Extending deep learning to new classes without retraining

Conference Paper · April 2020

DOI: 10.1117/12.2558134

CITATION
1

READS
217

9 authors, including:

Jeffrey Schulz
University of Missouri
3 PUBLICATIONS **1** CITATION
SEE PROFILE

Mihail Popescu
University of Missouri
219 PUBLICATIONS **3,413** CITATIONS
SEE PROFILE

Grant J. Scott
University of Missouri
56 PUBLICATIONS **1,017** CITATIONS
SEE PROFILE

Extending deep learning to new classes without retraining

Schulz, Jeffrey, Veal, Charlie, Buck, Andrew, Anderson, Derek, Keller, James, et al.

Jeffrey Schulz, Charlie Veal, Andrew Buck, Derek Anderson, James Keller, Mihail Popescu, Grant Scott, Dominic K. C. Ho, Timothy Wilkin, "Extending deep learning to new classes without retraining," Proc. SPIE 11418, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV, 1141803 (24 April 2020); doi: 10.1117/12.2558134

SPIE.

Event: SPIE Defense + Commercial Sensing, 2020, Online Only

Extending Deep Learning to New Classes without Retraining

Jeffrey Schulz[a], Charlie Veal[a], Andrew Buck[a], Derek Anderson[a], James Keller[a], Mihail Popescu[a], Grant Scott[a], Dominic K. C. Ho[a], and Timothy Wilkin[b]

[a]Department of Electrical Engineering and Computer Science, University of Missouri, Columbia MO, USA
[b]School of Information Technology, Deakin University, Geelong, Victoria, AU

ABSTRACT

The focus of this article is extending classifiers from N classes to N+1 classes without retraining for tasks like explosive hazard detection (EHD) and automatic target recognition (ATR). In recent years, deep learning has become state-of-the-art across domains. However, algorithms like convolutional neural networks (CNNs) suffer from the assumption of a closed-world model. That is, once a model is learned, a new class cannot usually be added without changes in the architecture and retraining. Herein, we put forth a way to extend a number of deep learning algorithms while keeping their features in a locked state; i.e., features are not retrained for the new N+1 class. Different feature transformations, metrics, and classifiers are explored to assess the degree to which a new sample belongs to one of the N classes and a decision rule is used for classification. Whereas this extends a deep learner, it does not tell us if a network with locked features has the potential to be extended. Therefore, we put forth a new method based on visually assessing cluster tendency to assess the degree to which a deep learner can be extended (or not). Lastly, while we are primarily focused on tasks like aerial EHD and ATR, experiments herein are for benchmark community data sets for sake of reproducible research.

Keywords: convolutional neural network, explosive hazard detection, EHD, automatic target recognition, ATR, possibilistic K nearest-neighbor, PKNN, clustering, clustering in ordered dissimilarity data, CLODD

1. INTRODUCTION

Our modern era of computational intelligence, machine learning, artificial intelligence, and related, is focused intensely on highly specialized domain/task specific learners. A well-known example is the convolutional neural network (CNN) and deep learning. While leaps in performance have been made via deep learning, a drawback is that they generally abide by a "closed world" assumption, meaning the network was trained to regress or predict a certain finite number of factors or classes that were present in the training data. However, what happens if a user desires to add a new class/object, e.g., new target in automatic target recognition (ATR)? Current practice is to retrain a network from scratch—which is generally an offline and resource expensive operation—or the feature extraction layers can be retained and the classification layers retrained (a type of transfer learning). The point is, we need a more elegant way to extend models like a CNN to class "N+1" without requiring extensive and expensive offline retraining. Furthermore, most CNNs utilize a normalization strategy like soft max. Whereas this often helps during training (and perhaps testing), it turns a network into a *one of N* classifier. The point it, machines should possess a sufficient way to say "I don't know what this is".

Herein, we focus on two challenges. The first is how to *efficiently* extend modern deep learning to class N+1 to support tasks like explosive hazard detection (EHD) and ATR—see Figure 1. To this end, we investigate different classifiers from computational intelligence, namely fuzzy set theoretic methods, versus the conventional CNN approach of using a multi layer perceptron (MLP). Specific classifiers explored herein include the fuzzy K nearest neighbor (FKNN) and the possibilistic K nearest neighbor (PKNN). We explore the performance of

Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV, edited by Steven S. Bishop, Jason C. Isaacs, Proc. of SPIE Vol. 11418, 1141803 · © 2020 SPIE
CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2558134

Table 1: Acronyms and Notation

KNN	K nearest neighbor
PKNN	Possibilistic KNN
CLODD	Clustering in ordered dissimilarity data
K	Number of neighbors for evaluation
$\hat{K}$	Number of neighbors to fit in PKNN
$\mathbf{x}_i \in X$	Train dataset of N samples and M dimension
$\mathbf{f}_i \in \Re^{\hat{M}}$	Neural feature encoding
$[D]_{ij} = D(i,j)$	Dissimilarity of $\mathbf{x}_i$ and $\mathbf{x}_j$

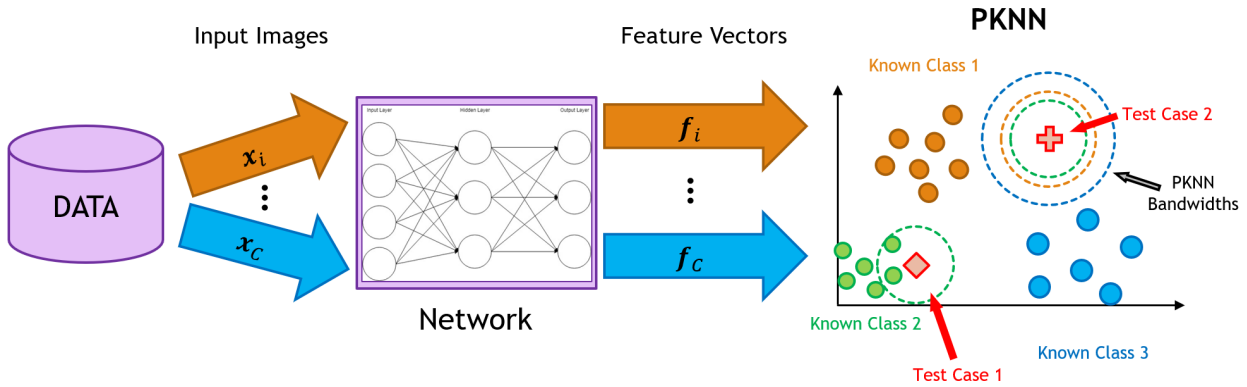


Figure 1: High-level illustration of how we extend a deep learner to class N+1. First, a network, e.g., CNN, is trained. Second, we strip off the classification layers (the MLP). As a result, each input is presented to a CNN for feature extraction. These features are then passed to a classifier like the KNN, FKNN, or PKNN. Based on the resultant typicalities, a sample is classified into a known class or the system reports "I don't know".

these classifiers and their ability to say "I don't know". To this end, we propose a way to learn the *bandwidth* (the mechanism by which we say "I don't know") in the PKNN. Next, we investigate the challenge of predicting our models ability to go to N+1—see Figure 2. The reality is, a CNN is generally at least two parts, feature extraction and classification. What features did the network learn? Instead of assuming that we can just add "N+1", we propose an approach based on cluster validity assessment to measure the degree, in $[0, 1]$, to which we can add class N+1 (or not).

The remainder of this article is organized as follows. First, in Section 2 we discuss features and feature extraction, in Section 3 we discuss our PKNN implementation and how to learn the PKNN from data, Section 4 discusses our method of assessing if a deep learning model can be extended to class N+1, and Section 5 discusses our experiments and results.

2. FEATURE EXTRACTION AND DIMENSIONALITY REDUCTION

To no surprise, artificial intelligence, machine learning, pattern recognition, etc., are only as good as the information (e.g., data and associated features) they are provided. For decades our research community has exerted significant effort in manually identifying "manual or hand crafted features". In signal, and specifically image, processing, this typically equates to information like color, texture, and shape. An array of features have been proposed to date, from Gabor filters to histogram of gradients, fractals, Harris, etc. However, the last decade has witnessed a shift from hand crafted features to "machine learned" features.

In the context of neural networks, specifically CNNs, the story is that they learn basic shape, color, and/or texture features in lower layers, deeper layers are associated with more complicated and compound shapes/objects, followed eventually by higher-level relationships and semantic information. Regardless of what is learned, the fact is that CNNs consist of layers of filters, which when presented with data provide a response field. Consider the popular Resnet51. If one removes the "classification layers" of Resnet51, then they obtain a final response

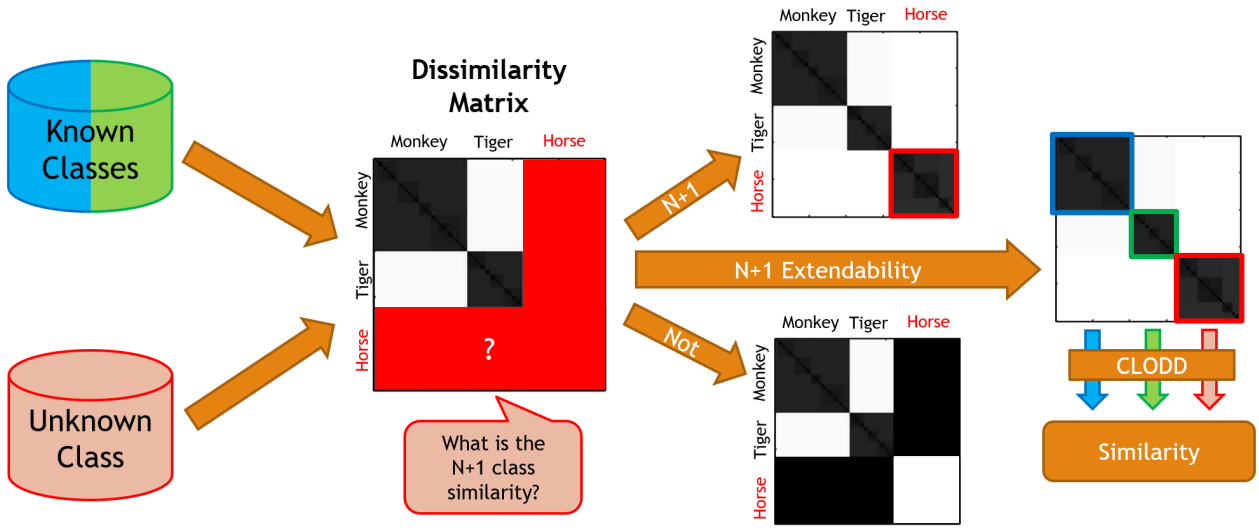


Figure 2: Illustration of our procedure for measuring the degree to which a model can be extended to class N+1. We start with known classes that the model has been trained to recognize. Next, a set of instances from class N+1 is provided. A dissimilarity matrix is constructed, which we can visually assess to determine how well the model can extend. Furthermore, aspects of the CLODD algorithm are used to formally extract a number, or the $[0, 1]$ degree to which the network is extendable.

field of $2,048$ *features*. Specifically, the standard Resnet51 has a $7x7$ response field per feature, which means that Resnet51 has a dimensionality of $2,048 * 7 * 7$. Herein, in our pursuit of extending our models to class $N + 1$, we investigate nearest neighbors versus an MLP classifier layer. This presents a dilemma as we have to compute, and use, distances between samples in high dimensional spaces. The reader can refer to[1] for a discourse on the challenges of assessing proximity in high dimensional spaces. Herein, in an effort to combat the curse of dimensionality, we explore a few reduction strategies to empirically assess if they help classification.

2.1 Reduction Strategy 1: Pooling

The first strategy that we explore is pooling. In the context of a CNN, pooling helps us combat a number of factors, from dimensionality reduction to combating scale and even noise. The output feature vector for Resnet51 is of size $2,048 * 7 * 7$. In an attempt to side step affine transformations that we are not concerned with, we pool within each feature map. Herein, we use max pooling, i.e., we take the largest response per feature (response field). A demonstration of max pooling across a $4x4$ response field is

$$\begin{matrix} 0.2 & 0.1 & 0 & 0.6 \\ 0.1 & 0.3 & 0.4 & 0.5 \\ 0.7 & 0.6 & 0.8 & 0.5 \\ 0.4 & 0.3 & 0.4 & 0.2 \end{matrix} \rightarrow 0.8.$$

Whereas we use max pooling, the reader can engage in a variety of aggregation operations for pooling, i.e., min, median, or more complicated and unique operators like the ordered weighted average[2] or fuzzy integral.[3,4] Max, a generalized union operator, is an appropriate fit for our task as it captures the idea of what was our strongest response per feature. In summary, for a CNN like Resnet51, pooling is a simple procedure that helps us reduce dimensionality from $2,048x7x7$ to $2,048$, at the expense of where in the spatial domain the feature occurred and how many instances of that feature were observed.

2.2 Reduction Strategy 2: Principal Component Analysis

A second, yet simple, dimensional reduction technique explored herein is principal component analysis (PCA). PCA is an unsupervised method that is a linear transformation whose basis vectors are identified on the premises

of maximizing variance. We perform an eigen decomposition, which results in D eigenvectors with corresponding eigenvalues for a D dimensional space. Herein, we select the fewest number of eigenvectors whose eigenvalue sum corresponds to more than %99 of the overall data variation. However, PCA is a linear transformation and there is no guarantee that patterns or clusters in the original high dimensional space are preserved in the reduced low dimensional space. For additional methods, the reader can refer to additional unsupervised methods like random projections or supervised methods like Fishers linear discriminate analysis.

3. CLASSIFICATION: POSSIBILISTIC KNN

Next, we discuss our strategy for taking neural features and realizing a classifier that can help extend to $N+1$ and say "I don't know". To this end, the PKNN improves the KNN by improving the semantics of the nearest neighbor memberships and enabling outlier detection. A membership value can be calculated from the KNN via

$$\mu^i_{knn}(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}_i(\mathbf{y}_k), \tag{1}$$

$$\mathbb{1}_i(\mathbf{y}_k) = \begin{cases} 1 & \text{if sample } \mathbf{y}_k \text{ belongs to class } i \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Clearly, $\mu^i_{knn}(x) \in [0,1]$ is the number of points in class i relative to all of the nearest neighbors (K). In,[5] Keller et al. proposed an extension to the KNN, the fuzzy KNN (FKNN),

$$\mu^i_{fknn}(\mathbf{x}) = \frac{\sum_{j=1}^{K} \tilde{\mu}^i(\mathbf{y}_j)\left(\frac{1}{||\mathbf{x}-\mathbf{y}_j||^{\frac{2}{m-1}}}\right)}{\sum_{j=1}^{K}\left(\frac{1}{||\mathbf{x}-\mathbf{y}_j||^{\frac{2}{m-1}}}\right)}, \tag{3}$$

which is a membership weighted and inverse distance ratio of the K nearest neighbors. For each training sample data point, a fuzzy membership is computed offline,

$$\tilde{\mu}^i(\mathbf{x}) = \begin{cases} 0.51 + (\frac{n_i}{K}) \times 0.49, & \text{if } \mathbf{x} \text{ belongs to class } i \\ (\frac{n_i}{K}) \times 0.49, & \text{otherwise,} \end{cases} \tag{4}$$

where $1 \leq n_i \leq K$ is the number of neighbors that belong to class i. Thus, a membership degree is computed per data point and per class relative to its local K nearest neighbors. If the sample point ($\mathbf{x}$) belongs to class i, then a *reward* of 0.51 is added to start, plus the ratio of K points in class i. Otherwise, the membership is restricted to $0 \leq \tilde{\mu}^i(\mathbf{x}) \leq 0.49$. Imagine a sample surrounded by data points from the opposing class. The assumption here is that our sample (that belongs to class i) is not *noise*. The resultant membership degree (i.e., $\tilde{\mu}^i(\mathbf{x})$) is 0.51, whereas it would have little-to-no voice in the KNN. In general, one expects the FKNN to be the most beneficial in overlapping regions. During test time, we calculate $\mu^i_{fknn}(\mathbf{x})$, which exploits these membership degrees. Consider a scenario in which a test sample has one retrieval from class i and the other nearest neighbors belong to a different class. In the KNN, we would not select class i. However, in the FKNN, depending on the proximity of $\mathbf{x}$ to the sample in class i, it is possible to generate $\mu^i_{fknn}(\mathbf{x}) = 1$ when $\mathbf{x}$ is sufficiently *close* to the ith sample. This is one story. There are clearly other FKNN benefits that are more complicated, e.g., related to memberships and relative neighbor distances.

In,[6] Frigui and et al. created the possibilistic KNN (PKNN), an extension of the FKNN and KNN,

$$\mu^i_{pknn_1}(\mathbf{x}) = \frac{1}{K}\sum_{j=1}^{K} \tilde{\mu}^i(\mathbf{y}_j) w^p(\mathbf{x}, \mathbf{y}_j), \tag{5}$$

where $\mu^i_{pknn_1}(\mathbf{x})$ is the typicality–e.g., degree of similarity–of a test sample to a known class. Equation 5 sums over K neighbors the product of the training data fuzzy memberships and the possibilistic score w^p (explained below). Herein, we also explore an unweighted version of the PKNN,

$$\mu^i_{pknn_2}(\mathbf{x}) = \frac{1}{K}\sum_{j=1}^{K} w^p(\mathbf{x}, \mathbf{y}_j). \tag{6}$$

The possibilistic score, $w^p(\cdot)$, is where the PKNN truly differs from the FKNN and KNN. The PKNN uses a *bandwidth factor*, η, to take into account the distance of the test case to the prototypes. If a data point is within the bandwidth, then the membership value is 1. Otherwise, the membership value decays, allowing points to have low-to-no membership. The possibilistic scoring is

$$w^p(\mathbf{x}, \mathbf{y}_j) = \frac{1}{1 + [\max(0, \|\mathbf{x} - \mathbf{y}_j\| - \eta)]^{2/(m-1)}}. \tag{7}$$

In,[6] Frigui et al. derived η from data. Specifically, Frigui let $\eta = \frac{\eta_1}{\eta_2}$, where η_1 is the mean μ of the distances of the K closest neighbors in the training data and η_2 is three times the standard deviation of the same set.

3.1 Bandwidth Parameter Optimization

Not all data and patterns are created equal, which poses problems for the η parameter presented in Equation 7. Whereas the parameter estimation method proposed by Frigui et al. led to improvement for their explosive hazard detection task, it does not necessarily generalize. In our experiments, datasets like ImageNet have relatively large standard deviations which result in tiny η values. This required us to scale Frigui's η to produce positive results. While this is useful, it increased the amount of hand-picking and fine tuning required to get a PKNN classifier running. To combat this problem, we propose to learn the bandwidths from data. To this end, we formulate a cost function, $J(w^p, \eta, \alpha)$, and a genetic algorithm is used for optimization as the equation does not have a nicely differential or analytical solution. The cost function explored herein is

$$J(w^p, \eta, \alpha) = (1 - \frac{w^p_{i,i}}{C}) + \frac{w^p_{i,j}}{C} + \alpha \sum_{i=1}^{C} |\eta_i|, \tag{8}$$

where w^p is the typicalities of our validation samples, $w^p_{i,i}$ is the sum of true-positive typicalities (where only the typicalities belonging to the correct class are totaled), and $w^p_{i,j}$ is the sum of false-positive typicalities (where only the typicalities belonging to incorrect classes are totaled). The cost function is rounded out with a summation factor, scaled by α (a user defined parameter), meant to punish larger bandwidths. This factor prevents bandwidths from ballooning farther than they need to be, increasing the chance for future $N+1$ classes to fall outside of η in feature space.

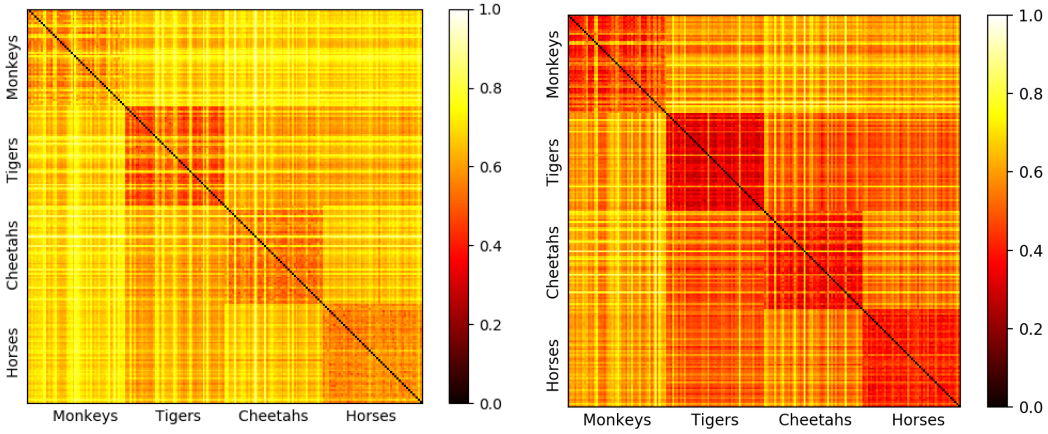
4. ASSESSING EXTENDING TO CLASS N+1 AND SAYING "I DON'T KNOW"

The goal of this section is two fold. First, we desire a visual way to see and understand the quality of different deep learning algorithms and models with respect to the N classes it was trained on and new unknown classes. Second, we desire a way to move this qualitative process into a quantitative procedure. The following section outlines these two approaches.

4.1 Qualitative: Visualizing Architectures, Models, and Features

First, we focus on qualitative assessment. If we, as humans, can determine separability between the features of known and unknown classes when displayed in certain data visualization strategies, then it might be good news for our deep learning models. In order to qualitatively determine if a model is able to separate the features of a new class from an existing class, we will use an ordered dissimilarity matrix (ODM). ODMs are matrices built from our data set by computing a distance function, e.g., l_p norm, between all possible pairings of image feature vectors. For sake of description, assume we have N classes. Furthermore, assume, without loss of generality that there is an equal number of samples per class. In the ideal case, all samples in a class are similar and have a low distance. Conversely, samples in a class are dissimilar to samples from other classes. As a result, when one "looks" at an ODM they will see "dark blocks" along the diagonal per class and "white rectangles" in the opposing classes. Whereas all distances are positive, in order to draw an ODM one can engage in a strategy like range compressing the ODM between the min and max value for visual display. A dissimilarity matrix (not ordered), like what we will use in this paper, is shown in Figure 3.

In the context of clustering (unsupervised learning), one typically uses a procedure like VAT or iVAT,[7] for ordering the samples and enhancing the visual structure of an ODM. Herein, we have class labels, we are doing



(a) No Dimensionality Reduction (b) Max-Pooling Reduction

Figure 3: ODM displaying 50 images per class of Monkeys, Tigers, Cheetahs, and Horses (the N+1 class). The displayed data is the feature activation from ResNet101 max-pooled to size 1x2048. For this class balanced example, the best case scenario is four dark diagonal squares with white in the off diagonals.

supervised learning. While we could reorder samples in each class via VAT/iVAT, one can regardless make out the structure and class separation in Figure 3. However, for larger sets of samples, we recommend running VAT per class and the enhancement step of iVAT on the permuted VAT ODM/image. As Figure 3 shows, our N+1 class, horses, has a distinguishable dark box, indicating similarity to itself and dissimilarity to the rest of the data set (aka other classes). As a result, it suggests that even though the network was not trained on horses, it learned visual features to recognize and distinguish horses. This is a positive indicator that the model can be extended, i.e., the network might not need to be retrained.

4.2 Quantitative: Numerical Value Indicating "Goodness"

In order to rank models according to their ability to extend to N+1, we need to establish a measure that somehow determines the separability of the N+1 class features to known class features. This takes the human out of the loop and it enables automating the ranking for large numbers of trained models. Herein, we leverage the measure component of the Clustering in Ordered Dissimilarity Data (CLODD)[8] algorithm, which is a method to automatically discover the number of clusters in a VAT/iVAT image. The equation for CLODD is

$$E_\alpha(U; D*) = \alpha E_{sq}(U; D^*) + (1-\alpha)E_{edge}(U; D^*); 0 \leq \alpha \leq 1, \tag{9}$$

where $E_\alpha(U; D*)$ is the weighted score between the "edginess" $E_{edge}(U; D^*)$ of the clusters and the "squaredness" $E_{sq}(U; D^*)$ of the clusters. The weight, a user defined parameter, of the "edginess" and "squaredness" factors are determined by the mixing factor α. The equation for squaredness in CLODD is

$$E_{sq}(U; D^*) = \left(\frac{\sum_{i=1}^{c} \sum_{s \in i, t \notin i} d^*_{st}}{\sum_{i=1}^{c} (n - n_i) n_i} \right) - \left(\frac{\sum_{i=1}^{c} \sum_{s,t \in i, s \neq t} d^*_{st}}{\sum_{i=1}^{c} (n_i^2 - n_i)} \right), \tag{10}$$

where the average dissimilarity within dark regions is subtracted from the average dissimilarity between dark and non-dark regions. In the first factor, dissimilarity between classes and all other classes ($d^*_{s \in i, t \notin i}$) is averaged. In the second factor, dissimilarity between classes and themselves ($d^*_{s,t \in i, s \neq t}$) is averaged. The equation for edginess in CLODD is

$$E_{edge}(U; D^*) = \frac{1}{c-1} \sum_{j=1}^{c-1} \frac{\sum_{i=m_{j-1}}^{m_j} \left| d^*_{i,m_j} - d^*_{i,m_j+1} \right| + \sum_{i=m_j+1}^{m_{j+1}} \left| d^*_{i,m_j} - d^*_{i,m_j+1} \right|}{n_j + n_{j+1}}, \tag{11}$$

where the dissimilarity between one cluster and the next is summed up over all clusters, averaged over the number of samples. Equations 10 and 11 are combined to create Equation 9, which is maximized in the CLODD algorithm to find good clusters.

Herein, we have a slightly different problem than CLODD. Specifically, we are not working with unlabeled data and we do not permute the dissimilarity matrix with respect to underlying cluster structure. Instead, the dissimilarity matrix is organized according to know class labels, which alters the semantics of edginess. As such, two procedures are explored herein.

In Method 1, only squaredness is calculated, not edginess, with respect to class N+1. The idea is to subtract the average of the class N+1 to not N+1 classes from the average of the class N+1 to class N+1 instances. However, we first normalize the dissimilarity matrix by subtracting its minimum and then dividing by the maximum. This normalization is performed for sake of interpretability, i.e., a value of one is best and negative one is the worst possible outcome. In Method 2, we compute Equation 10, just squardness on the entire matrix. The difference between Method 1 and Method 2 is, Method 1 says "how well can we separate class N+1 from the other classes", and Method 2 says that plut "how well does the not N+1 classes separate from one another". It might be important to consider both, as the goal would be an ODM with all dark blocks across the diagonal, indicating that class N+1 can be detected and discriminated, but it does not come at the expense of any of the existing classes. This nuance is subtle and elaborated on via example in the results section.

5. EXPERIMENTS AND RESULTS

In this section, we investigate three questions: what is our classification accuracy relative to different models and projections; does optimal bandwidth parameter selection lead to performance gain; and is it possible to rank order approaches based on our predictive metric. While our primarily goal is detection relative to aerial EHD and ATR, our experiments are performed on benchmark community data sets for sake of reproducible research.

5.1 PKNN-Based Classification

In this subsection, we focus on two data sets with varying levels of complexity, noise, and class similarity.

5.1.1 Experiment 1: Visually Challenging Classes and Imperfect Dataset

In Experiment 1 (see Figure 4), we demonstrate the classification accuracy of the PKNN on a three class dataset; Oranges, Bananas, and Donuts. Oranges and Bananas are from ImageNet but Donuts (the "class N+1" here) are from the Food101 dataset. The Food101 dataset has noisy labels, at an estimated level of twenty percent. The model under test is ResNet101, pretrained on ImageNet. Since the model is pretrained on ImageNet and ImageNet does not include donuts, presumably our model has not seen class N+1 and it may not be equipped with features to recognize donuts. The weight factor α, from the cost function 8 is 0.001 and the typicality threshold for classficiation is 0.34. Optimized bounds are found using a DEAP genetic algorithm in 10 generations with 100 individuals, a 0.5 cross-over rate, and 0.2 mutation rate. In the case of PCA, the data is reduced to 1x200 per sample. When reducing the max-pooled data with PCA, the data is reduced to 1x180.

Experiment 1 tells the following story. First, when no dimensionality reduction is performed, donuts worsen banana classification. For sake of page count, we do not focus on why these miss-classifications occur—e.g., shape, background versus foreground object features, etc.—the reader can refer to methods like GradCAM or convolution matrix transpose (e.g., "deconvolution") for visual explainable AI if desired. PCA is the worst performer and the best solution is max pooling with PCA. That is, the best answer is to reduce dimensionality after performing an optimistic pooling step per feature map. The reader should recall that Food101 has noisy labels and complex backgrounds (aka, image content that is not our class of interest). This is a major reason for selecting these datasets versus a dataset like MNIST; which consists of foreground digits with no background complexity. Furthermore, we determined that it was important to start with a relatively hard visual task, e.g., distinguishing foods, versus something simpler like different animals (Experiment 2).

(a) No Dimensional Reduction

(b) Max-Pooling

(c) PCA

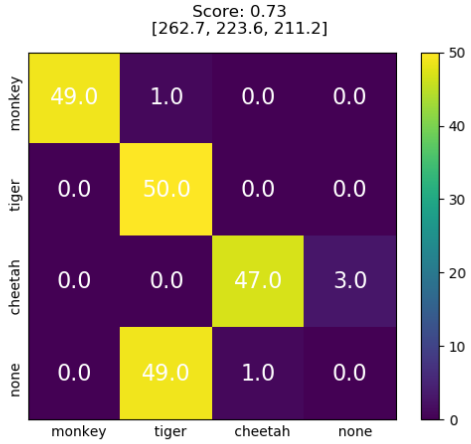
(d) Max-Pooling & PCA

Figure 4: Confusion matrices of test data after bandwidth parameter estimation using the raw (high dimensional) data, max-pooled, PCA, and max-pooled data reduced with PCA.

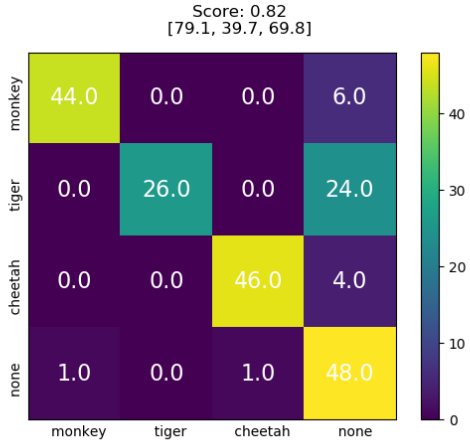
5.1.2 Experiment 2: More Relevant, Albeit Simpler Dataset

In Experiment 2 (Figure 5), we demonstrate the PKNN on a four animal class dataset; Monkeys, Tigers, Cheetahs, and Horses. Monkeys, Tigers, and Cheetahs are from ImageNet, whereas Horses are not, their reference imagery were handpicked by us. The model under test is ResNet101, pretrained on ImageNet. Since the model is pretrained on ImageNet and ImageNet does not include horses, the question is, did our model learn features that can help detect and distinguish horses. The weight factor α, from the cost function 8 is 0.001 and the typicality threshold for classficiation is 0.34. Optimized bounds are found using a DEAP genetic algorithm in 10 generations with 100 individuals, 0.5 cross-over rate, and 0.2 mutation rate. When reduced via PCA, the data is reduced to 1x300 per sample and the max-pooled data with PCA is reduced to 1x264.

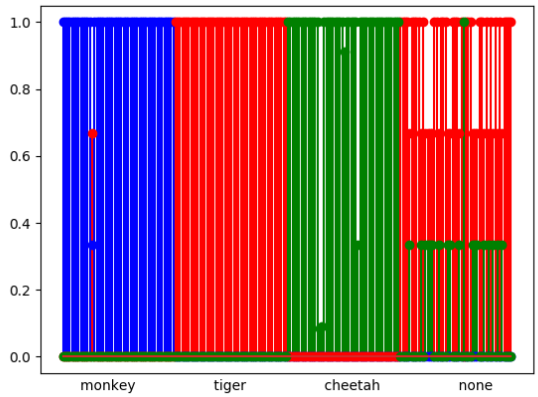
A limitation of only reporting classification rates and confusion matrices is that the reader cannot see the typicality degrees. For example, a confusion matrix is built with respect to which class has the highest typicality. In Figure 5, we show the classification rates, confusion matrices, and stem plots of the typicalities. The PCA stem plot paints a picture where the machine is almost always certain about the known classes, but class N+1 has many high typicalities. On the other hand, max-pooling shows varying confidences in the class examples



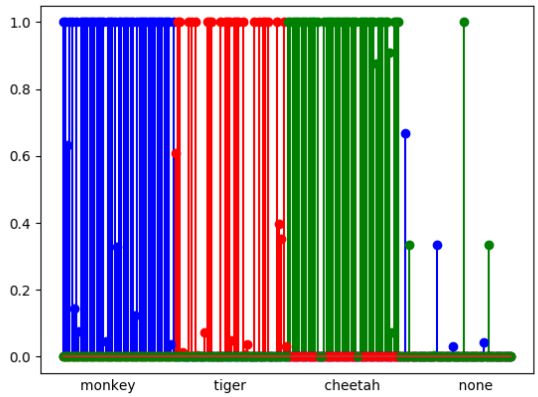
(a) PCA



(b) Max-Pooling



(c) Stem Plot - PCA



(d) Stem Plot - Max-Pooling

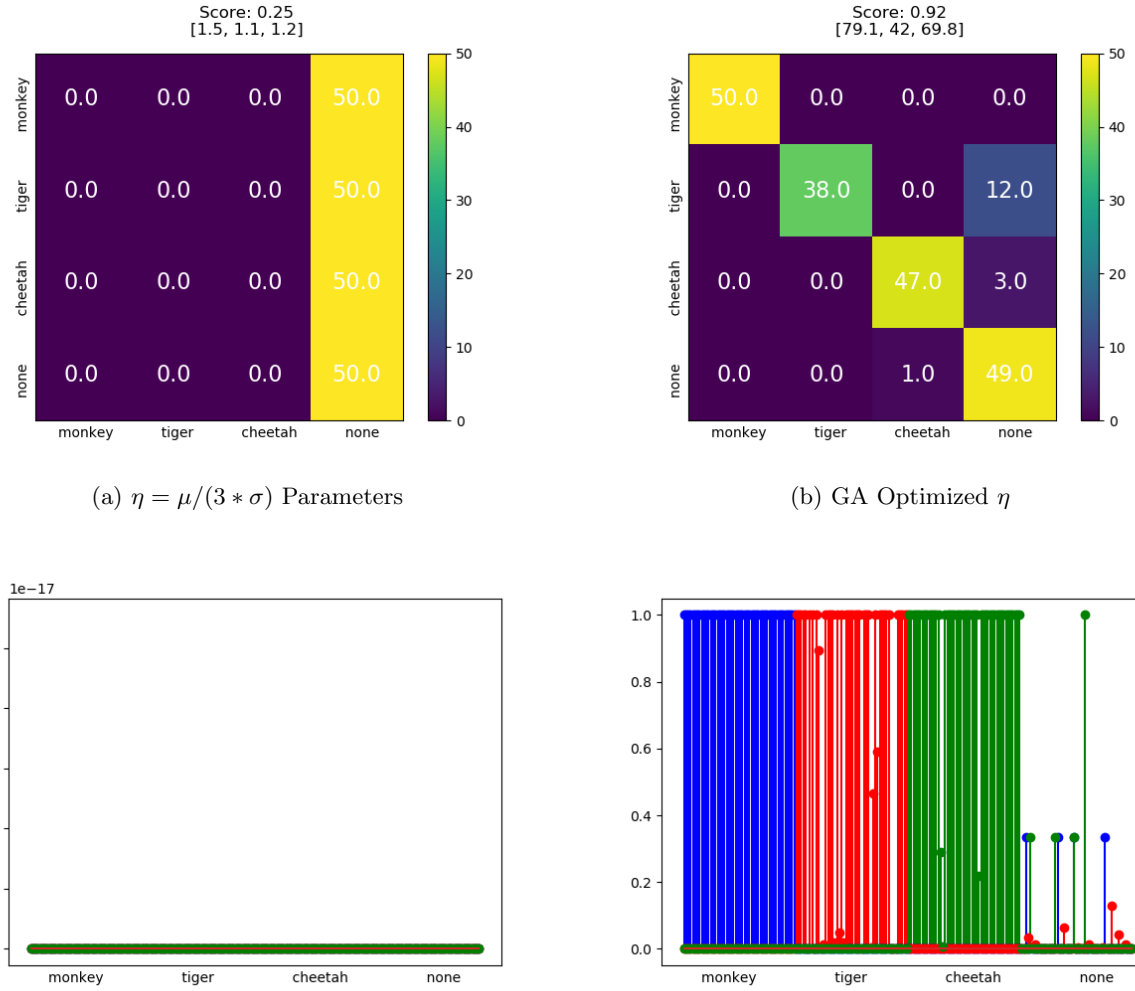
Figure 5: Confusion matrices and corresponding typicality stem plots for test data after bandwidth estimation.

with little-to-no typicalities in the class N+1. The reader can see that max-pooling reduction led to an accuracy of 82%, while PCA led to a drop of 9%.

Furthermore, we would like to stress that just because the model does not have a class does not mean that a model has not seen, and possibly built features for class N+1. For example, two unrelated objects can share features and learning features across classes can lead to discriminatory potential. Furthermore, even though ImageNet does not have a class for Horses, it does have a synset for "horse-cart" and other horse-drawn vehicles. We discovered this after identifying and running this experiment. Hence, our success with respect to this experiment—i.e., our ability to add and discriminate class N+1—could be due to the fact that a model pretrained on ImageNet learned/remembers features for horses, even though the specific class is not an option for classification in a traditional CNN pretrained on ImageNet. It is an interesting tidbit and something that the reader should be aware of.

5.1.3 Experiment 3: Bandwidth Optimization

In Experiment 3, we demonstrate that it is important to individually optimize bandwidths for each class as the Frigui et al.[6] method of calculating η based on class statistics does not hold for every dataset. The results of the originally proposed η are shown in Figure 6, with respect to the same setup as outlined in Experiment 2, except the data was max-pooled then reduced with PCA to 1x264.



(a) $\eta = \mu/(3 * \sigma)$ Parameters

(b) GA Optimized η

(c) Stem Plot: $\eta = \mu/(3 * \sigma)$

(d) Stem Plot: GA Optimized η

Figure 6: Visualization of the confusion matrices and typicality stem plots for the two bandwidth estimation procedures. Figures (a) and (c) show that the data-derived bandwidths are too small, resulting in total classification of "none". The optimized bandwidths for this dataset are clearly higher, as shown in (b) and (d).

Clearly, the bandwidths calculated from class statistics are too small. While they lead to perfect classification for class N+1, they are so tiny that they do not generate any high typicalities for the known N classes. While this selection scheme worked for the explosive hazard detection problem and set of features that Frigui et al. investigated, it does not hold across datasets. This is not alarming as we did expect the optimized bandwidths to perform better, as it has the ability to adapt to the underlying data/needs.

We want to make sure that the following is clear. The bandwidth optimization achieves a score of 92%, at the expense of mistaking some tigers as horses. While horses are almost perfectly classified, same with cheetahs

and monkeys, the neural network has not learn enough features to properly distinguish all of its classes. We can optimize the bandwidth parameters all we like, but we cannot overcome this limitation with the features.

5.2 Predicting a Models Ability to Extend to Class N+1

In this subsection we switch gears and we explore both qualitative and quantitative ways to assess if its possible to extend a model to class N+1. The above subsection approached this challenge with respect to the PKNN and classification accuracy. The aim of this subsection is to weaken our assumptions. We desire to determine if it is possible to take an ODM and directly predict a networks extension potential.

5.2.1 Experiment 4: Dimensionality Reduction Technique Assessment

In Experiment 4, we use squaredness (Equation 10) on the full ODM (Method 1) and only the N+1 rows (Method 2) to assess the different dimensionality reduction techniques explored herein. That is, we desire to observe if its advantageous to retain the full set of original features or their reduced and more efficient counterparts; the latter being our intuition. For this experiment, we use features from ResNet101 on the animal dataset from Experiment 2. The results for both measures are reported in Table 2 and Figure 7.

Table 2: Dimensionality reduction techniques on ResNet101 features.

Model	Squaredness, Full Matrix	Squaredness, Sub Matrix
No reduction	0.1106	0.1523
Max-pooling	0.1652	0.4253
PCA	0.2889	0.2344
Max-pooling & PCA	0.2257	0.6732

Figure 7 and Table 2 tell the following story. Overall, max-pooling with PCA is the best. However, PCA achieves a better score with respect to Method 1, Equation 10 for just class N+1. That is, PCA alone does the best job rejecting class N+1 samples. However, it does the such at the expense of the other N classes. As the overall squaredness tells us, max-pooling does a better job discriminating between the N classes and with class N+1. We included this example to illustrate the fact that only listening to rejecting unknown samples is not enough, it cannot come at the expense of the N classes.

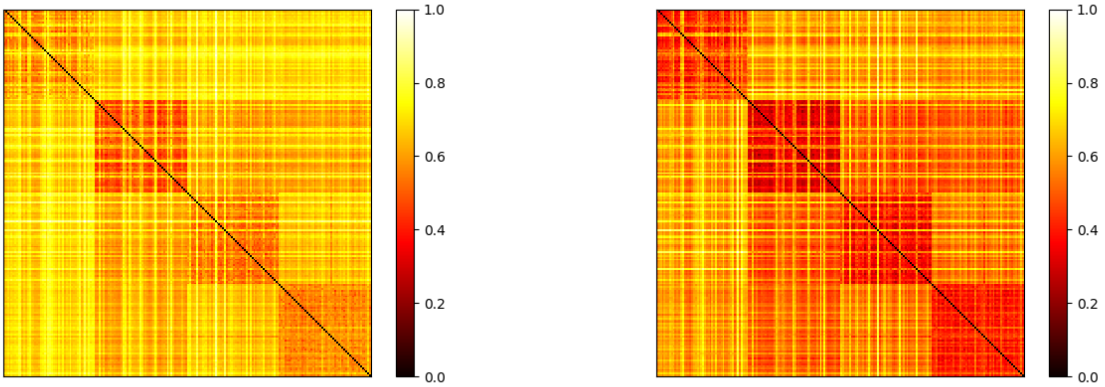
5.2.2 Experiment 5: Assessing Different Architectures and Models

In Experiment 5, we use our squaredness measure to compare and rank various models with respect to their ability to extend to class N+1. For this experiment, we max-pool the features coming out of the network on the animal dataset from Experiment 2. The results for each of the models are reported in Table 3 and Figure 8.

Table 3: Experiment 5 results.

Model	Squaredness, Full Matrix	Squaredness, Sub Matrix	Ranking	Num Features
ResNet101	0.1652	0.2565	1	2048
AlexNet	0.0992	0.0733	4	256
VGG19	0.1237	0.1511	3	512
DenseNet201	0.1333	0.1992	2	1920

Table 3 and Figure 8 tell the following story. First, the measure scores align with how we visually would rank the four architectures. Namely, ResNet101 was best, followed by DenseNet201, VGG19, then AlexNet. While this is reinforcing, there is a trend. Namely, the rank ordering of our models align with the number of features in the respective architectures. This might lead one to believe, in general, that the more features the better. However, we are not able to deduce such a conclusion based on such a simple basis; set of experiments. What Table 3 really highlights is the fact that this experiment is apples-2-oranges. That is, each model has a different number of features and as such they are challenging to compare. Again, our results are nice in the respect that they help support the validity of our qualitative and quantitative processes, however the reader would benefit from using the proposed measure across a wider range of models or perhaps apples-2-apples experiments, e.g., like architectures trained on different data, different initializations, etc. All we can logically extract from Experiment



(a) None (b) Max-pooling

(c) PCA (d) Max-pooling & PCA

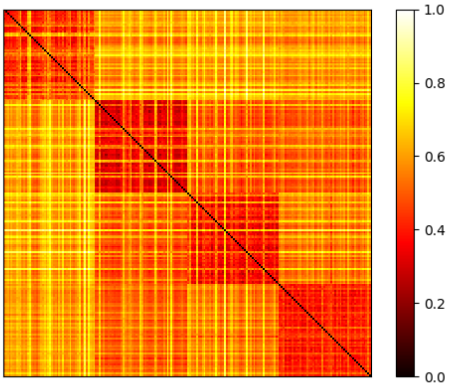
Figure 7: Dissimilarity matrices (normalized to $[0, 1]$) for Experiment 4.

5 is that our measure lines up with what a human would assess and that more features, up to some limiting or diminishing point, possibly result in a richer visual vocabulary that help with extending to class N+1.

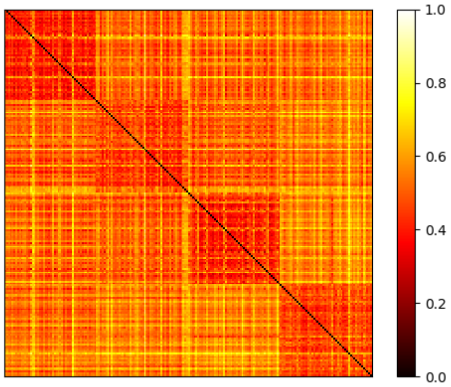
6. SUMMARY AND FUTURE WORK

Herein, we explore how to extend artificial neural networks to class N+1 (aka a new class that the network has not been trained on). To this end, we investigate different dimensionality reduction methods to remediate the impact of undesired affine transformations and the curse of dimensionality relative to the proposed possibilistic k nearest neighbor classifier (PKNN). As the PKNN depends on bandwidth parameters, we optimize them using a genetic algorithm. We couple these methods with the generation of ordered dissimilarity matrices and automatic scoring based on the notion of squaredness in CLODD. Our experiments show that the combination of max pooling and PCA lead to the best classification accuracy, typicalities, and ordered dissimilarity matrices. All of the above results were observed on community benchmark datasets for sake of reproducible research versus some underlying EHD or ATR dataset that cannot be shared. The particular experiments were selected to highlight classes of varying visual complexity.

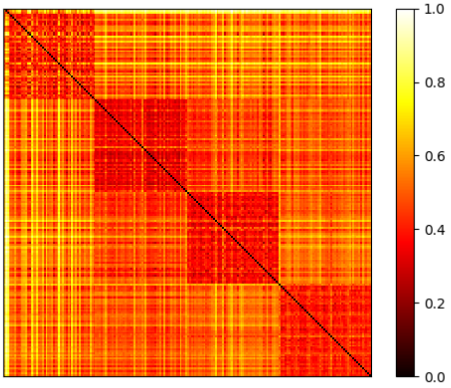
In future work, we will investigate the following. First, we will explore how to extend the notion of edginess



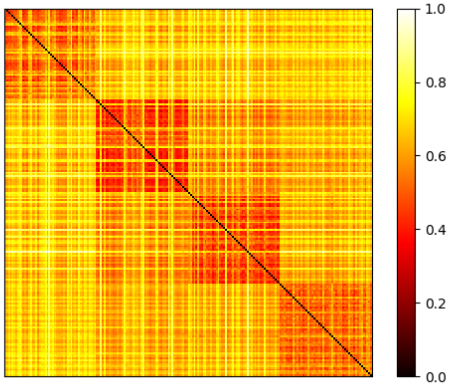
(a) ResNet101



(b) AlexNet



(c) VGG19



(d) DenseNet201

Figure 8: Dissimilarity matrices of max-pooled feature activations from each CNN, normalized to $[0, 1]$.

in a supervised context. Second, while the PKNN is useful for rejecting outliers, the mechanism (equation) needs improvement, beyond parameter (bandwidth) optimization. Next, our procedures, e.g., the PKNN and bandwidth selection, are learned independent of the neural network, which is merely performing feature extraction. Ideally, these would be learned in conjunction—parallel or simultaneously—with one another. From an experimental standpoint, a deeper and more thorough analysis is needed across existing architectures and models. Last, this article focuses on detection. A next step will be assessing how to extend the proposed ideas to detection and localization. In summary, we are excited about the preliminary results but more work is needed before a robust real-time solution is in hand.

ACKNOWLEDGMENTS

This work is partially funded by the Army Research Office (ARO) grants numbered W911NF-18-1-0153 and W911NF-19-1-0181 to support the U.S. Army RDECOM CERDEC NVESD.

REFERENCES

[1] Charu C. Aggarwal, A. H. and Keim, D. A., "On the surprising behavior of distance metrics in high dimensional space," (2001).

[2] Pagola, M., Forcen, J. I., Barrenechea, E., Lopez-Molina, C., and Bustince, H., "Use of owa operators for feature aggregation in image classification," in [*2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*], 1–6 (July 2017).

[3] Dias, C., Bueno, J., Borges, E., Botelho, S., Dimuro, G., Giancarlo, L., Fernandez, J., Sola, H., and Drews-Jr, P., [*Using the Choquet Integral in the Pooling Layer in Deep Learning Networks*], 144–154 (07 2018).

[4] Price, S. R., Price, S. R., and Anderson, D. T., "Introducing fuzzy layers for deep learning," in [*2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*], 1–6 (June 2019).

[5] James M. Keller, Michael R. Gray, J. A. G., "A fuzzy k-nearest neighbor algorithm," (1985).

[6] Hichem Frigui, P. G., "Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic k-nearest neighbor classifier," (2008).

[7] Wang, L., Nguyen, U. T. V., Bezdek, J. C., Leckie, C., and Ramamohanarao, K., "ivat and avat: Enhanced visual analysis for cluster tendency assessment," in [*PAKDD*], (2010).

[8] Timothy C. Havens, James C. Bezdek, J. M. K. M. P., "Clustering in ordered dissimilarity data," (2009).