Evidential fully convolutional network for semantic segmentation

Zheng Tong, Philippe Xu, Thierry Denoeux

Applied Intelligence manuscript No.
(will be inserted by the editor)

Evidential fully convolutional network for semantic segmentation

Zheng Tong · Philippe Xu · Thierry Denœux

Received: date / Accepted: date

Abstract We propose a hybrid architecture composed of a fully convolutional network (FCN) and a Dempster-Shafer layer for image semantic segmentation. In the so-called evidential FCN (E-FCN), an encoder-decoder architecture first extracts pixel-wise feature maps from an input image. A Dempster-Shafer layer then computes mass functions at each pixel location based on distances to prototypes. Finally, a utility layer performs semantic segmentation from mass functions and allows for imprecise classification of ambiguous pixels and outliers. We propose an end-to-end learning strategy for jointly updating the network parameters, which can make use of soft (imprecise) labels. Experiments using three databases (Pascal VOC 2011, MIT-scene Parsing and SIFT Flow) show that the proposed combination improves the accuracy and calibration of semantic segmentation by assigning confusing pixels to multi-class sets.

Keywords Evidence theory · belief function · fully convolutional network · decision analysis · semantic segmentation

1 Introduction

In the past few decades, one of the most difficult problems in computer vision has been image semantic segmentation, defined as the process of partitioning a digital image into multiple sets of pixels. The result of image segmentation is a set of segments that collectively cover the entire image, called the *segmentation mask*. This mask constitutes a simplified representation, more meaningful and easier to analyze than the original image. Semantic segmentation has been widely applied

Zheng Tong, Philippe Xu
Université de technologie de Compiègne, CNRS, Heudiasyc, Compiègne, France
E-mail: zheng.tong@hds.utc.fr, philippe.xu@hds.utc.fr

Thierry Denœux
Université de technologie de Compiègne, CNRS, Heudiasyc, Compiègne, France, and
Institut universitaire de France, Paris, France
E-mail: thierry.denoeux@utc.fr

to advanced driver assistance systems [6, 16, 36], human-machine interaction [39], medical imaging [18], and so on.

In the last decade, deep learning-based models, especially fully convolutional networks (FCNs) [25] and variants [22, 28], have been developed for semantic segmentation and have achieved remarkable success. FCNs take advantage of existing deep neural networks, which have the capacity to learn reliable and robust features. An FCN transforms existing and well-known classification models such as VGG (16-layer net) [31] or ResNet [21] into fully convolutional ones by replacing the fully connected layers with convolutional ones to output spatial maps instead of classification scores. Those maps are upsampled using fractionally-strided convolutions (called *deconvolutions* [41, 42]) to produce dense per-pixel labeled outputs. This approach has allowed for significant improvements in segmentation accuracy over traditional methods on benchmark databases like Pascal VOC 2011 [17]. However, despite the power and flexibility of the FCN-based models, they still face the following three problems:

1. *How to perform novelty detection?* In many learning sets, not all classes are labeled, especially for some objects in the background. An ideal image segmentation algorithm should detect "unknown" objects belonging to classes that are not represented in the learning set. This capacity is called *novelty detection* [8]. FCN-based models generally randomly assign unknown objects to one of the known classes, though some models tend to assign unknown objects to the background class.
2. *How to process pixels with confusing information?* In image-segmentation training sets, all pixels are precisely labeled, even if the true label is actually uncertain. This is the case, for example, for the pixels at object borders. Pixels with precise but incorrect labels may have negative effects on the performance of learning systems [2, 27].
3. *When will the FCN-based methods fail?* In decision-making systems, a neural network should not only be as accurate as possible, but it should also have the ability to signal when it is likely to be incorrect. Neural networks developed nowadays tend not to be well calibrated [20], though they are more accurate than they were a decade ago. In other words, the accuracy of modern neural networks, including FCN-based models, does not match their confidence.

In this paper, the above problems are tackled using the Dempster-Shafer (DS) theory of belief functions [7, 30]. This mathematical framework, also referred to as *Evidence Theory*, is based on representing independent pieces of evidence by mass functions and combining them using a generic operator called Dempster's rule. It is a well-established formalism for reasoning and making decisions under uncertainty [10, 13, 38]. A mass function has more degrees of freedom than a probability distribution, which allows it to represent a wider range of belief states, from complete ignorance to full certainty.

One of the applications of the DS theory is to design *evidential classifiers* (e.g., [9, 14, 24, 33]), which compute a predicted mass function for each input vector. The output mass function can then be used for decision-making [4, 8, 19]. Over the years, two main principles have been developed to design an evidential classifier: the *model-based* and *distance-based* approaches [15]. The former uses estimated class-conditional distributions [32], while the latter constructs mass functions based on distances to prototypes [9, 14, 24]. Thanks to the generality and

expressiveness of the belief-function formalism, an evidential classifier provides more informative outputs than those of conventional classifiers (e.g., a neural network with a softmax output layer). The flexibility of evidential classifiers can be exploited for uncertain data classification [40] and set-valued classification [8, 26]. Therefore, it may be advantageous to combine an FCN-based model with an evidential classifier for semantic segmentation.

The objective of this study is to take advantage of object representations generated by an FCN and use them as the input features of an evidential classifier for decision-making. The proposed model, referred to as the *evidential fully convolutional network* (E-FCN), transforms an FCN model by replacing its softmax layer by a distance-based DS layer and a utility layer. In an E-FCN, an FCN architecture is used to extract pixel-wise high-order features from an input image. Then, the features are converted into pixel-wise mass functions by the DS layer. Finally, the mass functions are used to compute the utilities of acts assigning pixels to a set of classes for semantic segmentation in a so-called "utility layer". An end-to-end learning procedure allows us to train the E-FCN using a learning set with soft labels. The effectiveness of the E-FCN is demonstrated and discussed based on experiments using three benchmark databases (Pascal VOC 2011 [17], MIT-scene Parsing [43], and SIFT Flow [34]).

The rest of the paper is organized as follows. Section 2 starts with a brief reminder of DS theory, the DS layer for constructing mass functions, and feature representation via FCN. The E-FCN model is then introduced in Section 3. Section 4 presents numerical experiments, which demonstrate the advantages of the E-FCNs. Finally, we conclude the paper in Section 5.

2 Background

This section first recalls some necessary definitions regarding DS theory (Section 2.1), and the evidential neural network at the basis of the DS layer (Section 2.2). A brief description of feature representation via FCNs is then provided in Section 2.3.

2.1 Dempster-Shafer theory

The main concepts underlying DS theory are only briefly presented in this section, and some basic notations are introduced. Detailed information can be found in Shafer's original work [30] and in the recent review [13].

Let $\Omega = \{\omega_1, \ldots, \omega_M\}$ be a set of classes, called the *frame of discernment*. A *mass function* on Ω is a mapping m from 2^Ω to [0,1] such that $m(\emptyset) = 0$ and

$$\sum_{A \subseteq \Omega} m(A) = 1. \tag{1}$$

For any $A \subseteq \Omega$, each mass $m(A)$ is interpreted as a share of a unit mass of belief allocated to the hypothesis that the truth is in A, and which cannot be allocated to any strict subset of A based on the available evidence. Set A is called a *focal set* of m if $m(A) > 0$. A mass function is said to be *logical* if it has only one focal set.

Two mass functions m_1 and m_2 representing independent items of evidence can be combined conjunctively by Dempster's rule $\oplus$ [30] as

$$(m_1 \oplus m_2)(A) = \frac{(m_1 \cap m_2)(A)}{1 - (m_1 \cap m_2)(\emptyset)} \tag{2a}$$

for all $A \neq \emptyset$, with

$$(m_1 \cap m_2)(A) = \sum_{B \cap C = A} m_1(B)\, m_2(C), \tag{2b}$$

and $(m_1 \oplus m_2)(\emptyset) = 0$. Mass functions m_1 and m_2 can be combined if and only if the denominator on the right-hand side of Eq. (2a) is strictly positive. The operator $\oplus$ is commutative and associative.

For decision-making with belief functions, let $u_{ij} \in [0, 1]$ denote the utility of selecting ω_i when the true state is ω_j, and f_{ω_i} the act of selecting ω_i. We define the *pignistic expected utility* [11] of act f_{ω_i} as

$$\mathbb{E}_m(f_{\omega_i}) = \sum_{j=1}^{M} u_{ij} BetP_m(\{\omega_j\}), \tag{3a}$$

where $BetP_m$ is the *pignistic* probability measure computed from mass function m by the *pignistic transformation*, defined as

$$BetP_m(\{\omega_j\}) = \sum_{\{A \subseteq \Omega : \omega_j \in A\}} \frac{m(A)}{|A|}, \tag{3b}$$

for all $\omega_j \in \Omega$. The act with the highest pignistic expected utility can then be selected. Other decision criteria in the belief function framework are reviewed in [11] and [26].

2.2 Evidential neural network

Denœux [9] proposed a distance-based neural-network based on DS theory, known as the *evidential neural network (ENN) classifier*. The ENN classifier summarizes the learning set by a small number of prototypes, and treats the proximity of an input vector to each prototype as a piece of evidence about its class. The different pieces of evidence are represented by mass functions, which are combined using Dempster's rule (2). This section provides a brief description of the ENN classifier.

We consider a training set $\mathcal{X} = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N\} \subset \mathbb{R}^P$ of N examples represented by P-dimensional feature vectors, and n prototypes $\{\boldsymbol{p}^1, \ldots, \boldsymbol{p}^n\} \subset \mathbb{R}^P$. For a test sample $\boldsymbol{x}$, the ENN classifier constructs mass functions that quantify the uncertainty about its class in $\Omega = \{\omega_1, \ldots, \omega_M\}$, using a three-step procedure. This procedure can be implemented in a neural network with two hidden layers and one output layer. These three layers can be considered as a single complex layer called the "DS layer", which will be plugged into an FCN architecture as explained in Section 3.1. The three-step procedure can be described as follows.

Step 1: The similarity between $\boldsymbol{x}$ and each prototype $\boldsymbol{p}^l$ is computed as

$$s^l = \alpha^l \exp\left(-\left(\eta^l d^l\right)^2\right), \quad l = 1, \ldots, n, \tag{4}$$

where $d^l = \left\|\boldsymbol{x} - \boldsymbol{p}^l\right\|$ is the Euclidean distance between $\boldsymbol{x}$ and prototype $\boldsymbol{p}^l$, $\eta^l \in \mathbb{R}$ is a scale parameter and α^l is a parameter in $[0, 1]$. Prototypes $\boldsymbol{p}^1, \ldots, \boldsymbol{p}^n$ can be considered as vectors of connection weights between the input layer and a hidden layer of n Radial Basis Function (RBF) units. The number n of prototypes is a hyper-parameter and can be tuned using a validation set or by cross-validation.

Step 2: The mass function m^l associated to reference pattern $\boldsymbol{p}^l$ is computed as

$$m^l(\{\omega_j\}) = v_j^l s^l, \quad j = 1, \ldots, M \tag{5a}$$

$$m^l(\Omega) = 1 - s^l, \tag{5b}$$

where $v_j^l \geq 0$ is the degree of membership of prototype $\boldsymbol{p}^l$ to class ω_j with $\sum_{j=1}^M v_j^l = 1$. We denote the vector of masses induced by prototype $\boldsymbol{p}^l$ as

$$\boldsymbol{m}^l = (m^l(\{\omega_1\}), \ldots, m^l(\{\omega_M\}), m^l(\Omega))^T.$$

Eq. (5) can be regarded as computing the activation of units in a "mass functions" layer composed of n modules of $M + 1$ units each. The activations of the units in module l correspond to the belief masses assigned by m^l.

Step 3: The n mass functions $\boldsymbol{m}^l$, $l = 1, \ldots, n$, are aggregated by Dempster's rule (2). The combined mass function can be computed iteratively as $\mu^1 = m^1$ and $\mu^l = \mu^{l-1} \cap m^l$ for $l = 2, \ldots, n$. From (2a), we have

$$\mu^l(\{\omega_j\}) = \mu^{l-1}(\{\omega_j\}) m^l(\{\omega_j\}) + \mu^{l-1}(\{\omega_j\}) m^l(\{\Omega\}) + \mu^{l-1}(\Omega) m^l(\{\omega_j\}) \tag{6a}$$

for $l = 2, \ldots, n$ and $j = 1, \ldots, M$, and

$$\mu^l(\Omega) = \mu^{l-1}(\Omega) m^l(\Omega) \quad l = 2, \ldots, n. \tag{6b}$$

The output vector $\boldsymbol{m} = (m(\{\omega_1\}), \ldots, m(\{\omega_M\}), m(\Omega))^T$ is finally obtained by normalizing μ^n as

$$m(A) = \frac{\mu^n(A)}{\mu^n(\Omega) + \sum_{j'=1}^M \mu^n(\{\omega_{j'}\})},$$

with $A \in \{\{\omega_1\}, \ldots, \{\omega_M\}, \Omega\}$.

The network parameters are the prototypes $\boldsymbol{p}^l$, the coefficients α^l and η_l, and the membership degrees v_j^l for $l = 1, \ldots, n$ and $j = 1, \ldots, M$. They are learnt by minimizing a loss function. To enforce the constraints $0 \leq \alpha^l \leq 1$, we introduce new variables $\xi^l \in \mathbb{R}$ such that

$$\alpha^l = \frac{1}{1 + \exp(-\xi^l)} \in (0, 1).$$

Similarly, the constraints on parameters v_j^l are enforced by introducing new parameters $\delta_j^l \in \mathbb{R}$ such that

$$v_j^l = \frac{(\delta_j^l)^2}{\sum_{j'=1}^{M}(\delta_{j'}^l)^2} \tag{7}$$

for $l = 1, \ldots, n$ and $j = 1, \ldots, M$. More details can be found in [9].

2.3 Fully convolutional network

The performance of a classifier in semantic segmentation tasks heavily depends on the information contained in its input features. Feature representation, an essential part of the machine learning workflow, consists in discovering the predictors needed for semantic segmentation from input images. In recent years, FCNs [25] and their variants [22, 28] have achieved remarkable performances thanks to their ability to construct rich pixel-wise deep feature representations.

FCNs owe their name to their architecture, which is built only from locally connected layers, such as convolution, pooling, and upsampling layers. No dense layer is used in this kind of architecture. Generally, an FCN consists of two main parts: an encoder-decoder architecture for pixel-wise object representation and a softmax layer for pixel-wise assignments. In the encoder-decoder architecture, an input image is encoded by several convolutional and pooling layers and then decoded by one or more upsampling layers. The softmax layer assigns each pixel in the input image to one of the classes based on the outputs of the encoder-decoder architecture. Therefore, the outputs of the encoder-decoder architecture, called the *pixel-wise feature maps*, are considered as a feature representation of the input image. In the study, these feature maps are used as input to a DS layer allowing for set-valued semantic segmentation, as will be shown in Section 3.1.

To understand the feature representation of FCNs, we briefly recall the encoder-decoder architecture illustrated in Figure 1. The encoder part consists of several convolutional and pooling layers. Each convolutional layer performs convolutions its input to produce a set of feature maps. Let $\boldsymbol{z} = (z^1, \ldots, z^D)$ be the input made up of D input maps or *input channels* z^i $(i = 1, \ldots, D)$ of size $H \times W$. The processes in a convolutional layer with input $\boldsymbol{z}$, consisting of e convolution kernels with size $a \times b$, are expressed as

$$c^j = f(\lambda^j + \sum_i \varepsilon^{i,j} * z^i), \tag{8}$$

where $\varepsilon^{i,j}$, a matrix of size $a \times b$, is the convolution kernel between the i-th input map and the j-th output map; λ^j is the bias of kernel $\varepsilon^{i,j}$; $*$ denotes the convolution operation; c^j is the j-th output feature map, with size $\frac{h-a+1}{r} \times \frac{w-b+1}{r}$, $j = 1, \ldots, e$; r is the stride with which the kernel slides over input map z^i, and f is the activation function, such as the rectified linear unit $\mathsf{ReLU}(x) = \max(0, x)$ [23]. A pooling layer follows the convolutional layer to sub-sample feature map c^j by computing some statistics of feature values within non-overlapping $s \times s$ windows. In the case of max-pooling used in this paper, the statistic is the maximum. Thus, the outputs of the pooling layer is composed of the D feature maps sub-sampled by factor s. For example, feature map c^j with size $\frac{h-a+1}{r} \times \frac{w-b+1}{r}$ is downsized to $\frac{h-a+1}{2r} \times \frac{w-b+1}{2r}$ by a pooling layer with a 2×2 non-overlapping window.

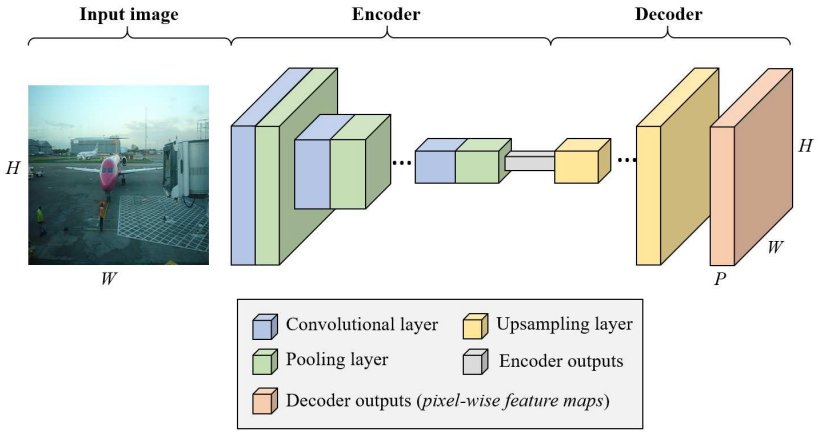

Fig. 1: Illustration of the encoder-decoder architecture. An encoder downsizes its input by convolution and pooling operations. The outputs of the encoder, as the sparse feature maps, are imported into a decoder. A decoder upsamples and densifies its inputs by performing the reverse operation of convolution and pooling. The final decoder outputs are the pixel-wise feature maps.

Although the convolution and pooling operations in the encoder part contribute to feature representation by retaining only robust activations, spatial information within a receptive field is lost, which may be critical for image semantic segmentation. To address the issue, a decoder part made up of one or more upsampling layers is added at the output of the encoder part. The decoder performs the reverse operation of convolution and pooling for reconstructing a set of activations with the same size of the input image, as shown in Figure 1. Thus, the outputs of the decoder part are enlarged feature maps. In this study, we use a *deconvolution layer* [28] to implement the upsampling operation.

A deconvolutional layer densifies its inputs of sparse feature maps through convolution-like operations with multiple learned kernels. However, contrary to convolutional layers, which connect multiple inputs within a kernel to a single activation, a deconvolutional layer associates a single input in a feature map to multiple outputs. Thus, the outputs of a deconvolutional layer are enlarged and dense feature maps. The processes of a deconvolution operation can also be summarized as Eq. (8), but its kernel sizes are larger than the input sizes, i.e., $a \geq H$ and $b \geq W$.

3 Evidential fully convolutional network

In this section, we describe the proposed E-FCN. Section 3.1 presents the overall architecture composed of an encoder-decoder module for feature representation, a DS layer to construct mass functions, and a utility layer for decision-making. The details of the utility layer are described in Section 3.2. Section 3.3 introduces the strategy for training E-FCN models using a learning set with soft labels.

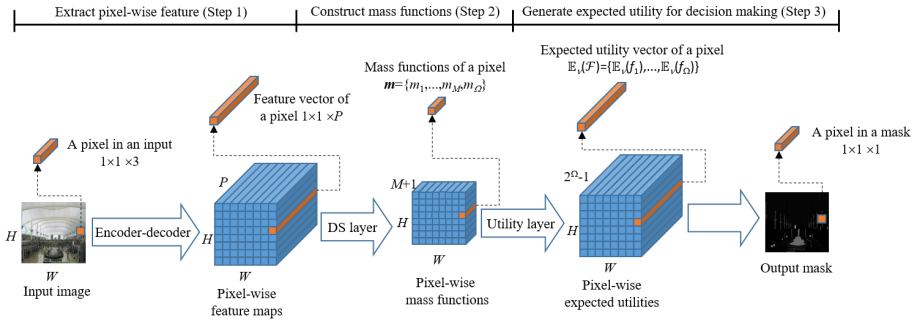

Fig. 2: Architecture of an evidential fully convolutional network (E-FCN). The E-FCN performs semantic segmentation using a three-step procedure. In the first step, an encoder-decoder architecture extracts pixel-wise feature maps from the input image. Each vector in the feature maps is fed into a DS layer to construct the pixel-wise mass functions in the second step. These mass functions are finally fed into a utility layer to generate the pixel-wise expected utilities of all acts. Finally, the segmentation mask is computed based on the expected utilities.

3.1 Network architecture

The main idea of this work is to hybridize the ENN classifier presented in Section 2.2 and the FCN recalled in Section 2.3 by "plugging" a DS layer followed by a utility layer at the output of the final deconvolutional layer in the FCN. The architecture of the proposed method, called the *evidential FCN* (E-FCN), is illustrated in Figure 2. An E-FCN classifier performs set-valued semantic segmentation and quantifies the uncertainty about the class of each pixel, taking values in $\Omega = \{\omega_1, \ldots, \omega_M\}$, using a three-step procedure defined as follows.

Step 1: As in a probabilistic FCN (P-FCN), an image of size $W \times H \times 3$ is presented as input to the encoder-decoder architecture of an FCN to generate pixel-wise feature maps of size $W \times H \times P$, where P is the number of *output channels*. Each feature vector $1 \times 1 \times P$ from a pixel-wise feature map is a P-dimensional representation of the corresponding pixel, ready to be fed into the DS layer. This architecture generates reliable pixel-wise representations of the input image. Thanks to the representations, the E-FCN yields similar or even better performance for precise semantic segmentation than does a P-FCN with the same encoder-decoder architecture, as will be shown in Section 4.2.

Step 2: Each feature vector from the encoder-decoder architecture is fed into the DS layer, in which it is used to compute a mass function as explained in Section 2.2. The output of the DS layer for a given feature vector is an $(M+1)$-dimensional mass vector

$$\boldsymbol{m} = (m(\{\omega_1\}), \ldots, m(\{\omega_M\}), m(\Omega))^T.$$

Thus, given pixel-wise feature maps of size $W \times H \times P$ from Step 1, the output of the DS layer is a tensor of size $W \times H \times (M+1)$. Each mass vector in the tensor represents the uncertainty about the class of the corresponding pixel. More precisely, the mass $m(\{\omega_i\})$ is the degree of belief that the true

class of the corresponding pixel is ω_i. The DS layer tends to allocate uniform masses if the representations contain confusing information. The additional degree of freedom $m(\Omega)$ makes it possible to quantify the lack of evidence [12] and to verify whether the model is well trained [35]. The advantages of this uncertainty representation will be demonstrated in the performance evaluation of set-valued semantic segmentation using E-FCN in Section 4.3.

Step 3: The output pixel-wise mass vectors are fed into a utility layer, where they are used to compute the expected utility of acts. Each act is defined as the assignment of a pixel to a non-empty subset A of Ω. Therefore, the output of the layer for each feature vector from Step 2 is a vector of at most $2^\Omega - 1$ expected utilities when all of the possible acts are considered. The utility layer allows the E-FCN to perform set-valued semantic segmentation. This capability will be demonstrated by comparing the performances of the two types of FCNs in set-valued segmentation (Section 4.3) and novelty detection (Section 4.4) tasks. More details about the utility layer are given in the next section.

3.2 Utility layer

In this section, we describe in greater detail the decision-making process taking place in the utility layer. Section 3.2 begins by introducing the precise semantic segmentation method using mass functions and utilities. Section 3.2 then describes a method for computing the utility of set-valued pixel-wise classification, after which an approach to set-valued classification based on mass functions is described in Section 3.2. In Section 3.2, we summarize the workflow as a neural network layer in the E-FCN model.

3.2.1 Precise semantic segmentation

Let $\Omega = \{\omega_1, \ldots, \omega_M\}$ be the set of classes. For semantic segmentation problems with precise prediction, each pixel in an image is assigned to exactly one class. An act is thus defined as the assignment of a pixel to one and only one of the M classes, and the set of acts is $\mathcal{F} = \{f_{\omega_1}, \ldots, f_{\omega_M}\}$, where f_{ω_i} denotes assignment to class ω_i. To make decisions, we define a utility matrix $\boldsymbol{U}$ of size $M \times M$, whose general term $u_{ij} \in [0, 1]$ is the utility of assigning a pixel to class ω_i when the true class is ω_j.

When uncertainty about Ω is described by a DS mass function, each act f_{ω_i} induces an expected utility, such as the pignistic expected utility defined by Eq. (3). Given utility matrix $\boldsymbol{U}$ and the output of the DS layer $\boldsymbol{m}$ for a given pixel, the pignistic expected utility of assigning that pixel to class ω_i is given by Eq. (3a), where $BetP_m$ is the pignistic probability defined by Eq. (3b). The pixel is finally assigned to class ω_{i^*} such that

$$i^* = \arg \max_{i \in \{1,\ldots,M\}} \mathbb{E}_m(f_{\omega_i}). \tag{9}$$

3.2.2 Extending the utility matrix

For semantic segmentation problems with imprecise prediction, we adopt the approach described in [26] for set-valued classification under uncertainty, which allows the assignment of a pixel to any non-empty subset A of Ω. The set of acts thus potentially becomes $\mathcal{F} = \{f_A : A \subseteq \Omega, A \neq \emptyset\}$, where f_A denotes the assignment to a subset A. (In practice, when the cardinality of Ω is very large, we may only consider acts f_A for *some* subsets A of Ω). In this study, f_A is referred to as an *imprecise assignment* when subset A is a *multi-class set* with $|A| \geq 2$. For decision-making with $\mathcal{F}$, the utility matrix $\boldsymbol{U}$ has to be extended to a matrix $\widetilde{\boldsymbol{U}}$ of size $(2^M - 1) \times M$, where each element $\widetilde{u}_{A,j}$ denotes the utility of assigning a pixel to set A of classes when the true label is ω_j. Following [26], this extension is performed as follows.

When the true class is ω_j, the utility of assigning a pixel to set A is defined as an *Ordered Weighted Average (OWA)* aggregation [37] of the utilities of each precise assignment in A as

$$\widetilde{u}_{A,j} = \sum_{k=1}^{|A|} g_k \, u^A_{(k)j}, \tag{10}$$

where $u^A_{(k)j}$ is the k-th largest element in the set $\{u_{ij} : \omega_i \in A\}$ made up of the elements in the utility matrix $\boldsymbol{U}$, and weights $\boldsymbol{g} = (g_1, \ldots, g_{|A|})$ represent the preference to choose $u_{(k)j}(A)$ if forced to select a single value in $\{u_{ij} : \omega_i \in A\}$. The components of weight vector $\boldsymbol{g}$ represent the *tolerance to imprecision* of a decision maker (DM). For example, full tolerance to imprecision is achieved when the assignment act f_A has utility 1 once set A contains the true label, no matter how large A is. In this case, only the maximum utility of elements in set $\{u_{ij}, \omega_i \in A\}$ is considered: $(g_1, g_2, \ldots, g_{|A|}) = (1, 0, \ldots, 0)$. At the other extreme, a DM attaching no value to imprecision would consider the act f_A as equivalent to selecting one class uniformly at random from A: this is achieved when

$$(g_1, g_2, \ldots, g_{|A|}) = \left(\frac{1}{|A|}, \frac{1}{|A|}, \ldots, \frac{1}{|A|}\right),$$

in which case the OWA operator becomes the average. In this study, following [26], we determine the weight vector $\boldsymbol{g}$ of the OWA operator by adapting O'Hagan's method [29]. We define the tolerance to imprecision as

$$TDI(\boldsymbol{g}) = \sum_{k=1}^{|A|} \frac{|A| - k}{|A| - 1} g_k = \gamma, \tag{11}$$

which equals 1 for the maximum, 0 for the minimum, and 0.5 for the average. In practice, we only need to consider values of γ between 0.5 and 1 as a precise assignment is always more desirable than an imprecise one when γ<0.5 [26]. Given a value of γ, we can compute the weights of the OWA operator by maximizing the entropy

$$ENT(\boldsymbol{g}) = -\sum_{k=1}^{|A|} g_k \log g_k \tag{12}$$

subject to the constraints $TDI(\boldsymbol{g}) = \gamma$, $\sum_{k=1}^{|A|} g_k = 1$, and $g_k \geq 0$.

Table 1: Utility matrix extended by an OWA operator with $\gamma = 0.8$.

	Classes		
	ω_1	ω_2	ω_3
$f_{\{\omega_1\}}$	1	0	0
$f_{\{\omega_2\}}$	0	1	0
$f_{\{\omega_3\}}$	0	0	1
$f_{\{\omega_1,\omega_2\}}$	0.8	0.8	0
$f_{\{\omega_1,\omega_3\}}$	0.8	0	0.8
$f_{\{\omega_2,\omega_3\}}$	0	0.8	0.8
f_{Ω}	0.6819	0.6819	0.6819

Example 1 *Table 1 shows an example of the extended utility matrix generated by an OWA operator with $\gamma = 0.8$. The first three rows constitute the original utility matrix, indicating that the utility equals 1 when assigning a sample to its true class, and 0 otherwise. The remaining rows are the matrix of the aggregated utilities. For example, we get a utility of 0.8 when assigning a sample to set $\{\omega_1, \omega_2\}$ if the true label is ω_1.*

3.2.3 Set-valued semantic segmentation

Based on an extended utility matrix $\widetilde{\boldsymbol{U}}$ and the output of the DS layer $\boldsymbol{m}$ for a given pixel, we can compute the pignistic expected utility of assigning that pixel to set A as

$$\mathbb{E}_m(f_A) = \sum_{j=1}^{M} \widetilde{u}_{A,j} BetP_m(\{\omega_j\}), \tag{13}$$

where $BetP_m$ is the pignistic probability defined by Eq. (3b). The pixel is finally assigned to set A such that

$$A = \arg \max_{\emptyset \neq B \subseteq \Omega} \mathbb{E}_m(f_B). \tag{14}$$

3.2.4 Utility layer

The procedure of assigning a pixel to a set of classes using utility theory is implemented in the neural network as a *utility layer*. In this layer, the inputs and outputs are, respectively, the pixel-wise mass vectors $\boldsymbol{m}$ from the preceding DS layer and the pixel-wise expected utilities of all acts in $\mathcal{F}$. The connection weight between unit j of the DS layer and output unit $A \subseteq \Omega$ corresponding to the assignment to set A is the utility value $\widetilde{u}_{A,j}$. As coefficient γ describing the imprecision tolerance degree is predetermined, the connection weights of the expected utility layer are fixed and do not need to be updated during training.

In practice, the connections between the DS and utility layers can be designed by the user. For example, one can build a utility layer using the utility values $\widetilde{u}_{A,j}$ with $|A| = 1$ to only consider precise assignments, or $0 < |A| \leq 2$ to consider assignment to sets classes of cardinality one or two. In this paper, we have only

considered the acts f_A such that A is a singleton, Ω, or one of the soft labels present in the learning set (as explained in Section 3.3 below).

3.3 Learning with soft labels

In traditional learning systems for image semantic segmentation, all pixels are labeled with a single class even when their true class cannot be determined with full certainty. For example, the true class may be uncertain at object borders, but the border pixels are still given precise labels. Additionally, one cannot reliably label some small objects in an image, such as distant objects in a driving scene. Arbitrarily giving precise labels to pixels with confusing information may negatively impact the performance of learning systems in image semantic segmentation tasks. The notion of *soft label* [5, 14] is a way to address this problem.

Here, we define a soft label as a nonempty subset $A_* \in 2^\Omega \backslash \emptyset$ of classes a pixel may belong to, based on our current knowledge. For example, label $A_* = \{\omega_i, \omega_j\}$ indicates that the true class of a pixel is known to be either ω_i or ω_j but we cannot determine which one specifically. A strategy of end-to-end learning is proposed to train an E-FNC from an image learning set with soft labels. All parameters in the DS layer are first initialized randomly using normal distributions. For a given pixel with nonempty soft label $A_* \subseteq \Omega$, let m_l be the logical mass function with focal set A_*, i.e., such that $m_l(A_*) = 1$. The *labeling* pignistic expected utilities $\mathbb{E}_{\boldsymbol{m}_l}(f_A)$ for $A \in 2^\Omega \backslash \emptyset$ can be computed using Eq. (13) and the pignistic belief-probability transformation Eq. (3b). Similarly, we consider the *predicted* pignistic expected utilities $\mathbb{E}_m(f_A)$ for $A \in 2^\Omega \backslash \emptyset$, where m is the predicted mass function from the DS layer of the E-FCN, with focal sets $\{\omega_1\}, \ldots, \{\omega_M\}$ and Ω. For a given pixel with soft label m_l and predicted mass function m, the loss $\mathcal{L}(m, m_l)$ is defined as the squared Euclidean distance between the vectors of expected utilities w.r.t. m_l and m:

$$\mathcal{L}(m, m_l) = \sum_{\emptyset \neq A \subseteq \Omega} \left[\mathbb{E}_{m_l}(f_A) - \mathbb{E}_m(f_A)\right]^2 . \tag{15}$$

The derivatives of $\mathcal{L}(m, m_l)$ of the error w.r.t the output masses $m(\{\omega_k\})$ are

$$\begin{aligned}
\frac{\partial \mathcal{L}(m, m_l)}{\partial m(\{\omega_k\})} &= \sum_{\emptyset \neq A \subseteq \Omega} \frac{\partial \mathcal{L}(m, m_l)}{\partial \mathbb{E}_m(f_A)} \cdot \frac{\partial \mathbb{E}_m(f_A)}{\partial m(\{\omega_k\})} \\
&= -2 \sum_{\emptyset \neq A \subseteq \Omega} \left[\mathbb{E}_{m_l}(f_A) - \mathbb{E}_m(f_A)\right] \sum_{j=1}^{M} \frac{\partial \mathbb{E}_m(f_A)}{\partial BetP_m(\omega_j)} \frac{\partial BetP_m(\omega_j)}{\partial m(\{\omega_k\})} \\
&= -2 \sum_{\emptyset \neq A \subseteq \Omega} \left[\mathbb{E}_{m_l}(f_A) - \mathbb{E}_m(f_A)\right] \sum_{j=1}^{M} \widetilde{u}_{A,j} \left(\delta_{kj} - \frac{1}{M}\right),
\end{aligned} \tag{16}$$

where $\delta_{kj} = 1$ if $k = j$ and $\delta_{kj} = 0$ otherwise. The derivatives of $m(\{\omega_k\})$ w.r.t p_k^l, η^l, and ξ^l in the DS layer are the same as in Denœux's original work [9], and the gradient with respect to all network parameters can be back-propagated from the output layer to the input layer.

4 Experiments

In this section, we present numerical experiments that demonstrate the advantages of the proposed model. The databases and metrics are first introduced in Section 4.1. Precise and imprecise segmentation results are then reported, respectively, in Sections 4.2 and 4.3. Finally, novelty detection results are presented in Section 4.4.

4.1 Databases and metrics for performance evaluation

Databases

Three benchmark databases were used in the study: Pascal VOC 2011 [17], MIT-scene Parsing [43], and SIFT Flow [34]. These databases were used to train and test the E-FCNs as well as probabilistic FCNs (P-FCNs) for comparison.

The Pascal VOC 2011 database contains 20 object classes in 5034 images, with segmentation masks that indicate the class of each pixel, or label it as "background" if the object does not belong to one of the twenty specified classes. The MIT-scene Parsing and SIFT Flow databases are similar to the Pascal VOC 2011 database but have, respectively, 150 categories in 20K images and 33 classes in 2688 images. The list of classes for the three databases are given in Table 2. Each of the three databases was split into 50% for training/validation and 50% for testing. The validation sets were used to determine hyper-parameters, such as the number of prototypes in each DS layer. The optimal tolerance to imprecision γ can be determined in the same way since it can also be considered as a hyper-parameter.

There is no confidence value associated with the pixel labels in any of the three databases. Thus, we defined soft labels for them. For the Pascal VOC 2011 database, we assigned each pixel in a boundary area a soft label $A \subseteq \Omega$, where A consists of the object classes around the boundary area. Some examples are shown in Figure 3a. For the MIT-scene Parsing and SIFT Flow databases with no identified boundary areas, we assigned soft labels to the pixels situated between every two objects, as shown in Figures 3b and 3c.

A semantic segmentation model should not only be accurate for the classes in the learning set, but it should also be able to detect objects belonging to classes that are not included in the learning set. To evaluate this novelty detection capacity, we mixed the three databases: for example, an FCN model trained using the Pascal VOC 2011 database was tested on the other two databases.

Metrics

We used three metrics for the performance evaluation of semantic segmentation: pixel utility (PU), utility of intersection over union (UIoU), and expected calibration error (ECE).

Pixel utility. For an image with T pixels, the *pixel utility* is defined as

$$PU = \frac{1}{|T|} \sum_{i=1}^{|T|} \widetilde{u}_{A(i),A_*(i)} \tag{17}$$

Table 2: Lists of classes for the three databases used in this study. Classes in bold characters are included in two or three databases. Classes with close meanings, such as "minibike" and "motorbike", are considered as identical.

Database	Class list
Pascal VOC 2011	background, cat, dog, horse, sheep, train, **sofa**, **aeroplane**, **bicycle**, **bird**, **boat**, **bottle**, **bus**, **car**, **chair**, **cow**, **diningtable**, **motorbike**, **person**, **pottedplant**, **tv**.
MIT-scene parsing	wall, floor, ceiling, bed, cabinet, earth, curtain, water, painting, shelf, house, mirror, rug, armchair, seat, desk, wardrobe, lamp, bathtub, railing, cushion, base, box, column, chest, counter, sink, skyscraper, fireplace, refrigerator, grandstand, path, stairs, runway, case, pool, pillow, screen, bookcase, blind, coffee, toilet, flower, book, hill, bench, countertop, stove, palm, kitchen, computer, swivel, bar, arcade, hovel, towel, light, truck, tower, chandelier, booth, dirt track, apparel, land, bannister, escalator, ottoman, buffet, poster, stage, van, ship, fountain, conveyer, canopy, washer, plaything, swimming, stool, barrel, basket, waterfall, tent, bag, minibike, cradle, oven, ball, food, step, tank, trade, microwave, pot, animal, lake, dishwasher, screen, blanket, sculpture, hood, sconce, vase, traffic, tray, ashcan, fan, pier, screen, plate, monitor, bulletin, shower, radiator, glass, clock, flag, **sofa**, **airplane**, **building**, **sky**, **tree**, **road**, **windowpane**, **grass**, **sidewalk**, **person**, **door**, **table**, **mountain**, **plant**, **chair**, **car**, **sea**, **field**, **fence**, **rock**, **sign**, **sand**, **staircase**, **river**, **bridge**, **boat**, **bus**, **awning**, **streetlight**, **tv**, **pole**, **bottle**, **minibike**, **bicycle**.
SIFT Flow	balcony, crosswalk, desert, moon, sun, window, **awning**, **bird**, **boat**, **bridge**, **building**, **bus**, **car**, **cow**, **door**, **fence**, **field**, **grass**, **mountain**, **person**, **plant**, **pole**, **river**, **road**, **rock**, **sand**, **sea**, **sidewalk**, **sign**, **sky**, **staircase**, **streetlight**, **tree**.

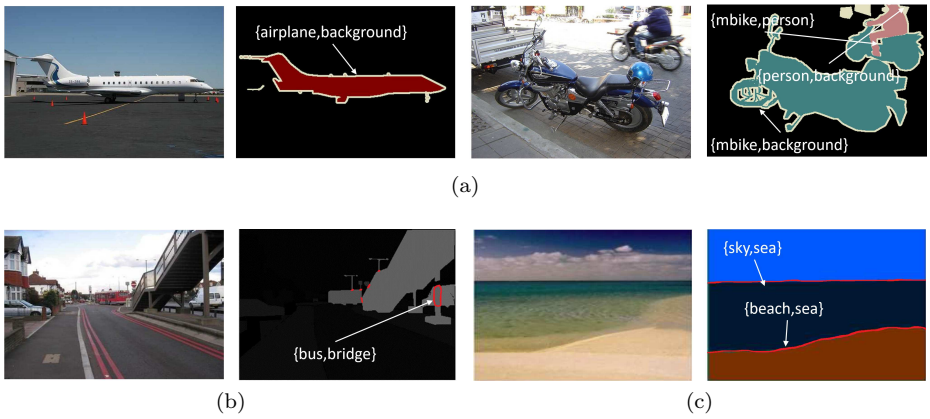

Fig. 3: Segmentation masks with soft labels: (a) Pascal VOC 2011, (b) MIT-scene Parsing, and (c) SIFT Flow.

where $A_*(i)$ is the label of pixel i, $A(i)$ is the selected set of classes for pixel i determined from Eq. (14), and using the notations introduced in Section 3.2, $\widetilde{u}_{A(i),A_*(i)}$ is the utility of assigning pixel i to subset $A(i) \subseteq \Omega$ when its label is $A_*(i)$. Thus, PU is the same as pixel accuracy when only considering precise assignments and precise labels. To consider soft labels, the utility matrix $\widetilde{\boldsymbol{U}}$ defined

Table 3: Utility matrix considering soft labels with $\gamma = 0.8$.

		Label						
		ω_1	ω_2	ω_3	$\{\omega_1, \omega_2\}$	$\{\omega_1, \omega_3\}$	$\{\omega_2, \omega_3\}$	Ω
Act	$f_{\{\omega_1\}}$	1	0	0	0.625	0.625	0	0.489
	$f_{\{\omega_2\}}$	0	1	0	0.625	0	0.625	0.489
	$f_{\{\omega_3\}}$	0	0	1	0	0.625	0.625	0.489
	$f_{\{\omega_1,\omega_2\}}$	0.8	0.8	0	1	0.5	0.5	0.782
	$f_{\{\omega_1,\omega_3\}}$	0.8	0	0.8	0.5	1	0.5	0.782
	$f_{\{\omega_2,\omega_3\}}$	0	0.8	0.8	0.5	0.5	1	0.782
	f_{Ω}	0.682	0.682	0.682	0.853	0.853	0.853	1

in Section 3.2 should be extended to a matrix $\widetilde{\boldsymbol{U}}'$ of size $(2^M - 1) \times (2^M - 1)$ with general term $\widetilde{u}_{A,A_*}$ defined as the utility of assigning a pixel to subset $A \subseteq \Omega$ when its label is A_*, with $|A_*| \geq 1$. Soft label A_* means that we only know the true class of a pixel is in set A_*, and nothing more. To define the utility $\widetilde{u}_{A,A_*}$, we first compute the average of the utilities of selecting subset A when the true class is in A_* as

$$\overline{u}_{A,A_*} = \frac{1}{|A_*|} \sum_{w_k \in A_*} \widetilde{u}_{A,k}, \tag{18a}$$

where $\widetilde{u}_{A,k}$ is the utility of selecting subset A when the true class is ω_k, and we normalize this average utility to ensure that $\widetilde{u}_{A_*,A_*} = 1$:

$$\widetilde{u}_{A,A_*} = \frac{\overline{u}_{A,A_*}}{\overline{u}_{A_*,A_*}}. \tag{18b}$$

Example 2 *Table 3 shows an example of the utility matrix considering soft labels, which is extended from Example 1. The last four columns correspond to the utility matrix for soft labels. An act achieves utility 1 only if $A = A_*$, 0 if $A \cap A_* = \emptyset$, and a value between 0 and 1 if $A \neq A_*$ and $A \cap A_* \neq \emptyset$.*

Utility of intersection over union. The segmentation performance was also evaluated by *the utility of intersection over union* (UIoU) defined as

$$UIoU = \frac{1}{2^{|\Omega|} - 1} \sum_{B \subseteq \Omega} \frac{\sum_{i \in \boldsymbol{G}^B \cap \boldsymbol{P}^B} \widetilde{u}_{A(i),B}}{|\boldsymbol{G}^B \cup \boldsymbol{P}^B|}, \tag{19}$$

where $\boldsymbol{P}^B = \{i : A(i) \cap B \neq \emptyset\}$ is the predicted area containing pixels assigned to a set of classes that intersect B, and $\boldsymbol{G}^B = \{i : A_*(i) = B)\}$ is the ground truth area composed of pixels with label B. Thus, in the special case of precise segmentation with only precise labels, UIoU boils down to *intersection over union*, a widely used metric for semantic segmentation [22, 25, 28].

Expected calibration error. In decision systems, a neural network should not only be accurate, but it should also indicate when it is likely to be incorrect. Thus, the confidence of an E-FCN should be *calibrated*. To characterize this property,

we extend the *expected calibration error* (ECE) introduced in [20] as follows. We define the *prediction confidence* of pixel i as

$$co(i) = BetP_i(A_*(i)) = \sum_{\omega_j \in A_*(i)} BetP_i(\{\omega_j\}), \tag{20}$$

where $BetP_i$ is the predicted pignistic probability measure for pixel i. Let I_q be the set of pixels whose prediction confidence lies in the interval $(\frac{q-1}{Q}, \frac{q}{Q}]$, $q = 1, \ldots, Q$. The average utility and confidence of I_q are defined, respectively, as

$$au(I_q) = \frac{1}{|I_q|} \sum_{i \in I_q} \widetilde{u}_{A(i), A_*(i)}, \tag{21a}$$

and

$$co(I_q) = \frac{1}{|I_q|} \sum_{i \in I_q} co(i). \tag{21b}$$

We consider that the classifier is well calibrated if $co(I_q) \approx au(I_q)$ for all q, and we define the ECE as

$$ECE = \frac{\sum_{q=1}^{Q} |I_q| \times |co(I_q) - au(I_q)|}{\sum_{q'=1}^{Q} |I'_q|} \tag{22}$$

When only considering precise acts and labels, ECE defined by (22) boils down to the original definition in [20].

4.2 Precise segmentation results

In precise segmentation, each pixel of an image is assigned to exactly one class, the set of acts being defined as $\mathcal{F} = \{f_{\omega_1}, \ldots, f_{\omega_M}\}$. Three databases without soft labels mentioned in Section 4.1 were used to train and test the E-FCNs and probabilistic FCNs (P-FCNs). The metrics defined in Section 4.1 with the utility matrix $\boldsymbol{U}$ equal to the identity matrix were used for performance assessment.

In the experiment with each database, three widely used encoder-decoder architectures were combined with the DS and utility layers, as shown in Table 4. All encoder-decoder architectures in Table 4 have the same encoder part, which consists of four stages and two convolutional layers with 3×3 kernels. Each stage is made up of three convolutional layers with 3×3 kernels and a max-pooling layer with a 2×2 non-overlapping window. Figure 4a illustrates the differences between the FCN-32s, FCN-16s, and FCN-8s architectures in their decoder parts with a deconvolutional layer. The FCN-SegNet architecture uses four deconvolutional layers to upsample the sparse feature maps from the end of the encoder part, as well as the feature maps from the corresponding pooling layers based on pooling indices [1], as shown in Figure 4b. The FCN-DilatedVGG architecture is the same as FCN-SegNet except that it adds a fully connected conditional random field at the end of the last deconvolutional layer [3]. The numbers P of feature maps for the Pascal, MIT and SIFT databases were, respectively, 31, 128 and 64. The numbers n of prototypes in the DS layer for these three databases were set, respectively, to 75, 300 and 95.

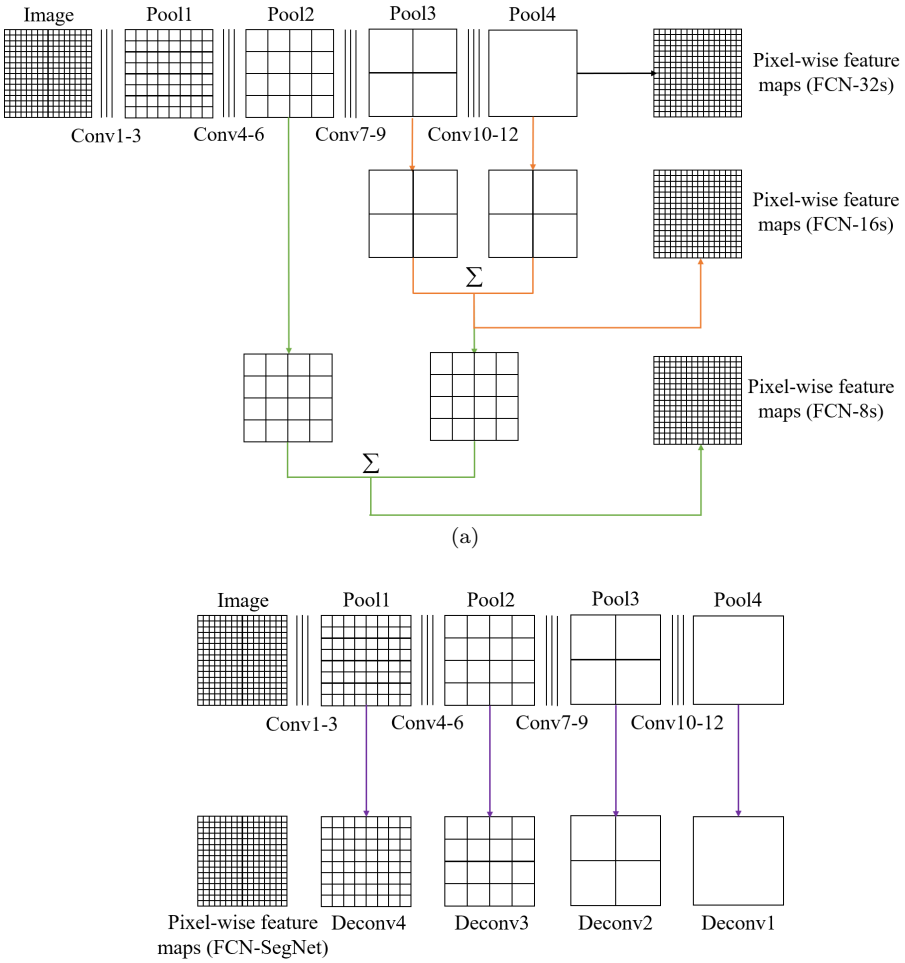

Fig. 4: Illustration of the encoder-decoder architectures used in this paper. Pooling layers are represented as grids that show relatively sparse information. Intermediate convolution layers are omitted. (a) The FCN-32s, FCN-16s, FCN-8s architectures are used to combine sparse and high-layer information with dense and low-layer information for upsampling. Black arrow: the deconvolutional layer in FCN-32s directly upsamples the outputs of Pool 4 to pixel-wise feature maps; orange arrows: the deconvolutional layer in FCN-16s combines outputs from Pool 3 and 4, lets the net predict finer details, while retaining high-level semantic information; green arrows: the deconvolutional layer in FCN-8s acquire additional feature maps from Pool 2 to provide further precision; (b) The FCN-SegNet architecture uses four deconvolutional layers to upsample the sparse feature maps from the end of the encoder part, as well as the feature maps from the corresponding pooling layers based on pooling indices (purple arrows).

Table 4: Performance evaluation of precise segmentation: (a) Pascal VOC 2011, (b) MIT-scene Parsing, and (c) SIFT Flow. P-FCN and E-FCN are, respectively, probabilistic and evidential FCNs. The rests of the notations, such as "-32s" and "-16s", stand for different encoder-decoder architectures. The results are in form of "mean value ± standard deviation". The best results for each encoder-decoder architecture are shown in bold.

(a)

	PU	UIoU
P-FCN-32s [25]	0.8912 ± 0.0019	0.5941 ± 0.0033
P-FCN-16s [25]	0.9001 ± 0.0015	0.6243 ± 0.0025
P-FCN-8s [25]	0.9033 ± 0.0017	0.6269 ± 0.0021
E-FCN-32s	**0.8973** ± 0.0021	**0.6128** ± 0.0024
E-FCN-16s	**0.9045** ± 0.0014	**0.6304** ± 0.0019
E-FCN-8s	**0.9074** ± 0.0015	**0.6337** ± 0.0020

(b)

	PU	UIoU
P-FCN-16s [25]	0.7009 ± 0.0030	0.289 ± 0.0051
P-FCN-8s [25]	0.7128 ± 0.0024	0.294 ± 0.0048
P-FCN-SegNet [1]	0.7153 ± 0.0023	0.305 ± 0.0042
E-FCN-16s	**0.7090** ± 0.0026	**0.292** ± 0.0048
E-FCN-8s	**0.7148** ± 0.0025	**0.296** ± 0.0046
E-FCN-SegNet	**0.7167** ± 0.0026	**0.330** ± 0.0043

(c)

	PU	UIoU
P-FCN-16s [25]	0.8489 ± 0.0034	0.3922 ± 0.0047
P-FCN-8s [25]	0.8525 ± 0.0032	0.3948 ± 0.0042
P-FCN-DilatedVGG [3]	0.8643 ± 0.0036	0.4168 ± 0.0043
E-FCN-16s	**0.8521** ± 0.0030	**0.3937**± 0.0042
E-FCN-8s	**0.8528** ± 0.0031	**0.3961** ± 0.0040
E-FCN-DilatedVGG	**0.8649** ± 0.0035	**0.4182** ± 0.0038

The DS and utility layers slightly improve the accuracy of precise assignments performed by FCN models, even though the performance of FCN models on precise segmentation mainly depends on the encoder-decoder architecture. Table 4a presents the results of PU and UIoU for the Pascal VOC database. E-FCNs achieved higher PU and UIoU than P-FCNs with the same encoder-decoder architecture, which shows the E-FCNs outperform the P-FCNs for precise segmentation. Similar improvements can also be found in the MIT-scene Parsing and SIFT Flow databases as shown, respectively, in Tables 4b and 4c.

The use of DS and utility layers also makes the FCN models better calibrated. Figure 5 presents a visual calibration representation of the FCN-8s models in the Pascal VOC database. The top row shows the pixel distribution of prediction confidence (21b) as histograms. The average confidence of the E-FCN-8s model closely matches its average pixel utility, while the average confidence of the P-FCN-8s model is substantially higher than its average pixel utility. This is further illustrated in the pixel utility diagrams (bottom row of Figure 5), which show pixel utility as a function of confidence. The E-FCN-8s model is well calibrated

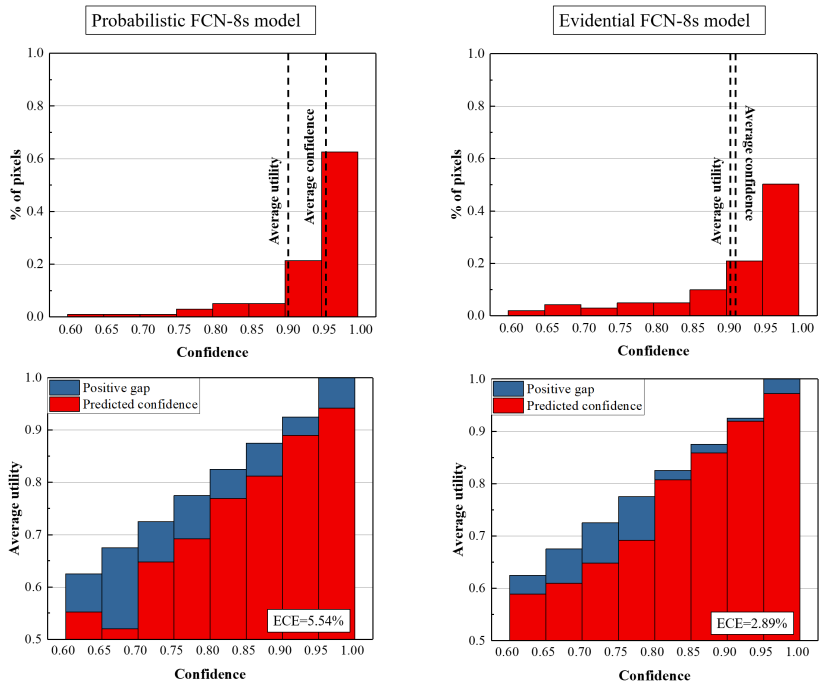

Fig. 5: Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-8s (left) and E-FCN-8s (right) on the Pascal VOC database.

since its confidence in each bin approximates the expected average utility, whereas the predicted utility of the P-FCN-8s model does not match its confidence. As a consequence, the E-FCN-8s model achieves a smaller ECE than the probabilistic one. The effect of the DS and utility layers on the calibration can also be found in the FCN-SegNet and FCN-DialtedVGG models on the MIT-scene Parsing and SIFT Flow databases as shown, respectively, in Figures 6 and 7.

4.3 Imprecise segmentation results

In imprecise segmentation, each pixel of an image is assigned to a non-empty subset A of Ω; the set of acts is $\mathcal{F} = \{f_A, A \in 2^\Omega \backslash \emptyset\}$, or a subset thereof. Here we only considered acts f_A such that A is a singleton, Ω or one of the soft labels in the training set. For performance evaluation, we used the metrics and the three databases described in Section 4.1. For each database, the segmentation masks with and without soft labels were used to train different FCN models. The same encoder-decoder architectures used for precise segmentation in Section 4.2 were combined with the DS and utility layers.

Figure 8 displays the test results according to PU and UIoU for imprecise segmentation of the Pascal VOC database. For a wide range of imprecision tolerance degree γ, the E-FCN models reach higher PU and UIoU values than those obtained by the P-FCN models; this is due to the fact that the E-FCN models

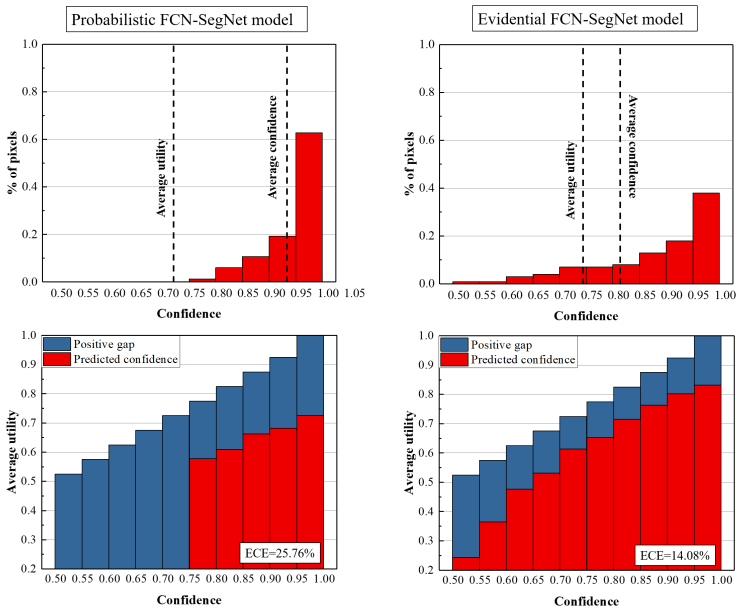

Fig. 6: Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-SegNet (left) and E-FCN-SegNet (right) on the MIT-scene Parsing database.

tend to assign ambiguous pixels to multi-class sets, instead of making precise decisions. Such imprecise assignments avoid pixel misclassification in case of high uncertainty, especially when feature vectors from an encoder-decoder architecture do not contain sufficient information to predict a precise class, and multiple classes have similar probabilities. Figure 9 shows the pixel confidence distributions for the FCN models with $\gamma = 0.8$. We can see that the average confidences of the E-FCN models are smaller than those of the P-FCN models. This observation suggests that the E-FCN models make cautious decisions for ambiguous pixels by assigning them to multi-class sets, rather than classifying them arbitrarily into a single class. The E-FCN models are thus better calibrated than those based on P-FCN, which can be over-confident. Similar results are observed with the MIT-scene Parsing (Figures 10-11) and SIFT Flow (Figures 12-13) databases. We can thus conclude the DS and utility layers improve the performance of the FCN models in imprecise segmentation tasks by allowing the assignment of some ambiguous pixels to multi-class sets.

In Figures 8, 10 and 12, we can see that the value of UIoU first increases and then decreases when γ increases from 0.5 to 1. To explain this behavior, Figure 14 shows some segmentation examples generated by the E-FCN-8s model trained on the Pascal VOC database with soft labels. The first and second columns of Figure 14 contain, respectively, the original images and their precise segmentation predicted masks, while the third to sixth columns show the imprecise segmentation results for values of γ ranging from 0.6 to 0.9. When γ increases from 0.5 to 0.8, the majority of the green masks (the areas whose pixels are assigned to multi-class

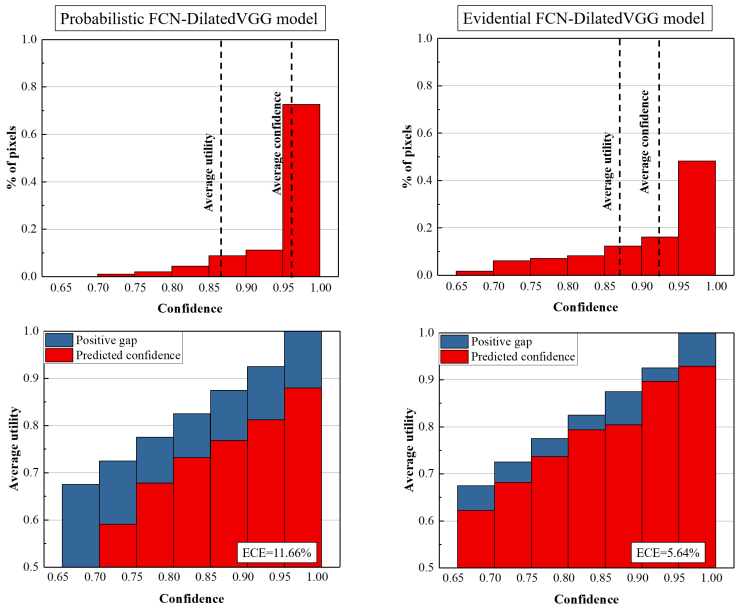

Fig. 7: Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-DilatedVGG (left) and E-FCN-DilatedVGG (right) on the SIFT Flow database.

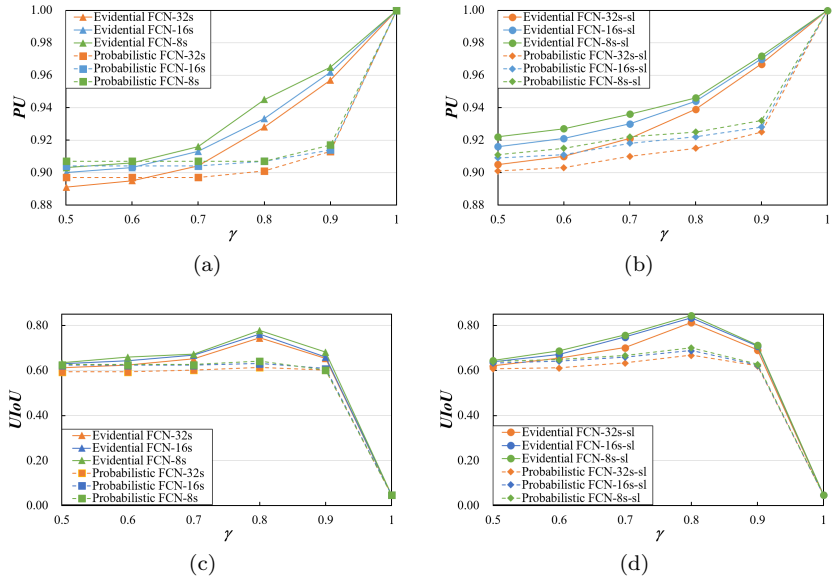

Fig. 8: Testing PU and UIoU vs. γ on the Pascal VOC database. The first and second columns are the models trained with/without soft labels, respectively.

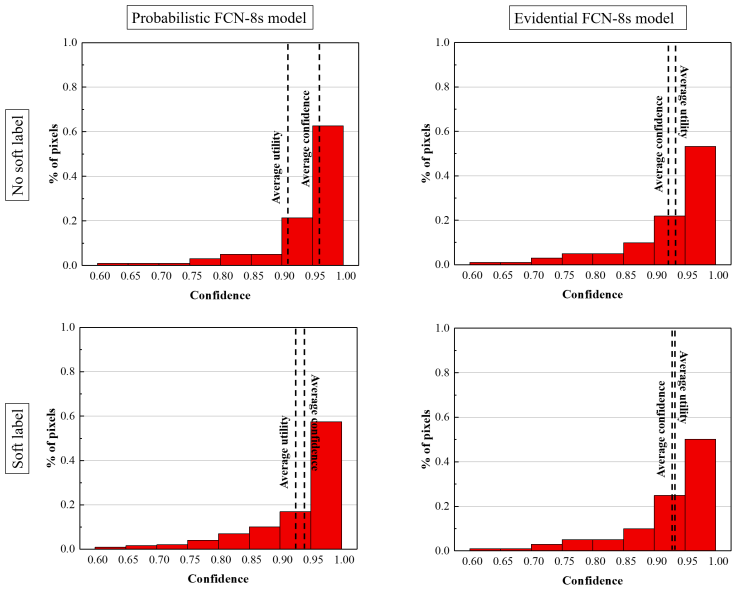

Fig. 9: Pixel confidence distributions for the P-FCN-8s (left) and E-FCN-8s (right) models on the Pascal VOC 2011 database without (top)/with (bottom) soft labels.

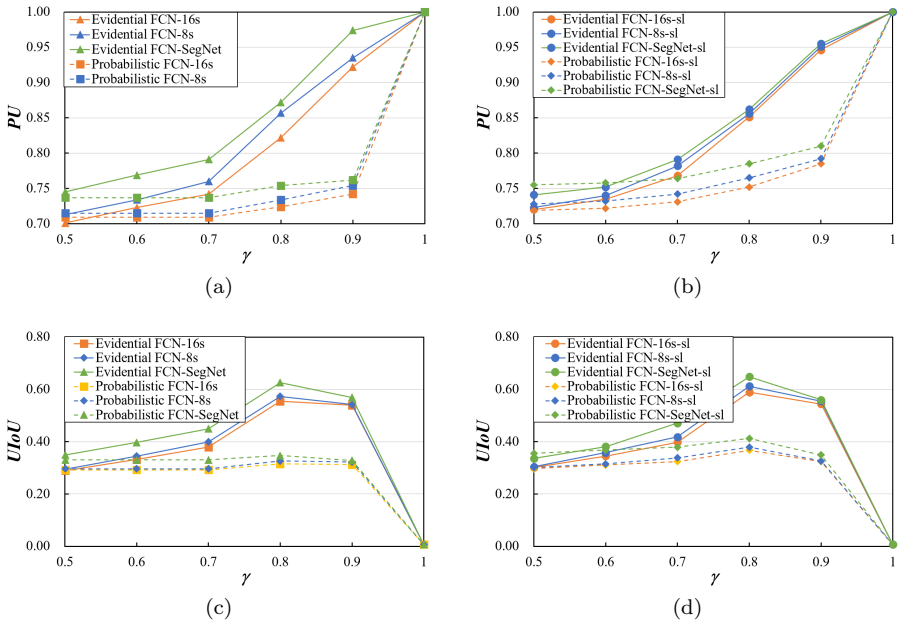

Fig. 10: Testing PU and UIoU vs. γ on the MIT-scene Parsing database. The first and second columns are the models trained with/without soft labels, respectively.

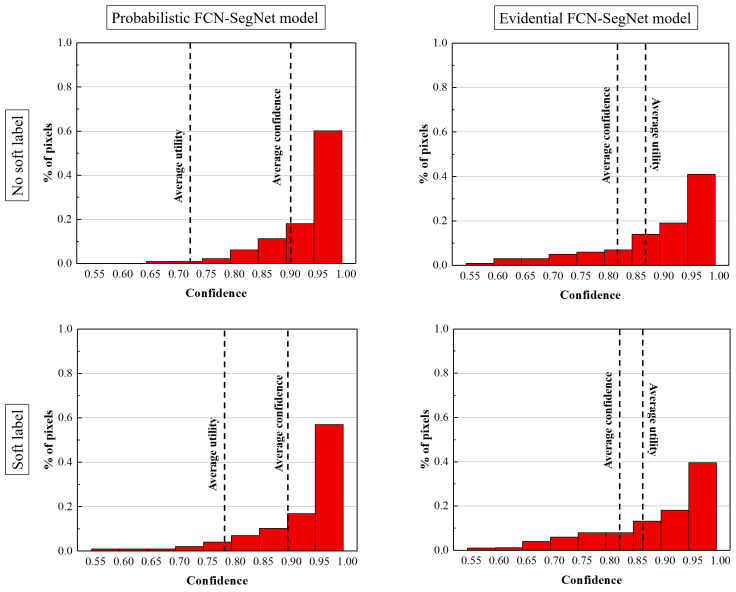

Fig. 11: Pixel rate histograms for the P-FCN-SegNet (left) and E-FCN-SegNet (right) models on the MIT-scene Parsing database without (top)/with (bottom) soft labels.

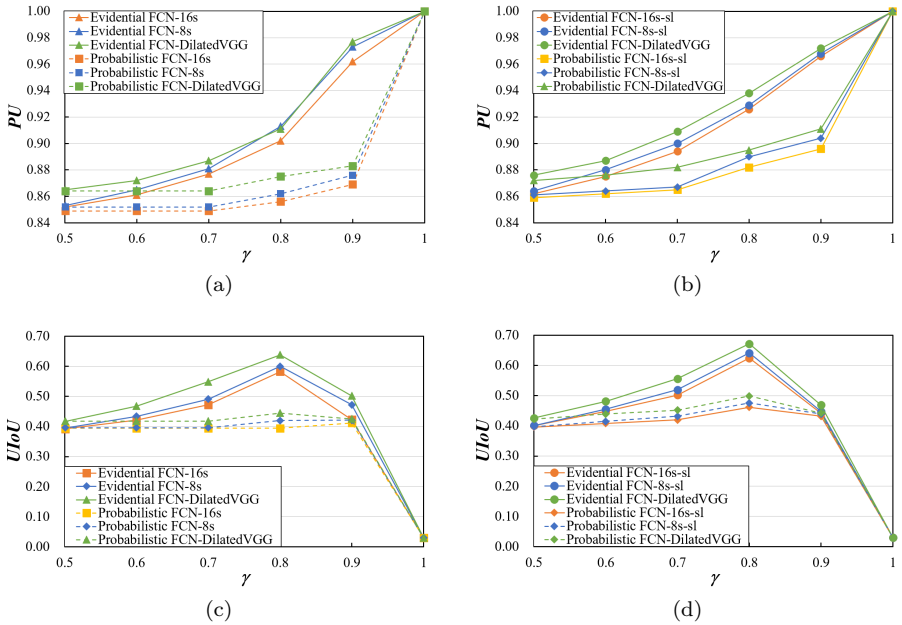

Fig. 12: Testing PU and UIoU vs. γ on the SIFT Flow database. The first and second columns are the models trained with/without soft labels, respectively.

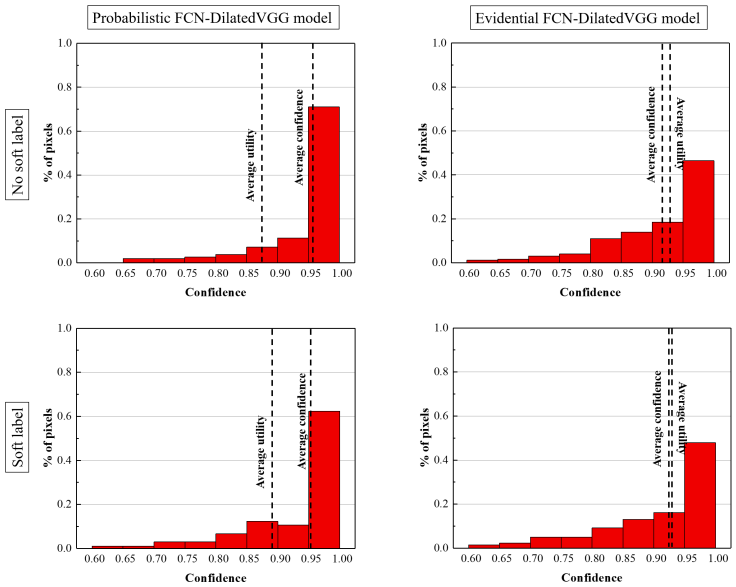

Fig. 13: Pixel rate histograms for the P-FCN-DilatedVGG (left) and E-FCN-DilatedVGG (right) models on the SIFT Flow database without (top)/with (bottom) soft labels.

sets) tends to cover the red masks (the areas whose pixels are incorrectly classified in the precise segmentation). This observation can be explained by the fact that, in Eq. (19), the increase in the utility of the intersection between predicted and labeled areas is larger than the increase in the union between the two areas. As a result, UIoU increases when γ increases from 0.5 to 0.8. However, when γ increases from 0.8 to 1.0, the majority of the green masks cover the areas predicted correctly in the precise segmentation, which causes the increase in the utility of intersection to be smaller than the increase in the union areas. This phenomenon leads to the decrease of UIoU when γ is larger than 0.8.

The use of soft labels improves the performance of the FCN models for imprecision segmentation tasks. As shown in Figure 8, the FCN models trained on the Pascal VOC database with soft labels have larger testing PU and UIoU than the ones without soft labels, which demonstrates the accuracy improvement using soft labels. Additionally, the use of soft labels can also improve the calibration of the FCN models. Figure 15 shows that the ECEs and bin gaps in the E-FCN and P-FCN models are smaller when using the learning set with soft labels. These results demonstrate the feasibility of processing pixels with ambiguous information using soft labels when training FCN models. An improvement of accuracy and calibration due to learning from soft labels is also observed with the MIT-scene Parsing and SIFT Flow databases, as shown, respectively, in Figures 16 and 17. Therefore, we can conclude that the use of soft labels improves the accuracy and calibration of FCN models.

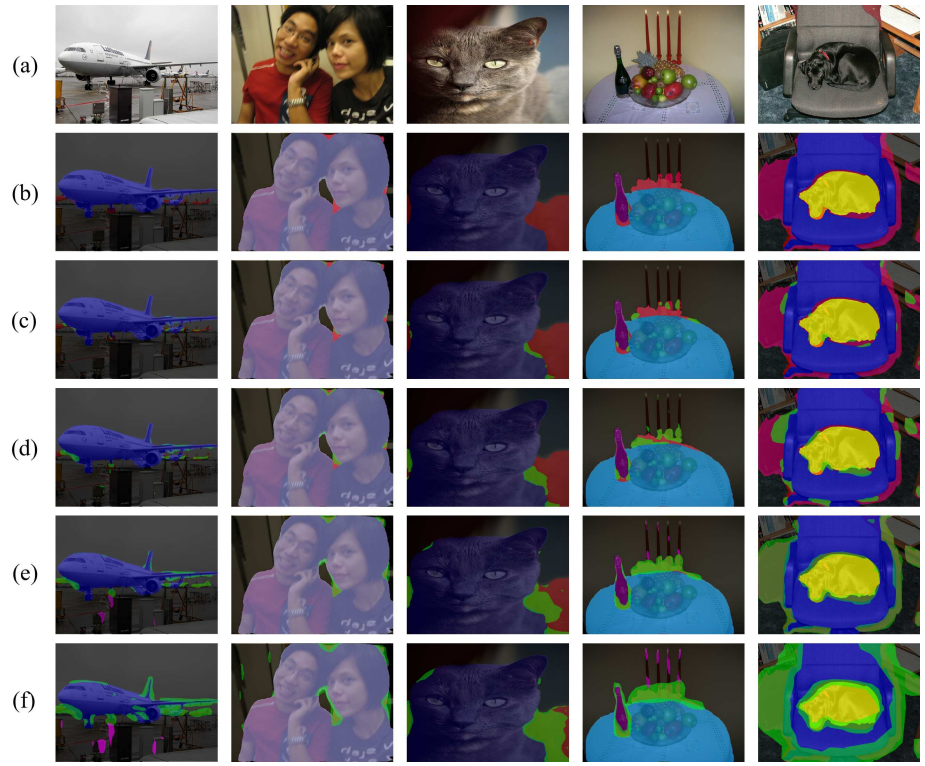

Fig. 14: Segmentation examples from the Pascal VOC 2011 database: (a) Original image, (b) Precise segmentation, (c) Imprecise segmentation with $\gamma = 0.6$, (d) Imprecise segmentation with $\gamma = 0.7$, (e) Imprecise segmentation with $\gamma = 0.8$, and (f) Imprecise segmentation with $\gamma = 0.9$. Red masks are pixels incorrectly classified in the precise segmentation; green masks are pixels assigned to multi-class sets except set Ω; pink masks are pixels assigned to set Ω; other masks are pixels assigned to correct single-class sets.

4.4 Novelty detection results

For novelty detection, a pixel is considered as an outlier or an ambiguous sample if it is assigned to set Ω. Figures 18, 19 and 20 show the results of novelty detection using the E-FCN and P-FCN models when the learning set is extracted, respectively, from the Pascal VOC, MIT-scene Parsing and SIFT Flow databases, and the test set is composed of images from the other two databases. In each testing set composed of two databases, only the pixels whose class is not represented in the corresponding learning set are reported in Figures 18-20. The E-FCN models assign outliers and some known-class pixels to set Ω for values of γ between 0.7 and 0.9, while the P-FCN models do not. This observation shows that the E-FCN models are more efficient than the probabilistic ones for rejecting outliers together with ambiguous samples. The proposed architecture thus has the potential to perform novelty detection once given a reasonable value of tolerance to imprecision.

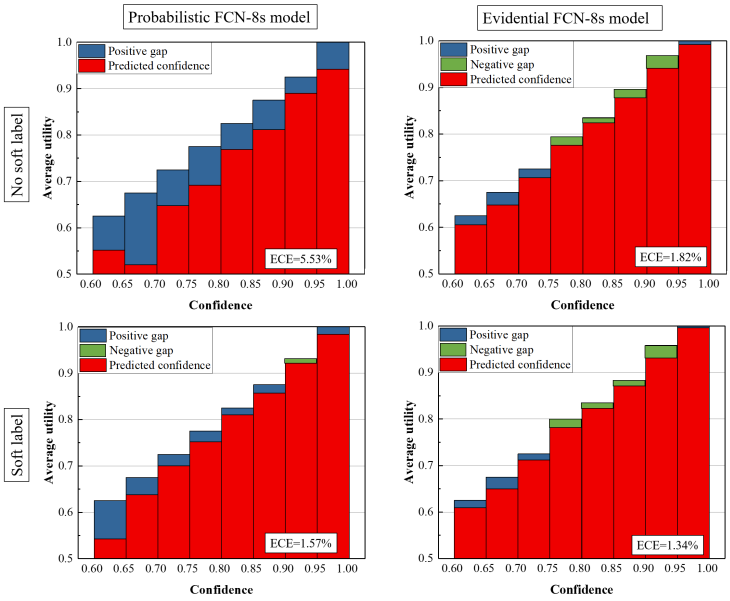

Fig. 15: Average utility histograms for P-FCN-8s (left) and E-FCN-8s (right) with $\gamma = 0.8$ on the Pascal VOC 2011 database without (top)/with (bottom) soft labels.

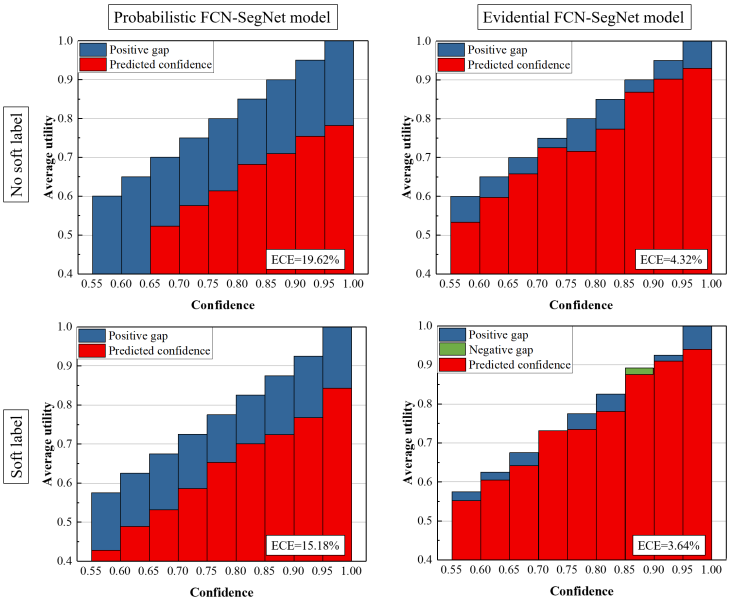

Fig. 16: Average utility histograms for P-FCN-SegNet (left) and E-FCN-SegNet (right) with $\gamma = 0.8$ on the MIT-scene Parsing database without (top)/with (bottom) soft labels.

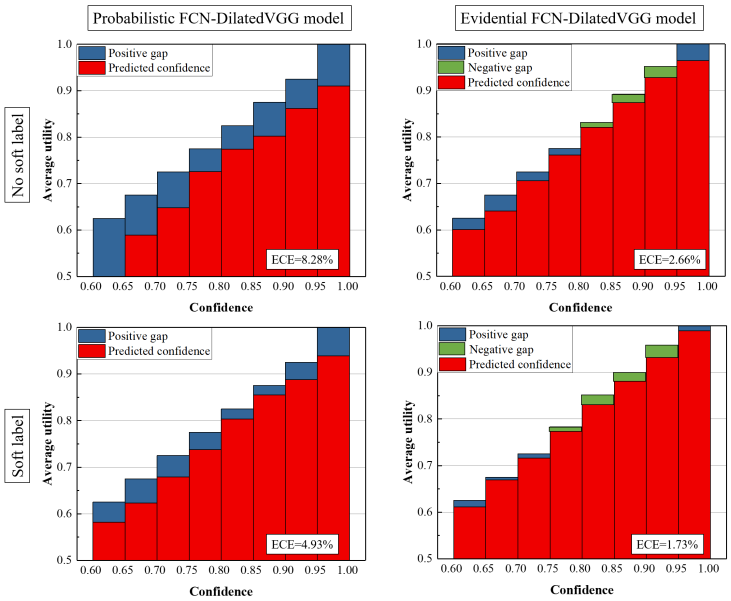

Fig. 17: Average utility histograms for P-FCN-DilatedVGG (left) and E-FCN-DilatedVGG (right) with $\gamma = 0.8$ on the SIFT Flow database without (top)/with (bottom) soft labels.

However, none of the FCN models performs well when γ is less than 0.7 since these models favor precise decisions.

The E-FCN models tend to reject unknown objects whose features are very different from those of the known objects in the learning set. For example, Figure 21 shows images from the MIT-scene Parsing database in which pixels representing 'bag', 'street light' and 'ball' objects are rejected by an E-FCN-8s model trained using the Pascal VOC database, which does not contain these objects. As shown in Table 5, 75.2% of the pixels representing a ball in the MIT-scene Parsing and and SIFT Flow databases are assigned to Ω, while 16.1% are assigned to a set of classes containing "bottle". For the "bag" and "street light" classes, these numbers are, respectively, 68.4%/21.8% and 77.3%/16.3%. Some unknown objects are not so easily rejected because of their similarity with known objects. For instance, 84.7% of the pixels representing a seat and 81.7% of pixels representing a bench are assigned to a set of classes containing "chair", and 88% of "wall" pixels are assigned to a set of classes containing "background".

We can also observe that the FCN models trained using a leaning set with soft labels reject more outliers than those trained without soft labels, as shown in Figures 18, 19 and 20. This is because the use of soft labels makes the FCN models more cautious and better calibrated, as discussed in Section 4.3. More precisely, for ambiguous pixels or outliers, the output mass functions of the FCN models trained with soft labels are more uniform than those computed by FCN models trained without soft labels. As a result, ambiguous pixels and outliers are more

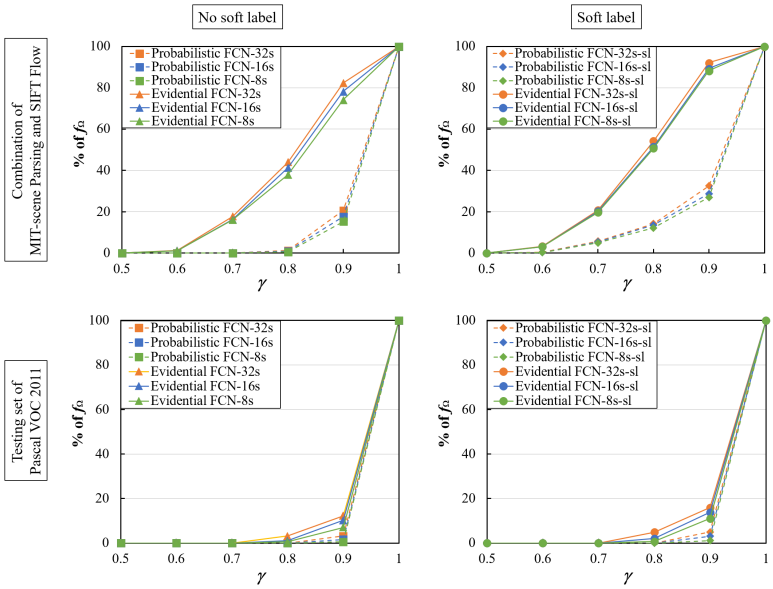

Fig. 18: Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of MIT-scene Parsing and SIFT Flow databases (top) and the testing set from the Pascal VOC 2011 database (bottom) when the learning set is from the Pascal VOC database without (left)/with (right) soft labels.

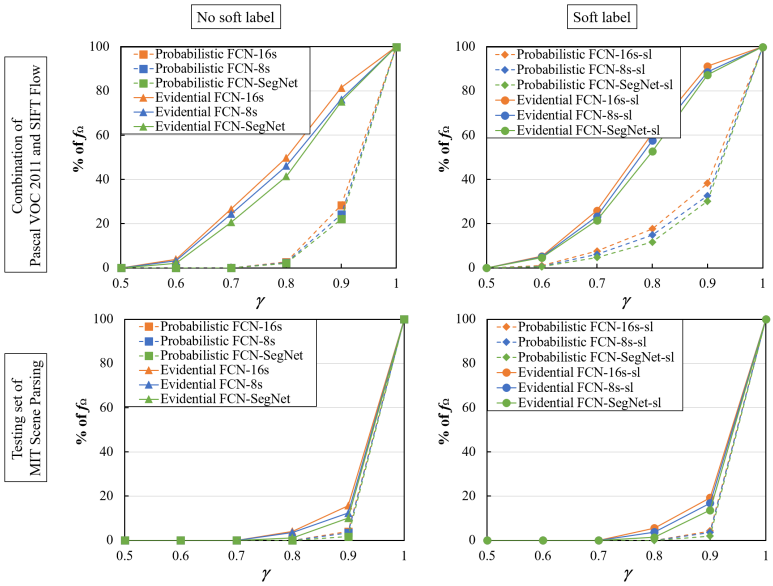

Fig. 19: Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of Pascal VOC 2011 and SIFT Flow (top) and the testing set of the MIT-scene Parsing database (bottom) when the learning set is from the Pascal VOC database without (left)/with (right) soft labels.

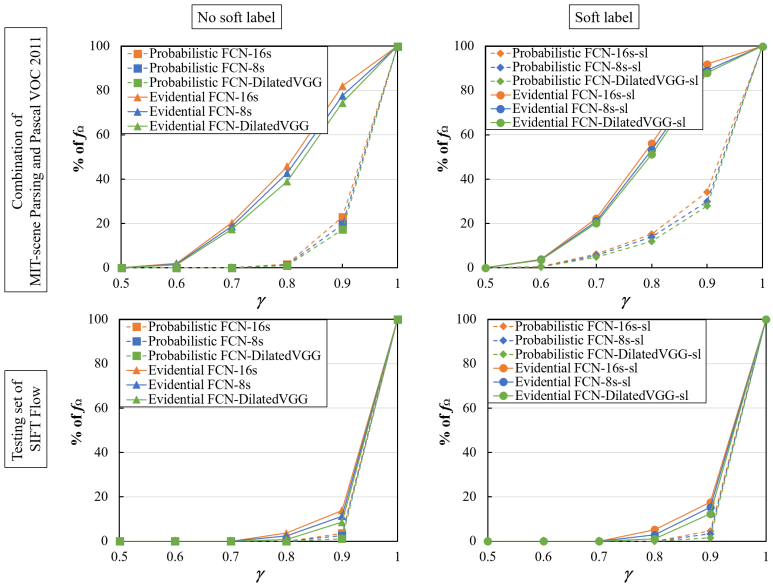

Fig. 20: Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of Pascal VOC 2011 and MIT-scene Parsing (top) and the testing set of the SIFT Flow database (bottom) when the learning set is from the Pascal VOC database without (left)/with (right) soft labels.

Table 5: Percentage of pixels from some unknown classes in the MIT-scene Parsing and SIFT Flow databases classified by an E-FCN-8s model trained on the Pascal VOC database into some sets of classes. The model was trained with soft labels and $\gamma = 0.8$. For instance, 68.4% of the pixels representing a bag were rejected (i.e., assigned to Ω), and 84.7% of pixels representing a seat were assigned to a set of classes containing the class "chair".

		True class						
		bag	street light	ball	seat	bench	bed	wall
Assigned set	Ω	68.4	77.3	75.2	7.8	4.7	15.9	4.9
	{bottle, ...}	21.8	16.3	16.1	48.5	39.7	30.3	0.2
	{chair, ...}	11.3	9.2	8.5	84.7	81.7	58.6	0.3
	{background, ...}	15.2	13.7	11.5	58.7	48.6	46.9	88.0
	Others	4.2	2.4	1.5	2.7	3.5	5.2	3.7

easily assigned to set Ω. We can thus conclude that soft labels have the potential to enhance novelty detection performance.

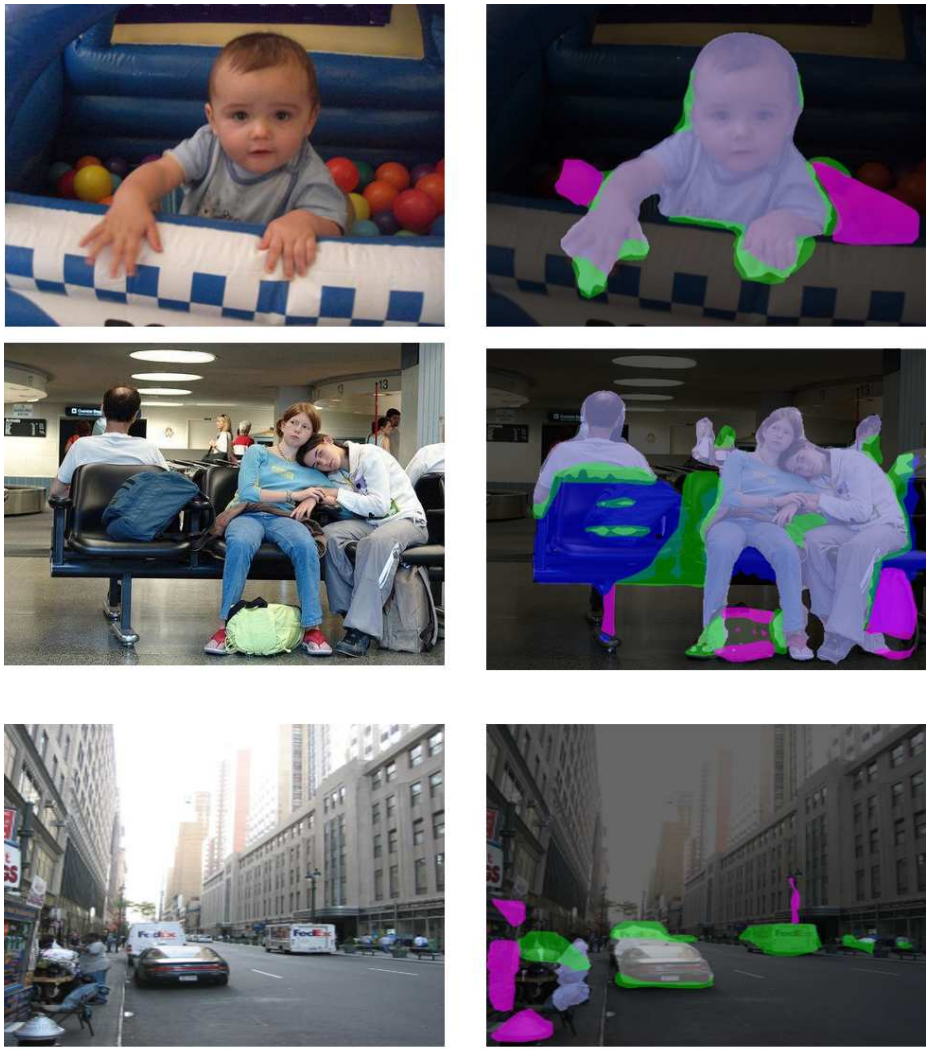

Fig. 21: Examples of novelty detention from the MIT-scene Parsing database and their segmentation masks given by the E-FCN-8s model trained using the Pascal VOC database with soft labels when γ equals 0.8. Red masks are pixels incorrectly assigned in the precise segmentation; green masks are pixels assigned to multi-class sets except set Ω; pink masks are pixels assigned to set Ω; other masks are pixels assigned to correct single-class sets.

5 Conclusions

In this paper, we have presented a new approach based on the combination of DS theory and FCN for image semantic segmentation. In the proposed model, called evidential fully convolutional network (E-FCN), an encoder-decoder architecture first extracts pixel-wise feature maps from an input image. A Dempster-Shafer

layer then computes mass functions at each pixel location based on distances to prototypes. Finally, a utility layer performs semantic segmentation based on pixel-wise mass functions. The proposed model can be trained using a learning set with soft labels in an end-to-end way.

The main finding of this study is that the proposed combination of FCNs and ENNs makes it possible to improve accuracy and calibration of FCN models by assigning ambiguous pixels to multi-class sets, while maintaining the good performance of FCNs in precise segmentation tasks. The E-FCN model is able to select a set of classes when the object representation does not allow us to select a single class unambiguously, which easily leads to incorrect decision-making in probabilistic FCNs. This result provides a new direction to improve the performance of FCN models for semantic segmentation. The learning strategy using soft labels further improves the accuracy and calibration of the FCN models. Additionally, the proposed approach makes it possible to reject outliers together with ambiguous pixels when the tolerance to imprecision is between 0.7 and 0.9.

Future work will focus on two main aspects. First, we will investigate multi-FCN-model information fusion for semantic segmentation based on the definition of soft labels, using an approach similar to that introduced in [36]. Other advanced evidential classifiers, such as the contextual-discounting evidential K-nearest neighbor [14] will also be considered to improve the performance of the proposed neural network architecture.

Acknowledgements This research was supported by a scholarship from the China Scholarship Council and by the Labex MS2T (reference ANR-11-IDEX-0004-02).

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(12), 2481–2495 (2017)
2. Biggio, B., Nelson, B., Laskov, P.: Support vector machines under adversarial label noise. In: Asian conference on machine learning, pp. 97–112 (2011)
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(4), 834–848 (2017)
4. Chen, X.l., Wang, P.h., Hao, Y.s., Zhao, M.: Evidential KNN-based condition monitoring and early warning method with applications in power plant. Neurocomputing **315**, 18–32 (2018)
5. Côme, E., Oukhellou, L., Denœux, T., Aknin, P.: Learning from partially supervised data using mixture models and belief functions. Pattern Recognition **42**(3), 334–348 (2009)

6. 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: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3213–3223 (2016)
7. Dempster, A.P.: Upper and lower probabilities induced by a multivalued mapping. Annals of Mathematical Statistics **38**, 325–339 (1967)
8. Denœux, T.: Analysis of evidence-theoretic decision rules for pattern classification. Pattern Recognition **30**(7), 1095–1107 (1997)
9. Denœux, T.: A neural network classifier based on Dempster-Shafer theory. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **30**(2), 131–150 (2000)
10. Denoeux, T.: 40 years of Dempster-Shafer theory. International Journal of Approximate Reasoning **79**, 1–6 (2016)
11. Denoeux, T.: Decision-making with belief functions: a review. International Journal of Approximate Reasoning **109**, 87–110 (2019)
12. Denœux, T.: Logistic regression, neural networks and Dempster-Shafer theory: A new perspective. Knowledge-Based Systems **176**, 54–67 (2019)
13. Denœux, T., Dubois, D., Prade, H.: Representations of uncertainty in artificial intelligence: Beyond probability and possibility. In: A Guided Tour of Artificial Intelligence Research, vol. 1, chap. 4, pp. 119–150. Springer Verlag (2020)
14. Denœux, T., Kanjanatarakul, O., Sriboonchitta, S.: A new evidential k-nearest neighbor rule based on contextual discounting with partially supervised learning. International Journal of Approximate Reasoning **113**, 287–302 (2019)
15. Denœux, T., Smets, P.: Classification using belief functions: relationship between case-based and model-based approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **36**(6), 1395–1406 (2006)
16. Ess, A., Müller, T., Grabner, H., Van Gool, L.: Segmentation-based urban traffic scene understanding. In: BMVC, vol. 1, p. 2. Citeseer (2009)
17. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International journal of computer vision **111**(1), 98–136 (2015)
18. Forouzanfar, M., Forghani, N., Teshnehlab, M.: Parameter optimization of improved fuzzy c-means clustering algorithm for brain mr image segmentation. Engineering Applications of Artificial Intelligence **23**(2), 160–168 (2010)
19. Guettari, N., Capelle-Laizé, A.S., Carré, P.: Blind image steganalysis based on evidential K-Nearest Neighbors. In: Proceedings of the 2016 IEEE International Conference on Image Processing, pp. 2742–2746. Phoenix, USA (2016)
20. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. arXiv preprint arXiv:1706.04599 (2017)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
22. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: Advances in neural information processing systems, pp. 109–117 (2011)
23. Kumar, N., Berg, A.C., Belhumeur, P.N., Nayar, S.K.: Attribute and simile classifiers for face verification. In: Proceedings of the 12th International Conference on Computer Vision, pp. 365–372. IEEE, Kyoto, Japan (2009)

669 24. Lian, C., Ruan, S., Denœux, T.: An evidential classifier based on feature selec-
670 tion and two-step classification strategy. Pattern Recognition **48**, 2318–2327
671 (2015)
672 25. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic
673 segmentation. In: Proceedings of the IEEE conference on Computer Vision
674 and Pattern Recognition, pp. 3431–3440 (2015)
675 26. Ma, L., Denœux, T.: Partial classification in the belief function framework.
676 Knowledge-Based Systems **214**, 106742 (2021)
677 27. Natarajan, N., Dhillon, I.S., Ravikumar, P.K., Tewari, A.: Learning with noisy
678 labels. In: Advances in neural information processing systems, pp. 1196–1204
679 (2013)
680 28. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic seg-
681 mentation. In: Proceedings of the IEEE International Conference on Computer
682 Vision, pp. 1520–1528 (2015)
683 29. O'Hagan, M.: Aggregating template or rule antecedents in real-time expert
684 systems with fuzzy set logic. In: Twenty-Second Asilomar Conference on Sig-
685 nals, Systems and Computers, vol. 2, pp. 681–689 (1988)
686 30. Shafer, G.: A mathematical theory of evidence. Princeton University Press,
687 Princeton (1976)
688 31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale
689 image recognition. arXiv preprint arXiv:1409.1556 (2014)
690 32. Smets, P.: Belief functions: the disjunctive rule of combination and the gen-
691 eralized Bayesian theorem. International Journal of approximate reasoning
692 **9**(1), 1–35 (1993)
693 33. Su, Z.G., Denoeux, T., Hao, Y.S., Zhao, M.: Evidential K-NN classification
694 with enhanced performance via optimizing a class of parametric conjunctive
695 t-rules. Knowledge-Based Systems **142**, 7–16 (2018)
696 34. Tighe, J., Lazebnik, S.: Superparsing: scalable nonparametric image parsing
697 with superpixels. In: European conference on computer vision, pp. 352–365.
698 Springer (2010)
699 35. Tong, Z., Xu, P., Denœux, T.: ConvNet and Dempster-Shafer theory for object
700 recognition. In: Processing of the 13th international conference on Scalable Un-
701 certainty Management, pp. 368–381. Springer International Publishing, Cham
702 (2019)
703 36. Xu, P., Davoine, F., Bordes, J.B., Zhao, H., Denœux, T.: Multimodal informa-
704 tion fusion for urban scene understanding. Machine Vision and Applications
705 **27**(3), 331–349 (2016)
706 37. Yager, R.R.: On ordered weighted averaging aggregation operators in multicri-
707 teria decision-making. IEEE Transactions on systems, Man, and Cybernetics
708 **18**(1), 183–190 (1988)
709 38. Yager, R.R., Liu, L.: Classic works of the Dempster-Shafer theory of belief
710 functions, vol. 219. Springer, Berlin, Heidelberg (2008)
711 39. Yoon, Y., Jeon, H.G., Yoo, D., Lee, J.Y., So Kweon, I.: Learning a deep con-
712 volutional network for light-field image super-resolution. In: Proceedings of
713 the IEEE international conference on computer vision workshops, pp. 24–32
714 (2015)
715 40. Yuan, B., Yue, X., Lv, Y., Denoeux, T.: Evidential deep neural networks for
716 uncertain data classification. In: International Conference on Knowledge Sci-
717 ence, Engineering and Management, pp. 427–437. Springer (2020)

41. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision, pp. 818–833. Springer (2014)
42. Zeiler, M.D., Taylor, G.W., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: 2011 International Conference on Computer Vision, pp. 2018–2025. IEEE (2011)
43. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. arXiv preprint arXiv:1608.05442 (2016)