Noname manuscript No.
(will be inserted by the editor)

Deep Bayesian Self-Training

Fabio De Sousa Ribeiro · Francesco Calivá · Mark Swainson · Kjartan Gudmundsson · Georgios Leontidis · Stefanos Kollias

Abstract Supervised Deep Learning has been highly successful in recent years, achieving state-of-the-art results in most tasks. However, with the ongoing uptake of such methods in industrial applications, the requirement for large amounts of annotated data is often a challenge. In most real world problems, manual annotation is practically intractable due to time/labour constraints, thus the development of automated and adaptive data annotation systems is highly sought after. In this paper, we propose both a (i) Deep Bayesian Self-Training[1] methodology for automatic data annotation, by leveraging predictive uncertainty estimates using variational inference and modern Neural Network (NN) architectures, as well as (ii) a practical adaptation procedure for handling high label variability between different dataset distributions through clustering of NN latent variable representations. An experimental study on both public and private datasets is presented illustrating the superior performance of the proposed approach over standard Self-Training baselines, highlighting the importance of predictive uncertainty estimates in safety-critical domains.

Keywords Bayesian CNN · variational inference · self-training · uncertainty weighting · deep learning · clustering · representation learning · adaptation

✉ Fabio De Sousa Ribeiro, Francesco Calivá, Georgios Leontidis, and Stefanos Kollias
Machine Learning Group
University of Lincoln, UK
E-mail: {fdesousaribeiro, fcaliva, gleontidis, skollias}@lincoln.ac.uk

Mark Swainson, Kjartan Gudmundsson
National Centre for Food Manufacturing
Holbeach Technology Park, UK
E-mail: {mswainson, kgudmundsson}@lincoln.ac.uk

[1] Code available at: https://github.com/fabio-deep/Deep-Bayesian-Self-Training

1 Introduction

With the advent of Big Data in industrial applications, the ability to automatically label datasets using limited supervision is increasingly sought after. In most real world problems, manual annotation is practically intractable due to time and labour constraints. Furthermore, recent advances in Supervised Deep Learning have shown that training over parameterised models on large datasets significantly increases performance [1]. With that in mind - and despite the high demand for annotated data - deep learning practitioners have not yet explored or leveraged many of deep learning tools for automatic annotation systems. This is evidenced by the scarcity of existing research in the field, compared to related others [2]. Automated annotation techniques typically involve semi-supervised algorithmic variants, wherein learning systems are often trained on a small initial sample of labelled data, and leverage information from unlabelled data to generalise better [3]. Well established semi-supervised methods such as Self-Training [4], Transfer Learning [5], Co-Training [6], Active Learning [7] and Tri-Training [8] among others have shown to be useful for labelling in the past, but some challenges remain with regards to their scalability to high dimensional data and their suitability to modern Deep Learning settings [9,2]. Prominent recent works have explored some of these ideas in the context of modern deep models, proposing new paradigms such as Co-teaching [10], Active Learning on Image data [2] and analysing Deep Transfer Learning [11,12] with good levels of success. Taking inspiration from these works, in this paper we primarily focus on exploring the Self-Training algorithm in combination with modern Bayesian Deep Learning methods, and leverage predictive uncertainty estimates for self-labelling of high dimensional data.

1.1 Background on Application Domain

In addition to public domain datasets, we evaluate our methods on a real world task involving Optical Character Verification (OCV) of real food packaging images, expanding on earlier work in [13] by reducing manual data annotation.

Incorrectly labelled food products (e.g. bearing an incorrect/illegible *use-by* date) result in product recalls and food waste, as label faults can lead to food safety incidents. Label faults are primarily attributed to human error during error-prone manual checking. Automatic approaches typically involve OCV, whereby a supervisory system holds the correct date code string and transfers it to both the printer and the vision system. The latter will then verify its read, and take appropriate action. Such a system could also be used alongside other systems, such as blockchain, within the food chain for food traceability [14]. Current OCV systems require accurately labelled data to be utilised for training, but the labelling process is time consuming, expensive and requires expertise. They also rely on consistency in date code format, packaging and camera view angle which is difficult to ensure in a manufacturing environment, so there is a great need for a more robust solution.

1.2 Contribution

We propose a Deep Bayesian Self-Training methodology orthogonal to [2], that leverages approximate variational inference in DNNs to estimate predictive uncertainty during a Self-Training setting. Both aleatoric and epistemic uncertainties of predicted pseudo-labels for unseen data are estimated, and the samples with the lowest predictive uncertainty (highest confidence) are added to the training set in an automated manner. We offer ways to mitigate the known problem of propagating errors in Self-Training by including: (i) an entropy penalty on the log likelihood loss to punish over confident output distributions and facilitate thresholding, and (ii) an adaptive sample-wise weight on the influence of predicted pseudo-labelled samples over gradient updates to be inversely proportional to their predictive uncertainty. Lastly, we propose a new simple methodology for visualising and analysing variability between two dataset distributions in DNNs, and attempt to adapt information from one problem to the other by clustering learnt latent variable representations in the context of our application domain. An experimental study on both public and private (real) datasets is presented demonstrating the increased performance of our algorithm over standard Self-Training baselines.

2 Related Work

Deep Learning model's ability to learn abstract hierarchical representations from data has pushed the state-of-the-art in most machine learning related tasks [1, 15]. The uptake of these methodologies in academia and industry has resulted in many diverse and interesting DNN applications, wherein patterns learned from data have been adapted to perform tasks in various domains, including Computer Vision [16,15,17,13], Medical Imaging [18,19,20] and Signal Processing [21,22]. Although many important improvements to DNNs have been made in various domains, there are still many adversities in training models which can be easily adapted to other tasks; and the lack of annotated data is one of the contributing factors.

2.1 Deep Semi-Supervised Learning

Most related work addressing the aforementioned issues is often related to domain adaptation philosophy and semi-supervised learning algorithms such as: Self-Training [4], which is an iterative procedure for self-labelling data points in an unlabelled pool, and re-training a classifier until stop conditions are met. Co-Training [6], can be considered multi-view variant of Self-Training wherein two separate classifiers are trained on different views of the data, and augment each others training sets with their predicted labels. Tri-Training [8] extends Co-Training by having three classifiers, and unlabelled examples are added to a classifier's training set iff. the other two agree on the predicted label. Active Learning [7] selects the most informative samples from a pool of unlabelled data, and retrains the classifier with human given labels in an effort to maximise performance and minimise data labelling requirements. Transfer Learning [5] is often used when there is a lack of annotated data in the target domain, and the goal is to adapt knowledge from one task to another by initialising the weights of the target task with the pre-trained weights of another, often performing better than random initialisation. Among these algorithms, Transfer Learning has undoubtedly had the most success in the context of deep models, and it is widely used in computer vision for adapting visual features from large source domains, to target domains with limited annotated data. Notably, [11] find that; initializing a network with transferred features boosts generalization that lingers even after fine-tuning to the target dataset, and transferring features from distant tasks is still better than using random weights. Recent work in [23] suggests that a single DL model can jointly learn a number of tasks from multiple domains successfully. In fact, it was observed that adding knowledge from unrelated tasks never hurts performance, rather mostly improves it on all tasks. This phenomenon is complimented by research in [24], with results suggesting that combining tasks, even via a naïve multihead architecture, always improves performance. Authors in [25] propose learning a network comprised of the most successful layers from many different source networks, which are continuously generated and evaluated by a Recurrent Neural Network (RNN) controller. Task Transfer Learning was recently studied in great depth by [12], where a fully computational approach termed Taskonomy was proposed. This was achieved by identifying dependencies between 26 different tasks in latent space, producing a computational taxonomic map for task Transfer Learning. Deep generative modelling is also gaining popularity in tackling adaptation of knowledge learnt from data generating distributions to pool sets of unlabelled data [26,27,28]. Other notable related works presented more recently include Co-Teaching [10], wherein two neural networks are trained simultaneously and teach each other to select clean labels, then decide what data to use for training. Mean teacher models [29] maintain an exponential moving average of model weights and penalise inconsistent predictions, enabling training with fewer labels as an added benefit. Deep Co-Training [30] extends the original Co-Training algorithm by training multiple DNNs with different views generated by exploiting adversarial examples. In [31] a simple method termed Pseudo-Label similar to Entropy Regularisation [32] is proposed, and it consists of iteratively assigning pseudo-labels via the maximum predicted probability of a NN. Although research on Self-Training with deep models is scarce, notable work in [33] presents an unsupervised domain adaptation (UDA) framework based on Self-Training for semantic segmentation using DNNs. They develop a self-paced policy that increases the number of pseudo-labels incorporated in each additional round, and demonstrate performance benefits over other popular methods. However, as is the case with all previous works mentioned thus far, their proposed approach does not provide principled predictive uncertainty estimates. The black box nature of DNNs is a concern in most real world applications, and by quantifying what a model doesn't know with uncertainty measures, we can not only better trust our predictions but also avoid potentially harmful outcomes [34]. With that in mind, perhaps the most significant related work is in [2], where the authors propose a Bayesian formulation of Active learning for image data using DNNs, obtaining a significant improvement on existing active learning approaches by considering uncertainty estimates in approximating acquisition functions.

2.2 **Uncertainty Estimation**

The estimation of uncertainty as a measure of confidence over a model's predictions is desireable for self-labelling, and for safety-critical systems in general [34]. Bayesian Neural Networks (BNNs) were studied by many in the past [35,36,37] and have more recently regained popularity. In BNNs, uncertainty is typically captured by placing a prior distribution, such as a Gaussian, over the weights and averaging over all possible parameters, rather than optimising them directly. Bayesian inference is then used to compute the posterior over the weights capturing the set of likely parameters. However, BNNs are difficult to perform inference in with traditional methods, as they do not scale well scale to high dimensional inputs or very complex DL models [34]. Recent promising methods including [34,38, 39] offer alternative ways of capturing uncertainty by simple modifications to loss functions, and having the network learn/predict aleatoric uncertainty in an unsupervised manner. Aleatoric uncertainty relates to sensory noise in the acquisition process of the data, and is therefore inherently irreducible [40]. However, we argue that it can be a great tool for quantifying our uncertainty about pseudo-label predictions. In [39], Dropout was shown to perform approximate variational inference, wherein stochastic forward passes with Dropout at test time are effectively samples from the approximate posterior. This technique is know as Monte Carlo (MC) Dropout [39], and can be used to quantify epistemic uncertainty in NN predictions. Epistemic uncertainty relates to our uncertainty about the model parameters, which is in fact reducible as we observe more data. This is because we can explain the uncertainties about the model parameters in the limit of observing all explanatory variables of the data [40,34]. This type of uncertainty is useful for identifying out-of-distribution data points, and is the most important type of uncertainty measure when assigning pseudo-labels to data.

In this paper we argue that, with some modifications, uncertainty estimation techniques in Bayesian deep learning can also be useful in a Self-Training setting, and to the best of our knowledge these ideas have yet to be explored in this context. All things considered, we propose a Deep Bayesian Self-Training algorithm, in which a DNN assigns pseudo-labels to new data, and automatically weighs their sample-wise importance for the next Self-Training iteration to be inversely proportional to the predictive uncertainly of the assigned pseudo-label. In this way, we can reduce the burden of manual data annotation requirements, and also offer a measure of uncertainty about our predictions which is important in safety-critical domains.

3 Deep Bayesian Self-Training

In this section, we provide a brief background on Bayesian NNs, and explore the idea of uncertainty estimation of pseudo-label predictions for unlabelled data, in a Deep Bayesian Self-Training framework (see Algorithm 1). In order to quantify what our and does not know, we extend existing approaches for estimating uncertainty in deep CNNs [34,41]. To this end, we consider the following Bayesian formulation of a deep CNN for estimating both aleatoric and epistemic uncertainties.

3.1 Bayesian Neural Networks

Let $\mathcal{D} = \{(\mathbf{X}, \mathbf{Y})\}$ denote a dataset given as N pairs of inputs $\mathbf{x}_i \in \mathbb{R}^d$ of dimension d, and class labels $\boldsymbol{y}_i \in \{1, \ldots K\}$ of K total classes. Assuming a Bayesian Neural Network (BNN) formulation, we place a Gaussian prior probability distribution $p(\boldsymbol{\omega})$ over the set of trainable parameters $\boldsymbol{\omega} = \{\mathbf{W}_1, \ldots, \mathbf{W}_\ell\}$. We define the likelihood conditional output distribution $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})$ of NN for mapping inputs to labels, by finding parameters $\boldsymbol{\omega}$ that yield the Maximum Likelihood Estimate (MLE). MLE is the pillar of supervised learning in DNNs and is defined as

$$\widehat{\boldsymbol{\omega}}_{\text{ML}} = \arg\max_{\boldsymbol{\omega}} \sum_{i=1}^{N} \log p(\boldsymbol{y}_i|\mathbf{x}_i, \boldsymbol{\omega}), \tag{1}$$

yielding a point estimate for the most likely parameters to have generated the data. In a Bayesian sense, the MLE is a special case of Maximum A Posteriori (MAP) estimation when a uniform prior is assumed. In practical classification tasks, the MLE estimator is obtained by minimising the negative log-likelihood of a Bernoulli or softmax distribution depending on the number of classes. We define the softmax negative log-likelihood of our classification NN model as

$$-\log p(\boldsymbol{y}_i = k|\mathbf{x}, \boldsymbol{\omega}) = -\left(\mathbf{z}_k - \log \sum_{k'} \exp(\mathbf{z}_{k'})\right) \tag{2}$$

where $\mathbf{z}$ denotes the vector of output logits by the network and k denotes a class. Having defined a prior and a likelihood, we would like to compute the posterior probability distribution over the weights given the data by Bayes rule

$$p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega})}{p(\mathbf{Y}|\mathbf{X})} \propto p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\omega})p(\boldsymbol{\omega}), \tag{3}$$

with which we can also formulate the predictive distribution given new inputs $\mathbf{x}^*$ and labels $\boldsymbol{y}^*$

$$p(\boldsymbol{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\boldsymbol{y}^*|\mathbf{x}^*, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})\mathrm{d}\boldsymbol{\omega}, \tag{4}$$

enabling predictions using a full distribution over the parameters $\boldsymbol{\omega}$, which captures uncertainty over the model parameters, rather than using a point estimate. However, in most cases, the posterior distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ cannot be evaluated analytically. This is because to compute the marginal probability $p(\mathbf{Y}|\mathbf{X})$ we must integrate over all possible model parameters $\boldsymbol{\omega}$ with weighted probability $p(\boldsymbol{\omega})$, in order to obtain the normalising constant, also known as the model evidence. Since the true posterior distribution $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$ is intractable, various approximations exist [37,42,36]. Most of them were important early steps towards performing approximate inference in Bayesian NNs, but are unfortunately difficult to employ in modern applications due to scalability constraints or expert knowledge requirements. More recent work in [43,44,41,45] addressed some of these issues with variational inference, reigniting interest in the field of Bayesian NNs.

3.2 Variational Inference

Next, we provide a background on variational inference (VI) to contextualise some of the ideas presented in [41], wherein Dropout is shown to perform approximate variational inference in NNs when used at test time. In VI, a factorised variational distribution from a tractable family $q_\theta(\boldsymbol{\omega})$, parameterised by θ, is defined for approximating the posterior distribution by minimising the Kullback-Leibler (KL) divergence between $q_\theta(\boldsymbol{\omega})$ and $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$. Intuitively, the KL divergence is a non-negative asymmetric measure of similarity between the two distributions $\text{KL}(q_\theta(\boldsymbol{\omega}) \ || \ p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}))$, which we minimise via the variational parameters θ of our approximating distribution $q_\theta(\boldsymbol{\omega})$

$$\widehat{\theta} = \arg\min_{\theta} \ \mathbb{E}_{q_\theta(\boldsymbol{\omega})}\big[\log q_\theta(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})\big]. \tag{5}$$

However, optimising the KL divergence directly requires knowledge of the intractable posterior. This is circumvented by instead maximising the evidence lower bound (ELBO) on the marginal log-likelihood $\log p(\mathbf{Y}|\mathbf{X})$, derived via Jensen's inequality $\log(\mathbb{E}[X]) \geq \mathbb{E}[\log(X)]$

$$\mathcal{L}_{\text{ELBO}}(\theta) = \log p(\mathbf{Y}|\mathbf{X}) - \text{KL}(q_\theta(\boldsymbol{\omega}) \ || \ p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})), \tag{6}$$

and given that the KL divergence ≥ 0 then

$$\log p(\mathbf{Y}|\mathbf{X}) = \mathcal{L}_{\text{ELBO}}(\theta) + \text{KL}(q_\theta(\boldsymbol{\omega}) \ || \ p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})). \tag{7}$$

By maximising the lower bound we implicitly maximise $\log p(\mathbf{Y}|\mathbf{X})$, and minimise the KL divergence as intended. We extend these ideas in light of recent developments in [41] with the Monte Carlo Dropout approximation using $q_\theta(\boldsymbol{\omega})$, further explained in the following section.

Algorithm 1 Deep Bayesian Self-Training

```
1: Note: Pseudo code for training a progressively growing DenseNet in a Bayesian Self-Training setting.
   The incremental growth factor $\nu$ is adjusted for dataset size.
2: function DBST($\mathcal{D} = \{(\mathbf{x}, \boldsymbol{y}, \lambda)\}_{i=1}^{N}$ , $\mathcal{U} = \{\widetilde{\mathbf{x}}\}_{i=1}^{\widetilde{N}}$)    ▷ Input training and unlabelled datasets
3:     Initialise : $r \leftarrow 0$, $k \leftarrow 12$, $k_{\max} \leftarrow 24$
4:     Initialise $\forall\ \mathbf{x}_i \in \mathcal{D} : \lambda_i \leftarrow 1$
5:     while $|\mathcal{U}| > 0$ do    ▷ Unlabelled dataset cardinality
6:         $r \leftarrow r + 1$, $k \leftarrow \min(k + \nu \cdot (r-1), k_{\max})$
7:         $f(\mathbf{x}) \leftarrow$ TRAIN($\mathcal{D}, k$)    ▷ Train a DenseNet with growth rate $k$
8:         $(\widehat{\mathbf{p}}, \widehat{\mathbf{s}}) \leftarrow$ MC DROPOUT($f(\mathbf{x}), \mathcal{U}$)
9:         for $\widetilde{\mathbf{x}}_i \in \mathcal{U}$ do
10:            $\mathrm{Var}[\boldsymbol{y}_i] \leftarrow \exp(\widehat{\mathbf{s}}_i) + \mathbb{H}[\widehat{\mathbf{p}}_i]$    ▷ Aleatoric and Epistemic uncertainties
11:            $\widehat{\boldsymbol{y}}_i \leftarrow \arg\max \widehat{\mathbf{p}}_i$
12:            if $\mathrm{Var}[\boldsymbol{y}_i] < \tau$ then    ▷ $\tau$ is computed via IQR
13:                $\lambda_i \leftarrow 1/\exp(\mathrm{Var}[\boldsymbol{y}_i])^{\phi(r)}$
14:                $\mathcal{D} \leftarrow \mathcal{D} \cup \{\widetilde{\mathbf{x}}_i, \widehat{\boldsymbol{y}}_i, \lambda_i\}$    ▷ Add weighted pseudo-labelled sample to $\mathcal{D}$
```

3.3 Continuous Relaxation of Dropout

Concrete Dropout is based on concrete relaxation of discrete distributions [46], allowing the replacement of Dropout's discrete Bernoulli distribution with its continuous relaxation [47]. To obtain calibrated uncertainty estimates with Monte Carlo Dropout, it is necessary to tune the Dropout probabilities. A grid-search is a common but costly approach for large models, highlighting the benefit of optimising them directly with Gradient Descent. This requires formulating an objective for minimising epistemic uncertainty [41] using the variational interpretation of Dropout.

Formally, Dropout can be treated as an approximating distribution $q_\theta(\boldsymbol{\omega})$ to the posterior in a BNN, where $\boldsymbol{\omega}$ represents the weight matrices of the ℓ^{th} of L layers in the network $\boldsymbol{\omega} = \{\mathbf{W}_\ell\}_{\ell=1}^{L}$, and θ are the variational parameters to optimise [47]. Let $\mathcal{F}(\boldsymbol{\omega})$ be the model with weight matrix realisation $\boldsymbol{\omega}$; given a random set S comprising M of all N data points, and denote the model's output on the $\mathbf{x}_i$ input as $\mathcal{F}(\mathbf{x}_i; \boldsymbol{\omega})$. The following NN objective function can then be formulated

$$\hat{\mathcal{L}}_{\text{MC}}(\theta) = -\frac{1}{M}\sum_{i \in S} \log p(\boldsymbol{y}_i | \mathcal{F}(\mathbf{x}_i; \boldsymbol{\omega})) + \frac{1}{N}\mathrm{KL}(q_\theta(\boldsymbol{\omega}) \mid\mid p(\boldsymbol{\omega})), \quad (8)$$

where $p(\boldsymbol{y}_i | \mathcal{F}(\mathbf{x}_i; \boldsymbol{\omega}))$ is the model's likelihood, a Gaussian with a predictive mean given by $\mathcal{F}(\mathbf{x}_i; \boldsymbol{\omega})$. KL is a regularisation term which constrains the approximate posterior $q_\theta(\boldsymbol{\omega})$ from deviating too far from prior $p(\boldsymbol{\omega})$.

Following [38] we can approximate the KL term with

$$\mathrm{KL}(q_M(\mathbf{W}) \mid\mid p(\mathbf{W})) \propto \frac{l^2(1-p)}{2}||\mathbf{M}||^2 - K\mathbb{H}[p], \quad (9)$$

where $\{\mathbf{M}_\ell, p_\ell\}_{\ell=1}^{L}$ is a set of mean weight matrices and Dropout probabilities, such that (s.t.) $q_{M_\ell}(\mathbf{W}_\ell) = \mathbf{M}_\ell \cdot \mathrm{diag}[\mathrm{Bernoulli}(1-p_\ell)^{K_\ell}]$ for a single NN weight matrix $\mathbf{W}_\ell \in \mathbb{R}^{K_{\ell+1} \times K_\ell}$. $\mathbb{H}[p]$ is simply the entropy of a Bernoulli random variable with probability p

$$\mathbb{H}[p] := -p\log p - (1-p)\log(1-p), \quad (10)$$

which can be interpreted as a regularisation term that only depends on Dropout probability p, so minimising the KL term is equivalent to maximising the entropy of a Bernoulli random variable with probability $(1-p)$. Rather than sampling the random variable from the discrete Bernoulli distribution, by adopting the Concrete distribution [46,47] with some temperature t, it is possible to sample variables in the interval $[0,1]$, s.t. the concrete relaxation distribution $\widetilde{\mathbf{z}}$

$$\widetilde{\mathbf{z}} = \mathrm{sigmoid}\Bigg(\frac{1}{t} \cdot \big[\log p - \log(1-p) + \log u - \log(1-u)\big]\Bigg), \quad (11)$$

parametrised by means of $u \sim \mathrm{Unif}(0,1)$, provides a relationship between $\widetilde{\mathbf{z}}$ and u, which is differentiable w.r.t. p. With the concrete relaxation of the dropout masks, the Dropout probabilities for each layer $\{p_\ell\}_{\ell=1}^{L}$ can be optimised using the pathwise derivative estimator [47].

3.4 Entropy Penalty on Output Distributions

The probabilities assigned to incorrect classes at test time help quantify a model's ability to generalise. By penalising output distributions with low entropy (i.e. confident predictions), we can obtain a similar effect to label smoothing and improve generalisation [48]. This can useful in Self-Training, since we assign pseudo labels-based on low uncertainty predictions, which are in some cases wrongly assigned. We suggest that by penalising very confident output distributions we can improve generalisation, and make thresholding easier since the output distributions are smoother, rather than overly concentrated at 0 or 1. The entropy of a NNs output conditional distribution is given by

$$\mathbb{H}\big[p(\boldsymbol{y}|\mathbf{x},\boldsymbol{\omega})\big] = -\sum_i p(\boldsymbol{y}_i|\mathbf{x},\boldsymbol{\omega}) \log p(\boldsymbol{y}_i|\mathbf{x},\boldsymbol{\omega}), \tag{12}$$

with $p(\boldsymbol{y}|\mathbf{x},\boldsymbol{\omega})$ as the probability distribution obtained from a softmax function. To penalise very confident predictions we can simply take the negative log-likelihood, and subtract the entropy of the output distribution as

$$\mathcal{L}_{\text{NLL}}(\boldsymbol{\omega}) = -\sum \log p(\boldsymbol{y}|\mathbf{x},\boldsymbol{\omega}) - \beta\mathbb{H}\big[p(\boldsymbol{y}|\mathbf{x},\boldsymbol{\omega})\big], \tag{13}$$

where the scaling hyperparameter β balances how much we'd like to penalise non-uniformity of the softmax.

3.5 Inverse Uncertainty Weighting

A known limitation of Self-Training is the potential accumulation of wrongly pseudo-labelled samples being added to the training set. A common approach is to remove less confident samples from the training set, and leave them in the unlabelled set. However, this tends to under perform in practice, as the algorithm can become biased by continuously adding the easiest unlabelled samples to the training set. This can hinder learning over time, as more difficult and potentially informative samples are neglected.

In attempt to mitigate this behaviour, we propose a sample-wise weighting scheme during training that places a weight on each training sample $\{\mathbf{x}_i, \widehat{\boldsymbol{y}}_i, \lambda_i\}$, proportional to the predictive uncertainty over its pseudo-label $\widehat{\boldsymbol{y}}_i$, such that its contribution to the loss function is inversely proportional to its predictive uncertainty (see Algorithm 1). To calculate the predictive uncertainty we can have the network predict the aleatoric uncertainty as one of its outputs, and add the epistemic uncertainty obtained from the variance of Monte Carlo Dropout samples.

Formally, let $\widehat{\mathbf{p}}_t = \text{softmax}(\mathcal{F}(\mathbf{x};\widehat{\boldsymbol{\omega}}_t))$ denote the softmax out of a BNN, and $\{\widehat{\mathbf{p}}\}_{t=1}^T$ be the set of outputs from T Monte Carlo Dropout samples at test time, each parameterised by weights drawn from the approximate posterior $\widehat{\boldsymbol{\omega}}_t \sim q_{\hat{\theta}}(\boldsymbol{\omega})$. We propose calculating the predictive uncertainty from these samples by generalising the binary variant approach in [49] to a multivariate classification setting. By the definition of variance of a multinomial distribution we can decompose the variance of $\widehat{\mathbf{p}}$ into

$$\text{Var}\big[\widehat{\mathbf{p}}\big] \approx \text{tr}\Big(\mathbb{E}\big[\text{diag}(\widehat{\mathbf{p}}) - \widehat{\mathbf{p}}\widehat{\mathbf{p}}^\top\big] + \mathbb{E}\big[\widehat{\mathbf{p}}^2\big] - \mathbb{E}\big[\widehat{\mathbf{p}}\big]^2\Big), \tag{14}$$

where the first term represents aleatoric uncertainty σ_{a}^2, and the second is the epistemic σ_{e}^2. Each diagonal entry of the resulting matrix is the variance of a binomially distributed random variable, and the off-diagonals are negative covariances for fixed T. Since we're only interested in a single number to measure our uncertainty, we take trace of the resulting uncertainty matrix.

Alternatively, we can have the NN predict the input noise variance σ_{a}^2 as one of its outputs [34], by assuming measurement error in our target function $y = \mathcal{F}(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_{\text{a}}^2)$. The predictive variance in a multivariate classification setting is then given by

$$\begin{aligned}\text{Var}\big[\widehat{\boldsymbol{y}}\big] &\approx \frac{1}{T}\sum_t \exp(\widehat{\mathbf{s}}_t) - \sum_j \mathbb{E}\big[\widehat{\mathbf{p}}\big]\log\mathbb{E}\big[\widehat{\mathbf{p}}\big] = \\ &= \mathbb{E}[\exp(\widehat{\mathbf{s}})] + \mathbb{H}\big[\text{softmax}(\mathcal{F}(\mathbf{x};\widehat{\boldsymbol{\omega}}))\big],\end{aligned} \tag{15}$$

the entropy term measures epistemic uncertainty in the output softmax distributions, whereas the log aleatoric uncertainty $\widehat{\mathbf{s}}_i := \log\sigma_{a,i}^2$ term is regressed by the NN for each input $\mathbf{x}_i$, for numerical stability. To capture aleatoric uncertainty in our classification task, we can use Monte Carlo integration on the NNs gaussian log-likelihood objective function, by drawing $t \in T$ samples of gaussian noise corrupted NN output logits $\mathcal{F}(\mathbf{x})$, yielding the following loss

$$\mathcal{L}_{\text{NLL}} = -\log\mathbb{E}\big[\text{softmax}(\mathcal{F}(\mathbf{x}) + \epsilon_t \odot \exp(\widehat{\mathbf{s}}|\mathbf{x}))\big], \tag{16}$$

with $\epsilon_t \sim \mathcal{N}(0, I)$ parameterised by the predicted aleatoric uncertainty $\exp(\widehat{\mathbf{s}}|\mathbf{x})$ for each sample $\mathbf{x}_i$, which learns to capture measurement error.

Having calculated the predictive uncertainty $\text{Var}[\widehat{\mathbf{p}}]$ of our pseudo-labels, we calculate a per sample importance weight $\{\mathbf{x}_i, \widehat{\boldsymbol{y}}_i, \lambda_i\}$ with

$$\lambda_i = \frac{1}{\exp(\text{Var}[\widehat{\mathbf{p}}_i])^{\phi(r)}}, \tag{17}$$

where $\phi(\cdot)$ is a parameterised hyperbolic tangent function

$$\phi(r) = \frac{1 - \exp(\gamma \cdot r + b)}{1 + \exp(\gamma \cdot r + b)}, \tag{18}$$

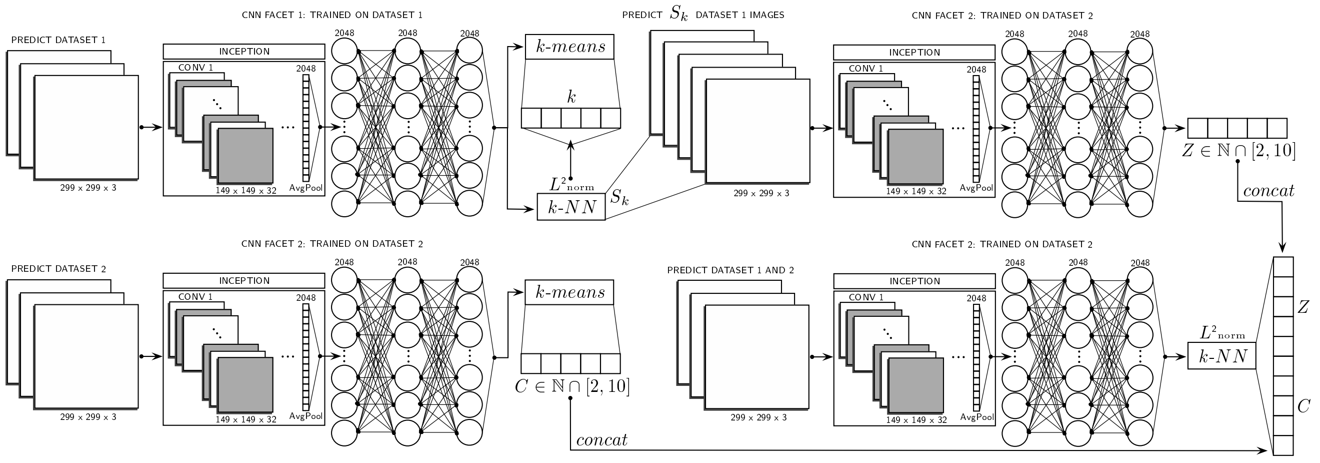

PREDICT DATASET 1
CNN FACET 1: TRAINED ON DATASET 1
INCEPTION
CONV 1
2048
299 x 299 x 3
149 x 149 x 32
AvgPool
k-means
k
L^2_{norm}
k-NN
S_k
PREDICT S_k DATASET 1 IMAGES
CNN FACET 2: TRAINED ON DATASET 2
$Z \in \mathbb{N} \cap [2, 10]$
concat
PREDICT DATASET 2
$C \in \mathbb{N} \cap [2, 10]$
PREDICT DATASET 1 AND 2
Z
C

Fig. 1 Illustration of the multiple CNN facet adaptation framework proposed, which is based on clustering of extracted latent variable representations. The architectural details of each CNN are as previously described in Figure 8.

with γ, b as scale and intercept terms, and r denotes the Self-Training iteration. The weighted penalised log likelihood of our NN with weights $\boldsymbol{\omega}$ is then

$$\mathcal{L}_{\text{PLL}} = \sum_i \lambda_i \log p(\boldsymbol{y}_i|\mathbf{x}_i, \boldsymbol{\omega}) - \beta\mathbb{H}\big[p(\boldsymbol{y}_i|\mathbf{x}_i, \boldsymbol{\omega})\big], \qquad (19)$$

where $p(\boldsymbol{y}|\mathbf{x}, \boldsymbol{\omega})$ is computed via softmax, and the optional confidence entropy penalty term is balanced by β. By tuning γ and b we can obtain the desired behaviour over r iterations, s.t. when the uncertainty is low we assign high weight to the predicted pseudo-labelled sample $\{\mathbf{x}_i, \widehat{\boldsymbol{y}}_i, \lambda_i \approx 1\}$. We can incrementally encourage the model to assign more weight to uncertain pseudo-labelled samples as Self-Training progresses, since in the $\lim_{r\to\infty} \phi(r) = -1$. Intuitively, this procedure inverts eq. (17) over time, incrementally forcing exploration by adding more uncertain, and potentially informative samples, to the training set. In summary, using this logic along with entropy penalties on over-confident output distributions, we can mitigate the effect of pseudo-labelling error accumulation in the training set, and adjust risk taking by tuning γ and b. Once per-sample predictive uncertainties are calculated, we decide on which pseudo-labelled samples to add to the training set via a Tukey Fence. Intuitively, assume a NN has been trained on data $\mathcal{D} = \big\{(\mathbf{x}_i, \boldsymbol{y}_i)\big\}_{i=1}^N$, learning a function $\mathcal{F}(\mathbf{x}; \boldsymbol{\omega})$ for mapping inputs to labels. At inference time, we take the correct predictions where $\boldsymbol{y}_i = \mathcal{F}(\mathbf{x}_i; \boldsymbol{\omega})$, and retrieve their predictive uncertainty. We then summarise variability by calculating the Interquartile Range (IQR) outlier statistic, and define an uncertainty upper bound τ, under which pseudo-labelled samples from $\mathcal{U} = \big\{\widetilde{\mathbf{x}}_i\big\}_{i=1}^{\widetilde{N}}$ have to be, in order to be added to $\mathcal{D}$ following

$$\mathcal{D}^* = \forall_i \in \big\{\mathcal{D} \cup \{\widetilde{\mathbf{x}}_i, \widehat{\boldsymbol{y}}_i, \lambda_i\} \mid \text{Var}[p(\boldsymbol{y}_i|\mathbf{x}_i)] < \tau\big\}, \qquad (20)$$

where $\widehat{\boldsymbol{y}}_i$ denotes the pseudo-label assigned to sample $\mathbf{x}_i$ computed as $\widehat{\boldsymbol{y}}_i = \arg\max \widehat{\mathbf{p}}_i$, and $\mathcal{D}^*$ is the augmented training set. Lastly, we can also easily adjust the uncertainty upper bound τ by selecting higher or lower quartiles to reflect how confident we'd like to be about predictions before adding samples to $\mathcal{D}^*$.

4 Latent Variable Adaptive Clustering

We propose a new simple methodology for visualising and analysing variability between distributions and attempt to adapt information from one problem to another in DNNs. In Figure 1 an illustration of our adaptation framework is shown using an example backbone InceptionV3 CNN. Let the following denote 2 training sets from separate datasets targeting the same task

$$\begin{aligned}\mathcal{D}_1 &= \big\{(\mathbf{x}_1^{(i)}, \boldsymbol{y}_1^{(i)}) \;;\; i = 1, \dots, N_1\big\},\\ \mathcal{D}_2 &= \big\{(\mathbf{x}_2^{(i)}, \boldsymbol{y}_2^{(i)}) \;;\; i = 1, \dots, N_2\big\},\end{aligned} \qquad (21)$$

and the 2 respective test sets as

$$\begin{aligned}\mathcal{T}_1 &= \big\{(\widetilde{\mathbf{x}}_1^{(i)}, \widetilde{\boldsymbol{y}}_1^{(i)}) \;;\; i = 1, \dots, \widetilde{N}_1\big\},\\ \mathcal{T}_2 &= \big\{(\widetilde{\mathbf{x}}_2^{(i)}, \widetilde{\boldsymbol{y}}_2^{(i)}) \;;\; i = 1, \dots, \widetilde{N}_2\big\}.\end{aligned} \qquad (22)$$

Let $\mathcal{F}(\mathcal{D}_1; \mathbf{W}_1)$ and $\mathcal{F}(\mathcal{D}_2; \mathbf{W}_2)$ denote 2 architecturally identical CNNs trained separately on each dataset. For each CNN we extract the final Fully-connected layer activations $\{\mathbf{x}_1^{(i)}, \widetilde{\mathbf{x}}_1^{(i)}\} \in \mathbb{R}^{2048}$ and $\{\mathbf{x}_2^{(i)}, \widetilde{\mathbf{x}}_2^{(i)}\} \in \mathbb{R}^{2048}$ as latent variables representations, by simply forward-propagating each image through as is typically done at inference time.

Utilising these, our adaptation methodology is then performed as follows:

1. Given $\mathcal{D}_2$, produce a set of clusters $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ by minimising the within-cluster L^2 norms of the following clustering objective function
$$\widehat{\mathbf{C}}_{k\text{-means}} = \arg\min_{\mathbf{C}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in\mathbf{C}_i} \left|\left|\mathbf{x} - \boldsymbol{\mu}_i\right|\right|^2. \tag{23}$$
2. Repeat step 1 with $\mathcal{D}_1$ to generate k clusters $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$, and compute the k closest instances in $\mathcal{D}_1$ to each centroid in $\mathbf{U}$. Fetch the corresponding set of images $\mathbf{S} = \{\mathbf{S}_1, \ldots, \mathbf{S}_k\}$, whose latent variables are closest to $\mathbf{U}$;
3. Forward-propagate $\mathbf{S}$ through $\mathcal{F}(\mathcal{D}_2; \mathbf{W}_2)$ to obtain a new set of adapted clusters $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$, where $\mathbf{S}$ is considered an approximation of $\mathbf{U}$ from $\mathcal{F}(\mathcal{D}_1; \mathbf{W}_1)$;
4. Derive an augmented cluster representation that encapsulates knowledge from both facets of the trained CNNs, by concatenating the respective $\mathbf{C}$ and $\mathbf{Z}$ clusters into a set $\mathbf{A} = \{\mathbf{c}_1, \ldots, \mathbf{c}_k, \mathbf{z}_1, \ldots, \mathbf{z}_k\}$;
5. Compute the euclidean distance between $\mathcal{T}_1$ and $\mathbf{A}$ and evaluate the classification performance;
6. Iteratively remove the lowest performing cluster in $\mathbf{A}$ and repeat step 5 until the performance stops improving.

In all cases, the k-means++ [50] seeding strategy was used, whereby the first cluster center $\mathbf{c}_1$ is chosen uniformly at random from $\mathcal{X}$, and all preceding cluster centers $\mathbf{x} \in \mathcal{X}$ are chosen with probability
$$\mathbf{c}_i = \frac{D(\mathbf{x})^2}{\sum_{\mathbf{x}\in\mathcal{X}} D(\mathbf{x})^2}, \tag{24}$$
where $D(\mathbf{x})$ denotes the distance between $\mathbf{x}$ and the closest $\mathbf{c}_i$. Moreover, we assign the class label of a given cluster $\mathbf{c}_i$ as simply the mode class j of all data points within it
$$\mathbf{c}_i^j = \max_{j\in J} \left|\mathbf{c}_i \cap j\right|. \tag{25}$$
In the experimental study of Section 6, we demonstrate that our method distills and adapts knowledge from both trained CNNs on real data, achieving better performance than direct inference of $\mathcal{T}_1$ with $\mathcal{F}(\mathcal{D}_2; \mathbf{W}_2)$, without any parameter retraining.

5 Experimental Study

This section is divided into two separate subsections, the first subsection presents experiments using Deep Bayesian Self-Training applied to the MNIST public domain dataset. An ablation study is presented and comparisons are made with baseline methods. The second subsection comprises a study using private (real) datasets, in which we perform some preliminary experiments using Transfer Learning and then we evaluate our proposed Latent Variable Adaptable Clustering method. We then finish off the second subsection by evaluating Deep Bayesian Self-Training on the self-annotation of the real datasets.

5.1 MNIST Dataset

In order to validate our algorithm we conduct a series of self-labelling experiments on the popular MNIST dataset. The MNIST dataset is comprised of 60,000 training, and 10,000 testing handwritten digit examples respectively. Firstly, we try to create a realistic scenario by splitting the 60,000 training examples into a smaller but balanced training set of only 50 examples per class, a validation set of 500 training examples per class, and allocate all remaining data to the unlabelled pool set. We begin by defining our backbone NN architecture of choice as a DenseNet [15]. DenseNets have revealed several well founded advantages over previous architectures, from mitigating vanishing gradients, to encouraging feature propagation and reuse with shorter connections between layers [15,51]. The dense connectivity in DenseNets can be formally defined as
$$\mathbf{A}^{[\ell]} = f\Big(\text{BN}\Big(\mathbf{W}^{[\ell]} \cdot \big[\mathbf{A}^{[0]}, \mathbf{A}^{[1]}, \ldots, \mathbf{A}^{[\ell-1]}\big]\Big)\Big), \tag{26}$$
where $f(\cdot)$ is the ReLU activation function, BN$(\cdot)$ is Batch Normalisation [52] and $\big[\mathbf{A}^{[0]}, \mathbf{A}^{[1]}, \ldots, \mathbf{A}^{[\ell-1]}\big]$ represents feature map-wise concatenation of all layers preceding ℓ. A sequential composite function consisting of BN, ReLU and 3×3 convolution can then be defined as $H^{[\ell]}$. Each function $H^{[\ell]}$ produces ω feature maps, known as the growth rate of the network, and each layer ℓ takes as input $f + \omega \times (\ell - 1)$ total feature maps, where f denotes the number of channels in the visible layer. To reduce spatial dimensionality of feature maps, a Transition layer is introduced between densely connected DenseBlocks. Transition layers in [15] are composed of BN followed by 1×1 convolution and 2×2 average pooling with a feature map compression factor $\theta = 0.5$.

Following Algorithm 1 closely, we propose a progressively growing NN scheme by starting off with a 40 layer deep DenseNet with a growth rate $k = 12$, and incrementally increasing the growth rate (width) of the network as more data is added to the training set. In the first iteration, the network has only 181k parameters to avoid overfitting on the small initial training set, but complexity of the network is incrementally increased in an automated way. As described in

Table 1 Deep Bayesian Self-Training results on self-labelling the MNIST dataset. τ is the upper bound uncertainty threshold for augmenting $\mathcal{D}^*$, λ_i are sample-wise inverse uncertainty weights, and r is the number of Self-Training iterations taken before stop conditions were met. All metrics (precision, recall, F1-score and cohen's κ) are reported in 1−metric format.

Deep Bayesian Self-Training (DBST) Results on MNIST									
Model	τ	λ_i	$\mathcal{L}_{\text{PNLL}}$	Precision	Recall	F1-score	Unlabelled	Cohen's κ	r iters
DST	-	✗	✗	.0103	.0103	.0103	781	**.0115**	15
DEST	Q3	✓	✗	.0044	.0044	.0044	4,391	.0049	20
DBST-1	Q3	✗	✗	.0042	.0043	.0043	5,044	.0045	15
DBST-2	Q3	✓	✗	.0032	.0032	.0032	4,092	.0035	21
DBST-3	Q2	✓	✗	.001	.001	.001	17,828	**.0011**	27
DBST-4	Q2	✓	✓	.0071	.007	.0071	**762**	.0079	26

greater detail in section 3.5, we employ Monte Carlo Dropout at test time to calculate the predictive uncertainty of the assigned pseudo labels samples. In all cases, we take $T = 30$ samples, equating to 30 different Dropout masks. We compare the performance of our proposed approach with a baseline ensemble method (DEST) similar to [53] for estimating predictive uncertainty, and the vanilla Self-Training methodology, albeit in a Deep Learning model, considering only the output probability of the NN as a measure of confidence, similarly to [31]. We also evaluate the effect of our inverse uncertainty weighting scheme, as well as the entropy penalty on confident output distributions on the performance of our Bayesian Self-Training algorithm.

5.1.1 Training Details

In all MNIST experiments, we use the same DenseNet model and hyperparameters for fair comparisons. Specifically, we train the networks using Stochastic Gradient Descent (SGD) with a Nesterov momentum of 0.9, a batch size of 32, and an initial learning rate of 0.1. We train all models for 75 epochs and reduce the learning rate by a factor of 10 at 50 and 75% of the way through training. All models are trained using the same train/valid/test/unlabelled splits and no data augmentation is used aside from simple image standardisation (mean 0 sd. 1) and we take $T = 30$ Monte Carlo Dropout samples to at test time as explained in section 3.5. With regards to the ensemble, we train $M = 5$ models each initialised with random weights, and capture the predictive uncertainty following equation (15), but without using Dropout at test time. Lastly, the stop conditions can be adjusted depending on the application at hand, but here they were kept consistent in all experiments for fairness of comparison. Specifically, we stipulate that if less than the current batch size number of images are selected to be added to the training set in the next Self-Training iteration, the algorithm stops.

5.1.2 Ablation Study

The results are reported in Table 1, and illustrated in Figures 5, 3, 4 and 2. In our experiments we simply have the NNs predict the labels for the 54,500 unlabelled MNIST samples, and evaluate how well the system is doing at predicting the correct labels at the end of each Self-Training iteration. The evaluation is primarily considered in terms of the cohen's kappa statistic (κ) as it is more robust than accuracy by taking into account random luck, and the number of images left unlabelled after Self-Training. As can be observed from the results, the addition of our proposed inverse uncertainty weighting scheme improves the performance of the algorithm by leaving less images unlabelled, and achieving a higher κ score (DBST-1 to DBST-2).

We also test the effect of the quartile uncertainty thresholds for τ from Q3 to Q2 (DBST-2 to DBST-3), meaning we are more strict about which pseudo-labelled samples we can add to the training set. This only con-

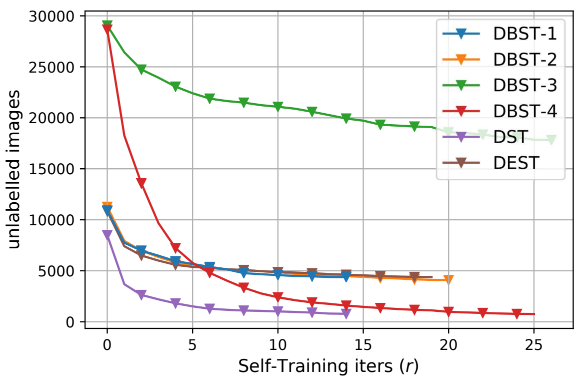
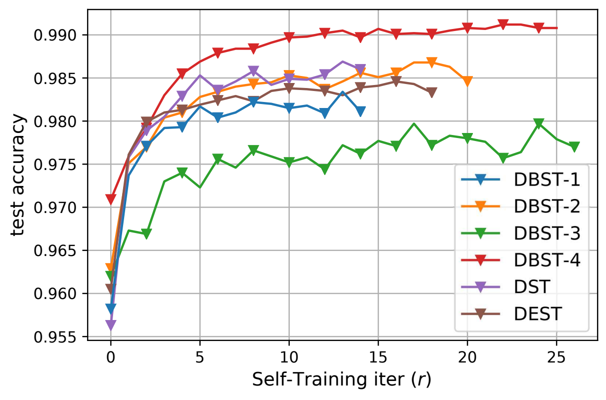
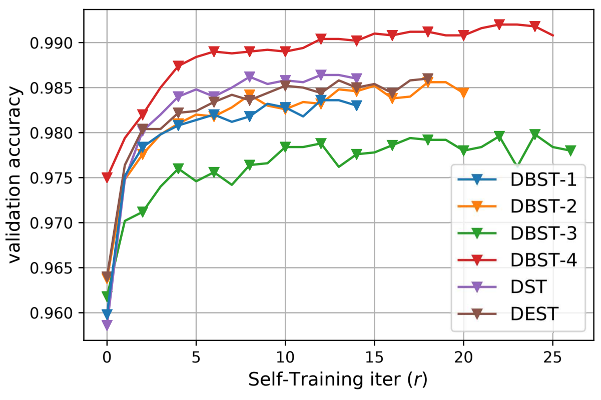


Fig. 2 Self-Training model comparisons regarding number of images left unlabelled after r iterations. Notice how the baseline Self-Training (DST) is overconfident by wrongly pseudo-labelling more samples early and propagating these errors, resulting in a lower cohen's κ score as reported in Table 1.

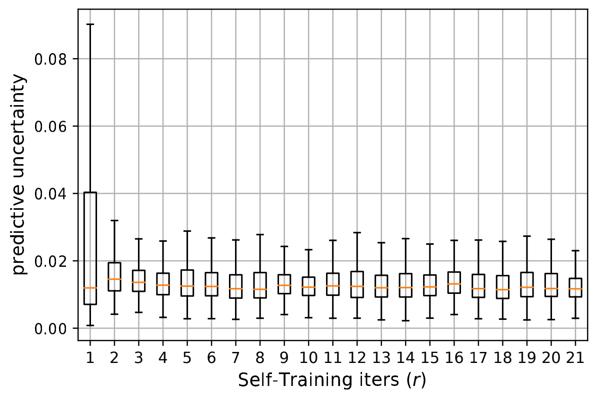


(a)

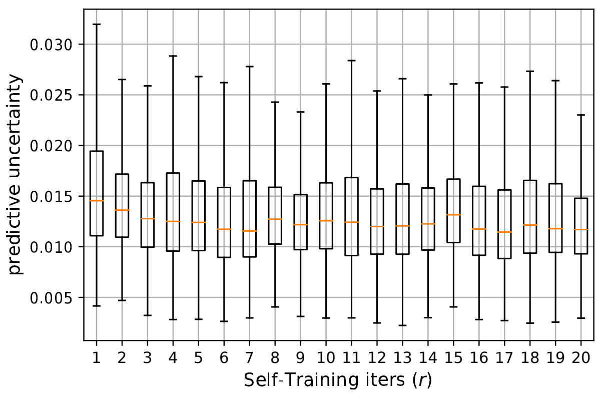


(b)

Fig. 3 Model performance comparisons over r Self-Training iterations. **(a)** MNIST test set performance after each Self-Training iteration. **(b)** As in (b) but comparing validation set performance. Notice that every model uses the same stop condition for fair comparison, but they stop at different times due to their uncertainty level. DBST-4 using both inverse uncertainty sample-weights and an entropy penalty on the log-likelihood loss ($\mathcal{L}_{\text{PNLL}}$), generalises better as reported Table 1.



(a)



(b)

Fig. 4 Box plots (IQR) depicting the quartiles for setting the uncertainty upper bound threshold threshold τ over r iterations in the DBST-2 model as an example. Note: these IQR stats are calculated using the predictive uncertainies of correctly classified samples in the train/valid/test sets only. **(a)** Shows all iterations ($r = 21$) whereas **(b)** omits the first one for better visibility.

siders very highly condident pseudo-label predictions resulting in a higher κ score, at the cost of labelling less examples as expected.

In the DBST-4 model, we combine both the sample-wise inverse uncertainty weighting scheme and the entropy penalty on the log likelihood loss ($\mathcal{L}_{\text{PNLL}}$) using $\beta = 1$ as described in section 3.5. As reported in Table 1, the number of examples left unlabelled is significantly less, whilst maintaining a good cohen's κ agreement between predicted and actual labels. In comparison to the others, the DBST-4 model provides the best balance between the number of unlabelled images left after self-training and a high cohen's κ score.

5.1.3 Comparitive Discussion

Lastly, we compare our Bayesian models (DBST) with two baseline method for estimating uncertainty in a similar way to [53], known as a Deep Ensemble of NNs (DEST), and the standard Self-Training (DST) following the logic in [31], and simply using the NNs predicted probability of an assigned pseudo-label as a level of confidence. The predictions from each NN in the ensemble (DEST) can be used as to calculate predictive uncertainty as the deviations capture model parameter uncertainty. Here we do not employ any bootstrap methods as the randomness from the NN weight initialisation and shuffled training has been shown to be sufficient ex-

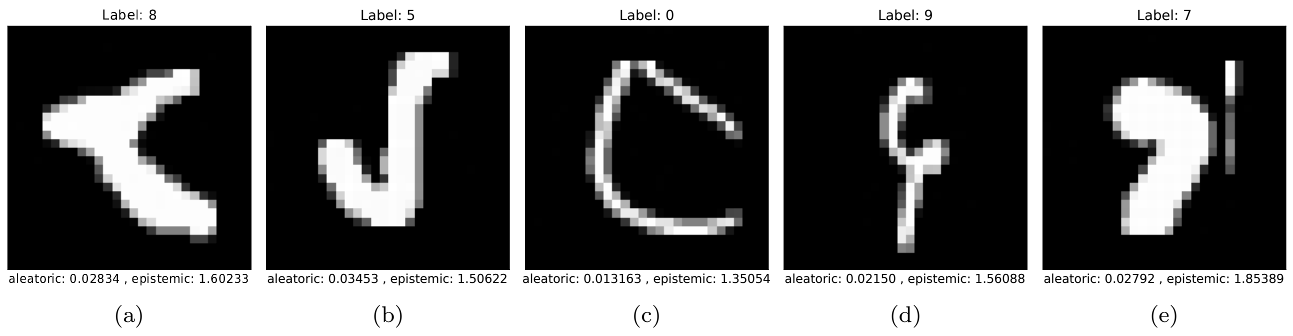

Label: 8
Label: 5
Label: 0
Label: 9
Label: 7
aleatoric: 0.02834 , epistemic: 1.60233
aleatoric: 0.03453 , epistemic: 1.50622
aleatoric: 0.013163 , epistemic: 1.35054
aleatoric: 0.02150 , epistemic: 1.56088
aleatoric: 0.02792 , epistemic: 1.85389
(a)
(b)
(c)
(d)
(e)

Fig. 5 Examples of images left in the unlabelled pool set for model DBST-2. Images with the highest epistemic uncertainty were selected for each digit class, along with their corresponding aleatoric uncertainties reported in the x-axis. The actual label of each image is found on top. As we can see from these difficult examples, these digits were automatically identified as problematic (too uncertain) in the DBST pseudo-labelling process, so they were not added to the training set $\mathcal{D}^*$.

perimentally [53]. We use the same DenseNet architecture, including related hyperparameters and identical dataset splits to train an ensemble of 5 models. Table 1 shows that our methods (DBST) are better than using an ensemble ($M = 5$) for predicting uncertainty for our Self-Training purpose, whilst taking approximately 5× less time to run in our experiments. Note that Monte Carlo Dropout samples are very cheap to compute at inference time compared to training multiple models, thus we can afford to take multiple samples i.e. $T = 30$ as compared to an ensemble of $M = 5$, which is also an advantage of our approach.

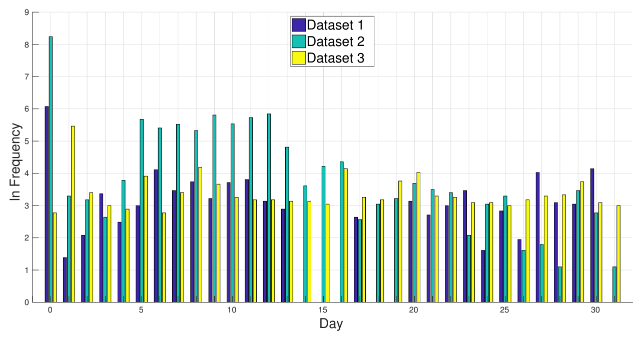
With regards to the vanilla Self-Training baseline (DST), again we use the exact same DenseNet architecture and related hyperparameters for fair comparisons. As previously outlined, in standard Self-Training we take the NNs predicted probability as a measure of confidence, and to demonstrate the inadequacy of this method we threshold with a very high confidence probability of .99. This simply means that only pseudo-label predictions above the .99 probability (confidence) threshold in a 10-way softmax (MNIST digit classes) are added to the training set. As reported in Table 1 and Figure 2, DST under performs compared to our methods since it is over confident early on, resulting in the addition of more wrong pseudo-labels to the training set thus propagating the errors forward. Although the number of images left unlabelled is low, the cohen's κ score is significantly lower

5.2 Real Datasets

Four datasets of food package photographs were collected by a leading food company and provided to us for research purposes. The four sets include 1404, 6739, 1154 and 13948 captured images respectively. In order to produce trainable datasets, a portion of the images was first manually annotated w.r.t. the presence of *use-by* dates, and lack thereof. In the case of unreadable images, in which dates were not discernible from the the background - potentially due to heavy distortion, non-homogeneous illumination or blur - were then set aside in a separate category. Conversely, images in which either day or month, or both were missing, were considered as incomplete, and subsequently grouped into their own category. Lastly, images of good quality, reporting the date including both the day and month, were considered as good candidates for OCV. The first three sets of images were annotated as mentioned above to form 5 categories: complete dates, missing day, missing month, no date and unreadable (Table 2), whereas photographs belonging to the fourth dataset were annotated as good or bad candidates for OCV, and utilised to test our proposed Bayesian self-annotating framework. After annotating all the images in the first three datasets, it was possible to plot some statistics (see Figure 6) on the frequency of specific dates within each dataset, and thus devise a methodology for conducting experiments with balanced sets of classes. Moreover, by inspecting the images with partially missing data, it was observed that most of them were photographs of package labels which had been folded at crucial points, included photographic glare, digits fainting over time, or included human made occlusions. With regard to the fourth dataset, 8931 images were annotated as including readable dates, and the remaining 5017 as unreadable.

5.2.1 Transfer Learning

It was of particular interest to conduct transfer learning in order to assess the adaptability of pre-trained CNN weights [54] on the current food datasets. Specifically, each image from our datasets was fed through a



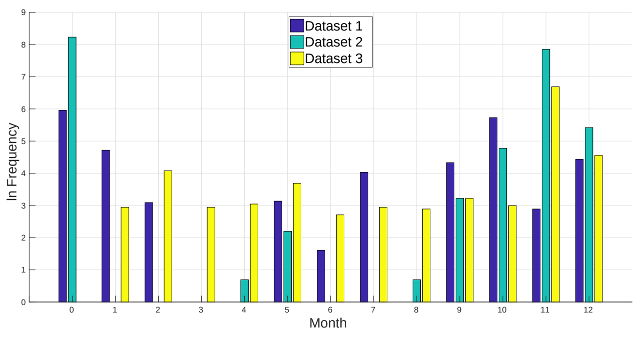


Fig. 6 **Left:** Frequency (ln scale) of appearance per 'Day' in *use-by* dates. **Right:** Respective appearance per 'Month'.

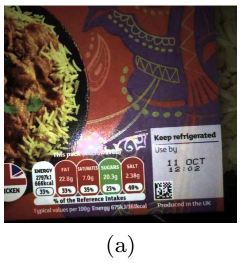
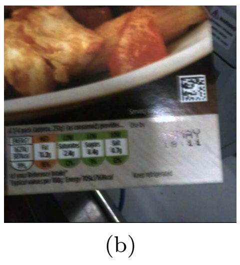
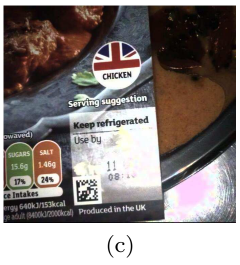
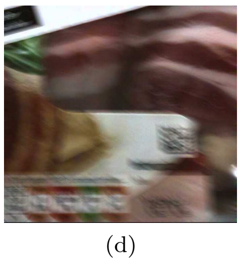
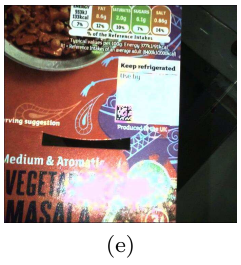






Fig. 7 Per category examples of images in our datasets. **(a)** Complete Date (day and month visible). **(b)** Partial Date (no day visible). **(c)** Partial Date (no month visible). **(d)** Unreadable. **(e)** No date (neither day or month visible).

previously trained InceptionV3 CNN on the ImageNet dataset, up to the last global average pooling (GAP) layer, where a 2048 dimensional vector representation of each instance was extracted. The 2048 dimensional vectors then became the input to a new series of FC layers and a final softmax layer able to predict N classes (see Figure 8). In order to optimise the training performance of the new FC layer network, a series of architectural decisions were made empirically, and the best performances were achieved using a FC network consisting of two 2048 unit hidden layers with Rectified Linear Unit (ReLU) activations and Batch Normalization (BN) [52] layers.

The risk of overfitting rises as the number of parameters increases w.r.t. number of training examples. Due to the limited amount of training data, available for experimentation, it is infeasible to train state-of-the-art models from scratch. Therefore, we introduced an effective regulariser in the new network as well as adapted previously learned low-level features through transfer learning. One of the most effective regularisation techniques is Dropout [55]. In practice, to preserve more information in the input layer $\ell^{(0)}$ (of L total layers) in the network and thus aid learning, the probability of keeping ($p(z^{(i)}) :\neq 0$) any given neuron $z^{(i)}$ in layer i was as defined per the following schema

$$\ell^{(i)} = \begin{cases} p(z^{(i)}) = 0.8 & \text{if } i = 0 \\ p(z^{(i)}) = 0.5 & \text{otherwise.} \end{cases} \tag{27}$$

In view of the unbalance present among the various classes, it was beneficial to use a weighted negative log-likelihood as a loss function (28). In (28), λ_j is a weight coefficient computed for the j^{th} of all classes J as a function of the proportion of instances N_j compared to the most densely populated class (29). During training, λ encourages the model to focus on under-represented classes

$$\mathcal{L}_{\text{NLL}} = -\sum_i \lambda_j \boldsymbol{y}_i \log(\widehat{\boldsymbol{y}}_i) - (1 - \boldsymbol{y}_i)\log(1 - \widehat{\boldsymbol{y}}_i) \tag{28}$$

calculating the per-class weight parameter λ_j with

$$\lambda_j = \frac{1}{N_j} \max\Big(\{N_i\}_{i=[1:J]}\Big). \tag{29}$$

In the case of multiclass classification, where $J > 2$, the weighted cross entropy loss function can be defined as

$$\mathcal{L}_{\text{NLL}} = -\sum_{i=1}^{M}\sum_{j=1}^{J} \lambda_j \boldsymbol{y}_{ij} \log(\widehat{\boldsymbol{y}}_{ij}), \tag{30}$$

where $\log p(\widehat{\boldsymbol{y}} = j|\boldsymbol{z}_j)$ is calculated as

$$\log \text{softmax}(\boldsymbol{z}_j) = \log\left[\frac{\exp(\boldsymbol{z}_j)}{\sum_k \exp(\boldsymbol{z}_k)}\right], \tag{31}$$

$\boldsymbol{z}$ is a vector of NN output logits, and M denotes the batch size of choice for stochastic optimisation of $\mathcal{L}_{\text{NLL}}$ via backpropagation. In all cases, we use Adaptive Moment Estimate (Adam) as an optimiser [56].

Table 2 Number of images per category in each dataset.

Annotation (DD/MM)	Dataset		
	1	2	3
Missing/Missing	375	3715	0
Missing/Complete	59	68	16
Complete/Missing	10	39	0
Complete/Complete	645	2847	1138
Unreadable	315	46	0

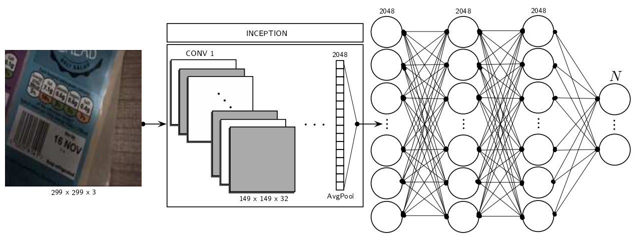


Fig. 8 Depiction of the classification architecture. From left to right, input images were resized to 299 × 299 × 3 to accommodate the CNN's convolutional layer parameters and arithmetic. There exist 2 hidden layers with 2048 units each and ReLu activations. The number of units N in the softmax layer was adjusted as per the number of classes being classified in different experiments.

In this framework, 3 sets of experiments were conducted and the obtained results are reported in Tables 3, 4. The goal of the **1st** experiment was to establish a baseline for images that would be classified as acceptable according to human standards. The appearance of unreadable images was especially prominent in the 1st of the three datasets. Conversely, the average image quality of the 2nd and 3rd datasets was higher, therefore they were not considered in this experiment. Moreover, the 1st dataset contained images from seven different locations, and as such, there were at least seven different types of food packaging present. To devise a balanced experiment, images from all locations were combined and categorised into 2 classes: 'Complete Dates' and 'Unreadable'. As reported in Table 3, **90.1**% classification accuracy was achieved over all seven locations. The **2nd** experiment aimed at distinguishing between acceptable and not-acceptable, missing dates. This meant that the absence of either day or month digits in a *use-by* date is not acceptable. The 2nd dataset was the largest, containing approximately 50% of examples with partial or missing dates. Images missing the day/month or both were assigned to one class and 'Complete Dates' to the other. As reported in Table 3, an accuracy of **96.8**% was achieved. Similarly, a performance of **94.8**% was achieved when applying the same procedure to the 1st dataset. As for the 3rd dataset, it includes images of higher quality, but there is a very small number of missing value examples available. To address this, we performed data augmentation in order to produce a larger set of 'Partial Dates'. The accuracy achieved on this synthetic set was **85.8**%. Lastly, a small variation of this experiment (2.1 in Table 3) was conducted in order to assess how well the network can identify the presence of any type of date, be it complete or partial, versus the absence of a date altogether. This experiment offered insight into how well the network can produce inferred localisation of dates, as it must learn to filter out the abundant non-date related text/numbers in the images. Table 3 shows that good accuracies were achieved across all three datasets, with the best case of **96.2**% date presence detection on the 2nd dataset.

In a brief **3rd** experiment, a global approach to OCV was tested by targeting the classification of specific digits and letters. Successful text recognition systems typically begin with the detection of text presence within a given image, followed by a segmentation or localisation of the desired region-of-interest (ROI) in order to perform classification of segmented digits thereafter. Here we assess how well the NN can perform without specifying any additional labels or local information. Given that almost all images in the 3rd dataset contained 'Complete Dates', we conducted a brief digit classification experiment (see Table 4 for results). Despite the small number of training examples (1138) and limited possible class combinations, four digit classes were identified, namely: 5, 8, 16 and 20. With these labelled examples, an accuracy of **90**% was achieved. Similarly for the 2nd dataset - due to limited data - a brief global OCV classification experiment between the the months of October and Novemeber in *use-by* dates was conducted. An accuracy of **92.7**% was achieved despite the small number of training examples. In reflection of these results, it is important to remember the great variety of text and numbers included in each image. Without providing any local knowledge and given limited training examples, the networks were still able to automatically infer the importance of specific digits and their respective locations in a global manner, whilst ignoring the same or other digits located in close proximity.

5.2.2 Latent Variable Adaptive Clustering

A major challenge spanning the three datasets was the high variability in the captured images characteristics. This variability made the reuse of a DNN trained on one dataset, for classifying the data of another, very difficult leading to poor performances. Fundamentally,

Table 3 Experiment results of OCV binary classification.

CNN Optical Character Verification				
Exper.	Dataset	OK	NOT-OK	Accuracy (%)
1	1	645	645	**90.1%**
2	1	645	444	89.3%
	2	2847	2847	**96.8%**
	3	577	577	85.8%
2.1	1	714	375	94.8%
	2	2954	2954	**96.2%**
	3	199	199	88.1%

Table 4 Experiment results for date character recognition.

CNN Date Character Recognition			
Exper.	Dataset	Images per Class	Accuracy (%)
3	2	381, 381, 381	**92.7%**
	3	55, 67, 63, 61	90%

this is because each dataset comes from a different distribution, as the images were taken by different people, with different cameras and at differing supplier locations. With limited data available to us, the use of transfer learning among different environments and datasets was ineffective. To overcome these challenges, we demonstrate the possibility of designing a new facet of the same CNN architecture, for learning each considered problem associated with different datasets. The approach focuses on: *i)* detecting bad image capturing conditions; *ii)* detecting missing dates (*i.e.* either day and/or month of *use-by* date); *iii)* showing the ability to recognise day and/or month of an existing *use-by* date. The CNN architectures proved to be quite accurate in identifying the missing/complete dates classification problem. Subsequently, we explored whether the respective trained networks were suitable for carrying out the proposed network adaptation approach (see Table 5 for results).

To this end, consider $\mathcal{F}(\mathcal{D}_2; \mathbf{W}_2)$ as a trained CNN with a test performance of 95.9% on a binary classification problem of *use-by* date verification on a real dataset. Let $\mathcal{T}_1$ be the test set of a dataset from a different distribution targeting the same classification task. We forward-propagate $\mathcal{T}_1$ through $\mathcal{F}(\mathcal{D}_2; \mathbf{W}_2)$ and achieve a lower accuracy of 63.8% as expected. We employed our adaptation procedure to classify $\mathcal{T}_1$ without any parameter retraining, decreasing the relative error by 34.81% with an improved accuracy of **76.4**%. Interestingly, the original performance achieved by $\mathcal{F}(\mathcal{D}_2; \mathbf{W}_2)$ on $\mathcal{T}_2$ also increased from 95.9% to **97.1**% when classifying $\mathcal{T}_2$ with $\mathbf{A}$ instead of the CNN it was originally trained on. Figure 9 depicts a 3D visualisation of all 2048-dimensional cluster centroids, for $k = 7$ for both datasets (14 in total). Squares (Red) and (Blue) crosses denote the centroids corresponding to the complete date class in the first and second datasets respectively. (Green) circles and (Pink) diamonds are the centroids in the missing date category and the (Black) stars indicate the centroids not used in the final classification as per the centroid exclusion policy explained previously in section 4.

Table 5 Experiment results of our adaptation procedure.

Latent Variable Adaptive Clustering		
Test Dataset	Classification Accuracy (%)	
	CNN $\mathcal{F}(\mathcal{D}_2; \mathbf{W}_2)$	Our Method ($\mathbf{A}$)
$\mathcal{T}_1$	63.8%	**76.4%**
$\mathcal{T}_2$	95.9%	**97.1%**

5.2.3 Deep Bayesian Self-Training on Real Data

In order to validate our approach, we conducted a series of experiments on a pool of held-out annotated data comprised of 11948 real food package images. The results can be seen in Table 6 and Figure 10. We begin by introducing Concrete Dropout layers after every convolutional layer in the last DenseBlock of a DenseNet-201, pre-trained on ImageNet. We then fine-tuned the last DenseBlock on a small portion of 500 images, with binary annotated labels representing whether the *use-by* date was readable (OK) or not (NOT-OK). As observable in Figure 10a, we first applied these ideas to the full set of unlabelled 11948 images and simply selected the 500 most certain predicted labels to be added to the initial training set of 500 images. This process was re-

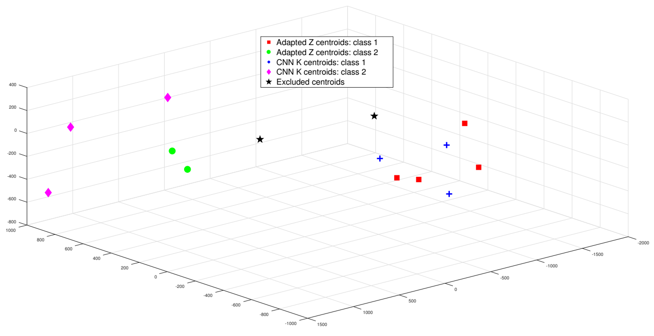
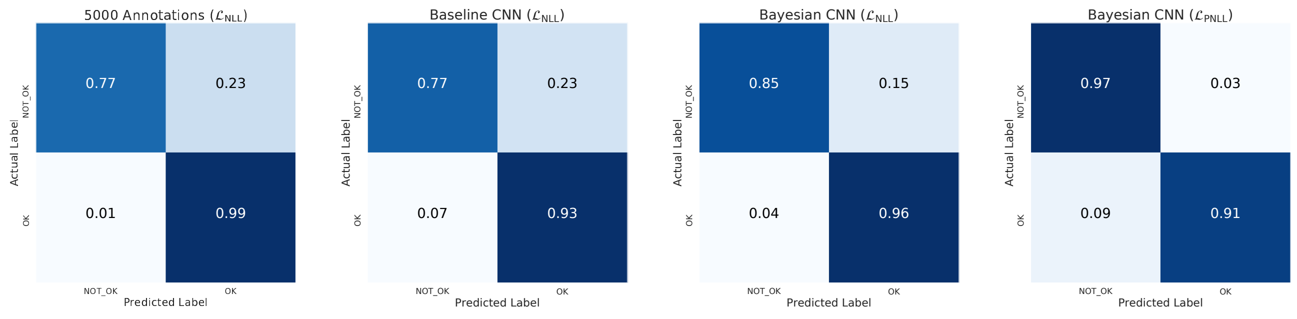


Fig. 9 t-SNE visualisation of the derived centroids **A** with best $k = 7$, achieving the results reported in Table 5. The 'Excluded centroids' (2 black stars) were removed as per the policy outlined in step 6 of our proposed adaptation procedure.



(a) (b) (c) (d)

Fig. 10 Normalised confusion matrices of the results obtained from our self-annotation procedure. **(a)** Refers to the 5000 predicted labels obtained with the lowest prediction uncertainty. **(b)** Deterministic baseline CNN predicted labels, wherein the thresholds were set based on the network's sigmoid output. **(c)** Predicted labels from our Bayesian Self-Training approach, trained with a standard binary negative log-likelihood loss. **(d)** Similar to (c) but using a Bayesian CNN trained with the entropy penalised binary negative log-likelihood loss.

peated 10 times in order to collect a total of 5000 images with predicted labels, which we then compared with our annotated labels as shown in Table 6. In the remaining set of experiments, instead of selecting a pre-determined number of images, we filtered out uncertain predictions based on a threshold τ as in Algorithm 1. Figures 10c and 10d depict the confusion matrices for the automatically annotated images w.r.t. true labels, and highlight the benefits of applying a confidence penalty on the log-likelihood loss ($\mathcal{L}_{\text{PNLL}}$), as opposed to using a standard log-likelihood ($\mathcal{L}_{\text{NLL}}$) which often outputs overconfident distributions. The uncertainties were calculated based on 50 Monte Carlo Dropout samples at test time, following the description in section 3.5. In order to compare our approach to standard Self-Training, we took the same network and datasets splits, and trained it without the Bayesian components. The threshold was set based on the confidence of the CNN output to only consider very confident predictions with over 0.999 predicted probability. As can be seen in Table 6, even with a high threshold, the deterministic CNN tends to be overconfident in its wrong predictions. This causes an increase of the propagated error as more images with wrong predicted labels are added to the training set and the model starts to under perform. To ensure a fair comparison between the Self-Training methods, the *stop* conditions were set to be identical s.t. the procedure was interrupted after 3 consecutive iterations without selecting more images to be added to the training set.

Table 6 Deep Bayesian Self-Training performance on real datasets. Cohen's Kappa score κ is also reported.

Bayesian CNN ($\mathcal{L}_{\text{PNLL}}$), $\kappa = \mathbf{0.8891}$				
Class	Precision	Recall	F1-score	#img
NOT-OK	0.9532	0.9694	0.9612	294
OK	0.9427	0.9136	0.9279	162
Avg./Total	0.9494	0.9496	**0.9494**	**456**

Bayesian CNN ($\mathcal{L}_{\text{NLL}}$), $\kappa = \mathbf{0.8383}$				
Class	Precision	Recall	F1-score	#img
NOT-OK	0.9679	0.8538	0.9073	212
OK	0.889	0.9764	0.9306	254
Avg./Total	0.9248	0.9206	**0.9200**	**466**

Baseline CNN ($\mathcal{L}_{\text{NLL}}$), $\kappa = \mathbf{0.6964}$				
Class	Precision	Recall	F1-score	#img
NOT-OK	0.9158	0.7682	0.8355	453
OK	0.7989	0.9287	0.8589	449
Avg./Total	0.8576	0.8481	**0.8472**	**902**

6 Conclusion & Future Work

In this paper we propose a Deep Bayesian Self-Training methodology that leverages modern approximate variational inference in DNNs to estimate predictive uncertainty during a Self-Training setting. Both aleatoric and epistemic uncertainties of predicted pseudo-labels for unseen data are estimated, and the samples with the lowest predictive uncertainty (highest confidence) are added to the training set in an automated manner. We offer ways to mitigate the known problem of propagating errors in Self-Training by including: (i) an entropy penalty on the log likelihood loss to punish over confident output distributions and facilitate thresholding, and (ii) an adaptive sample-wise weight on the influence of predicted pseudo-labelled samples over gradient updates to be inversely proportional to their predictive uncertainty. Lastly, we propose a new simple methodology for visualising and analysing variability between two dataset distributions in DNNs, and attempt to adapt information from one problem to the other by clustering learnt latent variable representations in the context of our application domain. An experimental study on both public and private (real) datasets is presented demonstrating the increased performance of our algorithm over standard Self-Training baselines, and also highlighting the importance of predictive uncertainty estimates in safety-critical domains.

Our future work will extend the experimental study to large dataset, consisting of about half a million real food packaging images, and we intend to apply the presented DNN based methodologies for adaptation and self-annotation of this data.

Acknowledgements The authors would like to thank Mr. George Marandianos, Mrs. Mamatha Thota and Mr. Samuel Bond-Taylor for manually annotating datasets used in this study and of course the reviewers for their constructive feedback that helped to improve the manuscript. We would also like to thank Professor Luc Bidaut for enabling this collaboration.
Funding: The research presented in this paper was funded by Engineering and Physical Sciences Research Council (Reference number EP/R005524/1) and Innovate UK (Reference number 102908), in collaboration with the Olympus Automation Limited Company, for the project Automated Robotic Food Manufacturing System.
Conflict of Interest: The authors declare that they have no conflict of interest.

References

1. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
2. Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017.
3. Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.
4. David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.
5. Lorien Y Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pages 204–211, 1993.
6. Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.
7. David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.
8. Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge & Data Engineering*, (11):1529–1541, 2005.
9. Simon Tong. *Active learning: theory and applications*, volume 1. Stanford University USA, 2001.
10. Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in Neural Information Processing Systems*, pages 8527–8537, 2018.
11. Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.
12. Amir R Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.
13. Fabio De Sousa Ribeiro, Francesco Calivá, Mark Swainson, Kjartan Gudmundsson, Georgios Leontidis, and Stefanos Kollias. An adaptable deep learning system for optical character verification in retail food packaging. In *Evolving and Adaptive Intelligent Systems, IEEE International Conference on*, 2018.
14. Simon Pearson, David May, Georgios Leontidis, Mark Swainson, Steve Brewer, Luc Bidaut, Jeremy G Frey, Gerard Parr, Roger Maull, and Andrea Zisman. Are distributed ledger technologies the panacea for food traceability? *Global Food Security*, 20:145–149, 2019.
15. Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.
16. Fabio De Sousa Ribeiro, Liyun Gong, Francesco Calivá, Mark Swainson, Kjartan Gudmundsson, Miao Yu, Georgios Leontidis, Xujiong Ye, and Stefanos Kollias. An end-to-end deep neural architecture for optical character verification and recognition in retail food packaging. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2376–2380. IEEE, 2018.
17. Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852. IEEE, 2017.
18. Piotr Chudzik, Somshubra Majumdar, Francesco Calivá, Bashir Al-Diri, and Andrew Hunter. Microaneurysm detection using fully convolutional neural networks. *Computer methods and programs in biomedicine*, 158:185–192, 2018.
19. Dimitrios Kollias, Miao Yu, Athanasios Tagaris, Georgios Leontidis, Andreas Stafylopatis, and Stefanos Kollias. Adaptation and contextualization of deep neural network models. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2017.
20. Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical image analysis*, 35:18–31, 2017.
21. Francesco Caliva, Fabio De Sousa Ribeiro, Antonios Mylonakis, Christophe Demaziere, Paolo Vinai, Georgios Leontidis, and Stefanos Kollias. A deep learning approach

to anomaly detection in nuclear reactors. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018.
22. Fabio De Sousa Ribeiro, Francesco Caliva, Dionysios Chionis, Abdelhamid Dokhane, Antonios Mylonakis, Christophe Demaziere, Georgios Leontidis, and Stefanos Kollias. Towards a deep unified framework for nuclear reactor perturbation analysis. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2018.
23. Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.
24. Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.
25. Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017.
26. Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.
27. Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 4, 2017.
28. Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.
29. Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017.
30. Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–152, 2018.
31. Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.
32. Yves Grandvalet and Yoshua Bengio. Entropy regularization. *Semi-supervised learning*, pages 151–168, 2006.
33. Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 289–305, 2018.
34. Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.
35. John S Denker and Yann Lecun. Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859, 1991.
36. Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.
37. David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.
38. Yarin Gal. Uncertainty in deep learning. *University of Cambridge*, 2016.
39. Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.
40. Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *arXiv preprint arXiv:1705.07115*, 2017.
41. Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.
42. Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.
43. Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.
44. Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.
45. Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
46. Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.
47. Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems*, pages 3581–3590, 2017.
48. Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.
49. Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation. *Medical Imaging with Deep Learning*, 2018.
50. David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
51. Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1175–1183. IEEE, 2017.
52. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
53. Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.
54. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

55. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
56. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.