



Université de technologie de Compiègne

HEUDIASYC, UMR CNRS 7253

Doctoral Thesis

*Submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy in Information Technologies and Systems*

Present by:

Zheng TONG

**Evidential deep neural network in the
framework of Dempster-Shafer theory**

JURY

Thierry DENŒUX	Professor, UTC	Supervisor
Frédéric PICHON	Professor, Artois University	Reviewer
Sylvie LE HEGARAT	Professor, Paris-Sud University	Reviewer
Philippe XU	Associate professor, UTC	Examiner
Véronique CHERFAOUI	Professor, UTC	Examiner
Emmanuel RAMASSO	Associate professor, ENSMM School	Examiner

March 15, 2022

Abstract

Deep neural networks (DNNs) have achieved remarkable success on many real-world applications (e.g., pattern recognition and semantic segmentation) but still face the problem of managing uncertainty. Dempster-Shafer theory (DST) provides a well-founded and elegant framework to represent and reason with uncertain information. In this thesis, we have proposed a new framework using DST and DNNs to solve the problems of uncertainty.

In the proposed framework, we first hybridize DST and DNNs by plugging a DST-based neural-network layer followed by a utility layer at the output of a convolutional neural network for set-valued classification. We also extend the idea to semantic segmentation by combining fully convolutional networks and DST. The proposed approach enhances the performance of DNN models by assigning ambiguous patterns with high uncertainty, as well as outliers, to multi-class sets. The learning strategy using soft labels further improves the performance of the DNNs by converting imprecise and unreliable label data into belief functions.

We have also proposed a modular fusion strategy using this proposed framework, in which a fusion module aggregates the belief-function outputs of evidential DNNs by Dempster's rule. We use this strategy to combine DNNs trained from heterogeneous datasets with different sets of classes while keeping at least as good performance as those of the individual networks on their respective datasets. Further, we apply the strategy to combine several shallow networks and achieve a similar performance of an advanced DNN for a complicated task.

Keywords: Dempster-Shafer theory, belief function, deep neural network, deep learning, pattern classification, semantic segmentation

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisors Prof. Thierry Dencœux and Ass. Prof. Philippe Xu for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

Besides my supervisors, I would like to thank the members of the jury Prof. Frédéric Pichon, Prof. Sylvie Le Hégarat, Prof. Véronique Cherfaoui, and Ass. Prof. Emmanuel Ramsso, for their insightful comments and encouragement, but also for the hard questions which incited me to widen my research from various perspectives.

I thank my colleagues from Heudiasyc in Compiègne with whom I spent a huge amount of time during office hours and beyond. Thanks to all of you for the moments we shared.

I wish to thank the unconditional support of my parents and girl friend Miss Yao QIU despite the long distance. Without your support, nothing would have been possible.

I would like to gratefully acknowledge the support provided for this work under the framework of the Labex MS2T. Last but not least, I would also like to thank the Chinese Scholarship Council for supporting my study in France.

List of Publications

International journals

- Z. Tong, Ph. Xu, T. Denœux. An evidential classifier based on Dempster-Shafer theory and deep learning. *Neurocomputing*, August 2021, 450, 275–293.
- Z. Tong, Ph. Xu, T. Denœux. Evidential fully convolutional network for semantic segmentation. *Applied Intelligence*, April 2021, 51, 6376–6399.

International conferences

- Z. Tong, Ph. Xu, T. Denœux. ConvNet and Dempster-Shafer Theory for Object Recognition. In: *International Conference on Scalable Uncertainty Management* (SUM 2019) , pp. 368-381. Springer, Cham, France, 2019.
- Z. Tong, Ph. Xu, T. Denœux. Fusion of evidential CNN classifiers for image classification. In: *International Conference on International Conference on Belief Functions* (BELIEF 2021) . Springer, Shanghai, China, 2021. (Best paper award)

Available codes

Evidential deep-learning classifier: The python code using open-source software library TensorFlow can be downloaded at <https://github.com/tongzheng1992/E-CNN-classifier>.

Evidential fully convolutional network: The python code using open-source software library TensorFlow can be downloaded at <https://github.com/tongzheng1992/E-FCN>.

Contents

Abstract	iii
Acknowledgements	v
List of Publications	vii
List of Figures	xv
List of Tables	xviii
Acronyms & notations	xix
Introduction	1
I Background	5
1 Dempster-Shafer theory	7
1.1 Information representation	7
1.1.1 Mass function	7
1.1.2 Discounting	8
1.2 Operations of belief functions	9
1.2.1 Dempster's rule	9
1.2.2 Change of frame of discernment	9
1.3 Decision-making with belief functions	11
1.3.1 Precise decision with belief functions	11
1.3.2 Imprecise decision with belief functions	12
1.4 Evidential neural network based on Dempster-Shafer theory	15
1.5 Conclusion	17
2 Deep neural networks	19
2.1 Convolution neural network	20
2.1.1 Convolution operation and its motivation	20
2.1.2 Pooling operation	22
2.1.3 Data types	23
2.1.4 Efficient modern variants of convolutional neural networks	23
2.2 Fully convolutional networks	28

2.2.1	Overall architecture of fully convolutional network	28
2.2.2	Upsampling operations	30
2.2.3	Variants of fully convolution networks	32
2.3	Conclusion	36
II Evidential deep neural networks		37
3	Evidential convolutional neural network classifier	39
3.1	Evidential CNN classifier	39
3.1.1	Network architecture	40
3.1.2	Learning	41
3.1.3	Act selection	43
3.2	Experimental evaluation	44
3.2.1	Evaluation metrics	45
3.2.2	Image classification experiment	45
3.2.3	Signal classification experiment	54
3.2.4	Semantic-relationship classification experiment	56
3.3	Conclusion	64
4	Evidential fully convolutional network	65
4.1	Evidential FCN model	65
4.1.1	Network architecture	66
4.1.2	Learning with soft labels	67
4.2	Experimental evaluation	68
4.2.1	Datasets	68
4.2.2	Evaluation metrics	70
4.2.3	Precise segmentation results	72
4.2.4	Imprecise segmentation results	76
4.2.5	Novelty detection results	81
4.3	Conclusion	86
5	Evidential fusion of heterogeneous deep neural networks	87
5.1	Introduction	88
5.2	Fusion approach	89
5.2.1	Evidential fusion approach	89
5.2.2	Learning with soft labels	92
5.3	Experiments on multi-model fusion	93
5.3.1	Image-classification experiment #1	93
5.3.2	Image-classification experiment #2	98
5.3.3	Semantic-segmentation experiment #1	99
5.3.4	Semantic-segmentation experiment #2	103

5.4	Experiments on training shallow networks for complex tasks	105
5.4.1	Experiment on the Tiny ImageNet dataset	106
5.4.2	Experiment on the Cityscapes dataset	109
5.5	Conclusion	109
	Conclusions and perspectives	111
A	Appendix: gradient calculation in evidential neural network	115
	Bibliography	119

List of Figures

1.1	Evidential neural network classifier [20].	16
2.1	An example of the convolution operation.	21
2.2	Examples of pooling operations.	23
2.3	Comparison of traditional convolution layer and NIN convolutional layer [74].	24
2.4	Depth and width comparison of teacher and student CNN stages in knowledge distillation.	26
2.5	Example of skip connection	26
2.6	Architecture of a ViT [27].	28
2.7	Transforming fully connected layers into convolution layers to output a heatmap [77].	29
2.8	An illustration of the encoder-decoder architecture.	30
2.9	Upsampling examples.	31
2.10	An instance of transposed convolution operation.	32
2.11	Illustration of the FCN-32s, FCN-16s, and FCN-8s architectures	33
2.12	Illustration of a U-net architecture [107].	34
2.13	Illustration of the SegNet architecture	35
2.14	Illustration of a conditional random field.	35
3.1	Architecture of an evidential CNN classifier.	40
3.2	Architecture of the utility layer.	41
3.3	An example of act selection.	44
3.4	Rejection-error curves on the CIFAR-10 testing set	48
3.5	Illustration of the 5-fold cross validation for determining λ_0 with	48
3.6	Average utility vs. ν for the evidential CNN classifiers on the CIFAR-10 dataset.	50
3.7	Average utility (a) and average cardinality (b) vs. γ for the evidential and probabilistic CNN classifiers on the CIFAR-10 dataset.	52
3.8	Dendrograms for the CIFAR-10 dataset.	53
3.9	Rate of f_Ω vs. γ for novelty detection in the image-classification experiment.	55
3.10	Rejection-error curves on the UrbanSound 8K testing set	57

3.11	Average utility vs. ν for the evidential CNN classifiers on the UrbanSound 8K dataset.	58
3.12	Average utility (a) and average cardinality (b) vs. γ for the proposed classifiers and the probabilistic CNN classifiers on the UrbanSound 8K dataset.	58
3.13	Rate of f_{Ω} vs. γ for novelty detection in the signal-classification experiment.	59
3.14	Dendrograms for the UrbanSound 8K dataset.	59
3.15	Rejection-error curves on the SemEval-2010 Task 8 testing set	61
3.16	Curves in ν -utility for the evidential CNN classifiers on the SemEval-2010 Task 8 dataset.	62
3.17	Average utility (a) and average cardinality (b) vs. γ for the proposed classifiers and the probabilistic CNN classifiers on the SemEval-2010 Task 8 dataset.	62
3.18	Rate of f_{Ω} vs. γ for novelty detection in the semantic-relationship-classification experiment.	63
3.19	Dendrograms for the SemEval-2010 Task 8 dataset.	63
4.1	Architecture of an evidential fully convolutional network.	66
4.2	Segmentation masks with soft labels.	70
4.3	Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-8s (left) and E-FCN-8s (right) on the Pascal VOC dataset.	74
4.4	Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-SegNet (left) and E-FCN-SegNet (right) on the MIT-scene Parsing dataset.	75
4.5	Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-CRF (left) and E-FCN-CRF (right) on the SIFT Flow dataset.	75
4.6	Testing PU and UIoU vs. γ on the Pascal VOC dataset.	77
4.7	Pixel confidence distributions for the P-FCN-8s (left) and E-FCN-8s (right) models on the Pascal VOC dataset without (top)/with (bottom) soft labels.	77
4.8	Testing PU and UIoU vs. γ on the MIT-scene Parsing dataset.	78
4.9	Pixel rate histograms for the P-FCN-SegNet (left) and E-FCN-SegNet (right) models on the MIT-scene Parsing dataset without (top)/with (bottom) soft labels.	78
4.10	Testing PU and UIoU vs. γ on the SIFT Flow dataset.	79
4.11	Pixel rate histograms for the P-FCN-CRF (left) and E-FCN-CRF (right) models on the SIFT Flow dataset without (top)/with (bottom) soft labels.	79

4.12	Segmentation examples from the Pascal VOC dataset.	80
4.13	Average utility histograms for P-FCN-8s (left) and E-FCN-8s (right) with $\gamma = 0.8$ on the Pascal VOC dataset without (top)/with (bottom) soft labels.	81
4.14	Average utility histograms for P-FCN-SegNet (left) and E-FCN-SegNet (right) with $\gamma = 0.8$ on the MIT-scene Parsing dataset without (top)/with (bottom) soft labels.	82
4.15	Average utility histograms for P-FCN-CRF (left) and E-FCN-CRF (right) with $\gamma = 0.8$ on the SIFT Flow dataset without (top)/with (bottom) soft labels.	82
4.16	Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of MIT-scene Parsing and SIFT Flow datasets (top) and the testing set from the Pascal VOC dataset (bottom) when the learning set is from the Pascal VOC dataset without (left)/with (right) soft labels.	83
4.17	Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of Pascal VOC and SIFT Flow (top) and the testing set of the MIT-scene Parsing dataset (bottom) when the learning set is from the MIT-scene Parsing dataset without (left)/with (right) soft labels.	83
4.18	Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of Pascal VOC and MIT-scene Parsing (top) and the testing set of the SIFT Flow dataset (bottom) when the learning set is from the SIFT Flow dataset without (left)/with (right) soft labels.	84
4.19	Examples of novelty detection from the MIT-scene Parsing dataset.	85
5.1	Architecture of the evidential information-fusion approach	90
5.2	Semantic relationship of the classes in the Tiny ImageNet, Flower-102 and CIFAR-10 datasets.	94
5.3	Semantic relationship of the classes on the CIFAR-10, CUB and Pet datasets.	99
5.4	Segmentation examples from the Cityscapes dataset before and after fusion.	103
5.5	Architecture of a shallow CNN with 91k parameters.	107
5.6	Results of CNN combination experiment on the Tiny ImageNet dataset.	108
5.7	Results of FCN combination experiment on the Cityscapes dataset.	110

List of Tables

1.1	Utility matrix extended by an OWA operator ($\gamma = 0.8$).	14
2.1	Examples of different formats of data in CNNs [37].	24
2.2	Performance-Efficiency FitNet architectures [106].	26
2.3	Performance-Efficiency ResNet backbones [48].	27
3.1	Examples of DS layer outputs	42
3.2	Example of utilities and losses	42
3.3	Three CNN backbones used on CIFAR-10 dataset.	46
3.4	Test average utilities in precise classification on CIFAR-10 dataset. . .	46
3.5	Test average utilities for precise classification of the CIFAR-100 data set after transfer learning.	47
3.6	Prediction distribution for the evidential CNN classifier with the NIN backbone on the CIFAR-10 dataset.	49
3.7	Label classification/utilities with different γ	51
3.8	Set-valued assignment rates using the selected and 2^Ω acts (unit:%). .	53
3.9	Proportions of samples correctly assigned to acts in 2^Ω and incorrectly assigned to selected acts, for different values of γ	53
3.10	Three baseline CNN backbones used on UrbanSound 8K.	56
3.11	Test average utilities in precise classification on UrbanSound 8K. . . .	56
3.12	Three baseline CNN backbones used on SemEval-2010 Task 8.	60
3.13	Test average utilities in precise classification on SemEval-2010 Task 8. .	60
4.1	Lists of classes for the Pascal VOC, MIT-scene Parsing and SIFT Flow datasets in the semantic segmentation experiments.	69
4.2	Utility matrix considering soft labels with $\gamma = 0.8$	71
4.3	Performance evaluation of precise segmentation.	73
4.4	Percentage of pixels from some unknown classes in the MIT-scene Parsing and SIFT Flow datasets.	85
5.1	Numbers of prototypes in Dempster-Shafer layers.	95
5.2	Average test error rates (in percent) of different classifiers on the classification experiment #1.	96
5.3	Test error rates (in percent) before and after information fusion on CIFAR-10 using the FitNit-4 architecture.	97

5.4	Examples of probability mass functions on the Tiny ImageNet dataset before and after fusion by the MFE strategy.	97
5.5	Error rates (in percent) of some classes on the Tiny ImageNet dataset before and after fusion.	98
5.6	Lists of classes in the CIFAR-10, CUB, Oxford-IIIT Pet datasets. The notations ω_0^2 and ω_0^3 stand for the “anything else” class added to the frames of the CUB and Oxford datasets.	99
5.7	Average test error rates (in percent) of different classifiers on the classification experiment #2.	100
5.8	Examples of mass functions on the CIFAR-10 and Oxford-IIT pet datasetS before and after fusion by the MFE strategy.	101
5.9	Lists of classes for the Pascal VOC, Cityscapes, and Stanford background datasets in the semantic segmentation experiments #1.	101
5.10	Numbers of output units in the encoder-decoder architectures.	102
5.11	Mean intersection over union of different FCN models on the segmentation experiment #1.	104
5.12	Test IoU before and after information fusion on the Cityscapes dataset using the FCN-CRF architecture.	104
5.13	Mean intersection over union of different FCN models on the segmentation experiment #2.	105

Acronyms & notations

Acronyms

DNN	Deep Neural Network
CNN	Convolutional Neural Network
E- and P-CNN	Evidential and Probabilistic Convolutional Neural Network
FCN	Fully Convolutional Network
E- and P-FCN	Evidential and Probabilistic Fully Convolutional Network
DST	Dempster-Shafer Theory
ENN	Evidential Neural Network
OWA	Ordered Weighted Average
DM	Decision-Maker
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
NIN	Network In Network
ViT	Vision Transformer
MLP	Multi-Layer Perceptron
CRF	Conditional Random Field
HAC	Hierarchical Agglomerative Clustering
CHI	Calinski-Harabasz Index
AA	Average Accuracy
AU	Average Utility
PU	Pixel Utility
IoU	Intersection over Union
UIoU	Utility of Intersection over Union
ECE	Expected Calibration Error
CUB	Caltech-UCSD Birds-200-2011
GPU	Graphics Processor Unit
FLOPs	FLating point OPerations

Symbols

Ω	frame of discernment
m	mass function

α_m	discounted mass function
$m^{\Omega \uparrow \Theta}$	vacuous extension of mass m on Ω to frame Θ
Bel and Pl	belief and plausibility functions
pl	contour function
$BetP$	Pignistic probability
κ	degree of conflict
\mathcal{F}	set of acts
\mathbb{U}	utility matrix
$\underline{\mathbb{E}}_m, \overline{\mathbb{E}}_m, \mathbb{E}_{m,\nu}, \mathbb{E}_{m,p}$	lower, upper, Hurwicz, and pignistic expected utilities associated to m
ν	pessimism index
γ	imprecision tolerance degree
$\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$	a learning set of N examples
$\{\mathbf{p}^1, \dots, \mathbf{p}^n\}$	n prototypes in an ENN classifier or Dempster-Shafer layer
s^i	distance-based support between \mathbf{x} and prototype \mathbf{p}^i
d^i	Euclidean distance between \mathbf{x} and prototype \mathbf{p}^i
$\tau^i, \eta^i, \text{ and } \xi^i$	parameters associated with prototype \mathbf{p}^i
h_j^i	degree of membership of prototype \mathbf{p}^i to class ω_j
$\mathbf{z} = (z^1, \dots, z^D)$	input data consisting of D tensors
$H \times W$	size of tensor z^i
w and b	weight and bias of a convolution kernel
c	output tensor after a convolution operation
r	stride with which a kernel slides over an input tensor
S	size of non-overlapping window in a pooling operation
co	prediction confidence
au	average utility
I_q	set of samples with prediction confidence in interval $(\frac{q-1}{Q}, \frac{q}{Q}]$
$p_{m^1 \oplus \dots \oplus m^V}$ and p_V	aggregated probability mass functions with V mass functions $m^1 \dots m^V$
ω_*	precise class label
A_*	soft label
$\mathbb{1}_A(\hat{\omega})$	indicator function

Introduction

Machine learning is widely used in different applications, such as advanced learning driver assistance systems [13, 29, 134, 146], human-machine interaction [110, 151], medical imaging [34, 83], etc. In recent years, one approach to machine learning, deep learning [70], has achieved state-of-the-art results in these applications. For example, convolutional neural networks (CNNs) [113] and fully convolutional networks (FCNs) [77] are powerful techniques for pattern recognition [61, 123] and semantic segmentation [104, 126], respectively. Such achievements mainly benefit from the robust and reliable feature representation of deep neural networks (DNNs) with multiple layers, which progressively extract higher-level features from raw data and then convert them into class probabilities [18]. In this thesis, we call such DNNs in the framework of probability theory “*probabilistic DNNs*”. However, despite the power and flexibility of probabilistic DNNs, they still face the problem of data uncertainty in many real-world applications, mainly including ambiguous, unreliable, imprecise, and incomplete data.

One problem of data uncertainty is the ambiguity in raw data or their representations, with which a machine-learning algorithm cannot make a reliable prediction. For instance, in many applications of DNNs, we may not be able to reliably classify a sample into a single class; multiple classes have similar probabilities because the feature representations of the sample are ambiguous and close to the ones of the learning samples in two or more similar, but different, classes. In such a case, probabilistic DNNs often arbitrarily assign the sample to one of the possible classes, which may result in misclassification, even though some of them try to make imprecise decisions using precise probability [10, 89].

Sometimes, uncertainty arises from imprecise and unreliable data. For instance, in the majority of the learning sets for DNNs, such as CIFAR-10 [63] and Pascal VOC 2012 [30], all samples are precisely labeled with one and only one class, called the *precise labels*, even if the true labels are actually uncertain. This is the case, for example, for the pixels at object borders and small objects in a semantic-segmentation learning set. Samples with precise but incorrect labels are unreliable and may have negative effects on learning systems [3, 90]. However, common probabilistic DNNs typically ignore such uncertainty since probability theory cannot capture the imprecision aspect of such label data, even though they sometimes use label smoothing to represent such uncertainty [60, 79], which allows one class to have the largest probability and the rests have very small and equal probabilities when labeling a sample.

Unfortunately, such labels still cannot represent the uncertainty well because not all labels are unreliable and not all classes in a smoothed label have the same probability for an object.

In addition, data may be inherently incomplete, which causes two problems. The first one concerns the capacity of *novelty and outlier detection* [21]. An ideal algorithm should detect “unknown” objects belonging to classes that are not represented in the learning set. However, not all classes are labeled in many learning sets and a trained probabilistic DNNs usually randomly assigns the “unknown” objects to one of the known classes. In probability theory, there are two main directions to solve the problem: assigning to an extra class or designing an additional outlier detector. The former tends to assign some outliers to an extra class, such as “background” in some semantic-segmentation tasks [13, 30], which requires a learning set to provide outlier examples. The latter requires an extra outlier detector before the probabilistic DNNs perform their task [36, 97].

Another problem arising from incomplete data is the partial and imperfect outputs of probabilistic DNNs, which make it difficult to combine heterogeneous DNNs. In the past ten years, hundreds of deep networks have been developed using available data sets with different sets of classes and different granularities. To utilize the most of existing techniques, one tries to combine networks trained from such heterogeneous datasets to obtain a general one. However, these existing DNNs output different probabilistic information that may be uncertain but also partial. For example, given a new image, a network outputs class probabilities for the CIFAR-10 dataset [63], including the probability of class “bird”. However, compared to the probabilistic outputs of a network trained by the Caltech-UCSD Birds 200 dataset with 200 species of birds [141], the output information of the CIFAR-10 network is partial and imperfect. Unfortunately, Bayesian probability theory is not flexible enough to fuse such partial and imperfect outputs when combining heterogeneous networks [146].

These uncertainty problems mainly derive from the fact that most DNNs work within the probabilistic framework. Probability theory only captures the randomness aspect of the data, but neither ambiguity nor incompleteness, which are inherent in uncertain data. Therefore, during the last decade, many theories have been combined with DNNs to solve these uncertainty problems, such as Dempster-Shafer theory (DST) [16, 114], fuzzy sets [152], random sets [91], and imprecise probability [138]. This thesis aims at combining DST and DNNs to solve these problems. As a generalization of probability theory, Dempster-Shafer theory [16, 114], also referred to as *evidence theory* or *theory of belief functions*, is a well-established formalism for representing and combining a large variety of uncertain information for decision-making [149]. It is based on representing independent pieces of evidence by completely monotone capacities and combining them using a generic operator called Dempster’s rule [114].

DST has been increasingly applied to machine learning with uncertain data, following two main directions: designing evidential classifiers [20, 25] and combining information from multiple models [49, 68, 86, 87, 135]. Typically, an evidential classifier breaks down the evidence of each input vector into elementary mass functions and combines them by Dempster’s rule. These mass functions represent the uncertainty in the input features. For example, there exists ambiguous data in an input vector when the values of two masses in an evidential classifier outputs are very close [25]. Data unreliability and imprecision can also be represented by mass functions in the framework of DST [59, 58, 99, 131]. In the direction of multi-model combination, classifier outputs are expressed as belief functions and combined by Dempster’s rule or any other rule [2, 57, 102]. This direction provides the possibility to process incomplete data by extending heterogeneous imperfect outputs into a common frame and combining them by Dempster’s rule [146].

Motivated by the high expressivity of DST as an uncertainty representation framework, the goal of our study in this thesis is to 1) develop new DNNs in the framework of DST with the capacity to deal with data uncertainty introduced above, and 2) demonstrate the advantages of the new DNNs by applying them to pattern classification, semantic segmentation, and multi-network fusion. The basic idea of our study is to combine the frameworks of DST and DNN, where a DNN provides the feature representations of input samples and a DST-based evidential classifier converts the representations into mass functions for decision-making with data uncertainty. The contributions of the thesis can be summarized by the following two points:

1. *Evidential DNNs*: We propose a new DNN in the DST framework, called *evidential DNN*. An evidential DNN handles confusing and ambiguous samples, as well as outliers, by making set-valued and cautious decisions. An evidential DNN can also update its parameters using a learning set with uncertain and imprecise labels represented in the form of DST-based mass functions. These advantages of the evidential DNNs have been demonstrated in pattern classification and semantic segmentation applications.
2. *Evidential fusion of heterogeneous DNNs*: The proposed combined framework of DST and DNNs provides an evidential-fusion approach to combine heterogeneous DNNs. This approach is flexible enough to combine different pre-trained DNNs with different sets of classes at any stage to obtain a more general network. In addition, the approach provides a new way to combine simple and shallow networks for a complicated task, which has the potential to make training easier and avoid the use of very deep networks. These advantages of the evidential DNNs have also been verified in pattern classification and semantic segmentation tasks.

This thesis contains five chapters organized in two parts:

Part I introduces the theoretical background that supports the thesis. Chapter 1 recalls some notions of DST useful for machine learning with uncertain data. We describe how information is represented and combined in the framework of DST. Some useful operations and decision rules with belief functions are presented. This chapter also recalls the evidential neural network classifier based on DST introduced in [20]. Chapter 2 focuses on the modern practices of DNNs. We introduce the fundamental neural network layers and the state-of-the-art DNNs used in the study.

Part II is devoted to our own contributions. In Chapter 3, we describe an evidential CNN classifier that hybridizes DST and DNN by “plugging” an evidential neural network classifier at the backbone output of a CNN. The idea of hybridizing DST and DNN is then extended to FCNs with the applications of semantic segmentation in Chapter 4. Finally, Chapter 5 describes the proposed approach of evidential fusion to combine heterogeneous DNNs. This approach aggregates the belief functions computed by deep neural networks expressed in different frames of discernment using Dempster’s rule.

Part I

Background

Chapter 1

Dempster-Shafer theory

Machine learning-based tasks, such as pattern recognition and semantic segmentation, require powerful tools to represent and combine different types of uncertain information. The Dempster-Shafer theory (DST) of belief functions [16, 114], also referred to as *evidence theory*, offers a well-founded and workable framework for the problem. It represents independent pieces of evidence by completely monotone capacities and combines them using a generic operator called Dempster's rule [114]. It is a well-established formalism for reasoning and making decisions with uncertainty [19, 24, 67, 149]. It is also a generalization of possibility theory and is closely linked to other theories including fuzzy sets [26, 152], random sets [26, 91] and imprecise probability [138].

This chapter first recalls how information can be represented in the framework of belief functions in Section 1.1. In Section 1.2, we describe some operations of belief functions, followed by the issue of decision-making using belief functions in Section 1.3. Finally, in Section 1.4, we recall the distance-based evidential neural network (ENN) classifier based on DST introduced in [20], which will be used in this study.

1.1 Information representation

1.1.1 Mass function

Let $\Omega = \{\omega_1, \dots, \omega_M\}$ be a set of *state of nature*, 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

$$\sum_{A \subseteq \Omega} m(A) = 1. \quad (1.1)$$

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 *Bayesian* if its focal sets are singletons; it is then equivalent to a probability distribution. Thus, a probability distribution is a particular kind of mass function that encodes precise information. A mass function

is said to be *logical* if it has only one focal set, i.e., if it is such that $m(A) = 1$ for some $A \subseteq \Omega$.

Given a mass function m , *belief* and *plausibility* functions are defined, respectively, as:

$$Bel(A) = \sum_{B \subseteq A} m(B), \quad \forall A \subseteq \Omega, \quad (1.2)$$

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B), \quad \forall A \subseteq \Omega. \quad (1.3)$$

For $A \subseteq \Omega$, $Bel(A)$ measures the degree of support of A , while $Pl(A) = 1 - Bel(\bar{A})$ measures the lack of support of the complement of A . The *contour function* $pl: \Omega \rightarrow [0, 1]$ is the restriction of the plausibility function Pl to singletons; it is defined as

$$pl(\omega) = Pl(\{\omega\}), \quad \forall \omega \in \Omega. \quad (1.4)$$

A mass function can be transformed into a probability distribution $BetP$ by the pignistic transformation [117] defined as

$$BetP(\omega) = \sum_{A \subseteq \Omega, \omega \in A} \frac{m(A)}{|A|}, \quad \forall \omega \in \Omega. \quad (1.5)$$

The mass assigned to a set A is simply equally distributed to its elements.

1.1.2 Discounting

In the theory of belief functions, knowledge about the reliability of a source of information can be handled by a discounting factor [25]. A discounting factor is used to *weaken* a mass function by transferring some masses to the ignorance state.

For a factor $\alpha \in [0, 1]$, the *discounted mass function* ${}^\alpha m$ is defined as

$${}^\alpha m(A) = (1 - \alpha)m(A) \quad \forall A \subseteq \Omega \quad (1.6a)$$

$${}^\alpha m(\Omega) = (1 - \alpha)m(\Omega) + \alpha. \quad (1.6b)$$

If $\alpha = 0$, the information is considered reliable and is kept as is. On the other hand, if $\alpha = 1$, the information is totally unreliable and leads to the vacuous mass function. Smets [118] showed that the discounting equation (1.6) can be derived by interpreting $1 - \alpha$ as the degree of belief that the information is reliable.

1.2 Operations of belief functions

1.2.1 Dempster's rule

Two mass functions m_1 and m_2 on the same frame Ω representing independent items of evidence can be combined conjunctively by *Dempster's rule* [114] defined as follows:

$$(m_1 \oplus m_2)(A) = \frac{1}{1 - \kappa} \sum_{B \cap C = A} m_1(B) m_2(C) \quad (1.7)$$

for all $A \subseteq \Omega$, $A \neq \emptyset$, and $(m_1 \oplus m_2)(\emptyset) = 0$. In (1.7), κ is the *degree of conflict* between the two mass functions, defined as

$$\kappa := \sum_{B \cap C \neq \emptyset} m_1(B) m_2(C) \quad (1.8)$$

Mass functions m_1 and m_2 can be combined if and only if $\kappa < 1$. The mass function $m_1 \oplus m_2$ is called the *orthogonal sum* of m_1 and m_2 . The operator \oplus is commutative and associative, and the vacuous mass function is its only neutral element. The contour function $pl_1 \oplus pl_2$ associated to $m_1 \oplus m_2$ can be computed as

$$pl_1 \oplus pl_2(\omega) = \frac{pl_1(\omega)pl_2(\omega)}{1 - \kappa}. \quad (1.9)$$

Denceux [23] remarks that it is sometimes useful to approximate a mass function of DST m by a probability mass function $p_m : \Omega \rightarrow [0, 1]$. One such approximation with good properties is obtained by normalizing the contour function [11, 137], such that

$$p_m(\omega_i) := \frac{pl(\omega_i)}{\sum_{j=1}^M pl(\omega_j)} \quad i = 1, \dots, M, \quad (1.10)$$

where M is the number of classes on Ω . As a consequence of (1.9), the so-called *plausibility transformation* (1.10) has the following interesting property in relation with Dempster's rule:

$$p_{m_1 \oplus m_2}(\omega_i) \propto p_{m_1}(\omega_i)p_{m_2}(\omega_i), \quad i = 1, \dots, M, \quad (1.11)$$

i.e., the probability distribution associated to $m_1 \oplus m_2$ can be computed in $O(M)$ arithmetical operations by multiplying the probability distributions p_{m_1} and p_{m_2} element-wise, and re-normalizing. This probability can be used to simplify the processes of orthogonal sum of m_1 and m_2 using Dempster's rule.

1.2.2 Change of frame of discernment

Refinement. Because mass functions are directly defined over sets of classes, refinement and imprecise information can be easily handled. A *refining* from frame Ω

to frame Θ , as defined in [114], is a mapping $\rho : 2^\Omega \rightarrow 2^\Theta$ such that:

$$\bullet \{\rho(\{\omega\}), \omega \in \Omega\} \subseteq 2^\Theta \text{ is a partition of } \Theta, \quad (1.12a)$$

$$\bullet \forall A \subseteq \Omega, \rho(A) = \bigcup_{\omega \in A} \rho(\{\omega\}). \quad (1.12b)$$

The frame Θ is then called a *refinement* of Ω .

Given a mass function m^Ω on Ω , its vacuous extension $m^{\Omega \uparrow \Theta}$ in Θ is the mass function defined on frame Θ as

$$m^{\Omega \uparrow \Theta}(B) = \begin{cases} m^\Omega(A) & \text{if } \exists A \subseteq \Omega, \quad B = \rho(A), \\ 0 & \text{otherwise,} \end{cases} \quad (1.13)$$

for all $B \subseteq \Theta$. Two frames of discernment are said to be *compatible* if they have a common refinement. Given two mass functions m^{Ω_1} and m^{Ω_2} on compatible frames Ω_1 and Ω_2 , their orthogonal sum $m^{\Omega_1} \oplus m^{\Omega_2}$ is defined as the orthogonal sum of their vacuous extensions in their common refinement Θ : $m^{\Omega_1} \oplus m^{\Omega_2} = m^{\Omega_1 \uparrow \Theta} \oplus m^{\Omega_2 \uparrow \Theta}$. Also, the orthogonal sum of $m^{\Omega_1 \uparrow \Theta}$ and $m^{\Omega_2 \uparrow \Theta}$ can be simplified by the plausibility transformation (1.10) and (1.11) when only considering singleton focal sets.

Coarsening. The opposite operation to refining is called *coarsening*. If a frame of discernment Θ is a refinement of Ω , then Ω is a coarsening of Θ . By definition, the cardinality of a refinement Θ is greater than that of the original frame Ω . This implies that a refining $\rho : 2^\Omega \rightarrow 2^\Theta$ cannot be bijective, thus not invertible. The inner reduction $\underline{\varphi} : 2^\Theta \rightarrow 2^\Omega$ and outer reduction $\overline{\varphi} : 2^\Theta \rightarrow 2^\Omega$ associated to a refining ρ are defined, respectively, as

$$\underline{\varphi}(B) = \{\omega \in \Omega | \rho(\{\omega\}) \subseteq B\}, \quad \forall B \subseteq \Theta, \quad (1.14)$$

$$\overline{\varphi}(B) = \{\omega \in \Omega | \rho(\{\omega\}) \cap B \neq \emptyset\}, \quad \forall B \subseteq \Theta. \quad (1.15)$$

The inner and outer reduction of a mass function m^Θ in Ω are defined, respectively, as

$$\underline{m}^\Omega(A) = \sum_{B \subseteq \Theta, \underline{\varphi}(B)=A} m^\Theta(B), \quad \forall A \subseteq \Omega, \quad (1.16)$$

$$\overline{m}^\Omega(A) = \sum_{B \subseteq \Theta, \overline{\varphi}(B)=A} m^\Theta(B), \quad \forall A \subseteq \Omega. \quad (1.17)$$

The inner reduction \underline{m}^Ω in (1.16) is not normalized, i.e., we may have $\underline{m}^\Omega(\emptyset) > 0$. However, the outer reduction \overline{m}^Ω in (1.17) is always a normalized mass function.

Given a mass function m^Θ defined on Θ and its outer reduction \overline{m}^Ω on Ω , the following propositions hold [145]:

$$\overline{Bel}^\Omega(A) = Bel^\Theta(\rho(A)), \quad \forall A \subseteq \Omega, \quad (1.18)$$

$$\overline{Pl}^\Omega(A) = Pl^\Theta(\rho(A)), \quad \forall A \subseteq \Omega. \quad (1.19)$$

In practice, the outer reduction is often preferred as it is consistent with respect to the belief and plausibility functions.

1.3 Decision-making with belief functions

There are several strategies for decision-making with belief functions [22]. This section introduces the strategies that will be used in the rest of this thesis, including precise and imprecise decision with belief functions addressed in Sections 1.3.1 and 1.3.2, respectively.

1.3.1 Precise decision with belief functions

A machine-learning algorithm should predict the class of each new sample based on a learning set of labeled instances. The most common decision is *precise decision*, in which a sample is assigned into one and only one possible class. Let $\Omega = \{\omega_1, \dots, \omega_M\}$ be the set of classes. For a problem with only precise decisions, two simple strategies consist in choosing the class with maximum belief or plausibility, called the pessimistic and optimistic strategies, respectively [22]. The optimistic strategy amount to choosing the class with maximum contour function since the contour function is the restriction of the plausibility function to singletons (1.4). This strategy will be used for decision-making in Chapter 5 because it can reduce the computation complexity when using mass functions on different compatible frames for decision-making [23]. Another widely used strategy is to use the pignistic probability transformation (1.5) and select the singleton with the maximum probability.

Dencœux and Ma [22, 82] propose a general framework of decision-making with belief functions. For a problem with only precise decisions, an act is defined as the assignment of an example to one and only one of the M classes. The set of acts is $\mathcal{F} = \{f_{\omega_1}, \dots, f_{\omega_M}\}$, where f_{ω_i} denotes assignment to class ω_i . To make a decision, they define a utility matrix $\mathbb{U}_{M \times M}$, whose general term $u_{ij} \in [0, 1]$ is the utility of assigning an example to class ω_i when the true class is ω_j . Here, $\mathbb{U}_{M \times M}$ is called the *original utility matrix*. For decision-making with belief functions, each act f_{ω_i} induces expected utilities, such as the lower and upper expected utilities:

$$\mathbb{E}_m(f_{\omega_i}) = \sum_{B \subseteq \Omega} m(B) \min_{\omega_j \in B} u_{ij}, \quad (1.20a)$$

$$\bar{\mathbb{E}}_m(f_{\omega_i}) = \sum_{B \subseteq \Omega} m(B) \max_{\omega_j \in B} u_{ij}. \quad (1.20b)$$

A pessimistic decision maker (DM) typically selects the act with the largest lower expected utility, while an optimistic DM maximizes the upper expected utility. The generalized Hurwicz decision criterion [22, 55, 56, 120] models the DM's attitude to ambiguity by a *pessimism index* ν and defines the expected utility of act f_{ω_i} as the weighted sum

$$\mathbb{E}_{m,\nu}(f_{\omega_i}) = \nu \underline{\mathbb{E}}_m(f_{\omega_i}) + (1 - \nu) \bar{\mathbb{E}}_m(f_{\omega_i}). \quad (1.21)$$

It is clear that the pessimistic and optimistic attitudes correspond, respectively, to $\nu = 1$ and $\nu = 0$. Besides, we can also compute the pignistic expected utility of each act f_{ω_i} as

$$\mathbb{E}_{m,p}(f_{\omega_i}) = \sum_{j=1}^M u_{ij} \text{Bet}P_m(\{\omega_j\}), \quad (1.22)$$

where $\text{Bet}P_m$ is the pignistic probability defined by Eq. (1.5).

1.3.2 Imprecise decision with belief functions

A hard and precise decision often leads to an error in case of high uncertainty. For example, ambiguity occurs when a feature vector does not contain sufficient information to identify a precise class, and multiple classes have similar probabilities. Also, a classifier with only precise-decision options may fail to identify outliers belonging to a class that is not represented in the learning set. *Imprecise decision* is a potential way to solve this problem. In the thesis, it is defined as the assignment of a new observation to a non-empty subset of the class set when the uncertainty is too high to make a precise decision. We focus on two types of imprecise decisions: precise decision with a rejection option and set-valued decision.

Precise decision with a rejection option is defined as assigning a sample into one possible class or rejection. The semantics of rejection is that a classifier rejects to make a precise decision using the given information with too high uncertainty. Dencœux [21] proposed three strategies for precise decision with a rejection option. The set of acts is $\mathcal{F} = \{f_{\omega_0}, f_{\omega_1}, \dots, f_{\omega_M}\}$, where f_{ω_0} is a rejection act. Assuming the cost of a correct act to be 0, the cost of an incorrect act to be 1 and the cost of rejection to be $\lambda_0 \in (0, 1)$, the three strategies for rejection can be expressed as

Maximum credibility: $\max_{j=1, \dots, M} \text{Bel}(\{\omega_j\}) < 1 - \lambda_0$

Maximum plausibility: $\max_{j=1, \dots, M} \text{Pl}(\{\omega_j\}) < 1 - \lambda_0$

Maximum pignistic probability: $\max_{j=1, \dots, M} \text{Bet}P(\omega_j) < 1 - \lambda_0$.

Otherwise, the pattern is assigned to class ω_j with $j = \arg \max_{k=1, \dots, M} m(\{\omega_k\})$ if the focal sets of mass functions consist of the singletons and Ω . For the maximum plausibility and maximum pignistic probability rules, rejection is possible if and only

if $0 \leq \lambda_0 \leq 1 - 1/M$, whereas a rejection action for the maximum credibility rule only requires $0 \leq \lambda_0 \leq 1$.

Set-valued decision [45, 82, 89] is defined as the assignment of a new observation to a non-empty subset of classes when the uncertainty is too high to make a precise classification. For instance, given a class set $\Omega = \{\omega_1, \omega_2, \omega_3\}$, we may not be able to reliably classify a sample \mathbf{x} into a single class, but it may be almost sure that it does not belong to ω_3 . In this case, it is more cautious to assign \mathbf{x} to the set $\{\omega_1, \omega_2\}$. Ma and Dencœux [82] propose an approach to conduct set-valued assignment under uncertainty by generalizing the set of acts as partially assigning a sample to a non-empty subset A of Ω . Thus, the set of acts becomes $\mathcal{F} = \{f_A, A \in 2^\Omega \setminus \{\emptyset\}\}$, in which 2^Ω is the power set of Ω and f_A denotes the assignment to a subset A . In the thesis, subset A is defined as a *multi-class set* if $|A| \geq 2$. Precise decision with a reject option in [8, 130] can be regarded as a special case of set-valued classification, rejection being equivalent to assigning a sample to the entire set of possible classes.

For decision-making with \mathcal{F} , the original utility matrix $\mathbb{U}_{M \times M}$ is extended to $\mathbb{U}_{(2^\Omega - 1) \times M}$, where each element $\hat{u}_{A,j}$ denotes the utility of assigning a sample to set A of classes when the true label is ω_j . When the true class is ω_j , the utility of assigning a sample to set A is defined as an Ordered Weighted Average (OWA) aggregation [148] of the utilities of each precise assignment in A as

$$\hat{u}_{A,j} = \sum_{k=1}^{|A|} g_k \cdot u_{(k)j}^A, \quad (1.23)$$

where $u_{(k)j}^A$ is the k -th largest element in the set $\{u_{ij}^A, \omega_i \in A\}$ made up of the elements in the original utility matrix $\mathbb{U}_{M \times M}$, and weights $\mathbf{g} = (g_1, \dots, g_{|A|})$ represent the preference to choose $u_{(k)j}^A$ when a DM has to make a precise decision among a set of possible choices. The elements in weight vector \mathbf{g} represent the DM's *tolerance to imprecision*. 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 imprecise A is. In the case, only the maximum utility of elements in set $\{u_{ij}^A, \omega_i \in A\}$ is considered: $(g_1, g_2, \dots, g_{|A|}) = (1, 0, \dots, 0)$. 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

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

Following [82], we can also determine the weight vector \mathbf{g} of the OWA operators by adapting O'Hagan's method [96]. *The imprecision tolerance degree* can be defined as

$$TDI(\mathbf{g}) = \sum_{k=1}^{|A|} \frac{|A| - k}{|A| - 1} g_k = \gamma, \quad (1.24)$$

Table 1.1: Utility matrix extended by an OWA operator ($\gamma = 0.8$).

	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
$f_{\{\Omega\}}$	0.682	0.682	0.682

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 preferable to an imprecise one when $\gamma < 0.5$ [82]. Given a value of γ , we can compute the weights of the OWA operator by maximizing the entropy

$$ENT(\mathbf{g}) = - \sum_{k=1}^{|\mathcal{A}|} g_k \log g_k \quad (1.25)$$

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

Example 1.1 Table 1.1 shows an example of the extended utility matrix generated by an OWA operator with $\gamma = 0.8$ for a classification problem. The first three rows constitute the original utility matrix, indicating that the utility equals 1 when assigning a sample to its true class, otherwise it equals 0. 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 .

Based on an extended utility matrix $\mathbb{U}_{(2^\Omega - 1) \times M}$ and a mass function \mathbf{m} , we can compute the expected utility of an act assigning a sample to set A using the generalized Hurwicz criterion (1.21) as

$$\mathbb{E}_{\mathbf{m}, \nu}(f_A) = \nu \underline{\mathbb{E}}_{\mathbf{m}}(f_A) + (1 - \nu) \overline{\mathbb{E}}_{\mathbf{m}}(f_A), \quad (1.26a)$$

where $\underline{\mathbb{E}}_{\mathbf{m}}(f_A)$ and $\overline{\mathbb{E}}_{\mathbf{m}}(f_A)$ are, respectively, the lower and upper expected utilities

$$\underline{\mathbb{E}}_{\mathbf{m}}(f_A) = \sum_{B \subseteq \Omega} m(B) \min_{\omega_j \in B} \hat{u}_{A,j}, \quad (1.26b)$$

$$\overline{\mathbb{E}}_{\mathbf{m}}(f_A) = \sum_{B \subseteq \Omega} m(B) \max_{\omega_j \in B} \hat{u}_{A,j}, \quad (1.26c)$$

and ν is the pessimism index that should be considered as a hyperparameter when the generalized Hurwicz criterion is used in a classifier. We can also compute the

pignistic expected utility of assigning that sample to set A as

$$\mathbb{E}_{m,p}(f_A) = \sum_{j=1}^M \hat{u}_{A,j} \text{Bet}P_m(\{\omega_j\}), \quad (1.27)$$

where $\text{Bet}P_m$ is the pignistic probability defined by Eq. (1.5). The sample is finally assigned to set A such that

$$A = \arg \max_{\emptyset \neq B \subseteq \Omega} \mathbb{E}_m(f_B). \quad (1.28)$$

1.4 Evidential neural network based on Dempster-Shafer theory

An evidential classifier quantifies prediction uncertainty using mass functions, see [20, 25, 73, 122]. The output mass functions can then be used for decision-making [7, 40], as introduced in Section 1.3. Thanks to the generality and expressiveness of the DST formalism, the outputs of an evidential classifier provide more information than those of conventional classifiers (e.g., a neural network with a softmax layer), which quantify prediction uncertainty using probability distribution. Over the years, two main principles for designing an evidential classifier have been proposed: the model-based and distance-based approaches. The former uses estimated class-conditional distributions [118], while the latter constructs mass functions based on distances to prototypes [20, 25]. This section introduces a particular evidential classifier that will be combined with deep neural networks in the study.

Based on the DST, Dencœux [20] proposed a distance-based neural-network layer for constructing mass functions, also known as the *evidential neural network (ENN) classifier*. In the ENN classifier, the proximity of an input vector to prototypes is considered as evidence about its class. This evidence is converted into mass functions and combined using Dempster’s rule.

We consider a learning set $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subset \mathbb{R}^P$ of N examples represented with P -dimensional feature vectors, and an ENN classifier composed of n prototypes $\{\mathbf{p}^1, \dots, \mathbf{p}^n\}$ in \mathbb{R}^P . For a test sample \mathbf{x} , the ENN classifier constructs mass functions that quantify the uncertainty about its class in $\Omega = \{\omega_1, \dots, \omega_M\}$, using a three-step procedure. This procedure can be implemented in a complex neural-network layer composed of three simple layers L_1 , L_2 , and L_3 , as shown in Figure 1.1a. The “DS layer” will be plugged into deep neural networks in Chapters 3-5. The three-step procedure is defined as follows.

Step 1: The distance-based support between \mathbf{x} and each reference pattern \mathbf{p}^i is computed as

$$s^i = \tau^i \exp(-(\eta^i d^i)^2) \quad i = 1, \dots, n, \quad (1.29)$$

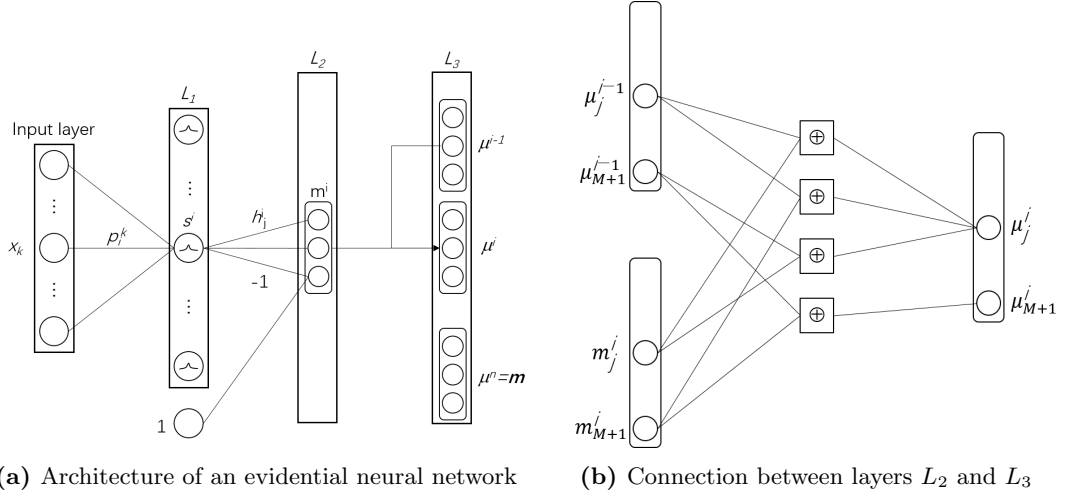


Figure 1.1: Evidential neural network classifier [20].

where $d^i = \|\mathbf{x} - \mathbf{p}^i\|$ is the Euclidean distance between \mathbf{x} and prototype \mathbf{p}^i , and $\tau^i \in (0, 1)$ and $\eta^i \in \mathbb{R}$ are parameters associated with prototype \mathbf{p}^i . A new parameter $\xi^i \in \mathbb{R}$ is introduced as $\tau^i = (1 + \exp(-\xi^i))^{-1}$ subject to the constraint $\tau^i \in (0, 1)$. Prototype vectors $\mathbf{p}^1, \dots, \mathbf{p}^n$ can be considered as vectors of connection weights between the input layer and a hidden layer of n radial basis function (RBF) units.

Step 2: The mass function m^i associated to reference pattern \mathbf{p}^i is computed as

$$m^i(\{\omega_j\}) = h_j^i s^i, \quad j = 1, \dots, M \quad (1.30a)$$

$$m^i(\Omega) = 1 - s^i, \quad (1.30b)$$

where h_j^i is the degree of membership of prototype \mathbf{p}^i to class ω_j with $\sum_{j=1}^M h_j^i = 1$. We denote the vector of masses induced by prototype \mathbf{p}^i as

$$\mathbf{m}^i = (m^i(\{\omega_1\}), \dots, m^i(\{\omega_M\}), m^i(\Omega))^T.$$

Eq. (1.30) can be regarded as computing the activation of units in a second hidden layer of the ENN classifier, composed of n modules of $M + 1$ units each. The result of module i corresponds to the belief masses assigned by m^i .

Step 3: The n mass functions \mathbf{m}^i , $i = 1, \dots, n$, are aggregated by Dempster's rule (1.7), as shown in Figure 1.1b. The combined mass function is computed iteratively as $\mu^1 = m^1$ and $\mu^i = \mu^{i-1} \cap m^i$ for $i = 2, \dots, n$, where \cap denotes Dempster's rule without normalization. We have

$$\mu^i(\{\omega_j\}) = \mu^{i-1}(\{\omega_j\})m^i(\{\omega_j\}) + \mu^{i-1}(\{\omega_j\})m^i(\{\Omega\}) + \mu^{i-1}(\Omega)m^i(\{\omega_j\}) \quad (1.31a)$$

for $i = 2, \dots, n$ and $j = 1, \dots, M$, and

$$\mu^i(\Omega) = \mu^{i-1}(\Omega)m^i(\Omega) \quad i = 2, \dots, n. \quad (1.31b)$$

The vector of outputs from the ENN classifier $\mathbf{m} = (m(\{\omega_1\}), \dots, m(\{\omega_M\}), m(\Omega))^T$ is finally obtained as

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

and

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

The original work [20] presents the gradient calculation of the ENN when its output masses are used to define a loss function. Appendix A provides the gradient calculation of the ENN as a neural-network layer that receives the gradients w.r.t masses from another layer.

1.5 Conclusion

DST is a powerful tool to represent and combine different uncertain information by belief functions, including imperfect, imprecise, and incomplete data. In the thesis, Dempster's rule is used to combine information coming from different sources. The fusion of heterogeneous classifiers with different degrees of granularity is easily performed using the vacuous extension operation. DST also provides a flexible framework to make precise and imprecise decisions. In addition, the DST-based ENN classifier provides a way to convert features into mass functions, which will be used in the next three chapters to build evidential deep neural networks.

Chapter 2

Deep neural networks

Deep learning [70], as a subset of machine learning, provides a very powerful framework for feature representations. By adding more layers and more units within a layer, a deep neural network (DNN) can represent information of increasing complexity. Most perception tasks that consist of extracting high-level information from raw data as feature representations, even those considered to be difficult for humans, can be accomplished via DNNs, given sufficiently large models and sufficiently large datasets of labeled learning examples. In the framework of DNN, several models have been developed for feature representation, such as convolutional neural networks (CNNs) [48, 113], recurrent neural networks [84, 85], fully convolutional network (FCN) [5, 35, 77], graph neural networks [111, 112], and deep autoencoders [53, 78].

Even though DNNs have the powerful and flexible capacity of feature representations, they still face the problem of data uncertainty. The problems mainly derive from the fact that DNNs work within the probabilistic framework. To overcome this limitation, the goal of our study in this thesis is to combine the frameworks of Dempster-Shafer theory (DST) and DNNs to better deal with data uncertainty. In detail, we use the feature representations from the backbone of a DNN as the inputs of the DST-based evidential neural network classifier recalled in Section 1.4 for decision-making with uncertainty. The term *backbone* refers to the part of the feature extractor in a deep neural network. Considering several categories of deep neural networks have been developed in the last decade, we demonstrate the feasibility of the proposed idea by combining DST with two widely-used categories: CNN and FCN.

This chapter introduces the basic information of CNNs and FCNs. We begin by describing CNNs that are used for feature representations in Section 2.1, including convolution and pooling operations, input data types of CNNs, and the recent variants of CNNs. In Section 2.2, we present the techniques of fully convolutional networks (FCN) for pixel-wise feature representations, consisting of the overall architecture of a common FCN, upsample operations, and some variants of FCN.

2.1 Convolution neural network

Convolutional neural networks [37, 71], also known as convolutional networks, are neural networks that use convolution in place of general matrix multiplication in at least one of their layers. Convolution is a specialized kind of mathematical linear operation. In Section 2.1.1, we first describe what convolution is and explain the motivation behind the use of convolution in a network. In Section 2.1.2, we introduce an operation called *pooling*, which almost all convolutional networks employ. The most common CNNs consist of layers with convolution and pooling operations, called *convolutional and pooling layers*, respectively. A combination of stacked convolutional and pooling layers is defined as a *stage* that converts its input data into an intermediate representation, working as a feature extractor. In general, the *backbone* of a CNN is composed of several stacked stages that process raw data and repeatedly convert them into higher-level feature tensors, which will be considered as feature representations and converted into mass function in Chapters 3-5. In Section 2.1.3, we also show how convolution may be applied to many kinds of data, with different numbers of dimensions. Finally, we introduce several efficient and widely-used modern variants of CNNs in Section 2.1.4.

2.1.1 Convolution operation and its motivation

Let $\mathbf{z} = (z^1, \dots, z^D)$ be an input consisting of D input tensors or *input channels* z^i ($i = 1, \dots, D$) with size $H \times W$. A convolutional layer consists of several convolution kernels that perform convolution operations to extract feature maps from \mathbf{z} . A convolution kernel is a small matrix used to apply a convolution operation to each input tensor by sliding over the tensor, performing an element-wise multiplication with the part of the input tensor where the kernel is currently on, summing up the multiplied results into a single value, and then adding the bias of the kernel to the summed value. Figure 2.1 presents an example of convolution operation. Thus, the processes in a convolutional layer, consisting of e convolution kernels with size $a \times b$, are expressed as

$$c^j = f(b^j + \sum_i w^{i,j} * z^i), \quad (2.1)$$

where $w^{i,j}$ is the convolution kernel between the i -th input tensor and the j -th output tensor; b^j is the bias of kernel $w^{i,j}$; $*$ denotes the convolution operation; z^i is the i -th input tensor with size $H \times W$, $i = 1, \dots, D$; c^j is the j -th output tensor, with size $\frac{H-a+1}{r} \times \frac{W-b+1}{r}$, $j = 1, \dots, e$; r is the stride with which the kernel slides over input tensor z^i ; $f(x)$ is the activation function, such as the rectified linear unit $\text{ReLU}(x) = \max(0, x)$ [65].

The main motivation of the convolution operation in a CNN is to provide a means

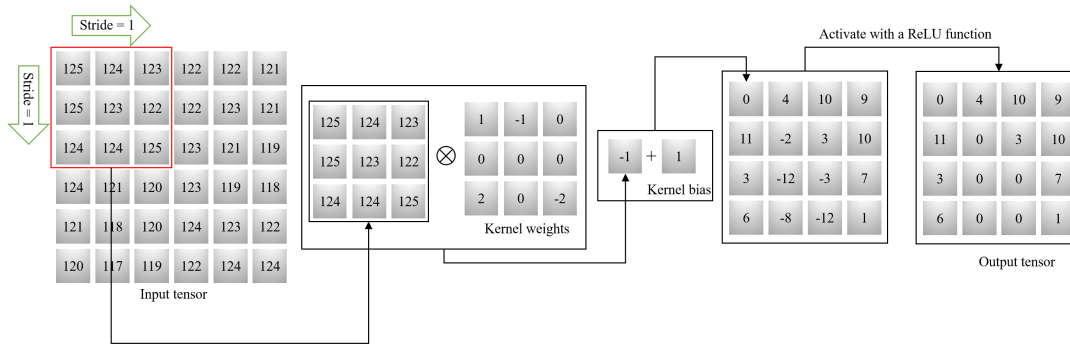


Figure 2.1: An example of two-dimensional convolution operation with stride=1. The red box on the input tensor indicates the part of the tensor where the kernel is currently on. The black box slides over the input tensor from upper-left to bottom-right with a one-step stride. We restrict the output to only positions where the kernel lies entirely within in the input tensor, called “valid” convolution. The black arrows indicate how the upper-left element of the output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor. Here, \otimes is element-wise multiplication.

for working with inputs of variable size. In general, convolution leverages three interesting properties that can help improve a machine learning system: sparse interactions, parameter sharing, and equivariant representations. Details of the three properties can be found in [37].

Sparse interaction, also referring to sparse connectivity or sparse weights, means that a neural-network layer does not need to connect every output unit with every input unit. In a traditional neural network layer, we use matrix multiplication by a matrix of parameters to describe the interaction between each input unit and each output unit. However, in a convolutional layer, sparse interaction is accomplished by using kernels smaller than the input. For instance, given m inputs and n outputs, a matrix multiplication requires $m \times n$ parameters with $O(m \times n)$ arithmetical operations per example. However, a convolution kernel with $k \ll m$ weights only requires $k \times n$ parameters with $O(k \times n)$ arithmetical operations. Thus, the kernel-based sparse interaction needs to store fewer parameters, which both reduces the memory requirements of a neural network and improves its statistical efficiency.

Parameter sharing means using the same parameter for more than one function in a neural network. In a traditional neural-network layer, each element of a weight matrix is used exactly once when computing the outputs. However, in a convolutional layer, each weight in a kernel is used at every position of an input tensor, where the kernel is currently on. This makes it possible to only train one set, rather than a separate set of parameters for every location of an input tensor.

The particular form of parameter sharing allows a convolution operation to have a property, called *equivariance to translation*. A function is equivariant means that if the input changes, the output changes in the same way. For example, a function $f(x)$ is equivariant to a function $g(x)$ if $f(g(x)) = g(f(x))$. In the case of convolution,

let g be any function that translates the input, i.e., shifting, then the convolution function is equivariant to g . For instance, I is a function giving brightness at integer coordinates (x, y) , and g is a function that shifts every pixel of I one unit to the right, such that $g(x, y) = I(x - 1, y)$. If we apply this translation to I , then perform a convolution operation, the result is the same as the condition where we first perform the convolution operation to $I' = g(I)$ and then apply the translation g to the output. This property improves the robustness of the object representation from a convolution operation.

2.1.2 Pooling operation

A typical convolution stage in a CNN consists of two parts. One is composed of convolutional layers that perform convolution operations in parallel to produce a set of linear activations. The other part is a pooling layer that further modifies the outputs of a convolutional layer in the stage by an operation called *pooling*. A pooling operation sub-samples each feature tensor c^j from a convolutional layer by computing some statistics of feature values within non-overlapping $S \times S$ windows of the tensor. We describe three types of pooling operations that will be used in the rest of the thesis: max-, mean-, and stochastic-pooling.

In the case of max-pooling, the statistic is the maximum and the outputs of the pooling layer is composed of the feature maps sub-sampled by factor S . For example, a feature tensor c^j with size $\frac{h-a+1}{r} \times \frac{w-b+1}{r}$ in (2.1) is downsized to $\frac{h-a+1}{2r} \times \frac{w-b+1}{2r}$ by a max-pooling operation with a 2×2 non-overlapping window. Figure 2.2 shows an example of the max-pooling operation applied to the output tensor in Figure 2.1.

The case of mean-pooling is similar to max-pooling except that the statistic is the mean. Figure 2.2 shows an example of mean-pooling operation to the output tensor in Figure 2.1. With a mean-pooling operation with a 2×2 non-overlapping window, feature tensor c^j with size $\frac{h-a+1}{r} \times \frac{w-b+1}{r}$ is also downsized to $\frac{h-a+1}{2r} \times \frac{w-b+1}{2r}$.

A stochastic-pooling operation randomly picks the activation within each pooling region according to a multinomial distribution, given by the activities within the pooling region [153]. In Figure 2.2, we perform the stochastic-pooling operation, in which the activities in each region are normalized by their sum to generate a pseudo-probability distribution for randomly picking.

After a pooling operation, a stage converts its input data into feature tensors that are an intermediate representation. The backbone of a CNN model composed of several stacked stages outputs high-dimensional and structured feature tensors. Typically, the tensors from the final stage, called *feature maps*, are used for predicting a class label for a classification task. Therefore, the final output of the stacked stages in a CNN can be considered as a feature representation of the raw data. In the rest of the thesis, these high-level features are used as inputs to a DS layer capable of set-valued classification as will be described in Chapter 3.

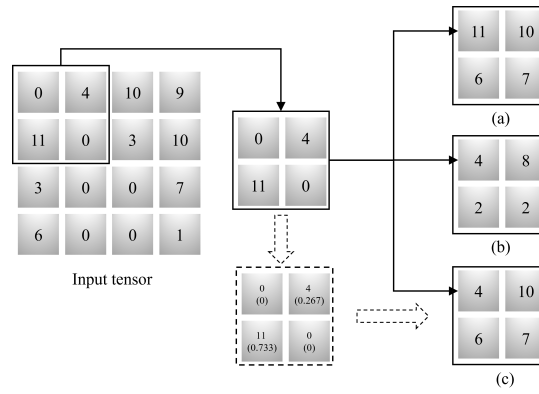


Figure 2.2: Examples of two-dimensional pooling with a 2×2 non-overlapping window: max-pooling (a), mean-pooling (b), and stochastic-pooling (c). The solid-line arrows indicate how the upper-left element of the output tensor is formed by applying one of the pooling operations to the corresponding upper-left region of the input tensor. The dotted-line arrows are the processes to generate a pseudo-probability distribution and randomly pick activities according to the distribution. The number in brackets is the pseudo-probability distribution.

2.1.3 Data types

The input data of a CNN model usually consists of several channels, each channel being the observation of a different quantity at some point in space or time. Table 2.1 summarizes the majority of types of input data with different dimensionalities and number of channels.

2.1.4 Efficient modern variants of convolutional neural networks

In the last years, more and more ambitious and advanced approaches of CNNs have been proposed to solve classification problems. This section briefly recalls several advanced and widely-used modern variants of CNNs, which will be used in the rest of the thesis.

Network in network (NIN) [74]. The main idea of a NIN is to replace linear convolution kernels and a nonlinear activation function in each convolution layer with a micro neural network for nonlinear feature extraction. A common convolution layer uses the combination of linear convolution kernels and a nonlinear activation function to extract features from the input tensors, as described in Section 2.1.1 and shown in Figure 2.3a. In each convolutional layer of a NIN, micro networks, e.g., multilayer perceptrons, are built to perform an operation similar to convolution, where the output feature tensors are obtained by sliding the micro networks over the inputs. The micro neural networks can be considered as a neural-network layer, called *NIN layer*. Figure 2.3b compares the structures of NIN and convolutional layers.

Table 2.1: Examples of different formats of data that can be used with convolutional networks [37].

Types	Single channel	Multi-channel
1D	Audio waveform: The axis we convolve over corresponds to time. We discretize time and measure the amplitude of the waveform once per time step.	Skeleton animation data: Animations of 3D computer-rendered characters are generated by altering
2D	Audio data that has been pre-processed with a Fourier transform: We can transform the audio waveform into a 2D tensor with different rows corresponding to different frequencies and different columns corresponding to different points in time. Using convolution in the time makes the model equivariant to shifts in time. Using convolution across the frequency axis makes the model equivariant to frequency, so that the same melody played in a different octave produces the same representation but at a different height in the network’s output.	Color image data: One channel contains the red pixels, one the green pixels, and one the blue pixels. The convolution kernel moves over both the horizontal and vertical axes of the image, conferring translation equivariance in both directions.
3D	Volumetric data: A common source of this kind of data is medical imaging technology, such as CT scans.	Color video data: One axis corresponds to time, one to the height of the video frame, and one to the width of the video frame.

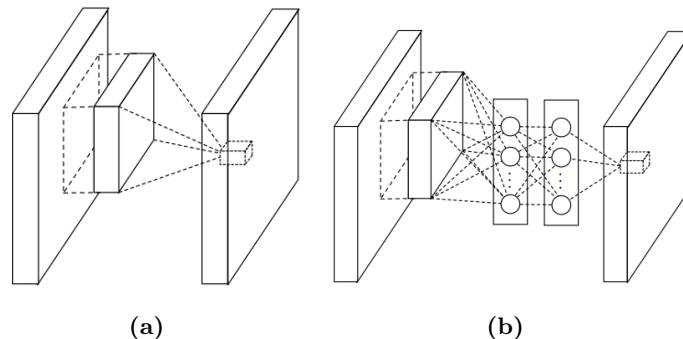


Figure 2.3: Comparison of traditional convolution layer (a) and NIN layer (b) [74].

Another interesting operation in the NIN is *Global Average Pooling (GAP)*. For classification, a common CNN vectorizes the feature maps of the last convolutional stage and feeds them into fully connected layers followed by a softmax layer for decision-making. This structure bridges the convolutional stages with traditional neural networks. Unfortunately, the fully connected layers are prone to overfitting because of their matrix-multiplication connection between the inputs and outputs. To solve the problem, a GAP operation takes the average of each feature map, and the resulting vector is fed directly into a softmax layer. A GAP operation does not introduce any parameters, which avoids overfitting and reduces the required memory. Besides, the relationship between feature maps and category confidences is easier to be interpreted since the operation directly enforces the correspondences between feature maps and categories.

FitNet [106]. In practice, depth tends to improve network performances since deeper networks are more non-linear. To achieve better performance than a ready-trained network in a classification task, the study of FitNet takes advantage of depth using a knowledge distillation approach. In this approach, the ready-trained network is called the “teacher” network and its weights are learned from the learning set of the task. The approach aims to train a deeper but thinner network with higher performance, called the “student” network, using the teacher one and learning set. “Deep” and “thin” mean a network has more layers but each layer has fewer kernels or units. In the approach, given a sample from the learning set, the student network is trained to imitate the intermediate and final feature representations of the teacher network using a gradient descent method. Generally, each stage in the student network introduces more hidden layers and reduces the units in each layer to imitate and predict the feature outputs of the corresponding stage in the teacher network, as shown in Figure 2.4. This allows one to train deeper students that can generalize and interfere better. Table 2.2 displays several thin deep CNNs trained by the approach of knowledge-distillation and achieving very good performances in image and speech recognition [106].

ResNet [48]. ResNet, short for Residual Networks, is a neural network with skip connections used as a backbone for feature extraction. Skip connections explicitly copy features from earlier layers into later layers in forward propagation and allow gradients to flow easily from later layer to earlier layer in back-propagation, even connecting the lowest layer and the top layer. Figure 2.5 is an instance of skip connection. This prevents neural networks from the problem of vanishing gradients and helps users to build really very deep networks from the very start, rather than at the beginning of another already-trained network, like FitNet [106]. Table 2.3 describes the widely-used ResNet-34, -50, and -101.

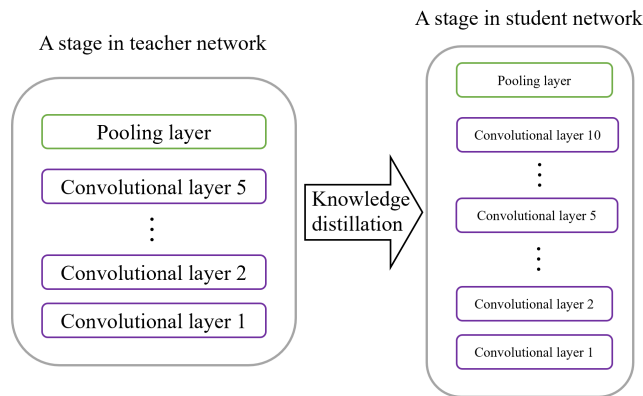


Figure 2.4: Depth and width comparison of teacher and student CNN stages in knowledge distillation. The width of the pink boxes indicates the number of kernels in each convolutional layer. The student stage is thinner than the teacher one since the former has fewer kernels than the latter. However, the student stage consists of more convolution layers than the teacher one, i.e., the student network is deeper than the teacher one.

Table 2.2: Performance-Efficiency FitNet architectures [106].

FitNet-1	FitNet-2	FitNet-3	FitNet-4
3×3 Conv. 16 <i>ReLU</i>	3×3 Conv. 16 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>
3×3 Conv. 16 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 48 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>
3×3 Conv. 16 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 64 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>
2×2 max-pool	2×2 max-pool	3×3 Conv. 64 <i>ReLU</i>	3×3 Conv. 48 <i>ReLU</i>
		2×2 max-pool	3×3 Conv. 48 <i>ReLU</i>
			2×2 max-pool
3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 48 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 64 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
2×2 max-pool	2×2 max-pool	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
		2×2 max-pool	3×3 Conv. 80 <i>ReLU</i>
			2×2 max-pool
3×3 Conv. 48 <i>ReLU</i>	3×3 Conv. 96 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
3×3 Conv. 48 <i>ReLU</i>	3×3 Conv. 96 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
3×3 Conv. 64 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
8×8 max-pool	8×8 max-pool	8×8 max-pool	3×3 Conv. 128 <i>ReLU</i>
			3×3 Conv. 128 <i>ReLU</i>
			8×8 max-pool

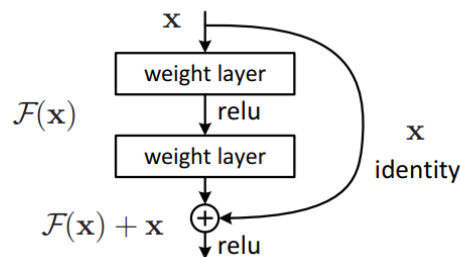


Figure 2.5: Example of skip connections [48]. A copied feature \mathbf{x} from a earlier layer is concatenated with the outputs of the “weight layer” as a new feature representation, while the gradient w.r.t \mathbf{x} can be directly back-propagated to the earlier layer.

Table 2.3: Performance-Efficiency ResNet backbones [48].

Layer name	ResNet-34	ResNet-50	ResNet-101
Stage 1	$7 \times 7 \times 64$, stride 2		
	3×3 max-pooling, stride 2		
Stage 2	$\begin{bmatrix} 3 \times 3 & 64 \\ 3 \times 3 & 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1 & 64 \\ 3 \times 3 & 64 \\ 1 \times 1 & 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1 & 64 \\ 3 \times 3 & 64 \\ 1 \times 1 & 256 \end{bmatrix} \times 3$
Stage 3	$\begin{bmatrix} 3 \times 3 & 64 \\ 3 \times 3 & 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 & 128 \\ 3 \times 3 & 128 \\ 1 \times 1 & 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 & 128 \\ 3 \times 3 & 128 \\ 1 \times 1 & 512 \end{bmatrix} \times 4$
Stage 4	$\begin{bmatrix} 3 \times 3 & 64 \\ 3 \times 3 & 64 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 & 256 \\ 3 \times 3 & 256 \\ 1 \times 1 & 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 & 256 \\ 3 \times 3 & 256 \\ 1 \times 1 & 1024 \end{bmatrix} \times 23$
Stage 5	$\begin{bmatrix} 3 \times 3 & 64 \\ 3 \times 3 & 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1 & 512 \\ 3 \times 3 & 512 \\ 1 \times 1 & 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1 & 512 \\ 3 \times 3 & 512 \\ 1 \times 1 & 2048 \end{bmatrix} \times 3$

Vision transformer (ViT) [27]. Figure 2.6a presents the architecture of an ViT. Compared to a traditional CNN directly using a full sample for classification, a ViT first divides an input sample into a grid of square patches. Each patch is flattened into a single vector by concatenating its channels of all elements and then linearly projected to the desired input dimension. As an alternative to flattening the patches, called *CNN-based ViT*, these patches can be imported into stacked CNN stages to form feature vectors. After dividing the sample, the ViT is agnostic to the position information about these patch vectors. Thus, learnable position embeddings are linearly added to each vector, which allows a ViT to learn about the relative or absolute positions of the patches. These embedded patch vectors are then sequentially imported into a module with stacked transformer encoders, such as L transformer encoders in Figure 2.6b. Each encoder consists of alternating layers of self-attention and multi-layer perceptron (MLP). *Self-attention* of an embedded patch vector is defined as its relationship to every other vector. Feeding the embedded patch vectors sequentially, a self-attention layer computes their self-attentions as introduced in [136]. These self-attentions are then fed into a multi-layer perceptron layer to handle their dimension. In addition, LayerNorm [144] is applied before every layer, and skip connection [48] after every layer, as shown in Figure 2.6b. A LayerNorm layer normalizes the outputs of its previous layer for each given sample in a batch independently. The two techniques can improve the training and overall performance. The self-attention outputs of the final transformer encoder are concatenated, and the concatenated vector can be considered as the feature representation of the input sample for classification.

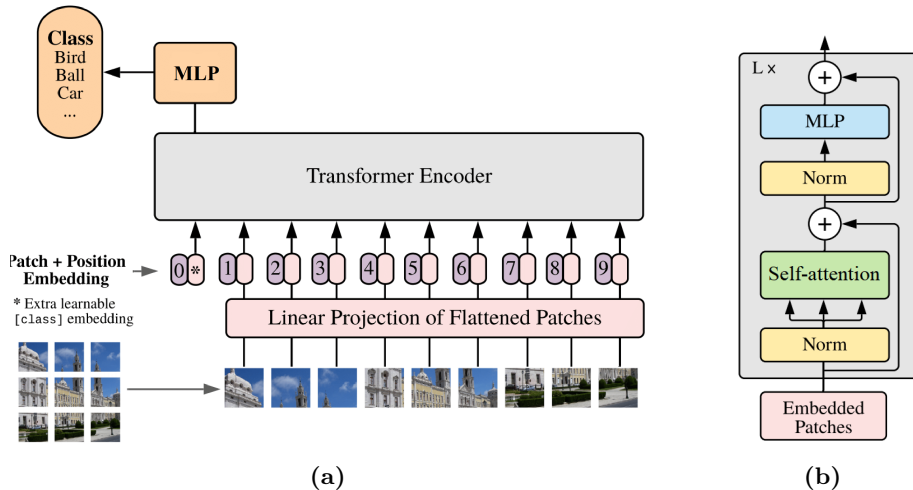


Figure 2.6: Architecture of a ViT [27]: overview (a) and transformer encoder module (b).

2.2 Fully convolutional networks

CNNs have very good performances in classification tasks thanks to their vectorized and high-dimensional feature representations. The natural next step in the progression from coarse to fine inference is to predict at every element of the input data. The prediction at every pixel of an input image, known as *semantic segmentation* and *pixel-wise classification*, is 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*. The mask constitutes a simplified representation, more meaningful and easier to analyze than the original image. Semantic segmentation has been widely applied to advanced driver assistance systems [31, 75], human-machine interaction [66, 124], medical imaging [125, 150], and so on.

Many deep learning-based approaches have been proposed for semantic segmentation [33, 32, 44, 47, 77, 94, 101], in which each pixel is predicted with the class of its enclosing object or region. One of the most successful approaches is fully convolutional networks (FCNs). This section first recalls the original architecture of an FCN model in Section 2.2.1, followed by the introduction of upsample operations in Section 2.2.2. Finally, we describe several efficient variants of FCNs in Section 2.2.3 that is used in the rest of the thesis.

2.2.1 Overall architecture of fully convolutional network

The main idea of FCN [77] comes from the architecture of CNN. In a CNN classifier, as described in Section 2.1, several fully connected layers are used to handle the number of channels in feature maps before probability predictions in a softmax layer, such as from 4096 to 1000 in the top example of Figure 2.7. Thus, a CNN classifier has to output feature maps with a fixed size and then vectorize them since its fully

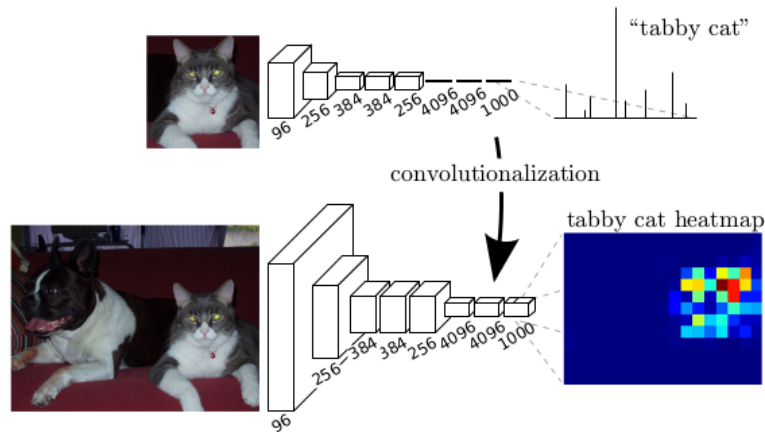


Figure 2.7: Transforming fully connected layers into convolution layers to output a heatmap [77].

connected layers have fixed neurons and require the inputs with a fixed dimension. The vectorizing operation throws away spatial coordinates of feature maps that are important for semantic segmentation. In [77], these fully connected layers are replaced by convolutional layers with 1×1 kernels. A 1×1 kernel only has a single weight for each channel in its inputs and performs the convolution operations over the input tensors pixel by pixel. Thus, the inputs and outputs of a 1×1 convolutional layer have the same size but different numbers of channels. Such transformation allows to handle the number of channels and retains the spatial information in feature maps without vectorizing operations, as illustrated in the bottom example of Figure 2.7. After that, feature maps are converted into a *heatmap* with the same size as the input image. A heatmap attempts to determine all the important regions in an image that the neural network pays attention to while performing semantic segmentation, such as the cat in Figure 2.7. Therefore, the heatmap can be considered as the *pixel-wise feature representation* of the image. The process of converting feature maps into a heatmap refers to *upsampling*.

Following the idea, Long et al. [77] propose the architecture of FCNs that owe their name to their architecture with only 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 classification. In the encoder-decoder architecture, an input image is encoded by several stacked convolutional and pooling layers and then decoded by one or more upsampling layers, where the encoder part also refers to as the *backbone* of the FCN. The softmax layer then classifies each pixel in the input image to one of the classes based on the outputs of the encoder-decoder architecture. Therefore, the outputs of encoder-decoder architecture, called the *pixel-wise feature maps* or *heatmaps*, are considered as the feature representations of the input image. In the thesis, these feature maps are used

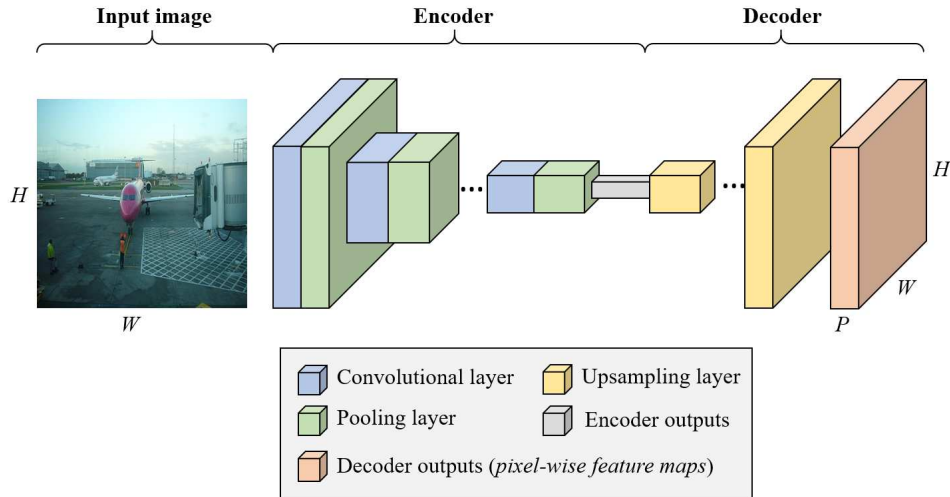


Figure 2.8: An illustration of the encoder-decoder architecture. An encoder downsizes its input by convolution and pooling operations. The outputs of the encoder, as the sparse feature maps, are imported into a decoder. A decoder upsamples and densifies its inputs by performing the reverse operation of convolution and pooling. The final decoder outputs are the pixel-wise feature maps.

as input to a DS layer allowing for set-valued semantic segmentation in Chapter 4.

To understand the feature representation of FCNs, we consider the encoder-decoder architecture illustrated in Figure 2.8. Each convolutional layer in the encoder part performs convolutions in its input to produce a set of feature maps, as mentioned in Section 2.1.1. A pooling layer follows the convolutional layer to sub-sample feature maps by computing some statistics of feature values within non-overlapping $S \times S$ windows (see Section 2.1.2). In the rest of this thesis, max-pooling is used and the statistic is the maximum. Although the convolution and pooling operations in the encoder part help 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 encoder output, and each upsampling layer performs an upsampling operation to its input. The widely-used types of upsampling operations will be introduced in Section 2.2.2. The final upsampling layer outputs feature maps with the same size as the input images, as the output of an encoder-decoder architecture.

2.2.2 Upsampling operations

One of the key operations in an FCN is upsampling, which maps the spatial features from the backbone of the FCN into dense pixel-wise feature maps. This section describes five widely-used types of upsampling operations.

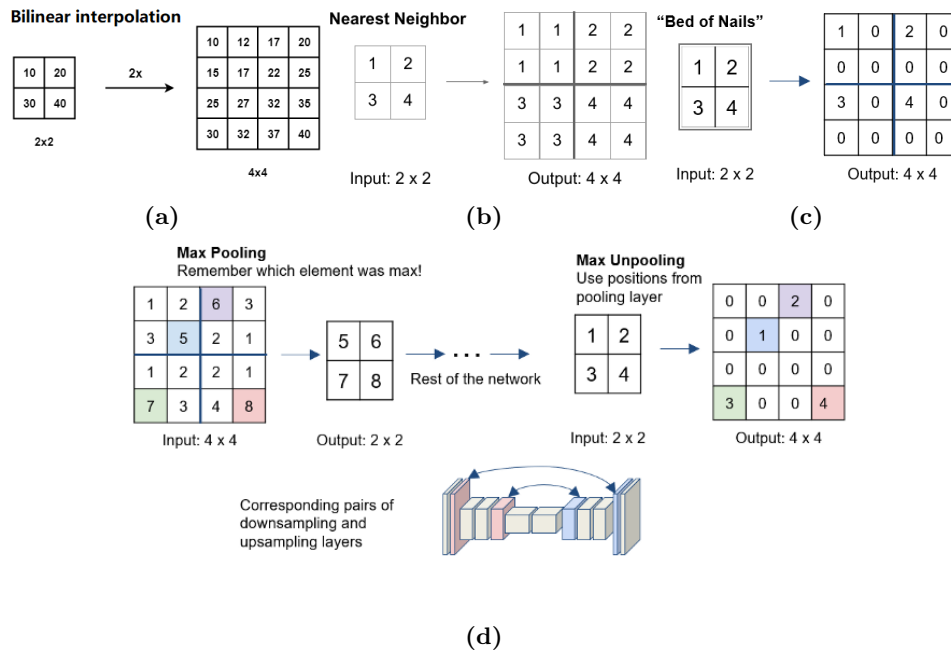


Figure 2.9: Examples of interpolation upsampling (a), nearest neighbors upsampling (b), bed of nails upsampling (c), and Max-unpooling (d).

Interpolation upsampling. Interpolation is the simplest way to connect coarse features from a convolutional layer to dense heatmaps. For instance, simple bilinear interpolation computes each output from the four nearest inputs by a linear map that depends only on the relative positions of the input and output cells, such as the example in Figure 2.9a.

Nearest neighbors upsampling. As the name suggests, we take each input value and copy it to the K -nearest neighbors where K depends on the expected output, like $k = 2$ in Figure 2.9b.

Bed-of-nails upsampling. In bed-of-nails upsampling, we copy the value of each input at the corresponding position in the output image and filling zeros in the remaining positions, such as the one in Figure 2.9c.

Max-unpooling. Max-pooling in a CNN encoder takes the maximum among all the values in the kernel. To perform max-unpooling, first, the index of the maximum value is saved for every max-pooling layer during the encoding step. The saved index is then used during the decoding step where the input pixel is mapped to the saved index, filling zeros everywhere else. An instance is shown in Figure 2.9d.

Transposed convolution [95]. A transposed convolution operation densifies its inputs of sparse feature maps through a convolution-like operation with a learned kernel. Contrary to the convolution operation, which connects multiple inputs within

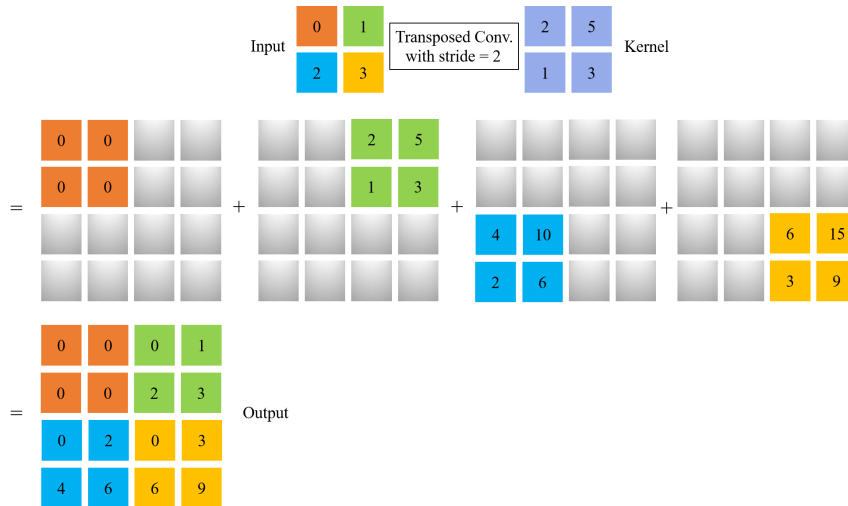


Figure 2.10: An instance of transposed convolution operation.

a kernel to a single activation, a transposed convolution operation associates a single input in a feature map to multiple outputs. Figure 2.10 is an example of the transposed convolution operation. Thus, the outputs of a transposed convolution operation are enlarged and dense feature maps. The processes of a transposed convolution operation can also be summarized as Eq. (2.1).

2.2.3 Variants of fully convolution networks

In recent years, many FCN-based models have been proposed to solve the problems of semantic segmentation. This section describes several widely-used models, which will be used in the rest of this thesis.

FCN-32s, -16s, -8s [77]. In an FCN-32s model, as shown in Figure 2.11, several stacked CNN stages extract feature maps from an input image, followed by one or more 1×1 convolution layers as described in Section 2.2.1. An upsampling layer then bilinearly upsample the maps to pixel-dense heatmaps that are used for decision-making in a softmax layer. Compared to the FCN-32s model, the FCN-16s and FCN-8s models combine sparse and high-layer information with dense and low-layer information during upsampling. We take the FCN-16s model in Figure 2.11 as an example. Its upsampling layer first doubles the density of the feature maps from the Pool 4 layer using bilinear interpolation upsampling, and then the upsampled maps are added to the maps from Pool 3 that also provide useful information for segmentation. The added maps are then bilinearly upsampled to pixel-dense heatmaps for segmentation. Compared to the FCN-16s model, the FCN-8s acquires additional feature maps from Pool 2 to provide further precision.

U-net [35, 107]. U-net was first developed for biomedical image segmentation. Its architecture looks like a ‘U’ that justifies its name, as shown in Figure 2.12. This

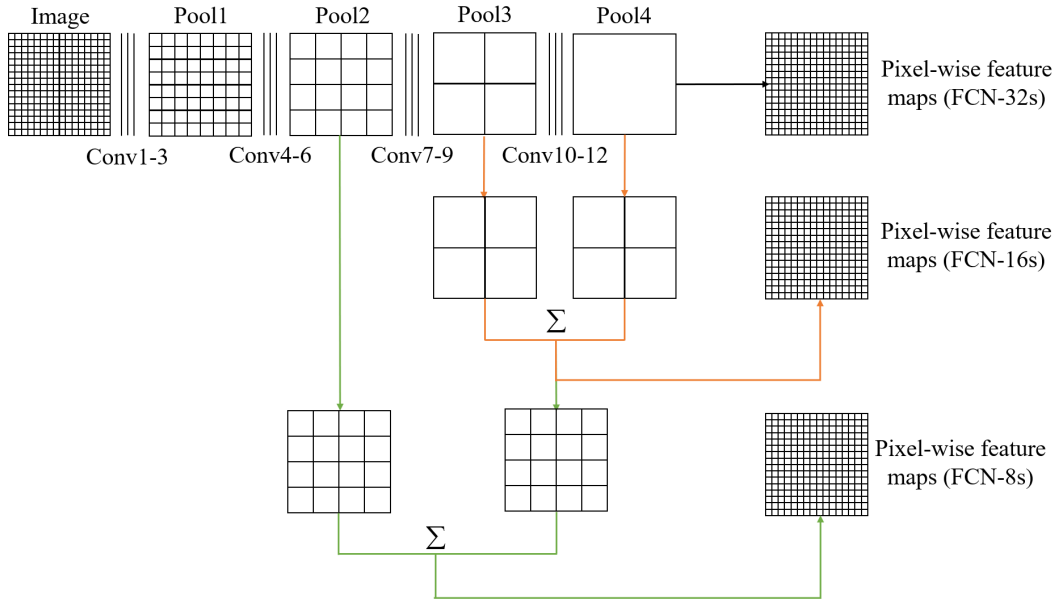


Figure 2.11: Illustration of the FCN-32s, FCN-16s, and FCN-8s architectures. Pooling layers are represented as grids that show relatively sparse information. Intermediate convolution layers are omitted. Black arrow: the upsampling layer in FCN-32s directly upsamples the outputs of Pool 4 to pixel-wise feature maps; orange arrows: the upsampling 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.

architecture consists of three parts: encoder, bottom, and decoder. The encoder part is made up of several convolutional stages for feature extraction. The bottom part then handles the number of the feature channels from the encoder using three 1×1 convolutional layers. The decoder part upsamples the features from the bottom part. The decoder has several stages and each stage consists of two convolutional layers followed by a transposed convolution layer. In addition, each stage in the decoder corresponds to one stage in the encoder, such as the pairs indicated by the gray arrows in Figure 2.12. A decoder stage concatenates its inputs with the outputs of its corresponding encoder stage, such as the gray arrows shown in Figure 2.12. This operation provides more useful information for segmentation. After that, the outputs of the final decoder stage are considered as the pixel-wise feature representations for decision-making in a softmax layer.

SegNet [1]. Figure 2.13 illustrates the overall architecture of a SegNet model. The SegNet architecture has several upsampling layers to upsample the sparse feature maps from the end of the encoder part. The upsampling operations can be bilinear interpolation or transposed convolution. Similar to U-net, the upsampling layers in a SegNet model are symmetric to the convolution stages, such as the four shown in Figure 2.13. The outputs of each upsampling layer are added to the outputs of

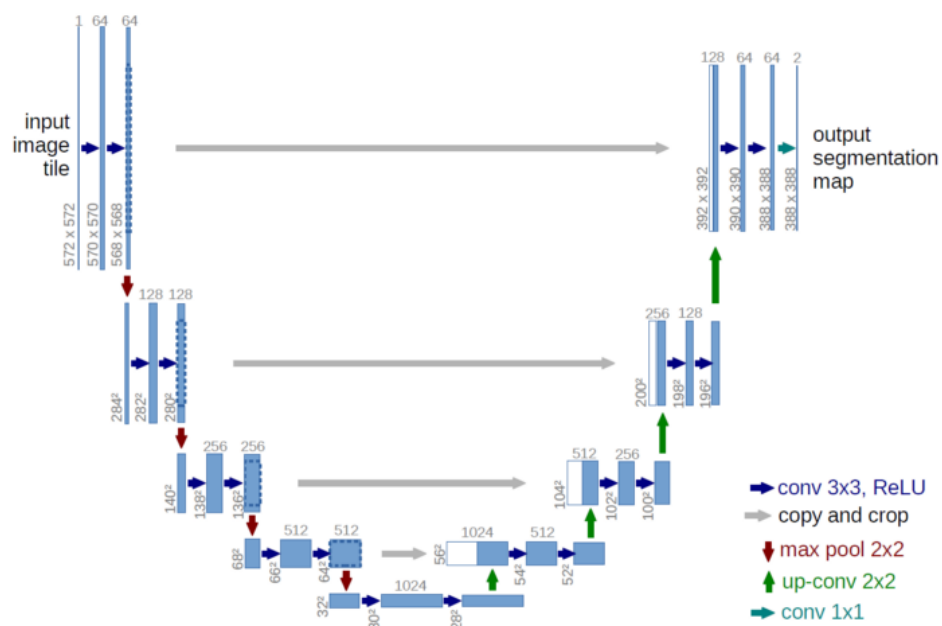


Figure 2.12: Illustration of a U-net architecture [107]. Each blue box corresponds to feature maps. The number of channels in the feature maps is denoted at the top of the box. The width and height are provided at the lower-left edge of the box. White boxes represent copied feature maps, and gray arrows indicate the transmitting direction of the copied feature maps between the pairs of upsampling layers and convolution stages.

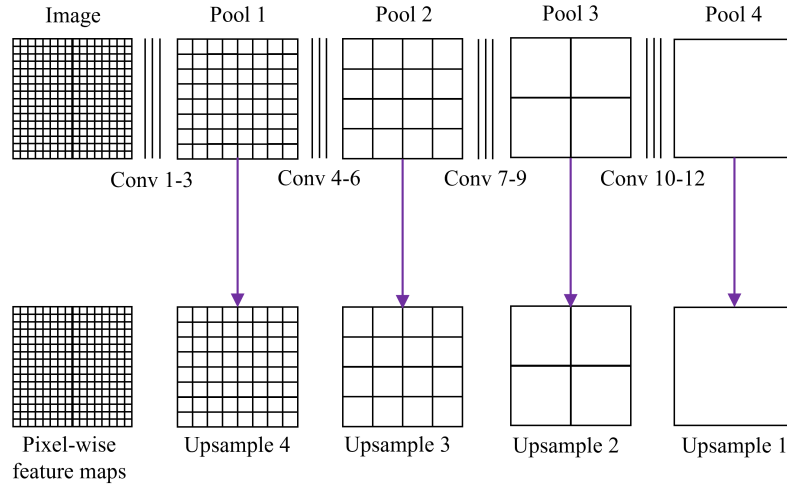


Figure 2.13: Illustration of the SegNet architecture. The architecture uses four deconvolutional layers to upsample the sparse feature maps from the end of the encoder part, as well as the feature maps from the corresponding pooling layers based on pooling indices (purple arrows).

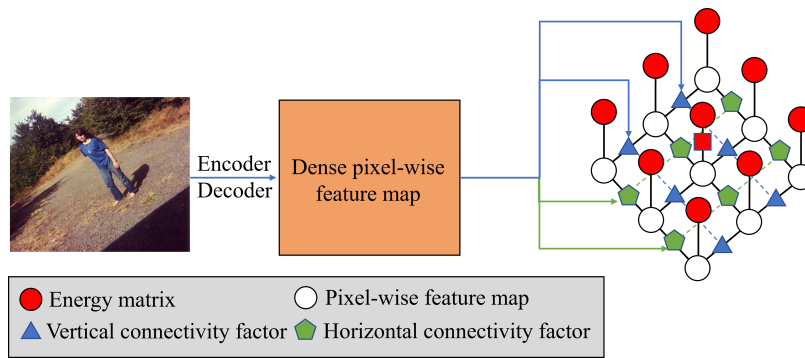


Figure 2.14: Illustration of a conditional random field.

the corresponding convolution stages, which can improve the accuracy of pixel-wise representations.

FCN-CRF (DeepLab v2) [5]. A common FCN-based model predicts the class of a pixel using its corresponding vector from the pixel-wise feature maps without considering “neighboring” pixels. However, the classes of the “neighboring” pixels are important factors to determine the class of a pixel. A conditional random field (CRF) can take context into account. In an FCN-CRF model, a CRF is added at the end of the last upsampling layer to generate an energy matrix of the pixel-wise feature maps, as illustrated in Figure 2.14. The value of each element in the matrix is computed using the corresponding vector in the pixel-wise feature maps and the vertical and horizontal factors w.r.t the “neighboring” vectors. The details of how to generate an energy matrix using a CRF can be found in [5, 6]. The energy matrix with “neighboring” information, instead of the pixel-wise feature maps, is used for pixel-wise segmentation, which further improves the segmentation performance.

2.3 Conclusion

Deep learning provides a powerful framework for feature representations in supervised learning. By adding more layers and more units within a layer, a DNN can represent information of increasing complexity. CNN and FCN are two successful cases of DNNs for feature representation in the field of pattern classification and semantic segmentation, respectively. CNNs use convolutions in place of general matrix multiplication in at least one of their layers. They are driving advances in classification tasks thanks to their vectorized and high-dimensional feature maps from CNN backbones. FCNs, which are built only from locally connected layers, provides an efficient way to generate pixel-wise feature representation for semantic segmentation.

Part II

Evidential deep neural networks

Chapter 3

Evidential convolutional neural network classifier

In this chapter, to deal with the data uncertainty in classification problems, we propose a new classifier based on DST and a convolutional neural network (CNN) allowing for set-valued classification. In this classifier, called *the evidential CNN classifier* [129], a backbone with convolutional and pooling layers first extracts high-dimensional features from input data. The features are then converted into mass functions and aggregated by Dempster’s rule in a Dempster-Shafer (DS) layer. Finally, a utility layer performs set-valued classification based on mass functions. We propose an end-to-end learning strategy for jointly updating the network parameters. Additionally, an approach for selecting partial multi-class acts is proposed. Experiments on image recognition, signal processing, and semantic-relationship classification tasks demonstrate that the proposed combination of CNN, DS layer, and utility layer makes it possible to improve classification accuracy and to make cautious decisions by assigning confusing and ambiguous patterns to multi-class sets. In addition, the proposed classifier can reject outliers together with ambiguous patterns.

This chapter is organized as follows. The proposed classifier is introduced in Section 3.1. Section 3.2 then reports numerical experiments, which demonstrate the advantages of the proposed classifier. Finally, we conclude the chapter in Section 3.3.

3.1 Evidential CNN classifier

In this section, we describe the proposed classifier. Section 3.1.1 presents the overall architecture composed of a CNN backbone with several stacked stages for feature representation, a DS layer to construct mass functions, and a utility layer for decision-making. The learning strategy for the proposed classifier is exposed in Section 3.1.2. Finally, an approach for selecting partial multi-class acts is introduced in Section 3.1.3.

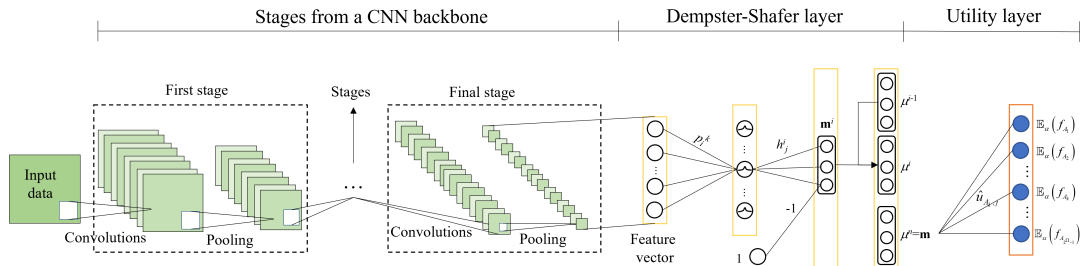


Figure 3.1: Architecture of an evidential CNN classifier.

3.1.1 Network architecture

The main idea of this work is to hybridize the ENN classifier presented in Section 1.4 and the CNN architecture recalled in Section 2.1 by “plugging” a DS layer followed by a utility layer at the output of a CNN backbone. The architecture of the proposed method, called the *evidential CNN classifier*, is illustrated in Figure 3.1. An evidential CNN classifier has the ability to perform set-valued classification and quantify the uncertainty about the class of the sample on $\Omega = \{\omega_1, \dots, \omega_M\}$ by a belief function. To distinguish the proposed classifier that converts features into belief functions, we named the common CNN classifier that transforms features into probabilities using a softmax layer as the *probabilistic CNN classifier*. Propagation of information through this evidential network can be described by the following three-step procedure:

Step 1: An input sample is propagated into a CNN backbone to extract latent features relevant for classification as introduced in Section 2.1. Thanks to this part, the evidential CNN classifier yields similar or even better performance for precise classification than does a probabilistic classifier with the same stages. This superiority will be demonstrated by performance comparisons between the evidential and probabilistic CNN classifiers in precise classification tasks (Section 3.2).

Step 2: The feature vector computed in Step 1 is fed into the DS layer, in which it is converted into mass functions aggregated by Dempster’s rule, as explained in Section 1.4. The output of the DS layer is an $(M + 1)$ mass vector

$$\mathbf{m} = (m(\{\omega_1\}), \dots, m(\{\omega_M\}), m(\Omega))^T,$$

which characterizes the classifier’s belief about the probability of the sample class and quantifies the uncertainty in the feature representation. The mass $m(\{\omega_i\})$ is a degree of belief that the sample belongs to class ω_i . The DS layer tends to allocate masses uniformly across classes when the feature representation contains confusing and ambiguous information. The additional degree of freedom $m(\Omega)$ makes it possible to quantify the lack of evidence and verify whether the model is well trained [130]. The advantages of the DS layer

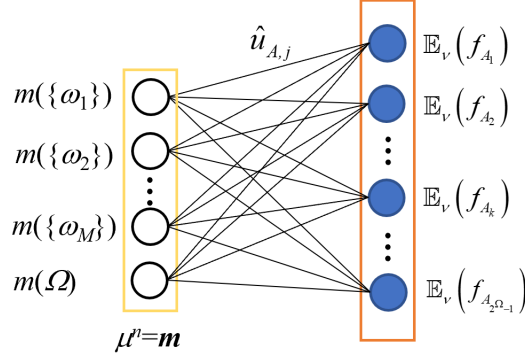


Figure 3.2: Architecture of the utility layer.

will be verified in the performance evaluation of set-valued classification using evidential CNN classifiers reported in Section 3.2.

Step 3: The output mass vector \mathbf{m} is used to compute the expected utilities of acts for performing set-valued classification, as introduced in Section 1.3.2. Thus, the output of the step is an expected-utility vector of size at most equal to $2^M - 1$ if all of the possible acts are considered. Similar to the DS layer, the procedure of assigning a sample to a set in \mathcal{F} using utility theory can be summarized as a layer of the neural network, called a *utility layer*, as shown in Figure 3.2. In this layer, the inputs and outputs are, respectively, the mass vector \mathbf{m} from the preceding DS layer and the 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 $\hat{u}_{A,j}$. As coefficient γ (1.24) describing the imprecision tolerance degree is fixed, the connection weights of the utility layer do not need to be updated during training. The capacity of a utility layer will be demonstrated by the performance comparison between the evidential and probabilistic CNN classifiers in set-valued classification and novelty detection tasks reported in Section 3.2.

3.1.2 Learning

The evidential CNN classifier can be trained by a stochastic gradient descent algorithm. Given a sample \mathbf{x} with class label ω_* , using the generalized Hurwicz criterion (1.21)¹, we define the prediction loss as

$$\mathcal{L}(\mathbb{E}_{m,\nu}, \omega_*) = - \sum_{k=1}^M y_k \log \mathbb{E}_{m,\nu}(f_{\omega_k}) + (1 - y_k) \log(1 - \mathbb{E}_{m,\nu}(f_{\omega_k})) \quad (3.1a)$$

¹The pignistic criterion (1.27) can also be used for set-valued prediction and will be exposed in Chapter 4.

Table 3.1: Examples of DS layer outputs

Examples	Outputs of a DS layer			
	$m(\{\omega_1\})$	$m(\{\omega_2\})$	$m(\{\omega_3\})$	$m(\Omega)$
#1	0.70	0.10	0.10	0.10
#2	0.97	0.01	0.01	0.01
#3	0.50	0.50	0	0
#4	0.40	0.40	0	0.2

Table 3.2: Example of utilities and losses

Examples	Expected utility			Loss ($\omega_* = \omega_1$)
	$\mathbb{E}_{m,1}(\{\omega_1\})$	$\mathbb{E}_{m,1}(\{\omega_2\})$	$\mathbb{E}_{m,1}(\{\omega_3\})$	
#1	0.70	0.10	0.10	0.303
#2	0.97	0.01	0.01	0.026
#3	0.50	0.50	0	0.602
#4	0.40	0.40	0	0.796

with

$$y_k = \begin{cases} 1 & \text{if } \omega_k = \omega_* \\ 0 & \text{if } \omega_k \neq \omega_* \end{cases}. \quad (3.1b)$$

The loss $\mathcal{L}(\mathbb{E}_{m,\nu}, \omega_*)$ is minimized when $\mathbb{E}_{m,\nu}(f_{\omega_k}) = 1$ for $\omega_k = \omega_*$ and $\mathbb{E}_{m,\nu}(f_{\omega_l}) = 0$ for $\omega_l \neq \omega_*$. The computation procedure of the loss is illustrated by Example 3.1.

Example 3.1 Table 3.1 shows several examples, whose utilities of single-valued assignments and losses are shown in Table 3.2. The extended utility matrix is shown in Table 1.1, and ν equals 1. We assume that $\Omega = \{\omega_1, \omega_2, \omega_3\}$ and $\omega_* = \omega_1$. Eq. (3.1) yields different losses given a set of DS layer outputs.

The derivatives of $\mathcal{L}(\mathbb{E}_{\nu}, \omega_*)$ of the error w.r.t \mathbf{m} in the utility layer are

$$\frac{\partial \mathcal{L}(\mathbb{E}_{m,\nu}, \omega_*)}{\partial m(\{\omega_k\})} = -\frac{y_k}{\mathbb{E}_{m,\nu}(f_{\omega_k})} \left[\hat{u}_{\{\omega_k\},k} + (1-\nu) \max_{i=1,\dots,M} \hat{u}_{\{\omega_k\},i} \right], \quad (3.2a)$$

$$\frac{\partial \mathcal{L}(\mathbb{E}_{m,\nu}, \omega_*)}{\partial m(\Omega)} = -\sum_{k=1}^M \frac{y_k}{\mathbb{E}_{m,\nu}(\{f_{\omega_k}\})} (1-\nu) \max_{i=1,\dots,M} \hat{u}_{\{\omega_k\},i}. \quad (3.2b)$$

The derivatives calculations of $\mathcal{L}(\mathbb{E}_{m,\nu}, \omega_*)$ w.r.t the parameters in a DS layer are shown in Appendix A. In the proposed classifier, the DS layer is connected to the pooling layer of the last convolutional stage, as shown in Figure 3.1. Thus, we can compute the derivatives of the error w.r.t. po^k as

$$\frac{\partial \mathcal{L}(\mathbb{E}_{m,\nu}, \omega_*)}{\partial po^k} = -2 \frac{\partial \mathcal{L}(\mathbb{E}_{m,\nu}, \omega_*)}{\partial s^i} (\eta^i)^2 s^i \sum_{i=1}^n (x_k - p_k^i), \quad k = 1, \dots, P, \quad (3.3)$$

where po^k is the k -th output map in the final pooling layer, which is a 1×1 tensor. Error propagation in the remaining stages is performed as in a probabilistic CNN.

3.1.3 Act selection

As explained in Section 1.3.2, the set of acts when considering multi-class assignment is $\mathcal{F} = \{f_A, A \in 2^\Omega \setminus \{\emptyset\}\}$, as instances can be assigned to any non-empty subset A of Ω . However, the cardinality of \mathcal{F} increases exponentially with the number of classes, which could preclude the application of this approach when the number M of classes is large.

In [130], we showed that a neural network with convolutional layers and a DS layer tends to assign samples to multi-class sets when candidate classes are similar, such as, e.g., “cat” and “dog”, or “horse” and “deer”. Thus, it may be advantageous to only consider partial multi-class acts assigning samples to subsets containing two or more similar classes.

We propose a strategy to determine similar classes in the frame of discernment and select partial multi-class acts from \mathcal{F} based on class similarity. Using the selected partial multi-class acts, rather than all acts in \mathcal{F} , we can reduce the compute cost in set-valued assignments. This strategy can be described as follows.

Step 1: A confusion matrix with only precise assignments is generated by a trained evidential CNN classifier using the training set. In the confusion matrix, each column represents the predicted sample distribution in one class, such as the example in Figure 3.3a.

Step 2: Each column in the confusion matrix is normalized using its total number. Each normalized column is regarded as the feature of its corresponding class. Figure 3.3b displays the normalized confusion matrix of the example in Figure 3.3a.

Step 3: The Euclidean distance between every two features is computed, and a dendrogram is generated by a hierarchical agglomerative clustering (HAC) algorithm [15, 115]. The distance between every two features represents the similarity of the two classes. The distance is close to 0 if two classes are similar. We draw the dendrogram of Example 3.2 in Figure 3.3c.

Step 4: The distance can be drawn versus the number of clusters based on the dendrogram, as shown in Figure 3.3d. A point of inflection in the curve can then be used to determine the threshold for cutting the dendrogram. We used the Calinski-Harabasz index (CHI) [4] to determine this point. The point of inflection is the one in the curve with the maximum CHI, as illustrated in Figure 3.3d of Example 3.2. The right of the point has a small number of highly similar classes. This can be explained by the nature of the HAC algorithm [15]. Very similar classes are consolidated first as the algorithm proceeds. Toward the end of the HAC run, we reach a stage where dissimilar classes are left to be merged but the distance between them is large; these classes are not similar and do not need to be clustered in the act-selection strategy.

Step 5: The distance corresponding to the inflection point is used as the threshold to cut the dendrogram. Similar patterns are the classes in the clustered groups with the distance lower than the threshold. Finally, we select the multi-class acts corresponding to similar classes.

Example 3.2 Figure 3.3 shows an example of act selection, in which a HAC algorithm with Ward linkage is used to generate a dendrogram. Figure 3.3d display a point of inflection whose CHI is 1.91 and corresponding distance is 0.927. The distance is used as the threshold of the Euclidean distance to cut the dendrogram. There are two pairs of similar patterns: $\{\omega_1, \omega_2\}$ and $\{\omega_3, \omega_4\}$. Thus, the selected partial multi-class acts are $f_{\{\omega_1, \omega_2\}}$ and $f_{\{\omega_3, \omega_4\}}$.

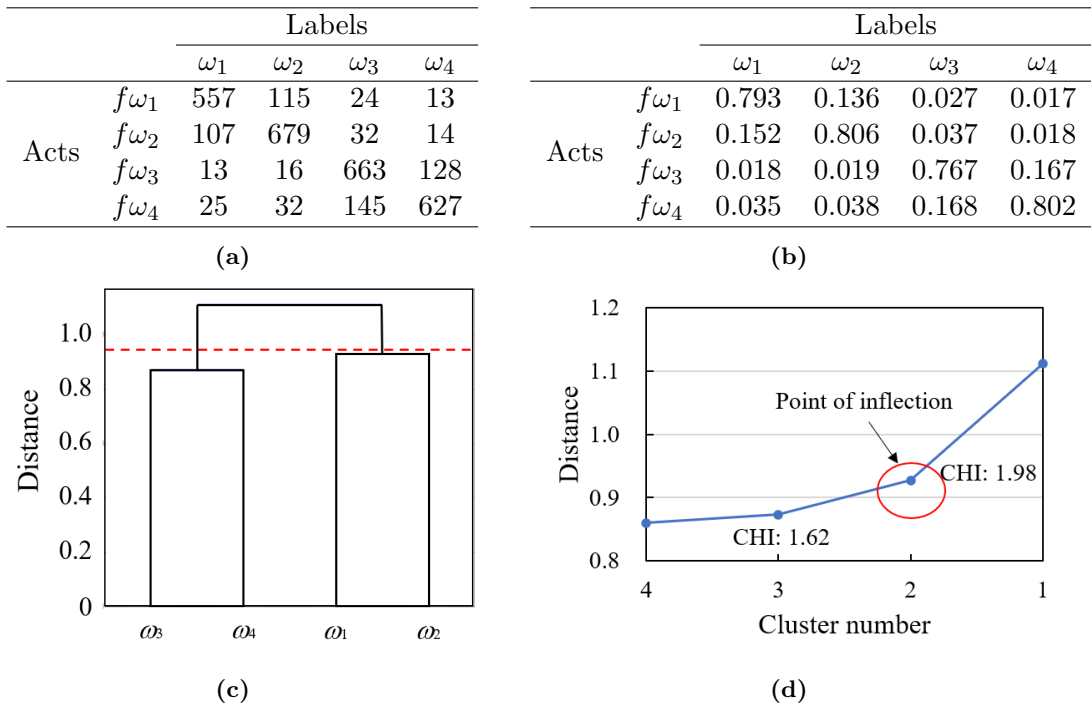


Figure 3.3: An example of act selection: confusion matrix (a), normalized confusion matrix (b), dendrogram (c), and a curve of distance vs. cluster number (d).

3.2 Experimental evaluation

In this section, we present numerical experiments demonstrating the advantages of the proposed classifier. In section 3.2.1, we provide two metrics for performance evaluation. Experimental results on image recognition, signal processing and semantic-relationship classification tasks are then reported and discussed, respectively, in Sections 3.2.2, 3.2.3 and 3.2.4.

3.2.1 Evaluation metrics

In the applications of evidential CNN classifiers, we use the extended utility matrix $\mathbb{U}_{(2^\Omega-1)\times M}$ for performance evaluation. For a dataset T , the classification performance is evaluated by the *averaged utility* as

$$AU(T) = \frac{1}{|T|} \sum_{i=1}^{|T|} \hat{u}_{A(i),y_i}, \quad (3.4)$$

where y_i is the true class of example i , $A(i)$ is the selected subset for example i using the notations and equations introduced in Section 1.3.2, $\hat{u}_{A(i),y_i}$ is the utility of assigning sample i to subset $A \subseteq \Omega$ when its true class is y_i . When only considering precise acts, the AU criterion defined by (3.4) boils down to classification accuracy. The *averaged cardinality* is also computed as

$$AC(T) = \frac{1}{|T|} \sum_{i=1}^{|T|} |A(i)|. \quad (3.5)$$

Additionally, we also consider the case where a dataset $T' = \{T'_O, T'_I\}$ is composed of a subset T'_O of outliers whose class does not belong to the frame of discernment Ω , and a subset T'_I of inliers whose class belongs to Ω . We compare the rate of f_Ω in T'_I and T'_O to evaluate the capacity of a classifier to reject outliers together with ambiguous samples. This capacity is called *novelty detection* in [20]. Generally, a well-trained classifier is expected to have a low rate of f_Ω in T'_I but a high rate in T'_O .

In this study, we compare the proposed classifiers with probabilistic CNNs. To ensure a fair comparison, we adopt the following strategy for probability-based set-valued classification in CNNs: $f_A \succeq_* f_{A'}$ if and only if $\mathbb{E}(f_A) \leq \mathbb{E}(f_{A'})$, with $\mathbb{E}(f_A) = \sum_{\omega_k \in A} p(\omega_k) \cdot \hat{u}_{A,k}$.

3.2.2 Image classification experiment

Datasets. We used the CIFAR-10 dataset to evaluate the performance of the proposed classifier in image classification. The CIFAR-10 dataset [63] consists of 60k RGB images of size 32×32 partitioned in 10 classes. There are 50k training examples and 10k testing examples. During training, we randomly selected 10k images as validation data. We used two datasets (CIFAR-100 [63] and MNIST [69]) for novelty detection. The CIFAR-100 dataset is just like the CIFAR-10 except it has 100 classes containing 600 images each, while the MNIST dataset of handwritten digits has 70k examples. All examples in the two datasets are used for novelty detection except some images whose classes are included in the CIFAR-10 dataset.

Precise classification. To only perform the precise classification, the utility layer connects its input mass functions and outputs of precise acts $\mathcal{F} = \{f_{\omega_1}, \dots, f_{\omega_M}\}$.

Table 3.3: Three CNN backbones used on CIFAR-10 dataset.

NIN [74]	FitNet-4 [106]	ViT-L/16 [27]
Input: $32 \times 32 \times 3$		
		$16 \times 16 \times 3 \times 4$ patches
5×5 Conv. NIN 64 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>
	3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>
	3×3 Conv. 32 <i>ReLU</i>	3×3 Conv. 32 <i>ReLU</i>
	3×3 Conv. 48 <i>ReLU</i>	3×3 Conv. 48 <i>ReLU</i>
	3×3 Conv. 48 <i>ReLU</i>	3×3 Conv. 48 <i>ReLU</i>
2×2 max-pooling with 2 strides		
5×5 Conv. NIN 64 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
2×2 mean-pooling with 2 strides	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
	3×3 Conv. 80 <i>ReLU</i>	3×3 Conv. 80 <i>ReLU</i>
2×2 max-pooling with 2 strides		
5×5 Conv. NIN 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
2×2 mean-pooling with 2 strides	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
	3×3 Conv. 128 <i>ReLU</i>	3×3 Conv. 128 <i>ReLU</i>
	8×8 max-pooling with 2 strides	4×4 max-pooling with 2 strides+position embedding
Average global pooling		Transformer encoder

Table 3.4: Test average utilities in precise classification on CIFAR-10 dataset.

Models	NIN [74]		FitNet-4 [106]		ViT-L/16 [27]	
	Probabilistic classifier	Evidential classifier	Probabilistic classifier	Evidential classifier	Probabilistic classifier	Evidential classifier
Utility	0.8959	0.8978	0.9353	0.9361	0.9921	0.9908
<i>p</i> -value (McNemar’s test)	0.0489		0.0492		0.0452	

In this experiment, three CNN backbones were combined with the DS and utility layers, as shown in Table 3.3. The detailed information of the three CNN backbones has been introduced in Section 2.1.4. The three backbones have the same number of output feature maps but different convolutional and pooling layers. As shown in Table 3.4, the proposed classifiers slightly outperform the probabilistic ones in precise classification, except with ViT-L/16 feature extraction. McNemar’s test results indicate a small but statistically significant effect of the proposed combination on the image classification task with *p*-values below 5%. These results suggest that the utility of an evidential classifier is larger than that of a probabilistic CNN classifier with the same backbone as the evidential one. They also demonstrate that the use of the convolutional and pooling layers in Step 1 of Section 3.1.1 allows for good precise-classification performance of the evidential CNN classifiers.

Transfer learning. The feasibility of transfer learning on the proposed classifier was also verified in this study. The three evidential CNN classifiers trained on the CIFAR-10 classification task, as well as the three probabilistic CNNs, were fine-tuned using the training set of the CIFAR-100 dataset as a new task. Table 3.5 shows the testing utilities of fine-tuned classifiers on the CIFAR-100 dataset. The evidential and probabilistic classifiers achieve close results for precise classification after fine-tuning. Besides, the average utilities of the evidential CNN classifiers are close to those already reported in [27, 74, 106]. This demonstrates the feasibility of transfer learning with the proposed classifiers.

Table 3.5: Test average utilities for precise classification of the CIFAR-100 data set after transfer learning.

Models	NIN [74]		FitNet-4 [106]		ViT-L/16 [27]	
	CNN classifier	Evidential classifier	Probabilistic classifier	Evidential classifier	Probabilistic classifier	Evidential classifier
Utility	0.3442	0.3461	0.6688	0.6714	0.8251	0.8217

Precise classification with a rejection option. We use the outputs of a DS layer and one of the evidence-theoretic decision rules with a parameter λ_0 (see Section 1.3.2) to perform the precise classification with a rejection option. The act set is $\mathcal{F} = \{f_{\omega_0}, f_{\omega_1}, \dots, f_{\omega_M}\}$. The test error rates with rejection of the evidential and probabilistic CNN classifiers are presented in Figure 3.4. A rejection decision is not regarded as an incorrect classification. When the rejection rate is 0, indicating that there are no rejection decisions, the test set error in Figure 3.4 equals to $(1 - AU) \times 100\%$ in Table 3.4. When the rejection rate increases, the test set error decreases, which shows that the evidential classifiers reject a part of the incorrect classification. However, the error decreases slightly when the rejection rate is higher than 7.5%. This demonstrates that the evidential classifiers reject more and more correctly classified patterns with the increase of rejection rates. Thus, a satisfactory λ_0 should be determined to guarantee that the evidential CNN classifiers have a desirable accuracy rate and a low correct-rejection rate. In [130], we propose a method of k -fold cross-validation for determining λ_0 to guarantee a classifier with a certain rejection rate, as shown in Figure 3.5. We randomly select four-fifths of a learning set to train an evidential classifier, while the rest of the set is used as a validation set to draw a $\lambda_0^{(1)}$ -rejection curve. We can determine the value of $\lambda_0^{(1)}$ for a certain rejection rate from the curve. We repeat the process k times and take the average of $\lambda_0^{(i)}$ as the final λ_0 for the desired rejection rate.

Compared with the probabilistic ones, the evidential classifiers reject significantly more incorrectly classified patterns using one of the evidence-theoretic decision rules. For example, the p -value of McNemar’s test for the difference of error rates between the evidential and probabilistic CNN classifiers with a 5.0% rejection rate is close to 0. We can conclude that an evidential classifier with an evidence-theoretic rejection rule is more suitable for making a decision allowing for pattern rejection than a softmax layer and the probability-based rejection rule.

Table 3.6 presents the prediction distribution of the evidential NIN classifier with a 5% rejection ratio. The classifier tends to select rejection when there are two or more similar patterns, such as the “dog” and “cat” classes, which can lead to incorrect classification. In the view of evidence theory, the CNN architecture provides conflicting evidence when two or more similar patterns exist. The maximally conflicting evidence corresponds to $m(\{\omega_i\}) = m(\{\omega_j\}) = 0.5$ [23]. Additionally, the additional mass function $m(\Omega)$ provides the possibility to verify whether the model is well trained because we have $m(\Omega) = 1$ when the CNN architecture cannot provide any useful evidence.

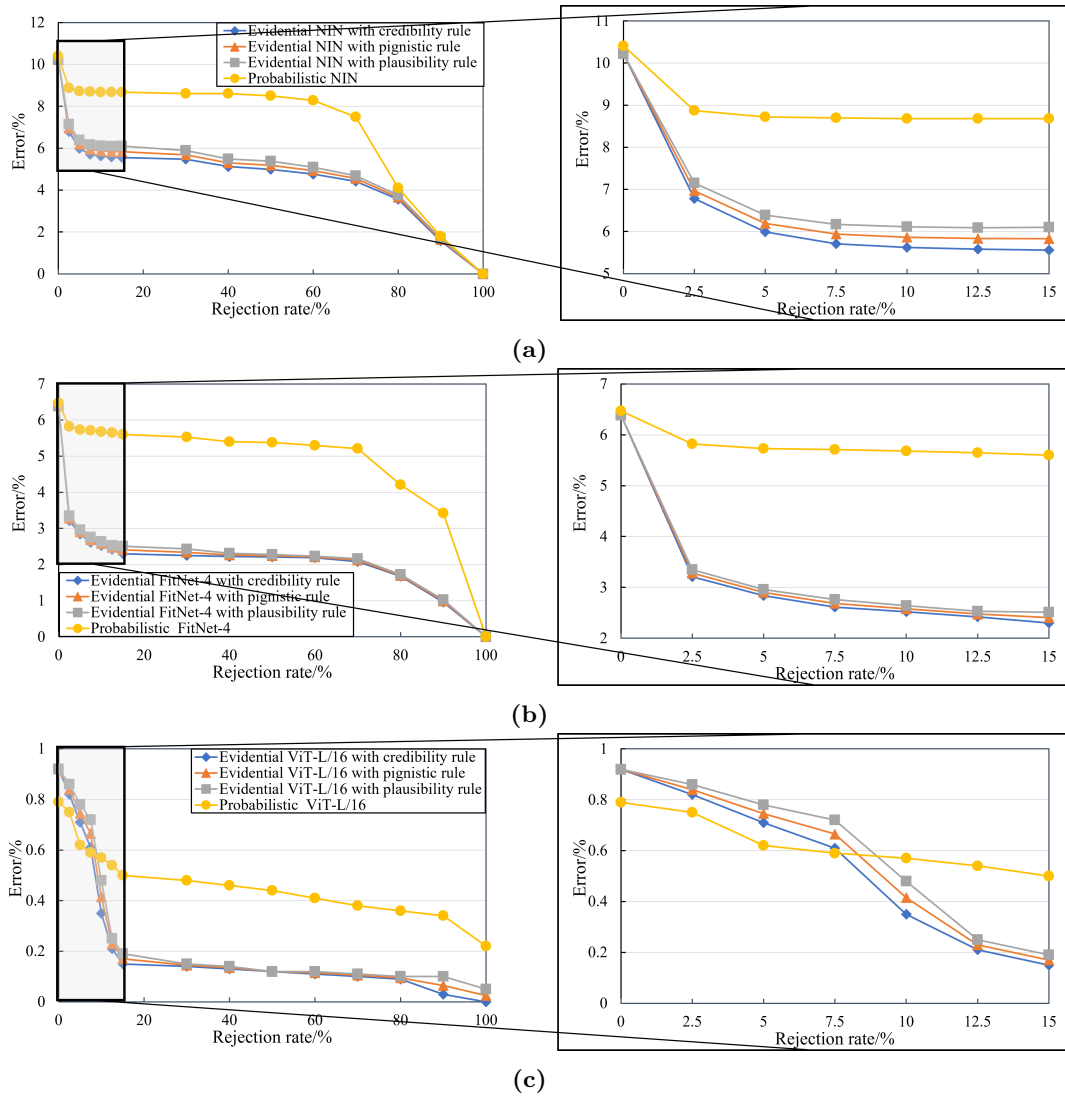


Figure 3.4: Rejection-error curves of evidential NIN (a), FitNet-4 (b), and ViT-L/16 (c) on the CIFAR-10 testing set. A rejection rate mean the percent of the samples with the reject act in a dataset.

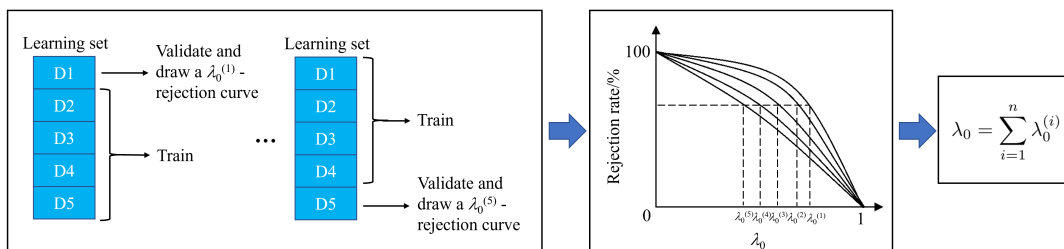


Figure 3.5: Illustration of the 5-fold cross validation for determining λ_0 with

Table 3.6: Prediction distribution for the evidential CNN classifier with the NIN backbone on the CIFAR-10 dataset when using maximum credibility rule and 5.0% rejection rate. The sums of the table and each column equal to 100% and 10%, respectively.

		Label									
		Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Prediction	Airplane	9.65	0.03	0.03	0.01	0.02	0.05	0.03	0.01	0.04	0.05
	Automobile	0	9.63	0.04	0.04	0.08	0.08	0.02	0.06	0.02	0.07
	Bird	0.02	0.04	9.39	0.02	0.04	0.07	0.03	0.08	0	0.02
	Cat	0.02	0.02	0.1	8.02	0.06	0.44	0.11	0.04	0.05	0.05
	Deer	0.01	0.04	0.07	0.12	8.3	0.03	0.12	0.34	0.04	0.08
	Dog	0.02	0.03	0.05	0.49	0.11	7.99	0.06	0.09	0.01	0.04
	Frog	0.01	0.04	0.08	0.06	0.1	0.06	9.35	0.06	0.06	0.05
	Horse	0.01	0.02	0.04	0.06	0.31	0.1	0.04	7.94	0.01	0.04
	Ship	0.04	0.02	0.02	0.04	0.12	0.05	0.04	0.18	9.55	0.02
	Truck	0.02	0	0.04	0.09	0.02	0.06	0.06	0.06	0.04	9.47
	Rejection	0.2	0.13	0.14	1.05	0.84	1.07	0.14	1.14	0.18	0.11

Set-valued classification. In the set-valued classification experiment, we consider all possible acts $\mathcal{F} = \{f_A, A \in 2^\Omega \setminus \{\emptyset\}\}$. Before evaluating the performance of the proposed classifiers in set-valued classification, we need to determine the optimal pessimism index ν in Eq. (1.26a) once given a value of imprecision tolerance degree γ . Based on the ν -utility curves on the training set (Figure 3.6), we can determine the optimal ν for any given γ . As we consider all of the $2^{|\Omega|}$ acts, the three proposed classifiers always achieve average utilities of 1 when γ equals 1. The value of ν has an apparent effect on the average utilities when γ is higher than 0.7. These curves show that parameter ν should be carefully tuned to ensure optimal performance of the proposed classifier in set-valued classification.

Figure 3.7 shows the test average utilities and cardinalities of the evidential CNN classifiers as functions of γ with the optimal ν . When the imprecision tolerance degree increases, the average cardinalities increase. This indicates that the proposed classifiers tend to perform set-valued assignments when given a large tolerance degree of imprecision. The test average utilities decrease slightly and then increase when γ increases. To explain this behavior, Table 3.7 provides four examples with their assignments and corresponding utilities. For the first example, the utility increases from 0 to 1 as γ becomes larger. However, for examples correctly classified when $\gamma = 0.5$ (#2 and #3), their utilities first decrease and then increase back to 1. The majority of examples in the CIFAR-10 testing set fall in the latter category. Therefore, the test average utilities decrease slightly and then increase when γ increases from 0.5 to 1.

The use of the DS and utility layers has an effect when there is a lack of evidence in a CNN backbone. In Figure 3.7, when γ is increased from 0.5 to 0.9, the largest gains in average utility are obtained by the evidential classifier with the NIN backbone [74], whose feature extraction was found to be the worst among the three proposed classifiers since it achieved the minimum utility in the precise assignments (Table 3.4). Thus, the classifier with the NIN backbone is more affected by the DS and utility

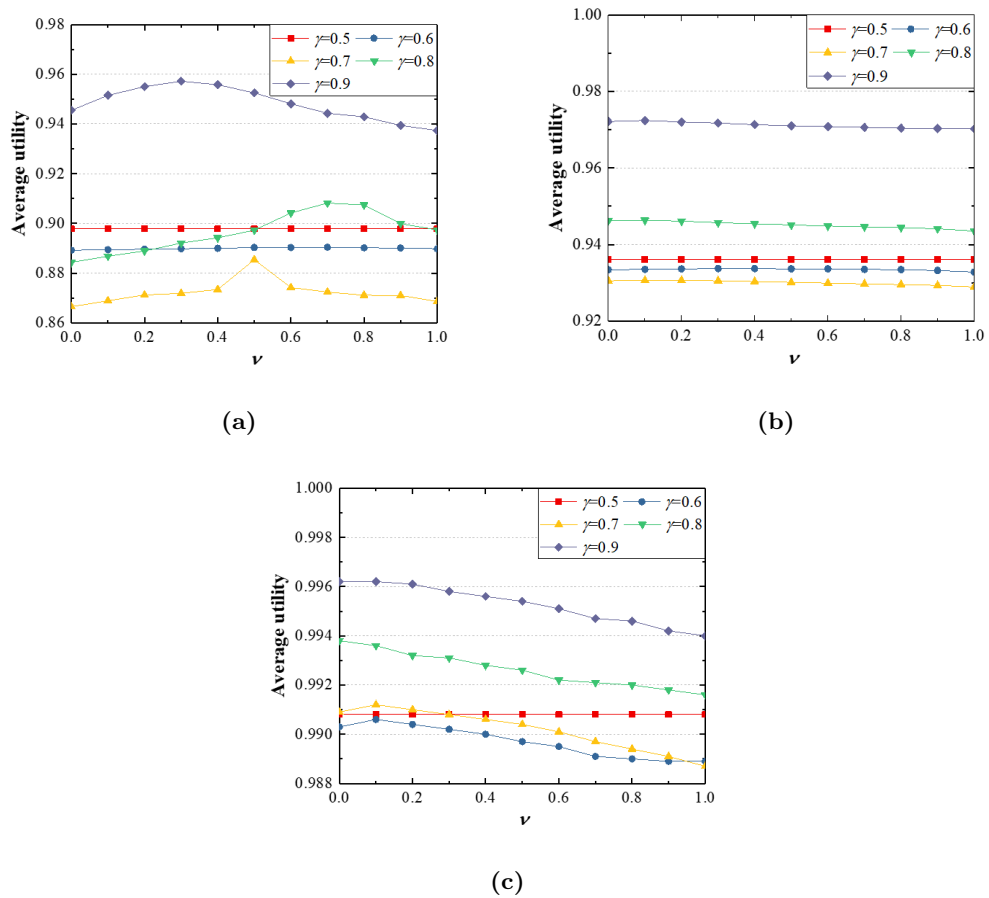






Figure 3.6: Average utility vs. ν for the evidential CNN classifiers on the CIFAR-10 dataset: NIN (a), FitNet-4 (b), and ViT-L/16 (c).

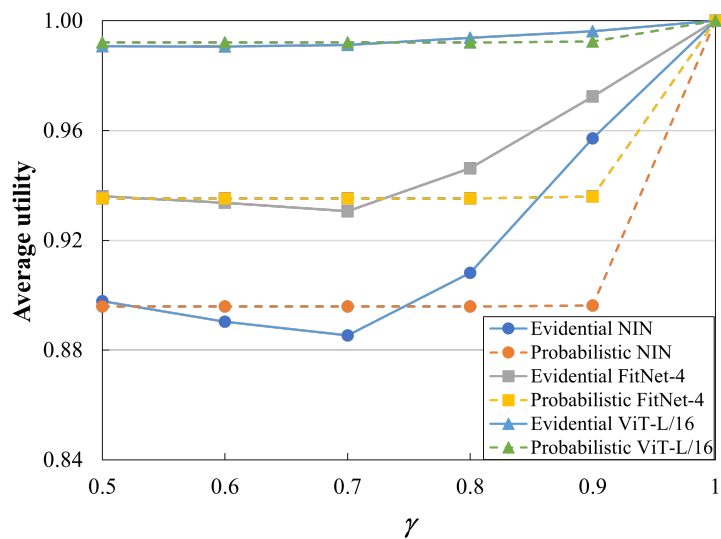
Table 3.7: Label classification/utilities with different γ .

	#1($\omega^* = \text{cat}$)	#2($\omega^* = \text{dog}$)	#3($\omega^* = \text{deer}$)	#4($\omega^* = \text{automobile}$)
$\gamma=0.5$	{dog}/0	{dog}/1	{deer}/1	{airplane}/0
$\gamma=0.6$	{cat,dog}/0.6	{cat,dog}/0.6	{deer}/1	{airplane}/0
$\gamma=0.7$	{cat,dog}/0.7	{cat,dog}/0.7	{deer,horse}/0.7	{airplane}/0
$\gamma=0.8$	{cat,dog}/0.8	{cat,dog}/0.8	{deer,horse}/0.8	{airplane}/0
$\gamma=0.9$	{cat,dog}/0.9	{cat,dog}/0.9	{cat,deer,dog,horse}/0.7104	{cat,deer,dog,horse}/0
$\gamma=1.0$	$\Omega/1.0$	$\Omega/1.0$	$\Omega/1.0$	$\Omega/1.0$
				

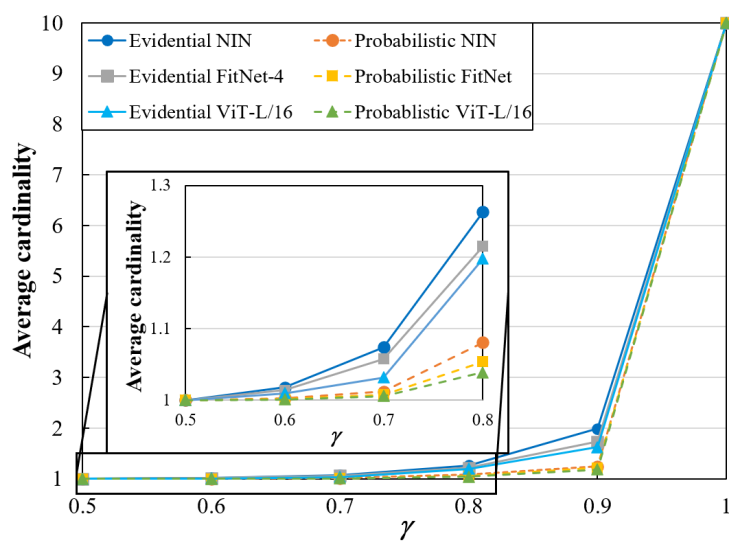
layers than the other two classifiers. Therefore, we can conclude that the effects of DS and utility layers are more significant if there is a lack of evidence in the feature extraction part.

As shown in Figure 3.7, the proposed model with a DS layer and a utility layer outperforms probabilistic CNN classifiers for set-valued classification. The average utilities of the proposed classifiers increase significantly when γ increases from 0.5 to 0.9. In contrast, the average utilities of the probabilistic CNN classifiers only increase sharply when γ increases from 0.9 to 1.0. This is evidence that the proposed classifiers make well-distributed set-valued classification based on the user’s tolerance degree of imprecision, while the probabilistic CNN classifiers only assign samples to the multi-class sets when the tolerance is large. This phenomenon is caused by the use of DS and utility layers in the proposed classifiers. The DS layer tends to generate uniformly distributed masses when the features are not informative. As a result, the expected utility of a set-valued classification is the maximum among all acts, rather than the utility of a precise classification. This effect explains the superiority of the proposed approach for set-valued classification. However, the average utilities of the evidential classifiers are less than those of the probabilistic CNN classifiers for $\gamma = 0.7$. The reason is that the probabilistic CNN classifiers make few set-valued assignments for $\gamma = 0.7$, and the evidential classifiers are so cautious that they perform set-valued assignments for some instances that are correctly classified when γ is less than 0.7, such as #2 and #3 in Table 3.7.

In the experiment of the precise classification with a rejection option, we found that some ambiguous patterns always led the incorrect classification. Thus, we do not need to consider all of the 2^Ω acts. In this experiment, the performances of the classifiers with partial acts are compared to those with all 2^Ω acts. Taking the evidential classifier with a NIN backbone [74] as an example, we used the strategy introduced in Section 3.1.3 to generate the dendrograms, as shown in Figure 3.8. When using Ward linkage [139], we get an inflection point to cut the dendrogram, with the CHI equal to 1.286 and the corresponding distance equal to 1.238. The selected multi-class sets consist of {cat, dog}, {deer, horse}, {cat, dog, deer, horse},



(a)



(b)

Figure 3.7: Average utility (a) and average cardinality (b) vs. γ for the evidential and probabilistic CNN classifiers on the CIFAR-10 dataset.

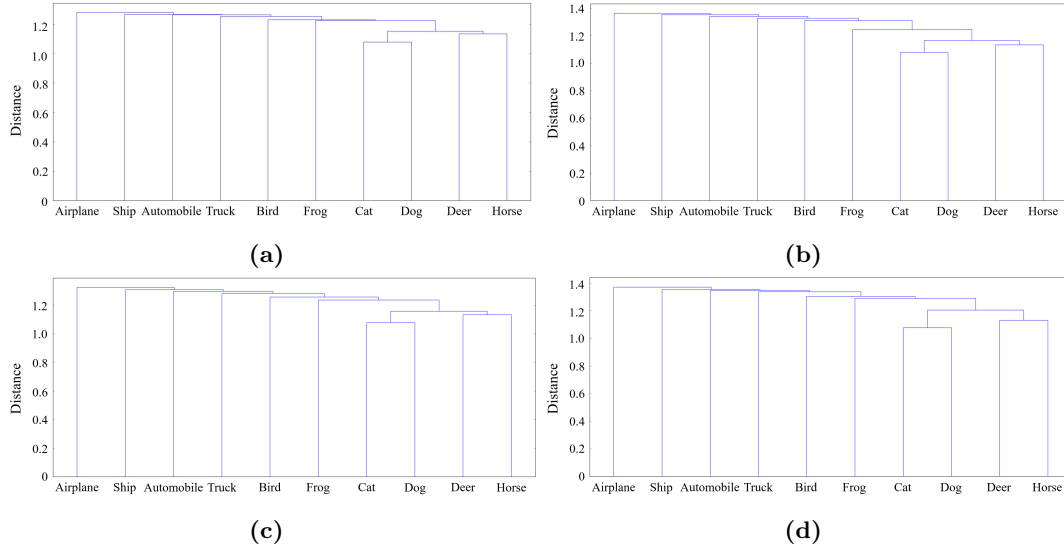


Figure 3.8: Dendrograms for the CIFAR-10 dataset: single linkage (a), complete linkage (b), average linkage (c), and Ward linkage (d).

Table 3.8: Set-valued assignment rates using the selected and 2^Ω acts (unit:%).

γ		0.5	0.6	0.7	0.8	0.9	1
CIFAR-10	Selected acts	0	0.52	1.74	13.24	19.62	52.04
	2^Ω acts	0	0.52	1.76	14.21	22.67	100
UrbanSound 8K	Selected acts	0	2.47	9.10	23.96	49.91	64.43
	2^Ω acts	0	2.47	9.71	28.74	55.62	100
SemEval-2010 Task 8	Selected acts	0	1.69	8.11	17.62	43.11	66.62
	2^Ω acts	0	1.69	8.57	27.71	52.77	100

and $\{cat, dog, deer, horse, frog\}$ in the comparison study. Table 3.8 reports the testing rates of set-valued classification using the selected and 2^Ω acts. The rates of the classifiers with the selected and 2^Ω acts are close when γ is less than 0.9. Besides, the rates of the samples assigned correctly using 2^Ω acts but incorrectly using the selected acts are small when γ is less than 0.9, as shown in Table 3.9. A set-valued assignment is regarded as correct if the multi-class set contains the true label. Thus, the proposed strategy is useful once an evidential classifier has a value of γ in the range of 0.5-0.9.

Table 3.9: Proportions of samples correctly assigned to acts in 2^Ω and incorrectly assigned to selected acts, for different values of γ .

γ	0.5	0.6	0.7	0.8	0.9	1
CIFAR-10	0	0	0	0.18	0.47	2.87
UrbanSound 8K	0	0	0	0.42	0.95	6.62
SemEval-2010 Task 8	0	0	0.11	0.48	0.74	4.43

Novelty detection. Figure 3.9 displays the results of novelty detection using evidential and probabilistic CNN classifiers. The evidential CNN classifiers can assign outliers and a few of the known-class examples to set Ω when values of γ are between 0.7 and 0.9, while the probabilistic CNN classifiers cannot, which demonstrates that the proposed models outperform the probabilistic CNN classifiers for rejecting outliers together with ambiguous samples. This is due to the fact that, when the feature vector fed into the DS layer is far from all prototypes, the activations of the RBF units in the DS layer become close to zero, as shown by Eq. (1.29). As a consequence, all the mass functions \mathbf{m}^i computed by Eq. (1.30) assign a large mass to set Ω , and so does their orthogonal sum \mathbf{m} . The output of the DS layer thus reflects ignorance about the class of the input sample (whereas the probabilistic output of the softmax layer does not), leading to the assignment of the sample to set Ω .

We also applied McNemar’s test with the CIFAR-100 and MNIST datasets, where outliers assigned to Ω are regarded as positive samples, and the others are negative ones. The results indicate the use of the DS and utility layers has a distinct effect on novelty detection since all p -values are smaller than 0.001. However, none of the classifiers performs well when γ is less than 0.7 since these classifiers favor precise decisions. The classifiers tend to reject outliers whose features are different from the known classes. For example, the proposed classifiers reject more samples in the MNIST dataset than in the CIFAR-100 dataset since the hand-written digits are very different from the patterns in the CIFAR-10 dataset.

3.2.3 Signal classification experiment

In the application of the proposed classifier on signal processing, we used the UrbanSound 8K dataset [108] composed of 8732 short (less than 4 seconds) excerpts of various urban sound sources (air conditioner (*AI*), car horn (*CA*), playing children (*CH*), dog bark (*DO*), drilling (*DR*), engine idling (*EN*), gun shot (*GU*), jackhammer (*JA*), siren (*SI*), street music (*ST*)) prearranged into 10 classes. The ratio between the training and testing set is about 3:1. We randomly selected 25% of the training samples as validation data. Free Spoken Digit Dataset (FSDD) [109], was used to evaluate the capacity of novelty detection in the signal classification experiment. FSDD is an audio/speech dataset with 2k recordings (50 of each digit per speaker) in English pronunciations.

The baseline CNN backbones in this experiment are shown in Table 3.10. The DS and utility layers show a significant difference in the precise classification as $0.01 < p < 0.05$ according to McNemar’s test (Table 3.11). Similarly to CIFAR-10, this demonstrates that the performance of the proposed classifiers is better than those of probabilistic CNN classifiers for precise classification. As shown in Figure 3.10, the results of the precise classification with a rejection option demonstrate that the use

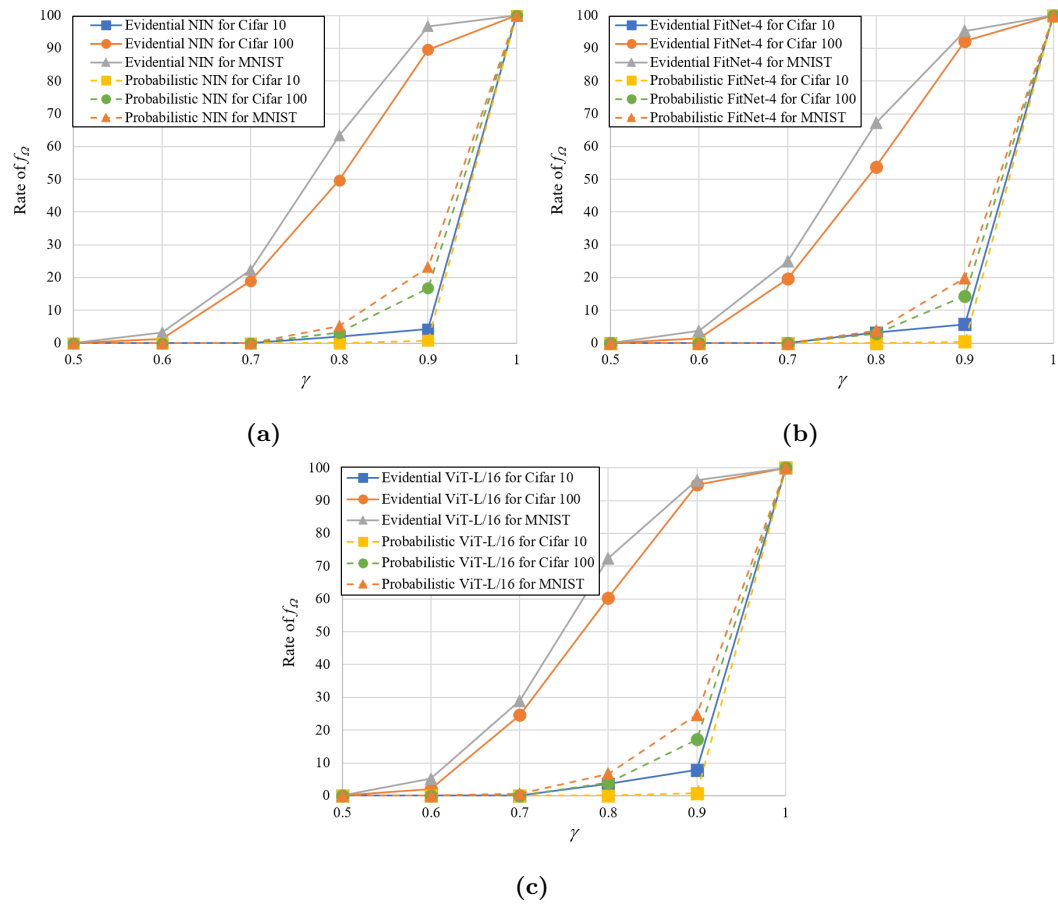


Figure 3.9: Rate of f_Ω vs. γ for novelty detection in the image-classification experiment: NIN (a), FitNet-4 (b), and ViT-L/16 (c).

Table 3.10: Three baseline CNN backbones used on UrbanSound 8K.

Stage 1 [100]	Stage 2	Stage 3
Pre-processing: clip, data augmentation, and segmentation		
Input: $60 \times 41 \times 2$		
57×6 Conv. 80 <i>ReLU</i>	57×6 Conv. 80 <i>ReLU</i> 1×1 Conv. 80 <i>ReLU</i>	29×3 Conv. 80 <i>ReLU</i> 29×3 Conv. 80 <i>ReLU</i>
4 \times 3 max-pooling stride 1×3 with 50% dropout		
1×3 Conv. 80 <i>ReLU</i>	1×3 Conv. 80 <i>ReLU</i> 1×1 Conv. 80 <i>ReLU</i>	1×2 Conv. 80 <i>ReLU</i> 1×2 Conv. 80 <i>ReLU</i>
1 \times 3 max-pooling stride 1×3 without dropout		

Table 3.11: Test average utilities in precise classification on Urban-Sound 8K.

Models	Stage 1 [100]		Stage 2		Stage 3	
	Probabilistic classifier	Evidential classifier	Probabilistic classifier	Evidential classifier	Probabilistic classifier	Evidential classifier
Utility	0.7132	0.7261	0.7164	0.7284	0.7210	0.7302
<i>p</i> -value (McNemar’s test)	0.0234		0.0319		0.0365	

of a DS layer can improve the performance of signal classification by rejecting some ambiguous samples.

After determining the optimal ν for each value of γ based on the ν -utility curves (Figure 3.11), we can compute the test average utilities and cardinalities of the evidential and probabilistic CNN classifiers, as shown in Figure 3.12. The proposed classifiers outperform the probabilistic ones with the same CNN backbones for the set-valued classification in the signal processing task. The proposed classifiers make more cautious decisions than do the probabilistic CNNs since it assigns ambiguous samples to multi-class sets. Additionally, the performance of the evidential classifiers exceeds those of the probabilistic classifiers in novelty detection (Figure 3.13). The use of the DS and utility layers has significant effects on novelty detection as the results of *p*-value are close to 0 according to McNemar’s test.

For the testing of act-selection strategy, an inflection point was used to cut off the complete-linkage dendrogram [15] in Figure 3.14, in which CHI is 2.198 and corresponding distance is 1.036. Thus, we selected partial multi-class sets including $\{DR, JA\}$, $\{AI, EN\}$, $\{CH, ST\}$, $\{DR, JA, AI, EN\}$, and $\{DR, JA, AI, EN, CH, ST\}$. From Tables 3.8 and 3.9, we can see that the strategy works well if γ is less than 0.9. This demonstrates that the proposed strategy is acceptable when the classifier has a reasonable γ .

3.2.4 Semantic-relationship classification experiment

For the semantic-relationship classification task, we used the SemEval-2010 Task 8 dataset [50]. It contains 10,717 annotated examples, including 8,000 training instances and 2,717 test instances. There are 10 semantic relationships in the dataset as cause-effect (*CE*), instrument-agency (*IA*), product-producer (*PP*), content-container

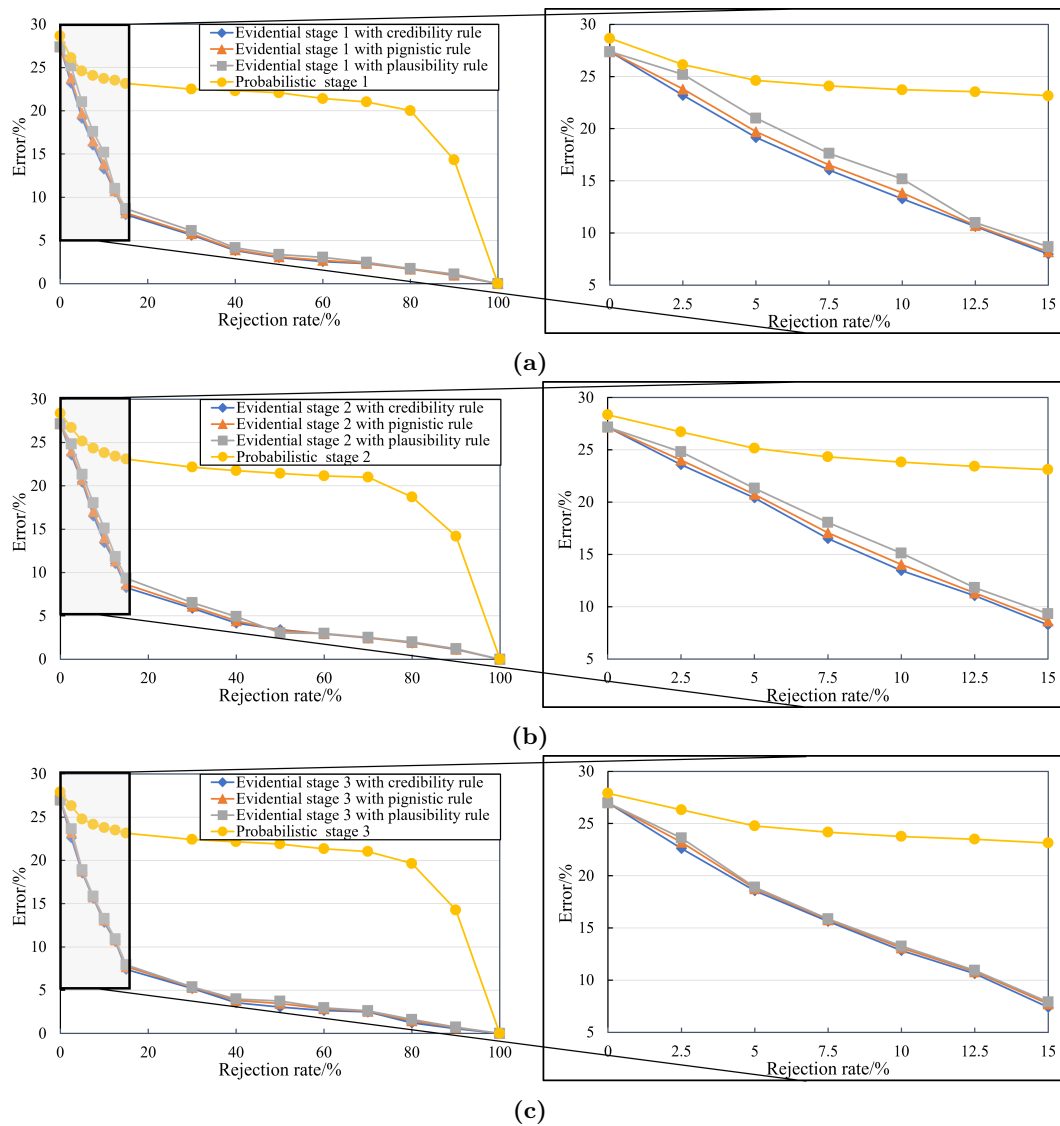


Figure 3.10: Rejection-error curves of evidential stage 1 (a), stage 2 (b), and stage 3 (c) on the UrbanSound 8K testing set. A rejection rate mean the percent of the samples with the reject act in a dataset.

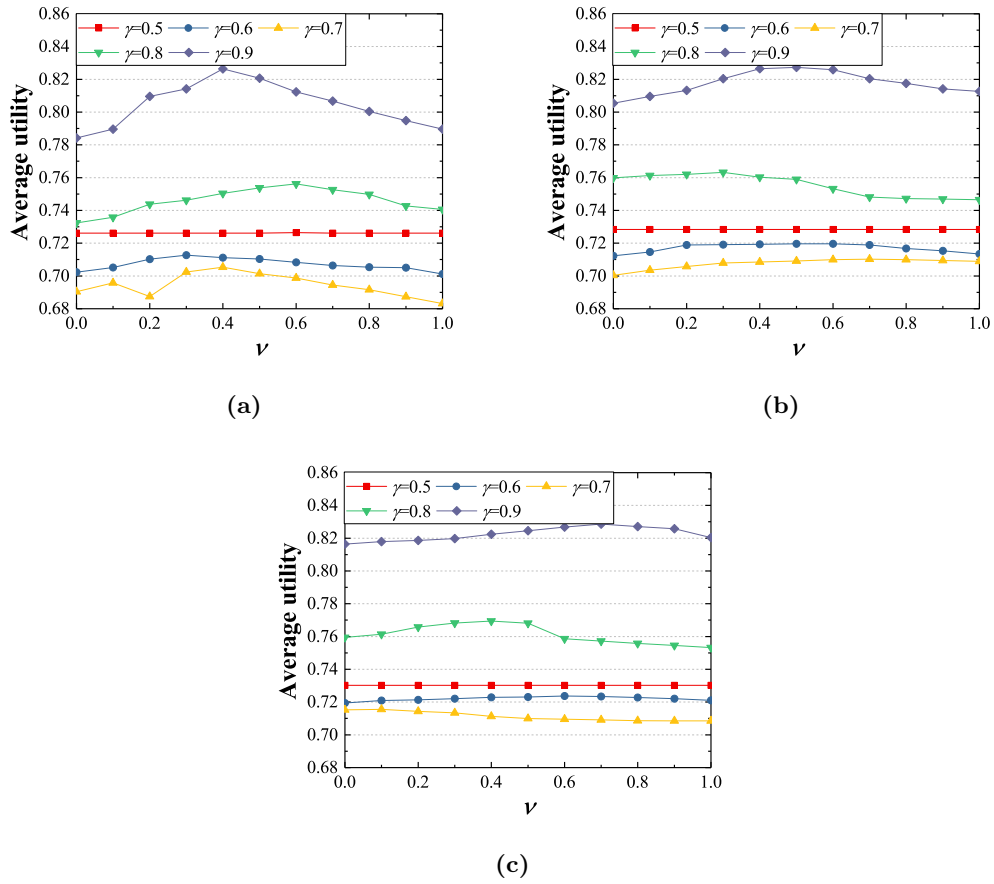


Figure 3.11: Average utility vs. ν for the evidential CNN classifiers on the UrbanSound 8K dataset: Stage 1 (a), Stage 2 (b), and Stage 3 (c).

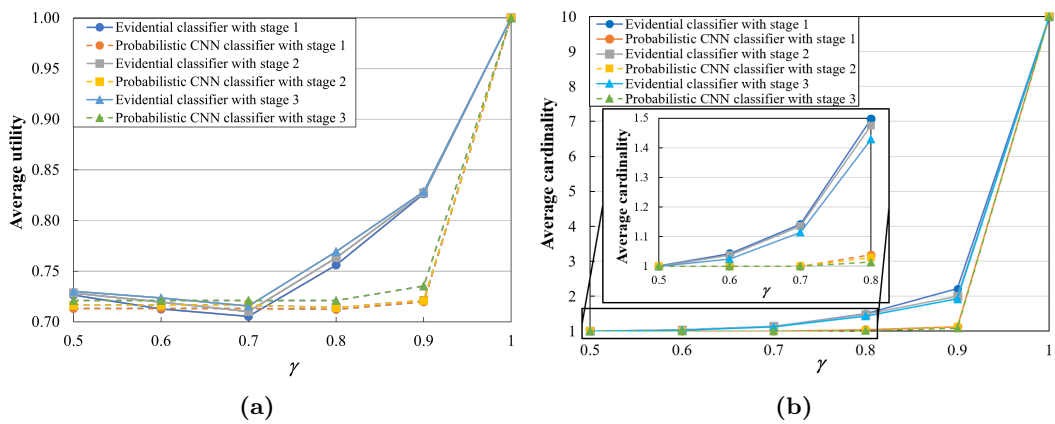


Figure 3.12: Average utility (a) and average cardinality (b) vs. γ for the proposed classifiers and the probabilistic CNN classifiers on the UrbanSound 8K dataset.

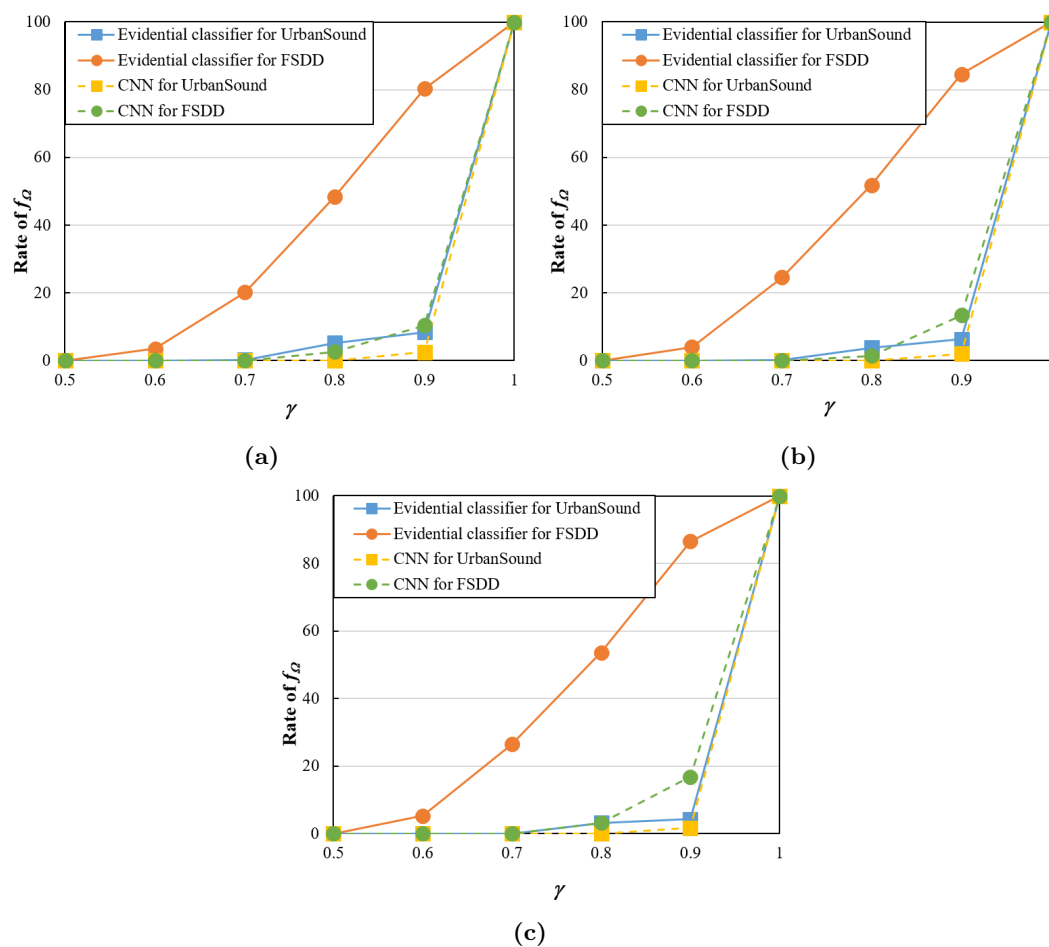


Figure 3.13: Rate of f_Ω vs. γ for novelty detection in the signal-classification experiment: Stage 1 (a), Stage 2 (b), and Stage 3 (c).

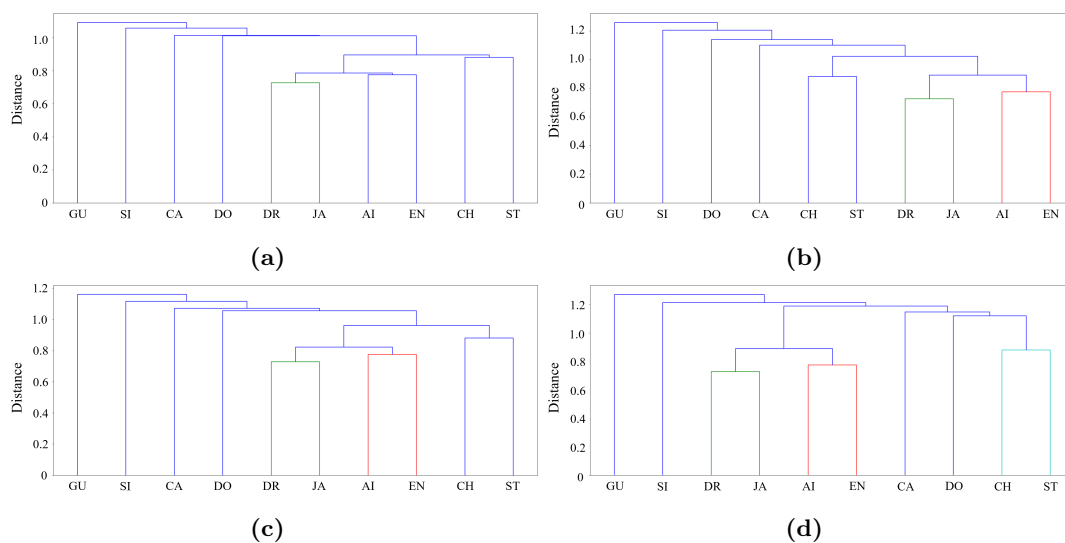


Figure 3.14: Dendrograms for the UrbanSound 8K dataset: single linkage (a), complete linkage (b), average linkage (c), and Ward linkage (d).

Table 3.12: Three baseline CNN backbones used on SemEval-2010 Task 8.

Stage 1 [154]	Stage 2	Stage 3
Pre-processing: word representation		
Input: $50 \times 1 \times t$, in which t is the number of input sentences		
3×1 Conv. 200 <i>ReLU</i>	3×1 Conv. 200 <i>ReLU</i> 1×1 Conv. 200 <i>ReLU</i>	2×1 Conv. 200 <i>ReLU</i> 2×1 Conv. 200 <i>ReLU</i>
1×1 Conv. 100 <i>tanh</i>	1×1 Conv. 200 <i>tanh</i> 1×1 Conv. 100 <i>tanh</i>	1×1 Conv. 200 <i>tanh</i> 1×1 Conv. 100 <i>tanh</i>
1×1 mean-pooling stride 1×1		

Table 3.13: Test average utilities in precise classification on SemEval-2010 Task 8.

Models	Stage 1 [154]		Stage 2		Stage 3	
	Probabilistic classifier	Evidential classifier	Probabilistic classifier	Evidential classifier	Probabilistic classifier	Evidential classifier
Utility	0.8255	0.8347	0.8351	0.8425	0.8370	0.8436
p -value (McNemar's test)	0.0301		0.0415		0.0430	

(*CC*), entity-origin (*EO*), entity-destination (*ED*), component-whole (*CW*), member-collection (*MC*), message-topic (*MT*), and other (*O*). The approach to generate the validation set in this experiment is the same as those used in the experiments on the CIFAR-10 and UrbanSound 8K datasets. The FewRel dataset [46] with 100 semantic-relationship classes and 70k examples was used in novelty detection, in which the known-class examples were excluded in the experiment.

We referred to the backbones shown in Table 3.12 to design the evidential CNN classifiers. In the precise classification, the use of DS and utility layers improves the test average utilities of the CNN models, as shown in Table 3.13. Thus, a DS layer and a utility layer instead of a softmax layer introduce a positive effect on the networks in the semantic-relationship classification. Figure 3.15 indicates that a DS layer with an evidential rejection rule exceeds a softmax layer with a probability rejection rule on processing semantic relationships with highly uncertain information.

The strategy for determining the optimal values of ν in this experiment was the same as those in the CIFAR-10 and UrbanSound 8K experiments (see Figure 3.16). The test average utilities in the set-valued classification of the two types of models are shown in Figure 3.17, which demonstrates the superiority of the evidential deep-learning classifiers. Figure 3.18 indicates the acceptable capacity of novelty detection in the evidential CNN classifiers. Similar to the CIFAR-10 and UrbanSound 8K dataset, the acts generated from the complete-linkage dendrogram (Figure 3.19 and an inflection point whose CHI is 2.627 and a distance equals 1.107) works as well as the 2^Ω acts if the classifier has a suitable γ .

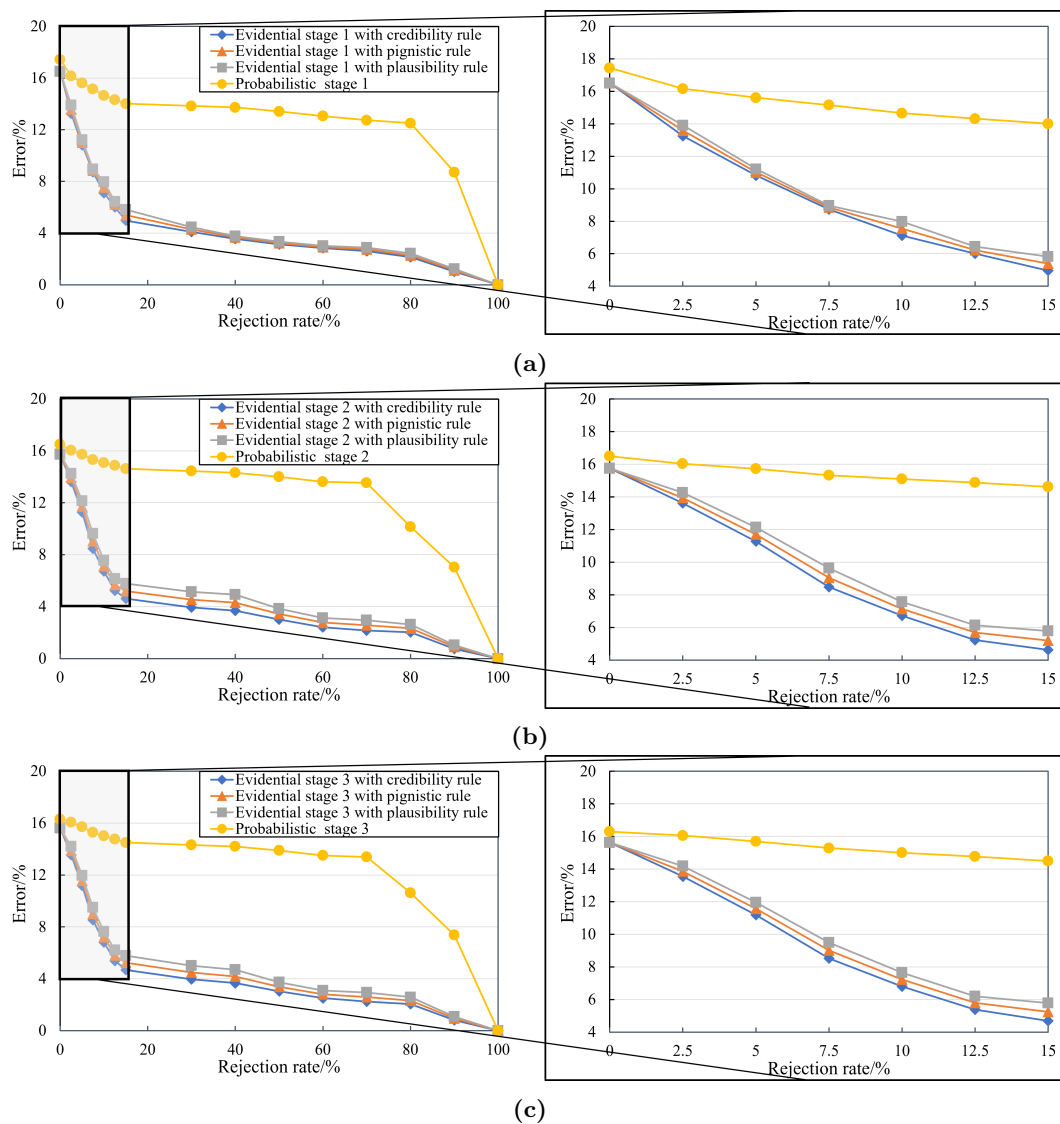


Figure 3.15: Rejection-error curves of evidential stage 1 (a), stage 2 (b), and stage 3 (c) on the SemEval-2010 Task 8 testing set. A rejection rate mean the percent of the samples with the reject act in a dataset.

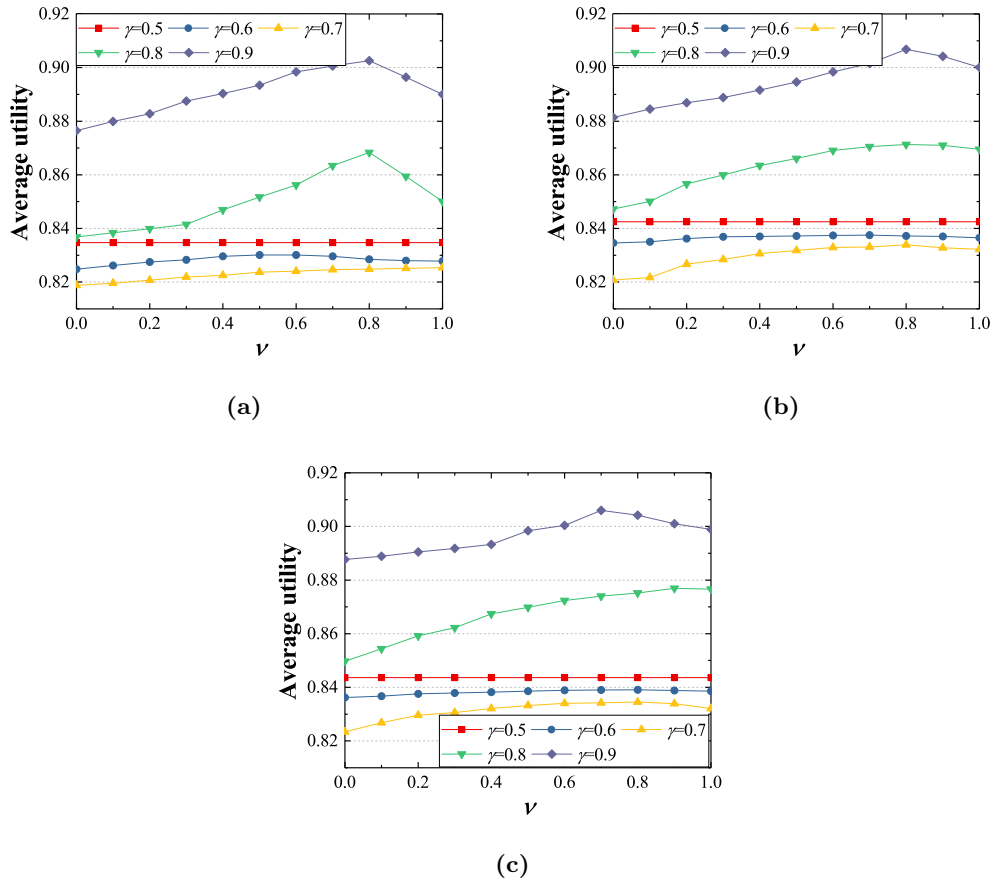


Figure 3.16: Curves in ν -utility for the evidential CNN classifiers on the SemEval-2010 Task 8 dataset: Stage 1 (a), Stage 2 (b), and Stage 3 (c).

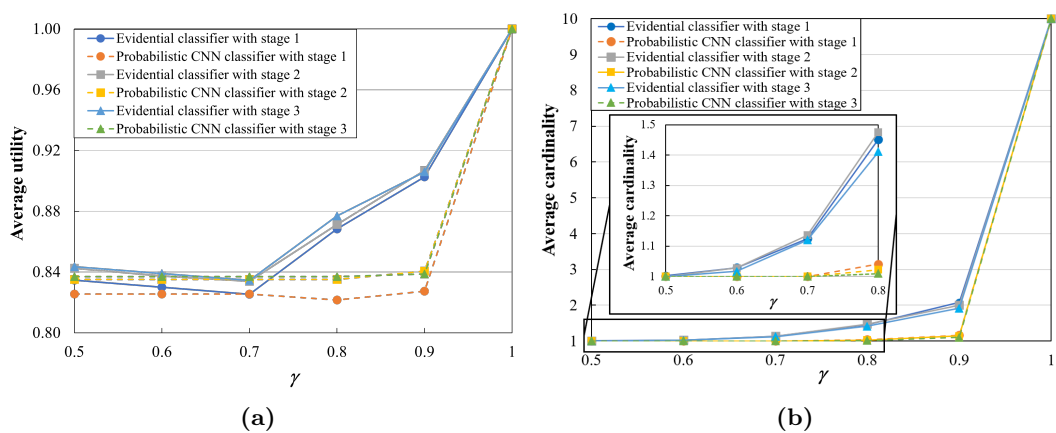


Figure 3.17: Average utility (a) and average cardinality (b) vs. γ for the proposed classifiers and the probabilistic CNN classifiers on the SemEval-2010 Task 8 dataset.

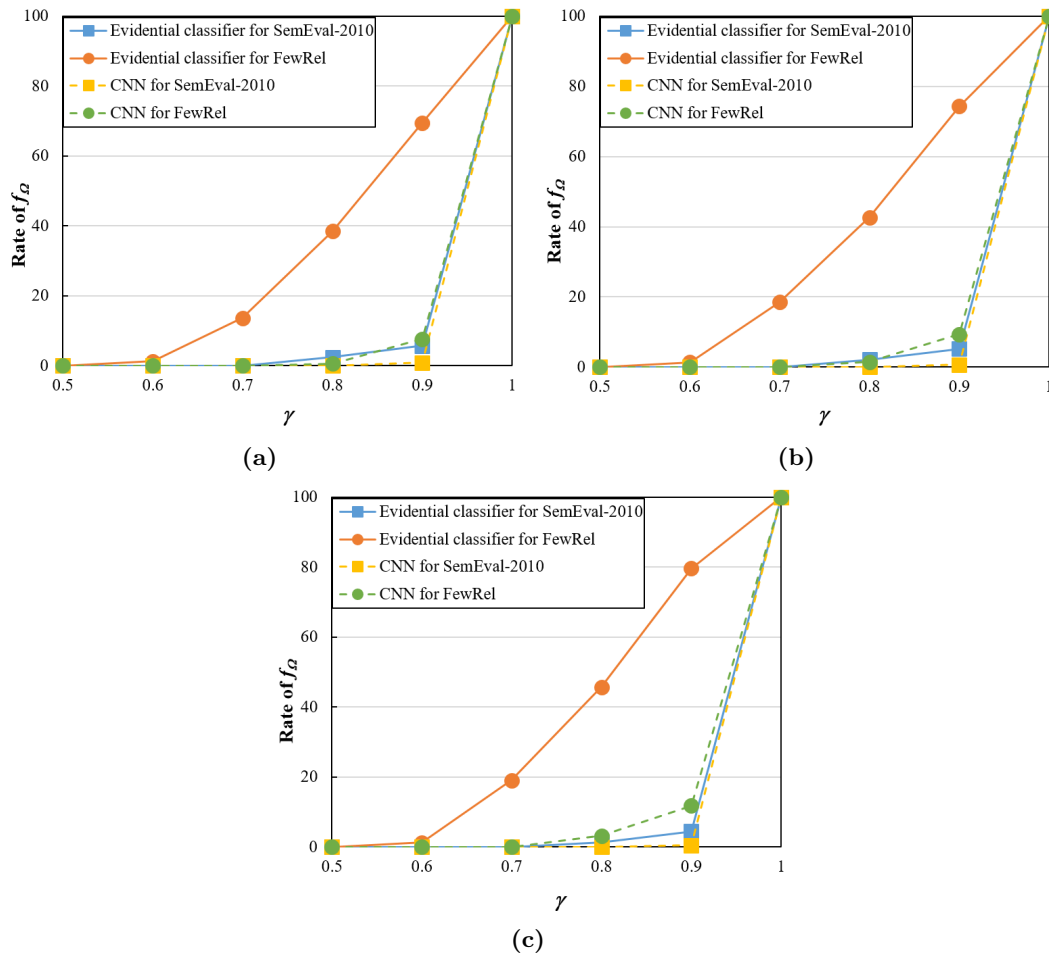


Figure 3.18: Rate of f_Ω vs. γ for novelty detection in the semantic-relationship-classification experiment: Stage 1 (a), Stage 2 (b), and Stage 3 (c).

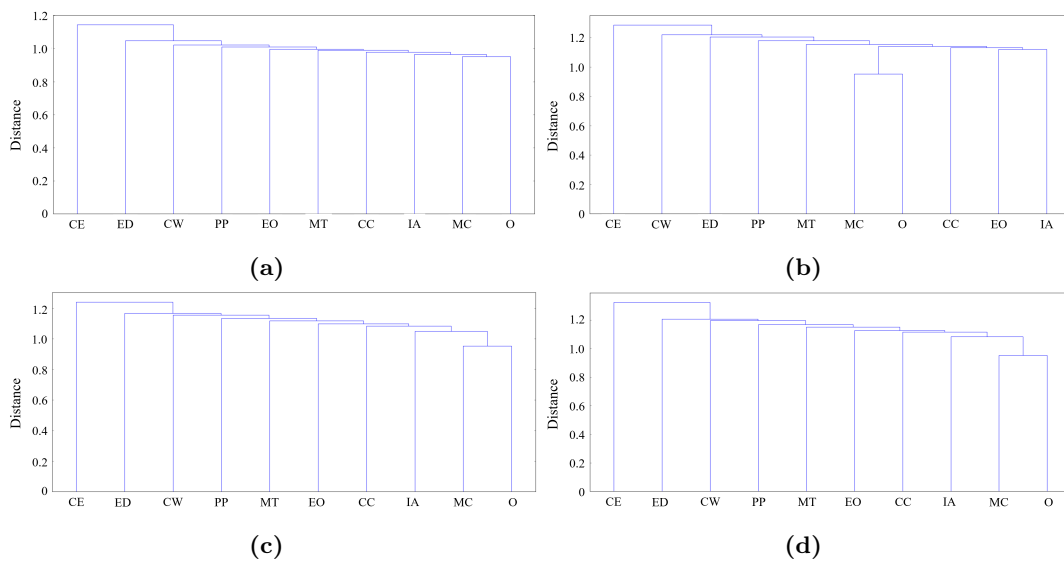


Figure 3.19: Dendrograms for the SemEval-2010 Task 8 dataset: single linkage (a), complete linkage (b), average linkage (c), and Ward linkage (d).

3.3 Conclusion

In this chapter, to deal with the problems of data uncertainty, we have presented a new neural-network classifier based on CNN and DST for set-valued classification, called the evidential CNN classifier. This new classifier consists of a CNN backbone with several convolutional and pooling layers for feature representation, a DS layer to construct mass functions, and a utility layer to make set-valued classification based on the mass functions. The classifier can be trained in an end-to-end way. Besides, we have proposed a strategy to select partial acts instead of considering all of them.

A major finding of this chapter is that the hybridization of CNNs and DST-based ENNs makes it possible to improve the performance of CNN models by assigning ambiguous patterns with uncertain information to multi-class sets. The proposed classifier is able to select a set of classes when the feature representation does not allow us to select a single class unambiguously, which easily leads to incorrect classification in probabilistic classifiers. This result provides a novel direction to improve the cautiousness of CNNs in classification problems. The use of DS and utility layers also improves precise classification performance. The hybridization also makes it possible to reject outliers together with ambiguous patterns when the tolerance degree of imprecise is between 0.7 and 0.9. Additionally, the strategy of selecting partial multi-class acts works as well as that of considering all $2^{|\Omega|}$ acts.

Chapter 4

Evidential fully convolutional network

In this chapter, to further verify the capacity of the evidential deep neural network to deal with data uncertainty, we extend its applications to pixel-wise semantic segmentation, where each pixel in an image must be assigned to one of the subsets of the frame of discernment. We propose a hybrid architecture composed of a fully convolutional network (FCN), a Dempster-Shafer (DS) layer and a utility layer for semantic segmentation [131]. In the so-called evidential FCN (E-FCN), an encoder-decoder architecture of an FCN first extracts pixel-wise feature maps from an input image. A DS 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 datasets (Pascal VOC 2012 [30], MIT-scene Parsing [156] and SIFT Flow [128]) show that the proposed combination improves the accuracy and calibration of semantic segmentation by assigning confusing pixels to multi-class sets.

In this chapter, the proposed E-FCN model is first introduced in Section 4.1. Section 4.2 presents numerical experiments, which demonstrate the advantages of the E-FCNs. Finally, we conclude the chapter in Section 4.3.

4.1 Evidential FCN model

In this section, we describe the proposed E-FCN. Section 4.1.1 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. Section 4.1.2 introduces the strategy for training E-FCN models using a learning set with soft labels.

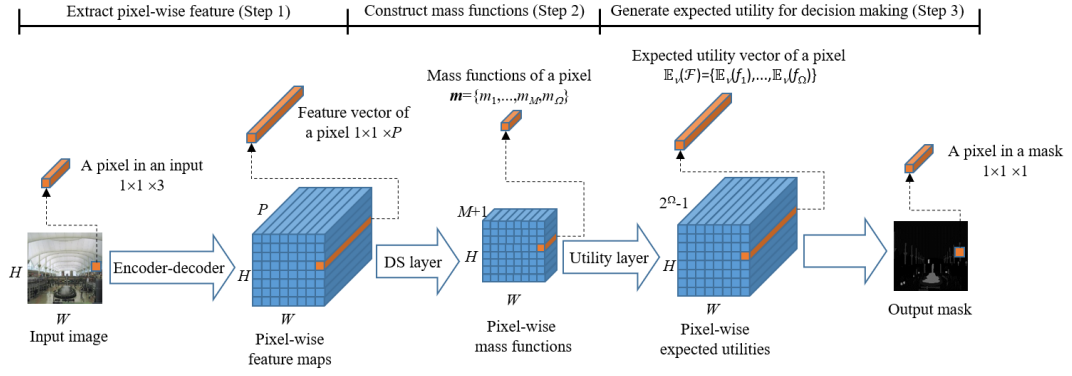


Figure 4.1: Architecture of an evidential fully convolutional network (E-FCN). The E-FCN performs semantic segmentation using a three-step procedure. In the first step, an encoder-decoder architecture extracts pixel-wise feature maps from the input image. Each vector in the feature maps is fed into a 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.

4.1.1 Network architecture

The main idea of this work is to hybridize the ENN classifier presented in Section 1.4 and the FCN recalled in Section 2.2 by “plugging” a DS layer followed by a utility layer at the output of the final upsampling layer in an FCN. The architecture of the proposed method, called the *evidential FCN* (E-FCN), is illustrated in Figure 4.1. An E-FCN classifier performs set-valued semantic segmentation and quantifies the uncertainty about the class of each pixel, taking values in $\Omega = \{\omega_1, \dots, \omega_M\}$, using a three-step procedure defined as follows.

Step 1: 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*, as introduced in Section 2.2. 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 probabilistic 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 converted into a mass function as explained in Section 1.4. The output of the DS layer for a given feature vector is an $(M + 1)$ -dimensional mass vector

$$\mathbf{m} = (m(\{\omega_1\}), \dots, m(\{\omega_M\}), m(\Omega))^T.$$

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 a degree of belief that the ground truth of the 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 [23] and verify whether the model is well trained [130]. The advantages of this uncertainty representation will be demonstrated in the performance evaluation of set-valued semantic segmentation using E-FCN in Section 4.2.4.

Step 3: The output pixel-wise mass vectors are fed into a utility layer to compute the expected utilities of acts for set-valued semantic segmentation, as introduced in Section 1.3.2. This pixel-wise utility layer can also be illustrated as Figure 3.2. In practice, when the cardinality of Ω is very large, we may consider *partial* subsets from 2^Ω . One direction for determining the subsets is presented in Section 3.1.3. For a problem of semantic segmentation, we can follow an easy method. The majority of confusing pixels are at object borders. Thus, we can simply define boundary pixels as soft labels consisting of the neighboring classes. Then 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 4.2.1 below). After selecting the acts, we only need to provide the connection weights between each output unit of the DS layer and each unit of the pixel-wise utility layer corresponding to the selected act. These connection weights do not need to be updated during training because coefficient γ describing the imprecision tolerance degree is fixed. This capability of this step will be demonstrated by the performance comparison between the two types of FCNs in the tasks of set-valued segmentation (Section 4.2.4) and novelty detection (Section 4.2.5).

4.1.2 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 have negative effects on learning systems for image semantic segmentation. The notion of *soft label* [12, 25] may be a way to solve this problem.

Here, we define a soft label as a nonempty subset $A_* \in 2^\Omega \setminus \emptyset$ 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 $\mathbb{E}_{m_l,p}(f_A)$ for $A \in 2^\Omega \setminus \emptyset$ can be computed using Eq. (1.27) and the pignistic transformation Eq. (1.5). Similarly, we consider the *predicted* pignistic expected utilities $\mathbb{E}_{m,p}(f_A)$ for $A \in 2^\Omega \setminus \emptyset$, where m is the predicted mass function from the DS layer of the E-FNC, with focal sets $\{\omega_1\}, \dots, \{\omega_M\}, \Omega$. For a given pixel with soft label m_l and predicted mass function m , using the pignistic criterion (1.27), 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 :

$$\mathcal{L}(m, m_l) = \sum_{\emptyset \neq A \subseteq \Omega} [\mathbb{E}_{m_l,p}(f_A) - \mathbb{E}_{m,p}(f_A)]^2. \quad (4.1)$$

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

$$\begin{aligned} \frac{\partial \mathcal{L}(m, m_l)}{\partial m(\{\omega_k\})} &= \sum_{\emptyset \neq A \subseteq \Omega} \frac{\partial \mathcal{L}(m, m_l)}{\partial \mathbb{E}_{m,p}(f_A)} \cdot \frac{\partial \mathbb{E}_{m,p}(f_A)}{\partial m(\{\omega_k\})} \\ &= -2 \sum_{\emptyset \neq A \subseteq \Omega} [\mathbb{E}_{m_l,p}(f_A) - \mathbb{E}_{m,p}(f_A)] \sum_{j=1}^M \frac{\partial \mathbb{E}_{m,p}(f_A)}{\partial \text{Bet}P_m(\omega_j)} \frac{\partial \text{Bet}P_m(\omega_j)}{\partial m(\{\omega_k\})} \quad (4.2) \\ &= -2 \sum_{\emptyset \neq A \subseteq \Omega} [\mathbb{E}_{m_l,p}(f_A) - \mathbb{E}_{m,p}(f_A)] \sum_{j=1}^M \hat{u}_{A,j} \left(\delta_{kj} - \frac{1}{M} \right), \end{aligned}$$

where $\delta_{kj} = 1$ if $k = j$ and $\delta_{kj} = 0$ otherwise. The gradient calculation of $\mathcal{L}(m, m_l)$ w.r.t the parameters in the DS layer are shown in Appendix A, and the gradient with respect to all network parameters can be back-propagated from the output layer to the input layer.

4.2 Experimental evaluation

In this section, we present numerical experiments that demonstrate the advantages of the proposed model. The datasets and metrics are first introduced in Section 4.2.1 and 4.2.2, respectively. Precise and imprecise segmentation results are then reported, respectively, in Sections 4.2.3 and 4.2.4. Finally, novelty detection results are presented in Section 4.2.5.

4.2.1 Datasets

Three benchmark datasets were used in the experiment: Pascal VOC 2012 [30], MIT-scene Parsing [156], and SIFT Flow [128]. These datasets were used to train and test the E-FNCs as well as the P-FNCs for comparison. The Pascal VOC 2012 dataset

Table 4.1: Lists of classes for the Pascal VOC, MIT-scene Parsing and SIFT Flow datasets in the semantic segmentation experiments. Classes in bold characters are included in two or three datasets. Classes with close meanings, such as “minibike” and “motorbike”, are considered as identical.

Dataset	Class list
Pascal VOC 2012	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.

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 datasets are similar to the Pascal VOC 2012 dataset but have, respectively, 150 categories in 22K labeled images and 33 classes in 2688 labeled images. The list of classes for the three datasets is given in Table 4.1. For the Pascal VOC and SIFT Flow datasets, we split each into 50% for training/validation and 50% for testing. For the MIT-scene Parsing dataset, we followed the pre-split protocol for 20k training and 2k for testing. In the study, the validation sets were used to determine hyper-parameters, such as the number of prototypes in each DS layer. In practice, a validation set can also be used to determine the optimal imprecision tolerance degree γ since it can also be considered as a hyper-parameter. The held-out images without labels on the Pascal VOC 2012 and MIT-scene Parsing datasets were not used in the experiments.

There is no confidence value associated with the pixel labels in any of the three

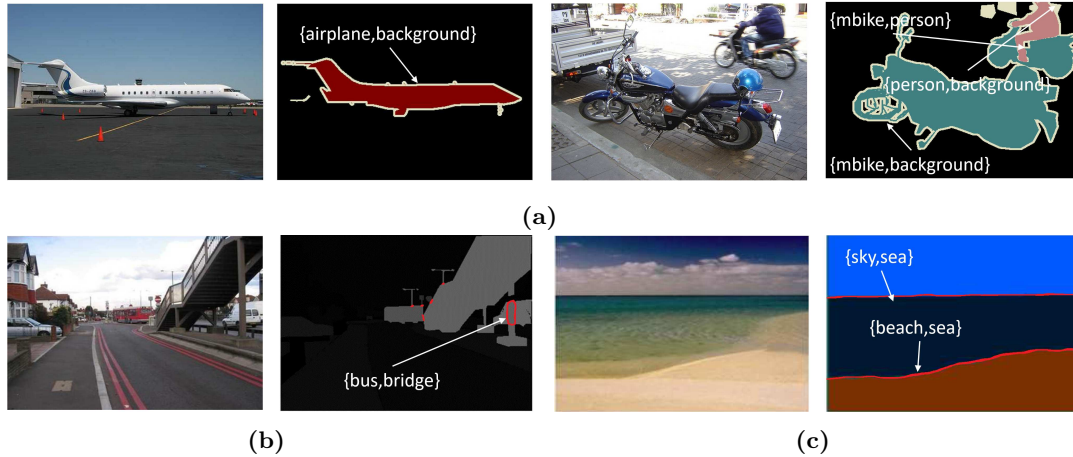


Figure 4.2: Segmentation masks with soft labels: (a) Pascal VOC 2012, (b) MIT-scene Parsing, and (c) SIFT Flow.

datasets. Thus, we defined soft labels for them. For the Pascal VOC 2012 dataset, 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 4.2a. For the MIT-scene Parsing and SIFT Flow datasets with no identified boundary areas, we assigned soft labels to the pixels situated between every two objects, as shown in Figures 4.2b and 4.2c.

A semantic segmentation model should not only be accurate for the classes in the learning set, but it should also be able to detect some objects whose classes are not included in the learning set. To evaluate this novelty detection capacity, we mixed the three datasets: for example, an FCN model trained using the Pascal VOC 2012 dataset was tested on the other two datasets.

4.2.2 Evaluation 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

$$PU = \frac{1}{|T|} \sum_{i=1}^{|T|} \tilde{u}_{A(i), A_*(i)} \quad (4.3)$$

where $A_*(i)$ is the label of pixel i , $A(i)$ is the selected set of classes for pixel i determined by the pignistic criterion, and using the notations introduced in Section 1.3.2, $\tilde{u}_{A(i), A_*(i)}$ 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 \mathbb{U} of size $(2^M - 1) \times M$ defined in Section 1.3.2 should be extended to a matrix \mathbb{U}' of size

Table 4.2: 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

$(2^M - 1) \times (2^M - 1)$ with general term \tilde{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 \tilde{u}_{A,A_*} , we first compute the average of the utilities of selecting subset A when the true class is in A_* as

$$\bar{u}_{A,A_*} = \frac{1}{|A_*|} \sum_{w_k \in A_*} \hat{u}_{A,k}, \quad (4.4a)$$

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

$$\tilde{u}_{A,A_*} = \frac{\bar{u}_{A,A_*}}{\bar{u}_{A_*,A_*}}. \quad (4.4b)$$

Example 4.1 Table 4.2 shows an example of the utility matrix considering soft labels, which is extended from Example 1.1. 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

$$UIoU = \frac{1}{2^{|\Omega|} - 1} \sum_{B \subseteq \Omega} \frac{\sum_{i \in \mathbf{G}^B \cap \mathbf{P}^B} \tilde{u}_{A(i),B}}{|\mathbf{G}^B \cup \mathbf{P}^B|}, \quad (4.5)$$

where $\mathbf{P}^B = \{i : A(i) \cap B \neq \emptyset\}$ is the predicted area containing pixels assigned to a set of classes that intersect B , and $\mathbf{G}^B = \{i : A_*(i) = B\}$ 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 [62, 77, 95].

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) defined in [41] as follows. We define the *prediction confidence* of pixel i as

$$co(i) = BetP_i(A_*(i)) = \sum_{\omega_j \in A_*(i)} BetP_i(\{\omega_j\}), \quad (4.6)$$

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 $(\frac{q-1}{Q}, \frac{q}{Q}]$, $q = 1, \dots, Q$. The average utility and confidence of I_q are defined, respectively, as

$$au(I_q) = \frac{1}{|I_q|} \sum_{i \in I_q} \tilde{u}_{A(i), A_*(i)}, \quad (4.7a)$$

and

$$co(I_q) = \frac{1}{|I_q|} \sum_{i \in I_q} co(i). \quad (4.7b)$$

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

$$ECE = \frac{\sum_{q=1}^Q |I_q| \times |co(I_q) - au(I_q)|}{\sum_{q'=1}^Q |I_{q'}|} \quad (4.8)$$

When only considering precise acts and labels, ECE defined by (4.8) boils down to the original definition in [41].

4.2.3 Precise segmentation results

In precise segmentation, each pixel of an image is assigned to exactly one class, the set of acts being defined as $\mathcal{F} = \{f_{\omega_1}, \dots, f_{\omega_M}\}$. Three datasets without soft labels mentioned in Section 4.2.1 were used to train and test the E- and P-FCNs. The metrics defined in Section 4.2.2 with the utility matrix \mathbb{U} equal to the identity matrix were used for performance assessment.

In the experiment with each dataset, three widely-used encoder-decoder architectures were combined with the DS and utility layers, as shown in Table 4.3. The details of these architectures have been introduced in Section 2.2.3. The numbers of feature maps from the encoder-decoder architectures for the Pascal, MIT and SIFT datasets were, respectively, 64, 128 and 64. The numbers of prototypes in the DS layer for these three datasets were set, respectively, to 75, 300 and 95.

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 architectures. Table 4.3a presents the results of PU and UIoU for the Pascal VOC dataset. E-FCNs achieved higher PU and UIoU than P-FCNs with the same encoder-decoder architecture, which

Table 4.3: Performance evaluation of precise segmentation: (a) Pascal VOC 2012, (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 \pm standard deviation”. The best results for each encoder-decoder architecture are highlighted in bold.

(a)		
	PU	UIoU
P-FCN-32s [77]	0.8912 \pm 0.0019	0.5941 \pm 0.0033
P-FCN-16s [77]	0.9001 \pm 0.0015	0.6243 \pm 0.0025
P-FCN-8s [77]	0.9033 \pm 0.0017	0.6269 \pm 0.0021
E-FCN-32s	0.8973 \pm 0.0021	0.6128 \pm 0.0024
E-FCN-16s	0.9045 \pm 0.0014	0.6304 \pm 0.0019
E-FCN-8s	0.9074 \pm 0.0015	0.6337 \pm 0.0020
(b)		
	PU	UIoU
P-FCN-16s [77]	0.7009 \pm 0.0030	0.2889 \pm 0.0051
P-FCN-8s [77]	0.7128 \pm 0.0024	0.2937 \pm 0.0048
P-FCN-SegNet [1]	0.7153 \pm 0.0023	0.3053 \pm 0.0042
E-FCN-16s	0.7090 \pm 0.0026	0.2919 \pm 0.0048
E-FCN-8s	0.7148 \pm 0.0025	0.2962 \pm 0.0046
E-FCN-SegNet	0.7167 \pm 0.0026	0.3103 \pm 0.0043
(c)		
	PU	UIoU
P-FCN-16s [77]	0.8489 \pm 0.0034	0.3922 \pm 0.0047
P-FCN-8s [77]	0.8525 \pm 0.0032	0.3948 \pm 0.0042
P-FCN-CRF [5]	0.8643 \pm 0.0036	0.4168 \pm 0.0043
E-FCN-16s	0.8521 \pm 0.0030	0.3937 \pm 0.0042
E-FCN-8s	0.8528 \pm 0.0031	0.3961 \pm 0.0040
E-FCN-CRF	0.8649 \pm 0.0035	0.4182 \pm 0.0038

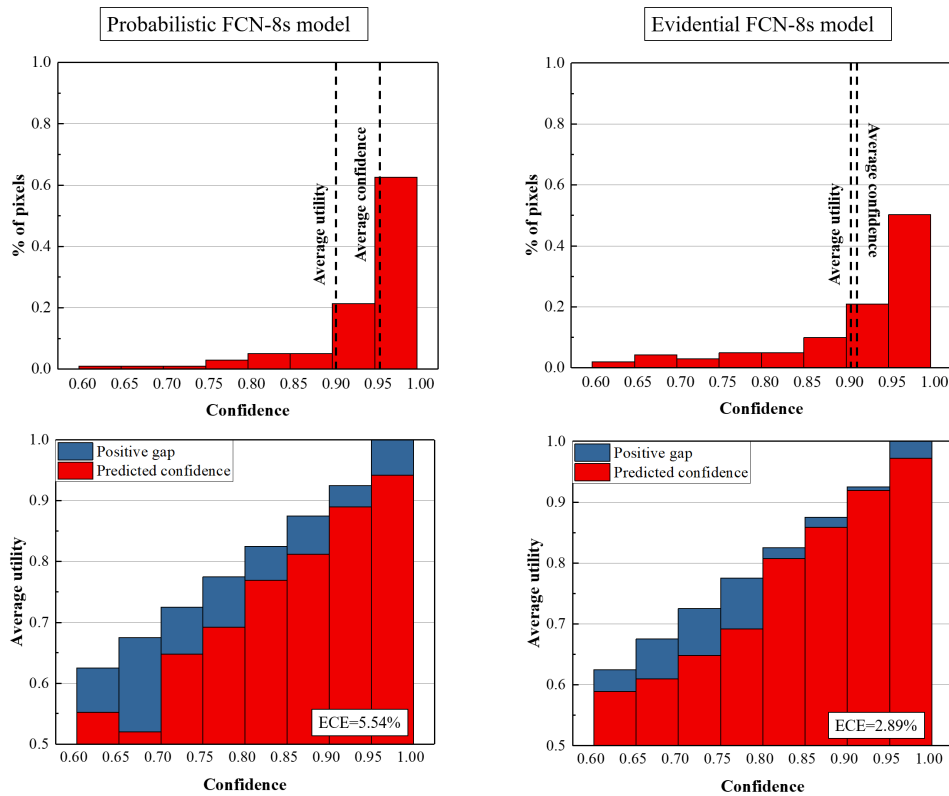


Figure 4.3: Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-8s (left) and E-FCN-8s (right) on the Pascal VOC dataset.

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 datasets as shown, respectively, in Tables 4.3b and 4.3c.

The use of DS and utility layers also makes the FCN models better calibrated. Figure 4.3 presents a visual calibration representation of the FCN-8s models in the Pascal VOC dataset. The top row shows the pixel distribution of prediction confidence (4.7b) 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 bottom row of pixel utility diagrams, which show pixel utility as a function of confidence. The E-FCN-8s model is well calibrated 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-CRF models on the MIT-scene Parsing and SIFT Flow datasets as shown, respectively, in Figures 4.4 and 4.5.

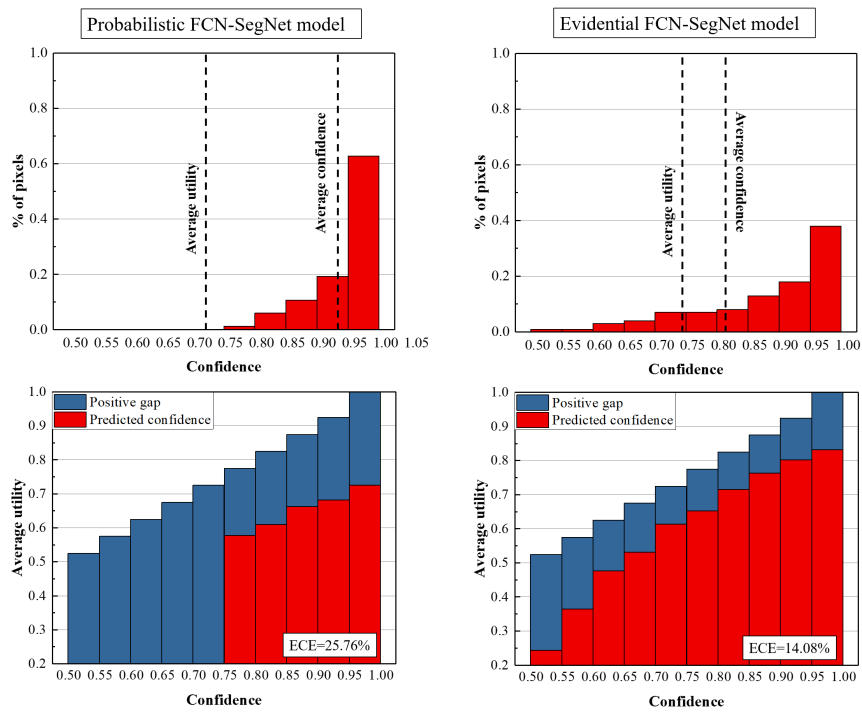


Figure 4.4: Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-SegNet (left) and E-FCN-SegNet (right) on the MIT-scene Parsing dataset.

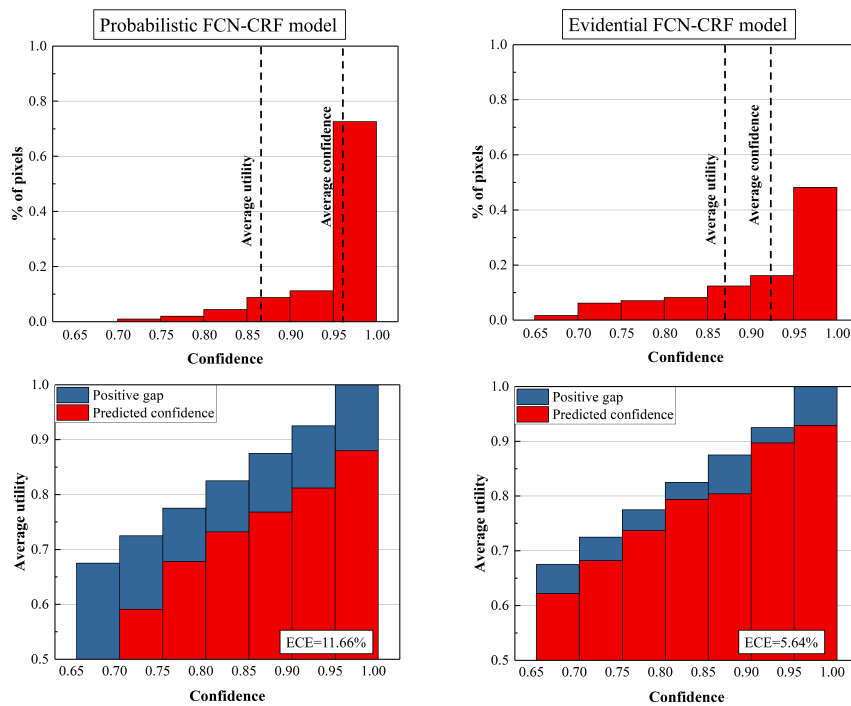


Figure 4.5: Pixel confidence distributions (top) and pixel utility histograms (bottom) for P-FCN-CRF (left) and E-FCN-CRF (right) on the SIFT Flow dataset.

4.2.4 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 $\mathcal{F} = \{f_A, A \in 2^\Omega \setminus \{\emptyset\}\}$, 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 datasets described in Sections 4.2.1 and 4.2.2, respectively. For each dataset, 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 4.2.3 were combined with the DS and utility layers.

Figure 4.6 displays the test results according to PU and UIoU for imprecise segmentation of the Pascal VOC dataset. 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 tend to assign ambiguous pixels to multi-class sets, instead of making precise decisions. Such imprecise assignments avoid pixel-wise misclassification in case of high uncertainty, especially when feature vectors from an encoder-decoder architecture do not contain sufficient information to identify a precise class, and multiple classes have similar probabilities. Figure 4.7 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 4.8-4.9) and SIFT Flow (Figures 4.10-4.11) datasets. We can thus conclude the DS and utility layers improve the performance of the FCN models in imprecise segmentation tasks by allowing us to assign some ambiguous pixels to multi-class sets.

In Figures 4.6, 4.8 and 4.10, 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 4.12 illustrates some segmentation examples generated by the E-FCN-8s model trained on the Pascal VOC dataset with soft labels. The first and second columns of Figure 4.12 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 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. (4.5), 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

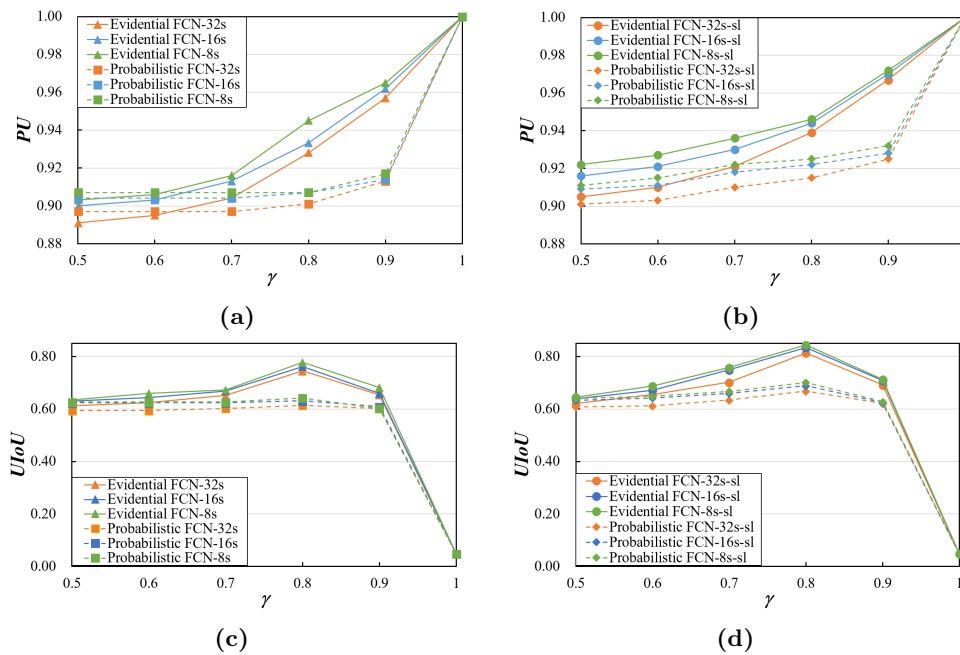


Figure 4.6: Testing PU and UIoU vs. γ on the Pascal VOC dataset. The first and second columns are the models trained with/without soft labels, respectively.

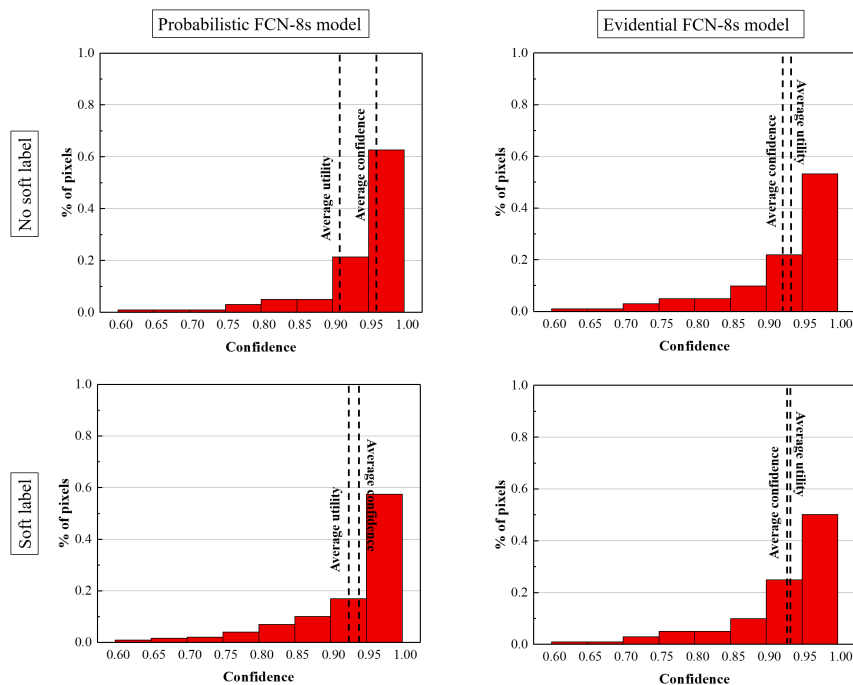


Figure 4.7: Pixel confidence distributions for the P-FCN-8s (left) and E-FCN-8s (right) models on the Pascal VOC dataset without (top)/with (bottom) soft labels.

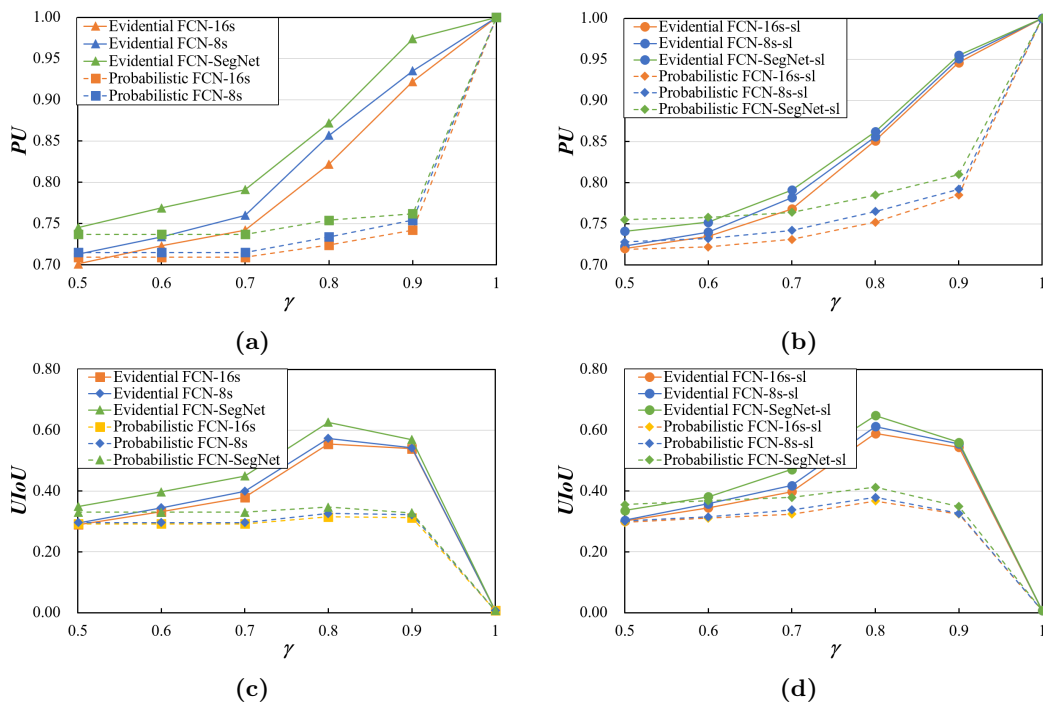


Figure 4.8: Testing PU and UIoU vs. γ on the MIT-scene Parsing dataset. The first and second columns are the models trained with/without soft labels, respectively.

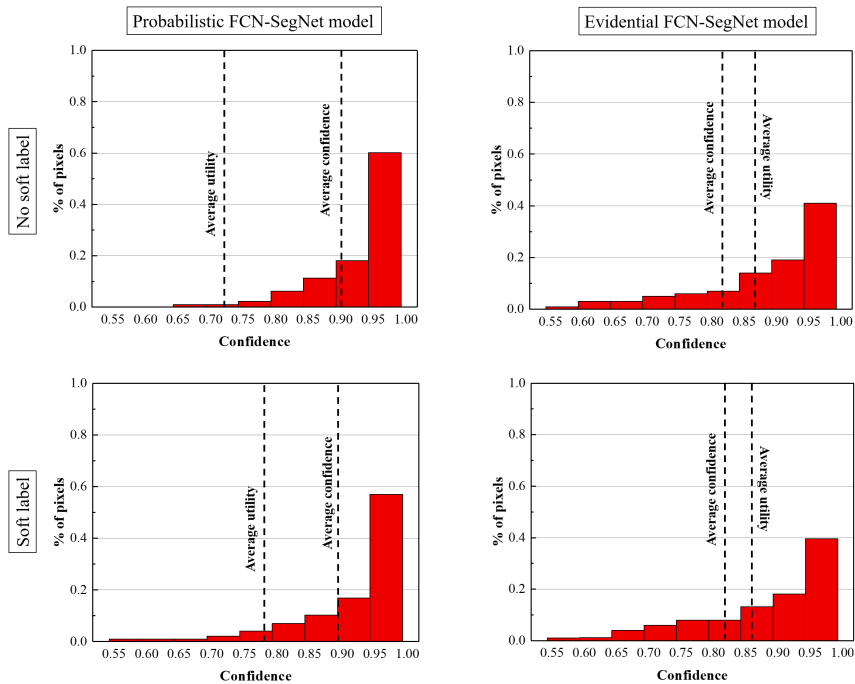


Figure 4.9: Pixel rate histograms for the P-FCN-SegNet (left) and E-FCN-SegNet (right) models on the MIT-scene Parsing dataset without (top)/with (bottom) soft labels.

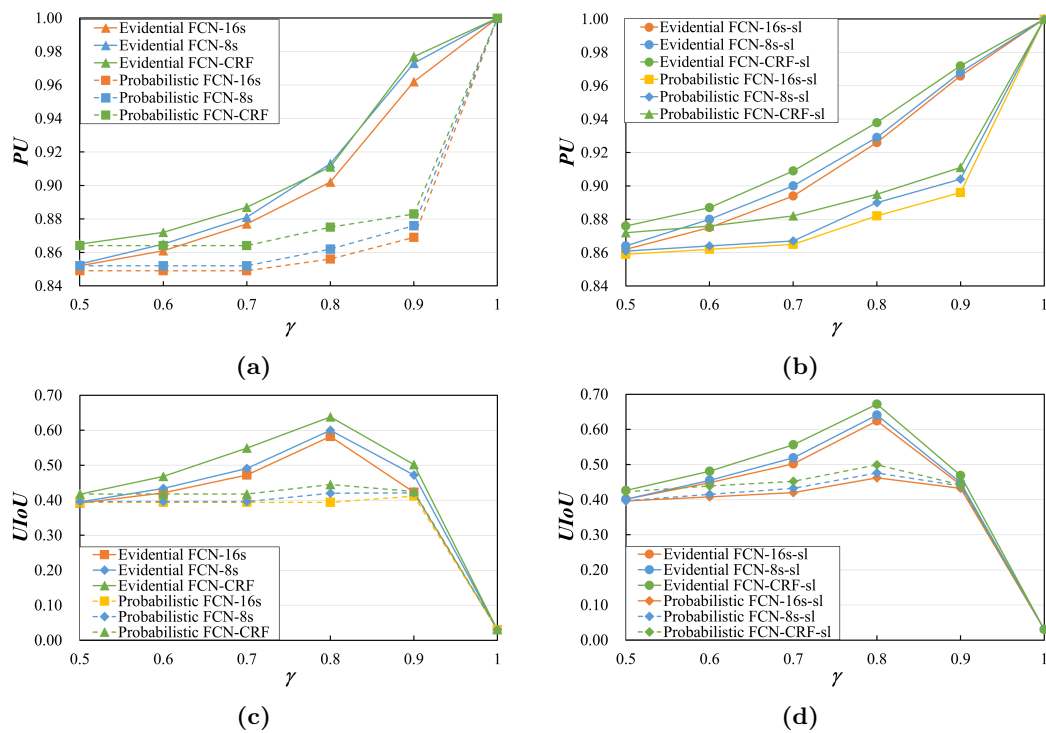


Figure 4.10: Testing PU and UIoU vs. γ on the SIFT Flow dataset. The first and second columns are the models trained with/without soft labels, respectively.

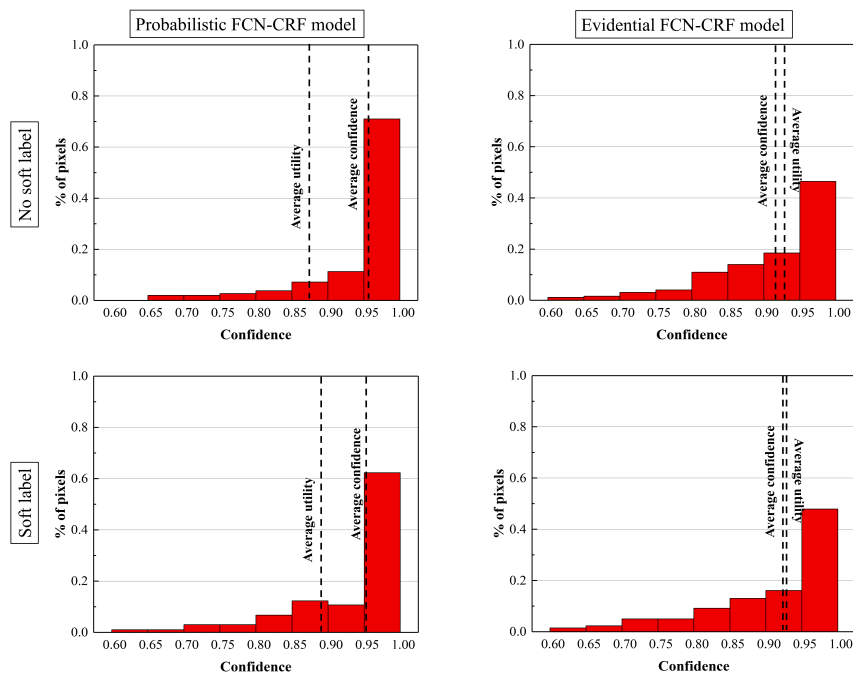


Figure 4.11: Pixel rate histograms for the P-FCN-CRF (left) and E-FCN-CRF (right) models on the SIFT Flow dataset without (top)/with (bottom) soft labels.

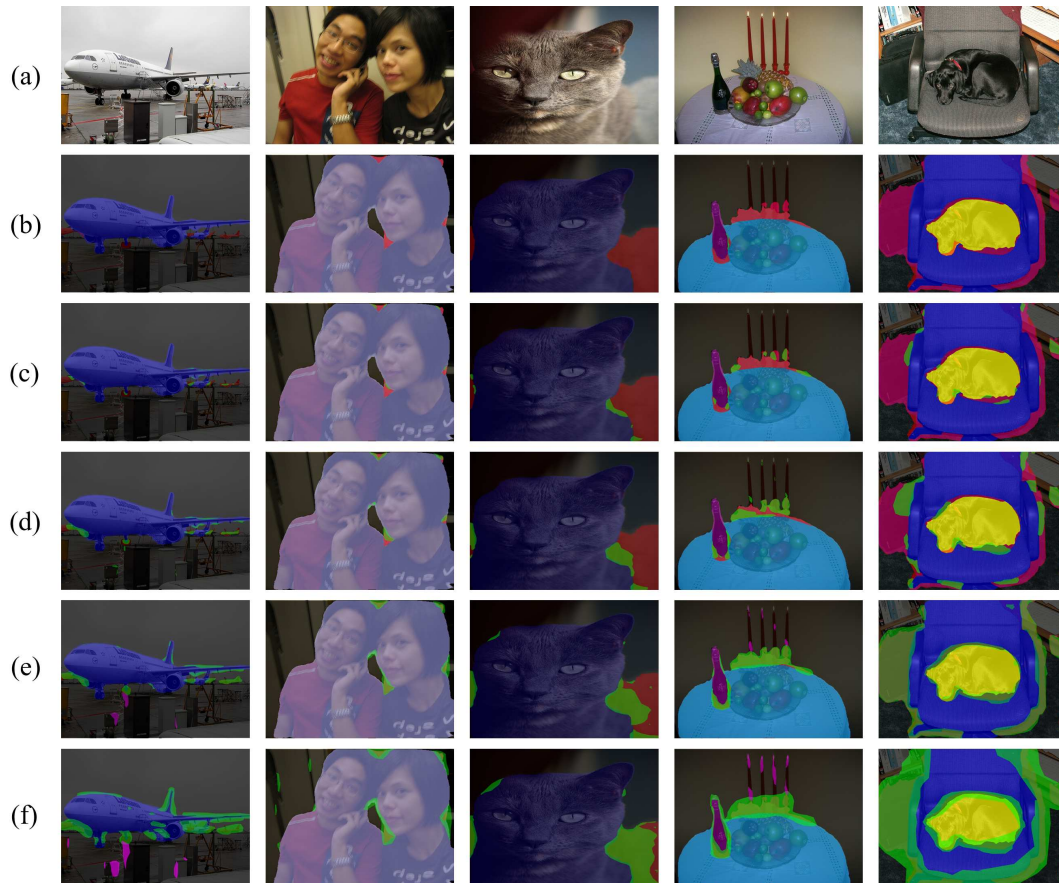


Figure 4.12: Segmentation examples from the Pascal VOC dataset: (a) Original image, (b) Precise segmentation, (c) Imprecise segmentation with $\gamma = 0.6$, (d) Imprecise segmentation with $\gamma = 0.7$, (e) Imprecise segmentation with $\gamma = 0.8$, and (f) Imprecise segmentation with $\gamma = 0.9$. 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.

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 4.6, the FCN models trained by the Pascal VOC dataset 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 4.13 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 confusing information by using soft labels when training FCN models. The improvement of accuracy and calibration due to learning from soft labels can also be found with the MIT-scene Parsing and SIFT Flow

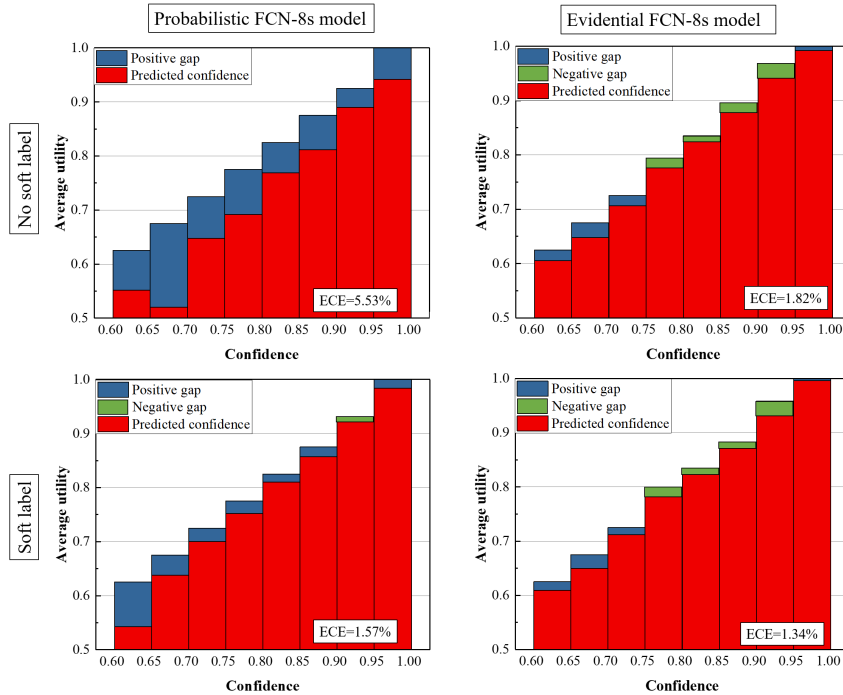


Figure 4.13: Average utility histograms for P-FCN-8s (left) and E-FCN-8s (right) with $\gamma = 0.8$ on the Pascal VOC dataset without (top)/with (bottom) soft labels.

datasets, as shown, respectively, in Figures 4.14 and 4.15. Therefore, we can conclude that the use of soft labels improves the accuracy and calibration of FCN models.

4.2.5 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 4.16, 4.17 and 4.18 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 datasets, and the test set is composed of images from the other two datasets. In each testing set composed of two datasets, only the pixels whose classes are not represented in the corresponding learning set are reported in Figures 4.16-4.18. 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 imprecision tolerance degree. 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 4.19 shows images from the MIT-scene Parsing dataset in which pixels representing ‘bag’, ‘street light’ and ‘ball’ objects are rejected by an E-FCN-8s model trained using the

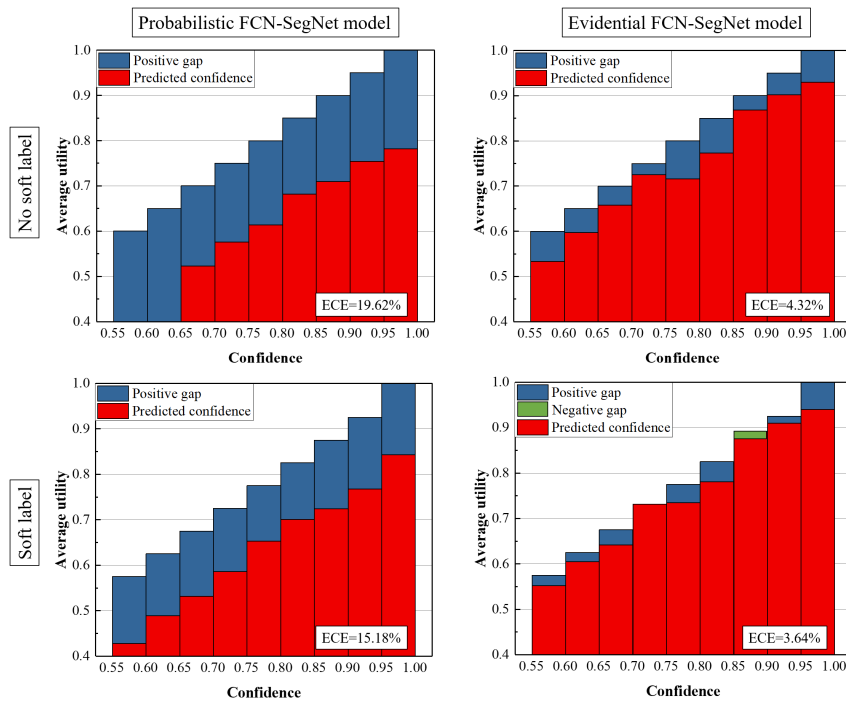


Figure 4.14: Average utility histograms for P-FCN-SegNet (left) and E-FCN-SegNet (right) with $\gamma = 0.8$ on the MIT-scene Parsing dataset without (top)/with (bottom) soft labels.

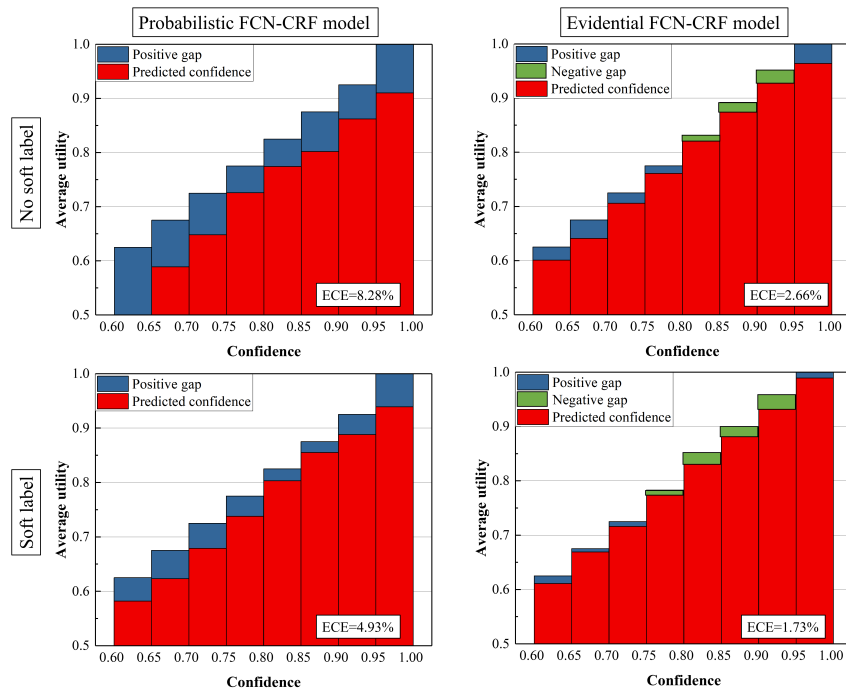


Figure 4.15: Average utility histograms for P-FCN-CRF (left) and E-FCN-CRF (right) with $\gamma = 0.8$ on the SIFT Flow dataset without (top)/with (bottom) soft labels.

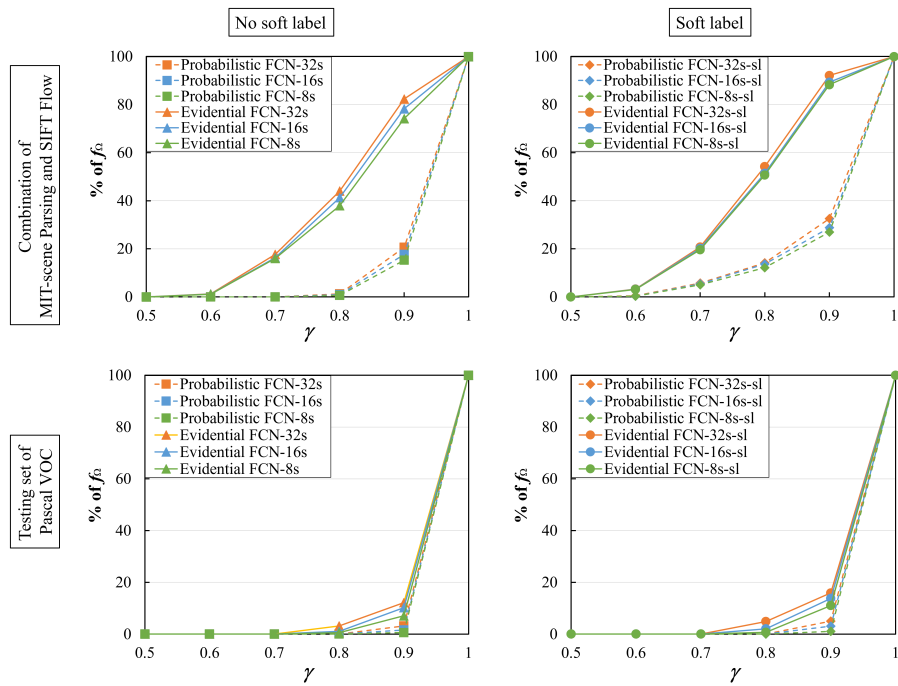


Figure 4.16: Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of MIT-scene Parsing and SIFT Flow datasets (top) and the testing set from the Pascal VOC dataset (bottom) when the learning set is from the Pascal VOC dataset without (left)/with (right) soft labels.

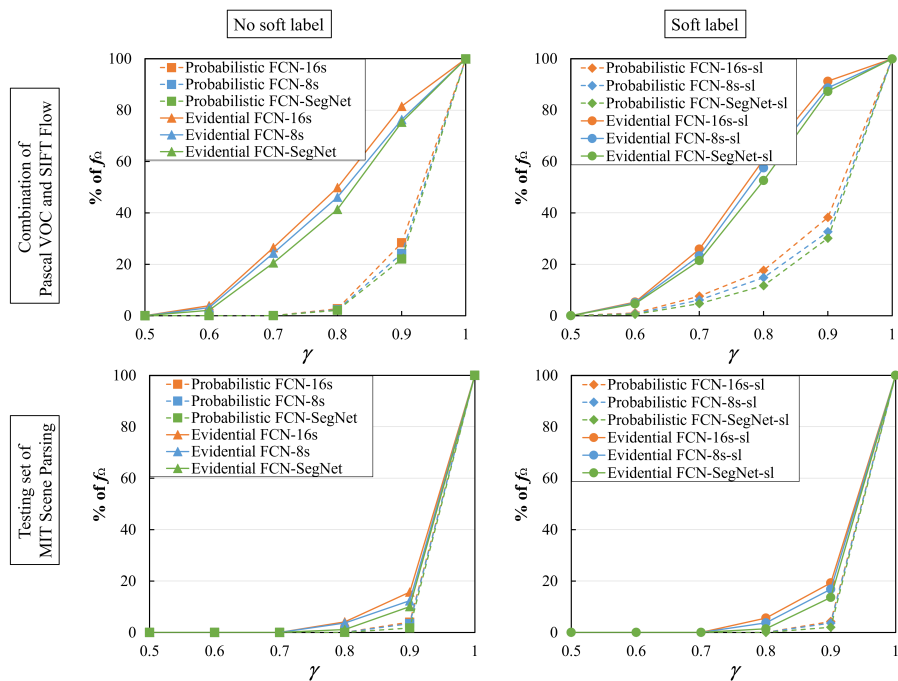


Figure 4.17: Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of Pascal VOC and SIFT Flow (top) and the testing set of the MIT-scene Parsing dataset (bottom) when the learning set is from the MIT-scene Parsing dataset without (left)/with (right) soft labels.

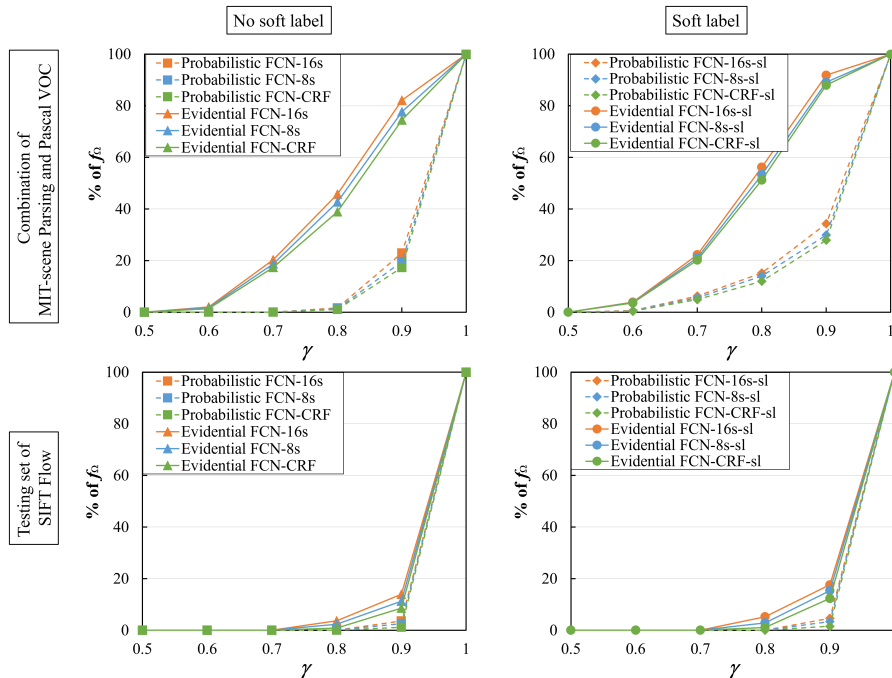


Figure 4.18: Proportion of pixels assigned to Ω as a function of γ for novelty detection on the combination of Pascal VOC and MIT-scene Parsing (top) and the testing set of the SIFT Flow dataset (bottom) when the learning set is from the SIFT Flow dataset without (left)/with (right) soft labels.

Pascal VOC dataset, which does not contain these objects. As shown in Table 4.4, 75.2% of the pixels representing a ball in the MIT-scene Parsing and SIFT Flow datasets 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 learning set with soft labels reject more outliers than those trained without soft labels, as shown in Figures 4.16, 4.17 and 4.18. This is because the use of soft labels makes the FCN models more cautious and better calibrated, as discussed in Section 4.2.4. 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 easily assigned to set Ω . We can thus conclude that soft labels have the potential to enhance novelty detection performance.

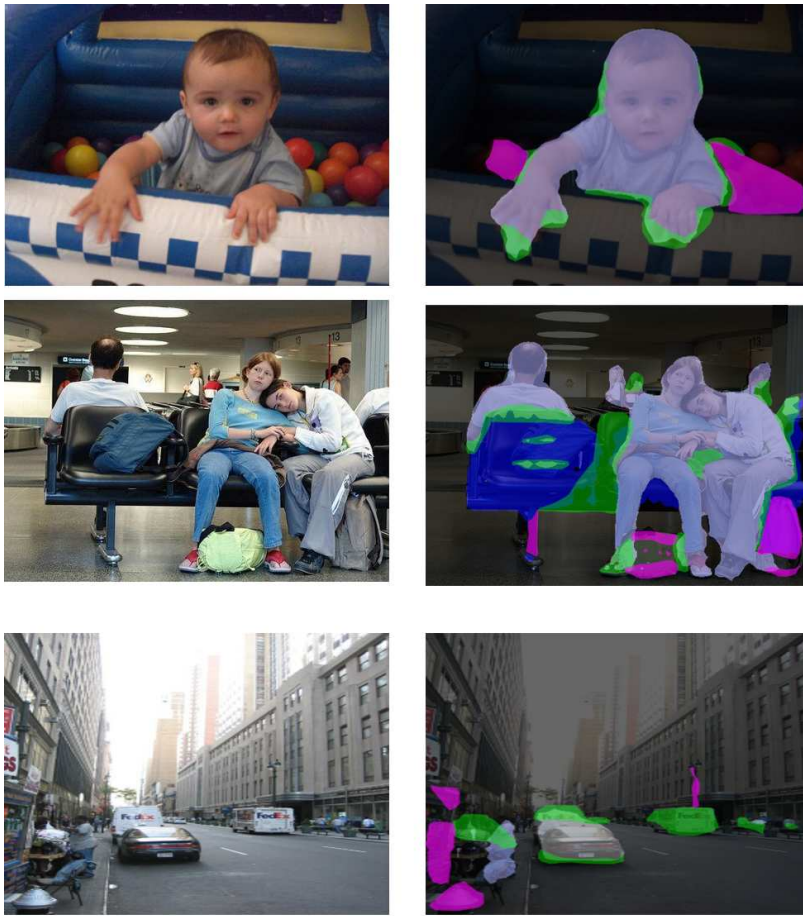


Figure 4.19: Examples of novelty detection from the MIT-scene Parsing dataset and their segmentation masks given by the E-FCN-8s model trained using the Pascal VOC dataset with soft labels when γ equals 0.8. Red masks are pixels incorrectly assigned in the precise segmentation; green masks are pixels 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.

Table 4.4: Percentage of pixels from some unknown classes in the MIT-scene Parsing and SIFT Flow datasets classified by an E-FCN-8s model trained on the Pascal VOC dataset 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

4.3 Conclusion

In this chapter, we extended the applications of evidential deep neural networks to pixel-wise 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 chapter 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 by converting imprecise and unreliable label data into mass functions. Additionally, the proposed approach makes it possible to reject outliers together with ambiguous pixels when the imprecision tolerance degree is between 0.7 and 0.9.

Chapter 5

Evidential fusion of heterogeneous deep neural networks

One challenge in machine learning is to combine the existing models trained from heterogeneous datasets for obtaining a more general one. However, the problems of data uncertainty, especially the partial and imperfect outputs of deep neural networks (DNNs), make it difficult to fuse heterogeneous networks. The proposed combined framework of Dempster-Shafer theory (DST) and DNNs provides an approach to solve the problem.

In this chapter, we extend the proposed combined framework into the evidential fusion of heterogeneous DNNs [132], such as introduced in Chapters 3 and 4. In this approach, several pre-trained evidential DNNs extract features from input data and convert them into mass functions on different frames of discernment. A fusion module then aggregates these mass functions using Dempster's rule. An end-to-end learning procedure allows us to fine-tune the overall architecture using a learning set with soft labels. This approach not only slightly improves the performances of pattern classification and semantic segmentation but, above all, it also allows us to combine heterogeneous DNNs pre-trained with different sets of classes at any stage to obtain a more general network. In addition, the approach provides a new way to combine simple and shallow networks for a complicated task, which has the potential to simplify training and avoid the use of very deep networks.

We organize the chapter as follows. Section 5.1 provides the background and motivation. The proposed information-fusion approach is then introduced in Section 5.2. Section 5.3 presents numerical experiments on the multi-network fusion, which demonstrate the flexibility of the proposed approach on the information fusion of different pre-trained DNNs with different sets of classes. Section 5.4 reports the experiments on training and combining shallow networks to solve complicated tasks, showing the potential of the approach to simplify the training difficulty. Finally, we conclude the chapter in Section 5.5.

5.1 Introduction

DNNs, such as CNNs [70] and FCNs [77] recalled in Chapter 2, have been widely used for supervised learning (e.g., pattern classification and semantic segmentation) and have achieved remarkable success. Such networks learn reliable and robust features from several datasets with different sets of classes and different granularities. For instance, an FCN learns the features from the Cityscapes dataset [13] for understanding the semantic urban scene, while another is trained for indoor- and outdoor-object segmentation using the Pascal VOC 2012 dataset [30]. However, this problem requires to combine DNN outputs with different levels of granularities. For example, given a new image, a network outputs class probabilities for the CIFAR-10 dataset [63], including a probability of class “bird”. However, compared to the probabilistic outputs of a network trained by the Caltech-UCSD Birds 200 dataset with 200 species of birds [141], the output information of the CIFAR-10 network is partial and imperfect. The problem then arises of combining networks trained from such heterogeneous datasets. Unfortunately, Bayesian probability theory is not flexible enough to fuse heterogeneous networks and allow the introduction of new datasets with different sets of classes at any stage [146].

DST provides a way to address the classifier-fusion problem. One of the applications of DST is evidential classifier fusion, in which classifier outputs are transformed into mass functions and aggregated by Dempster’s rule [102, 146]. The information-fusion capacity of DST makes it possible to combine DNNs. In recent few years, some studies using DST to combine DNNs have been reported, but most of these studies consider a fixed frame of discernment and do not address the fusion of classifiers trained with different sets of classes. For example, Soua et al. [119] use deep belief networks to independently predict traffic flow using streams of data and event-based data, and then update the beliefs from the networks by Dempster’s conditional rule to achieve enhanced prediction. Tian et al. [127] also use Dempster’s rule to fuse the beliefs from some deep-learning models with different types of data to detect anomalous network behavior patterns. Das et al. [14] use CNNs to perform super-pixel semantic segmentation with three levels; DST is then utilized to combine the segmentation results of the three levels into reliable ones. Besides, Guo et al. [43] propose an “iFusion” framework, which uses Dempster’s rule to combine different deep-learning discrimination models taking real-time or heterogeneous data as input. Similar studies using DST for the fusion of DNNs can also be found in the fields of posture recognition [72], remote-sensing images processing [28, 80], and emotion classification [147].

In this chapter, we extend the proposed combined framework of DST and DNN to the evidential fusion of heterogeneous DNNs [132], such as the ones described in Chapters 3 and 4. In detail, we present a modular fusion approach based on DST to combine different DNNs. Several pre-trained DST-based DNNs extract features

from input data and convert them to mass functions defined on different frames of discernment. A fusion module then aggregates these mass functions using Dempster’s rule. The aggregated mass function is used for decision-making in a refined frame. An end-to-end learning procedure allows us to fine-tune the overall architecture using a learning set with soft labels, which further improves the performance. Our two main contributions in this chapter are the following:

1. *Combining different pre-trained networks for a more general one.* The proposed approach makes it possible to combine DNNs trained from heterogeneous databases with different sets of classes at any stage to obtain a more general one, which provides a way to fuse the partial and imperfect outputs of DNNs. The combined network has at least as good performance as those of the individual networks on their respective datasets.
2. *Training shallow networks for a complex task.* Many studies have demonstrated that depth tends to improve network performances. However, deeper networks are more non-linear, increasing the training difficulty and requiring higher computing demands. Some approaches have to be used for gradient-based training, such as knowledge distillation (Figure 2.4) and skip connections (Figure 2.5). The proposed information-fusion approach provides a way to train simple and shallow networks to solve a complex task, which avoids the use of very deep networks and has the potential to make training easier.

5.2 Fusion approach

In this section, we describe the proposed approach for the fusion of evidential DNNs. The overall framework is first described in Section 5.2.1. The end-to-end learning procedure is then introduced in Section 5.2.2.

5.2.1 Evidential fusion approach

The proposed approach combines different pre-trained evidential DNNs by adding an information-fusion module at the belief-function outputs of these evidential DNNs, such as a Dempster-Shafer (DS) layer in an E-CNN (Chapter 3) or an E-FCN (Chapters 4). The architecture of the proposed approach, illustrated in Figure 5.1, can be defined by a three-step procedure. Here, we take the fusion of evidential CNN classifiers for object classification as an example to describe the three steps. The method is called “*mass-fusion evidential CNN (MFE-CNN) classifier*”.

Step 1: An input image is fed into V pre-trained evidential CNN architectures, as described in Section 3.1. The v -th backbone of the evidential CNN architecture, $v = 1, \dots, V$, extracts a feature vector from the input. The vector is then fed into a DS layer for constructing mass functions. Each unit in this layer computes

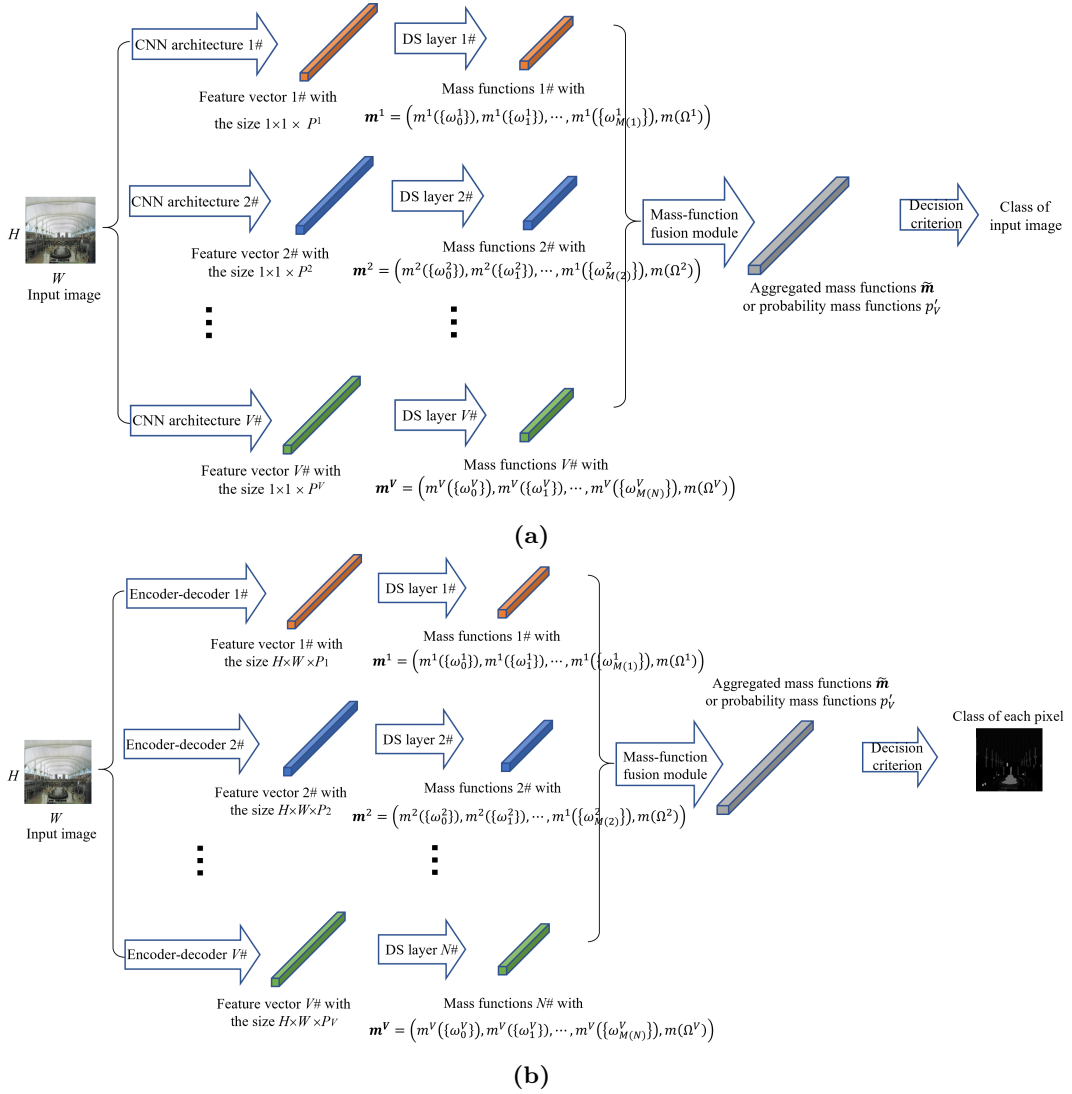


Figure 5.1: Architecture of the evidential information-fusion approach: mass-fusion evidential CNN (MFE-CNN) classifier for object classification (a) and mass-fusion evidential fully convolutional network (MFE-FCN) for semantic segmentation (b). The main difference between the two models is that the stacked CNN stages of the former one output feature vector for image-level classification, while the encoder-decoder architectures in the latter one output pixel-wise features for semantic segmentation.

a mass function on the frame of discernment Ω^v composed of $M(v)$ classes $\omega_1^v, \dots, \omega_{M(v)}^v$ and an “anything else” class ω_0^v , based on the distance between the feature vector and a prototype. The mass functions computed by each of the hidden units are then combined by Dempster’s rule. Given the design of the DS layer, the focal sets of the combined mass function m^v are the singletons $\{\omega_k^v\}$ for $k = 1, \dots, M(v)$ and Ω^v . Thus, the outputs after this first step are the V mass functions m^1, \dots, m^V defined on V compatible frames $\Omega^1, \dots, \Omega^V$. The mass m^v characterizes the v -th architecture’s belief about the probability of the sample class on Ω^v and quantifies the uncertainty in the object representation of the convolutional architecture. Compared to a probabilistic CNN architecture, which extracts a feature vector by a convolutional architecture and imports it into a softmax layer to generate a probability distribution, an evidential CNN architecture can improve the classification performance and output cautious DS-based mass functions. The details and advantages of the evidential CNN architectures have been exposed in Chapter 3.

Step 2: A belief-function fusion module aggregates the V mass functions. Let Ω^0 be a common refinement of the V compatible frames $\Omega^1, \dots, \Omega^V$. Each frame Ω^v can be refined to the common one Ω^0 using (1.12). We can then compute the vacuous extension $m^{\Omega^v \uparrow \Omega^0}$ in Ω^0 using (1.13). We simplify the notation $m^{\Omega^v \uparrow \Omega^0}$ as $m^{v \uparrow 0}$. A combined mass function \tilde{m} on Ω^0 is computed as the orthogonal sum of the V vacuous extensions $\tilde{m} = m^{1 \uparrow 0} \oplus \dots \oplus m^{V \uparrow 0}$. This final output of the belief-function fusion module represents the total evidence about the class of the input image based on the outputs of the V evidential CNN architectures. Dempster’s rule can be computed using contour functions, as explained in Section 1.2.1, providing a simple way to aggregates the V mass functions. Each vacuous extension $m^{v \uparrow 0}$ is approximated by a probability mass function $p_{m^{v \uparrow 0}}$ using the plausibility transformation (1.10), and these probability mass functions are then combined as $p_{m^1 \oplus \dots \oplus m^V}$ using (1.11). We simplify the notation $p_{m^1 \oplus \dots \oplus m^V}$ to p_V . This contour-based combination rule can be performed in $O(K)$ arithmetic operations by multiplying these probability mass functions element-wise. As will be discussed in Sections 5.3 and 5.4, this approach makes it possible to (a) incorporate the new classes of objects while retaining the good performance of these CNN architectures, and (b) to fuse shallow networks to solve a complicated task, instead of using a hard-to-train deep neural network.

Step 3: One of the evidence-theoretic rules recalled in Section 1.3 should be selected to make a decision using the aggregated mass function \tilde{m} or probability mass function p_V . The pignistic or generalized Hurwicz criteria can be used for decision-making allowing for set-valued assignments, which have been discussed in Chapters 3 and 4. In this chapter, we only focus on precise classification

using p_V . The sample is assigned to class ω such that

$$\omega = \arg \max_{\omega_i \in \Omega^0} p_V(\omega_i).$$

The procedure of fusing different FCNs for semantic segmentation, called the *mass-fusion evidential fully convolutional network (MFE-FCN)*, is similar to the procedure just described, except that the encoder-decoder architectures in the MFE-FCN provide pixel-wise features in the first step, instead of image-level feature vectors.

5.2.2 Learning with soft labels

Before fusion, evidential DNNs are trained using their individual learning sets with different frames of discernment. However, these frames are different from the refined one, even though the semantics of some classes do not change. The refined frame is more detailed, and the learned parameters in the ready-trained networks may not be very suitable for the new task. Thus, an end-to-end learning procedure should be applied to fine-tune all the parameters in the combination of evidential DNNs using a learning set that is made up of these individual learning sets.

In the end-to-end training procedure, the learning sets of different pre-trained DNNs are merged into a single one. Some labels then become imprecise after merging; they are referred to as *soft labels*. For example, the “bird” label in the CIFAR-10 [63] dataset becomes imprecise when the dataset is merged with the Caltech-UCSD Birds 200 dataset [141] containing 200 bird species. To fine-tune different DNNs using a learning set with soft labels, we define a label as a nonempty subset $A \in 2^\Omega \setminus \emptyset$ of classes an image may belong to. Label A indicates that the true class is known to be one element of set A , but one cannot determine which one specifically if $|A| > 1$.

In the fine-tuning phase, taking an MFE-CNN classifier for object classification as an example, its parameters are initialized by the parameters in the pre-trained evidential CNNs. Given a learning image with nonempty soft label $A_* \subseteq \Omega^0$, the MFE-CNN classifier outputs a probability mass function p_V . We first normalize the mass as

$$p'_V(\omega_i) = \frac{p_V(\omega_i)}{\sum_{j=1}^{M^0} p_V(\omega_j)}, \quad i = 1, \dots, M^0, \quad (5.1)$$

where M^0 is the number of classes in the common frame Ω^0 . We then define the loss as:

$$\mathcal{L}(p'_V, A_*) = -\log \sum_{\omega \in A_*} p'_V(\omega). \quad (5.2)$$

This loss function is minimized when the normalized probability mass p'_V of soft label A_* equals 1. The classifier tries to maximize the sum of the probability mass of the classes in soft label A_* , indicating the classifier tends to believe that the true class is in A_* but cannot determine which one. For example, given a sample with soft label “bird”, a classifier should predict a high value of $\sum_{\omega \in \text{bird}} p'_V(\omega)$, which means that

the classifier believes that the true class is one of the bird species. The gradient of this loss w.r.t all network parameters can be back-propagated from the output layer to the input layer, as discussed in Sections 3.1.2 and 4.1.2, and Appendix A.

5.3 Experiments on multi-model fusion

In this section, we study the performance of the above fusion method on multi-model fusion tasks through two image-classification experiments (Sections 5.3.1 and 5.3.2) and two semantic-segmentation experiments (Sections 5.3.3 and 5.3.4).

5.3.1 Image-classification experiment #1

Experiment setting. Three datasets are considered in this first experiment: Tiny ImageNet [17], Flower-102 [93], and CIFAR-10 [63]. The Tiny ImageNet dataset contains 110k labeled images of 200 classes with the size of $64 \times 64 \times 3^1$. Each class has 500 training images and 50 validation images². The Flower-102 dataset consists of 102 flower categories and each are represented by 40 to 258 images. The training/validation set contains 40 images per class (totaling 4080 images each), and the test set consists of the remaining 4,129 images (minimum 10 per class). The CIFAR-10 database with ten classes was pre-split into 50k training and 10k testing images. Figure 5.2 shows the semantic relationship of the classes in the three datasets. We added an “anything else” class to each dataset to make the three frames compatible. After merging the three datasets, we obtained 154,080 training samples and 24,129 testing samples for, respectively, fine-tuning and performance evaluation.

For a testing set T with soft labels, the *average error rate* is defined as

$$AE(T) = 1 - \frac{1}{|T|} \sum_{i \in T} \mathbb{1}_{A(i)}(\hat{\omega}(i)), \quad (5.3)$$

where $A(i)$ is the soft label of sample i , $\hat{\omega}(i)$ is the predicted class, and $\mathbb{1}_{A(i)}$ is the indicator function of set $A(i)$.

We designed three different MFE-CNN classifiers for the experiment. Each classifier consists of three pre-trained evidential CNN architectures with 128 output units, introduced in Section 2.1.4. Table 5.1a presents the optimized numbers of prototypes in the DS layers for the three datasets. We compared the MFE-CNN classifiers to four classifier fusion systems with the same CNN architectures:

Probability-to-mass fusion (PMF) [146]: we feed the feature vector from each CNN backbone v into a softmax layer to generate a Bayesian mass function on

¹The class list of the Tiny ImageNet dataset can be found in <https://github.com/rmccorm4/Tiny-Imagenet-200/blob/master/sets/words200.txt>.

²The Tiny ImageNet dataset contains 10k held-out and unlabeled images for testing only available for [the Stanford CS231n course](#). In the thesis, we used the validation set for testing and the training set for training and validation, as done in [27, 81, 103, 133].

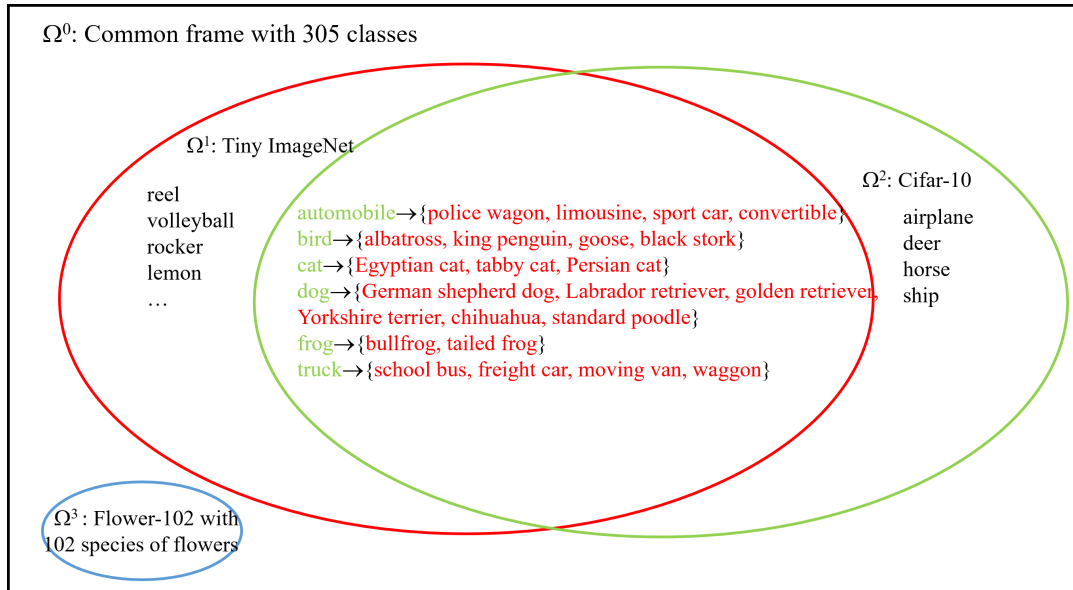


Figure 5.2: Semantic relationship of the classes in the Tiny ImageNet (red), Flower-102 (blue), and CIFAR-10 (green) datasets. The classes in black are unique to one of the three datasets. The colored classes with black arrows indicate the frame refinement. For example, $\text{automobile} \rightarrow \{\text{police wagon, limousine, sport car, convertible}\}$ means that the “automobile” class in the CIFAR-10 dataset becomes the soft label $\{\text{police wagon, limousine, sport car, convertible}\}$ with the classes in the Tiny ImageNet dataset after refinement.

Ω^v . The mass functions from the three CNNs are then aggregated by Dempster’s rule. (It should be noted that the vacuous extension of each Bayesian mass function in the common refinement Ω^0 is no longer Bayesian.)

Bayesian-fusion (BF) [140]: the feature vector from each evidential CNN architecture is converted into a Bayesian mass function on the common frame Ω^0 , by equally distributing the mass of a focal set to its elements; the obtained Bayesian mass functions are combined by Dempster’s rule. This procedure is equivalent to Bayesian fusion.

Probabilistic feature-combination (PFC) [92]: the three feature vectors are concatenated to form a new vector of length 384, fed into a softmax layer to generate the probability distribution on the common frame.

Evidential feature-combination (EFC): feature vectors are concatenated as in the above PFC approach, but the aggregated vector is fed into a DS layer to generate an output mass function. The dimension of the aggregated vector and the number of prototypes are, respectively, 384 and 400, to obtain an optimal performance of the EFC-CNN classifier.

Results and discussion. Table 5.2 shows the average test error rates of the evidential and probabilistic CNN classifiers trained from each of the three datasets,

Table 5.1: Numbers of prototypes in Dempster-Shafer layers: image-classification experiment #1 (a), image-classification experiment #2 (b), semantic-segmentation experiment #1 (c), and semantic-segmentation experiment #2 (d).

(a)				(b)			
	Tiny ImageNet	Flower-102	Cifar-10		Cifar-10	CUB	Oxford-IIIT Pet
FitNet-4	360	230	70	FitNet-4	70	350	80
UPANets	380	220	60	ResNet-101	70	300	65
ResNet-101	400	230	65	ViT-L/16	65	330	85

(c)				(d)			
	Pascal VOC	Cityscapes	Stanford background		Pascal VOC	MIT-scene	SIFT Flow
FCN-8s	75	90	30	FCN-8s	75	300	95
FCN-SegNet	65	80	25	FCN-SegNet	65	280	90
FCN-CRF	75	80	35	FCN-CRF	75	320	100

as well as the performances of the different fusion strategies (with and without fine tuning) on each individual dataset, and on the union of the three datasets.

Looking at the performance of the MFE strategy, we can see that, after fusion, the error rates on the Tiny ImageNet and CIFAR-10 datasets decrease, but the ones on the Flower-102 dataset do not change. More precisely, as shown in Table 5.3, the error rates for some classes (e.g., “cat”, “dog” and “bird”) on the CIFAR-10 database decrease, but the ones of other classes do not change after fusion. The classes whose errors decrease are the same as the classes in green in Figure 5.2. This is because an evidential CNN classifier trained by the Tiny ImageNet dataset can provide useful and detailed information for the image classification on the CIFAR-10 dataset. For instance, a “cat” image may be misclassified into the “dog” class when only using a CIFAR-10 classifier, but the image gets a high degree of belief in the “tabby cat” class from a Tiny-ImageNet classifier. After aggregating the mass functions from the CIFAR-10 and Tiny-ImageNet classifiers, the image is finally classified into the “tabby cat” class, which is a correct decision. The reason for the error decrease with the Tiny ImageNet datasets is a little different from the one with the CIFAR-10 dataset. Table 5.4, which shows examples of probability mass functions computed by the different classifiers, allows us to explain the reason. The first example from the Tiny ImageNet dataset labeled as “Egyptian cat” (a species of “cat”) is misclassified as a “chihuahua” (a species of “dog”) using only the probability mass function from the classifier trained from this dataset, but the decision is corrected after the evidential fusion because the mass function provided by the classifier trained from the CIFAR-10 data supports the “cat” class. A similar phenomenon can also be found in the second example. Thus, the CIFAR-10 classifier can provide useful information of some super-classes (e.g., “cat” and “dog”) for the Tiny-ImageNet classifier. However, the mass from the CIFAR-10 classifier does not help when samples are misclassified into some sub-classes by the Tiny-ImageNet classifier. Consider the third example. An image of the “bull frog” class is misclassified as “tailed frog” by the Tiny-ImageNet classifier, but the CIFAR-10 classifier can only provide its belief that


Table 5.2: Average test error rates (in percent) of different classifiers on the classification experiment #1: FitNit-4 (a), UPANets (b), and ResNet-101 (c). “E2E” stands for fine-tuned classifiers. The lowest error rates are in bold and second low are underlined. The initialized weights in ResNet-101 backbone is from a model pre-trained by the ImageNet-21k dataset [105], and we then fine-tune the backbone on the Tiny ImageNet dataset and the union of the three datasets.


(a)					
	Classifier	Tiny ImageNet	Flower-102	CIFAR-10	Overall
Before fusion	E-FitNit-4	41.21	13.06	6.50	-
	P-FitNit-4 [106]	41.38	13.28	6.58	-
After fusion	MFE-FitNit-4	<u>40.92</u>	<u>13.08</u>	<u>5.61</u>	<u>23.37</u>
	PMF-FitNit-4	41.16	13.27	5.84	23.59
	BF-FitNit-4	41.84	13.70	8.03	24.42
	E2E MFE-FitNit-4	40.80	13.04	5.52	23.29
	E2E PMF-FitNit-4	41.05	13.27	5.68	23.52
	E2E BF-FitNit-4	41.58	13.61	7.22	24.14
	E2E PFC-FitNit-4	41.49	13.12	6.33	23.75
	E2E EFC-FitNit-4	41.86	13.88	6.94	24.32
(b)					
	Classifier	Tiny ImageNet	Flower-102	CIFAR-10	Overall
Before fusion	E-UPANets	34.61	9.77	5.98	-
	P-UPANets [133]	34.72	9.83	6.05	-
After fusion	MFE-UPANets	<u>34.42</u>	<u>9.77</u>	<u>5.41</u>	<u>19.27</u>
	PMF-UPANets	34.61	9.83	5.92	19.46
	BF-UPANets	35.59	11.68	8.28	21.03
	E2E MFE-UPANets	34.31	9.77	5.35	19.21
	E2E PMF-UPANets	34.49	9.83	5.84	19.40
	E2E BF-UPANets	35.19	11.08	7.74	20.53
	E2E PFC-UPANets	34.79	12.12	6.48	20.58
	E2E EFC-UPANets	35.38	16.58	7.32	22.82
(c)					
	Classifier	Tiny ImageNet	Flower-102	CIFAR-10	Overall
Before fusion	E-ResNet-101	18.66	4.68	4.61	-
	P-ResNet-101 [81]	18.70	4.69	4.66	-
After fusion	MFE-ResNet-101	18.52	4.68	<u>3.94</u>	<u>10.31</u>
	PMF-ResNet-101	18.54	4.69	4.42	10.40
	BF-ResNet-101	19.18	5.07	6.04	11.10
	E2E MFE-ResNet-101	<u>18.50</u>	4.67	3.82	10.27
	E2E PMF-ResNet-101	18.49	<u>4.68</u>	4.28	10.35
	E2E BF-ResNet-101	18.87	4.99	5.74	10.89
	E2E PFC-ResNet-101	18.59	5.74	4.89	10.94
	E2E EFC-ResNet-101	21.68	5.46	7.57	12.56


Table 5.3: Test error rates (in percent) before and after information fusion on CIFAR-10 using the FitNit-4 architecture. The classes whose error rates decrease are highlighted in bold.


	Classifier	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Before fusion	E-FitNit-4	2.4	3.9	6.4	13.5	9.0	10.1	5.6	6.8	3.5	2.7
	P-FitNit-4	1.6	2.6	8.7	15.7	9.6	12.5	4.2	5.3	1.9	2.6
After fusion	E2E MFE	2.2	3.9	1.9	6.3	8.5	3.9	5.5	6.5	3.5	2.7
	E2E PMF	1.6	2.5	5.0	12.8	9.0	9.2	4.2	5.3	1.8	2.6
	E2E BF	1.5	2.5	8.1	14.0	9.0	11.0	4.1	5.2	1.8	2.5

Table 5.4: Examples of probability mass functions on the Tiny ImageNet dataset before and after fusion by the MFE strategy. Only some masses before and after fusion are shown for lack of space. The notations ω_0^i , $i = 0, \dots, 3$, stands for the “anything else” class to make the frames compatible.

Instance/label	Before fusion			p' on Ω^0 after fusion
	p' from Tiny ImageNet	p' from CIFAR-10	p' from Flower102	
 Egyptian cat	$p'(\text{Egyptian cat}) = 0.472$	$p'(\text{cat}) = 0.873$	$p'(\text{buttercup}) = 0.001$	$p'(\text{Egyptian cat}) = 0.860$
	$p'(\text{chihuahua}) = 0.511$	$p'(\text{dog}) = 0.116$	$p'(\text{camellia}) = 0$	$p'(\text{chihuahua}) = 0.125$

	$p'(\omega_0^1) = 0.001$	$p'(\omega_0^2) = 0.001$	$p'(\omega_0^3) = 0.998$	$p'(\omega_0^0) = 0.001$
 king penguin	$p'(\text{king penguin}) = 0.453$	$p'(\text{bird}) = 0.732$	$p'(\text{buttercup}) = 0$	$p'(\text{king penguin}) = 0.988$
	$p'(\text{academic gown}) = 0.532$	$p'(\{\text{frog}\}) = 0.102$	$p'(\text{camellia}) = 0.001$	$p'(\text{academic gown}) = 0.006$

	$p'(\omega_0^1) = 0.001$	$p'(\omega_0^2) = 0.004$	$p'(\omega_0^3) = 0.993$	$p'(\omega_0^0) = 0.001$
 bull frog	$p'(\text{bull frog}) = 0.382$	$p'(\text{frog}) = 0.972$	$p'(\text{buttercup}) = 0.001$	$p'(\text{bull frog}) = 0.388$
	$p'(\text{tailed frog}) = 0.602$	$p'(\text{cat}) = 0.010$	$p'(\text{camellia}) = 0$	$p'(\text{tailed frog}) = 0.611$

	$p'(\omega_0^1) = 0$	$p'(\omega_0^2) = 0$	$p'(\omega_0^3) = 0.999$	$p'(\omega_0^0) = 0$
 Yorkshire terrier	$p'(\text{Yorkshire terrier}) = 0.413$	$p'(\text{dog}) = 0.732$	$p'(\text{buttercup}) = 0$	$p'(\text{Yorkshire terrier}) = 0.476$
	$p'(\text{chihuahua}) = 0.452$	$p'(\text{cat}) = 0.158$	$p'(\text{camellia}) = 0.001$	$p'(\text{chihuahua}) = 0.521$

	$p'(\omega_0^1) = 0.001$	$p'(\omega_0^2) = 0.011$	$p'(\omega_0^3) = 0.983$	$p'(\omega_0^0) = 0.001$

the image belongs to class “frog”, which is useless. We can find a similar phenomenon in the final example. This explains why only the error rates of partial classes on the Tiny ImageNet dataset decrease, such as the “albatross” and “Egyptian cat” classes shown in Table 5.5. For the images misclassified by the Flower-102 mass function, the other two individual classifiers do not provide any useful information to correct the decisions. Consequently, the classification performance on the Flower-102 database is not improved. In summary, these observations show that the proposed approach makes it possible to combine CNN classifiers trained from heterogeneous databases to obtain a more general classifier able to recognize classes from any of the databases, without degrading the performance of the individual classifiers, and sometimes even yielding better results for some classes.

Comparing the test error rates of the MFE classifiers with and without the end-to-end learning procedure as shown in Table 5.2, we can see that the fine-tuning strategy further slightly boosts the overall performance, as well as the performance on the Tiny ImageNet and CIFAR-10 datasets. Thus, the fine-tuning procedure decreases the classification error rate of the proposed architecture, and can be seen as a way to improve the performance of CNN classifiers. This is because the end-to-end learning procedure adapts the individual classifiers to the new classification

Table 5.5: Error rates (in percent) of some classes on the Tiny ImageNet dataset before and after fusion. We boldface the classes with decreased error rates.

	Classifier	albatross	Egyptian cat	CD player	rocking chair	lemon
Before fusion	E-FitNit-4	42	36	28	4	12
	P-FitNit-4	40	38	28	6	12
After fusion	E2E MF-FitNit-4	26	16	27	4	12
	E2E PMF-FitNit-4	34	26	28	6	12
	E2E BF-FitNit-4	40	34	29	4	12

problem. More specifically, before fusion, the CNN classifiers are pre-trained for the classification tasks with the frames of discernment before refinement. The proposed end-to-end learning procedure fine-tunes the parameters in the CNN and DS layers to make them more suitable to the classification task in the refined frame.

Finally, Table 5.2 sheds some light on the relative performance of different classifier fusion strategies. The PMF fusion strategy also improves the performance of the probabilistic CNNs trained on each of the three databases, but it is not as good as the proposed method. In contrast, the BF strategy degrades the performance of the individual classifiers, which shows that the method is not effective when the numbers of classes in the different frames are very unbalanced. The relatively high error rates obtained of the two feature fusion strategies (E2E EFC and E2E PFC) show that the simple feature-concatenation methods is less effective than the other ones. All in all, the proposed evidential fusion strategy outperforms the other tested methods on the datasets considered in this experiment.

5.3.2 Image-classification experiment #2

Experiment setting. We used three datasets in this experiment: CIFAR-10 [63], Caltech-UCSD Birds-200-2011 (CUB) [141], and Oxford-IIIT Pet [98]. The CIFAR-10 dataset has been pre-split into 50k training and 10k testing images. For the CUB (11,788 images) and Oxford-IIIT Pet (7,349 images) datasets, we divided each into training and testing sets with a ratio of about 1:1. The training and testing sets keep a ratio of about 1:1 in each class. In the fine-tuning procedure, the frames of the three datasets are refined into a common one, as shown in Table 5.6. Figure 5.3 presents the semantic relationship of the classes in the three datasets. After merging the three sets, there are 59669 training samples and 19,468 testing samples for, respectively, fine-tuning and performance evaluation. We designed three MFE-CNN classifiers whose CNN architectures have been introduced in Section 2.1.4. The optimized numbers of prototypes in the DS layers are presented in Table 5.1b.

Results and discussion. Table 5.7 presents the experiment results. The overall error rates decrease after fusion, as well as the ones on the CIFAR-10 dataset, benefiting from the useful information given by the classifiers trained with the CUB

Table 5.6: Lists of classes in the CIFAR-10, CUB, Oxford-IIIT Pet datasets. The notations ω_0^2 and ω_0^3 stand for the “anything else” class added to the frames of the CUB and Oxford datasets.

Frame	Class
CIFAR-10	airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.
CUB	cardinal, house wren, . . . , (200 species of birds), ω_0^2 .
Oxford-IIIT pet	bengal, boxer, . . . , (37 species of cats and dogs), ω_0^3 .
Common frame	airplane, automobile, deer, frog, horse, ship, truck, cardinal, house wren, . . . , (200 species of birds), bengal, boxer, . . . , (37 species of cats and dogs).

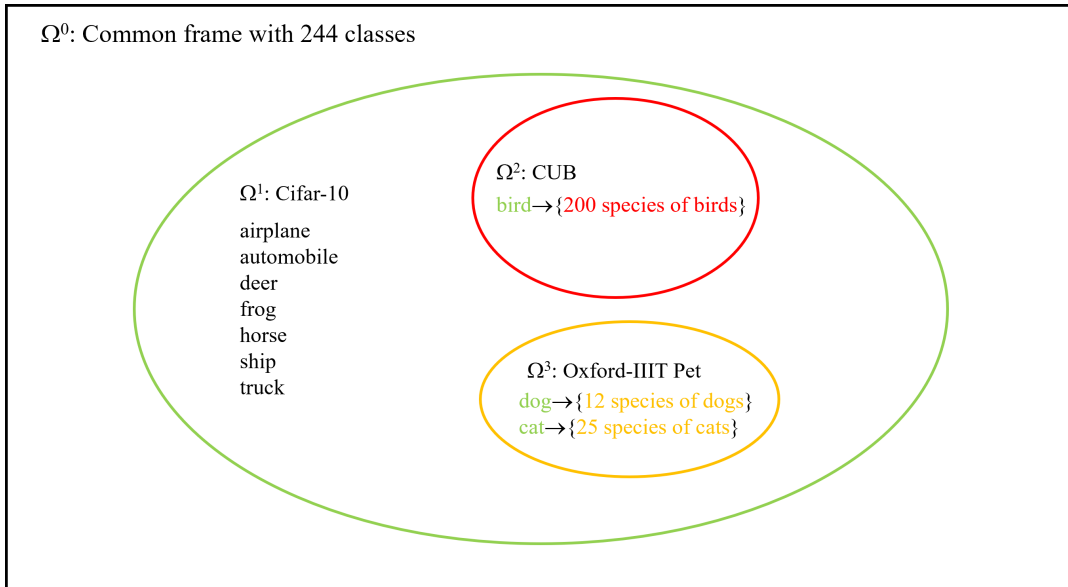


Figure 5.3: Semantic relationship of the classes on the CIFAR-10 (green), CUB (red) and Oxford-IIIT Pet (yellow) datasets using the representation way introduced in Figure 5.2.

and Oxford-IIIT Pet datasets. However, for images misclassified by the CUB probability mass function, the other two individual classifiers do not provide any useful information to correct the decisions. Consequently, the classification performance on the CUB dataset is not improved. Besides, the probability mass function from the CIFAR-10 classifier sometimes includes useful information for classifying samples into one species of cat or dog, even though the number of such examples is small, such as the last example in Table 5.8. This phenomenon is responsible for the small change in the performance on the Oxford-IIIT Pet dataset. Besides, the end-to-end can slightly improve the overall performance by making the learned parameters in the CNN backbones and DS layers more suitable for the refined task. In addition, the proposed approach outperforms the other four strategies on the multi-model fusion.

5.3.3 Semantic-segmentation experiment #1

Experiment setting. In the semantic-segmentation experiment #1, we used three datasets: Pascal VOC 2012 [30], Cityscapes [13] and Stanford background [38]. Table

Table 5.7: Average test error rates (in percent) of different classifiers on the classification experiment #2: NIN architecture (a), FitNet-4 (b), and ViT-L/16 (c). “E2E” stands for fine-tuned classifiers. The lowest error rates are in bold and second low are underlined. The initialized weights in ResNet-101 backbone is from a model pre-trained by the ImageNet-21k dataset [105], and we then fine-tune the backbone on the union of the three datasets.

(a)					
	Classifier	CIFAR-10	CUB	Oxford-IIIT pet	Overall
Before fusion	E-FitNit-4	6.50	25.07	10.17	-
	P-FitNit-4 [106]	6.58	25.18	10.56	-
After fusion	MFE-FitNit-4	<u>5.07</u>	<u>25.07</u>	<u>9.82</u>	<u>12.65</u>
	PMF-FitNit-4	5.86	25.16	10.13	13.12
	BF-FitNit-4	6.10	27.84	11.08	14.31
	E2E MFE-FitNit-4	4.49	<u>25.07</u>	9.81	12.37
	E2E PMF-FitNit-4	5.47	25.14	10.11	12.92
	E2E BF-FitNit-4	6.26	27.76	10.87	14.32
	E2E PFC-FitNit-4	6.20	25.11	9.78	13.21
	E2E EFC-FitNit-4	6.86	25.10	11.30	13.80
(b)					
	Classifier	CIFAR-10	CUB	Oxford-IIIT pet	Overall
Before fusion	E-ResNet-101	1.66	12.99	6.27	-
	P-ResNet-101 [81]	1.71	13.08	6.38	-
After fusion	MFE-ResNet-101	<u>1.39</u>	<u>13.00</u>	<u>6.11</u>	<u>6.14</u>
	PMF-ResNet-101	1.56	13.08	6.23	6.28
	BF-ResNet-101	3.75	13.68	7.86	7.83
	E2E MFE-ResNet-101	1.37	12.99	6.06	6.12
	E2E PMF-ResNet-101	1.50	13.08	6.18	6.24
	E2E BF-ResNet-101	3.58	13.47	7.32	7.58
	E2E PFC-ResNet-101	1.61	13.76	6.32	6.54
	E2E EFC-ResNet-101	2.78	14.59	8.61	7.80
(c)					
	Classifier	CIFAR-10	CUB	Oxford-IIIT pet	Overall
Before fusion	E-ViT-L/16	0.94	10.77	3.77	-
	P-ViT-L/16 [27]	0.78	10.77	4.06	-
After fusion	MF-ViT-L/16	1.16	10.77	3.67	4.85
	PMF-ViT-L/16	0.83	10.75	3.95	4.73
	BF-ViT-L/16	0.96	13.87	4.81	6.00
	E2E MF-ViT-L/16	<u>0.82</u>	10.74	<u>3.70</u>	4.68
	E2E PMF-ViT-L/16	<u>0.83</u>	<u>10.76</u>	<u>3.83</u>	<u>4.71</u>
	E2E BF-ViT-L/16	1.10	13.77	4.76	6.02
	E2E PFC-ViT-L/16	0.90	11.36	3.86	4.95
	E2E EFC-ViT-L/16	3.79	11.14	6.73	6.79

Table 5.8: Examples of mass functions on the CIFAR-10 and Oxford-IIT pet datasets before and after fusion by the MFE strategy. Only some masses before and after fusion are shown for lack of space.









Instance/label	Before fusion			MF on Ω after fusion
	MF from CIFAR-10	MF from CUB	MF from Oxford	
 bird	$m(\{\text{airplane}\}) = 0.506$ $m(\{\text{bird}\}) = 0.382$...	$m(\{\text{caspinan}\}) = 0.698$ $m(\{\text{horned grebe}\}) = 0.109$...	$m(\{\text{samyod}\}) = 0$ $m(\{\text{pyrenees}\}) = 0.001$...	$m(\{\text{airplane}\}) = 0.101$ $m(\{\text{caspinan}\}) = 0.672$...
 caspinan	$m(\Theta^1) = 0.065$ $m(\theta_0^2) = 0.098$ $m(\theta_0^3) = 0.905$	$m(\theta_0^2) = 0.098$ $m(\theta_0^3) = 0.905$	$m(\theta_0^3) = 0.905$	$m(\Omega) = 0.007$
 caspinan	$m(\{\text{airplane}\}) = 0.009$ $m(\{\text{bird}\}) = 0.823$...	$m(\{\text{caspinan}\}) = 0.423$ $m(\{\text{horned grebe}\}) = 0.452$...	$m(\{\text{samyod}\}) = 0$ $m(\{\text{pyrenees}\}) = 0.001$...	$m(\{\text{caspinan}\}) = 0.415$ $m(\{\text{horned grebe}\}) = 0.450$...
 caspinan	$m(\Theta^1) = 0.092$ $m(\theta_0^2) = 0.084$ $m(\theta_0^3) = 0.951$	$m(\theta_0^2) = 0.084$ $m(\theta_0^3) = 0.951$	$m(\theta_0^3) = 0.951$	$m(\Omega) = 0.009$
 byssinian	$m(\{\text{cat}\}) = 0.742$ $m(\{\text{dog}\}) = 0.131$...	$m(\{\text{caspinan}\}) = 0.002$ $m(\{\text{horned grebe}\}) = 0$...	$m(\{\text{byssinian}\}) = 0.412$ $m(\{\text{bengal}\}) = 0.503$...	$m(\{\text{byssinian}\}) = 0.414$ $m(\{\text{bengal}\}) = 0.505$...
 byssinian	$m(\Theta^1) = 0.032$ $m(\theta_0^2) = 0.931$ $m(\theta_0^3) = 0.038$	$m(\theta_0^2) = 0.931$ $m(\theta_0^3) = 0.038$	$m(\theta_0^3) = 0.038$	$m(\Omega) = 0.005$
 keeshond	$m(\{\text{cat}\}) = 0.158$ $m(\{\text{dog}\}) = 0.705$...	$m(\{\text{albatross}\}) = 0.001$ $m(\{\text{horned grebe}\}) = 0$...	$m(\{\text{rogdoll}\}) = 0.682$ $m(\{\text{keeshond}\}) = 0.254$...	$m(\{\text{rogdoll}\}) = 0.369$ $m(\{\text{keeshond}\}) = 0.485$...
 keeshond	$m(\Theta^1) = 0.058$ $m(\theta_0^2) = 0.975$ $m(\theta_0^3) = 0.001$	$m(\theta_0^2) = 0.975$ $m(\theta_0^3) = 0.001$	$m(\theta_0^3) = 0.001$	$m(\{\text{cat}\}) = 0.021$

Table 5.9: Lists of classes for the Pascal VOC, Cityscapes, and Stanford background datasets in the semantic segmentation experiments #1. Classes in bold characters have the same semantics as the “anything else” class.

Frame	Class
Pascal VOC Ω^1	person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv, background .
Cityscapes Ω^2	person, rider, car, truck, bus, rails, motorcycle, bicycle, caravan, trailer, road, sidewalk, parking, rail track, building, wall, fence, guard rail, bridge, tunnel, pole, pole group, traffic sign, traffic light, vegetation, terrain, sky, ground, void .
Stanford background Ω^3	sky, tree, road, grass, water, building, mountain, foreground .
Common frame Ω^0	person, rider, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, caravan, trailer, road, sidewalk, parking, rail track, building, wall, fence, guard rail, bridge, tunnel, pole, pole group, traffic sign, traffic light, terrain, sky, tree, grass, mountain, bottle, chair, dining table, potted plant, sofa, tv, ω_0^0 .

5.9 summarizes the classes on the three datasets. The information of the Pascal VOC 2012 has been introduced in Section 4.2.1. The Cityscapes dataset focuses on semantic understanding of urban street scenes, which consists of 5k annotated images labeled in 30 classes. Data was captured in 50 cities during several months, daytimes, and good weather conditions. The Stanford Background dataset with 715 labeled images is mainly used for geometric and semantic scene understanding. Its selection criteria are for the images to be of outdoor scenes, have approximately 320-by-240 pixels, contain at least one foreground object, and have the horizon position within the image (it need not be visible). The classes “background” (Pascal VOC), “void” (Cityscapes), and “foreground” (Stanford background) have the semantics of “anything else” that makes the three frames compatible.

Table 5.10: Numbers of output units in the encoder-decoder architectures: semantic-segmentation experiment #1 (a), and semantic-segmentation experiment #2 (b).

(a)				(b)			
	Pascal VOC	Cityscapes	Stanford background		Pascal VOC	MIT-scene	SIFT Flow
FCN-8s	64	96	28	FCN-8s	64	128	64
FCN-SegNet	64	96	28	FCN-SegNet	64	128	64
FCN-CRF	64	96	28	FCN-CRF	64	128	64

For a testing set T with pixel-wise soft labels, *intersection over union* (IoU) is defined as

$$IoU = \frac{1}{2^{|\Omega|} - 1} \sum_{B \subseteq \Omega} \frac{|\mathbf{G}^B \cap \mathbf{P}^B|}{|\mathbf{G}^B \cup \mathbf{P}^B|}, \quad (5.4)$$

where $\mathbf{P}^B = \{i : \hat{\omega}(i) \in B\}$ is the predicted area containing pixels classified to one of the classes in B ; and $\mathbf{G}^B = \{i : A_*(i) = B\}$ is the ground truth area composed of pixels with label B . In the special case without soft labels, IoU in this chapter boils down to the original definition of IoU [62, 77, 95].

We designed three different MFE-FCN models in the experiment. Each model consists of three pre-trained evidential FCN, referring to one of the encoder-decoder architectures introduced in Section 2.2.3. Tables 5.1c and 5.10a present, respectively, the optimized numbers of prototypes in the DS layers and the output units in the encoder-decoder architectures for the three datasets. We also compared the proposed approach with the four fusion systems, as introduced in Section 5.3.1.

Results and discussion. Table 5.11 presents the experiment results. The overall IoU rates increase after fusion, as well as the ones on the three datasets. This indicates that each model can provide useful information for the other two. Thus, the proposed approach has the potential to combine different FCN-based models trained from heterogeneous databases to obtain a more general one that is able to segment images from any of the databases. The multi-FCN fusion does not degrade the performance of the individual models, and sometimes even yields better results for some classes, as shown in Table 5.12. Figure 5.4 provides some segmentation results before and after fusion from the Cityscapes dataset.

Similar to the two image-classification experiments, the proposed information-fusion strategy outperforms the other four on semantic segmentation. In contrast to the image-classification experiments, the BF and EFC strategies can also improve the segmentation performance. For the BF strategy, this is because the numbers of classes in the three frames are not very unbalanced. In the EFC strategy, the small numbers of output units from the encoder-decoder architectures make the distance measurement using (1.29) in the DS layers still valid after concatenating pixel-wise feature vectors, which guarantees the segmentation performance in the EFC-FCN models. This phenomenon also indicates the limitations of the BF and EFC strategies. All in all, the proposed evidential fusion strategy outperforms the other tested

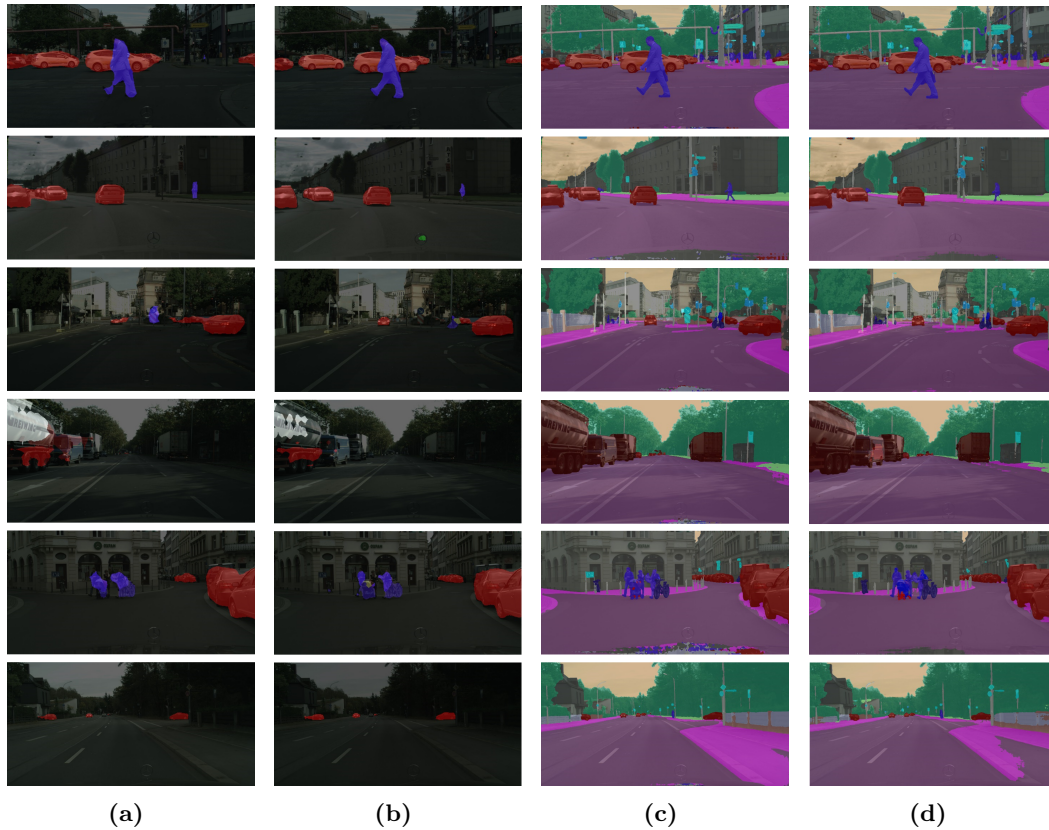


Figure 5.4: Segmentation examples from the Cityscapes dataset before and after fusion: Segmentation results from (a) E-FCN-CRF before fusion, (b) P-FCN-CRF before fusion, (c) MFE strategy after fusion, (d) PMF strategy after fusion. Different colors stands for different classes.

methods on the datasets considered in this experiment.

5.3.4 Semantic-segmentation experiment #2

Three benchmark datasets were used in this experiment: Pascal VOC 2012 [30], MIT-scene Parsing [156], and SIFT Flow [128], having been introduced in Section 4.2.1. The classes “background” (Pascal VOC and MIT-scene Parsing) and “foreground” (SIFT Flow) have the semantics of “anything else” that makes the three frames compatible. The encoder-decoder architectures and metrics for performance evaluation in this experiment are the same as the ones in experiment #1. Tables 5.1d and 5.10b present, respectively, the optimized numbers of prototypes in the DS layers and the output units in the encoder-decoder architectures. In this experiment, we also compared the MFE-FCN models to the four fusion systems.

The findings in the experiment are similar to the previous three, as shown in Table 5.13. The results demonstrate the advantage of the proposed approach on aggregating the information from different DNNs to make more general decisions without the negative effects on the performance of the individual FCNs. Besides, compared to the other widely-used methods, the proposed one is more flexible to introduce new

Table 5.11: Mean intersection over union of different FCN models on the segmentation experiment #1: FCN-8s (a), FCN-SegNet (b), and FCN-CRF (c). “E2E” stands for fine-tuned classifiers. The lowest error rates are in bold and second low are underlined.

(a)					
	Classifier	Pascal VOC	Cityscapes	Stanford background	Overall
Before fusion	E-FCN-8s	0.634	0.649	0.756	-
	P-FCN-8s [77]	0.627	0.648	0.748	-
After fusion	MFE-FCN-8s	<u>0.653</u>	<u>0.663</u>	<u>0.780</u>	<u>0.669</u>
	PMF-FCN-8s	0.638	0.658	0.769	0.661
	BF-FCN-8s	0.604	0.633	0.754	0.635
	E2E MFE-FCN-8s	0.656	0.665	0.782	0.671
	E2E PMF-FCN-8s	0.643	0.662	0.770	0.665
	E2E BF-FCN-8s	0.613	0.639	0.758	0.642
	E2E PFC-FCN-8s	0.651	0.653	0.769	0.660
	E2E EFC-FCN-8s	0.650	0.658	0.773	0.664
(b)					
	Classifier	Pascal VOC	Cityscapes	Stanford background	Overall
Before fusion	E-FCN-SegNet	0.652	0.565	0.778	-
	P-FCN-SegNet [1]	0.645	0.558	0.773	-
After fusion	MFE-FCN-SegNet	<u>0.662</u>	<u>0.578</u>	<u>0.785</u>	<u>0.609</u>
	PMF-FCN-SegNet	0.653	0.566	0.775	0.598
	BF-FCN-SegNet	0.650	0.538	0.761	0.576
	E2E MFE-FCN-SegNet	0.664	0.583	0.787	0.613
	E2E PMF-FCN-SegNet	0.655	0.570	0.777	0.601
	E2E BF-FCN-SegNet	0.652	0.549	0.765	0.585
	E2E PFC-FCN-SegNet	0.659	0.569	0.775	0.601
	E2E EFC-FCN-SegNet	0.660	0.572	0.779	0.604
(c)					
	Classifier	Pascal VOC	Cityscapes	Stanford background	Overall
Before fusion	E-FCN-CRF	0.789	0.694	0.843	-
	P-FCN-CRF [6]	0.784	0.691	0.842	-
After fusion	MFE-FCN-CRF	<u>0.803</u>	<u>0.710</u>	<u>0.861</u>	<u>0.739</u>
	PMF-FCN-CRF	0.801	0.697	0.861	0.729
	BF-FCN-CRF	0.774	0.680	0.848	0.711
	E2E MFE-FCN-CRF	0.804	0.712	0.861	0.741
	E2E PMF-FCN-CRF	0.802	0.701	0.863	0.733
	E2E BF-FCN-CRF	0.778	0.684	0.851	0.715
	E2E PFC-FCN-CRF	0.792	0.703	0.855	0.732
	E2E EFC-FCN-CRF	0.799	0.699	0.859	0.730

Table 5.12: Test IoU before and after information fusion on the Cityscapes dataset using the FCN-CRF architecture.

	Classifier	sky	building	person	rider	car	bus	bicycle
Before fusion	E-FCN-CRF	0.946	0.913	0.784	0.598	0.936	0.642	0.693
	P-FCN-CRF	0.940	0.903	0.792	0.585	0.931	0.668	0.684
After fusion	E2E MFE-FCN-CRF	0.951	0.919	0.801	0.594	0.950	0.661	0.709
	E2E PMF-FCN-CRF	0.944	0.914	0.799	0.586	0.936	0.675	0.687
	E2E BF-FCN-CRF	0.924	0.884	0.794	0.581	0.926	0.671	0.684

Table 5.13: Mean intersection over union of different FCN models on the segmentation experiment #2: FCN-8s (a), FCN-SegNet (b), and FCN-CRF (c). “E2E” stands for fine-tuned classifiers. The lowest error rates are in bold and second low are underlined.

(a)					
	Classifier	Pascal VOC	SIFT flow	MIT-scening	Overall
Before fusion	E-FCN-8s	0.634	0.396	0.296	-
	P-FCN-8s [77]	0.627	0.394	0.294	-
After fusion	MFE-FCN-8s	<u>0.649</u>	<u>0.420</u>	<u>0.311</u>	<u>0.449</u>
	PMF-FCN-8s	0.641	0.412	0.306	0.441
	BF-FCN-8s	0.638	0.412	0.299	0.440
	E2E MFE-FCN-8s	0.652	0.421	0.313	0.450
	E2E PMF-FCN-8s	0.643	0.414	0.308	0.443
	E2E BF-FCN-8s	0.640	0.415	0.304	0.443
	E2E PFC-FCN-8s	0.642	0.415	0.299	0.443
	E2E EFC-FCN-8s	0.644	0.413	0.291	0.441
(b)					
	Classifier	Pascal VOC	SIFT flow	MIT-scening	Overall
Before fusion	E-FCN-SegNet	0.652	0.400	0.310	-
	P-FCN-SegNet [1]	0.645	0.399	0.305	-
After fusion	MFE-FCN-SegNet	<u>0.671</u>	<u>0.422</u>	<u>0.315</u>	<u>0.455</u>
	PMF-FCN-SegNet	0.664	0.415	0.310	0.447
	BF-FCN-SegNet	0.661	0.414	0.304	0.445
	E2E MFE-FCN-SegNet	0.674	0.423	0.317	0.456
	E2E PMF-FCN-SegNet	0.666	0.417	0.312	0.449
	E2E BF-FCN-SegNet	0.663	0.417	0.308	0.448
	E2E PFC-FCN-SegNet	0.664	0.416	0.304	0.448
	E2E EFC-FCN-SegNet	0.666	0.415	0.297	0.447
(c)					
	Classifier	Pascal VOC	SIFT flow	MIT-scening	Overall
Before fusion	E-FCN-CRF	0.789	0.418	0.354	-
	P-FCN-CRF [6]	0.784	0.417	0.350	-
After fusion	MFE-FCN-CRF	<u>0.801</u>	<u>0.431</u>	<u>0.360</u>	<u>0.487</u>
	PMF-FCN-CRF	0.796	0.426	0.356	0.482
	BF-FCN-CRF	0.791	0.420	0.352	0.477
	E2E MFE-FCN-CRF	0.802	0.433	0.361	0.489
	E2E PMF-FCN-CRF	0.799	0.427	0.359	0.484
	E2E BF-FCN-CRF	0.794	0.423	0.355	0.480
	E2E PFC-FCN-CRF	0.794	0.419	0.352	0.476
	E2E EFC-FCN-CRF	0.791	0.423	0.357	0.479

classes at any stage. The end-to-end learning can slightly improve by making the learned parameters in an MFE-FCN model more suitable for the new task.

5.4 Experiments on training shallow networks for complex tasks

In the last decade, neural networks have become deeper and deeper from the original AlexNet [64] to the 19-layer VGG [116], even the ResNet with hundred layers [48], because depth tends to improve network performances in practice. One of the problems from the “deeper and deeper” practice is that gradient-based training becomes more

difficult since deeper networks are more non-linear. Besides, training procedures require more and more powerful graphics processor units (GPUs) and memories, which limits the possibility for some small research groups to achieve state-of-the-art results using the networks proposed by some big laboratories.

The main superiority of the proposed information-fusion approach is to combine DNNs trained from heterogeneous databases with different sets of classes while having at least as good performance as those of the individual networks on their respective datasets. In this section, we consider combining simple and shallow networks for a complicated problem using the information-fusion approach. More precisely, a problem with a large number of classes is first decomposed into some simple sub-problems, such as binary classifications. We then train shallow neural networks for each sub-problem and combine them using the proposed approach to conduct the original complicated task. Compared with a single DNN for the full problem, these shallow networks are easily trained and their combination has the potential to achieve a similar performance as that of the single DNN. Thus, in the experiments, we do not aim at introducing new algorithms to outperform the state-of-the-art ones in terms of average error rate (5.3) or intersection over union (5.4), but we investigate the combination of existing shallow neural networks for reducing training difficulty while having similar performances as the state-of-the-art models.

5.4.1 Experiment on the Tiny ImageNet dataset

Experiment setting. We tried to solve the classification problem on the Tiny ImageNet dataset by combining shallow evidential CNNs, whose CNN backbone is shown in Figure 5.5. We decomposed the classification problem with 200 classes into 200 binary classification problems, in which each CNN was used to distinguish one class from others, named the *200 mass-fusion evidential CNN (200-MFE-CNN) classifier*. The output of each evidential CNN is the mass on the frame of a binary-classification problem, such that

$$\mathbf{m}^i = \{m^i(\{\omega_i\}), m^i(\overline{\{\omega_i\}}), m^i(\Omega)\},$$

where $\overline{\{\omega_i\}}$ is the set of anything else except class ω_i and Ω is the frame of discernment on the Tiny ImageNet dataset. We then aggregate the 200 CNNs using Dempster's rule (1.11). We also decomposed the original classification problem into 25 multi-class classification problems. Precisely, the 200 classes in the dataset were divided into 25 groups and each group has 8 classes. Every two groups are disjoint. For each group A , we defined a frame of discernment Ω_A with all classes in the group and designed an evidential CNN classifier. The outputs of the classifier were the mass to each singleton class in A and frame of discernment Ω . Finally, we combined the 25 classifiers using the proposed method and the combination is referred to as the *25-MFE-CNN classifier*. Similarly, we designed the *50- and 100-MFE-CNN classifiers*.

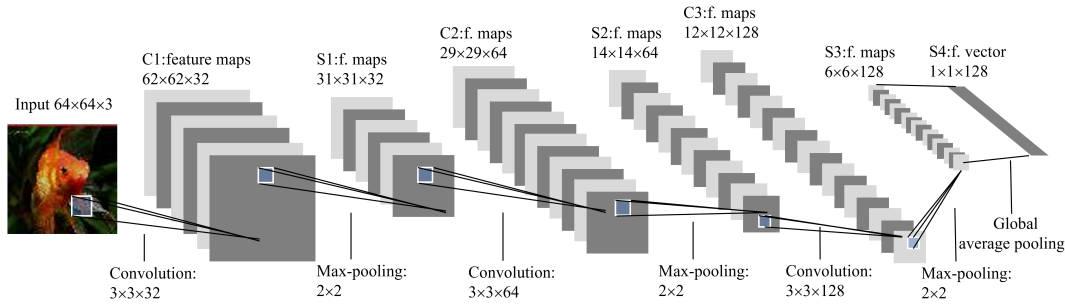


Figure 5.5: Architecture of a shallow CNN with 91k parameters.

Results and discussion. Figure 5.6a presents the test error rates of MFE-CNN classifiers with different numbers of CNNs, as well as the performances of the different fusion strategies with fine-tuning. The 200-MFE-CNN classifier achieves an error rate of 32.67%, which is lower than those of the evidential FitNet-4 and UPANets classifiers. This demonstrates that those of the proposed approach makes it possible to get a high accuracy using simple and shallow CNNs. Besides, the performances of 100-MFE-CNN and 50-MFE-CNN classifiers also exceed that of the FitNet-4 classifier and is similar to that of the UPANets, indicating that each shallow CNN can handle a classification problem with a small number of classes (e.g., two or four classes), and we should select a reasonable number of CNNs for the problem on the Tiny ImageNet dataset. In addition, we do not compare the proposed classifier with the ResNet-101 ones, as shown in Table 5.2c, because its pre-trained weights are obtained using an extra learning set, ImageNet-21k [105].

The PMF and PFC classifiers also have lower error rates than those of FitNet-4 and UPANets but are outperformed by the MFE-CNN classifiers with the same numbers of CNNs. This indicates the proposed approach method is better than the PMF and PFC approaches for training simple networks for a complex task. Figure 5.6a does not report the error rates of the BFC approach since they are too large. The BFC approach does not work because the distance measurement in a DS layer (1.29) becomes invalid when the dimension of the concentrated feature vectors is too large.

Figure 5.6b displays the training times of different information-fusion strategies to achieve the error rates reported in Figure 5.6a, as well as the training times of the evidential FitNet and UPANets classifiers. The 50-MFE-CNN classifier costs less time than the FitNet and UPANets classifiers, requiring no extra training technologies, such as knowledge distillation in FitNet-4 or skip connections in ResNet-101. This demonstrates that the training processes of the MFE-CNN classifiers with 50 or less CNN are easier than the ones of FitNet and UPANets classifiers. Therefore, the proposed approach has the potential to reduce the training difficulty, while having a similar performance as the state-of-the-art models. Compared to the PMF and PFC approaches, the proposed approach requires a slightly large training time owing to the

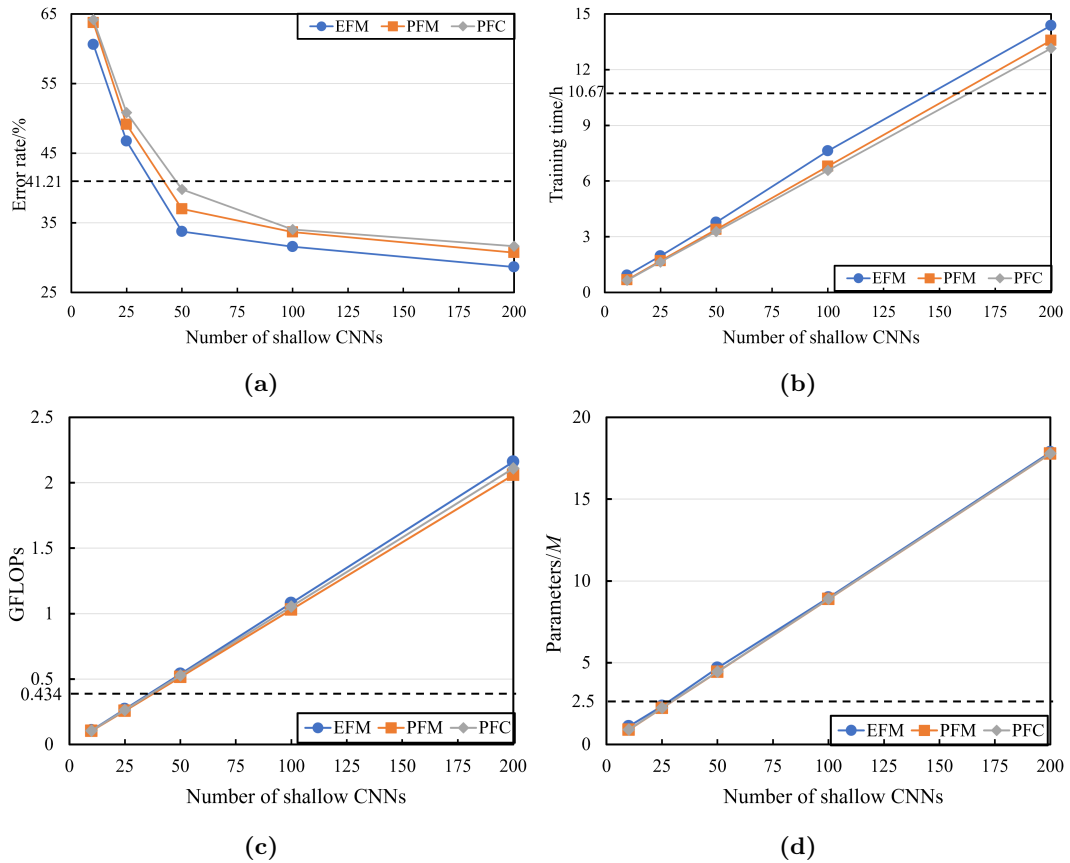


Figure 5.6: Results of CNN combination experiment on the Tiny ImageNet dataset: error rate (a), training time (b), GFLOPs (c), and the number of parameters (d) vs. the number of shallow CNNs. The black line indicates the results of the evidential FitNet-4 classifier. Error rate, training time, GFLOPs and parameters of the evidential UPANets classifier are 34.61%, $> 24 h$, 2.77 and 24.4 M , respectively. GFLOPs stands for 10^9 (giga) floating point operations and M means million.

DS layers, but the increase in computation cost is still acceptable, partially thanks to the simplified contour function-based combination rule. In addition, the training time almost linearly increases with the number of evidential CNNs. Thus, we should carefully select a reasonable number of evidential CNNs for the problem.

The number of network parameters and floating point operations (FLOPs) can be considered as the metrics to determine the number of evidential CNNs. FLOPs are widely used to describe how many operations are required to run a single instance in a deep neural network [27, 42, 143, 9]; calculation processes can be found in [54]. A lower value of FLOPs always means that an algorithm processes a new instance with less computation costs. Figures 5.6c and 5.6d provide, respectively, the parameters and FLOPs of the classifiers using different information-fusion strategies. The parameters and FLOPs increase with the increase in the number of shallow CNNs. The FLOPs of the 50-MFE-CNN classifier is still close to that of the FitNet-4 classifier, though its parameters exceed the FitNet-4 classifier. Thus, when using 50

shallow CNNs, the proposed approach achieves better performance than the FitNet-4 model but it does not introduce significant computation costs or training difficulty. We can also use 100 shallow CNNs to get a similar performance as that of the UPANets classifier but still have a reasonable value of FLOPs. The proposed classifier uses a little more FLOPs than the PMF and PFC classifiers owing to the use of DS layers, even though the gaps are small. All in all, the proposed evidential fusion strategy has the potential of combining shallow networks for the complicated task on the Tiny ImageNet dataset.

5.4.2 Experiment on the Cityscapes dataset

Experiment setting. The Cityscapes dataset was considered in this experiment. U-net models, as shown in Figure 2.12, were combined by the proposed approach to solve the problem of semantic segmentation. Similar to the classification experiment, we divided the segmentation task into 30 problems of binary segmentation or some easy problems of multi-class segmentation. We also compared the proposed approach with four other methods described in Section 5.3.1.

Results and discussion. Figure 5.7 shows the results of IoU from the MFE-FCN models with different numbers of U-nets. The 30-MFE-FCN model has the maximum value of IoU, and the 15-, 20-, 25-MFE-FCN ones also achieve the values of IoU higher than the FCN-CRF model. Considering training times and FLOPs, we can conclude the 15-MFE-FCN model is suitable for the segmentation task on the Cityscapes dataset. The proposed approach allows us to achieve a good performance of semantic segmentation even using some shallow and simple FCN-based models while reducing the training difficulty and introducing small extra costs on processing a new sample.

5.5 Conclusion

In this chapter, we have extended the proposed combined framework of DST and DNNs into the evidential fusion of heterogeneous DNNs. In this extension, pre-trained DST-based evidential DNNs extract features from input data and convert them into mass functions on different frames of discernment. A fusion module then aggregates these mass functions using Dempster’s rule. An end-to-end learning procedure allows us to fine-tune the overall architecture using a learning set with soft labels.

The proposed approach makes it possible to combine DNNs trained from heterogeneous databases with different sets of classes, which provides a way to fuse the partial and imperfect outputs of DNNs. In the experiments of classification and segmentation, the combined network can classify or segment images from any of these datasets while having at least as good performance as those of the individual networks on their respective datasets. The proposed approach has the capacity of introducing

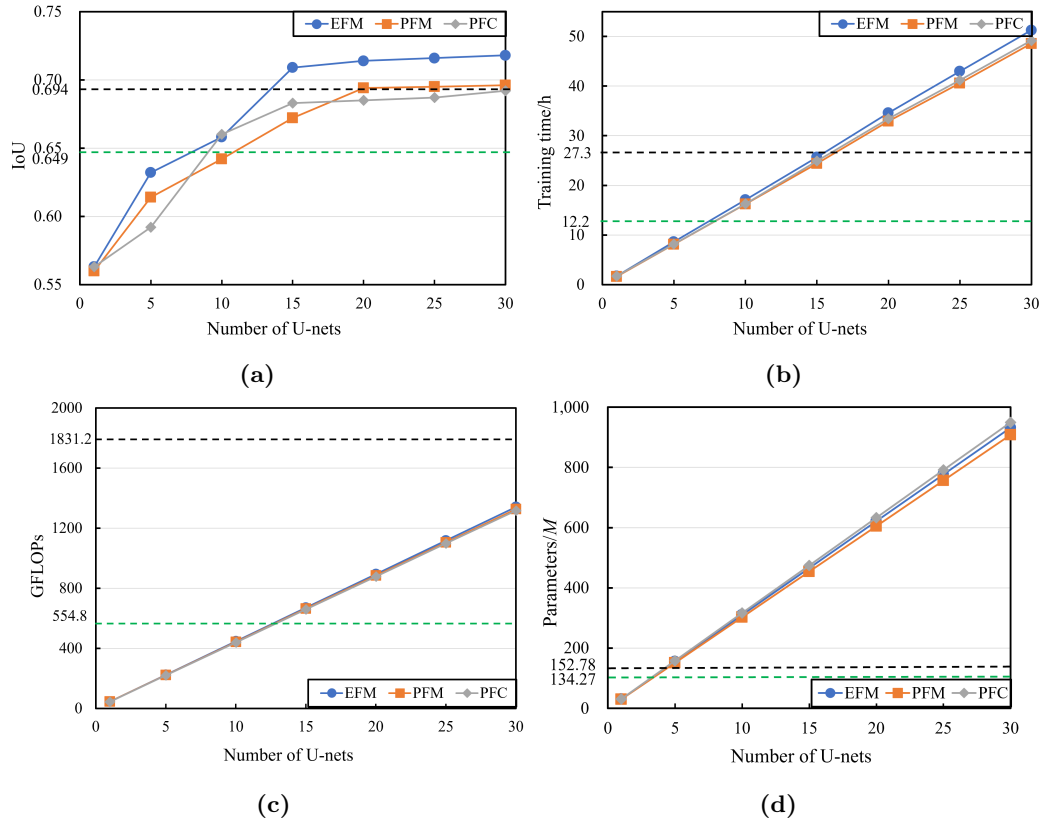


Figure 5.7: Results of FCN combination experiment on the Cityscapes dataset: mean IoU (a), training time (b), FLOPs (c), and parameters (d) vs. the number of U-nets. The black and green dotted lines indicate the results of the evidential FCN-CRF and FCN-8s models, respectively. GFLOPs stands for 10^9 (giga) floating point operations and M means million.

additional networks trained from new datasets with different sets of classes at any stage. An end-to-end learning procedure further slightly improves the performance of the proposed architecture. This approach is shown to outperform other decision-level or feature-level fusion strategies for combining DNNs. In addition, the proposed approach allows us to use some simple and shallow neural networks to achieve a similar performance as a state-of-the-art algorithm for a complex task, while reducing the training difficulty and introducing small extra FLOPs. These results indicate a potential direction to simplify the training procedure for some very difficult problems on object classification and semantic segmentation.

Conclusions and perspectives

In this thesis, we have proposed a new framework based on deep neural networks and DST to tackle the problem of data uncertainty in deep learning, mainly including ambiguous, unreliable, imprecise, and incomplete data. The advantages of this framework for decision-making with uncertainty have been demonstrated in pattern classification and semantic segmentation tasks by combining DST-based neural-network layers with CNNs and FCNs, respectively. In addition, the proposed framework using an evidential fusion strategy makes it possible to combine heterogeneous DNNs. The following three main contributions have been described in this thesis.

First, an evidential CNN classifier has been designed by plugging a DST-based neural-network layer followed by a utility layer at the backbone output of a convolutional neural network (CNN) to perform set-valued classification and outlier detection. A major finding is that this hybridization makes it possible to improve the performance of CNN models by assigning ambiguous patterns to multi-class sets. The proposed classifier is able to select a set of classes when the feature representation does not allow us to select a single class unambiguously, which easily leads to incorrect classification in probabilistic CNN classifiers. This result provides a novel direction to improve the cautiousness of CNNs for classification problems. The use of DS and utility layers also slightly improves precise classification performance, and the hybridization makes it possible to reject outliers together with ambiguous patterns. Additionally, the strategy of selecting partial multi-class acts works as well as that of considering all possible acts.

Second, we extended the idea of combining DNNs and DST to semantic segmentation. In the proposed approach, the pixel-wise features from a fully convolutional network (FCN) are converted into pixel-wise mass functions by a DST-based neural-network layer for set-valued segmentation. Experiments have shown that the proposed combination improves the 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 proposed learning strategy converts the imprecise and unreliable label data into mass functions, which further improves the accuracy and calibration of the FCN models. Additionally, the proposed approach makes it possible to reject outliers together with ambiguous pixels, which provides a way to handle learning sets with incompletely labeled data.

Finally, we have proposed a fusion approach based on belief functions to combine heterogeneous DNNs. The proposed approach makes it possible to combine DNNs

trained from heterogeneous databases with different sets of classes. The combined network can classify or segment images from any of these datasets while having at least as good performance as those of the individual networks on their respective datasets. Thus, using the proposed approach, it is possible to introduce additional networks trained from new datasets with different sets of classes at any stage. This approach was shown to outperform other decision-level or feature-level fusion strategies for combining DNNs. In addition, the proposed approach allows us to use some simple and shallow neural networks to achieve a similar performance as state-of-the-art algorithms for a complex task, while reducing training difficulty and introducing small extra floating point operations. This provides a potential direction to simplify the training procedure for some very difficult problems in supervised learning.

Perspectives

The work presented in this thesis can be continued in many directions. In the following paragraphs, we sketch three of them.

In the long term, we will further extend the main idea of hybridizing DST and DNNs to other deep learning-based algorithms. Chapters 3 and 4 presents the applications of the idea on CNNs and FCNs. Many other categories of DNNs have the potential to be combined with DST. For example, long short-term memory [39, 52] and recurrent neural networks [84, 85] are two successful DNN models for processing sequences of data, such as speech or video. A DST-based neural network layer could replace the softmax layer in a long short-term memory or recurrent neural network to construct mass functions, providing more informative outputs for decision-making. Similar extensions can also be considered for deep autoencoders [53, 78], deep belief neural networks [51, 88], transformers [27, 136, 142], and so on.

Other advanced DST-based evidential neural networks could also be considered for the hybridization of DNNs and DST. In Chapter 5, we found that the distance measurement (1.29) in a DS layer suffers from the curse of dimensionality if the dimension of an input feature vector is too large, which makes the evidential feature-combination strategy unsuitable for multi-model fusion. This problem also limits the applications of evidential DNNs since we have to use the backbone networks that output feature vectors with a moderate dimension or map the output feature vectors into ones with a smaller dimension. Other model-based evidential neural networks could solve the problem, such as the one introduced in [23], which avoids the distance calculation for converting features into belief functions.

In this thesis, we have proved the feasibility of evidential DNNs in classification and segmentation tasks by combining DST-based evidential neural networks with some widely-used DNNs. However, we do not achieve the most recent state-of-the-art performance on some datasets, such as the Cityspaces and MIT-scene parsing datasets, because a large number of DNN models have been proposed during the

writing of the thesis, such as [\[76, 121, 155\]](#). We will apply our approach to some up-to-date deep networks to achieve better performances.

Appendix A

Appendix: gradient calculation in evidential neural network

In the original study of Dencœux [20], when using an evidential neural network (ENN), the loss is defined as the gap between the predicted and target mass functions and the gradients are calculated based on the loss. In the thesis, we consider an ENN as a neural network layer, called the *DS layer*, and “plug” it followed by a decision-making layer at the output of a deep neural network. In the forward-propagation, a decision-making layer converts the mass from a DS layer into other forms for decision-making, such as expected utilities in Chapters 3 and 4 and probability mass functions in Chapter 5. In the end-to-end forward-propagation of such combination, a DS layer receives the gradients w.r.t the mass from another layer and then back-propagates the gradients with respect to all layer’s parameters, which is a little different from the gradient calculation in [20]. The appendix provides the *gradient calculation with normalized output mass functions* in a DS layer that is used in Chapters 3-5¹.

Let \mathbf{m} be the normalized outputs of a DS layer for an input vector \mathbf{x} . For i -th prototype in the layer, $i = 1, \dots, n$,

$$d^i = \|\mathbf{x} - \mathbf{p}^i\| \quad (\text{A.1})$$

$$s^i = \tau^i \exp(-(\eta^i d^i)^2) \quad (\text{A.2})$$

$$\tau^i = (1 + \exp(-\xi^i))^{-1} \quad (\text{A.3})$$

$$\mathbf{m}^i = (h_1^i s^i, \dots, h_M^i s^i, 1 - s^i)^T \quad (\text{A.4})$$

$$h_j^i = \frac{(\beta_j^i)^2}{\sum_{k=1}^M (\beta_k^i)^2}. \quad (\text{A.5})$$

The vector of outputs from the ENN $\mathbf{m} = (m(\{\omega_1\}), \dots, m(\{\omega_M\}), m(\Omega))$ is computed as

$$m_j = \frac{\mu^n(\{\omega_j\})}{\sum_{j'=1}^M \mu^n(\{\omega_{j'}\}) + \mu^n(\Omega)} \quad (\text{A.6a})$$

¹In practice, users can also adopt the method of automatic differentiation and gradients in TensorFlow as a simple way, see <https://www.tensorflow.org/guide/autodiff> and https://www.tensorflow.org/guide/advanced_autodiff.

$$m_{M+1} = \frac{\mu^n(\Omega)}{\sum_{j'=1}^M \mu^n(\{\omega_{j'}\}) + \mu^n(\Omega)}, \quad (\text{A.6b})$$

with $\mu^n = \bigcap_{i=1}^n \mathbf{m}^i$ as unnormalized aggregated mass-functions outputs of the DS layer using (1.31); μ^i is the conjunctive combination of the mass $\mathbf{m}^1, \dots, \mathbf{m}^i$ with $\mu^1 = m^1, i = 1, \dots, n$. We simply notate unnormalized and normalized aggregated mass functions in (A.6) as $\mu^n = (\mu_1^n, \dots, \mu_M^n, \mu_{M+1}^n)$ and $\mathbf{m} = (m_1, \dots, m_M, m_{M+1})$, receptively.

Derivatives w.r.t. β_j^i : In the back propagation of end-to-end learning, we assume the DS layer receives the gradients w.r.t \mathbf{m} as $\partial \mathcal{L}(\mathbf{x}) / \partial m_l, l = 1, \dots, M + 1$. The derivatives of $\mathcal{L}(\mathbf{x})$ w.r.t β_j^i is given by

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \beta_j^i} = \sum_{k=1}^M \frac{\partial \mathcal{L}(\mathbf{x})}{\partial h_k^i} \frac{\partial h_k^i}{\partial \beta_j^i} \quad (\text{A.7})$$

$$= \frac{2\beta_j^i}{\left(\sum_{j'=1}^M (\beta_{j'}^i)^2\right)^2} \left[\frac{\partial \mathcal{L}(\mathbf{x})}{\partial h_j^i} \sum_{j'=1}^M (\beta_{j'}^i)^2 - \sum_{j'=1}^M (\beta_{j'}^i)^2 \frac{\partial \mathcal{L}(\mathbf{x})}{\partial h_{j'}^i} \right]. \quad (\text{A.8})$$

We then compute $\partial \mathcal{L}(\mathbf{x}) / \partial h_j^i$ as

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial h_j^i} = \sum_{l=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mu_l^n} \frac{\partial \mu_l^n}{\partial h_j^i}. \quad (\text{A.9})$$

Because the l -th unnormalized output mass μ_l^n does not depend on h_j^i for $j \neq l$, Eq. (A.9) can be simplified as

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial h_j^i} = \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mu_j^n} \frac{\partial \mu_j^n}{\partial h_j^i} \quad (\text{A.10})$$

$$= \frac{\partial \mu_j^n}{\partial h_j^i} \sum_{l=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_l} \frac{\partial m_l}{\partial \mu_j^n}, \quad (\text{A.11})$$

with $K = \sum_{l'=1}^{M+1} \mu_{l'}^n$ and

$$\frac{\partial m_l}{\partial \mu_j^n} = \begin{cases} (K - \mu_j^n) / K^2 & l = j, \\ -\mu_j^n / K^2 & l \neq j. \end{cases} \quad (\text{A.12})$$

The derivative $\partial \mu_j^n / \partial h_j^i$ is the same as the original study in [20] as

$$\frac{\partial \mu_j^n}{\partial h_j^i} = s^i (\bar{m}_j^i + \bar{m}_{M+1}^i), \quad (\text{A.13})$$

with

$$\bar{m}_j^i = \frac{m_j - m_{M+1}m_j^i/m_{M+1}^i}{m_j^i + m_{M+1}^i} \quad j = 1, \dots, M, \quad (\text{A.14})$$

$$\bar{m}_{M+1}^i = \frac{m_{M+1}}{m_{M+1}^i}. \quad (\text{A.15})$$

Hence, we can re-express (A.11) as

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial h_j^i} = s^i(\bar{m}_j^i + \bar{m}_{M+1}^i) \cdot \left(\frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_j} \frac{K - \mu_j^n}{K^2} - \sum_{l \neq j} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_l} \frac{\mu_j^n}{K^2} \right). \quad (\text{A.16})$$

Derivatives w.r.t. s^i : The derivatives of $\mathcal{L}(\mathbf{x})$ w.r.t s^i can be expressed as

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial s^i} = \sum_{l=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mu_l^n} \frac{\partial \mu_l^n}{\partial s^i}, \quad (\text{A.17})$$

where each item $\partial \mathcal{L}(\mathbf{x})/\partial \mu_l^n$, $l = 1, \dots, M + 1$ can be calculated as

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mu_l^n} = \sum_{\nu=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_\nu} \frac{\partial m_\nu}{\partial \mu_j^n}, \quad (\text{A.18})$$

using Eq. (A.12); the expression of each item $\partial \mu_l^n/\partial s^i$, $l = 1, \dots, M + 1$, is the same as the original study, such that

$$\frac{\partial \mu_l^n}{\partial s^i} = h_l^i(\bar{m}_l^i + \bar{m}_{M+1}^i) - \bar{m}_l^i, \quad (\text{A.19})$$

$$\frac{\partial \mu_{M+1}^n}{\partial s^i} = -\bar{m}_{M+1}^i. \quad (\text{A.20})$$

Thus, we now re-express Eq. (A.17) as

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial s^i} = \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mu_{M+1}^n} \frac{\partial \mu_{M+1}^n}{\partial s^i} + \sum_{j=1}^M \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mu_j^n} \frac{\partial \mu_j^n}{\partial s^i} \quad (\text{A.21})$$

$$= -\bar{m}_{M+1}^i \sum_{l=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_l} \frac{\partial m_l}{\partial \mu_{M+1}^n} + \sum_{j=1}^M [h_j^i(\bar{m}_j^i + \bar{m}_{M+1}^i) - \bar{m}_j^i] \sum_{l=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_l} \frac{\partial m_l}{\partial \mu_j^n} \quad (\text{A.22})$$

$$= -\bar{m}_{M+1}^i \left(\frac{1}{K} \cdot \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_{M+1}} - \sum_{l=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_l} \frac{\mu_l^n}{K^2} \right) + \sum_{j=1}^M [h_j^i(\bar{m}_j^i + \bar{m}_{M+1}^i) - \bar{m}_j^i] \left(\frac{1}{K} \cdot \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_j} - \sum_{l=1}^{M+1} \frac{\partial \mathcal{L}(\mathbf{x})}{\partial m_l} \frac{\mu_l^n}{K^2} \right). \quad (\text{A.23})$$

Derivatives w.r.t. η^i , ξ^i , and p_j^i : The derivatives of $\mathcal{L}(\mathbf{x})$ w.r.t η^i , ξ^i , and p_j^i can be expressed as a function of $\partial\mathcal{L}(\mathbf{x})/\partial s^i$:

$$\frac{\partial\mathcal{L}(\mathbf{x})}{\partial\eta^i} = \frac{\partial\mathcal{L}(\mathbf{x})}{\partial s^i} \frac{\partial s^i}{\partial\eta^i} = \frac{\partial\mathcal{L}(\mathbf{x})}{\partial s^i} (-2\eta^i (d^i)^2 s^i), \quad (\text{A.24})$$

$$\frac{\partial\mathcal{L}(\mathbf{x})}{\partial\xi^i} = \frac{\partial\mathcal{L}(\mathbf{x})}{\partial s^i} \frac{\partial s^i}{\partial\tau^i} \frac{d}{\tau^i} \frac{d}{\xi^i} = \frac{\partial\mathcal{L}(\mathbf{x})}{\partial s^i} \exp(-(\eta^i d^i)^2) (1 - \tau^i) \tau^i, \quad (\text{A.25})$$

$$\frac{\partial\mathcal{L}(\mathbf{x})}{\partial p_j^i} = \frac{\partial\mathcal{L}(\mathbf{x})}{\partial s^i} \frac{\partial s^i}{\partial p_j^i} = \frac{\partial\mathcal{L}(\mathbf{x})}{\partial s^i} 2(\eta^i)^2 s^i (x_j - p_j^i), \quad (\text{A.26})$$

where x_j and p_j^i are j -th element in the feature vectors of input vector \mathbf{x} and prototype \mathbf{p}^i , respectively.

Bibliography

- [1] V. Badrinarayanan, A. Kendall, and R. Cipolla. “SegNet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), pp. 2481–2495.
- [2] Y. Bi. “The impact of diversity on the accuracy of evidential classifier ensembles”. In: *International Journal of Approximate Reasoning* 53.4 (2012), pp. 584–607.
- [3] B. Biggio, B. Nelson, and P. Laskov. “Support vector machines under adversarial label noise”. In: *Proceedings of the 2011 Asian Conference on Machine Learning*. Taiwan, China, 2011, pp. 97–112.
- [4] T. Caliński and J. Harabasz. “A dendrite method for cluster analysis”. In: *Communications in Statistics-theory and Methods* 3.1 (1974), pp. 1–27.
- [5] S. Chandra and I. Kokkinos. “Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian CRFs”. In: *Proceedings of the 2016 European Conference on Computer Vision*. Springer. Amsterdam, Netherlands, 2016, pp. 402–418.
- [6] L.-C. Chen et al. “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2017), pp. 834–848.
- [7] X.-l. Chen et al. “Evidential KNN-based condition monitoring and early warning method with applications in power plant”. In: *Neurocomputing* 315 (2018), pp. 18–32.
- [8] C. Chow. “On optimum recognition error and reject tradeoff”. In: *IEEE Transactions on Information Theory* 16.1 (1970), pp. 41–46.
- [9] X. Chu et al. “Twins: Revisiting spatial attention design in vision transformers”. In: *arXiv preprint arXiv:2104.13840* (2021).
- [10] E. Chzhen et al. “Set-valued classification—overview via a unified framework”. In: *arXiv preprint arXiv:2102.12318* (2021).
- [11] B. R. Cobb and P. P. Shenoy. “On the plausibility transformation method for translating belief function models to probability models”. In: *International Journal of Approximate Reasoning* 41.3 (2006), pp. 314–330.

- [12] E. Côme et al. “Learning from partially supervised data using mixture models and belief functions”. In: *Pattern Recognition* 42.3 (2009), pp. 334–348.
- [13] M. Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*. Paradise, United States, 2016, pp. 3213–3223.
- [14] A. Das et al. “Combining multilevel contexts of superpixel using convolutional neural networks to perform natural scene labeling”. In: *Proceedings of Recent Developments in Machine Learning and Data Analytics*. Singapore: Springer, 2019, pp. 297–306.
- [15] D. Defays. “An efficient algorithm for a complete link method”. In: *The Computer Journal* 20.4 (1977), pp. 364–366.
- [16] A. Dempster. “Upper and lower probabilities induced by a multivalued mapping”. In: *Annals of Mathematical Statistics* 38 (1967), pp. 325–339.
- [17] J. Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. Miami, USA, 2009, pp. 248–255.
- [18] L. Deng and D. Yu. “Deep learning: methods and applications”. In: *Foundations and Trends in Signal Processing* 7.3–4 (2014), pp. 197–387.
- [19] T. Dencœux. “40 years of Dempster-Shafer theory”. In: *International Journal of Approximate Reasoning* 79.C (2016), pp. 1–6.
- [20] T. Dencœux. “A neural network classifier based on Dempster-Shafer theory”. In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 30.2 (2000), pp. 131–150.
- [21] T. Dencœux. “Analysis of evidence-theoretic decision rules for pattern classification”. In: *Pattern Recognition* 30.7 (1997), pp. 1095–1107.
- [22] T. Dencœux. “Decision-making with belief functions: A review”. In: *International Journal of Approximate Reasoning* 109 (2019), pp. 87–110.
- [23] T. Dencœux. “Logistic regression, neural networks and Dempster-Shafer theory: A new perspective”. In: *Knowledge-Based Systems* 176 (2019), pp. 54–67.
- [24] T. Dencœux, D. Dubois, and H. Prade. “Representations of Uncertainty in Artificial Intelligence: Beyond Probability and Possibility”. In: *A Guided Tour of Artificial Intelligence Research*. Vol. 1. Compiègne, France: Springer Verlag, 2020. Chap. 4, pp. 119–150.
- [25] T. Dencœux, O. Kanjanatarakul, and S. Sriboonchitta. “A new evidential K-nearest neighbor rule based on contextual discounting with partially supervised learning”. In: *International Journal of Approximate Reasoning* 113 (2019), pp. 287–302.

- [26] S. Destercke, F. Pichon, and J. Klein. “From set relations to belief function relations”. In: *International Journal of Approximate Reasoning* 110 (2019), pp. 46–63.
- [27] A. Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *Proceedings of the 2021 International Conference on Learning Representations*. Vienna, Austria, 2021, pp. 1–21.
- [28] S. Du et al. “Incorporating DeepLabv3+ and object-based image analysis for semantic segmentation of very high resolution remote sensing images”. In: *International Journal of Digital Earth* 14.3 (2020), pp. 357–378.
- [29] A. Ess et al. “Segmentation-based urban traffic scene understanding”. In: *Proceedings of the 20th British Machine Vision Conference*. Vol. 1. London, UK, 2009, p. 2.
- [30] M. Everingham et al. “The pascal visual object classes challenge: A retrospective”. In: *International Journal of Computer Vision* 111.1 (2015), pp. 98–136.
- [31] S. Fang et al. “Real-time object detection and semantic segmentation hardware system with deep learning networks”. In: *Proceedings of the 2018 International Conference on Field-Programmable Technology*. IEEE. Naha, Japan, 2018, pp. 389–392.
- [32] C. Farabet et al. “Learning hierarchical features for scene labeling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2012), pp. 1915–1929.
- [33] D. FCiresan et al. “Deep neural networks segment neuronal membranes in electron microscopy images”. In: *Proceedings of the 2012 International Conference on Field-Programmable Technology*. Citeseer. Sierra Nevada, United States, 2012, pp. 2843–2851.
- [34] M. Forouzanfar, N. Forghani, and M. Teshnehlab. “Parameter optimization of improved fuzzy c-means clustering algorithm for brain MR image segmentation”. In: *Engineering Applications of Artificial Intelligence* 23.2 (2010), pp. 160–168.
- [35] M. Gamal, M. Siam, and M. Abdel-Razek. “Shuffleseg: Real-time semantic segmentation network”. In: *arXiv preprint arXiv:1803.03816* (2018).
- [36] I. Golan and R. El-Yaniv. “Deep Anomaly Detection Using Geometric Transformations”. In: *Proceedings of the 2018 Conference on Neural Information Processing Systems*. Montréal, Canada, 2018, pp. 9781–9791.
- [37] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- [38] S. Gould, R. Fulton, and D. Koller. “Decomposing a scene into geometric and semantically consistent regions”. In: *Proceedings of the 12th IEEE International Conference on Computer Vision*. IEEE. Kyoto, Japan, 2009, pp. 1–8.

- [39] A. Graves. “Long short-term memory”. In: *Proceedings of Supervised sequence labelling with recurrent neural networks*. Berlin, Heidelberg: Springer, 2012, pp. 37–45.
- [40] N. Guettari, A. S. Capelle-Laizé, and P. Carré. “Blind image steganalysis based on evidential K-Nearest Neighbors”. In: *Proceedings of the 2016 IEEE International Conference on Image Processing*. Phoenix, USA, 2016, pp. 2742–2746.
- [41] C. Guo et al. “On calibration of modern neural networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Sydney, NSW, Australia: JMLR.org, 2017, 1321–1330.
- [42] J. Guo et al. “CMT: Convolutional Neural Networks Meet Vision Transformers”. In: *arXiv preprint arXiv:2107.06263* (2021).
- [43] K. Guo et al. “iFusion: Towards efficient intelligence fusion for deep learning from real-time and heterogeneous data”. In: *Information Fusion* 51 (2019), pp. 215–223.
- [44] S. Gupta et al. “Learning rich features from RGB-D images for object detection and segmentation”. In: *Proceedings of the 2014 European conference on computer vision*. Springer. Zurich, Switzerland, 2014, pp. 345–360.
- [45] T. M. Ha. “The optimum class-selective rejection rule”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6 (1997), pp. 608–615.
- [46] X. Han et al. “FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Punta Cana, Dominican Republic, 2018, pp. 4803–4809.
- [47] B. Hariharan et al. “Simultaneous detection and segmentation”. In: *Proceedings of the 2014 European conference on computer vision*. Springer. Zurich, Switzerland, 2014, pp. 297–312.
- [48] K. He et al. “Deep residual learning for image recognition”. In: *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. Las Vegas, USA, 2016, pp. 770–778.
- [49] N. Helal et al. “The capacitated vehicle routing problem with evidential demands”. In: *International Journal of Approximate Reasoning* 95 (2018), pp. 124–151.
- [50] I. Hendrickx et al. “SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals”. In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden, 2010, pp. 33–38.
- [51] G. E. Hinton. “Deep belief networks”. In: *Scholarpedia* 4.5 (2009), p. 5947.

- [52] S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.
- [53] C. Hong et al. “Multimodal deep autoencoder for human pose recovery”. In: *IEEE Transactions on Image Processing* 24.12 (2015), pp. 5659–5670.
- [54] R. Hunger. *Floating point operations in matrix-vector calculus*. Munich University of Technology, Inst. for Circuit Theory and Signal, 2005.
- [55] L. Hurwicz. “The generalized Bayes minimax principle: a criterion for decision making under uncertainty”. Cowles Commission Discussion Paper 355. 1951.
- [56] J.-Y. Jaffray. “Linear utility theory for belief functions”. In: *Operations Research Letters* 8.2 (1989), pp. 107–112.
- [57] H. Jiang et al. “Evidence fusion-based framework for condition evaluation of complex electromechanical system in process industry”. In: *Knowledge-Based Systems* 124 (2017), pp. 176–187.
- [58] L. Jiao, Q. Pan, and X. Feng. “Multi-hypothesis nearest-neighbor classifier based on class-conditional weighted distance metric”. In: *Neurocomputing* 151 (2015), pp. 1468–1476.
- [59] L. Jiao et al. “Belief rule-based classification system: Extension of FRBCS in belief functions framework”. In: *Information Sciences* 309 (2015), pp. 26–49.
- [60] D. Karimi et al. “Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis”. In: *Medical Image Analysis* 65 (2020), p. 101759.
- [61] Y. Kim, H. Lee, and E. M. Provost. “Deep learning for robust feature generation in audiovisual emotion recognition”. In: *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, Vancouver, Canada, 2013, pp. 3687–3691.
- [62] P. Krähenbühl and V. Koltun. “Efficient inference in fully connected CRFs with gaussian edge potentials”. In: *Proceedings of the 25th Annual Conference of Advances in Neural Information Processing Systems*. Granada, Spain, 2011, pp. 109–117.
- [63] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. University of Toronto, 2009.
- [64] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*. Nevada, United States, 2012, pp. 1097–1105.
- [65] N. Kumar et al. “Attribute and simile classifiers for face verification”. In: *Proceedings of the 12th International Conference on Computer Vision*. IEEE, Kyoto, Japan, 2009, pp. 365–372.

- [66] A. L. Kun et al. “Human-machine interaction for vehicles: Review and outlook”. In: *Foundations and Trends® in Human-Computer Interaction* 11.4 (2018), pp. 201–293.
- [67] M. Lachaize et al. “Evidential framework for Error Correcting Output Code classification”. In: *Engineering Applications of Artificial Intelligence* 73.0952–1976 (2018), pp. 10–21.
- [68] M. Lachaize et al. “SVM classifier fusion using belief functions: application to hyperspectral data classification”. In: *Proceedings of the 4th International Conference on Belief Functions*. Springer, Prague, Czech Republic, 2016, pp. 113–122.
- [69] Y. LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [70] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.
- [71] Y. LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551.
- [72] G. Li et al. “Standing-posture recognition in human-robot collaboration based on deep learning and the Dempster-Shafer evidence theory”. In: *Sensors* 20.4 (2020), p. 1158.
- [73] C. Lian, S. Ruan, and T. Dencœux. “An evidential classifier based on feature selection and two-step classification strategy”. In: *Pattern Recognition* 48 (2015), pp. 2318–2327.
- [74] M. Lin, Q. Chen, and S. Yan. “Network in network”. In: *Proceedings of the 2014 International Conference on Learning Representations*. Banff, Canada, 2014, pp. 1–10.
- [75] X. Liu, Z. Deng, and Y. Yang. “Recent progress in semantic image segmentation”. In: *Artificial Intelligence Review* 52.2 (2019), pp. 1089–1106.
- [76] Z. Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *arXiv preprint arXiv:2103.14030* (2021).
- [77] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition*. Boston, USA, 2015, pp. 3431–3440.
- [78] K. G. Lore, A. Akintayo, and S. Sarkar. “LLNet: A deep autoencoder approach to natural low-light image enhancement”. In: *Pattern Recognition* 61 (2017), pp. 650–662.
- [79] M. Lukasik et al. “Does label smoothing mitigate label noise?” In: *Proceedings of the 37th International Conference on Machine Learning*. Vienna, Austria: PMLR, 2020, pp. 6448–6458.

- [80] H. Luo et al. “Urban change detection based on Dempster-Shafer theory for multitemporal very high-resolution imagery”. In: *Remote Sensing* 10.7 (2018), p. 980.
- [81] Y. Luo et al. “Direction concentration learning: Enhancing congruency in machine learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.6 (2019), pp. 1928–1946.
- [82] L. Ma and T. Denceux. “Partial classification in the belief function framework”. In: *Knowledge-Based Systems* 214 (2021), p. 106742.
- [83] A. Meyer-Baese, A. Meyer-Baese, and V. J. Schmid. *Pattern recognition and signal analysis in medical imaging*. Academic Press, 2004.
- [84] T. Mikolov et al. “Extensions of recurrent neural network language model”. In: *Proceedings of the 2011 IEEE international conference on acoustics, speech and signal processing*. IEEE. Prague, Czech Republic, 2011, pp. 5528–5531.
- [85] T. Mikolov et al. “Recurrent neural network based language model”. In: *Proceedings of the 11th Annual Conference of the International Speech Communication Association*. Makuhari, Japan, 2010.
- [86] P. Minary et al. “Evidential joint calibration of binary SVM classifiers using logistic regression”. In: *Soft Computing* 23.13 (2019), pp. 4655–4671.
- [87] P. Minary et al. “Face pixel detection using evidential calibration and fusion”. In: *International Journal of Approximate Reasoning* 91 (2017), pp. 202–215.
- [88] A.-r. Mohamed, G. E. Dahl, and G. Hinton. “Acoustic modeling using deep belief networks”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1 (2011), pp. 14–22.
- [89] T. Mortier et al. “Efficient set-valued prediction in multi-class classification”. In: *Data Mining and Knowledge Discovery* 35 (2021), pp. 1435–1469.
- [90] N. Natarajan et al. “Learning with noisy labels”. In: *Proceedings of the 2013 Advances in Neural Information Processing Systems*. Sierra Nevada, USA, 2013, pp. 1196–1204.
- [91] H. T. Nguyen. *An introduction to random sets*. Chapman and Hall/CRC, 2006.
- [92] L. D. Nguyen et al. “Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation”. In: *Proceedings of the 2018 IEEE International Symposium on Circuits and Systems*. IEEE. Florence, Italy, 2018, pp. 1–5.
- [93] M.-E. Nilsback and A. Zisserman. “Automated flower classification over a large number of classes”. In: *Proceedings of the 6th Indian Conference on Computer Vision, Graphics & Image Processing*. IEEE. Bhubaneswar, India, 2008, pp. 722–729.

- [94] F. Ning et al. “Toward automatic phenotyping of developing embryos from videos”. In: *IEEE Transactions on Image Processing* 14.9 (2005), pp. 1360–1371.
- [95] H. Noh, S. Hong, and B. Han. “Learning deconvolution network for semantic segmentation”. In: *Proceedings of the 2015 IEEE International Conference on Computer Vision*. Santiago, Chile, 2015, pp. 1520–1528.
- [96] M. O’Hagan. “Aggregating template or rule antecedents in real-time expert systems with fuzzy set logic”. In: *Proceedings of Twenty-Second Asilomar Conference on Signals, Systems and Computers*. Vol. 2. Pacific Grove, USA, 1988, pp. 681–689.
- [97] G. Pang et al. “Deep Learning for Anomaly Detection: A Review”. In: *ACM Computing Surveys* 54.2 (2021), pp. 1–38.
- [98] O. M. Parkhi et al. “Cats and Dogs”. In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*. Rhode Island, USA: IEEE, 2012, pp. 3498–3505.
- [99] F. Pichon, A.-L. Jousselme, and N. B. Abdallah. “Several shades of conflict”. In: *Fuzzy Sets and Systems* 366 (2019), pp. 63–84.
- [100] K. J. Piczak. “Environmental sound classification with convolutional neural networks”. In: *Proceedings of the 25th International Workshop on Machine Learning for Signal Processing*. IEEE. Boston, USA, 2015, pp. 1–6.
- [101] P. Pinheiro and R. Collobert. “Recurrent convolutional neural networks for scene labeling”. In: *Proceedings of the 2014 International Conference on Machine Learning*. PMLR. Beijing, China, 2014, pp. 82–90.
- [102] B. Quost, M.-H. Masson, and T. Dencœux. “Classifier fusion in the Dempster-Shafer framework using optimized t-norm based combination rules”. In: *International Journal of Approximate Reasoning* 52.3 (2011), pp. 353–374.
- [103] A. Rame, R. Sun, and M. Cord. “MixMo: Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks”. In: *arXiv preprint arXiv:2103.06132* (2021).
- [104] M. Ren et al. “Learning to Reweight Examples for Robust Deep Learning”. In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Stockholm, Sweden: PMLR, 2018, pp. 4334–4343.
- [105] T. Ridnik et al. *ImageNet-21K Pretraining for the Masses*. 2021. arXiv: [2104.10972](https://arxiv.org/abs/2104.10972) [cs.CV].
- [106] A. Romero et al. “Fitnets: Hints for thin deep nets”. In: *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, USA, 2015, pp. 1–13.

- [107] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, Strasbourg, France, 2015, pp. 234–241.
- [108] J. Salamon, C. Jacoby, and J. P. Bello. “A Dataset and Taxonomy for Urban Sound Research”. In: *Proceedings of the 22nd ACM International Conference on Multimedia*. New York, USA: Association for Computing Machinery, 2014, pp. 1041–1044.
- [109] H. Salehinejad, Z. Wang, and S. Valaee. “Ising Dropout with Node Grouping for Training and Compression of Deep Neural Networks”. In: *Proceedings of the 2019 IEEE Global Conference on Signal and Information Processing*. Ottawa, Canada, 2019, pp. 1–5.
- [110] E. Sánchez-Nielsen, L. Antón-Canalís, and M. Hernández-Tejera. “Hand gesture recognition for human-machine interaction”. In: *Journal of WSCG* 12 (2004), pp. 1–8.
- [111] F. Scarselli et al. “Graph neural networks for ranking Web pages”. In: *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*. Compiègne, France, 2005, pp. 666–672.
- [112] F. Scarselli et al. “The graph neural network model”. In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.
- [113] C. N. N. for Sentence Classification. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, 2014, pp. 1746–1751.
- [114] G. Shafer. *A mathematical theory of evidence*. Princeton: Princeton University Press, 1976.
- [115] R. Sibson. “SLINK: An optimally efficient algorithm for the single-link cluster method”. In: *The Computer Journal* 16.1 (Jan. 1973), pp. 30–34.
- [116] K. Simonyan and A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *Proceedings of the 3rd International Conference on Learning Representations*. San Diego, USA, 2015, pp. 1–14.
- [117] P. Smets. “Constructing the pignistic probability function in a context of uncertainty”. In: *Proceedings of the 5th Uncertainty in Artificial Intelligence*. Ed. by M. Henrion et al. Amsterdam, Netherlands: North-Holland, 1990, pp. 29–40.
- [118] P. Smets. “Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem”. In: *International Journal of Approximate Reasoning* 9.1 (1993), pp. 1–35.

- [119] R. Soua, A. Koesdwiady, and F. Karray. “Big-data-generated traffic flow prediction using deep learning and dempster-shafer theory”. In: *Proceedings of the 2016 International Joint Conference on Neural Networks*. IEEE. Vancouver, Canada, 2016, pp. 3195–3202.
- [120] T. M. Strat. “Decision analysis using belief functions”. In: *International Journal of Approximate Reasoning* 4.5–6 (1990), pp. 391–417.
- [121] R. Strudel et al. “Segmenter: Transformer for Semantic Segmentation”. In: *arXiv preprint arXiv:2105.05633* (2021).
- [122] Z.-G. Su et al. “Evidential K-NN classification with enhanced performance via optimizing a class of parametric conjunctive t-rules”. In: *Knowledge-Based Systems* 142 (2018), pp. 7–16.
- [123] H.-I. Suk and D. Shen. “Deep learning-based feature representation for AD/MCI classification”. In: *Proceedings of the 2013 International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. Berlin, Heidelberg, 2013, pp. 583–590.
- [124] V Suma. “Computer vision for human-machine interaction-review”. In: *Journal of Trends in Computer Science and Smart Technology* 1.02 (2019), pp. 131–139.
- [125] K. Suzuki. “Overview of deep learning in medical imaging”. In: *Radiological Physics and Technology* 10.3 (2017), pp. 257–273.
- [126] P. Thodoroff, J. Pineau, and A. Lim. “Learning robust features using deep learning for automatic seizure detection”. In: *Proceedings of the 2016 Machine Learning for Healthcare Conference*. PMLR. Durham, USA, 2016, pp. 178–190.
- [127] Z. Tian et al. “Deep learning and Dempster-Shafer theory Based insider threat detection”. In: *Mobile Networks and Applications* (2020), pp. 1–10.
- [128] J. Tighe and S. Lazebnik. “Superparsing: scalable nonparametric image parsing with superpixels”. In: *Proceedings of the 2010 European conference on Computer Vision*. Springer. Berlin, Heidelberg, 2010, pp. 352–365.
- [129] Z. Tong, P. Xu, and T. Denœux. “An evidential classifier based on Dempster-Shafer theory and deep learning”. In: *Neurocomputing* 450 (2021), pp. 275–293.
- [130] Z. Tong, P. Xu, and T. Denœux. “ConvNet and Dempster-Shafer Theory for Object Recognition”. In: *Processing of the 13th International Conference on Scalable Uncertainty Management*. Compiègne, France: Springer International Publishing, 2019, pp. 368–381.
- [131] Z. Tong, P. Xu, and T. Denœux. “Evidential fully convolutional network for semantic segmentation”. In: *Applied Intelligence* 51 (2021), pp. 6376–6399.

- [132] Z. Tong, P. Xu, and T. Denœux. “Fusion of evidential CNN classifiers for image classification”. In: *Proceedings of the 6th International Conference on Belief Functions*. Springer, Shanghai, China, 2021, pp. 168–176.
- [133] C.-H. Tseng et al. “UPANets: Learning from the Universal Pixel Attention Networks”. In: *arXiv preprint arXiv:2103.08640* (2021).
- [134] J. Vandoni, E. Aldea, and S. Le Hégarat-Masclé. “An evidential framework for pedestrian detection in high-density crowds”. In: *Proceedings of the 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Lecce, Italy, 2017, pp. 1–6.
- [135] J. Vandoni, S. L. Hégarat-Masclé, and E. Aldea. “Augmenting deep learning performance in an evidential multiple classifier system”. In: *Sensors* 19.21 (2019), p. 4664.
- [136] A. Vaswani et al. “Attention is all you need”. In: *Proceedings of the 31st Conference on Neural Information Processing Systems*. Long Beach, USA, 2017, pp. 5998–6008.
- [137] F. Voorbraak. “A computationally efficient approximation of Dempster-Shafer theory”. In: *International Journal of Man-Machine Studies* 30.5 (1989), pp. 525–536.
- [138] P. Walley. “Statistical reasoning with imprecise probabilities”. In: *Journal of the American Statistical Association* 88.422 (1993), pp. 700–703.
- [139] J. H. Ward Jr. “Hierarchical grouping to optimize an objective function”. In: *Journal of the American Statistical Association* 58.301 (1963), pp. 236–244.
- [140] Q. Wei, N. Dobigeon, and J.-Y. Tourneret. “Bayesian fusion of multi-band images”. In: *IEEE Journal of Selected Topics in Signal Processing* 9.6 (2015), pp. 1117–1127.
- [141] P. Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.
- [142] T. Wolf et al. “Transformers: State-of-the-art natural language processing”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online, 2020, pp. 38–45.
- [143] S. Xie et al. “Aggregated residual transformations for deep neural networks”. In: *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*. Honolulu, USA: IEEE, 2017, pp. 1492–1500.
- [144] J. Xu et al. “Understanding and Improving Layer Normalization”. In: *Proceedings of the 2019 Conference on Neural Information Processing Systems*. Vol. 32. Vancouver, Canada: Curran Associates, Inc., 2019, pp. 1–11.
- [145] P. Xu. “Information fusion for scene understanding”. PhD thesis. Université de Technologie de Compiègne, 2014.

- [146] P. Xu et al. “Multimodal information fusion for urban scene understanding”. In: *Machine Vision and Applications* 27.3 (2016), pp. 331–349.
- [147] Q. Xu, C. Zhang, and B. Sun. “Emotion recognition model based on the Dempster-Shafer evidence theory”. In: *Journal of Electronic Imaging* 29.2 (2020), p. 023018.
- [148] R. R. Yager. “On ordered weighted averaging aggregation operators in multi-criteria decision-making”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 18.1 (1988), pp. 183–190.
- [149] R. R. Yager and L. Liu. *Classic works of the Dempster-Shafer theory of belief functions*. Vol. 219. Berlin, Heidelberg: Springer, 2008.
- [150] X. Yi, E. Walia, and P. Babyn. “Generative adversarial network in medical imaging: A review”. In: *Medical Image Analysis* 58 (2019), p. 101552.
- [151] Y. Yoon et al. “Learning a deep convolutional network for light-field image super-resolution”. In: *Proceedings of the 2015 IEEE International Conference on Computer Vision Workshops*. Santiago, Chile, 2015, pp. 24–32.
- [152] L. A. Zadeh. “Fuzzy sets as a basis for a theory of possibility”. In: *Fuzzy Sets and Systems* 1.1 (1978), pp. 3–28.
- [153] M. D. Zeiler and R. Fergus. “Stochastic pooling for regularization of deep convolutional neural networks”. In: *Proceedings of the 1st International Conference on Learning Representations*. Scottsdale, USA, 2013, pp. 1–10.
- [154] D. Zeng et al. “Relation classification via convolutional deep neural network”. In: *Proceedings of the 25th International Conference on Computational Linguistics*. Dublin, Ireland, 2014, pp. 2335–2344.
- [155] S. Zheng et al. “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers”. In: *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Online, 2021, pp. 6881–6890.
- [156] B. Zhou et al. “Semantic understanding of scenes through the ADE20k dataset”. In: *International Journal of Computer Vision* 127 (2019), pp. 302–321.