32 (2.83, 3.81) |
| | | spl | rmse | 0.59 (0.58, 0.59) | 0.71 (0.70, 0.71) | 0.17 (0.17, 0.18) | 1.07 (1.04, 1.10) | 0.28 (0.27, 0.28) | 0.32 (0.32, 0.32) |
| | | | ece | 2.44 (2.21, 2.67) | 2.98 (2.69, 3.27) | -0.32 (-0.36, -0.27) | 9.94 (8.01, 11.88) | 0.28 (0.22, 0.33) | 0.62 (0.56, 0.69) |
| | 0.2 Noise | base | rmse | 0.68 (0.68, 0.69) | 0.80 (0.79, 0.81) | 0.28 (0.27, 0.28) | 1.13 (1.10, 1.16) | 0.38 (0.37, 0.39) | 0.46 (0.45, 0.47) |
| | | | ece | 11.19 (10.38, 11.99) | 13.28 (12.22, 14.35) | 1.30 (1.16, 1.44) | 19.86 (16.17, 23.55) | 3.03 (2.79, 3.27) | 4.97 (4.40, 5.55) |
| | | spl | rmse | 0.64 (0.64, 0.65) | 0.75 (0.74, 0.76) | 0.28 (0.27, 0.28) | 1.11 (1.08, 1.14) | 0.37 (0.37, 0.38) | 0.41 (0.40, 0.41) |
| | | | ece | 2.76 (2.51, 3.02) | 3.21 (2.90, 3.53) | 0.25 (0.21, 0.30) | 10.20 (8.27, 12.13) | 0.87 (0.79, 0.95) | 1.19 (1.11, 1.26) |
| | 0.4 Noise | base | rmse | 0.82 (0.81, 0.83) | 0.91 (0.91, 0.92) | 0.48 (0.47, 0.48) | 1.23 (1.19, 1.26) | 0.57 (0.56, 0.57) | 0.64 (0.63, 0.65) |
| | | | ece | 14.09 (13.04, 15.14) | 15.21 (13.93, 16.49) | 5.45 (5.08, 5.81) | 21.77 (17.92, 25.61) | 7.10 (6.61, 7.60) | 9.18 (8.32, 10.03) |
| | | spl | rmse | 0.77 (0.76, 0.77) | 0.85 (0.84, 0.86) | 0.46 (0.46, 0.46) | 1.19 (1.17, 1.22) | 0.54 (0.54, 0.55) | 0.57 (0.56, 0.57) |
| | | | ece | 3.54 (3.22, 3.86) | 3.82 (3.44, 4.20) | 1.44 (1.34, 1.55) | 10.93 (8.95, 12.92) | 1.95 (1.82, 2.09) | 2.25 (2.13, 2.38) |
| Noise setup 0.3-0.7 | No Noise | base | rmse | 0.63 (0.62, 0.64) | 0.76 (0.75, 0.77) | 0.17 (0.16, 0.18) | 1.08 (1.05, 1.12) | 0.28 (0.28, 0.29) | 0.38 (0.37, 0.40) |
| | | | ece | 12.74 (11.45, 14.03) | 15.78 (14.13, 17.42) | -0.01 (-0.16, 0.14) | 22.14 (17.41, 26.87) | 2.01 (1.73, 2.28) | 5.00 (4.16, 5.83) |
| | | spl | rmse | 0.59 (0.59, 0.59) | 0.71 (0.70, 0.71) | 0.18 (0.17, 0.19) | 1.06 (1.05, 1.08) | 0.28 (0.28, 0.29) | 0.33 (0.32, 0.34) |
| | | | ece | 2.67 (2.36, 2.98) | 2.98 (2.60, 3.35) | -0.33 (-0.40, -0.26) | 21.71 (0.38, 43.04) | 0.40 (0.31, 0.50) | 0.89 (0.75, 1.03) |
| | 0.2 Noise | base | rmse | 0.69 (0.69, 0.70) | 0.81 (0.80, 0.81) | 0.28 (0.27, 0.29) | 1.13 (1.09, 1.16) | 0.39 (0.38, 0.39) | 0.48 (0.46, 0.49) |
| | | | ece | 14.23 (12.81, 15.66) | 16.73 (14.98, 18.48) | 2.02 (1.74, 2.30) | 22.82 (18.12, 27.51) | 4.05 (3.67, 4.44) | 7.17 (6.18, 8.17) |
| | | spl | rmse | 0.64 (0.64, 0.65) | 0.75 (0.74, 0.75) | 0.28 (0.27, 0.29) | 1.10 (1.09, 1.12) | 0.38 (0.38, 0.38) | 0.42 (0.41, 0.43) |
| | | | ece | 3.03 (2.70, 3.35) | 3.19 (2.79, 3.58) | 0.46 (0.33, 0.59) | 21.76 (0.97, 42.56) | 1.16 (1.00, 1.32) | 1.56 (1.35, 1.76) |
| | 0.4 Noise | base | rmse | 0.83 (0.83, 0.84) | 0.92 (0.91, 0.93) | 0.48 (0.47, 0.49) | 1.22 (1.19, 1.25) | 0.58 (0.57, 0.58) | 0.66 (0.64, 0.68) |
| | | | ece | 17.72 (15.87, 19.56) | 18.86 (16.77, 20.94) | 7.33 (6.64, 8.02) | 25.16 (20.29, 30.03) | 9.30 (8.49, 10.12) | 12.68 (11.19, 14.17) |
| | | spl | rmse | 0.77 (0.76, 0.77) | 0.85 (0.84, 0.85) | 0.46 (0.45, 0.47) | 1.19 (1.17, 1.20) | 0.55 (0.54, 0.55) | 0.58 (0.57, 0.58) |
| | | | ece | 3.78 (3.37, 4.19) | 3.80 (3.32, 4.28) | 1.75 (1.58, 1.91) | 22.64 (1.80, 43.47) | 2.26 (2.10, 2.42) | 2.59 (2.37, 2.80) |
| Noise setup 0-1 | No Noise | base | rmse | 0.63 (0.62, 0.64) | 0.76 (0.75, 0.76) | 0.18 (0.17, 0.18) | 1.13 (1.11, 1.15) | 0.29 (0.28, 0.30) | 0.39 (0.38, 0.40) |
| | | | ece | 14.50 (12.82, 16.17) | 18.17 (16.07, 20.28) | 0.36 (0.09, 0.63) | 25.19 (18.59, 31.79) | 2.52 (2.05, 2.99) | 6.20 (5.09, 7.32) |
| | | spl | rmse | 0.62 (0.61, 0.62) | 0.75 (0.74, 0.76) | 0.16 (0.15, 0.17) | 1.11 (1.08, 1.13) | 0.27 (0.26, 0.27) | 0.34 (0.33, 0.34) |
| | | | ece | 22.17 (17.04, 27.30) | 29.45 (22.32, 36.58) | 0.45 (0.22, 0.68) | 26.83 (18.73, 34.93) | 3.63 (2.72, 4.54) | 5.98 (4.68, 7.27) |
| | 0.2 Noise | base | rmse | 0.69 (0.69, 0.70) | 0.80 (0.79, 0.81) | 0.29 (0.28, 0.29) | 1.17 (1.14, 1.19) | 0.40 (0.39, 0.41) | 0.48 (0.47, 0.50) |
| | | | ece | 16.55 (14.67, 18.43) | 19.00 (16.72, 21.27) | 3.53 (3.11, 3.95) | 26.30 (19.50, 33.11) | 6.00 (5.34, 6.67) | 9.88 (8.60, 11.16) |
| | | spl | rmse | 0.68 (0.67, 0.69) | 0.79 (0.78, 0.80) | 0.27 (0.27, 0.28) | 1.15 (1.12, 1.17) | 0.37 (0.37, 0.38) | 0.43 (0.42, 0.44) |
| | | | ece | 25.27 (19.39, 31.16) | 30.87 (23.40, 38.35) | 4.89 (3.73, 6.04) | 28.31 (19.74, 36.87) | 8.79 (6.63, 10.94) | 11.52 (8.87, 14.16) |
| | 0.4 Noise | base | rmse | 0.83 (0.82, 0.84) | 0.91 (0.90, 0.92) | 0.48 (0.48, 0.49) | 1.26 (1.24, 1.28) | 0.59 (0.57, 0.60) | 0.66 (0.65, 0.68) |
| | | | ece | 19.86 (17.61, 22.12) | 20.33 (17.78, 22.88) | 9.80 (8.91, 10.68) | 28.85 (21.88, 35.81) | 12.01 (10.87, 13.15) | 15.87 (14.17, 17.56) |
| | | spl | rmse | 0.81 (0.80, 0.82) | 0.90 (0.89, 0.91) | 0.47 (0.46, 0.47) | 1.24 (1.21, 1.26) | 0.56 (0.55, 0.57) | 0.61 (0.60, 0.62) |
| | | | ece | 30.59 (23.32, 37.86) | 33.14 (25.02, 41.25) | 14.16 (10.74, 17.59) | 31.93 (22.41, 41.45) | 18.32 (13.80, 22.84) | 21.59 (16.35, 26.82) |

Table 2: Table of evaluation means per setup mentioned in Table 1. For each noisy training setup and level of noise. The values in the parenthesis show the 95% confidence intervals lower and upper bounds. Aggregations are over 10 runs.

Table 3: Table of In task distribution based evaluations for ECE and RMSE for both Base and SPL based models. For each noisy training setup and level of noise. The values in the parenthesis show the 95% confidence intervals lower and upper bounds. Aggregations are over 10 runs.

| Noise setup | Noise level | Model | Metric | Mean (95% CI) |
|---|---|---|---|---|
| Noise setup 0.6-0.4 | 0.0 | base | rmse | 0.22 (0.21, 0.22) |
| | | | ece | -0.07 (-0.13, -0.02) |
| | | spl | rmse | 0.26 (0.25, 0.27) |
| | | | ece | 0.08 (0.04, 0.12) |
| | 0.2 | base | rmse | 0.26 (0.26, 0.26) |
| | | | ece | 0.64 (0.59, 0.68) |
| | | spl | rmse | 0.28 (0.27, 0.29) |
| | | | ece | 0.21 (0.18, 0.24) |
| | 0.3 | base | rmse | 0.35 (0.35, 0.35) |
| | | | ece | 2.01 (1.88, 2.13) |
| | | spl | rmse | 0.36 (0.36, 0.37) |
| | | | ece | 0.73 (0.68, 0.77) |
| | 0.4 | base | rmse | 0.45 (0.45, 0.45) |
| | | | ece | 3.83 (3.59, 4.08) |
| | | spl | rmse | 0.46 (0.45, 0.46) |
| | | | ece | 1.40 (1.32, 1.48) |
| | 0.5 | base | rmse | 0.56 (0.55, 0.56) |
| | | | ece | 6.01 (5.61, 6.40) |
| | | spl | rmse | 0.55 (0.55, 0.56) |
| | | | ece | 2.18 (2.05, 2.32) |
| | 0.6 | base | rmse | 0.67 (0.66, 0.67) |
| | | | ece | 8.39 (7.82, 8.95) |
| | | spl | rmse | 0.65 (0.65, 0.66) |
| | | | ece | 3.04 (2.85, 3.24) |
| Noise setup 0.3-0.7 | 0.0 | base | rmse | 0.22 (0.21, 0.23) |
| | | | ece | -0.05 (-0.12, 0.03) |
| | | spl | rmse | 0.27 (0.26, 0.28) |
| | | | ece | 0.16 (0.10, 0.22) |
| | 0.2 | base | rmse | 0.26 (0.26, 0.27) |
| | | | ece | 0.88 (0.80, 0.97) |
| | | spl | rmse | 0.29 (0.28, 0.29) |
| | | | ece | 0.37 (0.32, 0.43) |
| | 0.3 | base | rmse | 0.35 (0.35, 0.36) |
| | | | ece | 2.53 (2.33, 2.72) |
| | | spl | rmse | 0.37 (0.36, 0.37) |
| | | | ece | 1.04 (0.96, 1.13) |
| | 0.4 | base | rmse | 0.45 (0.45, 0.46) |
| | | | ece | 4.68 (4.33, 5.04) |
| | | spl | rmse | 0.46 (0.45, 0.46) |
| | | | ece | 1.90 (1.77, 2.02) |
| | 0.5 | base | rmse | 0.56 (0.55, 0.56) |
| | | | ece | 7.19 (6.63, 7.75) |
| | | spl | rmse | 0.56 (0.55, 0.56) |
| | | | ece | 2.87 (2.68, 3.05) |
| | 0.6 | base | rmse | 0.67 (0.66, 0.67) |
| | | | ece | 9.86 (9.04, 10.68) |
| | | spl | rmse | 0.66 (0.65, 0.66) |
| | | | ece | 3.90 (3.66, 4.14) |
| Noise setup 0-1 | 0.0 | base | rmse | 0.25 (0.24, 0.26) |
| | | | ece | 0.19 (0.09, 0.29) |
| | | spl | rmse | 0.22 (0.20, 0.24) |
| | | | ece | 0.09 (-0.06, 0.25) |
| | 0.2 | base | rmse | 0.27 (0.27, 0.28) |
| | | | ece | 1.13 (0.98, 1.27) |
| | | spl | rmse | 0.26 (0.25, 0.27) |
| | | | ece | 1.43 (1.03, 1.83) |
| | 0.3 | base | rmse | 0.36 (0.36, 0.36) |
| | | | ece | 2.82 (2.56, 3.08) |
| | | spl | rmse | 0.35 (0.34, 0.36) |
| | | | ece | 3.67 (2.78, 4.57) |
| | 0.4 | base | rmse | 0.46 (0.45, 0.46) |
| | | | ece | 5.01 (4.59, 5.43) |
| | | spl | rmse | 0.45 (0.44, 0.45) |
| | | | ece | 6.60 (5.04, 8.16) |
| | 0.5 | base | rmse | 0.56 (0.56, 0.56) |
| | | | ece | 7.53 (6.93, 8.13) |
| | | spl | rmse | 0.55 (0.55, 0.56) |
| | | | ece | 10.00 (7.66, 12.35) |
| | 0.6 | base | rmse | 0.67 (0.66, 0.67) |
| | | | ece | 10.18 (9.37, 10.99) |
| | | spl | rmse | 0.66 (0.65, 0.66) |
| | | | ece | 13.59 (10.41, 16.76) |