Surprisingly Simple Deep Confidence Prediction

Anonymous ECCV submission

Paper ID ***

Abstract. Computing confidence estimates that are representative of the true uncertainty for a prediction is an important and yet unsolved problem in deep learning. Despite significant improvements in using neural networks to perform complex predictions, there is still no unified approach to compute a calibrated confidence measurement in non-Bayesian deep models that can capture the true probability of success associated with its predictions. In this paper, we propose to use a multi-task learning scheme to simultaneously learn to perform a given classification or segmentation task as well as output the probability that its prediction for each instance of the task is correct. The confidence is computed at the instance level using Bayesian inference with quantized distributions. Since predicting success or failure on a given task is inherently related with the confidence in such a prediction, jointly optimizing predictions as well as confidences in a multi-task setup helps in the calibration of uncertainty across training inputs. Furthermore, our approach is able to generalize confidence prediction by associating certain types of inputs with higher or lower levels of confidence. We validate our method on image classification and semantic segmentation benchmarks. [1]

Keywords: multi-task learning, uncertainty, calibration

1 Introduction

Deep neural networks have achieved significant success in domains such as computer vision, speech recognition, and natural language processing. They are also used as a major component in decision making pipelines in sensitive applications such as medical diagnosis [1, 2], autonomous driving [3, 4], and economic forecasting [5]. However, despite the impressive performance of deep neural networks at vision tasks, they do not provide direct *confidence* measurement of their predictions in either the classification or regression regime [6]. Well-calibrated probabilities of failure are essential for sensitive decision-making processes and are also valuable in other settings. For instance, predictive uncertainty can be used to boost performance either in supervised learning problems by down-weighting low-confidence predictions [7, 8] or in the reinforcement learning regime by using an uncertainty-dependent cost function to avoid taking dangerous actions [9]. Also, predictive uncertainty estimates can be directly incorporated in active

[1] Our code will be available upon camera-ready publication.

learning-based data collection pipelines that prevent collecting redundant data if the uncertainty is expected to be below threshold.

This paper presents a simple multi-task learning (MTL) method to model confidence followed by a Bayesian inference that determines the true probability of success, which outperforms recent state of the art baselines. Our goal in this paper is quantifying confidence, the true correctness likelihood at any given instance. We define *learning when we succeed* as the extra task that we wish to learn in parallel to the main task. Our approach can be used in both Bayesian and non-Bayesian frameworks, in both classification and regression settings. The main task can be any task that needs to be learned with supervision. The performance prediction is the extra task that we believe the required information to learn it is mutually inclusive with that needed for learning the main task. We show below how the results can be superior to the single-task learning (STL) regime. MTL can be implemented simply using a multi-task loss function with contemporary deep architectures, with one or more additional layers in each network path for each loss. Once the training completes, we use the outputs of the second task to compute the failure probability using Bayes' rule on binned probabilities.

The paper consists of the following sections: In Section 3 we review our method in detail and explain how we use multitask learning to compute confidence. In Section 4 we describe baseline methods, including the best currently existing calibration tool that scales the raw softmax probabilities to their calibrated counterparts. In Section 5 we show our performance on two well-established problems in computer vision: image classification (sec 5.1) and semantic segmentation using fully convolutional networks [10]. We will show adding performance prediction as an extra task will cause the MTL model to surpass its STL equivalent in image classification. In Section 5.2 we use MTL with FCNs on Cityscapes [11] and Semantic Boundaries Dataset, (SBD), and demonstrate superior performance regarding predicting successful labeling of segmented pixels.

2 Related Work

Model uncertainty acquisition, sometimes referred to as confidence modeling, in deep and/or non-linear learning frameworks has been studied from different prospectives. Bayesian neural networks are perhaps the most popular framework for this purpose [12, 13]. They often use Markov Monte Carlo methods for small networks but can be too expensive to be implemented at scale. In order to mitigate the computational limits of using Bayesian methods, faster solutions were developed over the years including ”distilling” a Monte Carlo posterior density approximation to a single deep neural network [14] which surpassed its precedents [15, 16] which used expectation propagation and variational Bayes, respectively. However, even these faster approximations do not provide a complete representation for uncertainty mainly because they rely on simplifications to reduce the computational barriers leading to suboptimal posterior uncertainty distributions. [17] casted droupout [18] as approximate Bayesian inference at test time. This method did not require modifications in the training pipeline and was fast and easy to implement. Within Bayesian framework, [19, 6] defined two types of uncertainty in computer vision by placing prior distributions over model parameters and model outputs to obtain posterior distributions for calculating epistomic and aleatoric uncertainty, respectively. Nevertheless, despite the mathematically grounded framework and the applicability for both regression and classification settings, most Bayesian neural networks are as-yet intractable for contemporary large-scale computer vision tasks.

Another approach to quantify model confidence is to use post-processing techniques to calibrate output probabilities of existing models. This approach works for classification problems and has been shown to be effective in calibrating softmax scores to reflect the true correctness likelihood. The main advantage of these methods is they directly output a scalar value for confidence; they are easy to implement and computationally inexpensive to compute. Methods in this paradigm can be largely divided into two categories, parametric and non-parametric approaches, depending on the method they use to obtain the mapping between raw softmax probabilities and their calibrated counterparts.

Various binning techniques, which fall into non-parametric approaches, have been used to find this mapping using different partitioning methods. Histogram binning [20] sorts out the uncalibrated probabilities into mutually exclusive bins with a predefined calibrated score assigned to each bin. This score will be considered as the calibrated score at test time, once a prediction falls into a bin. The bins are commonly chosen to be equisized or to have equal number of samples inside. The predefined calibrated score is also chosen to minimize the squared loss per bin. One particular issue with this method is choosing the bin size as it significantly changes the results. Also, the bin size remains fixed over all predictions [21]. A general version of histogram binning, called isotonic regression, was introduced soon after in which the bin size and calibrated score were jointly optimized. [21]. An extended version of histogram binning, called Bayesian Binning into Quantiles (BBQ), was introduced as a binary classifier calibration method that explores multiple different binning and their combination to yield more robust calibration scores [22]. As opposed to histogram binning and isotonic regression which return a single binning layout, BBQ searches over all possible layouts on a held-out validation set and performs Bayesian averaging of the obtained probabilities generated by each layout.

Within the parametric paradigm, Platt's scaling is perhaps the most widely used approach [23] which learns parameters of a sigmoidal transformation function in the context of a neural network as the mapping function trained on a held-out validation set. An extension of Platt's method was recently introduced by [24] which uses a single scalar parameter, temperature T, in softmax function to calibrate the output probabilities. T is optimized with respect to negative log-likelihood on a validation set. It should be noted that neither of these methods have any influence on the model's accuracy.

The idea of *predicting* the performance of a model is itself not new and has been around in various communities. Inspired by "meta-cognition" [25], cognitive scientists refer to the task of performance prediction and analyzing the post-recognition scores as ”meta-recognition” [26]. Their work is based on thresholding the similarity score between a test case and the training data. They require to use the output of the original model on an input to make the match/non match decision while this will be a limitation if the original model is computationally expensive. In our method, learning the extra task (performance prediction) is added to the original model in the simplest way such that the extra computation needed is negligible. Once the performance predictor learns its task during training, it will be able to provide the performance measurement using only the given input to the original model at test time.

A similar approach has been also taken by [8] where they created a so called ALERT system by training a supporting vector machine classifier that predicts whether the original model, so called BASESYS, fails at a given task. The learning occurs in a supervised learning fashion where the data consists of BAYESYS' input (images, video, etc) paired with the BAYESYS's evaluation metric (either loss or accuracy) on the corresponding input. By thresholding the output of the SVM classifier the failure/success decision is made. Our method differs from this work in the following way: we use multitask learning and train a performance predictor along with training the model itself. This way the performance learner has the opportunity to be exposed to all the mistakes that the model has made and hence is able to generalize better as opposed to waiting for the original model to become an expert on training data leaving no room to learn its failure cases.

We frame our method within a transfer learning paradigm, which we implement using contemporary deep learning frameworks. MTL is an well-known inductive transfer method that improves generalization accuracy on the main task by using the information contained in the training signals of other related task(s) [27]. It does this by learning the extra task(s) in parallel with the main task while using a shared representation and a joint optimization; what is learned for each task can help other task(s) be learned better. (See [27, 28] for additional discussion about MTL).

3 Approach

We propose a model for quantifying the probability of success, or confidence, when using deep neural networks as a prediction function at an instance level. Given a neural network trained to predict classification or segmentation labels corresponding to an input instance, we want to determine whether or not and with what probability its prediction is correct. Considering confidence prediction as an auxiliary task trained along with classification or segmentation, our proposed model learns when its prediction fails during training. After training, we use the raw softmax probabilities of this performance predictor followed by a Bayesian inference to compute the true probability of success. In the next section, we describe our proposed auxiliary performance prediction in more detail.

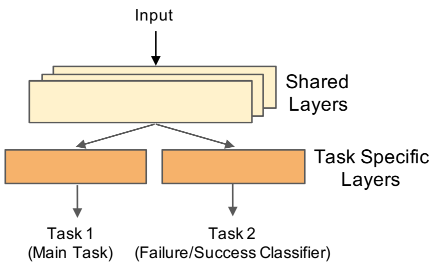

Fig. 1. A general multi-task model configuration to learn failure/success as an auxiliary task along with the main task

3.1 Performance Prediction

In a regular supervised learning problem with training data consisting of N instances $\{x_1, \cdots, x_N\}$ paired with their ground-truth labels $\{y_1, \cdots, y_N\}$, the *task* is to find a function mapping each arbitrary input x_i to its corresponding output y_i where input instances can be images, text, video frames, etc. In the multi-task setting with an additional auxiliary task to be learned, a simultaneous training of the two tasks reinforces the training capacity for each individual if they are *related*. This mainly occurs due to the useful information in training signals of the auxiliary task [27, 28]. In a deep network framework, the more conceptually *related* the two tasks are, the more layers they can share implying that the representation learned for the secondary task is more beneficial to the main one.

In this paper, mapping the main task's classification or segmentation predictions, $\hat{y}$, to one of the *failure or success* categories is the additional task we wish to learn. Given every (x_i, y_i) training pair, the $\{0, 1\}$ ground truth label for this binary classification problem can be generated by comparing $\hat{y}_i$ with its corresponding true label y_i. Figure 1 illustrates a general multi-task model configuration that trains this binary classifier head sharing the representation layers with the main task. Depending on how tasks are related to each other in a multi-task learning setting, a few hidden layers, between the shared representation layers and the final head, can be trained disjointly for each individual task, so called task specific layers. Since the main classification or segmentation task is inherently related with the failure and success predictions, we found out that only one hidden layer suffices to be unshared between the two tasks.

The objective for our proposed performance prediction head is a cross-entropy loss function which will be added to the main task loss term. Network parameters are then updated through back-propagation from the total additive loss.

3.2 Conditional Probability of Success

Jointly training of the main task and our performance predictor reinforces the prediction of each head of our deep network architecture to get closer to its corresponding ground truth label. On the other hand, our main goal is to train a performance predictor that is able to predict both failure and success instances correctly requiring equal number of examples per each binary category. However, as the main task is improved during training, fewer failure examples will be generated, i.e., the model will be biased toward predicting *success* more frequently. We therefore stop training of the performance predictor in earlier stages to have a more desirable balanced binary data distribution. However, our experiments reveal that pursuing our multi-task training process improves the performance for the main classification or segmentation tasks. As a result, we checkout the performance predictor model early in the training process in order to calculate the probability of success for each given instance at inference. In summary, we save network parameters as checkpoints in two different training states: checkpoint one includes the auxiliary task parameters early in the training procedure while checkpoint two contains the well-trained multi-task model parameters, i.e., model with the least measured error on the training set.

Estimating the probability of success in classification or segmentation label predictions for each given instance at inference time requires computing the conditional probability of success given the performance predictor softmax scores, $P(\text{Success}|\text{score})$. We acheive this conditional probablity from our training examples by dividing the interval for the *probability of success* $[0, 1]$ into M interval bins (each of size $1/M$). Performing inference on the training set using the secondary task paramters saved as checkpoint one, we output failure or sucess binary label predictions along with their raw softmax probabilities. For each data point with a 0/1 prediction label, its corresponding predicted softmax score falls into one interval $I_m = [\frac{m-1}{M}, \frac{m}{M})$ and thus fills the m-th bin up with one index. Therefore, all or some of the bins will be finally filled with the number of instances whose success probability scores have fallen into that bin. Given the failure or success label predicted for each data point at training time, we normalize the number of failure and success instances at every bin, and hence compute the conditional softmax probability scores as $P(\text{score}|\text{Failure})$ or $P(\text{score}|\text{Success})$. Employing Bayes' Rule will result in $P(\text{Success}|\text{score})$, indicated by $P(S|s)$:

$$P(S|s) = \frac{P(s|S)}{P(s|S) + \frac{P(s|F)P(F)}{1-P(F)}},$$

where S and F denote "Success" and "Failure" predicted by our performance predictor head and s is the softmax probability associated with that prediction.

3.3 Confidence Evaluation

In section 3.2 we defined how we measure confidence or probability of success using our MTL model. Here we discuss what a good confidence modeling system is and what metrics can be used for evaluation. An ideal confidence modeling system is expected to be confident (outputs high PoS) for correct predictions and uncertain (outputs low PoS) otherwise. We have set a threshold of 0.5 for the PoS below which the confidence is low. We also consider evaluating the performance

on two different testset distributions: in-domain; a never-seen before dataset but sampled from the same distribution as training, off-domain; a never-seen before dataset that is drawn from a completely different distribution.
We use three metrics to evaluate confidence score:

1. **Accuracy**: computed as (tp+tn)/total and presents the sum of confident successful and unconfident wrong predictions over the total predictions.
2. **Precision**: computed as tp/(tp + fp) and denotes how often we successfully predict with high confident over total number of high confident wrong/correct predictions.
3. **F-score**: We also use F-score to combine recall and precision together with assigning double weight on precision. This is due to the fact that we think a model that is more often highly confident despite being wrong (low precision) can be more catastrophic than a model that is less often confident despite being right (low recall).

4 Baseline: Temperature-scaled Confidence Calibration

Modern neural networks tend to produce high softmax probabilities on data inputs at test time, even on a new distributions that they have not seen before [29, 24]; uncalibrated probabilities may not represent the true probability of success at prediction time. Calibrating them to the proper scale is a popular way of confidence modeling at instance level [20–24]. Recent work [24] introduces a simple optimization technique to compute a *Temperature* parameter of the softmax function so that it alters the outputted probabilities such that they become calibrated. Using various benchmarks of image classification, [24] reported state-of-the-art calibration results compared to previous baselines, hence, we used this method as our baseline.

For completeness, we summarize the *Temperature Scaling* method [24] here: the method extends *Platt* scaling, and finds a single scalar parameter T for all classes in the softmax probability function acting on the logit vector $\mathbf{z}_i$. The new confidence score ($\hat{q}_i$) for class k will be then computed as:

$$\hat{q}_i = \max_k \sigma_{SM}(\mathbf{z}_i/T)^{(k)}$$

It should be noted that despite the simplicity of these calibration methods, they do not affect the model's accuracy on the main task. They also can perform poorly when tested in out-distribution data (See Fig. 2 for images and Table 4 for predictions). In contrast, our model is able to not only improve the performance on the main task, but also—by using a simple Bayes' rule on the outputted probabilities of success/failure—is able to compute a more accurate true probability of success.

5 Experimental Evaluation

We have implemented our multitask learning technique on various benchmark image classification architectures as well as fully-convolutional networks for se-

Table 1. Top1 %error rate for Task 1 using STL and MTL (Task1) models on CIFAR10/100 for various model architectures. The number following a model's name is the network depth

Dataset	Model	STL		MTL (Task 1)	
		%Error	Parameters(M)	%Error	Parameters(M)
CFR-10	ResNet 101	4.94 ± 0.02	42.51	$\mathbf{4.69 \pm 0.03}$	42.52
CFR-10	ResNet 18	5.19 ± 0.04	11.17	$\mathbf{4.96 \pm 0.01}$	11.17
CFR-10	DPN 92	5.25 ± 0.04	34.24	$\mathbf{4.96 \pm 0.02}$	34.24
CFR-10	DenseNet 121	$\mathbf{4.94 \pm 0.03}$	6.96	4.98 ± 0.04	6.96
CFR-10	MobileNet_V2	6.29 ± 0.03	2.30	$\mathbf{6.10 \pm 0.03}$	2.30
CFR-100	ResNet 18	25.03 ± 0.04	11.22	$\mathbf{24.81 \pm 0.06}$	11.22
CFR-100	ResNext 29_2x64	23.44 ± 0.05	9.13	23.45 ± 0.01	9.22
CFR-100	DenseNet 121	$\mathbf{22.95 \pm 0.05}$	6.96	23.19 ± 0.06	7.05
CFR-100	MobileNet_V2	25.17 ± 0.04	2.41	$\mathbf{24.91 \pm 0.03}$	2.41

mantic segmentation. Implementation details are given for each example as well as an evaluation on both in-distribution and out-of-distribution data.

5.1 Image Classification

We have used CIFAR10 and CIFAR100 datasets. They both consist of 50000/10000 training and test color images of 32×32 with 10/100 classes, respectively. Our goal is to train a performance predictor along with the image classifier to output confidence values in predictions. In the following section we will discuss our results and observations on confidence prediction in image classification.

Improving main task performance As discussed in section 3.1 multitask learning often helps with improving the main task if the two tasks are related. Our results of implementing multi-task learning for confidence prediction are all shown in Table 1. We have consistently used the same fashion of head insertion for all architectures; adding one single hidden layer to perform a binary classification with softmax nonlinearity. This results in slightly more number of parameters in some architectures compared to their STL counterpart. A comparison is made between the number of parameters in Table 1. It can be seen that the slight increase in number of parameters which not only results in outperforming the state-of-the-art models, but also enables us to obtain the confidence score, is well worth it. Results shown in Table 1 are averaged across 5 runs for both CIFAR10 and CIFAR100 datasets.

In/out-distributional data We use the pretrained models presented in Table 1 and we evaluate on two types of testset distributions representing in-domain and off-domain data. For testing CIFAR10/100 in-domain we use CIFAR10/100

Table 2. CIFAR-10 confidence prediction on in-domain (CIFAR10 testset) and off-domain (Down-sampled ImageNet dogs [30]) data distributions

Testset	Model	STL			STL scaled[24]			MTL		
		Acc	Pr	F1	Acc	Pr	F1	Acc	Pr	F1
ImgNt	ResNet18	27.62	0.24	0.37	43.54	0.26	0.39	**45.98**	0.26	**0.41**
CFR10		94.61	0.95	0.96	94.61	0.96	0.96	**95.02**	**0.97**	**0.98**
ImgNt	DPN92	39.40	0.98	0.55	41.5	0.39	0.55	**42.87**	**0.46**	**0.63**
CFR10		94.83	0.95	0.97	94.81	0.95	0.97	**95.89**	**0.97**	**0.98**
ImgNt	DenseNet121	25.65	0.21	0.35	29.03	0.22	0.36	**33.23**	**0.35**	**0.42**
CFR10		95.20	0.95	0.98	95.22	0.95	0.98	**96.40**	**0.97**	0.98
ImgNt	MobNet_v2	32.57	0.29	0.45	35.10	0.30	0.46	**38.45**	**0.41**	**0.60**
CFR10		94.15	0.94	0.97	94.19	0.94	0.97	**96.02**	**0.98**	**0.98**

Table 3. CIFAR-100 confidence prediction on in-domain (CIFAR100 testset) and off-domain (ImageNet fox) data distributions

Testset	Model	STL			STL scaled[24]			MTL		
		Acc	Pr	F1	Acc	Pr	F1	Acc	Pr	F1
ImgNt	ResNet18	28.66	0.00	0.00	85.70	0.00	0.00	**57.55**	**0.34**	**0.35**
CFR100		80.37	0.81	0.88	79.91	0.91	0.86	**86.29**	**0.90**	**0.91**
ImgNt	ResNext 29_2x64	37.23	0.00	0.00	89.66	0.00	0.00	70.21	**0.36**	**0.38**
CFR100		79.16	0.79	0.87	80.08	0.89	0.86	**85.14**	**0.91**	**0.91**
ImgNt	DenseNet121	30.36	0.00	0.00	87.14	0.00	0.00	**43.45**	**0.34**	**0.35**
CFR100		80.61	0.81	0.88	81.51	0.90	0.88	**86.34**	**0.93**	**0.92**
ImgNt	MobNet_v2	38.84	0.00	0.00	92.91	0.00	0.00	**53.92**	**0.25**	**0.27**
CFR100		79.80	0.77	0.85	78.32	0.92	0.85	**83.76**	**0.88**	**0.95**

testsets and for off-domain, we use images of "dog" and "fox" classes from down-sampled ImageNet dataset [30], respectively. We used class numbers 86, 87, 114, 149, 202, 253 to collect 7800 images of class "dog" and class number $1, 53, 62, 67, 159, 207$ for total of 7160 class "fox" images.

For comparison, we use two other models to compare against. First, the most naive approach of using raw output probabilities of the final softmax layer as a measurement of confidence. We refer to his method as STL here. The second model, which is our baseline [24], is using the temperature calibration tool on top of STL model to calibrate the raw STL probabilities to be representative of the model's confidence (PoS). It will be denoted as STL-scaled model [24].

Table 2 and 3 demonstrate the confidence prediction results using STL, STL-scaled [24], and MTL models on CIFAR10 and 100, respectively. On image classification task with a relatively small dataset such as CIFAR10 with low number of classes, our model is able to often surpass the baseline and sometimes performs the same. However, on a slightly more complicated task where number of classes

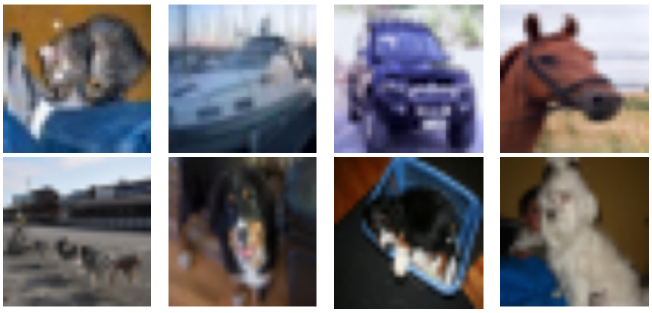

Fig. 2. Top row: random images from CIFAR-10 testset, left to right: cat, ship, car, horse. Bottom row: random images from down-sampled ImageNet, left to right: all dogs

increases (CIFAR100), STL and scaled-STL [24] lose their performance on off-domain dataset by not being able to assign low probabilities to miss-classified predictions and high probabilities to correct ones using ResNet18, ResNext29, and MobileNet. In contrast, MTL keeps being able to not only perform better on the main image classification task, but also it is able to detect its failures by assigning them low confidence score as well as being confident in the correctly classified "fox" instances. While outperforming the baseline on both in/off domain distributions, MTL appears to be significantly useful on confidence predictions on off-domain datasets where all other methods completely fail at. This is in fact what happens in the real world applications. At test time, there is usually very little control on distributions. Being able to model confidence appropriately is significantly important in AI safety community [31].

It should be noted that the accuracy shown in Tables 2 and 3 should not be confused with the accuracy of the main task (1.0 - Top1% error) shown in Table 1). Accuracy on the main task cannot be altered by calibration tools (such as the baseline's method) whereas our model is capable of increasing it (sec 5.1).

Some qualitative results from inferring a pre-trained ResNet18 with CIFAR10 on in/off-domain data distribution are depicted here. In Fig. 2 there are shown randomly selected imaged from CIFAR10 test set (top row) and down-sampled ImageNet [30] (bottom row). Table 4 and 5 show the predicted class for each image along with its confidence score. The 'horse' class gets misclassified by all models but it is MTL that knows this mistake will occur, therefore tags it with confidence score of 0.0 whereas both STL and STL-scaled [24] seem to be confident in their mistake with PoS = 0.6-0.7.

5.2 Semantic Segmentation

Here we discuss using MTL in semantic segmentation. Our goal here is to model confidence in segmentation predictions per pixel. We have used two datasets: Cityscapes [11] and SBD [32]. Cityscapes is a dataset with focus on semantic understanding of urban scenes. It includes 2975/500 color images in training/test

Table 4. Confidence scores predicted by STL, STL-scaled [24], and MTL, pre-trained on CIFAR10, on example images drawn from CIFAR10 testset shown in Fig. 2, top row

STL raw/scaled STL [24]	Cat	Ship	Car	Dog
Raw STL PoS	1.	0.99	0.99	0.75
Scaled PoS [24]	1.	0.99	0.99	0.62
MTL (ours)	Cat, Success	Ship, Success	Car, Success	Dog, Failure
MTL PoS (ours)	0.99	1	1	0.00

Table 5. Confidence scores predicted by STL, STL-scaled [24] , and MTL, pre-trained on CIFAR10, on example images drawn from ImageNet dogs shown in Fig. 2, bottom row

STL raw/scaled STL [24]	Ship	Cat	Cat	Cat
Raw STL PoS	0.88	0.99	0.99	0.69
Scaled PoS [24]	0.78	0.99	0.99	0.66
MTL (ours)	Ship, Failure	Dog, Success	Cat, Failure	Cat, Failure
MTL PoS (ours)	0.02	0.98	0.15	0.22

set with 19 classes per pixel. We scaled the images to size 256×256. SBD is a widely used dataset for image segmentation containing annotations from 8829/1449 in training and test set with 21 number of classes per pixels.

For our segmentation model we have used fully-convolutional networks [10] throughout our experiments. We have added an extra head (a hidden layer with two number of classes) for all the classification heads in the architecture resulting in less that 0.001% increase in the number of parameters. We have used a Pytorch implementation of FCN provided by [33] for our experiments which differs from the original Caffe implementation of FCN. Due to this difference, the results for mean IoU and mean accuracy differ from the original implementation. However, it is still a fair comparison to compare our MTL model against the STL and STL-scaled [24] models using the same base implementation.

Improving main task performance Similar to the image classification regime, we have been able to increase the performance of the model on the main task using MTL compared to the original STL model. Results for the maximum achieved mean IoU and accuracy over 100 epochs are tabulated in Table 6 and plotted in Fig. 3. On SBD, we have unarguably outperformed the STL performance using both evaluation metrics while on Cityscapes, MTL and STL performance match together. Note that this is just a side benefit of MTL method and is not guaranteed that it boosts the performance on all tasks. However it is extremely useful when it occurs considering the fact that confidence calibration tools have no control over model's accuracy.

Table 6. Maximum mean IoU and mean accuracy achieved by training FCN8s using STL and MTL on Cityscapes and SBD datasets

	Cityscapes		SBD	
	STL	MTL (Task1)	STL	MTL (Task1)
Mean IoU	0.36	0.36	0.59	**0.61**
Mean accuracy	0.45	0.45	0.77	**0.81**

Table 7. Confidence score prediction on semantic segmentation with FCN8s using STL, STL-scaled [24], and MTL methods, trained on entire SBD or Cityscapes. Testing cross-domains has been done on cars only

Trainset	Testset	STL			STL Scaled [24]			MTL		
		Acc	Pr	F1	Acc	Pr	F1	Acc	Pr	F1
CScapes	CScapes	16.30	0.31	0.32	16.44	0.41	0.43	**77.10**	**0.63**	**0.58**
	SBD Cars	48.30	0.08	0.09	20.6	0.21	0.24	**53.51**	**0.61**	**0.63**
SBD	CScapes Cars	48.60	0.05	0.16	49.39	0.23	0.22	**56.78**	**0.64**	**0.65**
	SBD	54.17	0.08	0.10	20.6	0.21	0.34	**57.01**	**0.65**	**0.63**

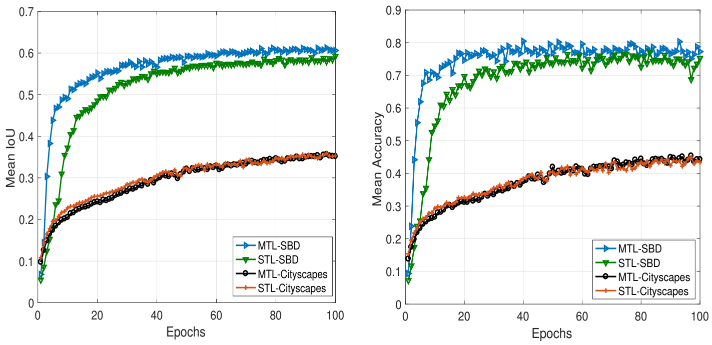

Fig. 3. Mean IoU and mean accuracy per epoch for training FCN8s on Cityscapes and SBD datasets using STL and MTL models

In/out-distributional data : There are few common classes between Cityscapes and SBD datasets. For the off-domain dataset we have chosen one of these common classes (car) to test performance cross domains. Results for MTL performance against naive STL and STL-scaled [24] are presented in Table 7. On segmentation confidence prediction, MTL outperforms the baseline in all the evaluation metrics on both in/off domain data distributions. It not only detects better segmentations masks , but also predicts its performance well enough to have a 6 times better F1 score compared to STL-scaled [24].

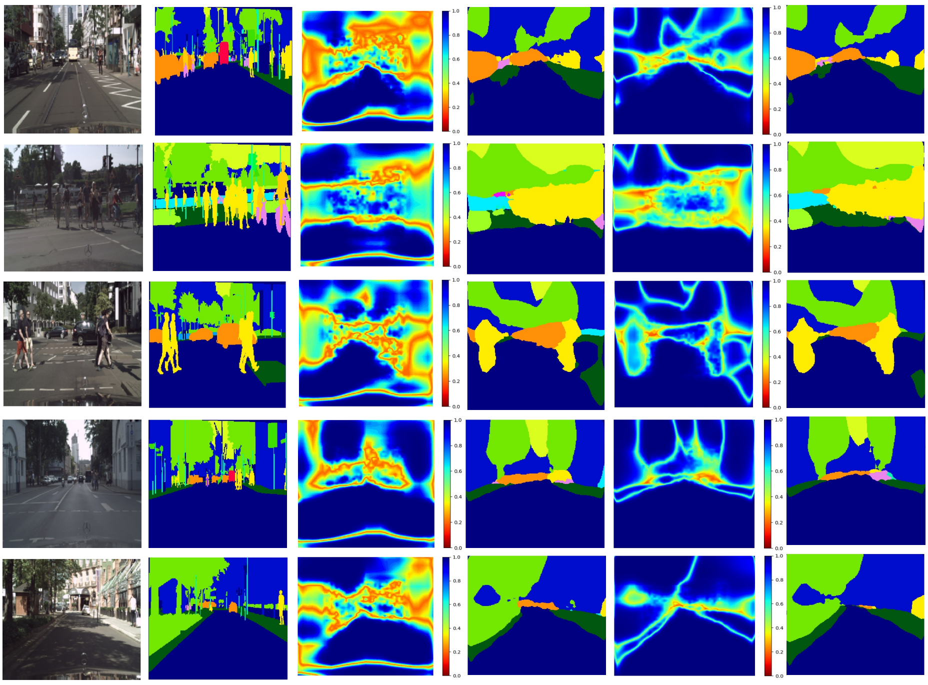

Fig. 4. Confidence prediction on Cityscapes dataset using STL-scaled [24] and MTL methods. From left to right: input image, ground truth segmentation labels, confidence heat-map obtained by MTL (ours), predicted segmentation labels by MTL (ours), confidence heat-map obtained by STL-scaled (baseline, [24]), predicted labels by STL-scaled (baseline, [24])

Here we also include some qualitative results on confidence measurement per pixel for semantic segmentation. Figures 4 and 5 show the confidence predictions per segmented pixels on Cityscapes and SBD datasets, respectively. We may compare the predicted confidence scores using the heat-map visualization for probability of success. Comparing the heat-map obtained by MTL against the STL-scaled [24] method, it is apparent how sensitive MTL is to not being able to segmenting out the fine-grained labels. While STL-scaled method [24] appears to be uncertain only on undetected boundaries, MTL assigns low confidence scores on both undetected boundaries as well as undetected fine-grained segments such that one can immediately concludes that there must be a crowd of fine-grain segmentation labels when MTL heat-map appears to heat-up (correspondent to low confidence) in a specific region.

In Fig. 5 on images where a large region (not just boundaries) has been wrongly segmented out. MTL heatmaps appear to pronounce uncertainty more drastically than STL-scaled [24] (rows 1, 2, and 4) while STL-scaled [24]seems to be overconfident by assigning minimum score of ≈ 0.6 versus near-zero confidence predictions made by MTL.

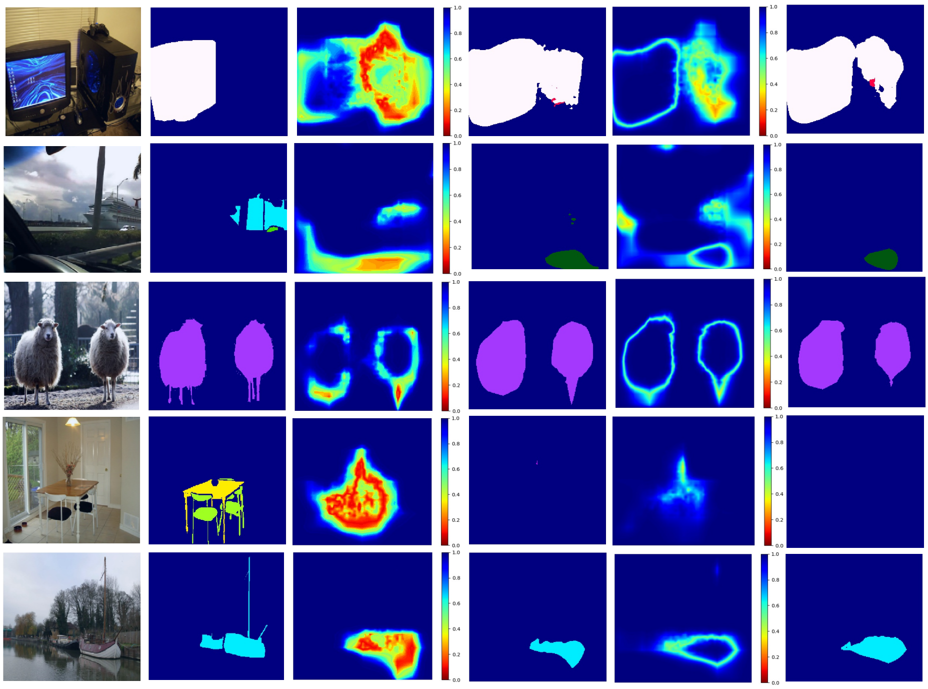

Fig. 5. Confidence prediction on SBD dataset using STL-scaled [24] and MTL methods. From left to right: input image, ground truth segmentation labels, confidence heat-map obtained by MTL (ours), predicted segmentation labels by MTL (ours), confidence heat-map obtained by STL-scaled (baseline, [24]), predicted labels by STL-scaled (baseline [24])

6 Conclusion

In this paper we have defined *performance learning* as an additional task that every neural network can learn in parallel to its main task(s) in a multitask learning fashion. We explained how these two tasks are mutually inclusive and can be beneficial to another in gaining high performance at both. We also proposed a simple Bayesian inference on the output probabilities of the performance learner and map them to the true probability of success or confidence at a given instance. We first showed that MTL can improve the model performance on the main task; on image classification benchmarks we were able to outperform nearly all the state-of-the-art architectures and on segmentation problem we were able to outperform in mean IoU and mean accuracy of the equivalent STL model. We evaluated our confidence learner on its accuracy, precision, and f-score on in/out distributional datasets and were able to surpass the baseline in having a higher f-score and accuracy.

References

1. Amato, F., López, A., Peña-Méndez, E.M., Vaňhara, P., Hampl, A., Havel, J.: Artificial neural networks in medical diagnosis (2013)
2. Khan, J., Wei, J.S., Ringner, M., Saal, L.H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C.R., Peterson, C., et al.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature medicine **7**(6) (2001) 673–679
3. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
4. Ebrahimi, S., Rohrbach, A., Darrell, T.: Gradient-free policy architecture search and adaptation. In: Proceedings of the 1st Annual Conference on Robot Learning. Volume 78 of Proceedings of Machine Learning Research., PMLR (13–15 Nov 2017) 505–514
5. Kaastra, I., Boyd, M.: Designing a neural network for forecasting financial and economic time series. Neurocomputing **10**(3) (1996) 215–236
6. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? In: Advances in Neural Information Processing Systems. (2017) 5580–5590
7. Gao, T., Koller, D.: Discriminative learning of relaxed hierarchy for large-scale visual recognition. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 2072–2079
8. Zhang, P., Wang, J., Farhadi, A., Hebert, M., Parikh, D.: Predicting failures of vision systems. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 3566–3573
9. Kahn, G., Villaflor, A., Pong, V., Abbeel, P., Levine, S.: Uncertainty-aware reinforcement learning for collision avoidance. arXiv preprint arXiv:1702.01182 (2017)
10. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 3431–3440
11. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
12. Denker, J.S., Lecun, Y.: Transforming neural-net output levels to probability distributions. In: Advances in neural information processing systems. (1991) 853–859
13. MacKay, D.J.: Bayesian methods for adaptive models. PhD thesis, California Institute of Technology (1992)
14. Balan, A.K., Rathod, V., Murphy, K.P., Welling, M.: Bayesian dark knowledge. In: Advances in Neural Information Processing Systems. (2015) 3438–3446
15. Hernández-Lobato, J.M., Adams, R.: Probabilistic backpropagation for scalable learning of bayesian neural networks. In: International Conference on Machine Learning. (2015) 1861–1869
16. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424 (2015)
17. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: international conference on machine learning. (2016) 1050–1059

18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1) (2014) 1929–1958
19. Gal, Y.: Uncertainty in deep learning. University of Cambridge (2016)
20. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: ICML. Volume 1., Citeseer (2001) 609–616
21. Zadrozny, B., Elkan, C.: Transforming classifier scores into accurate multiclass probability estimates. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2002) 694–699
22. Naeini, M.P., Cooper, G.F., Hauskrecht, M.: Obtaining well calibrated probabilities using bayesian binning. In: AAAI. (2015) 2901–2907
23. Platt, J., et al.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers **10**(3) (1999) 61–74
24. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. arXiv preprint arXiv:1706.04599 (2017)
25. Hagen, J.W., Jongeward, R.H., Kail Jr, R.V.: Cognitive perspectives on the development of memory. In: Advances in child development and behavior. Volume 10. Elsevier (1975) 57–101
26. Scheirer, W.J., Rocha, A., Micheals, R.J., Boult, T.E.: Meta-recognition: The theory and practice of recognition score analysis. IEEE transactions on pattern analysis and machine intelligence **33**(8) (2011) 1689–1695
27. Caruana, R.: Learning many related tasks at the same time with backpropagation. In: Advances in neural information processing systems. (1995) 657–664
28. Caruana, R.: Multitask learning. In: Learning to learn. Springer (1998) 95–133
29. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. arXiv preprint arXiv:1610.02136 (2016)
30. Chrabaszcz, P., Loshchilov, I., Hutter, F.: A downsampled variant of imagenet as an alternative to the cifar datasets. arXiv preprint arXiv:1707.08819 (2017)
31. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D.: Concrete problems in ai safety. arXiv preprint arXiv:1606.06565 (2016)
32. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: International Conference on Computer Vision (ICCV). (2011)
33. Shah, M.P.: Semantic segmentation architectures implemented in pytorch. https://github.com/meetsahh1995/pytorch-semseg (2017)