Bayesian Semi Supervised Learning with Function-Space Variational Inference

Aydin Abiar[1] **Annan Deng**[1] **Haoran Wang**[1] **Bogeng Song**[1]

Abstract

Existing work on Semi-Supervised Learning with Variational Bayesian Inference and Maximum Uncertainty Regularization has shown clear improvements in classification errors of various Consistency Regularization based methods. Functional Space Variational Inference is an improvement to the Variational Inference. We propose a method combining Functional Space Variational Inference and Consistency Regularization by minimizing the KL divergence of distributions over functions. We apply our method to the partially labeled datasets and compare the three ways to realize our method.

1. Introduction

The availability of large datasets in recent years has fueled much progress of development of deep neural networks. Semi-supervised learning (SSL) is a popular framework as it deals with the more common case where only some training data is labeled (Van Engelen & Hoos, 2020). Consistency regularization (CR) methods are currently state of the art methods in approaching this kind of tasks. Examples include Π-model (Laine & Aila, 2016) and Mean Teacher (Tarvainen & Valpola, 2017). These ensemble models encourage data points in the same neighborhood to share the same labels, so that they will give the same predictions. Technically, both use deterministic weights with data perturbation to obtain more robust models by induce randomness into training data.

Nevertheless, high computational costs and out of sample errors still haunts these models. Despite much success on specific datasets, they tend to fail on testing data with different structures compared to the training set. An alternative approach is proposed by (Do et al., 2021), which uses virtual points to enhance prediction. These virtual points are generated from the vicinity of real data points that have the most uncertainty in prediction, and their prediction results are compared to the prediction of real data points. Consistency regularization is performed by minimizing this distance. This 'maximum uncertainty regularization (MUR)' model is innovative in proposing a plausible target for optimization. However, the original paper did not incorporate the likelihood of unlabeled data, which forms the majority of training data in SSL tasks.

In this paper, we propose an improvement from a function space perspective. The Bayesian method in this area has been Variational Bayesian Inference (VBI), which places priors on these weights and estimates an approximate distribution by minimizing the KL Divergence. With large datasets, variational dropout is used to compute efficiently. Functional Space Variational Inference (FSVI) is an improvement to the VBI method (Rudner et al., 2021). In parameter space, small bias in parameter estimation can cause large shaking in objective function value. However, in real-life we care more about final prediction than the parameters themselves. Therefore, using distributions to replace the deterministic parameters is very likely to be an appropriate way. By expressing the posterior distribution as a distribution over functions, this allows us to impose more meaningful priors and better control the distribution over functions induced by the network parameters. By implementing FSVI designs on the deterministic CR models, we aim to harvest both the gains from ensembling and the flexibility from FSVI, and see an improvement in prediction.

In Section 2, we introduce the mainstream methods for consistency regularization in SSL networks, which covers the background of the MUR model we improve upon. Section 3 describes our method of implementing function-space variational inference on the MUR model, and discusses three algorithms used in finding the MUR data points. Section 4 outlines the results of our model tested on benchmark datasets. Section 5 discusses the implications of our model and next steps to take.

2. Related Work

In this section we cover traditional CR based semi-supervised methods on parameter space that form the basis of our approach. The main idea of Consistency Regularization is that for an input, even if it is slightly disturbed, its prediction should be consistent. In related work part, we mainly introduce three methods — Π Model, Mean Teacher

[1]New York University, New York City, NY, US. .

Model, and Maximum Uncertainty Regularization.

The first two SSL models both incorporate training stabilizers based on the notion of CR (Zhang et al., 2019). As a popular technique in the SSL literature, CR based methods encourage neighbor samples to share labels by enforcing consistent predictions for inputs under perturbations (Do et al., 2021).

2.1. Π Model

Π-model (Laine & Aila, 2016) encourages consistent network output between two realizations of the same input stimulus, under two different dropout conditions. In other words, it passes all samples through a classifier twice, each time with different dropout, noise and image translation parameters. The core idea for π model is that the dropout method can be regarded as a method of data perturbation (DP). Specificallymultiple passes of an individual sample through the network might lead to different predictions due to the non-deterministic behavior of dropout method (Sajjadi et al., 2016).

During each training iteration a mini-batch of samples is drawn from the dataset, consisting of both labeled and unlabeled samples. As we can see from Figure 1(a), we evaluate the network for each training input x_i twice, resulting in prediction vectors z_i and $\tilde{z}_i$. The unsupervised consistency loss is the average of the square difference of the class probability prediction generated by these two presentations of each input. The training loss of the Π-model in each iteration is given by the weighted sum of cross-entropy loss on labeled samples and the consistency loss on both labeled and unlabeled samples, and the network parameters θ are updated in the meantime. The final loss of a classifier f with deterministic weights θ yields:

$$\begin{aligned}\mathcal{L}_{\Pi}(\theta) =& \mathbb{E}_{(x_l,y_l)\sim\mathcal{D}_l}\left[-\log p\left(y_l \mid x_l, \theta\right)\right] + \\ & \lambda(t)\mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{K}\sum_{k=1}^{K}\left(p(k \mid x, \theta) - p\left(k \mid x', \theta_{\text{sg}}\right)\right)^2\right] \\ =& \mathcal{L}_{\text{xent},l}(\theta) + \lambda(t)\mathcal{L}_{\Pi,\text{cons}}\left(\theta, \theta_{\text{sg}}\right)\end{aligned}$$

where $\mathcal{D}_l, \mathcal{D}_u$ represent disjoint labeled and unlabeled training datasets; $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$; K is the number of classes; $\lambda(t)$ is a "ramp" function which depends on the training step t; θ_{sg} denotes θ with no gradient update; $\mathcal{L}_{\text{xent}}(\theta)$ is the cross-entropy loss on labeled samples and $\mathcal{L}_{\Pi,\text{cons}}(\theta)$ is the consistency loss on all samples.

2.2. Mean Teacher Model

The Mean teacher model is based on π model by adding the idea of temporal ensemble, which simplifies and extends CR by taking into account the network predictions over multiple previous training epochs and encourages subsequent predictions to be consistent with the average (Laine & Aila, 2016). Mean teacher model uses two networks; a student network and a teacher network, where the student is trained using gradient descent and the weights of the teacher are the exponential moving average of those of the student.

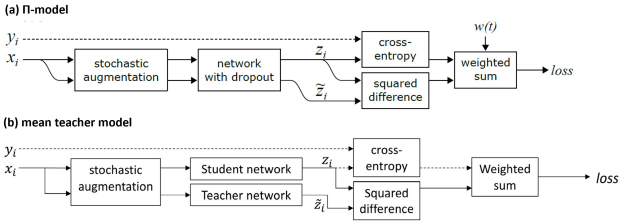

Figure 1. Structure of the training pass in our methods. (a): Π-model. (b): mean teacher. Labels y_i are available only for the labeled inputs, and the associated cross-entropy loss component is evaluated only for those.

Similar to Π model, the training loss of the mean teacher model is also the sum of a supervised and an unsupervised component. In Figure 1(b),z_i means student prediction and $\tilde{z}_i$ means teacher prediction. The unsupervised consistency loss is computed using the mean squared difference between the class probability predictions z_i and $\tilde{z}_i$ for the same input sample x_i. The final loss of a classifier f with deterministic weights θ yields:

$$\begin{aligned}\mathcal{L}_{\text{MT}}(\theta) =& \mathbb{E}_{(x_l,y_l)\sim\mathcal{D}_l}\left[-\log p\left(y_l \mid x_l, \theta\right)\right] + \\ & \lambda(t)\mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{K}\sum_{k=1}^{K}\left(p(k \mid x, \theta) - p\left(k \mid x', \bar{\theta}\right)\right)^2\right] \\ =& \mathcal{L}_{\text{xent},l}(\theta) + \lambda(t)\mathcal{L}_{\text{MT, cons}}\left(\theta, \bar{\theta}\right)\end{aligned}$$

where $\bar{\theta}$ are the exponential moving averages (EMA) of θ across training steps: $\bar{\theta}_t = \alpha\bar{\theta}_{t-1} + (1-\alpha)\theta (\alpha \in [0,1])$

The structure of the Π model and the mean teacher model is shown in Figure 1.

2.3. Maximum Uncertainty Regularization

In the previous two SSL models, the DP of the model is based on the dropout method. However, standard DP methods (e.g., Gaussian noise, dropout) often generate perturbations in the vicinity of each data point and ignore those in the vacancy among data points, which means consistency losses equipped with standard DPs can only train locally smooth classifiers that do not generalize well in general. In order to make the model have better generalization ability, a new loss method is proposed, called Maximum Uncertainty Regularization (MUR). They start by assuming a series of 'virtual' points x^*, which are located near the real data points with

the highest uncertainty and then make the results predicted by the model for the virtual points are the same as the values predicted by the real data points x_0. In this way, they can learn a smoother classifier that generalizes better (Do et al., 2021). In addition, Shannon entropy can be used to measure uncertainty. So, the virtual point x^* can be found using:

$$x^* = \arg\max \mathbf{H}(p(y|x)), s.t. |x - x_0| \leq r$$

which r is the largest distance between x_0 and x^*. The choice of r can affect the performance of the model. If r is too small, it is hard to find an adequate virtual point that the classifier is uncertain about. By contrast, if r is too big, virtual point is very different from real point and forcing consistency between these points may be inappropriate. In terms of model performance, researches have shown that when using the MUR method as a data perturbation method, the model performance has been significantly improved.

In general, it is hard to compute x^*, because the previous equation has lots of local minimum points. However, we can approximate x^* by optimizing a linear approximation of $\mathbf{H}(p(y|x))$ instead. In this case, the original question can covert to solve:

$$x^* \approx x_0 + r\frac{g_0}{|g_0|_2}$$

which g_0 is the gradient of $\mathbf{H}(p(y|x))$ at $x = x_0$.

MUR method is similar to Adversarial Learning (AL) (Szegedy et al., 2013). However, there are still some differences between the two: In MUR, the two sub-problems optimize two distinct objectives (the consistency loss and the conditional entropy) while in AL, the two sub-problems share the same objective. Moreover, since MUR's objectives do not use label information, MUR is applicable to SSL while AL is not.

By reading related research, we found that the current exploration of MUR methods is not mature enough. In addition, current research still stays in the parameter space. In our project, we propose some new methods for detecting virtual points, and compare the model performance between different methods, and we hope to examine the SSL model from the function space.

3. Method

Our model builds upon the Function-Space Variational Inference, which we will describe. Subsequently we will present our modifications that enable function-space MUR.

3.1. Function-Space Variational Inference

All the above-mentioned CR approaches are based on parameter space. However, as mentioned in 1, they all share the problems of defining meaningful priors, so now some research has gradually turned to the function space to fit the data. For functional space variational inference, (Sun et al., 2019) consider a variational objective defined explicitly in terms of distributions over functions induced by distributions over parameters.

Considering supervised learning tasks on data $\mathcal{D} \doteq \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N = (\mathbf{X}_\mathcal{D}, \mathbf{y}_\mathcal{D})$:

On traditional parameter space, we have

- likelihood function $p_{\mathbf{y}|(\mathbf{X};\boldsymbol{\Theta})}$,
- prior $p_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$,
- posterior $p_{\boldsymbol{\Theta}|\mathcal{D}}(\boldsymbol{\theta} \mid \mathcal{D})$.

On functional space,

- letting $p_{\mathbf{y}|f(\mathbf{X};\boldsymbol{\Theta})}$ be a likelihood function, and $p_{\mathbf{y}|f(\mathbf{X};\boldsymbol{\Theta})}(\mathbf{y}_\mathcal{D} \mid f(\mathbf{X}_\mathcal{D};\boldsymbol{\theta}))$ be the likelihood of observing the targets $\mathbf{y}_\mathcal{D}$ under the stochastic function $f(\cdot;\boldsymbol{\Theta})$ evaluated at inputs $\mathbf{X}_\mathcal{D}$.
- The prior distribution over functions $p_{f(\cdot;\boldsymbol{\Theta})}(f(\cdot;\boldsymbol{\theta}))$ induced by a prior distribution over parameters $p_{\boldsymbol{\Theta}}$ is defined as:
  $\int_{\mathbb{R}^P} p_{\boldsymbol{\Theta}}(\boldsymbol{\theta}')\,\delta\left(f(\cdot;\boldsymbol{\theta}) - f(\cdot;\boldsymbol{\theta}')\right)\mathrm{d}\boldsymbol{\theta}'$
- The posterior distribution over functions $p_{f(\cdot;\boldsymbol{\Theta})|\mathcal{D}}(f(\cdot;\boldsymbol{\theta}) \mid \mathcal{D})$ is defined as:
  $\int_{\mathbb{R}^P} p_{\boldsymbol{\Theta}|\mathcal{D}}(\boldsymbol{\theta}' \mid \mathcal{D})\,\delta\left(f(\cdot;\boldsymbol{\theta}) - f(\cdot;\boldsymbol{\theta}')\right)\mathrm{d}\boldsymbol{\theta}'$

where $\delta(f(\cdot))$ is the Dirac delta function.

Like the inference of the parameter space, we need to judge the distribution of $p(f(x)|D)$ but introduce another simple distribution $q(f(x))$ when determining the distribution of $p(f(x)|D)$, so that $q(f(x))$ is as close as possible to $p(f(x)|D)$. Specifically, we need to maximize functional ELBO (fELBO) (Sun et al., 2019). fELBO is defined as:

$$\begin{aligned}&\mathcal{F}(q_{\boldsymbol{\Theta}})\\ &= \mathbb{E}_{q_{f(\mathbf{X}_\mathcal{D};\boldsymbol{\Theta})}}\left[\log p_{\mathbf{y}|f(\mathbf{X};\boldsymbol{\Theta})}(\mathbf{y}_\mathcal{D} \mid f(\mathbf{X}_\mathcal{D};\boldsymbol{\theta}))\right]\\ &\quad - \mathbb{D}_{\mathrm{KL}}\left(q_{f(\cdot;\boldsymbol{\Theta})} \| p_{f(\cdot;\boldsymbol{\Theta})}\right)\end{aligned}$$

To make our object tractable, (Sun et al., 2019) proposed a simple estimator of the Kullback-Leibler divergence between distributions over functions that allows for stochastic variational inference:

$\mathbb{D}_{\mathrm{KL}}\left(q_{f(\cdot;\boldsymbol{\Theta})} \| p_{f(\cdot;\boldsymbol{\Theta})}\right)$ can be expressed as the supremum of the KL divergence from $q_{f(\cdot;\Theta)}$ to $p_{f(\cdot;\Theta)}$ over all finite sets

of evaluation points $\sup_{\mathbf{X}\in\mathcal{X}_{\mathbb{N}}} \mathbb{D}_{\mathrm{KL}}\left(q_f(\mathbf{X};\boldsymbol{\Theta})\|p_f(\mathbf{X};\boldsymbol{\Theta})\right)$, where $\mathcal{X}_{\mathbb{N}} \doteq \bigcup_{n\in\mathbb{N}}\left\{\mathbf{X}\in\mathcal{X}_n||\mathcal{X}_n\subseteq\mathbb{R}^{n\times D}\right\}$ is the collection of all finite sets of evaluation points.

In addition to this, (Rudner et al., 2021) used local linearization to allow for scalable gradient-based optimization of $\mathcal{F}\left(q_{\boldsymbol{\Theta}}\right)$. Then, an estimator of variational objective is obtained by using the Monte Carlo estimator over the induced distributions under the linearized mapping.

Functional Space variational inference method leads to state-of-the-art uncertainty estimation and predictive performance on a range of prediction tasks (Rudner et al., 2021)

3.2. Function-Space MUR

In order to improve the generalization ability of our model. We often need data perturbation methods to train models. We find 'virtual' points, while here they are unlabeled points. These "virtual" points usually lie beyond the local area of real data points and prevent a smooth transition of the class prediction from a data point to another. We hope that the predicted value obtained by these virtual points is the same as the predicted value of the real data point, which can improve the generalization ability of the model. Specifically, we use the hope to get:

$$softmax(q(f(x_*))) = softmax(p(f(x_0)))$$

Where x_* is defined as the:

$$x^* = \underset{x}{\operatorname{argmax}} H(p(y \mid x)) \text{ s.t. } \|x - x_0\|_2 \leq r$$

And $q(f(x))$ and $p(f(x))$ is the probability distribution for x_*'s function and x_0's function on function space. We also want the two distributions to be closed with each other, specifically, We need the q distribution to gradually approach the p distribution, and we use the KL divergence $D_{KL}\left(q(f(x))\|p(f(x))\right)$ to define this difference.

For experiments that involve uncertainty quantification, we have to choose a prior distribution over parameters that induces a prior distribution over functions $p_{f(\cdot;\boldsymbol{\Theta})}$. There exist some ideas about how to choose the prior distribution. First, we choose the Gaussian distribution as the prior distribution, but we have different considerations for the Gaussian mean. First, we can calculate the mean, i.e. $f(x_0)$, from the existing data. Another mean is to use mean of $f(x_*)$. We will also continue to explore the possibility of Gaussian distribution priors, and try to change the parameters to better fit our data, and try to get the relationship between the definition of the prior distribution and our results. Previous studies also use prior to mimic GP prior distribution (Flam-Shepherd et al., 2017) or using Noise Contrastive Priors (Hafner et al., 2018). For the sake of time limit, we do not consider them in this study.

3.3. Most Uncertain Unlabeled Data Selection

We assume the prior $p(f(.))$ as a Gaussian distribution of mean μ_p and covariance Σ_p. With this assumption, the KL divergence $D_{KL}\left(q(f(x))\|p(f(x))\right.$ becomes a KL Divergence between 2 gaussians $p(f(.)) \sim \mathcal{N}(\mu_p, \Sigma_p)$ and $q_\theta(f(.)) \sim \mathcal{N}(\mu_{q\theta}, \Sigma_{q\theta})$ which simplifies as an affine function of $\|\mu_{q\theta} - \mu_p\|^2$

$$D_{KL}\left(q(f(x))\|p(f(x)-\right) = Cst_1 + Cst_2 * ||\mu_{q\theta} - \mu_p||^2$$

With Cst_1 and Cst_2 function of the other parameters of the gaussians

This shows that optimizing the KL Divergence will minimize the difference $\|\mu_{q\theta} - \mu_p\|^2$ and with a smart choice of μ_p, we can try to imitate the behaviour of the MUR Loss.

Choosing μ_p as x_* is an approximated way to make the KL Divergence of the Function Space Loss behave like the MUR Loss in parameter space. The method now relies in the choice of computation of this x_*

3.3.1. Greedy Computation

A natural way to find x_* is to greedily search for it in the unlabeled dataset. For each data x, we look through each unlabeled data and compute their entropy until we find the most uncertain one in a close area.

Algorithm 1 Greedy x* search

$\mathbf{D_{nl}} \leftarrow$ Non labeled Dataset
$x \leftarrow$ labeled data point
$H_{max} \leftarrow -\infty$
$x_* \leftarrow None$
for $x_{nl} \in \mathbf{D}_{nl}$ to do begin$f(x_{nl}) \leftarrow model(x_{nl})$
  $H \leftarrow Compute\_Entropy(f(x_{nl}))$
  if $H \geq H_{max}$ **then**
    $H_{max} \leftarrow H$
    $x_* \leftarrow x_{nl}$
  end if
  return x_*

In practice the greedy algorithm is very slow but accurate. If the hardware is optimized, it is preferred to opt for the greedy algorithm. Details about the algorithm's usability is discussed in the Experimental result section.

3.3.2. 1st-Order Approximation

Since we define x_* with an argmax of the entropy function, a natural approximation is to follow the first order expansion of the entropy (Huber et al., 2008) around the data point we consider. This approximation is the one followed by

the original paper. While not accurate for more complex problems, it leverages a very fast computation compared to the workload of a greedy search

$$x_* = x + \lambda \frac{\nabla f(x)}{||\nabla f(x)||}$$

With $\lambda < r$ an hyperparameter

Algorithm 2 1st-Order Approximation

$x \leftarrow$ labeled data point
$f(x) \leftarrow model(x)$
$\nabla f(x) \leftarrow compute_grad_entropy(f(x))$
$x_* \leftarrow x + \lambda \frac{\nabla f(x)}{||\nabla f(x)||}$
return x_*

In this method, we need to determine the size of the r value, we first choose $r = 7$, because according to the previous research when using $r = 7$, the error of the model result is the smallest (Do et al., 2021). In addition, we want to compare the model performance when we choose different value of r, specifically, we choose the $r = 4, 7, 10, 20, 40$.

3.3.3. K-NN APPROXIMATION

Using a k-nearest neighbors (K-NN) approach (Guo et al., 2003), we can balance between the accuracy/efficiency of a greedy approach and the computational speed of the 1st-order approximation.

Using a specific data structure such as a KDTree (Moore, 1991) from the sklearn library, we can store the K nearest neighbour for each of the data point of the dataset.Therefore, we can approximate x_* by comparing the entropy of these K neighbours only.

The K parameter of the algorithm can be tuned to balance between accuracy and speed. With $K \lim size(\mathbf{D}$, we get back the greedy search algorithm

Algorithm 3 K-NN Approximation

$x \leftarrow$ labeled data point
$f(x) \leftarrow model(x)$
$\mathbf{D} \leftarrow Dataset$
$T \leftarrow build_K DTree(\mathbf{D})$
$H_{max} \leftarrow -\infty$
$x_* \leftarrow None$
for x' $\in$ $T(x).children$ to do begin$f(x') \leftarrow model(x')$
$H \leftarrow Compute\_Entropy(f(x'))$
if $H \geq H_{max}$ **then**
$H_{max} \leftarrow H$
$x_* \leftarrow x'$
end if
return x_*

4. Experiments

Our implementation was developed from a template using Pytorch and JAX. We then translated everything in Pytorch. Because the translation takes a while we first experiment with the template code and then the fully pytorch code.

Details about the datasets, data preprocessing scheme, the classifier's architecture and settings, and the training hyperparameters are all the same than the original paper. The objective of the experiment is to find priors that would behave the same way the MUR loss behaves and compare the three uncertain data selection methods.

4.1. Experiment Datasets

We evaluate our approaches on two standard benchmark datasets: *SVHN* and *CIFAR-10*.

4.1.1. SVHN

SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be seen as similar in flavor to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images.

The dataset contains 10 classes, 1 for each digit. There are 73,257 digits for training, 26,032 digits for testing, and 531,131 additional, somewhat less difficult samples, to use as extra training data.The cropped images are centered in the digit of interest, but nearby digits and other distractors are kept in the image. For the purpose of these experiments, we will use the 32x32 image format since it is easier to deal than the original format

4.1.2. CIFAR-10

The CIFAR-10 dataset is a set of labeled images used for object recognition. It consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. The 10 classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset is split into 50,000 training images and 10,000 test images. It was developed by the Canadian Institute for Advanced Research (CIFAR). It is widely used for machine learning and computer vision applications.

The CIFAR-10 dataset is a great choice for object recognition tasks as it is relatively small, yet contains a diverse set of images. The images also have a uniform size, which makes it easier to work with and process. The CIFAR-10 dataset is one of the most widely used datasets in the world and for

	Greedy	1st-Order	K-NN
Accuracy 25% epochs	38 ± 5.3	45 ± 5.5	44 ± 3.5
Accuracy 50% epochs	53 ± 2.3	59 ± 4.7	60 ± 2.0
Accuracy 75% epochs	57 ± 2.2	58 ± 5.0	61 ± 1.5
Accuracy 100% epochs	57 ± 2.1	58 ± 5.0	61 ± 1.5

Table 1. Accuracy (%) over epochs for all 3 methods. Measured on 5 different runs

this reason can be considered as a reasonable benchmark for the experiments.

4.2. Evaluation Measures

We would like to compare the classification errors of each method on SVHN and CIFAR-10. Each setting of our models would be run 5 times.

For the kNN model, we fixed the "k" to be 8.

4.3. Experiment Results

In Table 1, we report the accuracies over epochs of the different x_* approximation methods. We observe that contrary to our prior belief, greedy algorithm is not doing the best performance even though it gives the most accurate x_* since it sweeps all over the data one by one. The finding is supported by previous studies (Wilt & Ruml, 2014). K-NN gives better results with higher accuracy and lower variance compared to both other methods. The 1st-Order approximation seems to approach the greedy performance but with much higher variance.

It is important to note that out of all different runs, the 1st-Order achieved both best and worst accuracy, denoting its very high variance in performance. K-NN seems the most reliable methods of all three in both performance and computational speed if built with the proper data structures.

Figure 2 shows different images given the approximation methods. We can observe that for the same input image, very different x_* are chosen by each. In our human perception, greedy algorithms seems to give the most reasonable "closest" image in the sense of minimizing the Euclidian distance of the images. On the other hand, the 1-st order gives completely unrelated picture, in both shape, label and color. Since the approximation doesn't return a direct image of the unlabeled dataset, we had to project the approximation on the dataset just for the sake of this figure comparison. K-NN however also seems to give a reasonable "close" image, retaining the same label, shape and more or less the same color distribution.

In the end, we can observe from the experiments that even though sometimes the 1st-Order doesn't seem to find meaningful images, it still achieves performance relatable to the greedy algorithm. The K-NN algorithm however seems to be a good balance between the brute-force performance of the greedy algorithm and the computational speed of the 1st-Order approximation.

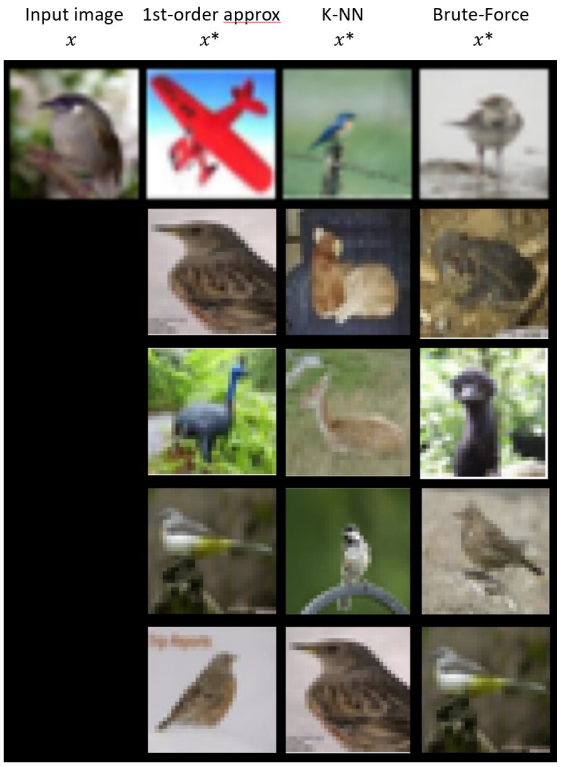

Figure 2. Different x_* approximation for different methods given one input image

5. Discussion

We have proposed an effective approach to function-space variational inference in SSLs. It innovatively incorporated FSVI in MUR and applied on real-life datasets with three different realization methods.

To further elaborate this study, we will implement 5 additional runs of the experiment to achieve a more stable and accurate result; Currently we used a fixed "k" value for our kNN method, in the future we plan to treat "k" as a hyperparameter and fine-tune the "k" in each set of experiment; Also, although we mentioned SVHN dataset in our presentation, we did no have enough time to run experiments on it. Should time allows, we hope to have a try on SVHN and see if it validates our current conclusion; In addition, we also plan to change the value of r, but due to the increase in the amount of calculation, we cannot get the model performance of different r values in time, but we think that the results obtained by changing the value of r should be similar to the results obtained in previous studies . Finally, we used

ResNet architecture in our Python program but we would like to try smaller neural network architecture in the future. It would be exciting to see if there is a smaller NN taking shorter time to run but still yielding good accuracy.

For SSL, current models in parameter space have developed rapidly. We covered the π model and mean teacher model in detail. Both of these approaches focus on adding noise to the data to improve model smoothness for better model performance. In addition, some current methods can also optimize the model from the perspective of data training. For example, the method of Virtual Adversarial Training (VAT) is to selectively pick noise for training by finding the weak points of the model network (Miyato et al., 2018). In addition, the idea of Interpolation Consistency Training (ICT) is to assume that if two points are similar, the output corresponding to any point between the two points should also be similar (Verma et al., 2019). This method can effectively reduce the calculation amount of the model. In addition, we have some Proxy-label Methods (Shen et al., 2019), including self-training. That means that given a data set, use the labeled data to train the network, and then let the network predict the unlabeled data, take the most confident data and prediction, integrate with the original labeled data as a new training set, and then train the network, and repeat this. These methods can be well combined with current data perturbation methods and further improve model performance. Our method is the first to investigate semi-supervised learning in function space, so our method does not integrate some other methods of model optimization. We think that in the future, we can gradually combine the optimization method of this model with our regularization method in the function space to further improve the ability of the model.

In our paper, we did not compare our model with previous results of MUR in parameter space. We cannot guarantee that models in function space can outperform models in parameter space, but previous research has also shown that function space has performed well in neural networks (Rudner et al., 2020; Sun et al., 2019). However, SSL is different from the previous tasks. We will prepare to use the parameter space model (mean teacher model or MUR model) to train and compare the accuracy with our model.

We admit that, due to time limitation, we did not fully optimize our model, but the existed results already have good indications. We hope that this work will lead to further research on function-space variational inference and development of more appropriate data-driven prior distributions on functions.

References

Do, K., Tran, T., and Venkatesh, S. Semi-supervised learning with variational bayesian inference and maximum uncertainty regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7236–7244, 2021.

Flam-Shepherd, D., Requeima, J., and Duvenaud, D. Mapping gaussian process priors to bayesian neural networks. In *NIPS Bayesian deep learning workshop*, volume 3, 2017.

Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 986–996. Springer, 2003.

Hafner, D., Tran, D., Irpan, A., Lillicrap, T., and Davidson, J. Reliable uncertainty estimates in deep neural networks using noise contrastive priors. *stat*, 1050:24, 2018.

Huber, M. F., Bailey, T., Durrant-Whyte, H., and Hanebeck, U. D. On entropy approximation for gaussian mixture random vectors. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 181–188. IEEE, 2008.

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Moore, A. W. An intoductory tutorial on kd-trees. 1991.

Rudner, T. G., Chen, Z., and Gal, Y. Rethinking function-space variational inference in bayesian neural networks. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

Rudner, T. G., Chen, Z., Teh, Y. W., and Gal, Y. Tractable function-space variational inference in bayesian neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Sajjadi, M., Javanmardi, M., and Tasdizen, T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, 29, 2016.

Shen, T., Gong, D., Zhang, W., Shen, C., and Mei, T. Regularizing proxies with multi-adversarial training for unsupervised domain-adaptive semantic segmentation. *arXiv preprint arXiv:1907.12282*, 2019.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

Van Engelen, J. E. and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

Verma, V., Kawaguchi, K., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. Interpolation consistency training for semi-supervised learning. *arXiv preprint arXiv:1903.03825*, 2019.

Wilt, C. M. and Ruml, W. Speedy versus greedy search. In *Seventh Annual Symposium on Combinatorial Search*, 2014.

Zhang, H., Zhang, Z., Odena, A., and Lee, H. Consistency regularization for generative adversarial networks. *arXiv preprint arXiv:1910.12027*, 2019.

A. Progress since midway report

We researched on consistency regularization theory and developed function-space variational inference code before midway report.

Since midway report, we figured out how to incorporate function-space variational inference in Maximum Uncertainty Regularization and proposed a doable prior searching approach with gaussian assumptions. We found three methods to find the Maximum Uncertainty point and implemented experiments to visualize their performance.