©2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Published in 2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE).

DOI: 10.1109/FUZZ-IEEE.2019.8858790

Introducing Fuzzy Layers for Deep Learning

Stanton R. Price
U.S. Army Engineer
Research and Development Center
Vicksburg, MS USA
stantonprice@yahoo.com

Steven R. Price
Department of Electrical Engineering
Mississippi College
Clinton, MS USA
srprice1@mc.edu

Derek T. Anderson

Department of Electrical Engineering

and Computer Science

University of Missouri

Columbia, MO, USA

andersondt@missouri.edu

Abstract—Many state-of-the-art technologies developed in recent years have been influenced by machine learning to some extent. Most popular at the time of this writing are artificial intelligence methodologies that fall under the umbrella of deep learning. Deep learning has been shown across many applications to be extremely powerful and capable of handling problems that possess great complexity and difficulty. In this work, we introduce a new layer to deep learning: the fuzzy layer. Traditionally, the network architecture of neural networks is composed of an input layer, some combination of hidden layers, and an output layer. We propose the introduction of fuzzy layers into the deep learning architecture to exploit the powerful aggregation properties expressed through fuzzy methodologies, such as the Choquet and Sugueno fuzzy integrals. To date, fuzzy approaches taken to deep learning have been through the application of various fusion strategies at the decision level to aggregate outputs from state-of-the-art pre-trained models, e.g., AlexNet, VGG16, GoogLeNet, Inception-v3, ResNet-18, etc. While these strategies have been shown to improve accuracy performance for image classification tasks, none have explored the use of fuzzified intermediate, or hidden, layers. Herein, we present a new deep learning strategy that incorporates fuzzy strategies into the deep learning architecture focused on the application of semantic segmentation using per-pixel classification. Experiments are conducted on a benchmark data set as well as a data set collected via an unmanned aerial system at a U.S. Army test site for the task of automatic road segmentation, and preliminary results are promising.

Index Terms—fuzzy layers, deep learning, fuzzy neural nets, semantic segmentation, fuzzy measure, fuzzy integrals

I. INTRODUCTION

Artificial intelligence (AI) has emerged over the past decade as one of the most promising technologies for advancing mankind in a multitude of ways, from medicine discovery and disease diagnostics to autonomous vehicles, semantic segmentation, and personal assistants. For many years there has been a desire to create computer algorithms/machines that are able to replace or assist humans in signal understanding for tasks such as automatic buried explosive hazard detection, vehicle navigation, object recognition, and object tracking. AI has grown to loosely encompass a number of state-of-the-art

This work was partially supported under the Maneuver in Complex Environments R&D program to support the U.S. Army ERDC. This effort is also based on work supported by the Defense Threat Reduction Agency/Joint Improvised-Threat Defeat Organization (DTRA/JIDO). Any use of trade names is for descriptive purposes only and does not imply endorsement by the U.S. Government. Permission to publish was granted by Director, Geotechnical and Structures Laboratory, U.S. Army ERDC. Approved for public release; distribution is unlimited.

technologies across many fields, e.g., pattern recognition, *machine learning* (ML), *neural networks* (NNs), computational intelligence, evolutionary computation, and so on. Recently, much buzz has surrounded *deep learning* (DL) for its ability to provide desirable results for a number of different applications.

AI has been shown to have great potential for finding optimized solutions to various problems across multiple domains. This technology has been heavily researched for computer vision applications [1], [2], speech translations [3], [4], and optimization tasks [5], [6]. Herein, the focus is on an extremely relevant and popular branch of AI: deep learning. DL has achieved much success in recent years on computer vision applications and has benefited from the surge of attention being given to advance its theories to being extremely generalizable for many problems. Part of DL's recent resurgence is the implementation of its network architecture being well suited for processing on powerful, highly parallelized GPUs. This has allowed for extremely complex and deep network architectures to be developed that were infeasible to implement on older compute technologies.

Fusion is a powerful technique used to combine information from different sources. These sources could be different features extracted, decisions, sensor output, etc., as well as different combinations thereof. Fusion methodologies often strive to improve system performance by combining information in a beneficial way that enables more discriminatory power to the system in some form. This could be through the realization of more robust features that generalize well from one domain to another. For the task of classification, fusion could be used to combine multiple decision makers, e.g., classifiers, to improve overall accuracy performance. Fusion is a very rich and powerful technique that, when implemented appropriately, can lead to major algorithm improvements. Fusion is commonly associated with fuzzy techniques, as Fuzzy Logic [7], [8] naturally lends itself to gracefully considering data with varying degrees of belief. Ensemble approaches [9] are also commonly used for fusion tasks. Generally, most fusion strategies attempt to properly assign weights that encode the significance, or importance, to the different information sources, and these weights are the driving mechanism behind the fused result. Historically, weights used when fusing multiple information sources are either human-derived or found via an optimization function/strategy such as group lasso [10], [11]. However, there has been little-to-no research done on utilizing DL for optimizing fusion performance. Herein, we propose a new strategy to explore the potential benefits of combining DL with fuzzy-based fusion techniques. Specifically, we introduce *fuzzy layers* to the DL architecture.

The remainder of this work is organized as follows. In Section II, related works are presented that explore using fusion strategies to improve the classification performance of the outputs from different state-of-the-art DL models (fusion strategies and DL have been compartmentalized in their utilization). Fuzzy layers are introduced in Section III along with the intuition behind this new strategy and their integration into the DL architecture. Experiments and results are detailed in Section IV, and in Section V, we conclude the paper.

II. RELATED WORK

As noted previously, DL is becoming the standard approach for classification tasks. However, the performances exhibited by DL classifiers are often the result of exhaustive evaluation of hyperparamenters and various network architectures for the particular data set used in the work. Fusion techniques can help alleviate the comprehensive evaluation by combining the classification outputs from multiple DL classifiers and thus taking advantage of different DL classifier strengths. That is, if the strengths of multiple classifiers can be appropriately fused, then finding the optimal solution may not require finding the ideal parameters and architecture for a particular data set.

Recently, fusion strategies have been employed that aggregate pre-trained models for improved classification performance. The appropriate fusion strategy largely considers the formats of classifier outputs. Traditionally, a classifier output consists of a definitive label (i.e., hard-decision), and typically, majority voting is the fusion strategy implemented. However, if the classifier can generate soft membership measures (e.g., fuzzy measures), the fusion strategy implemented can vary greatly [12]–[15].

Fusion strategies, notably those associated with *fuzzy measures* (FMs), are conventionally applied at the decision level to aggregate outputs and improve performance. For example, DL classifier outputs were fused in [16]–[18] to improve classification performance in remote sensing applications by deriving FMs for each class and then fusing the measures with the classifiers' outputs through either the Sugeno or *Choquet integral* (ChI) [19]. Still, fusion strategies occurring at the input level can also benefit classification performance.

Rather than attempt to perform fusion at either the output or feature level, it is the attempt of this work to incorporate fusion techniques (utilizing FMs) within the architecture of a DL classification system. While efforts have developed techniques applying fuzzy sets and fuzzy inference systems in NNs, application of fuzzy strategies concerning DL architectures is limited [20]. One recent approach to implementing fuzzy sets in DL evaluated the use of Sugeno fuzzy inference systems as the node in the hidden layer of an NN and therefore could be extended to DL architectures by extending the concept [21].

III. METHODOLOGY

In this section, we introduce our proposed fuzzy layer to incorporate fuzzy-based fusion approaches directly into the DL architecture. First, we briefly discuss the problem that is being considered herein: semantic image segmentation using DL. Semantic segmentation is the process of assigning class labels (e.g., car, tank, road, building, etc.) to each pixel in an image. DL for semantic segmentation is most commonly implemented in what can be separated into two parts: (1) standard CNN network with the exception of an output layer and (2) an upsampling CNN network on the back half that produces a perpixel class label output. Zeiler et al. introduced deconvolution networks in [22] for the task of visualizing learned CNNs to help bridge the gap of understanding what a CNN has learned. Therein, Zeiler et al. defined a deconvolution network that attempts to reconstruct a feature map that identifies what a CNN has learned through unpooling, rectification, and filtering. In [23], Noh et al. modified this approach for semantic segmentation by using DL to learn the up-sampling, or deconvolution, filters rather than inverting (via transposing) the corresponding filters learned in the convolution network. Herein, we implement a similar approach as Noh et al., utilizing DL strategies to learn the up-sampling filters rather than performing true deconvolution to reconstruct the feature maps at each layer. Additionally, this work is focused strictly on road segmentation, i.e., each pixel is labeled as either road or non-road. Representing the architecture of our learned model as $f(x, \gamma)$, where γ represents the parameters that are learned by the network such that the error for an example x_i given its ground-truth y_i is minimized and can be described

$$\hat{\gamma} = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} (L(f(x_i, \gamma), y_i), \tag{1}$$

where N is the training data set and L is the softmax (cross-entropy) loss.

As this paper is focused on the introduction of a new fuzzy layer that can be injected directly into the DL architecture, a defined network architecture is not presented. Rather, we explore different use cases of the fuzzy layers at different points throughout the network architecture. For comparison, the fuzzy layers are utilized either in the down-sampling (convolution network), up-sampling ("deconvolution" network), or both sections of the semantic segmentation network. To maintain consistency in our exploration, a template network architecture was used such that the only change in the network architecture was the inclusion or removal of one or more fuzzy layers. The details of the architecture template used are given in Table I, with '*' denoting the points in which a fuzzy layer might be included in the architecture. **Note:** it is *not required* that a fuzzy layer be incorporated after a rectified linear unit (ReLU); this occurs in the results presented herein to maintain more consistency across experiments in this exploratory work. It would have been equally valid to implement a fuzzy layer after any convolution or pooling layer (referencing layers utilized in this architecture). The best way(s) to implement fuzzy layers

TABLE I

Template architecture in detail. The '*' represents locations in the architecture that a fuzzy layer might be included herein. This is not a restriction. N_f and N_{cl} represent the number of fused outputs at that layer and the number of classes, respectively.

Name	Kernel Size	Stride	Output Size
input_data	-	-	$512 \times 512 \times 3$
conv1_1	5×5	1	$512 \times 512 \times 64$
conv1_2	5×5	1	$512 \times 512 \times 64$
relu1	-	-	$512 \times 512 \times 64$
*	-	-	$512 \times 512 \times N_f$
pool1	2×2	2	$256 \times 256 \times 64$
conv2_1	5×5	1	$256 \times 256 \times 64$
relu2	-	-	$256 \times 256 \times 64$
*	-	-	$256 \times 256 \times N_f$
pool2	2×2	2	$128 \times 128 \times 64$
conv3_1	5×5	1	$128 \times 128 \times 64$
relu3	-	-	$128 \times 128 \times 64$
*	-	-	$128 \times 128 \times N_f$
up-conv1	6×6	2	$256 \times 256 \times 30$
relu4	-	-	$256 \times 256 \times 30$
*	-	-	$256 \times 256 \times N_f$
up-conv2	6×6	2	$512 \times 512 \times 30$
relu5	-	-	$512 \times 512 \times 30$
*	-	-	$512 \times 512 \times N_f$
output	-	-	$512 \times 512 \times N_{cl}$

within the DL architecture is an open question and one that requires additional research. Technically, a fuzzy layer can be implemented anywhere in the DL architecture as long as it follows the input layer and precedes the output layer.

A. Fuzzy Layer

Theoretically, the fuzzy layer can encompass any fuzzy aggregation strategy desired to be utilized. Herein, we focus on the Choquet integral as the means for fusion. Let $X = \{x_1, \ldots, x_N\}$ be N sources, e.g., sensor, human, or algorithm. In general, an aggregation function is a mapping of data from our N sources, denoted by $h(x_i) \in \mathcal{R}$, to data, $f(h(x_1), \ldots, h(x_N), \Theta) \in \mathcal{R}$, where Θ are the parameters of f. The ChI is a nonlinear aggregation function parameterized by the FM. FMs are often used to encode the (possibly subjective) worth of different subsets of information sources. Thus, the ChI parameterized by the FM provides a way to combine the information encoded in the FM with the (objective) evidence or support of some hypothesis, e.g., sensor data, algorithm outputs, expert opinions, etc. The FM and ChI are defined as follows.

Definition 1. (Fuzzy Measure) For a finite set of N information sources, X, the FM is a set-valued function $g: 2^X \rightarrow [0,1]$ with the following conditions:

- 1) (Boundary Conditions) $g(\emptyset) = 0$ and g(X) = 1
- 2) (Monotonicity) If $A, B \subseteq X$ with $A \subseteq B$, then $g(A) \le g(B)$.

Note, if X is an infinite set, there is a third condition guaranteeing continuity.

Definition 2. (Choquet Integral) For a finite set of N information sources, X, FM g, and partial support function $h: X \to [0, 1]$, the ChI is

$$\int h \circ g = \sum_{i=1}^{N} w_i h(x_{\pi(i)}), \tag{2}$$

where $w_i = (G_{\pi(i)} - G_{\pi(i-1)})$, $G_{(i)} = g(\{x_{\pi(1)}, \dots, x_{\pi(i)}\})$, $G_{\pi(0)} = 0$, $h(x_i)$ is the strength in the hypothesis from source x_i , and $\pi(i)$ is a sorting on X such that $h(x_{\pi(1)}) \geq \dots \geq h(x_{\pi(N)})$

The FM can be obtained in a number of ways: human defined, quadratic program, learning algorithm, S-Decomposable measure (e.g., Sugeno λ -fuzzy measure), etc. Herein, we define the FM to be five known OWA operators and one random (but valid) OWA operator. Specifically, the more well known operators used are max, min, average, soft max, and soft min. The top 5 sources (i.e., convolution/deconvolution filter outputs) were sorted based on their entropy value and fused via the ChI. Therefore, the fuzzy layer accepts the output from the previous layer as its input, sorts the images (sources) by some metric (entropy used herein), and performs the ChI for each of the defined FMs resulting in six fused outputs (we have six different FMs) that are passed on to the next layer in the network. An example of a potential fuzzy layer implementing the ChI as its aggregation method is shown in Figure 1.

B. Why Have Fuzzy Layers?

As the architecture of DL continues to grow more complex, there is a need to help alleviate the ill-conditioning that is prone to occur during learning due to the weights approaching zero. Additionally, a well-known problem when training deep networks is the internal-covariate-shift problem [24]. This results in difficulty to optimize the network due to the input distributions changing at each layer over iterations during training with the changes in distribution being amplified through propagation across layers. While there are other approaches that seek to help with this (e.g., batch normalization [24]), fusion poses itself as a viable solution to aiding with this problem. One example of the potential benefit of this is fusion can take 10s, 100s, etc. of inputs (outputs of previous layers) and condense that information into a fraction of images. For example, if the output of a convolution, ReLU, or pooling layer had 256 feature maps, a fuzzy layer could be utilized to fuse these 256 feature maps down to some arbitrary reduced number of feature maps, e.g., 30, that capture relevant information in unique ways from all, or some subset of the 256 feature maps (dependent on the FMs used as well as the metric used for sorting). Thus, this alone has two potential major benefits: (1) reduced model complexity and (2) improving the utilization of the information learned at the previous layer in the network.

IV. EXPERIMENTS & RESULTS

This section first describes the dataset and implementation details. Next, we present and analyze the results for various

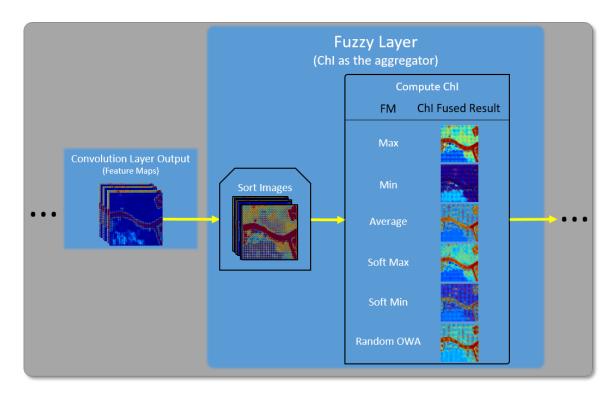


Fig. 1. Illustration of the fuzzy layer. In this example, the layer feeding into the fuzzy layer is a convolution layer. The feature maps are passed as inputs to the fuzzy layer where they are then sorted, as required for the ChI, based on some metric (entropy used herein). The ChI is computed for six different FMs, producing six ChI fused resultant images. These six images are then passed on to the next layer in the network.

network configurations as we investigate the implementation of fuzzy layers.

A. Dataset

The dataset was collected from an UAS with a mounted MAPIR Survey2 RGB camera. Specifically, the sensor used is a Sony Exmor IMX206 16MP RGB sensor, which produces a 24-bit 4,608×3,456 pixel RGB image. The UAS was flown at an altitude of approximately 60 meters from the ground. The dataset was captured by flying the UAS in a grid-like pattern over an area of interest at a U.S. Army test site. The dataset used in this work comes from a single flight over this area, which contains 252 images, 20 of which were selected as training data. The imagery was scaled to size (i.e., 512×512 pixels) using bilinear interpolation to make them more digestible by the DL algorithms implemented on the computer system used herein. As is common with training for DL algorithms (and ML algorithms in general), data augmentation strategies are employed in part to increase the amount of training data available during learning, but also to lead to more robust solutions. Herein, each image from the training data set is rotated 90°, 180°, and 270° to provide a total of 80 images used for training (example shown in Figure 2). Finally, the image road mask for each of the 252 instances were annotated by hand (see Figure 3).

B. Implementation Details

We based the template network architecture shown in Table I on the VGG-16 framework [25]. There are modifications to



Fig. 2. Example of data set augmentation used (image rotation). Starting at the top left and going clockwise: 0° , 90° , 180° , 270° .

the number of convolution layers and filters used throughout the network; however, the VGG-16 framework served as the motivation behind the defined template architecture implemented. Initially, we implemented the standard stochastic gra-



Fig. 3. Road masks shown (right column) for two sample images.

dient descent with momentum for optimization but achieved poor results. The Adam algorithm [26] provided the best optimization performance on this dataset and network design and was used for all experiments reported, with the initial learning rate, gradient decay rate, and squared gradient decay weight set to 0.1, 0.9, and 0.999, respectively. Dropout [27] is used after pooling with a dropout rate of 50%.

C. Evaluation

To measure network classification accuracy performance, we utilize an evaluation protocol that is based on *Intersection over Union* (IoU) between the ground-truth and predicted segmentation. We report the mean and standard deviation of the IoU scores for all test images for each approach investigated. For clarity, we denote the different experiments (i.e., different architecture configurations) as follows

- baseline- no fuzzy layers;
- conv-FLs- fuzzy layers are implemented after 'relu1', 'relu2', and 'relu3' in the convolution network (down-sampling half);
- deconv-FLs- fuzzy layers are implemented after 'relu4' and 'relu5' in the "deconvolution" network (up-sampling half):
- **conv-FLs+deconv-FLs-** fuzzy layers are implemented after every ReLU.

The quantitative results for these different architectures are presented in Table II.

From these preliminary results, we see that the inclusion of fuzzy layers shows promise for improving DL performance (in terms of accuracy). In particular, these results indicate that fuzzy layers are better utilized in the deconvolution phase of the architecture. Example feature maps randomly selected from one instance at each layer (the ReLU output is ommited in the deconvolution network for compactness) are shown in Figure 4. Looking specifically at the feature maps denoted

TABLE II EVALUATION RESULTS ON TEST DATASET FOR ROAD SEGMENTATION.

Mean	Std. Dev.
78.22%	12.3%
62.43%	21.5%
80.79%	14.8%
68.76%	20.7%
	78.22% 62.43% 80.79 %

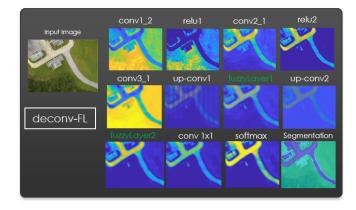


Fig. 4. Example feature maps and final segmentation for a randomly selected image. Also, the feature maps were randomly chosen at each layer.

as 'fuzzyLayer1' and 'fuzzyLayer2', we see evidence of the fuzzy layers' aggregation strategy accumulating evidence of road information. We note that 'decony-FLs' only performs approximately 2\% better than the baseline method, while having a slightly higher standard deviation. Nevertheless, this helps show fuzzy layers potential of improving classification performance. It is our conjecture that, for this problem, applying the fuzzy layers during the convolution stage (results shown as 'conv-FLs') results in the loss of too much information from prior layers (after each ReLU, we summarize 64 filters down to 6- this is likely too extreme for such an early stage of learning). Hence, we see a noticeable drop in performance for both experiments that include fuzzy layers during the convolution stage ('conv-FLs' and 'conv-FLs+deconv-FLs'). However, there are a number of factors involved that could lead to improved performance during the convolution phase, e.g., increased number of FMs, perhaps a different metric for sorting should be used, different fuzzy aggregation method, etc. It should also be noted that the inclusion of the fuzzy layers had minimal impact on training time (total training time increased by seconds to a few minutes at most).

V. CONCLUSION

We proposed a new layer to be used for DL: the fuzzy layer. The proposed fuzzy layer is extremely flexible, capable of implementing any fuzzy aggregation method desired, as well as capable of being included anywhere in the network architecture, depending on the desired behavior of the fuzzy layer. This work was focused on the introduction and early exploration of the fuzzy layer, and additional research is needed to further advance the fuzzy layer for DL. For ex-

ample, future work should consider investigating the metric used for sorting the information sources and its effect on accuracy performance. Future work is planned to investigate how the FM should be defined for aggregating via fuzzy integrals. Additionally, where are the fuzzy layers best utilized in the network architecture (problem dependent; however, can a general guidance be developed)? These are but a few questions that need to be addressed for the fuzzy layer and its implementation.

REFERENCES

- M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural infor*mation processing systems, 2012, pp. 1097–1105.
- [3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.
- [4] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.
- [5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [6] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.
- [7] L. A. Zadeh, "A fuzzy-algorithmic approach to the definition of complex or imprecise concepts," in *Systems Theory in the Social Sciences*. Springer, 1976, pp. 202–282.
- [8] T. J. Ross, Fuzzy logic with engineering applications. John Wiley & Sons, 2005.
- [9] C. Zhang and Y. Ma, Ensemble machine learning: methods and applications. Springer, 2012.
- [10] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B* (Statistical Methodology), vol. 68, no. 1, pp. 49–67, 2006.
- [11] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 53–71, 2008.
- [12] D. Ruta and B. Gabrys, "An overview of classifier fusion methods," Computing and Information systems, vol. 7, no. 1, pp. 1–10, 2000.
- [13] Z. Liu, Q. Pan, J. Dezert, J. Han, and Y. He, "Classifier fusion with contextual reliability evaluation," *IEEE Transactions on Cybernetics*, vol. 48, no. 5, pp. 1605–1618, May 2018.
- [14] L. I. Kuncheva, J. C. Bezdek, and R. P. Duin, "Decision templates for multiple classifier fusion: an experimental comparison," *Pattern recognition*, vol. 34, no. 2, pp. 299–314, 2001.
- [15] N. J. Pizzi and W. Pedrycz, "Aggregating multiple classification results using fuzzy integration and stochastic feature selection," *International Journal of Approximate Reasoning*, vol. 51, no. 8, pp. 883–894, 2010.
- [16] G. J. Scott, R. A. Marcum, C. H. Davis, and T. W. Nivin, "Fusion of deep convolutional neural networks for land cover classification of high-resolution imagery," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1638–1642, 2017.
- [17] G. J. Scott, K. C. Hagan, R. A. Marcum, J. A. Hurt, D. T. Anderson, and C. H. Davis, "Enhanced fusion of deep neural networks for classification of benchmark high-resolution image data sets," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 9, pp. 1451–1455, 2018.
- [18] D. T. Anderson, G. J. Scott, M. A. Islam, B. Murray, and R. Marcum, "Fuzzy choquet integration of deep convolutional neural networks for remote sensing," in *Computational Intelligence for Pattern Recognition*. Springer, 2018, pp. 1–28.
- [19] J. M. Keller, D. B. Fogel, and D. Liu, Fundamentals of computational intelligence: neural networks, fuzzy systems, and evolutionary computation. John Wiley & Sons, 2016.
- [20] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," Fuzzy sets and systems, vol. 66, no. 1, pp. 1–13, 1994.

- [21] S. Rajurkar and N. K. Verma, "Developing deep fuzzy network with takagi sugeno fuzzy inference system," in *Fuzzy Systems (FUZZ-IEEE)*, 2017 IEEE International Conference on. IEEE, 2017, pp. 1–6.
- [22] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.
- [23] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international* conference on computer vision, 2015, pp. 1520–1528.
- [24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167, 2015.
- [25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
- [26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
- [27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.