DeepSetNet: Predicting Sets with Deep Neural Networks

Supplemental Material

S. Hamid Rezatofighi Vijay Kumar B G Anton Milan
Ehsan Abbasnejad Anthony Dick Ian Reid
School of Computer Science, The University of Adelaide, Australia

This document accompanies our ICCV submission titled "DeepSetNet: Predicting Sets with Deep Neural Networks" and completes the main submission. We first provide more background on finite set statistics. Further, we add the details of our derivations for Deep Set Network that were omitted due to space constraints. To do this, here we augment Sections 3 and 4 of the main text. Finally, we provide more discussions and results on all object counting, multi-label classification and pedestrian detection applications.

Video. The accompanying video shows an example sequence from the UCSD dataset for people counting. It shows the video frame along with the input to our network, which is a superimposed image of pedestrian detection proposals, as described in Sec. 5.2 in the main submission. The plot on the bottom shows the number of ground truth pedestrians and our prediction.

1. Background on Finite Set Statistics

Finite Set Statistics provide powerful and practical mathematical tools for dealing with random finite sets, based on the notion of integration and density that is consistent with the point process theory [5]. In this section, we review some basic mathematical background about this subject of statistics.

In the conventional statistics theory, a continuous random variable y is a variable that can take an infinite number of possible values. A continuous random vector can be defined by stacking several continuous random variables into a fixed length vector, $Y = (y_1, \cdots, y_m)$. The mathematical function describing the possible values of a continuous random vector, and their associated joint probabilities, is known as a probability density function (PDF) $p(Y)$ such that $\int p(Y)dY = 1$.

A random finite set (RFS) $\mathcal{Y}$ is a finite-set valued random variable $\mathcal{Y} = \{y_1, \cdots, y_m\} \subset \mathbb{Y}$. The main difference between an RFS and a random vector is that for the former, the number of constituent variables, m, is random and the variables themselves are random and unordered, while the latter is of a fixed size with a predefined order.

A statistical function describing a finite-set variable $\mathcal{Y}$ is a combinatorial probability density function $p(\mathcal{Y})$, which consists of a discrete probability distribution, the so-called cardinality distribution, and a family of joint probability densities on the values of the constituent variables for each cardinality. Similar to the definition of a PDF for a random variable, the PDF of an RFS must sum to unity over all possible cardinality values and all possible element values and their permutations, *i.e.*

$$\int p(\mathcal{Y})\mu(d\mathcal{Y}) \triangleq \sum_{m=0}^{\infty} \frac{1}{m!U^m} \int p(\{y_1, \cdots, y_m\}_{||})dy_1 \cdots dy_m = 1, \tag{1}$$

where μ is the dominating measure and U is the unit of hypervolume in $\mathbb{Y}$ [6]. The PDF of an m-dimensional random vector can be defined in terms of an RFS as:

$$p(y_1, \cdots, y_m) \triangleq \frac{1}{m!} p(\{y_1, \cdots, y_m\}_{||}). \tag{2}$$

The denominator $m! = \prod_{k=1}^{m} k$ appears because the probability density for a set $\{y_1, \cdots, y_m\}_{||}$ must be equally distributed among all the $m!$ possible permutations of the vector [5].

The cardinality distribution $p(m)$ over the number of elements in the random finite set $\mathcal{Y}$ is obtained by

$$p(m) = \int_{|\mathcal{Y}|=m} p(\mathcal{Y})\mu(d\mathcal{Y}) \triangleq \frac{1}{m!U^m} \int p(\{y_1, \cdots, y_m\}_{||})dy_1 \cdots dy_m. \tag{3}$$

Algorithm 1: Drawing samples from a set distribution.

Sampling an RFS Probability Distribution

- Initialise $\mathcal{Y} \leftarrow \emptyset$
- Sample cardinality $m \sim p(m)$
- Sample m points from an m-dimensional joint distribution
  $\mathcal{Y} \sim p(\{y_1, y_2, \cdots .y_m\}_{||}) \leftarrow m! \times p(y_1, y_2, \cdots .y_m)$

In the case of i.i.d. samples:

for $i \leftarrow \{1, \ldots, m\}$
  sample $y_i \sim p(y)$
  set $\mathcal{Y} \leftarrow \mathcal{Y} \cup y_i$
end

Similar to the conventional statistics for random variables, the expectation of an RFS has been defined above. The first statistical moment, or the expected value, of an RFS is known as intensity density or probability hypothesis density (PHD) and is calculated by definition as

$$v(y) \triangleq \int \delta_{\mathcal{Y}}(y) p(\mathcal{Y}) \mu(d\mathcal{Y}), \tag{4}$$

where $\delta_{\mathcal{Y}}(y) = \sum_{x \in \mathcal{Y}} \delta_x(y)$ and $\delta_x(y)$ denotes the Dirac delta function concentrated at x. The PHD function $v(y)$ is interpreted as the instantaneous expected number of the variables that exist at that point y. Moreover, the integral of the PHD over a region gives the expected number of elements in that region and the peaks of the PHD indicate highest local concentrations of the expected number of elements.

Having an RFS distribtuion $p(\mathcal{Y})$, the samples can be drawn from this distribution as shown in Algorithm 1.

2. Deep Set Network

Let us begin by defining a training set $\mathcal{D} = \{\mathcal{Y}_i, \mathbf{x}_i\}$, where each training sample $i = 1, \ldots, n$ is a pair consisting of an input feature $\mathbf{x}_i \in \mathbb{R}^l$ and an output (or label) set $\mathcal{Y}_i = \{y_1, y_2, \ldots, y_{m_i}\}, y_k \in \mathbb{R}^d, m_i \in \mathbb{N}^0$. In the following we will drop the instance index i for better readability. Note that $m := |\mathcal{Y}|$ denotes the cardinality of set $\mathcal{Y}$. The probability of a set $\mathcal{Y}$ with an unknown cardinality is defined as:

$$\begin{aligned} p(\mathcal{Y}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{w}) =& p(m|\mathbf{x}, \mathbf{w}) \times U^m \times p(\{y_1, y_2, \cdots, y_m\}_{||}|\mathbf{x}, \boldsymbol{\theta}) \\ =& p(m|\mathbf{x}, \mathbf{w}) \times m! \times U^m \times p(y_1, y_2, \cdots, y_m|\mathbf{x}, \boldsymbol{\theta}), \end{aligned} \tag{5}$$

where $p(m|\cdot, \cdot)$ and $p(y_1, y_2, \cdots, y_m|\cdot, \cdot)$ are respectively a cardinality distribution and a symmetric joint probability density distribution of the elements. U is the unit of hypervolume in $\mathbb{R}^d$, which makes the joint distribution unitless [6]. $\boldsymbol{\theta}$ denotes the parameters that estimate the joint distribution of set element values for a fixed cardinality, while $\mathbf{w}$ represents the collection of parameters which estimate the cardinality distribution of the set elements.

The above formulation represents the probability density of a set which is very general and completely independent from the choices of both cardinality and spatial distributions. It is thus straightforward to transfer it to many applications that require the output to be a set. However, to make the problem amenable to mathematical derivation and implementation, we adopt two assumptions: *i)* the outputs (or labels) in the set are independent and identically distributed (*i.i.d.*) and *ii)* their cardinality follows a Poisson distribution with parameter λ. Thus, we can write the distribution as

$$p(\mathcal{Y}|\mathbf{x}, \boldsymbol{\theta}, \mathbf{w}) = \int p(m|\lambda) p(\lambda|\mathbf{x}, \mathbf{w}) d\lambda \times m! \times U^m \times \left(\prod_{k=1}^{m} p(y_k|\mathbf{x}, \boldsymbol{\theta}) \right). \tag{6}$$

2.1. Posterior distribution

To learn the parameters $\boldsymbol{\theta}$ and $\mathbf{w}$, it is valid to assume that the training samples are independent from each other and the distribution over the input data $p(\mathbf{x})$ is independent from the output and the parameters. Therefore, the posterior distribution

over the parameters can be derived as

$$\begin{aligned} p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D}) &= \frac{1}{Z} p(\mathcal{D}|\boldsymbol{\theta}, \mathbf{w}) p(\boldsymbol{\theta}) p(\mathbf{w}) \\ &= \frac{1}{Z} p(\{\mathcal{Y}_i, \mathbf{x}_i\}_{\forall i}|\boldsymbol{\theta}, \mathbf{w}) p(\boldsymbol{\theta}) p(\mathbf{w}) \\ &= \frac{1}{Z} \prod_{i=1}^{n} \Big[p(\mathcal{Y}_i|\mathbf{x}_i, \boldsymbol{\theta}, \mathbf{w}) p(\mathbf{x}_i) \Big] p(\boldsymbol{\theta}) p(\mathbf{w}) \\ &= \frac{1}{Z} \prod_{i=1}^{n} \left[\int p(m_i|\lambda) p(\lambda|\mathbf{x}_i, \mathbf{w}) d\lambda \times m_i! \times U^{m_i} \times \left(\prod_{k=1}^{m_i} p(y_k|\mathbf{x}_i, \boldsymbol{\theta}) \right) p(\mathbf{x}_i) \right] p(\boldsymbol{\theta}) p(\mathbf{w}), \end{aligned} \tag{7}$$

where Z is a normaliser defined as

$$Z = \int \int \prod_{i=1}^{n} \left[\int p(m_i|\lambda) p(\lambda|\mathbf{x}_i, \mathbf{w}) d\lambda \times m_i! \times U^{m_i} \times \left(\prod_{k=1}^{m_i} p(y_k|\mathbf{x}_i, \boldsymbol{\theta}) \right) p(\mathbf{x}_i) \right] p(\boldsymbol{\theta}) p(\mathbf{w}) \quad d\theta d\mathbf{w}. \tag{8}$$

The probability $p(\mathbf{x}_i)$ can be eliminated as it appears in both the numerator and the denominator. Therefore,

$$p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D}) = \frac{1}{\tilde{Z}} \prod_{i=1}^{n} \left[\int p(m_i|\lambda) p(\lambda|\mathbf{x}_i, \mathbf{w}) d\lambda \times m_i! \times U^{m_i} \times \left(\prod_{k=1}^{m_i} p(y_k|\mathbf{x}_i, \boldsymbol{\theta}) \right) \right] p(\boldsymbol{\theta}) p(\mathbf{w}), \tag{9}$$

where

$$\tilde{Z} = \int \int \prod_{i=1}^{n} \left[\int p(m_i|\lambda) p(\lambda|\mathbf{x}_i, \mathbf{w}) d\lambda \times m_i! \times U^{m_i} \times \left(\prod_{k=1}^{m_i} p(y_k|\mathbf{x}_i, \boldsymbol{\theta}) \right) \right] p(\boldsymbol{\theta}) p(\mathbf{w}) \quad d\theta d\mathbf{w}. \tag{10}$$

A closed form solution for the integral in Eq. (9) can be obtained by using conjugate priors:

$$\begin{aligned} m &\sim \mathcal{P}(m; \lambda) \\ \lambda &\sim \mathcal{G}(\lambda; \alpha(\mathbf{x}, \mathbf{w}), \beta(\mathbf{x}, \mathbf{w})) \\ & \quad \alpha(\mathbf{x}, \mathbf{w}), \beta(\mathbf{x}, \mathbf{w}) > 0 \quad \forall \mathbf{x}, \mathbf{w} \\ \boldsymbol{\theta} &\sim \mathcal{N}(\boldsymbol{\theta}; 0, \sigma_1^2 \mathbf{I}) \\ \mathbf{w} &\sim \mathcal{N}(\mathbf{w}; 0, \sigma_2^2 \mathbf{I}), \end{aligned}$$

where $\mathcal{P}(\cdot, \lambda)$, $\mathcal{G}(\cdot; \alpha, \beta)$, and $\mathcal{N}(\cdot; 0, \sigma^2 \mathbf{I})$ represent respectively a Poisson distribution with parameters λ, a Gamma distribution with parameters (α, β) and a zero mean normal distribution with covariance equal to $\sigma^2 \mathbf{I}$.

We assume that the cardinality follows a Poisson distribution whose mean, λ, follows a Gamma distribution, with parameters which can be estimated from the input data $\mathbf{x}$. Note that the cardinality distribution in Eq. 5 can be replaced by any other discrete distribution. For example, it is a valid assumption to model the number of objects in natural images by a Poisson distribution [2]. Thus, we could directly predict λ to model this distribution by formulating the cardinality as $p(m|\mathbf{x}, \mathbf{w}) = \mathcal{P}(m; \lambda(\mathbf{x}, \mathbf{w}))$. However, this would limit the model's expressive power; because two visually entirely different images with the same number of objects would be mapped to the same λ. Instead, to allow for uncertainty of the mean, we model it with another distribution, which we choose to be Gamma for mathematical convenience. Consequently, the integrals in $p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D})$ are simplified and form a negative binomial distribution,

$$\text{NB}\,(m; a, b) = \frac{\Gamma(m + a)}{\Gamma(m + 1)\Gamma(a)} . (1 - b)^a b^m, \tag{11}$$

where Γ is the Gamma function. Finally, the full posterior distribution can be written as

$$p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D}) = \frac{1}{\tilde{Z}} \prod_{i=1}^{n} \left[\text{NB} \left(m_i; \alpha(\mathbf{x}_i, \mathbf{w}), \frac{1}{1 + \beta(\mathbf{x}_i, \mathbf{w})} \right) \times m_i! \times U^{m_i} \times \left(\prod_{k=1}^{m_i} p(y_k|\mathbf{x}_i, \boldsymbol{\theta}) \right) \right] p(\boldsymbol{\theta}) p(\mathbf{w}). \tag{12}$$

2.2. Learning

For simplicity, we use a point estimate for the posterior $p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D})$, *i.e.* $p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D}) = \delta(\boldsymbol{\theta} = \boldsymbol{\theta}^*, \mathbf{w} = \mathbf{w}^*|\mathcal{D})$, where $(\boldsymbol{\theta}^*, \mathbf{w}^*)$ are computed using the following MAP estimator:

$$(\boldsymbol{\theta}^*, \mathbf{w}^*) = \arg\max_{\boldsymbol{\theta}, \mathbf{w}} \quad \log\left(p\left(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D}\right)\right). \tag{13}$$

Since the solution to the above problem is independent from the normalisation constant $\tilde{Z}$, we have

$$\begin{aligned}(\boldsymbol{\theta}^*, \mathbf{w}^*) = \arg\max_{\boldsymbol{\theta}, \mathbf{w}} \quad & \log\left(p(\boldsymbol{\theta})\right) + \sum_{i=1}^{n}\left[\log\left(m_i!\right) + m_i \log U + \sum_{k=1}^{m_i} \log\left(p(y_k|\mathbf{x}_i, \boldsymbol{\theta})\right)\right. \\ & \left. + \log\left(NB\left(m_i; \alpha(\mathbf{x}_i, \mathbf{w}), \frac{1}{1+\beta(\mathbf{x}_i, \mathbf{w})}\right)\right)\right] + \log\left(p(\mathbf{w})\right) \\ = \arg\max_{\boldsymbol{\theta}, \mathbf{w}} \quad & f_1(\boldsymbol{\theta}) + f_2(\mathbf{w}).\end{aligned} \tag{14}$$

Therefore, the optimisation problem in Eq. (14) can be decomposed w.r.t. the parameters $\boldsymbol{\theta}$ and $\mathbf{w}$. Therefore, we can learn them independently in two separate problems

$$\begin{aligned}\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} \quad f_1(\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \quad -\gamma_1\|\boldsymbol{\theta}\| + \sum_{i=1}^{n}\left[\log\left(m_i!\right) + m_i \log U + \sum_{k=1}^{m_i} \log\left(p(y_k|\mathbf{x}_i, \boldsymbol{\theta})\right)\right] \\ &\equiv \arg\max_{\boldsymbol{\theta}} \quad -\gamma_1\|\boldsymbol{\theta}\| + \sum_{i=1}^{n}\sum_{k=1}^{m_i} \log\left(p(y_k|\mathbf{x}_i, \boldsymbol{\theta})\right)\end{aligned} \tag{15}$$

and

$$\begin{aligned}\mathbf{w}^* &= \arg\max_{\mathbf{w}} \quad f_2(\mathbf{w}) \\ &= \arg\max_{\mathbf{w}} \quad \sum_{i=1}^{n}\left[\log\left(\frac{\Gamma(m_i + \alpha(\mathbf{x}_i, \mathbf{w}))}{\Gamma(m_i+1)\Gamma(\alpha(\mathbf{x}_i, \mathbf{w}))}\right)\right. \\ &\qquad \left. + \log\left(\frac{\beta(\mathbf{x}_i, \mathbf{w})^{\alpha(\mathbf{x}_i, \mathbf{w})}}{(1+\beta(\mathbf{x}_i, \mathbf{w})^{\alpha(\mathbf{x}_i, \mathbf{w})+m_i)}}\right)\right] - \gamma_2\|\mathbf{w}\|,\end{aligned} \tag{16}$$

where γ_1 and γ_2 are the regularisation parameters, proportional to the predefined covariance parameters σ_1 and σ_2. These parameters are also known as weight decay parameters and commonly used in training neural networks.

The learned parameters $\boldsymbol{\theta}^*$ in Eq. (15) are used to map an input feature vector $\mathbf{x}$ into an output vector Y. For example, in image classification, $\boldsymbol{\theta}^*$ is used to predict the distribution Y over all categories, given the input image $\mathbf{x}$. Note that $\boldsymbol{\theta}^*$ can generally be learned using a number of existing machine learning techniques. In this paper we rely on deep CNNs to perform this task.

To learn the highly complex function between the input feature $\mathbf{x}$ and the parameters (α, β), which are used for estimating the output cardinality distribution, we train a second deep neural network. Using neural networks to predict a discrete value may seem counterintuitive, because these methods at their core rely on the backpropagation algorithm, which assumes a differentiable loss. Note that we achieve this by describing the discrete distribution by continuous parameters α, β (Negative binomial $\text{NB}(\cdot, \alpha, \frac{1}{1+\beta})$), and can then easily draw discrete samples from that distribution. More formally, to estimate $\mathbf{w}^*$, we compute the partial derivatives of the objective function in Eq. (16) w.r.t. $\alpha(\cdot, \cdot)$ and $\beta(\cdot, \cdot)$ and use standard backpropagation to learn the parameters of the deep neural network.

$$\frac{\partial f_2(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial f_2(\mathbf{w})}{\partial \alpha(\mathbf{x}, \mathbf{w})} . \frac{\partial \alpha(\mathbf{x}, \mathbf{w})}{\partial \mathbf{w}} + \frac{\partial f_2(\mathbf{w})}{\partial \beta(\mathbf{x}, \mathbf{w})} . \frac{\partial \beta(\mathbf{x}, \mathbf{w})}{\partial \mathbf{w}} - 2\gamma_2 \mathbf{w}, \tag{17}$$

where

$$\frac{\partial f_2(\mathbf{w})}{\partial \alpha(\mathbf{x}, \mathbf{w})} = \sum_{i=1}^{n}\left[\Psi\Big(m_i + \alpha(\mathbf{x}_i, \mathbf{w})\Big) - \Psi\Big(\alpha(\mathbf{x}_i, \mathbf{w})\Big) + \log\left(\frac{\beta(\mathbf{x}_i, \mathbf{w})}{1+\beta(\mathbf{x}_i, \mathbf{w})}\right)\right], \tag{18}$$

and

$$\frac{\partial f_2(\mathbf{w})}{\partial \beta(\mathbf{x},\mathbf{w})} = \sum_{i=1}^{n} \left[\frac{\alpha(\mathbf{x}_i,\mathbf{w}) - m_i.\beta(\mathbf{x}_i,\mathbf{w})}{\beta(\mathbf{x}_i,\mathbf{w}).\Big(1+\beta(\mathbf{x}_i,\mathbf{w})\Big)} \right], \tag{19}$$

where $\Psi(\cdot)$ is the digamma function defined as

$$\Psi(\alpha) = \frac{d}{d\alpha}\log\left(\Gamma(\alpha)\right) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}. \tag{20}$$

2.3. Inference

Having the learned parameters of the network $(\mathbf{w}^*, \boldsymbol{\theta}^*)$, for a test feature $\mathbf{x}^+$, we use a MAP estimate to generate a set output as

$$\mathcal{Y}^* = \arg\max_{\mathcal{Y}} p(\mathcal{Y}|\mathcal{D}, \mathbf{x}^+), \tag{21}$$

where

$$p(\mathcal{Y}|\mathcal{D}, \mathbf{x}^+) \quad = \quad \int p(\mathcal{Y}|\mathbf{x}^+, \boldsymbol{\theta}, \mathbf{w}) p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D}) d\boldsymbol{\theta} d\mathbf{w}$$

and $p(\boldsymbol{\theta}, \mathbf{w}|\mathcal{D}) = \delta(\boldsymbol{\theta} = \boldsymbol{\theta}^*, \mathbf{w} = \mathbf{w}^*|\mathcal{D})$. Since the unit of hypervolume U in most practical application in unknown, to calculate the mode of the set distribution $p(\mathcal{Y}|\mathcal{D}, \mathbf{x}^+)$, we use the sequential inference as explained in [5]. To this end, we first calculate the mode m^* of the cardinality distribution

$$m^* = \arg\max_{m} \quad p(m|\mathbf{w}^*, \mathbf{x}^+), \tag{22}$$

where

$$p(m|\mathbf{w}^*, \mathbf{x}^+) = \text{NB}\left(m; \alpha(\mathbf{w}^*, \mathbf{x}^+), \frac{1}{1+\beta(\mathbf{w}^*, \mathbf{x}^+)}\right). \tag{23}$$

Then, we calculate the mode of the joint distribution for the given cardinality m^* as

$$\mathcal{Y}^* = \arg\max_{\mathcal{Y}_{m^*}} \quad p(\{y_1, \cdots, y_{m^*}\}_{||}|\boldsymbol{\theta}^*, \mathbf{x}^+). \tag{24}$$

To estimate the most likely set $\mathcal{Y}^*$ with cardinality m^*, we use the first CNN with the parameters $\boldsymbol{\theta}^*$ which predicts $p(y_1, \cdots, y_M|\mathbf{x}^+, \boldsymbol{\theta}^*)$, where M is the maximum cardinality of the set, *i.e.* $\{y_1, \cdots, y_{m^*}\} \subseteq \{y_1, \cdots, y_M\} \quad , \forall m^*$. Since the samples are *i.i.d.*, the joint probability maximised when the probability of each element in the set is maximised. Therefore, the most likely set $\mathcal{Y}^*$ with cardinality m^* is obtained by ordering the probabilities of the set elements $y_1, \cdots, y_M$ as the output of the first CNN and choosing m^* elements with highest probability values.

Note that the assumptions listed in Sec. 2 are necessary to make both learning and inference computationally tractable and amenable to an elegant mathematical formulation. A major advantage of this approach is that we can use any state-of-the-art classifier/detector as the first CNN ($\boldsymbol{\theta}^*$) to further improve its performance. Modifying any of the assumptions, *e.g.* non-*i.i.d.* set elements, leads to serious mathematical complexities [5], and are left for future work.

3. Further Experimental Results

Here, we provide additional arguments, evaluation plots and qualitative results that could not be included in the main paper.

3.1. Object counting by regression

Regressing for cardinality may seem an obvious choice, but is not trivial to derive mathematically and cannot be easily justified in our framework because it *a)* cannot be accommodated in a Bayesian set formulation to model uncertainty and *b)* does not yield a discrete distribution. Nonetheless, we have conducted the experiment by replacing our loss with the regression loss while using the exact same architecture and setup as in Sec. 5.2 of the main text. Tab. 2 shows the comparison results between our cardinality loss and regression loss on two datasets from two reported tasks of multi-class image classification (MS-COCO) and pedestrian detection (MOT16). As expected, directly regressing for cardinality yields slightly worse results both in terms of the cardinality prediction and w.r.t. the F1 score. For completeness, Tab. 3 also reports the mean absolute error and standard deviation for cardinality estimation using our loss on four datasets.

Table 2: Loss comparison for cardinality estimation.

	Mean card. error		F1 score	
Loss	MOT16	MS COCO	MOT16	MS COCO
Regression	2.05	0.83	60.16	68.4
Negative Binomial	**1.94**	**0.74**	**61.86**	**69.4**

Table 3: Mean absolute error and standard deviation for cardinality estimation on test sets.

	Multi-label classification		Pedestrian detection	
Error	PASCAL VOC	MS COCO	Caltech	MOT16
Mean	0.32	0.74	0.54	1.94
Std	0.52	0.86	0.79	1.96

3.2. Pedestrian detection

Here, we first discuss the challenges that we confronted to use our set formulation for this application. Then we provide some information about the datasets and their split used for this experiment. Finally, we show more quantitative and qualitative results on this experiment.

Non-maximum suppression. In the main text, we argued that the non-maximum suppression (NMS) as a heuristic step makes the detection problem not as straightforward as what is expressed in our set formulation in Eq. (5). In fact, a major nuisance in detection is not the NMS algorithm itself as a greedy solver, but rather its hand-crafted objective. This process is traditionally formulated as maximising the joint distribution over pairs of samples, or equivalently as a quadratic binary program (QBP)

$$Y^* = \arg\max_Y Y^T Q Y, \tag{25}$$

where $Y \in \mathbb{B}^M$ is a binary vector, indicating which of the M boxes to keep or to suppress. The diagonal values of Q are proportional to the detection scores while the pairwise (exclusion) terms in Q are manually designed, *e.g.* to correspond to the overlap ratios. The aforementioned QBP is NP-hard and cannot be solved globally in general. NMS is one greedy and efficient approach to solve the problem locally. To enforce m^*, one could include a constraint into the QBP like

$$\begin{aligned} Y^* &= \arg\max_Y Y^T Q Y, \\ s.t. &\mathbf{1}^T Y = m^*. \end{aligned} \tag{26}$$

However, this may lead to an infeasible problem for a fixed Q with a predefined value of the threshold for an overlap ratio T_O. To this end, Q should be designed such that the above problem can have a feasible solution. Learning Q is perhaps a more elegant approach, but is not part of this paper. To this end, for the current setup, one solution is to find a threshold that can make the above problem feasible. Therefore, we start from the default value of T_O, and adjust it step-wise until the number of boxes reaches m^*. In the case if the number of final boxes is larger than m^*, we pick m^* boxes with the highest scores. To apply a solver, we experimented with the global QBP formulation using Gurobi for a small problem, but found NMS with an adaptive threshold to be the most efficient and effective approach.

Caltech Pedestrians [3] is a de-facto standard benchmark for pedestrian detection. The dataset contains sequences captured from a vehicle driving through regular traffic in an urban environment and provides bounding box annotations of nearly $350,000$ pedestrians. The annotations also include detailed occlusion labels. The number of pedestrians per image varies between 0 and 14. However, more than 55% of the images contain no people at all and around 30% of the data includes one or two persons. We use the MS-CNN [1] network model and its parameters learned on the Caltech training set as $\boldsymbol{\theta}^*$ in Eq. (15). To learn the cardinality, we use 4250 images provided as a training set, splitting it into training and validation ($80\% - 20\%$), reaching a mean absolute error of 0.54 (*cf*. Tab. 3).

MOTCallenge 2016. This benchmark is primarily targeted at multi-object tracking and is not yet commonly used for evaluating pedestrian detection. However, the variation in the number of pedestrians across the frames is relatively large (between 0 and 32) and is also distributed more uniformly, which makes correct cardinality estimation more important. Since the labels

for the test set are not available, we use the provided training set of this benchmark consisting of 5316 images from 7 different sequences, and divide it into training, validation and test set with split ratios 60%, 15% and 25%, respectively. We only learn the cardinality network $\mathbf{w}^*$ on training set and we use the MS-CNN network model and its parameters learned on the KITTI dataset [4] as $\boldsymbol{\theta}^*$ in Eq. (15).

Additional Results. ROC curves on two detection datasets are shown in Fig. 1. Qualitative results of pedestrian detection are shown in Figure 2.

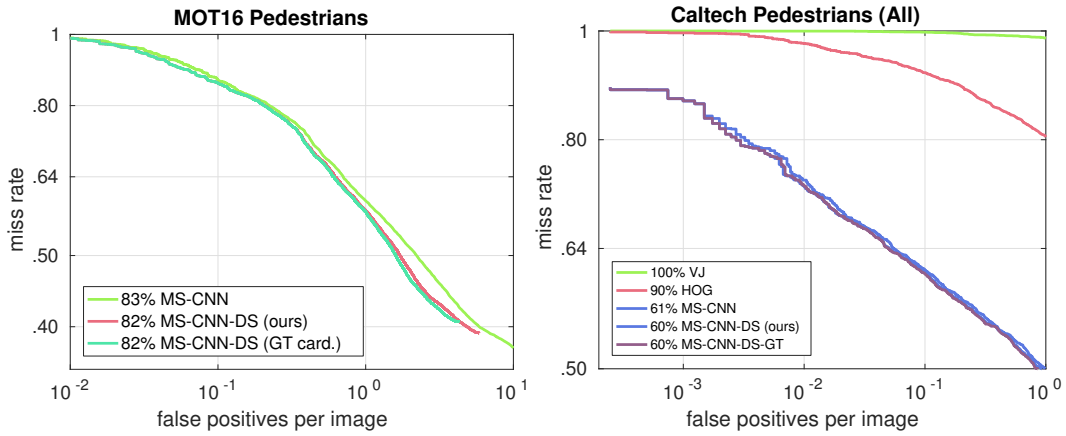

Figure 1: ROC curves on MOT16 and Caltech Pedestrians (experiment "all"). The overall performance of a detector is measured by the log-average miss rate as proposed by Dollár *et al*. [3].

3.3. Multi-class image classification.

Figure 3 shows more results for successful image tagging. Figure 4 points to some interesting failures and erroneous predictions.

References

[1] Z. Cai, Q. Fan, R. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV 2016*. 6, 8

[2] A. B. Chan and N. Vasconcelos. Bayesian poisson regression for crowd counting. In *CVPR*, pages 545–551, 2009. 3

[3] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE T. Pattern Anal. Mach. Intell.*, 34, 2012. 6, 7

[4] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI Vision Benchmark Suite. In *CVPR 2012*. 7

[5] R. P. Mahler. *Statistical multisource-multitarget information fusion*, volume 685. Artech House Boston, 2007. 1, 5

[6] B.-N. Vo et al. Model-based classification and novelty detection for point pattern data. In *ICPR*, 2016. 1, 2

Figure 2: More examples illustrating results of pedestrian detection generated using the state-of-the-art detector MS-CNN [1] (in blue, left) and our MS-CNN + DeepSetNet (right). To generate the MS-CNN results, we display the top m^* boxes after applying non-maximum suppression. Arrows indicate typical failures introduced by a suboptimal NMS threshold, which are corrected when considering the predicted number of persons for each image.

Figure 3: Further examples showing a perfect prediction w.r.t. both the number of tags and the labels themselves using our Deep Set Network.

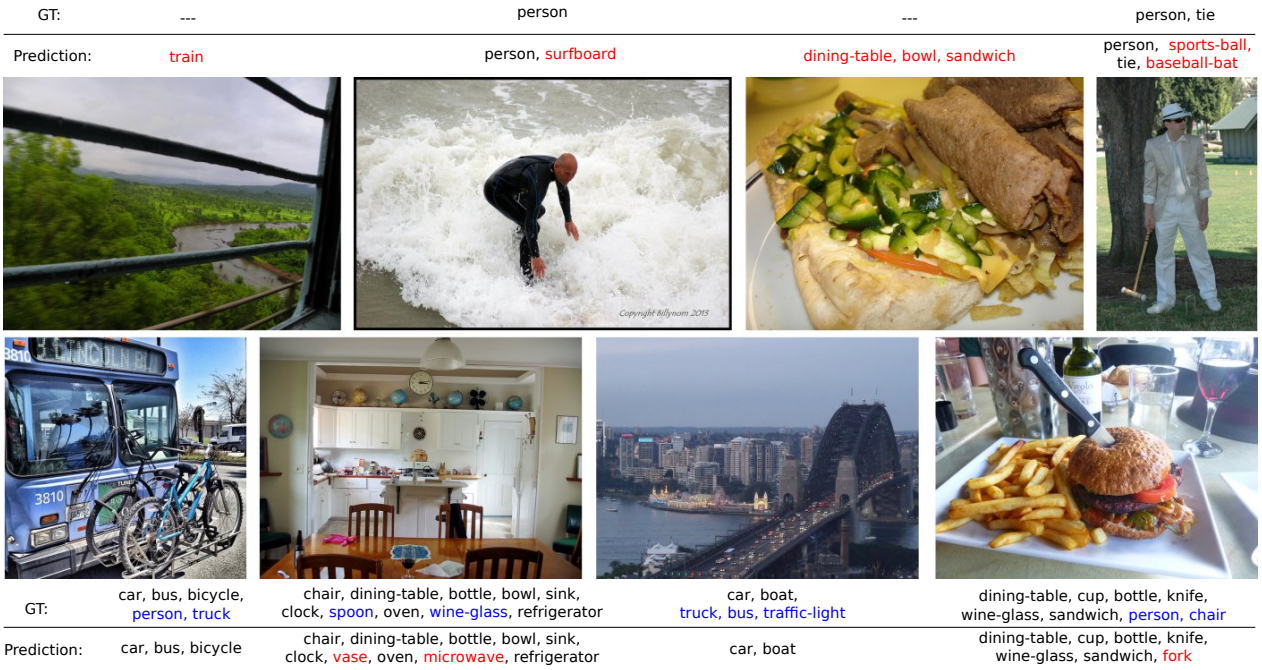

Figure 4: Additional examples illustrating interesting failure cases. False negatives and false positives are highlighted in blue and red, respectively. Note that in most examples, the mismatch between the ground truth and our prediction is due to the ambiguity or artifacts in the annotations. Two such examples are shown in the top row, where a train (window) and the surfboard are not annotated, probably because these are hardly visible in the image. Nevertheless, our network can predict the objects. The two bottom rows show real failure cases of our method. Note, however, that these include extremely challenging examples, where even for a human, it is fairly difficult to spot a traffic light in the aerial image or the person and the chair in the image on the bottom right.