Supplementary Material for GP-Tree: A Gaussian Process Classifier for Few-Shot Incremental Learning

A. Variational Bound & Updates

In Section 3.2 we presented the following variational lower bound for the log marginal likelihood at each node:

$$\mathcal{C}(\boldsymbol{c}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) = \mathbb{E}_{p(\boldsymbol{f}|\bar{\boldsymbol{f}})q(\bar{\boldsymbol{f}})q(\boldsymbol{\omega})}[log\ p(\boldsymbol{y}|\boldsymbol{\omega}, \boldsymbol{f})] - KL(q(\bar{\boldsymbol{f}}, \boldsymbol{\omega})\ ||\ p(\bar{\boldsymbol{f}}, \boldsymbol{\omega})).$$

Here, we present the closed-form expression of it and the update rules for the variational parameters $\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}$ and $\boldsymbol{c}$. In the following constants are omitted for conciseness.

A.1. Explicit Form for the Variational Bound

We begin with the expectation term:

$$\begin{aligned}
&\mathbb{E}_{p(\boldsymbol{f}|\bar{\boldsymbol{f}})q(\bar{\boldsymbol{f}})q(\boldsymbol{\omega})}[log\ p(\boldsymbol{y}|\boldsymbol{\omega}, \boldsymbol{f})] \\
&\propto \mathbb{E}_{p(\boldsymbol{f}|\bar{\boldsymbol{f}})q(\bar{\boldsymbol{f}})q(\boldsymbol{\omega})}[(\boldsymbol{y} - \mathbf{1/2})^T \boldsymbol{f} - \frac{1}{2}\boldsymbol{f}^T \boldsymbol{\Omega} \boldsymbol{f}] \\
&= \mathbb{E}_{q(\bar{\boldsymbol{f}})q(\boldsymbol{\omega})}[(\boldsymbol{y} - \mathbf{1/2})^T \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} \bar{\boldsymbol{f}} - \frac{1}{2} Tr(\boldsymbol{\Omega} \boldsymbol{Q}_{nn}) \\
&\quad - \frac{1}{2} \bar{\boldsymbol{f}}^T \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mn} \boldsymbol{\Omega} \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} \bar{\boldsymbol{f}}] \\
&= \frac{1}{2}\{2(\boldsymbol{y} - \mathbf{1/2})^T \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} - Tr(\boldsymbol{\Lambda} \boldsymbol{Q}_{nn}) \\
&\quad - Tr(\boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mn} \boldsymbol{\Lambda} \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} \tilde{\boldsymbol{\Sigma}}) \\
&\quad - \tilde{\boldsymbol{\mu}}^T \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mn} \boldsymbol{\Lambda} \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} \tilde{\boldsymbol{\mu}}\},
\end{aligned}$$

where $\lambda_i = \mathbb{E}_{q(\omega_i)}[\omega_i] = \frac{1}{2c_i} tanh(\frac{c_i}{2})$, $\boldsymbol{\Lambda} = diag(\lambda_i)$.

Now, we move to the KL divergence term. Due to independence between $p(\boldsymbol{\omega})$ and $p(\bar{\boldsymbol{f}})$, and the mean-field as a variational family assumption we have:

$$\begin{aligned}
&KL(q(\bar{\boldsymbol{f}}, \boldsymbol{\omega})\ ||\ p(\bar{\boldsymbol{f}}, \boldsymbol{\omega})) \\
&= KL(q(\bar{\boldsymbol{f}}) q(\boldsymbol{\omega})\ ||\ p(\bar{\boldsymbol{f}}) p(\boldsymbol{\omega})) \\
&= KL(q(\bar{\boldsymbol{f}})\ ||\ p(\bar{\boldsymbol{f}})) + KL(q(\boldsymbol{\omega})\ ||\ p(\boldsymbol{\omega})).
\end{aligned}$$

The first KL term is between two Gaussian distributions and has the following closed-form expression:

$$\begin{aligned}
&KL(q(\bar{\boldsymbol{f}})\ ||\ p(\bar{\boldsymbol{f}})) \\
&= KL(\mathcal{N}(\tilde{\boldsymbol{\mu}},\ \tilde{\boldsymbol{\Sigma}})\ ||\ \mathcal{N}(\mathbf{0},\ \boldsymbol{K}_{mm})) \propto \\
&\frac{1}{2}\{Tr(\boldsymbol{K}_{mm}^{-1} \tilde{\boldsymbol{\Sigma}}) + \tilde{\boldsymbol{\mu}}^T \boldsymbol{K}_{mm}^{-1} \tilde{\boldsymbol{\mu}} - log\ |\tilde{\boldsymbol{\Sigma}}| + log\ |\boldsymbol{K}_{mm}|\}.
\end{aligned}$$

The second KL term is between two Pólya-Gamma (PG) distributions, each of a mutually independent random variable, and has a closed-form expression as well:

$$\begin{aligned}
KL(q(\boldsymbol{\omega})||p(\boldsymbol{\omega})) &= \sum_{i=1}^{n} KL(q(\omega_i)\ ||\ p(\omega_i)) \\
&= \sum_{i=1}^{n} KL(PG(1,\ c_i)\ ||\ PG(1,\ 0)) \\
&= \sum_{i=1}^{n} log\ cosh\frac{c_i}{2} - \frac{c_i}{4} tanh(\frac{c_i}{2}).
\end{aligned}$$

The variational lower bound is obtained by summing all these terms according to Eq. 13.

A.2. Variational Parameters Update

The update rules for the variational parameters are given by taking the derivative of Eq. 13 w.r.t each of $\boldsymbol{c}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}$. At each iteration, based on a mini-batch of samples $\mathcal{B}$, we first update the parameters $\boldsymbol{c}^{\mathcal{B}} \subseteq \boldsymbol{c}$ corresponding to the samples seen in the batch using coordinate ascent scheme while holding $\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}$ fixed. Then, $\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}$ are updated according to a stochastic natural gradient ascent scheme.

The parameters $\boldsymbol{c}$ have a unique maximum which is given in a closed-form:

$$\begin{aligned}
c_i = (Q_{ii} + \boldsymbol{K}_{im} \boldsymbol{K}_{mm}^{-1} \tilde{\boldsymbol{\Sigma}} \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mi} + \\
\tilde{\boldsymbol{\mu}}^T \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mi} \boldsymbol{K}_{im} \boldsymbol{K}_{mm}^{-1} \tilde{\boldsymbol{\mu}})^{\frac{1}{2}},
\end{aligned} \tag{16}$$

where the subscript i denotes a specific row/column corresponding to the i^{th} sample.

For the parameters $\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}$, the natural parameterization of the variational Gaussian distribution can be used: $\boldsymbol{\eta} = \tilde{\boldsymbol{\Sigma}}^{-1} \tilde{\boldsymbol{\mu}}$ and $\boldsymbol{H} = -\frac{1}{2} \tilde{\boldsymbol{\Sigma}}^{-1}$. The update at each batch then becomes:

$$\begin{aligned}
&\tilde{\nabla}_{\boldsymbol{\eta}} \mathcal{C} = \frac{n}{2|\mathcal{B}|} \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mn}^{\mathcal{B}} (\boldsymbol{y}^{\mathcal{B}} - \mathbf{1/2}) - \boldsymbol{\eta}, \\
&\tilde{\nabla}_{\boldsymbol{H}} \mathcal{C} = -\frac{1}{2}(\boldsymbol{K}_{mm}^{-1} + \frac{n}{2|\mathcal{B}|} \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mn}^{\mathcal{B}} \boldsymbol{\Lambda}^{\mathcal{B}} \boldsymbol{K}_{nm}^{\mathcal{B}} \boldsymbol{K}_{mm}^{-1}) - \boldsymbol{H}.
\end{aligned} \tag{17}$$

Where we used the superscript $\mathcal{B}$ to denote only the rows/columns of samples in the batch. Note that the natural gradient updates maintain the positive-definiteness of $\tilde{\boldsymbol{\Sigma}}$.

B. Learning Algorithm with VI

Algorithm 1 summarizes GP-Tree learning with VI and DKL.

Algorithm 1 GP-Tree Inference with VI

Input: Data $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$, I_1 number of training iterations with a NN, I_2 Number of training iterations with GP-Tree
Init: g_θ a NN parameterized by θ
For $i = 1, ..., I_1$:
- Sample a mini-batch of data from $\mathcal{D}$
- Learn g_θ with a classification loss

End for
Build GP-Tree $\mathcal{T}$ as described in Section 3.1
Init: GP hyper-parameters ϕ, variational parameters $\boldsymbol{c}, \boldsymbol{\eta}, \boldsymbol{H}$, and inducing locations $\bar{\boldsymbol{X}}$ in the embedded space
For $i = 1, ..., I_2$:
- Sample a mini-batch of data from $\mathcal{D}$
- Obtain embedding for $\boldsymbol{X}$ with $g_\theta(\boldsymbol{X})$
- Traverse the tree (e.g., via in-order traversal)
  For each node in the path:
    - Update $\boldsymbol{c}$ according to Eq. 16
    - Update $\boldsymbol{\eta}, \boldsymbol{H}$ according to Eq. 17

  End for
- Update θ, ϕ and $\bar{\boldsymbol{X}}$ using Eq. 15 and by replacing the marginal likelihood terms with the variational lower bound $\mathcal{C}(\boldsymbol{c}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}})$ per node

End for
Return $g_\theta, \mathcal{T}, \bar{\boldsymbol{X}}$

C. Experimental Setup

This section provides further details about the experiments shown in Section 5.

C.1. Inference With Gibbs Sampling - Sec. 5.1

Data. We used the pre-trained features extracted by Xian et al. (2018) for the CUB 200-2011 dataset (Welinder et al., 2010). The CUB 200-2011 dataset contains 200 classes of bird species in 11,788 images with approximately 30 examples per class in the training set. Here, since the training set size is limited, we used all 5994 training instances according to the official split and we split the predefined test set to 2897 samples for validation and 2897 for testing.

Hyperparameter tuning. For all baselines, in all experiments, we applied a grid search over the kernel type, either normalized linear kernel or normalized RBF kernel (Snell & Zemel, 2021). We consistently found that under this setting the linear kernel generated better results (this was not true in other settings). The output scale for the linear kernel was chosen based on a grid search in $\{1, 4, 9, 18\}$. We used 20 Gibbs chains for the experiments with $\{4, 6, 8, 10, 20\}$ classes, 10 Gibbs chains for the experiments with 30 classes, and 1 Gibbs chain for the experiments with $\{40, 50, 60\}$ classes. For the OVE baseline, we were able to use only 10 chains for the experiments with 8 classes and 3 chains for the experiments with 10 classes. These experiments were done on an NVIDIA V100 32GB GPU. We applied 1 Gibbs sampling step before taking $\boldsymbol{\omega}$ for the predictive distribution calculations. In these experiments, we often found it useful to make predictions with a single sample at the expected value location instead of using the 1D Gaussian-Hermite quadrature.

C.2. GPC With DKL - Sec. 5.2

In all experiments we trained from scratch a ResNet-18 (He et al., 2016) adjusted for CIFAR images size with a final embedding layer size of 1024. The Batch size was set to 256. We used SGD with a momentum of 0.9 and a scheduler that decays the learning rate by a factor of 0.1 at epochs 100 and 150. We allocated 10% from the training set for validation using stratified sampling. We applied a grid search over the initial learning rate in $\{0.1, 0.01\}$ for all methods. We found that an initial learning rate of 0.01 was preferred for our method. We used natural gradient descent with a learning rate of 0.05 for learning the variational parameters. In GPDNN experiments we also searched for an initial learning rate in $\{0.001, 0.0005\}$ and experimented with the Adam optimizer (Kingma & Ba, 2014). In all methods, we applied pre-training using a NN with a softmax layer after the last embedding layer and the cross-entropy loss. We searched over the number of pre-training epochs in $\{0, 20, 40, 60, 80\}$. For GP-Tree, 80 epochs yielded the best results. We used 40 inducing points per class in both GP-Tree and GPDNN experiments. For the SV-DKL baseline, we experimented with a grid size of $\{64, 128, 256\}$. In all experiments of all methods, we used the RBF kernel over L2 normalized input vectors. In CIFAR-100 experiments of the GPDNN baseline, we also applied an extensive grid search for the probability of labeling error without any success to achieve reasonable accuracy. In GP-Tree experiments, we found it beneficial to assign a weight to the loss term at each node that is inversely proportional to the amount of data used by that node for inference. It is achieved by dividing the loss at each node by the total number of training samples relevant for that node.

C.3. Few-Shot Class-Incremental Learning - Sec. 5.3

Experimental protocol. The experiments in this part largely followed the protocol suggested in (Tao et al., 2020) for comparability. We adopted the 10- way 5-shot setting for CUB, the first 100 classes were set as base classes, the remaining 100 classes were split into 10 incremental sessions. For mini-ImageNet, we followed the 5-way 5-shot, with 60 base classes, and 40 novel classes for a total of nine sessions. We used the official train/test split published by Tao et al. (2020). We pre-allocated a small portion from the training set of the base classes for a validation set. From the CUB dataset, we took 2 samples per class. From the mini-ImageNet dataset, we allocated 5% using stratified sampling.

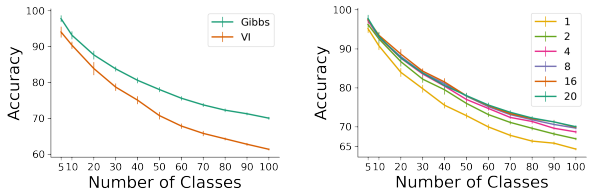
Table 4. Class-incremental few-shot learning on CUB-200-2011. Tree construction variants. Test accuracy averaged over 10 runs.

Method	Sessions										
	1	2	3	4	5	6	7	8	9	10	11
Session Tree	72.84	**67.00**	62.64	57.98	54.28	50.95	48.79	46.65	44.38	42.65	40.87
Rebuild Tree	72.84	65.98	61.76	57.19	53.85	51.04	48.79	47.06	44.61	43.26	41.72
GP-Tree	72.84	**67.00**	**62.98**	**58.19**	**54.84**	**51.77**	**49.40**	**47.57**	**45.47**	**44.05**	**42.72**

In CUB experiments we fine-tuned a pre-trained ResNet-18 on ImageNet while in mini-ImageNet experiments we trained it from scratch. The final embedding layer size was set to 512. The mini-batch size at the first session was set to 128 in all experiments and, in later sessions, it included all available samples. We used SGD with a momentum of 0.9.

Hyperparameter tuning. In the experiments of GP-Tree, SDC (Yu et al., 2020) and PODNet (Douillard et al., 2020), we applied a grid search over the initial learning rate at the first session in $\{0.1, 0.01, 0.001\}$. In CUB experiments it was set to 0.01. In mini-ImageNet experiments, it was set to 0.1. At the first session, we trained the models for 100 epochs with a scheduler that decreased the learning rate by a factor of 0.1 at epochs 40 and 60. At later sessions, for our method, there was nothing to set, as it doesn't require any learning. In SDC and PODNet experiments we trained for 100 epochs and followed the training protocols suggested by each. For SDC, we applied a grid search over the embedding network in $\{LwF, MAS\}$, and γ, the hyper-parameter that controls the trade-off between the metric learning loss and the other losses, in $\{5e-5, 5e-4, 5e-3, 5e-2, 5e-1, 1e4, 1e6\}$. For this baseline, we found it beneficial to start with a few epochs of training using a softmax layer and the cross-entropy loss and only afterward train with the triplet loss as advocated in the paper. When training with the triplet loss, we also needed to optimize the ratio of positive and negative examples at each batch to make it work.

GP-Tree configurations. In GP-Tree experiments, at the initial session, we first applied a few epochs of training using a NN only with a softmax layer and the cross-entropy loss. Then, after 20 epochs in CUB experiments, and 40 epochs in mini-ImageNet experiments, we transitioned to learning with GP-Tree as described in Section 3. We used 5 inducing points per class and the RBF kernel on all GPs with an initial length-scale of 1 and an initial output-scale of 4. We learned the variational parameters with natural gradient descent and a learning rate of 0.05. Here, as well, the inputs to the kernel were normalized by their L2 norm. During training, we assigned a weight to the loss term at each node that is inversely proportional to the amount of data used by that node for inference. During novel sessions, we used the RBF kernel with a fixed length-scale of 1 and a fixed output-scale of 8. At the end of each few-shot session, we saved the 512 dimensional representation of the samples for later sessions.



(a) Gibbs vs VI (b) # of Gibbs Chains

Figure 4. Test accuracy when varying the number of classes. **Left** Gibbs sampling vs variational inference, **Right** varying the number of Gibbs chains. Results are the average over 10 runs (± SEM) on pre-trained features of samples from the CUB 200-2011 dataset.

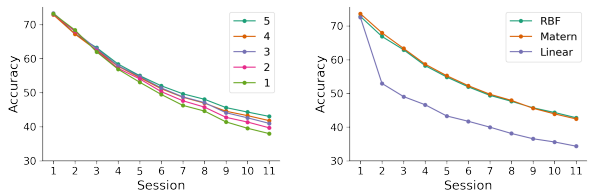
D. Additional Experiments

D.1. GP-Tree Inference

The performance of GP-Tree depends on several factors. Here, we test (1) the effect of using VI against using a Gibbs sampler and, (2) the effect of the number of Gibbs chains. Both analyses were made by observing the test accuracy on the CUB-200-2011 dataset under the setup presented in Section 5.1. The results are shown in Figure 4. Figure 4(a) shows a large performance gap in favor of the Gibbs sampling. This result is not surprising since the Gibbs sampler, asymptotically, samples from the correct posterior while in VI we use an approximate one. The figure also shows that the gap is amplified as the number of classes increases. This result is another justification for using Gibbs sampling when learning novel classes under the incremental learning setup. Figure 4(b) shows that as we increase the number of chains the accuracy increase as well; however, the difference is marginal when using four chains or more.

D.2. Sensitivity Analysis for FSCIL

When we adjusted GP-Tree to the few-shot class-incremental learning setting, we made several design choices. Here, we examine some of them. We will show that GP-Tree is fairly robust to these choices.



(a) # of Rep. Samples (b) Kernel Choice

Figure 5. Test accuracy for few-shot class-incremental learning as a function of the number of representative samples per class (left) and the kernel choice (right). Results are the average over 10 runs on the CUB 200-2011 dataset.

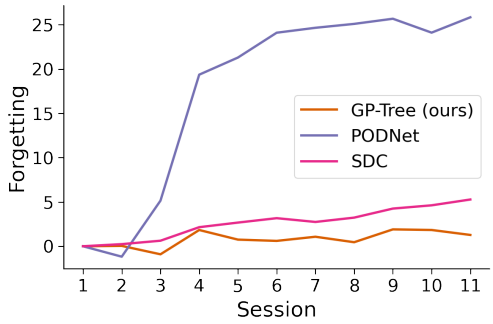


Figure 6. Average forgetting across sessions. Results are the average over 5 runs on the CUB 200-2011 dataset.

Tree construction. You may recall that under the FSCIL setup, after learning on the base classes, we retain the original tree intact, and at each novel session t, we build a sub-tree from samples' representations that appeared in sessions $D_2, ..., D_t$. We then connect this sub-tree to the base classes tree with a shared root node. In Table 4 we present two alternatives for the tree construction used for inference during novel sessions ($t > 1$). (1) *Session Tree*, instead of building a tree from samples in all sessions $D_2, ..., D_t$, in each session we build a tree from classes that appeared at the current session only. Then, we connect this sub-tree with the tree that is already built via a shared root; (2) *Rebuild Tree*, building the entire tree at each new session and fitting the Gibbs sampling variant of our approach to each node. To account for base classes we use the inducing inputs. Table 4 shows that both alternatives yield good results; however, the approach chosen for GP-Tree is superior.

Kernel analysis. The results presented in the main paper for few-shot class-incremental learning were with the RBF kernel and 5 representative samples per class. Here we investigate both choices in Figure 5. Figure 5(a) compares between $1-5$ representative samples per class. The figure shows that all alternatives achieve high accuracy across all sessions; however, as expected, when using less representative samples there is a slight degradation in performance. The impact becomes more severe in later sessions. Figure 5(b) shows a comparison between the RBF, Matern and Linear kernels. Similar to (Snell & Zemel, 2021) we found gain in normalizing the inputs to the kernels by their L2 norm. Therefore, we applied this method in all settings and for all kernel choices. However, unlike in the few-shot learning case presented in (Snell & Zemel, 2021), in which the normalized linear kernel (also referred to as cosine kernel in that study) yielded the best results, in our case either the RBF kernel or the Matern kernel are preferred by a large margin, especially on novel sessions. We hypothesis that on the base classes the NN outputs a representation that is more linearly separable. Therefore, all kernels perform similarly on them. However, the representation of novel classes is more mixed in the embedding space. Therefore, a stronger kernel that generates non-linear decision boundaries is required.

D.3. Forgetting Across Sessions

In this part, we examine how GP-Tree performance varies across sessions through the *average forgetting* (Chaudhry et al., 2018). The average forgetting was designed to estimate the forgetting of prior tasks. Let α_j^k denote the accuracy of the learner on the j^{th} task at session $k > j$. The forgetting of the j^{th} task is defined as $g_j^k = \max_{l \in \{1,...,k-1\}} \alpha_j^l - \alpha_j^k$. This quantity is measured for every task j seen thus far at each new session. Then, to get an estimate of the average forgetting we may use: $\frac{1}{k-1} \sum_{j=1}^{k-1} g_j^k$. Figure 6 shows the average forgetting across all classes and sessions on the CUB 200-2011 dataset for GP-Tree, SDC (Yu et al., 2020), and PODNet (Douillard et al., 2020). From the figure, we notice a minor forgetting for GP-Tree. When comparing the performance of the three methods across all sessions, we noticed that GP-Tree and SDC showed better performance on the base classes while PODNet was more balanced with a slight advantage to the novel classes.