Probabilistic Neighbourhood Component Analysis: Sample Efficient Uncertainty Estimation in Deep Learning

Ankur Mallick\(^1\), Chaitanya Dwivedi\(^1\), Bhavya Kailkhura\(^2\), Gauri Joshi\(^1\), and T. Yong-Jin Han\(^2\)

\(^1\)Carnegie Mellon University
\(^2\)Lawrence Livermore National Laboratory

July 22, 2020

Abstract

While Deep Neural Networks (DNNs) achieve state-of-the-art accuracy in various applications, they often fall short in accurately estimating their predictive uncertainty and, in turn, fail to recognize when these predictions may be wrong. Several uncertainty-aware models, such as Bayesian Neural Network (BNNs) and Deep Ensembles have been proposed in the literature for quantifying predictive uncertainty. However, research in this area has been largely confined to the big data regime. In this work, we show that the uncertainty estimation capability of state-of-the-art BNNs and Deep Ensemble models degrades significantly when the amount of training data is small. To address the issue of accurate uncertainty estimation in the small-data regime, we propose a probabilistic generalization of the popular sample-efficient non-parametric kNN approach. Our approach enables deep kNN classifier to accurately quantify underlying uncertainties in its prediction. We demonstrate the usefulness of the proposed approach by achieving superior uncertainty quantification as compared to state-of-the-art on a real-world application of COVID-19 diagnosis from chest X-Rays. Our code is available at [https://github.com/ankurmallick/sample-efficient-uq](https://github.com/ankurmallick/sample-efficient-uq).

1 Introduction

Deep Neural Networks (DNNs) have achieved remarkable success in a wide range of applications where a large amount of labeled training data is available \[10, 9\]. However, in many emerging applications of machine learning such as diagnosis and treatment of novel coronavirus disease (COVID-19) \[6\] a large labeled training datasets may not be available. Furthermore, test data in these applications may deviate from the training data distribution, e.g., due to sample selection bias, nonstationarity, and even can be from Out-of-Distribution in some extreme cases \[3\]. Note that several of these applications are high-regret in nature implying that incorrect decisions or predictions have significant costs. Therefore, such applications require not only achieving high accuracy but also accurate quantification of predictive uncertainties. Accurate predictive uncertainty in these applications can help practitioners to assess the true performance and risks and to decide whether the model predictions should (or should not) be trusted.

Unfortunately, DNNs often make overconfident predictions in the presence of distributional shifts and Out-of-Distribution data. As an example, Fig. 1 shows the predictions of different deep learning models trained to detect the presence of COVID-19 from chest X-ray images. All models achieve similar accuracy (\(~80\%\)) on in-distribution validation data. However, their quality of uncertainties is widely varied as explained next. While all models are forced to output some prediction, on every input image, we would want a model to not be very confident on input data that is very different from the data used to train it. However, we observe that state-of-the-art deep learning models make highly overconfident predictions on Out-of-Distribution data \[17\].
Interestingly, we found that even popular uncertainty-aware models, (e.g., BNNs, deep ensembles) that are designed to address this precise issue, perform poorly in small data regime. This is an extremely problematic issue especially owing to the flurry of papers that have been attempting to use DNNs for detecting COVID-19 using chest X-Ray images [15, 5, 21] as real-world test data is almost always different as compared to the training data.

While there have been separate efforts on improving the sample efficiency [14] and accurate uncertainty estimation [7] of deep learning, to the best of our knowledge there has not been any effort on studying these seemingly different issues in a unified manner. Therefore, this paper takes some initial steps towards (a) studying the effect of training data on the quality of uncertainty and (b) developing sample efficient uncertainty-aware predictive models. Specifically, to overcome the challenge of providing accurate uncertainties without compromising the accuracy in the small-data regime, we propose a probabilistic generalization of the popular non-parametric kNN approach (referred to as probabilistic neighborhood component analysis (PNCA)). By mapping data into distributions in a latent space before performing classification, we enable a deep kNN classifier to accurately quantify underlying uncertainties in its prediction. Following [11, 18], for a meaningful and effective performance evaluation, we compare the quality of predictive uncertainty of different models under conditions of distributional shift and Out-of-Distribution. We empirically show that the proposed PNCA approach achieves significantly better uncertainty estimation performance as compared to state-of-the-art approaches in small data regime.

2 Probabilistic Neighbourhood Component Analysis

In this section, we describe our model to achieve sample-efficient and uncertainty-aware classification. The details of the algorithm and proof of Proposition 1 are presented in Appendix [13].
2.1 Neighbourhood Components Analysis (NCA)

Our approach is a generalization of NCA proposed in [8] wherein the authors learn a distance metric for kNN classification of points $x_1, \ldots, x_n \in \mathbb{R}^d$ with corresponding class labels $y_1, \ldots, y_n$. A data point $x$ is projected into a latent space $Z \subseteq \mathbb{R}^d$ to give an embedding $z = g_w(x)$. Here $g$ can be a linear transformation like a $d \times D$ matrix or a non-linear transformation like a neural network with a $d$-dimensional output, and $w$ are the parameters of the transformation. The probability of a point $x_i$ selecting another point $x_j$ as its neighbor is given by applying a softmax activation to the distance between points in the latent space

$$q_{ij} = \frac{\exp(-||z_i - z_j||^2)}{\sum_{i' \neq i} \exp(-||z_i - z_{i'}||^2)}, \quad q_{ii} = 0. \quad (1)$$

The probability of $x_i$ selecting a point in the same class as itself is given by $q_i = \sum_{j: y_i = y_j} q_{ij}$ and the optimal model parameters are obtained by minimizing the loss

$$L(w) = -\sum_i \log(q_i), \quad (2)$$

which is the negative log-likelihood of the data under our model. The authors of [8] experiment with a variety of transformations $g_w(.)$ and classification tasks and show that NCA achieves competitive accuracy.

2.2 Our Model

The lack of data may cause the NCA model to overfit when learning the weights by optimizing the loss in (2). We expect that the uncertainty due to the scarcity of training data can be better captured by probability distributions in the latent space $Z$ than by individual data samples. Therefore, we propose a probabilistic generalization of the model, PNCA, which learns a distribution over the model parameters $w$ and, thus, deals with both the lack of training data and the task of accurate uncertainty estimation.

Latent Space Mapping using Probabilistic Neural Networks. Each data point $x$ passes through a probabilistic neural network with parameters $W \sim p(W)$ to give a random variable $Z = g_W(x) \in Z$. Due to the stochasticity of $W$, each data point $x_i$ corresponds to a different distribution $p(Z|x_i)$ in the latent space.

NCA over Latent Distributions. Observe that the individual terms in the softmax activation correspond to a kernel between latent embeddings $z$, e.g., the squared exponential kernel $k(z, z') = \exp(-||z - z'||^2)$. Since in our approach the embedding corresponding to a data point $x_i$ is the probability distribution $p(Z|x_i)$, we propose to use the following kernel between distributions

$$K_{ij} = \mathbb{E}_{z \sim p(Z|x_i), z' \sim p(Z|x_j)}[k(z, z')], \quad (3)$$

where $K_{ij}$ corresponds to the inner product between distributions $p(Z|x_i)$ and $p(Z|x_j)$ in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ defined by the kernel $k[10]$ and, thus, captures similarity between distributions in the same way as $k(z, z')$ captures the similarity between individual embeddings in NCA.

2.3 Training Algorithm

The forward pass described above, is used to compute a kernel between data points $x_i$ and $x_j$ which can then be used to compute the probability of $x_i$ selecting $x_j$ in an analogous fashion to NCA as

$$q_{ij} = \frac{K_{ij}}{\sum_{i' \neq i} K_{ii'}}, \quad q_{ii} = 0. \quad (4)$$

Since the latent embedding $Z$ for a data point $x$ is given by $Z = g_W(x)$, $W \sim p(W)$, we can rewrite (3) as

$$K_{ij} = \mathbb{E}_{w, w' \sim p(W)}[k(g_w(x_i), g_{w'}(x_j))] = K_{ij}[p]. \quad (5)$$
Thus, we can view $K_{ij}$ as a functional of $p(W)$. The negative log likelihood in (2) is then also a functional of $p$. The optimal distribution over the model parameters can be obtained by solving

$$p^*(W) = \arg \min_{p(W) \in \mathcal{P}} L[p]. \quad (6)$$

The choice of $\mathcal{P}$ is critical to the success of this approach. Following [13], we choose $\mathcal{P}$ to be $\mathcal{P} = \{p(u)|u = w + s(w), w \sim p_0(w), s \in \mathcal{H}_k\}$ where $\mathcal{H}_k$ is a RKHS given by a kernel $\kappa$ between model parameters $w$ (note that this is different from the RKHS $\mathcal{H}_k$ into which distributions in the latent space $Z$ are embedded and which is given by the kernel $k$). This choice of $\mathcal{P}$ includes all smooth transformations from the initial distribution $p_0$, and the optimization problem (6) now reduces to computing the optimal shift $s^*(w)$. Next, we provide an expression for the functional gradient of the negative log-likelihood under our model with respect to the shift $s$.

**Proposition 1.** If we draw $m$ realizations of model parameters $w_1, \ldots, w_m \sim p(w)$, $p \in \mathcal{P}$, then

$$\nabla_s L \big|_{s=0} \simeq \sum_{l=1}^{m} \kappa(w_l, \cdot) \nabla_{w_l} \hat{L}(w_1, \ldots, w_m) \quad (7)$$

where $\hat{L}$ is given by substituting $\hat{K}_{ij} = \frac{1}{m^2} \sum_{l, l'} k(g_{w_l}(x_i), g_{w_{l'}}(x_j))$ in (2).

To estimate the optimal shift, $s^*$ (or optimal distribution $p^*$) we draw an initial set of parameters $w_i \sim p_0(w)$ and iteratively apply the functional gradient descent transformation $u = w - \epsilon \nabla_s L \big|_{s=0}$ as described in Algorithm I in Appendix B.

### 3 Experiments

We consider two small data classification tasks: (1) handwritten digit recognition, and (2) COVID-19 detection from chest X-Ray images. For both tasks, we compare proposed PNCA to 4 baselines, a Deep Neural Network (DNN), a Bayesian Neural Network (BNN) trained using the approach of [13], Deep Ensembles [11], and NCA [8]. For PNCA and NCA, we use a neural network to map the data $x$ to embeddings $z$. For NCA and PNCA, the predicted class label for a test point $x_i$ is $\hat{y}_i = \arg \max_c \sum_j y_{ij} = c g_{ij}$. For the other models, the predicted class label is the one with the highest softmax probability (average softmax probability for BNNs and Ensembles). Following [13], we use $\max_c q(\hat{y} = c|x)$ as a measure of the confidence of the model.
in predicting class $\hat{y}$ for input $x$ and show the accuracy and number of examples vs. confidence for Out-of-Distribution data to quantify the quality of uncertainties. Please refer to Appendix A for further details on the experiments and additional results.

**MNIST Classification** All neural networks in this experiment have the same architecture (2 hidden layers, 200 nodes per layer). Models are trained on a random subset of 100 labeled examples from the MNIST dataset [12] and results are averaged over 10 trials. Figures 2a and 2b show the performance comparison on unseen rotated MNIST dataset (MNIST test images rotated by $60^\circ$). It can be seen that PNCA outperforms all other models in terms of (a) accuracy vs. confidence (high confidence examples should have high accuracy) and (b) number of examples with high confidence (only a few examples should have high confidence). Moreover, Fig. 2c shows the performance comparison on Out-of-Distribution, i.e., not-MNIST dataset [2] that contains letters instead of handwritten digits. We see that PNCA has significantly fewer examples with high confidence as compared to other approaches on the not-MNIST dataset illustrating the superior capability in quantifying uncertainty.

**COVID-19 detection** There has been an increasing interest in using deep learning to detect COVID-19 from Chest X-Ray (CXR) images [19, 21]. Successful prediction from CXR data can effectively complement the standard RT-PCR test [22]. However, the lack of large amount of training data and distributional shift between train and test data are two major challenges in this task [15].

We consider two sources of COVID-19 data – [6], which has been used by most existing works to train their models for COVID-19 classification and [4], which we use as our unseen test data as it comes from a different source than the images in [6]. We follow the transfer learning approach of [15] wherein a ResNet-50 model pre-trained on Imagenet is used as a feature extractor and the last layer of the model is re-trained on [6] with aforementioned approaches – DNN, BNN, Ensemble, NCA, PNCA on the training dataset. We consider a binary classification problem, i.e., each model outputs a probability $q$ of the presence/absence of COVID-19 in a given CXR image.

We use the version of [6] available on Kaggle [3] as our training dataset, which contains 275 COVID-19 X-Ray images and 76 non-COVID X-Ray images. On the other hand, [4] is used as our test data, which contains 58 COVID-19 X-Ray images and 127 non-COVID X-Ray images. There is a distributional shift present between train and test data resulting in relatively low test accuracy for all models in Fig. 3a. We also look at the number of examples classified with a high confidence for both the test data and completely Out-of-Distribution data (shoulder and hand X-Rays from [17]). As can be seen, on [4], which potentially has a different distribution, BNN has slightly lower number of examples classified with high confidence than the other models. Next, in Fig. 3c we can see that as the distribution shift increases, PNCA makes significantly fewer high confidence predictions than all other models corroborating its superior uncertainty quantification.

---

1https://www.kaggle.com/bachrr/covid-chest-xray
In summary, these experiments demonstrate that PNCA achieves much better uncertainty quantification than the baselines without losing accuracy in small-data regime.

4 Conclusion and Broader Impact

This work serves as a caution to practitioners interested in applying deep learning for disease detection especially during the current pandemic since we find that the issues related to overconfident and inaccurate predictions of DNNs become even more severe in small-data regime. While our approach appears to be less susceptible to making overconfident misclassifications and have good uncertainty estimation performance, we acknowledge that there is still room for improvement especially with respect to the accuracy of the model. With this in mind, we will explore approaches to improve the generalization capability of PNCA in future work. Further, sample efficient uncertainty calibration approaches such as [23] and more reliable evaluation approaches for small data regime can be explored.

Acknowledgement

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-811603).

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pages 265–283, 2016.

[2] Y. Bulatov. not-mnist dataset. http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html, 2011.

[3] S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, and D. Song. Anomalous instance detection in deep learning: A survey. arXiv preprint arXiv:2003.06979, 2020.

[4] A. Chung. Actualmed covid-19 data. https://github.com/agchung/Actualmed-COVID-chestxray-dataset, 2020.

[5] J. P. Cohen, L. Dao, P. Morrison, K. Roth, Y. Bengio, B. Shen, A. Abbasi, M. Hoshmand-Kochi, M. Ghassemi, H. Li, et al. Predicting covid-19 pneumonia severity on chest x-ray with deep learning. arXiv preprint arXiv:2005.11856, 2020.

[6] J. P. Cohen, P. Morrison, and L. Dao. Covid-19 image data collection. arXiv preprint arXiv:2003.11597, 2020.

[7] Y. Gal. Uncertainty in deep learning. University of Cambridge, 2016.

[8] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In Advances in neural information processing systems, pages 513–520, 2005.

[9] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 6645–6649. IEEE, 2013.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
A Additional Details on Experiments

All models are implemented in TensorFlow on a Titan X GPU with 3072 CUDA cores. We use the Adam Optimizer with Nesterov Momentum with a learning rate of 0.001 to train the models for 100 epochs. For DNN, BNN, and Ensemble we use minibatches of size 20 with 1 epoch corresponding to a pass over the entire dataset, while for NCA and DPNCA, the entire dataset is used to calculate gradients.

Following [13], we use the RBF kernel as the kernel $\kappa$ between model parameters $\mathbf{w}$ in PNCA, with bandwidth chosen according to the median heuristic described in their work since it causes $\sum_j \kappa(\mathbf{w}, \mathbf{w}_j) \simeq 1$ for all $\mathbf{w}$, leading $\kappa$ to behave like a probability distribution. We also use Orthogonal Random Features to approximate the kernel between probability distributions in the latent space in [5] for faster computation. We use $10d$ features where $d$ is the dimensionality of the latent space and a ReLU activation on the approximate kernel to set any spurious negative values to zero (since the original squared exponential kernel can never be negative).
Table 1 contains the accuracy of different models across experiments (MNIST test data [12], Rotated MNIST test data, COVID-19 validation data [6] and COVID-19 test data [4]). For MNIST, accuracy values are averaged over 10 trials (where each trial corresponds to a different set of 100 training examples). For COVID-19 accuracy values, the training data is split into 5 equal folds and in each trial we use 4 folds to train the model and the 5th fold to calculate in-distribution (validation) accuracy. Since each training data point is a part of the validation data (5th fold) only once, therefore we do not have any standard deviation values for the validation accuracy. The accuracy on COVID-19 test data is averaged across all folds.

| Method     | MNIST Test | Rotated MNIST | COVID-19 Validation | COVID-19 Test  |
|------------|------------|---------------|---------------------|----------------|
| BNN        | 0.74 ± 0.02| 0.20 ± 0.04   | 0.79                | 0.40 ± 0.04    |
| NCA        | 0.69 ± 0.01| 0.18 ± 0.04   | 0.77                | 0.44 ± 0.05    |
| DNN        | 0.75 ± 0.02| 0.17 ± 0.02   | 0.80                | 0.37 ± 0.02    |
| PNCA       | 0.67 ± 0.03| 0.18 ± 0.03   | 0.82                | 0.37 ± 0.03    |
| Ensemble   | 0.76 ± 0.02| 0.17 ± 0.03   | 0.79                | 0.38 ± 0.03    |

Table 1: Accuracy for all Models across experiments

### B Proof of Proposition 1

Observe that for any smooth one-to-one transform \( u = T(w), \ w \sim p(W) \), the kernel between the latent distributions \( p(Z|x_i), p(Z|x_j) \) corresponding to data points \( x_i \) and \( x_j \) under the transformed distribution \( p[T](u) \) can be written as

\[
K_{ij} = E_{u, u' \sim p[T](u)}[k(g_u(x_i), g_u'(x_j))] \\
= E_{w, w' \sim p(W)}[k(g_{T(w)}(x_i), g_{T(w')}(x_j))].
\]

Since the above holds for infinitesimal shifts \( u = w + s(w) \), a tractable choice of \( p_0 \) (For eg. Gaussian), enables efficient approximation of \( K_{ij} \) by sample averages with samples \( u_i, u'_j = w_i + s(w_i), w_i \sim p_0(W) \).

Moreover \( K_{ij} \) in (8) is a functional of the transformation \( T \) (for fixed \( p(W) \)) i.e. a functional of the shift \( s \) in our case. Therefore, the problem of finding \( p^*(W) \) in (6) reduces to the problem of finding the optimal shift (given \( p_0(W) \)) i.e.

\[
s^*(W) = \arg \min_{s(W)} L[s].
\]

Since \( s \in \mathcal{H}_k \), which is the RKHS for the kernel \( \kappa \), we can solve (10) via functional gradient descent. Defining \( k_{ij}(w, w') = k(g_w(x_i), g_w'(x_j)) \) we have

\[
K_{ij} = E_{w, w' \sim p(W)}[k_{ij}(w + s(w), w' + s(w'))]
\]
Assuming that the distributions \( p(W) \) and shifts \( s(W) \) are functions in a RKHS \( \mathcal{H} \) given by the kernel \( \kappa \) (\( \kappa, k, \) and \( K \) are all different), we have (from the definition of functional gradient \( \nabla_s K_{ij} [s] \)),

\[
K_{ij} [s + \epsilon r] = K_{ij} [s] + \epsilon \nabla_s K_{ij} [s], \quad r > \mathcal{H} + \mathcal{O}(\epsilon^2)
\]  

(12)

Thus we need to compute the difference \( K_{ij} [s + \epsilon r] - K_{ij} [s] \) which, from (11) is given by

\[
K_{ij} [s + \epsilon r] - K_{ij} [s] = E_p [k_{ij} (w + s(w) + \epsilon r(w), w' + s(w') + \epsilon r(w'))]
\]

\[-E_p [k_{ij} (w, w' + s(w'))]
\]

(13)

We use \( E_p \) to denote the expectation when \( w, w' \sim p(W) \). The above equation can be rewritten as

\[
k_{ij} [s + \epsilon r] - k_{ij} [s] = V_1 + V_2
\]

(14)

where the last line follows from the RKHS property. Similarly,

\[
V_1 = E_p [k_{ij} (w + s(w) + \epsilon r(w), w' + s(w') + \epsilon r(w'))] - E_p [k_{ij} (w + s(w), w' + s(w') + \epsilon r(w'))]
\]

\[= \epsilon E_p [\nabla w k_{ij} (w + s(w), w' + s(w') + \epsilon r(w'))] + \mathcal{O}(\epsilon^2)
\]

(15)

\[
V_2 = E_p [\nabla w k_{ij} (w + s(w), w' + s(w') + \epsilon r(w'))] - \epsilon E_p [\nabla w k_{ij} (w + s(w), w' + s(w'))\kappa(w, .)] + \mathcal{O}(\epsilon^2)
\]

(16)

Since we transform the weights \( w \) after every iteration, therefore we only ever need to compute the gradient at \( s(w) = 0 \). Thus, finally, we have the expression

\[
\nabla_s K_{ij} [s] |_{s=0} = E_p [\nabla w k_{ij} (w, w')\kappa(w, .) + \nabla w k_{ij} (w, w')\kappa(w', .)]
\]

(21)

If we draw \( m \) samples of model parameters \( w_1, \ldots, w_m \sim p(w) \), the empirical estimate of \( \nabla_s K_{ij} [s] |_{s=0} \) given by replacing expectations with sample averages is given by

\[
\nabla_s K_{ij} [s] |_{s=0} \approx \frac{1}{m^2} \sum_{l, l'=1}^m [\nabla w k_{ij} (w_l, w_{l'})\kappa(w_l, .) + \nabla w k_{ij} (w_l, w_{l'})\kappa(w_{l'}, .)]
\]

(22)

Without loss of generality consider all the terms in the above expression that contain the gradient with respect to \( w_l \) and let us call that part of the summation \( T_1 \). Therefore

\[
T_1 = \frac{1}{m^2} \sum_{l=1}^m \nabla w k_{ij} (w_l, w_{l'})\kappa(w_l, .) + \frac{1}{m^2} \sum_{l=1}^m \nabla w k_{ij} (w_l, w_l)\kappa(w_l, .)
\]

(23)

Recall the expression for the empirical estimate of the entries of the kernel matrix \( K_{ij} \)

\[
\hat{K}_{ij} \approx \frac{1}{m^2} \sum_{l, l'=1}^m k_{ij} (w_l, w_{l'})
\]

(24)

Differentiating both sides with respect to \( w_1 \),

\[
\nabla w_k \hat{K}_{ij} \approx \frac{1}{m^2} \sum_{l=1}^m \nabla w k_{ij} (w_l, w_{l'}) + \frac{1}{m^2} \sum_{l=1}^m \nabla w k_{ij} (w_l, w_l)
\]

(25)
Note that the term $\nabla_{w_1} k_{ij}(w_1, w_1)$ occurs in both summations. This is because $\nabla_{w_1} k_{ij}(u, v) = \nabla_u k_{ij}(u, v) \nabla_{w_1} u + \nabla_v k_{ij}(u, v) \nabla_{w_1} v = \nabla_{w_1} k_{ij}(w_1, w_1) + \nabla_{w_1} k_{ij}(w_1, w_1)$ when $u = v = w_1$ ($u, v, w_1$ are all variable).

Substituting (25) in (23)

$$T_1 = \kappa(w_1, \ldots) \nabla_{w_1} \hat{K}_{ij}$$

(26)

We can apply the same argument to simplify the terms in (22) that contain gradients with respect to other weights $w_2, \ldots, w_m$ in the same fashion. Therefore,

$$\nabla_s K_{ij}[s] \big|_{s=0} \simeq \frac{1}{m^2} \sum_{l,l'=1}^m \left[ \nabla_{w_l} k_{ij}(w_l, w_l') \kappa(w_l, \ldots) + \nabla_{w_{l'}} k_{ij}(w_l, w_{l'}) \kappa(w_{l'}, \ldots) \right]$$

(27)

$$= \sum_{l=1}^m \kappa(w_l, \ldots) \nabla_{w_l} \hat{K}_{ij}$$

(28)

From the chain rule for functional gradient descent we have $\nabla_s L = \sum_{i,j} \frac{\partial L}{\partial K_{ij}} \nabla_s K_{ij}[s]$ and the corresponding empirical estimate $\nabla_s L \big|_{s=0} \simeq \sum_{i,j} \frac{\partial \hat{L}}{\partial \hat{K}_{ij}} \nabla_s (\sum_{l=1}^m \kappa(w_l, \ldots) \nabla_{w_l} \hat{K}_{ij})$. Switching the order of the summations gives

$$\nabla_s L \big|_{s=0} \simeq \sum_{l=1}^m \kappa(w_l, \ldots) \nabla_{w_l} \hat{L}(w_1, \ldots, w_m)$$

(29)
Figure 4: Results in (a)-(c) are for $n = 100$ samples, in (d)-(f) are for $n = 200$ samples, and (g)-(i) are for $n = 400$ samples. These results show that while the performance of other models (including BNN) worsens (lower accuracy, higher confidence on OOD data) as the number of samples decreases, that of our approach PNCA is largely unaffected thus validating its efficacy in small data settings.