# Omni-Modeler: Rapid Adaptive Visual Recognition with Dynamic Learning

Michael Karnes and Alper Yilmaz

Photogrammetric Computer Vision Lab.
The Ohio State University
Columbus, Ohio 43210

***Abstract***

*Deep neural network (DNN) image classification has grown rapidly as a general pattern detection tool for an extremely diverse set of applications; yet dataset accessibility remains a major limiting factor for many applications. This paper presents a novel dynamic learning approach to leverage pretrained knowledge to novel image spaces in the effort to extend the algorithm knowledge domain and reduce dataset collection requirements. The proposed Omni-Modeler generates a dynamic knowledge set by reshaping known concepts to create dynamic representation models of unknown concepts. The Omni-Modeler embeds images with a pretrained DNN and formulates compressed language encoder. The language encoded feature space is then used to rapidly generate a dynamic dictionary of concept appearance models. The results of this study demonstrate the Omni-Modeler capability to rapidly adapt across a range of image types enabling the usage of dynamically learning image classification with limited data availability.*

***Keywords***

*Dynamic Learning, Few-shot Learning, Generalized Visual Classification*

## 1. Introduction

Visual classification has an extremely wide area of application. It is now used across almost every industry from inventory management to plant identification to medical diagnostics to manufacturing quality control to worker safety to UAS visual navigation [1, 2, 3, 4, 5, 6, 7, 8, 9]. Cameras are becoming cheaper, their sensors are becoming more complex, and wireless connection more widespread all generating a growing visual information flow. Deep neural network (DNN) visual classification is an effective tool for handling this river of information.

Each of these applications requires a a large training dataset that covers the expected appearance ranges seen while in operation. The more complex the scenes, the larger the necessary dataset, an often underestimated caveat to these state-of-the-art visual classification algorithms.

Further more, these training requirements limit implementation in real-time dynamic learning tasks. A rapid visual classification system with low dataset requirements can be used for variety of tasks such as dataset annotation, customized visual classification, online learning, tracking, re-detection, and in data environments with changing concept appearances.

Few-shot learning (FSL) provides a more adaptable framework focusing on reducing training sample sizes [10]. This has recently evolved to meta-learning where the task transforms into learning how to learn [11]. Further development of few-shot learning led to the addition of a mutable memory component known as dynamic learning. The Omni-Modeler

DOI: 10.5121/sipij.2023.14501 

is a simple, efficient, and generalized few-shot visual classification framework that introduces a novel language encoding methodology. The Omni-Modeler language generation provides a highly flexible approach for distilling the large latent feature space of a DNN into a compressed discriminative description. The Omni-Modeler differential advantage lays in its simplicity in implementation, its generalizability to any DNN, and its flexible image-to-description encoding.

The objective of the Omni-Modeler is to ease the implementation of deep visual recognition in new areas of application through a generalized, understandable, and dynamic framework. This work specifically provides the following contributions:

- The novel Omni-Modeler architecture which:
  - Reduces dataset annotation requirements
  - Extends DNN operational domains to novel spaces
  - Enables a dynamically adaptable knowledge domain
- A user interface to aid in the implementation
- Results demonstrating the Omni-Modeler performance

## 2. Related Work

The Omni-Modeler is an agnostic few-shot learning (FSL) visual classification algorithm that can dynamically adapt to variations in visual input. The focus of this paper is to demonstrate its ability to classify across concept domains with minimal training data. The remainder of this section will cover general visual classification, FSL image classification, and dynamic learning.

Classification is the prediction of an object's class based on its attributes compared to a known model. DNN based visual classification embeds the image space into a latent feature space which is then followed by a class separation task projecting the latent feature space into a predictive metric. With traditional DNN classification, the network is trained through an iterative regressive process. Batches of training images are passed through the network and the network parameters are updated according to a loss function; first attributed to LeNet in 1989 [12]. This is known as an end-to-end classification algorithm and is highly demonstrated as an effective approach for tasks with sufficient training data and static concepts.

FSL takes a different approach of generating object appearance models with a minimal number of shots (a.k.a. sample reference images). This is known as metric learning where query images are classified by a similarity score to a set of reference examples. There is a long history of FSL using DNN learned features. For example, the Siamese network architecture [13] and the Matching network [14] and the Prototypical-Net [15]. The current trend in FSL algorithms follow a line of transforming pretrained DNN embedded features using a series of linear transforms, DNN classifiers, and iterative solvers [16, 17, 18]. The current state-of-the-art FSL algorithm, PT+MAP+SF+SOT, combines power transform (PT) mapping with discrete cosine transform spatial frequency (SF) features and adds Self-Optimal-Transport Feature Transform (SOT) achieving 89.94% for a 5-way 1-shot on the CIFAR dataset [19].

FSL can also be viewed from a perspective of domain adaptation or meta-learning [10, 11]. Domain adaptation looks to map one latent feature space to another, calculating an optimal transform. With DNN, this can be in the form of transfer learning and fine tuning, both retraining a pretrained network on a new dataset using traditional back propagation. Meta-learning is a more recent approach to the task formulating the problem into a learn-to-learn training, focusing on optimizing the feature embedding space. Dynamic learning,

an even more recent approach, extends FSL learning into adaptable dictionary learning enabling management of the classifier knowledge domain [20].

The Omni-Modeler takes the dynamic learning approach a step closer to practical application through its novel encoding process and concept dictionary learning that leverage the widely available general visual classification networks trained on ImageNet. This gives a significant advantage by removing the need for updating any network parameters through back propagation and enables the knowledge domain to be updated without the need to recalculate the feature transform. The encoding process generates a rich compressed feature space adequate for rapidly calculated class descriptors without back propagation that can be periodically updated as new imagery is obtained. The presented Omni-Modeler application simplifies implementation of dynamic learning for a wide set of users and their particular classification needs.

## 3. Approach

The Omni-Modeler was developed to capture transferable patterns in the DNN latent feature space that are locally learned in a dictionary of bag-of-words representation models[21, 22]. The transference of knowledge occurs through a series of linear transforms that select the most distinctive features for the given task. The overview of this process is diagrammed in Figure 1 with the following sections presenting the details behind this process.

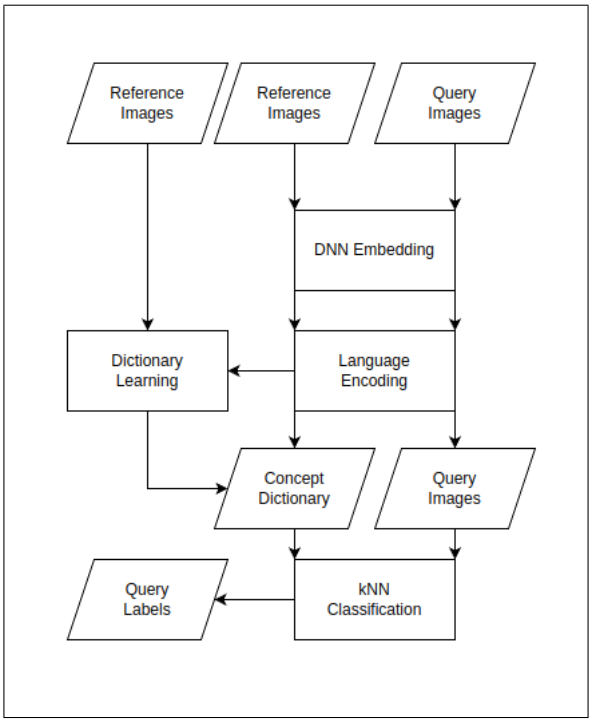


Figure 1: This algorithm overview illustrates the flow of the Omni-Modeler classification process. First a series of references images are DNN embedded, encoded using that calculated language, and then stored in concept dictionary. The query images are then embedded and encoded in the same manner. The query images are then classified using concept dictionary and k-Nearest Neighbors.

### 3.1. Problem Formulation

The objective of FSL visual classification is to generate the function that best predicts the class, $y_i \in 1, \ldots, C$, of a query sample image, $x_i \in R^{d_{img}}$, represented by the function:

$$f(x_i) = y_i \tag{1}$$

The function $f(x_i)$ is trained from a sub-sample of $k$ images random selected from a dataset of known images $D = (x_i, y_i)_{i=1}^{N}$, known as the support or reference set $S^\tau$. The best FSL function is defined by $E_\tau[\prod_{Q^\tau} p(y_i^*|f(x_i^*, S^\tau)]$.

### 3.2. Language Calculation

DNN visual classification trains to generate feature extracting filters corresponding to the most discriminative features between classes in a training set with an output of a class probability vector.

$$p(y^* = c|x^*) = \Phi(x^*) \tag{2}$$

Class probabilities can be viewed as a Gaussian mixture model (GMM) of class attributes in the latent feature manifold [23]. Extending this view, the images being passed through the DNN become instances of attributes produced by a GMM source. The Central Limit Theorem states that linear combinations of Gaussian distribution generates a Gaussian distribution. Even more generally it has been shown that the partial sum of randomly indexed m-dependent random variables tends toward Gaussian distributions as well [24]. Therefore, the DNN embedded latent features can be viewed as a GMM produced from random distribution of visual attributes with a class probability defined by:

$$p(y^* = c|x^*) = \frac{\pi_c \mathcal{N}_c(\mu_c, \Sigma_c^\tau)}{\Sigma_{c'} \pi_c' \mathcal{N}(\mu_{c'}, \Sigma_c^\tau)} \tag{3}$$

where $p(y^* = c|x^*)$ is the probability of $x^*$ belonging to class $c$ given the class distribution $\pi_c \mathcal{N}_c(\mu_c, \Sigma_c^\tau)$ over the sum of all class distribution $\Sigma_{c'} \pi_c' \mathcal{N}(\mu_{c'}, \Sigma_c^\tau)$.

The Omni-Modeler uses a combination linear transforms on the GMM of the latent feature manifold generating a local language of features for describing the novel image space from the embedded latent space of the DNN. This local language consists of letters, words, and sentences. The first level of transform on the embedded latent space forms the letters. Combinations of the transformed letters form the words. Each image is then translated into a sentence by concatenating the encoded words into a description vector.

The independent component analysis (ICA) is used to calculate the letters of the given dataset independent of class labels. This transforms the manifold to extract the most pronounced factors seen across the novel image space. Reducing the transformed space to only the most informative dimensions distills the GMM signal into persistent primitive patterns seen across the dataset.

The vocabulary of the feature space is organized into word structures using k-means clustering. This process condenses the common groupings of letters into semantically relevant concepts. When translating an image, each given region of the image space is given a description according to its Mahalanobis distance from the calculated words.

This feature encoding process is formally described in the equations below, where $\Phi$ is the embedding DNN and $a_i$ is the resulting embedded feature:

$$\Phi(x_i) = a_i. \tag{4}$$

The embedded feature is then encoded into letter space, $b_i$, the with the ICA transform $P$:

$$Pa_i^T = b_i. \tag{5}$$

.

Next the letter feature space is compared to each word in the vocabulary $V_j$ using the Mahalanobis distance resulting in the vocabulary description $r_{i,j}$:

$$M(b_i^*, \mu_{V_j}) = \frac{1}{2}(b_i^* - \mu_{V_j})^T \Sigma_{V_j}^{-1}(b_i^* - \mu_{V_j}) = r_{i,j}. \tag{6}$$

Repeating for each word in the vocabulary produces the word descriptor of the image space $r_i$:

$$distance(b_i^T, V) = r_i. \tag{7}$$

In practice, the DNN latent space embeds a grid of image tiles across the image space creating an embedded image (a.k.a. a class activation map). The word descriptions are concatenated across this grid of tiles to transform the encoded image space into sentence descriptor vector:

$$concatenate(r_i, x, y) = r_i. \tag{8}$$

The full language encoding process is visualized in the Figure 2.

### 3.3. Dictionary Generation

The conception dictionary is generated from a set of $k$ reference samples randomly selected from a selected dataset. Each of the selected images is encoded with the learned language into the image description vector $r_i$. The concept dictionary is stored as the list of learned descriptions from the reference set and the list of class labels for each description. It is this dictionary structuring that enables dynamic concept model updates with the addition or subtraction of concept references.

### 3.4. Training

The Omni-Modeler has three levels of training, shown in Figure 3. Please note that no DNN training was required for this application. The Omni-Modeler uses the off-the-shelf pre-trained DNN readily available from the PyTorch library. The second level of training is the language calculation. This training is completed using randomly selected, unlabeled images from the desired image space. The third level of training is the dictionary generation. This requires $k$ annotated samples from each class.

## 4. Implementation

The Omni-Modeler user interfaces (UI), shown in Figures 4 and 5, simplifies the process of implementing the application for a given dataset. Initially selecting the appropriate hyper-parameter configuration for a new image space and classification task can be a cumbersome process. The number of available DNN and the number of layers to each DNN creates an extremely large search space.

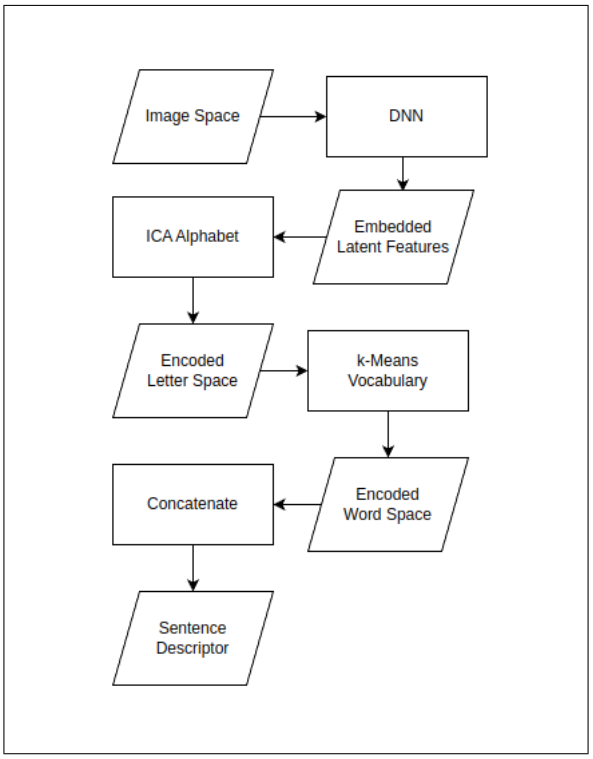


Figure 2: This diagram shows the encoding process used by the Omni-Modeler. First the image space is transformed into the embedded feature space by pulling from the latent feature space of the DNN. Then the embedded features are transformed into the letters space using the learned ICA alphabet. Next the letter space is transformed into the words space using the learned k-means vocabulary. Finally, the encoding process is completed by concatenating the words space into a full sentence descriptor.

The UI is split into screens focusing on the language generation, shown in Figure 4, and the dictionary generation, shown in Figure 5. The Language UI allows users to select the dataset, the network, the network layer, the alphabet length, and vocabulary size to be considered. The resulting embedded and encoded spaces are visualized wit tSNE plots allowing the user to asses the separability of classes for selected configuration. Once the language configuration is selected, the user then moves to the Dictionary UI. On this screen the user can explore the effects of the number of reference samples and the number of $k$ neighbors used in the classification process. The embedded and encoded samples are again visualized using tSNE. The classification performance is visualized with the confusion matrix and quantified with accuracy.

## 5. Experiments

### 5.1. Overview

The evaluation of the Omni-Modeler was designed to mimic the practical application of the algorithm. In this intended use scenario, the user is attempting to rapidly build a classification algorithm for their novel image space as their available training data grows. For example, a researcher in the medical field is collecting a set of tissue samples they would like to classify for a nuanced determination of disease tissue states.

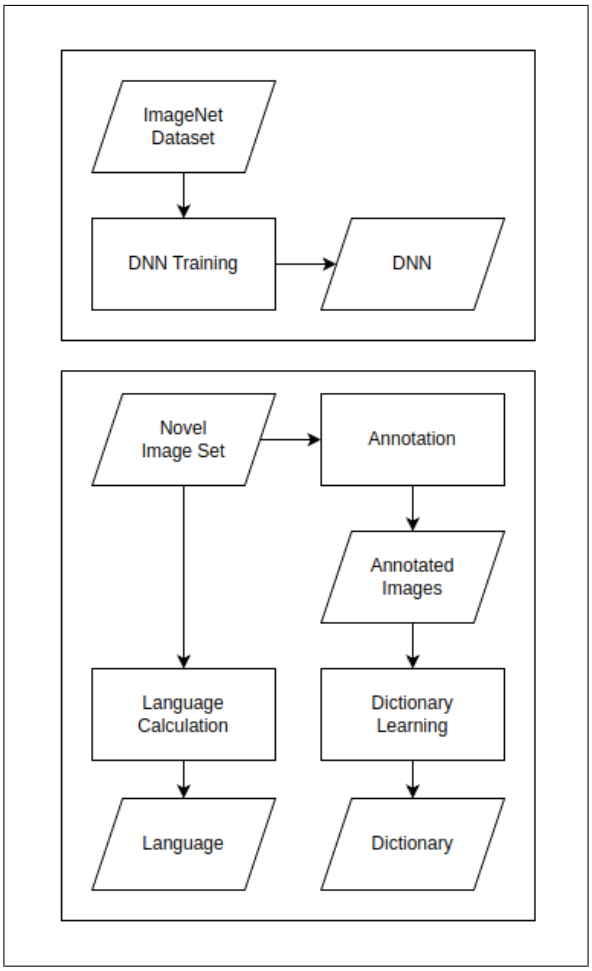


Figure 3: The training process occurs in three steps. First is the offline DNN training. Please note that the Omni-Modeler utilizes off-the-shelf DNN trained on the ImageNet dataset and readily available in the PyTorch library. The Omni-Modeler then calculates the language from a sampling of unlabeled images from the novel image set. The Omni-Modeler dictionary is the only portion that requires labeled images for training.

## 5.2. Datasets

A variety of datasets were gathered to evaluate the Omni-Modeler performance across range of image types. The consider datasets include character recognition MNIST [25], object classification CIFAR [26], and across modality object classification SAR [27]. These datasets were specifically selected to evaluate the Omni-Modeler across concept domains, complexity, and image modality. Each of these datasets contain 10 classes. Examples images from each dataset are shown in Figure 6.

## 5.3. Setup

The considered cases vary across two factors, the dataset and the number of samples used to generate the dictionary. The three considered datasets are presented above. The number of samples considered are set to 8, 64, and 128. All experiments are ran using the VGG19 DNN with all its layers using a 64 letter alphabet, 64 word vocabulary, 15 nearest neighbors. All images were resized to 56 by 56.

The Omni-Modeler has a stochastic performance dependent on the randomly selected set of

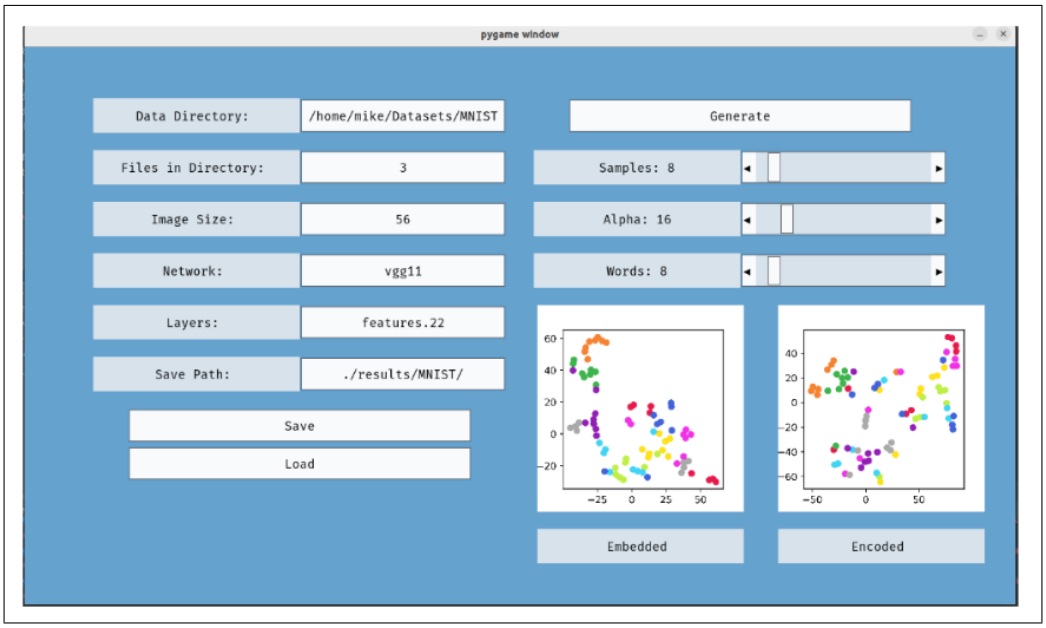


Figure 4: The Language Development UI allows the user to visualize the discriminability of the embbedded and encoded feature space as configuration hyper-parameters are selected.

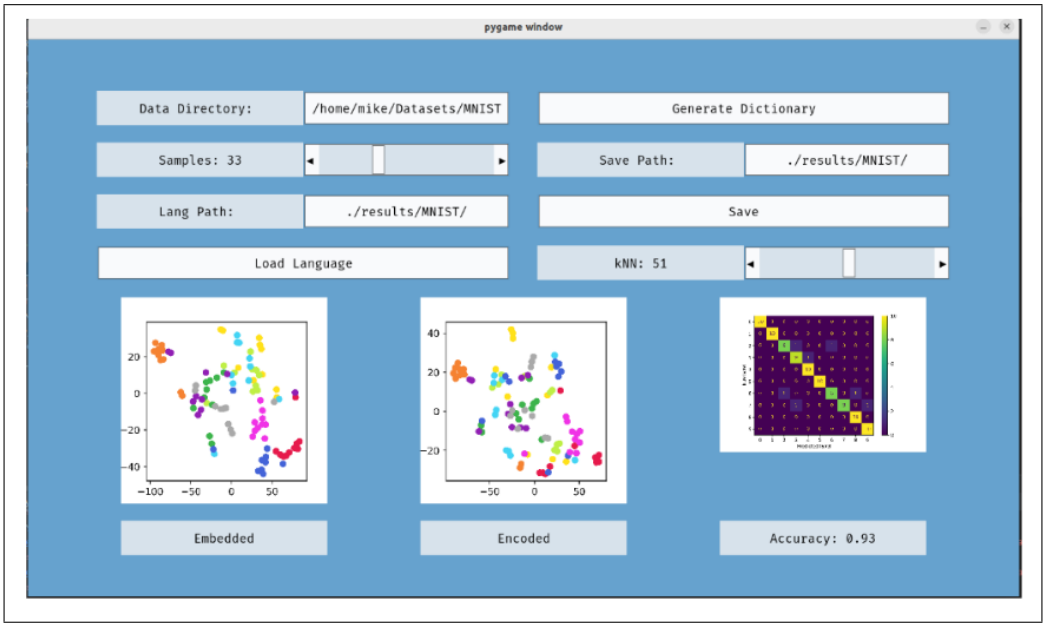


Figure 5: The Dictionary Generator UI allows to visualize and quantify the selection of the number of training samples and nearest neighbors used in the dictionary generation and classification process.

$k$ training samples for each class. Therefore, to understand its performance it is necessary to evaluate over multiple trials. In this study 10 trials are completed for each considered case. Each trial consist of language calculation, dictionary generation, and evaluation on 100 randomly selected images from the dataset. The results are presented as histograms and their probability distribution functions (PDF) in the following section.


Figure 6: Example images from the MNIST, CIFAR, and SAR datasets.

## 6. Results

The results on the Omni-Modeler evaluation are presented in Figure 7 for each of the three considered datasets. The histograms and resulting PDF are plotted side-by-side for each dataset. The Omni-Modeler classification performances are evaluated using 8, 64, 128 training samples per class.

These results show an increasing classification performance with increasing number of training samples. The CIFAR dataset proved to be the most challenging with an average accuracy of 43% using 128 training samples per class. MNIST was a less challenging task with an average accuracy of 97% using 128 training samples per class. The SAR dataset fell in-between with 88% using 128 training samples per class. The SAR data set showed the most rapid increase in performance with increasing training samples jumping from 40% to 82% with 8 and 64 training samples per class.

## 7. Discussion

Overall, these results demonstrate the Omni-Modeler capability to rapidly adapt to a variety of novel image datasets. The classification performance of 43% with 10-way 128-shot was not state-of-the-art compared to PT+MAP+SF+SOT with 89.94% with 5-way 1-shot [19]. However, the intended task and training requirements are not the same as traditional benchmark FSL. The Omni-Modeler was intended to a generalized approach for applications with limited training sets and from that perspective performed well shown by the SAR results. The Omni-Modeler was able to extend across imaging modalities achieving 88% accuracy with 128-shots while the previous state-of-the-art achieved 36.44% [27].

The Omni-Modeler provides several advantages for practical implementation of visual classification in training data limited applications. It provides: 1.) a direct generalized transform, the encoding language, that extracts, adapts, and compresses the latent feature space that only needs to be calculated once per dataset with a subset of unannotated images, 2.) a transparent design path allowing the user to visualize the effects of selecting different DNN embedding networks, layers, alphabet length and vocab size, 3.) a dynamic dictionary learning approach that allows for multi-modal class distributions and knowledge domain updates as simple as updating a list.

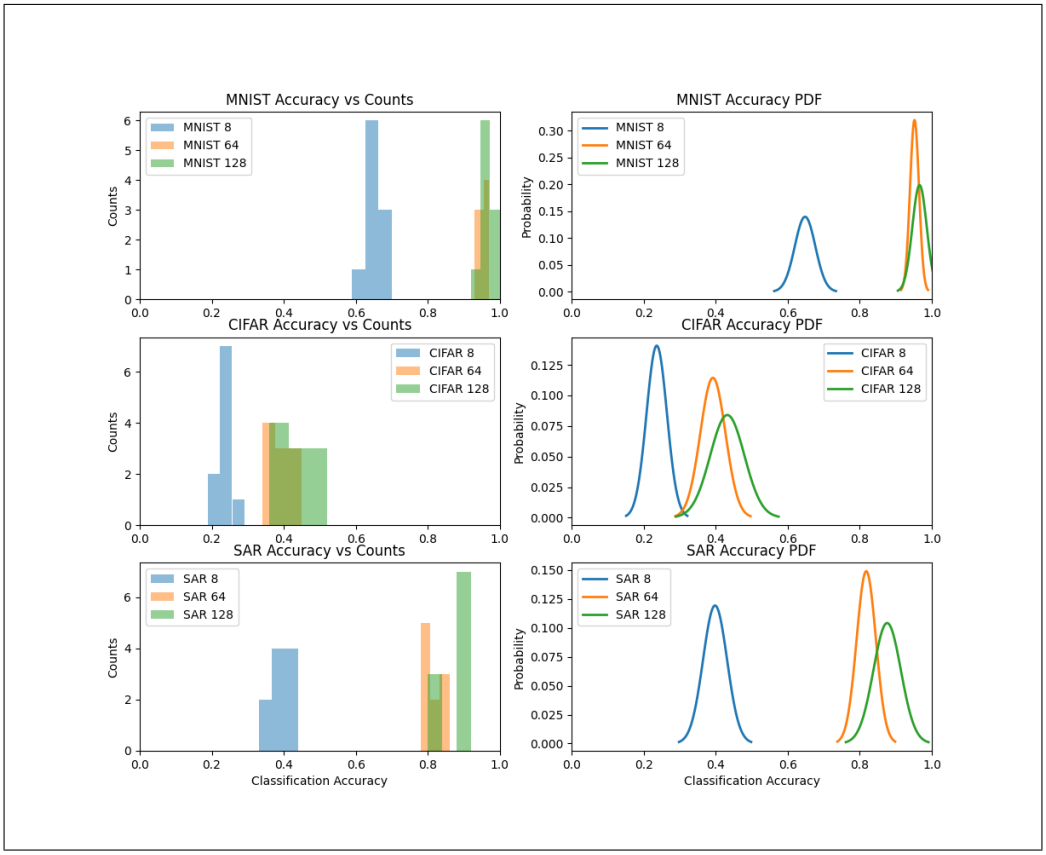


Figure 7: The results of the Omni-Modeler classification performance evaluation are presented as the raw histograms and the accompanying PDF. The evaluation looked at performance across three datasets (MNIST, CIFAR, and SAR) under three conditions of training samples: 8, 64, 128).

## 8. Conclusion

The Omni-Modeler was developed as a generalized image classification algorithm with minimal training requirements and dynamic learning. This study evaluated the Omni-Modeler classification performance on the MNIST, CIFAR, and SAR datasets and demonstrated its ability to rapidly learn to distinguish novel concepts with a few training samples, particularly shown with its 88% classification accuracy on the SAR dataset. With its generalized nature and UI aided implementation, the Omni-Modeler is a suitable classification approach for tasks with limited availability of training data.

## 9. References

[1] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[2] L. Brattain, B. Telfer, M. Dhyani, J. Grajo, and A. E. Samir, "Machine learning for medical ultrasound: status, methods, and future opportunities," *Abdominal Radiology*, vol. 43, pp. 786–799, 2018.

[3] S. Liu, Y. Wang, X. Yang, B. Lei, L. Liu, S. X. Li, D. Ni, and T. Wang, "Deep learning in medical ultrasound analysis: A review," *Engineering*, vol. 5, no. 2, pp. 261 – 275,

2019.

[4] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, 2019.

[5] Y. M. Tang, W. T. Kuo, and C. Lee, "Real-time mixed reality (mr) and artificial intelligence (ai) object recognition integration for digital twin in industry 4.0," *Internet of Things*, vol. 23, p. 100753, 2023.

[6] G. Gallo, F. D. Rienzo, F. Garzelli, P. Ducange, and C. Vallati, "A smart system for personal protective equipment detection in industrial environments based on deep learning at the edge," *IEEE Access*, vol. 10, pp. 110862–110878, 2022.

[7] R. Ozdemir and M. Koc, "A quality control application on a smart factory prototype using deep learning methods," in *2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT)*, vol. 1, pp. 46–49, 2019.

[8] K. Bhosle and V. Musande, "Evaluation of cnn model by comparing with convolutional autoencoder and deep neural network for crop classification on hyperspectral imagery," *Geocarto International*, vol. 37, no. 3, pp. 813–827, 2022.

[9] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.

[10] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, jun 2020.

[11] X. Li, Z. Sun, J.-H. Xue, and Z. Ma, "A concise review of recent few-shot meta-learning methods," *Neurocomputing*, vol. 456, pp. 463–468, 2021.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[13] G. R. Koch, "Siamese neural networks for one-shot image recognition," 2015.

[14] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," 2017.

[15] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," 2017.

[16] T. Chobola, D. Vašata, and P. Kordík, "Transfer learning based few-shot classification using optimal transport mapping from preprocessed latent space of backbone neural network," 2021.

[17] X. Chen and G. Wang, "Few-shot learning by integrating spatial and frequency representation," 2021.

[18] Y. Hu, V. Gripon, and S. Pateux, "Leveraging the feature distribution in transfer-based few-shot learning," 2021.

[19] D. Shalam and S. Korman, "The self-optimal-transport feature transform," 2022.

[20] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (Los Alamitos, CA, USA), pp. 4367–4375, IEEE Computer Society, jun 2018.

[21] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, and M. Cord, "Learning representations by predicting bags of visual words," 2020.

[22] W. A. Qader, M. M. Ameen, and B. I. Ahmed, "An overview of bag of words;importance, implementation, applications, and challenges," in *2019 International Engineering Conference (IEC)*, pp. 200–204, 2019.

[23] P. Bateni, R. Goyal, V. Masrani, F. Wood, and L. Sigal, "Improved few-shot visual classification," 2020.

[24] Y. Shang, "A central limit theorem for randomly indexed m-dependent random variables," *Filomat*, vol. 26, 11 2012.

[25] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[26] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[27] S. Low, O. Nina, A. D. Sappa, E. Blasch, and N. Inkawhich, "Multi-modal aerial view object classification challenge results - pbvs 2023," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 412–421, 2023.