DEEP LEARNING ARCHITECTURES FOR 2D AND 3D SCENE PERCEPTION

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

RINA BAO

Dr. Kannappan Palaniappan, Thesis Supervisor

JULY 2021

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

DEEP LEARNING ARCHITECTURES FOR 2D AND 3D SCENE PERCEPTION

presented by Rina Bao,

a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Kannappan Palaniappan

Dr. Wenjun Zeng

Dr. Yunxin Zhao

Dr. Zhihai He

Dr. Matthew R. Maschmann

ACKNOWLEDGMENTS

First of all, I wish to express sincere and deepest gratitude to my advisors, Professor Kannappan Palaniappan and previous advisor Professor Wenjun Zeng, for their vision and directions. Their priceless gift to me was not just skills and knowledge but also unwavering moral support through every stage of my doctoral studies. They have spent a substantial amount of time on my research progress. They inspired and encouraged me endless times during my study. Their excellent insights, brilliant inspiration, and encouragement will continue to instruct and inspire me long beyond my years as a PhD student.

I wish to express my sincere gratitude to Professor YunXin Zhao for her crucial suggestions, helpful reviews in my PhD directions, and her valuable time on fruitful discussions during my study. Her instructions and perspectives have motivated and helped me in more ways than one.

I wish to express my sincere gratitude to Professor Matthew R. Maschmann for his insightful suggestions and instructions, helpful reviews for our collaboration project, which broaden my mind and view.

I would like to thank Professor Zhihai He for being my committee member and providing helpful reviews of my research and his valuable time on discussions.

I want to say many thanks to my parents for their great support, friends, and collaborators for their encouragement.

This research was partially supported by awards from U.S. Army Research Laboratory W911NF-1820285 and Army Research Office DURIP W911NF-1910181 and included work that was approved for public release. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the U. S. Government or agency thereof.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

ABSTRACT

Scene understanding is a fundamental problem in computer vision tasks, that is being more intensively explored in recent years with the development of deep learning. In this dissertation, we proposed deep learning structures to address challenges in 2D and 3D scene perception. We developed several novel architectures for 3D point cloud understanding at city-scale point by effectively capturing both long-range and short-range information to handle the challenging problem of large variations in object size for city-scale point cloud segmentation. GLSNet++ is a two-branch network for multiscale point cloud segmentation that models this complex problem using both global and local processing streams to capture different levels of contextual and structural 3D point cloud information. We developed PointGrad, a new graph convolution gradient operator for capturing structural relationships, that encoded point-based directional gradients into a high-dimensional multiscale tensor space. Using the PointGrad operator with graph convolution on scattered irregular point sets captures the salient structural information in the point cloud across spatial and feature scale space, enabling efficient learning. We integrated PointGrad with several deep network architectures for large-scale 3D point cloud semantic segmentation, including indoor scene and object part segmentation. In many real application areas including remote sensing and aerial imaging, the class imbalance is common and sufficient data for rare classes is hard to acquire or has high-cost associated with expert labeling. We developed MDXNet for few-shot and zero-shot learning, which emulates the human visual system by leveraging multi-domain knowledge from general visual primitives with transfer learning for more specialized learning tasks in various application domains. We extended deep learning methods in various domains, including the material domain for predicting carbon nanotube forest attributes and mechanical properties, biomedical domain for cell segmentation.

Chapter 1

Thesis Contributions and Organization

We have investigated deep-neural-net (DNN) based new approaches to address the challenges for various 2D and 3D vital tasks, and carried out extensive experimental evaluations on large datasets. Our results are highly encouraging, and they suggest promising directions for further explorations. This dissertation presents the following contributions to the area of computer vision:

- **Global-Local Stream Processing Integration for 3D Point Cloud Classification** We investigate this problem in the context of urban semantic 3D point cloud classification or segmentation. A major difficulty in this task is the large variations of object sizes in large-scale point cloud segmentation, which includes extremely large objects like bridges, and very small objects like buildings and trees. To address this problem, we propose a novel deep neural net architecture, Global and Local Streams Network, referred to as GLSNet, to capture both the global and local structures and contextual information in the 3D point cloud data. The GLSNet employs two branches of networks to first decompose the complex problem of segmenting mixed scale objects to simpler processing

sub-tasks at the global and local scales. In addition, the proposed GLSNet++ incorporates a unique graph convolutional network module for demixing or unmixing voxel boundary regions composed of a mixture of object classes by using spatial context dependent feature fusion similar to conditional random fields.

- **PointGrad: Point Gradient Operator for Point Cloud Segmentation** We propose a new graph convolutional operator PointGrad to encode point directional gradients of different orders in 3D point clouds. The PointGrad has a strong power of representing important structural or salient features of point clouds. The proposed PointGradNet architecture embeds the PointGrad module in a set of networks to encode multiple-order and multiple-scale point directional gradients. We demonstrate the efficiency, effectiveness, and robustness of the PointGrad operator using several benchmark datasets at different scales ranging from single object part segmentation to indoor scenes and large city-scale point clouds.

- **MDXNet for Few-Shot Learning and Zero-Shot Learning** We investigate this problem from the standpoint of addressing a fundamental issue in AI, i.e., how to quickly learn to recognize new objects with limited visual data. Despite the recent progresses in AI, it still falls short of human ability in knowledge association in this regard. Although few-shot learning represents a direction in AI along this line, which strives to learn to classify new classes with limited supervision information. To address this problem, we make analogy to human ability of pattern or knowledge association, and propose to exploit rich information in natural images for use in novel application domains that lack informative descriptions of image objects or scenes. We design a novel two-stage DNN architecture, MDXNet, which is verified to achieve new state of the art performance in extensive application domain datasets.

- **DNN-based Carbon Nanotube Structure and Property Discovering** Deep convolutional neural networks combined with a physics-based simulation tool are employed to study the process-structure-property of carbon nanotubes. Deep learning (DL) enables prediction of ensemble properties based on morphological images that can further be correlated to the synthesis parameters through DL classification algorithm. We propose a novel deep neural net architecture, Carbon Nanotube Network, referred to as CNTNet to explore the relationship between the structure attributes and properties of carbon nanotubes.

This dissertation is organized into four chapters covering the overall research and publications.

The chapter on, GLSNet and GLSNet++ for City-Scale Point Cloud Segmentation, covers the semantic segmentation of LiDAR point sets on external environments. There are two publications related to this area:

Rina Bao, Kannappan Palaniappan, Yunxin Zhao, Guna Seetharaman, Wenjun Zeng "GLSNet: Global and Local Streams Network for 3D Point Cloud Classification", IEEE AIPR 2019 (Oral)

Rina Bao, Kannappan Palaniappan, Yunxin Zhao, Guna Seetharaman, Wenjun Zeng "GLSNet++: Global and Local-Stream Fusion for LiDAR Point Cloud Semantic Segmentation Using Contextual Demixing Block " (Revision for IEEE Transactions on Neural Networks and Learning Systems)

The chapter on, PointGrad: Point Gradient Operator for 3D Point Cloud Segmentation, develops a novel multidimensional graph convolution gradient operator for unstructured irregularly sampled data that combines spatial location, orientation and feature tensor information. The paper related to this research study:

Rina Bao, Kannappan Palaniappan, Yunxin Zhao, Guna Seetharaman, Wenjun Zeng "PointGrad: Spatially Varying Tensor Gradient Operator for 3D Point Cloud Segmentation " (Manuscript Revision)

The chapter on, MDXNet for Few-Shot Learning and Zero-Shot Learning, tackles the need for cross-domain generalization. In contrast to current deep learning approaches, which are centered around the collection, manual labeling, and training of extremely large datasets, we explore a learning approach which is an emulation of the human visual system using visual primitive processing streams that is able to learn effectively from very few training samples. The paper related to this research study:

Rina Bao, Kannappan Palaniappan, Yunxin Zhao, Wenjun Zeng "MDXNet: Multiple-Domain Explainable Latent Attribute Network for Cross-Domain Few-Shot Learning and Zero-Shot Learning" (Manuscript Revision)

The chapter on, Domain Applications, covers the use of the deep architectures to two domains including materials discovery, biomedical cell microscopy. There is one publication related to this research study:

Taher Hajilounezhad*, **Rina Bao***, Kannappan Palaniappan, Filiz Bunyak, Prasad Calyam, Matthew R. Maschmann Predicting Carbon Nanotube Forest Attributes and Mechanical Properties Using Simulated Images and Deep Learning (Nature Partner Journal Computational Materials 2021, *these authors contributed equally to this work)

Chapter 2

GLSNet & GLSNet++ for City-Scale Point Cloud Segmentation

Semantic point cloud segmentation assigns a class label to each unstructured 3D point and provides valuable information for navigation, landmark recognition, and building information modeling systems. We propose a novel deep learning architecture for 3D point cloud segmentation that fuses global and local feature representations about 3D scene geometry to capture multiscale structural information and handle large variation in object sizes typical of urban scenes, but is difficult to model using other architectures. The proposed Global and Local Streams Network (GLSNet++) further incorporates a unique graph convolutional network for demixing or unmixing voxel boundary regions composed of a mixture of object classes by using spatial context dependent feature fusion similar to conditional random fields. We validate GLSNet++ for semantic LiDAR point cloud segmentation using the five class labeled dataset from the IEEE GRSS Data Fusion Contest Urban Semantic 3D, Track 4 (IEEE DFT4) [11, 12, 13]. GLSNet++ is shown to generalize well when tested on an independently collected urban aerial LiDAR point cloud data set for Columbia,

Missouri. GLSNet++ delivers highly competitive segmentation results at city scale that can be refined for more classes, higher resolution and larger regions.

2.1 Introduction

Volumetric city-scale point cloud segmentation is a challenging problem due to high degree of variation in object scale, mixed classes, complex object shapes, high density of similar objects and high computational cost for volumetric processing. A typical point cloud has anywhere from several hundred thousand to millions of 3D points. The object classes may be very large, such as water, which requires a segmentation approach that can capture long range, global structural and contextual information. While some classes may have very small objects, such as buildings and bushes or trees, which require the segmentation approach to capture fine-scale local information in order to delineate the salient object boundaries. A novel approach for efficiently fusing both global and local, structural and contextual feature information for accurate large-scale point cloud segmentation is investigated.

We believe that designing a network with modules to explicitly target global and local information processing can help address such a challenging problem, where the global features capture long range and large scale structural and contextual information, and the local feature captures small scale structural and contextual information. Motivated as such, we previously proposed a two-branch network architecture, Global and Local Streams Network [1] (GLSNet), to decompose such a complex problem by leveraging global and local streams for large point-cloud segmentation.

In GLSNet, the global branch processed a full set of points of each cloud for accurate classification on large areas, while the local branch processed a subset of points each time for accurate classification on small areas, and the complementary geometric information of different scales was integrated by a max pooling operation. In the cur-

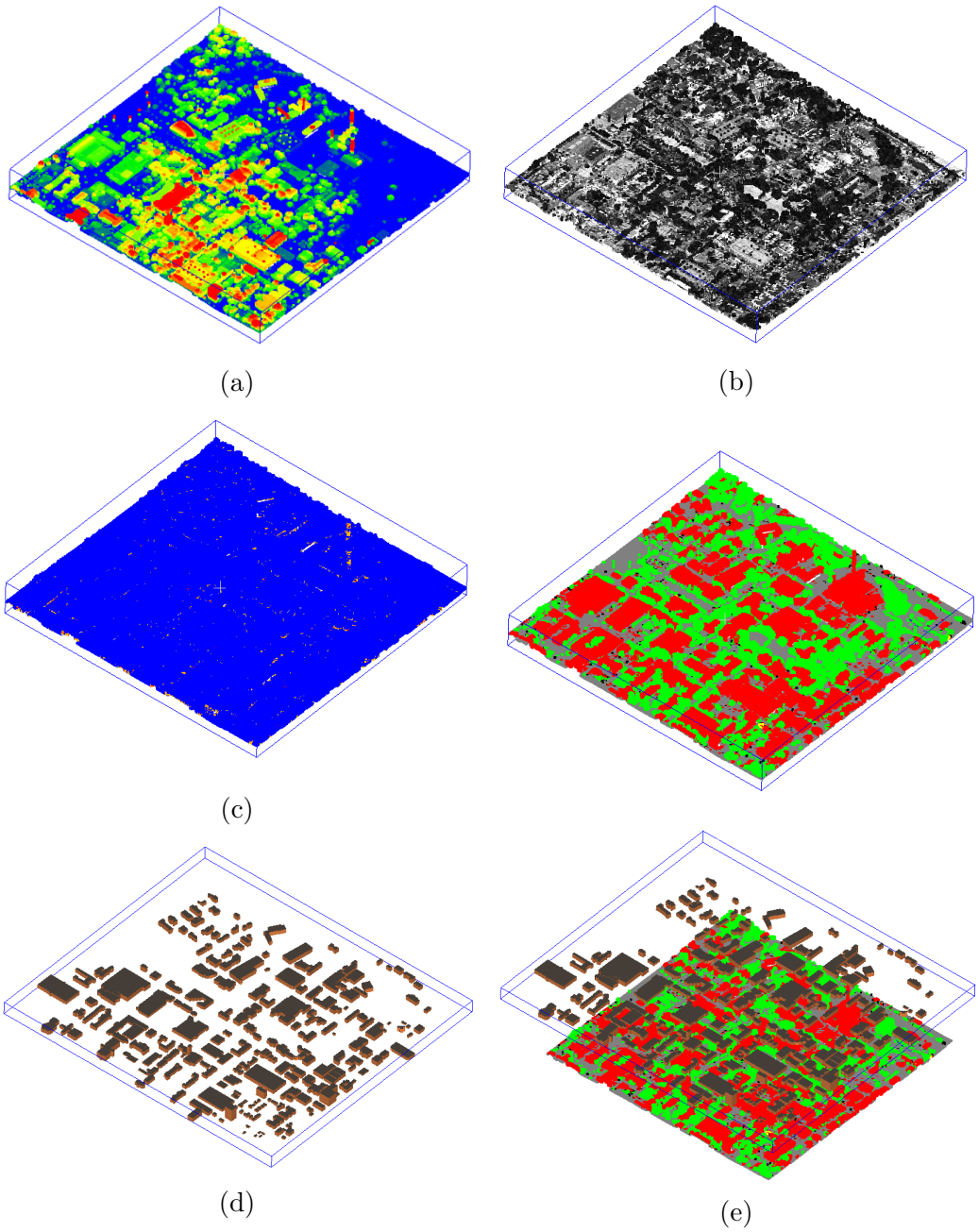


Figure 2.1: LiDAR point cloud for Columbia, Missouri (COU30) showing: (a) colored height map (z-value), (b) LiDAR return intensity (0 to 255), (c) LiDAR return number (0 to 4), (d) building footprints from OpenStreetMap (OSM), (e) LiDAR point cloud semantic segmentation using the proposed GLSNet++, and (f) GLSNet++ segmentation result overlaid with OSM building footprints in an orthographic projection.

rent work, we capitalize on the global-local information fusion approach of GLSNet to extend it significantly in several directions. In network architecture, we consider not

only parallel network branches of global-local information streams, called GLSNet-Parallel, but also a cascade global-to-local information flow, called GLSNet-Cascade, where a global network module first provides coarse-scale semantic segmentation which is followed by a local network module to refine the global branch produced semantic segmentation. In information fusion, different from GLSNet-Parallel and GLSNet-Cascade, we propose a novel context-dependent graph based fusion method. Our new method takes the global and local prediction results as input, and minimizes the fused prediction errors based on the different scales to optimize the semantic segmentation performance. Considering the fact that in many real applications, the object labels for point cloud semantic segmentation are hard to acquire, we further propose to use building footprints from OpenStreetMap as the groundtruth for evaluations on such unlabeled data. In Figure 2.1, we show the point-cloud data of the area of Colubmia, MO that were collected by our local collaborator team. Although these point clouds do not have ground truth labels, the building footprints from OpenStreetMap show very good agreements with the point-cloud segmentation based predictions.

2.1.1 Contribution and Novelty

Our contributions are four-fold: (1) We propose an effective global-local two-branch network GLSNet++ with a novel graph demixing block to effectively fuse global and local information for large-scale point cloud semantic segmentation. To the best of our knowledge, such a design has not been explored in the existing literature for city-scale point clouds. (2) We investigate a novel graph convolutional network architectures for fusing global and local feature embeddings and present extensive results on these architectures. (3) We propose using building footprints from OpenStreetMap for evaluating semantic segmentation on new point clouds when ground-truth labels are not available. (4) We present promising experimental results on airborne LiDAR

point clouds that demonstrate the benefit of our approach.

The subsequent parts of this chapter are organized as the following. Section 4.2 reviews the related work in point cloud segmentation. Section 2.3 describes different fusion structures of global and local branches, and details the design of our GLSNet++ with the demixing block. Section 2.4 describes the implementations details and experimental datasets and explains the procedure for evaluating semantic segmentation on unlabeled point clouds in real application scenarios. Section 4.5 presents quantitative results in mIoU and overall accuracy as well as visualization results of our methods on a public benchmark and our own collected LiDAR point cloud. We conclude this work in section 4.6.

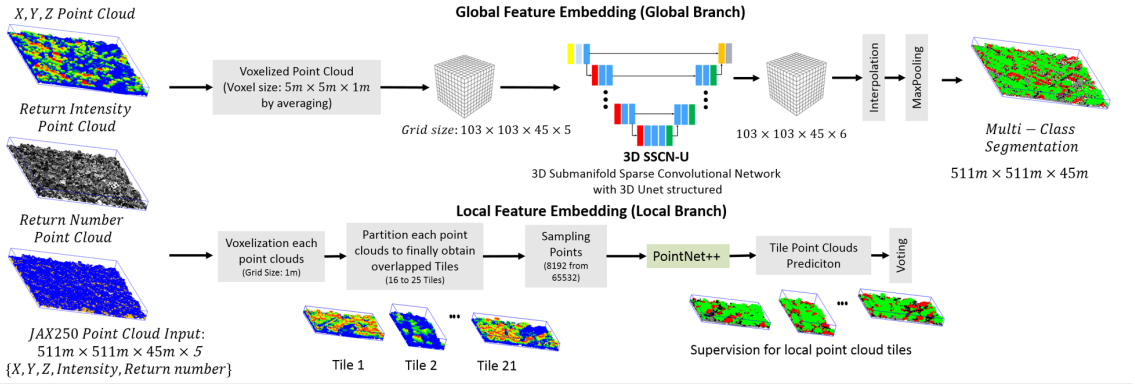
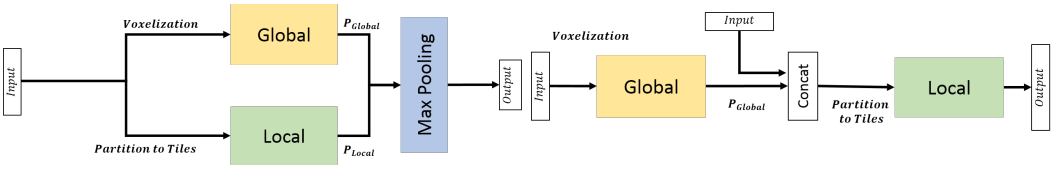


Figure 2.2: Our dual backbone GLSNet [1] uses two feature embeddings including a Global 3D-stream (top row) and a Local 3D-stream (bottom row) that extract coarse-scale and fine-scale geometric information from the input 3D point cloud. The two global and local streams are trained in parallel with the same supervision and same loss functions. The SoftMax classification output layer from each stream is fused together using a MaxPooling layer for the final classification.



(a) GLSNet-Parallel (b) GLSNet-Cascade

Figure 2.3: Different fusion methods of global and local streams.

2.2 Related Work

Currently, deep learning is becoming more widely used for point cloud semantic segmentation. We provide an overview of point cloud segmentation techniques, with a focus on recent deep learning methods for large-scale point cloud segmentation. We categorize the popular deep neural network methods for 3D point cloud segmentation into those with and without 3D convolutions. (I) **3D-convolutional neural networks**: The work of [14] proposes a 3D U-Net structured network for point cloud segmentation. To reduce the computation cost of 3D convolution, [15] proposes highly efficient convolution Submanifold Sparse Convolution (SSC), and the subsequently improved Submanifold Sparse Convolution Network (SSCN) [16] which can process a full 3D point cloud at one time with strong performance. Recent works explored specialized design of convolution kernels for 3D points [17, 18, 19]. (II) **Neural networks without 3D convolutions**: PointNet [20] and its follow-up works fall in this category. PointNet learns from unordered point clouds by point-wise encoding and aggregation through a global max pooling, but its capacity is insufficient in capturing contextual information. Thus, the followed-up work PointNet++ [21] proposes a hierarchical architecture to capture local geometric details at different scales to overcome the weakness of PointNet. [22] proposes a recurrent slice network to model local dependency information. PointSift [23] proposes an orientation-encoding unit to capture different orientation representations of points. PointHop [24] proposes an explainable learning method for point cloud classification. PointGrid [25] proposes a point quantization network to facilitate learning local geometry shapes. (III) **Graph Neural Networks** Graph Neural Networks are explored in many research problems[26, 27, 28, 29]. There are also graph network approaches to point cloud segmentation [30, 31]. PointCNN [32] proposes a $\mathcal{X}$-Conv layer to exploit certain canonical ordering of points. Dynamic Graph CNN (DGCNN) [33] suggests an alternative grouping method to ball query that is used in PointNet++ [21]. Superpoint

Graphs (SPG) [34] adaptively partitions point clouds by contextual relationship encoded superpoint graphs. PartNet [35] proposes a top-down recursive decomposition strategy for shape segmentation.

As has been pointed out by the analysis in [1], for large-scale point cloud segmentation, most existing methods need to slice the point clouds to multiple subsets and process one subset at one time, which inevitably weaken their ability in global contextual and structural reasoning. For 3D convolution networks, for example SSCN, regular strategies to capture long range information such as using deeper network, increasing kernel size will no doubt increase computation cost, and adding many pooling layers will cause local information loss.

City-Scale Point Cloud Segmentation Recently, several methods have shown good performance on large scale point cloud segmentation. The method of [34] tackles the large-scale point cloud segmentation problem by adaptively partitioning point clouds with contextual relationship encoded superpoint graphs. The works of [17, 18, 19] focus on designing convolution kernels for points. Specifically, the method of Pointwise Convolution Neural Network (PCNN) [17] proposes a convolution operator for each point which bins its nearest neighbors into kernel cells before convolving with the kernel weights. KPConv [36] proposes Kernel Point Convolution to directly operate on point clouds without any intermediate representation. Additionally, there are specialized convolution designs such as ConvPoint [37] that employs continuous convolutions for cloud processing. RandLA-Net [38] proposes to relying on expensive random point sampling technique to help with large-scale 3D point cloud segmentation. SqueezeSeg [39] designs a conditional random field (CRF) which is implemented as a recurrent layer.

Different from the above methods, we propose to use global and local feature embedding streams (or branches) to process different levels of structural and contextual information in point clouds, and in GLSNet++, we further design a novel demixing

block to perform global and local information decomposition and aggregation for 3D point cloud classification.

2.3 Semantic Segmentation of 3D Point Clouds

In this section, we first describe the point cloud semantic segmentation problem, then briefly review the global branch and local branch network architecture of GLSNet [1], and finally, introduce our three network design schemes for global and local information fusion.

Let $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_i, ..., \mathbf{s}_N\}$ be the set of points in a generalized point cloud representation, with N points in D-dimensional feature space. For LiDAR point clouds we use,

$$\mathbf{s}_i = \{x_i, y_i, z_i, I_i, R_i\} \tag{2.1}$$

where $\mathbf{s}_i$, is a 5-D vector with $\{x_i, y_i, z_i\}$ being the 3D spatial location of the point in world or relative geometric coordinates, I_i, the measured LiDAR *return intensity*, and R_i its LiDAR *return number*. The semantic segmentation task is to predict a semantic label for each point, $\mathbf{s}_i$. For the whole point set $\mathbf{S}$, the label set, $\mathbf{Q}$, is defined as the one-hot vector for each point. The point label set for $\mathbf{S}$, is $\mathbf{Q} \in N \times \mathbb{R}^C$, where C is the total number of total semantic classes. Accordingly, the semantic segmentation task is then, given a point set $\mathbf{S}$, to predict its class label set $\mathbf{Q}$. Denoting the estimate of $\mathbf{Q}$ as $\widehat{\mathbf{Q}}$, the prediction task, $\mathbf{F}$, is defined as,

$$\widehat{\mathbf{Q}} = \mathbf{F}(\mathbf{S}) \tag{2.2}$$

where $\mathbf{F}$ is the learned non-linear function that predicts semantic segmentation labels $\widehat{\mathbf{Q}}$ for the point set $\mathbf{S}$.

2.3.1 GLSNet Deep Learning Architectures

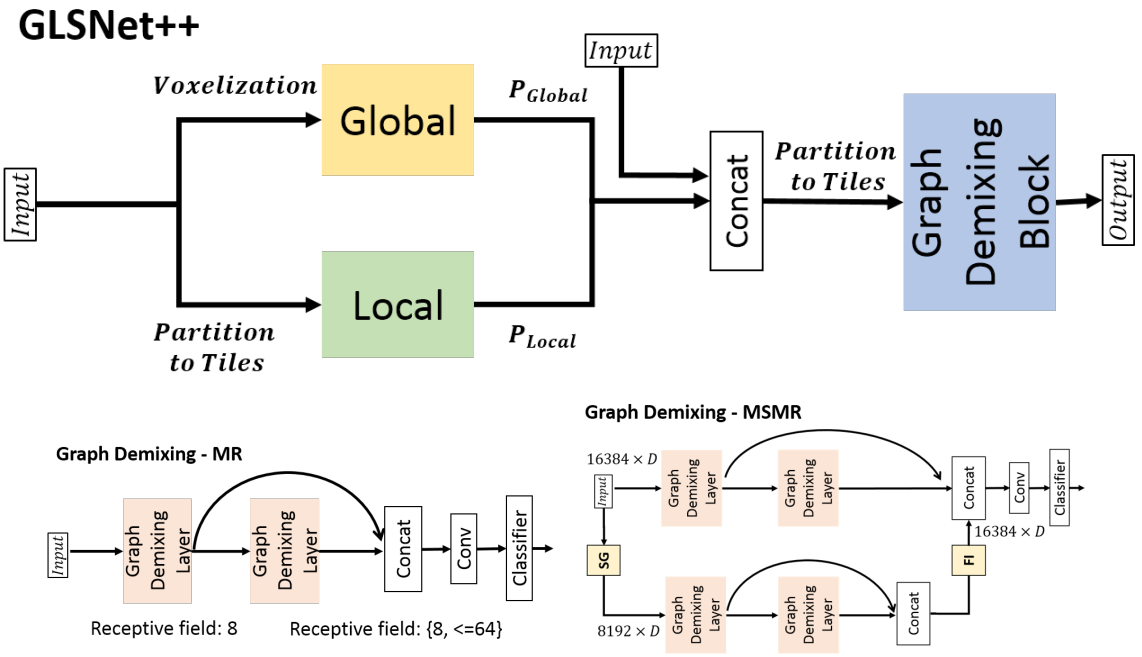


Figure 2.4: Proposed GLSNet++ structure with parallel feature embedding fusion followed by graph convolutional demixing (top). Two types of demixing blocks: Multi-Receptive (MR) Field Graph Demixing (middle) and Multiscale Multi-Receptive (MSMR) Field Graph Demixing (bottom) SG is a sample grouping downsampling block (i.e., choosing furthest points) and FI is a feature interpolation block.

We illustrate the GLSNet deep learning architecture consisting of a *global branch* and a *local branch* in Figure 2.2. The *global branch* is designed to capture long-range structural and contextual information with several considerations. (1) Sparse voxelized grids are taken as the input because they preserve point clouds structure, unlike scattered and unordered point set, and a sparse voxelization that employs large voxel grid radius and aggregation of input point clouds facilitates efficient global information capture without needing a very deep network. (II) A UNet structured Sparse Submanifold Covolution Network (SSCN-U) is employed in global feature representation learning for semantic segmentation due to the high efficiency of sparse submanifold convolution operation. In short, the global branch first voxelizes the

entire point cloud, and then performs SSCN-U on the sparse voxelized point cloud grids. The *local branch* is designed to capture local contextual information and handling fine-grained point cloud segmentation. The sophisticated network PointNet++ serves this purpose well and is taken for this branch. Specifically, the local branch partitions the entire point cloud into overlapped tiles and employs PointNet++ to operate on each tile (subset of points) to classify local areas. The two streams extract geometric information in a complementary way (see [1] for further details). The two branches are trained separately with the supervision of point cloud segmentation groudtruth. We define $\mathbf{P}_{Global} \in N \times \mathbb{R}^C$ as the output probability of the global branch $\mathbf{F}_{Global}$, and $\mathbf{P}_{local} \in N \times \mathbb{R}^C$ as the output probability of the local branch $\mathbf{F}_{Local}$.

Our global-local information fusion schemes: (I) GLSNet-Parallel: as shown in Figure 2.3 (a), the global branch and local branch work in parallel and they are trained independently, as proposed in GLSNet [1]. (II) GLSNet-Cascade: As shown in Figure 2.3 (b), instead of using the parallel structure of two branches, we design a cascade structure to make the global and local network modules work in tandem, following the coarse to fine refinement framework. (III) GLSNet++: as shown in Figure 2.4, GLSNet++ is designed to benefit from both the parallel and cascade structures of (I) and (II), and it has a novel graph demixing block for multi-scale (MS) and multi-receptive field (MR) context-dependent information fusion, producing the best results for 3D point cloud semantic segmentation.

2.3.2 GLSNet-Parallel Streams

As proposed in [1], GLSNet employs parallel global-local branches and use max pooling to leverage the complementary information from these two streams to improve the performance, as shown in Figure 2.3 (a). As a point-set to label set mapping function $\mathbf{F}$, GLSNet performs max pooling over the two prediction vectors of the global branch

network $\mathbf{P}_G$ and the local branch network $\mathbf{P}_L$. The final prediction $\mathbf{P}_{fuse}$ is computed as,

$$\begin{aligned} \mathbf{F}_{parallel}(\mathbf{S}) &= \max\{\mathbf{F}_{Global}(\mathbf{S}), \mathbf{F}_{Local}(\mathbf{S})\} \\ \mathbf{P}_{fuse} &= \max(\mathbf{P}_{Global}, \mathbf{P}_{Local}) \ . \end{aligned} \tag{2.3}$$

2.3.3 GLSNet-Cascade Streams

The GLSNet-Cascade network has a global-to-local structure, where the global module provides a coarse level semantic segmentation prediction, and the local module refines the global prediction. The structure of GLSNet-Cascade is shown in Figure 2.3 (b), where the input to the local module in GLSNet-Cascade is $[\mathbf{S}, \mathbf{P}_{Global}]$, with $[\cdot]$ denoting concatenation. The operation of GLSNet-Cascade is defined as,

$$\mathbf{F}_{cascade}(\mathbf{S}) = \mathbf{F}_{Local}([\mathbf{S}, \mathbf{F}_{Global}(\mathbf{S})]) \ . \tag{2.4}$$

2.3.4 GLSNet++ With Graph Convolutional Demixing Block

The design of GLSNet++ is rooted in our observation that the segmentation errors in GLSNet-parallel are mainly in regions where global and local branches have different predictions and around the object boundaries [1], which can be attributed to the fact that GLSNet-parallel uses a simple max pooling to fuse the outputs of the two branches. Therefore, we design a graph demixing block to help demix the predictions of the two branches in the confused areas for improving global-local fusion. The GLSNet++ workflow is illustrated in Figure 2.4. The graph demixing block is composed of two graph demixing layers, and they are designed to improve segmentation performance for each point by taking into account of the two-branch predictions at the point as well as the predictions at the point's neighborhood. A demixing layer first performs graph aggregation over the contextural points of each point's features,

global probability, and local probability, and the demixing block next utilizes multiple neighborhood sizes or receptive fields by stacking up demixing layers to improve point prediction. The operation of GLSNet++ is,

$$\mathbf{F}_{GLSNet++}(\mathbf{S}) = \mathbf{F}_{Demix}([\mathbf{S}, \mathbf{F}_{Global}(\mathbf{S}), \mathbf{F}_{Local}(\mathbf{S})]) \tag{2.5}$$

The graph convolutional network demixing layer in Figure 2.5, takes as input a neighborhood point set defined as $\mathbf{S} \in N \times 1 \times \mathbb{R}^{D^{c_{in}}}$, where N is the number of points, $D^{c_{in}}$ is the dimension of an input feature vector. For the first demixing layer in the demixing block, $D^{c_{in}}$ is 17 for the IEEE DFT4 dataset, because for each point p_i, its feature representation $\mathbf{s}_i$ is composed of the self LiDAR geometry feature $\{x_i, y_i, z_i\}$ and LiDAR Return Intensity I_i, LiDAR Return Number R_i, and the two-branch probabilities $\mathbf{P}^i_{Global}$ and $\mathbf{P}^i_{Local}$ (when there are six classes, $\mathbf{P}^i_{Global}$ and $\mathbf{P}^i_{Local}$ are each 6-dimensional vector). For the other demixing layers in the demixing block, $D^{c_{in}}$ is the number of channels of the input feature at that layer.

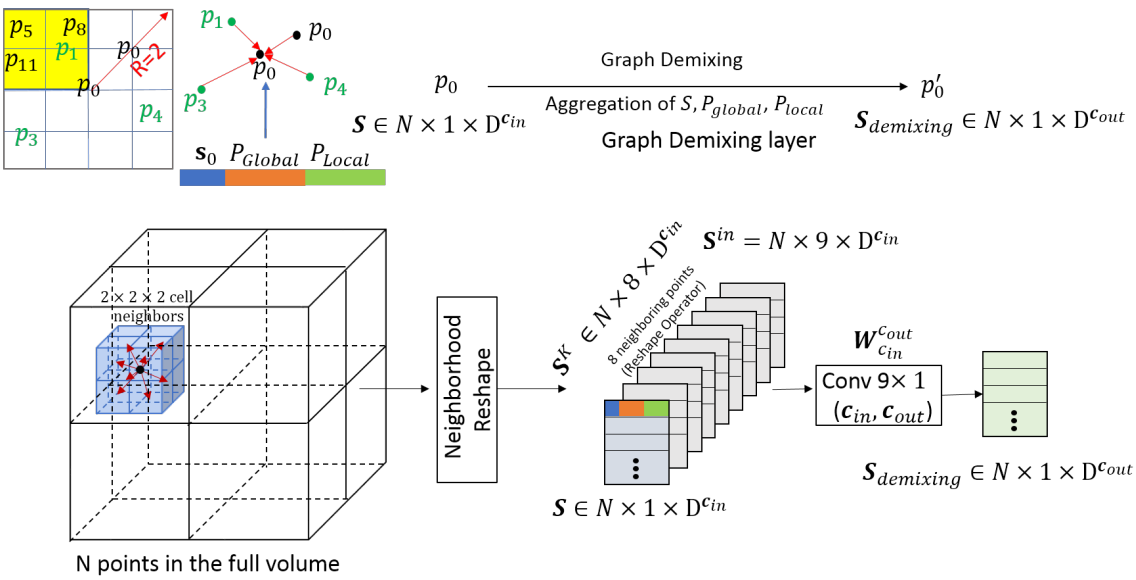


Figure 2.5: Graph convolutional network demixing layer captures local point cloud topology for capturing spatial context in unstructured grids.

We construct a directed graph $G = (V, E)$ to represent the feature vector of each point, where $V = \{1, ..., K\}$ and $E \in V \times V$ are the vertices and edges, respectively. In the demixing layer, we construct G as the neighborhood graph of $\mathbf{S}$ with radius R that includes K points, where $K = R^3$, which is defined as $\mathbf{S}^K$. Note that when multiple points are in one voxels, we choose the nearest point in the grid, and when the grid is empty, we fill the center point in that grid. The graph includes self-loop updates, meaning each node also points to itself. The edge features are learnable, and defined as,

$$\mathbf{S}^{Demixing} = \mathbf{\Theta}(\mathbf{S}, \mathbf{S^K}) = \mathbf{W}_{c_{in}}^{c_{out}}([\mathbf{S}, \mathbf{S^K}]) \tag{2.6}$$

where the nonlinear function $\mathbf{\Theta}$ has a set of learnable parameters $\mathbf{W}_{c_{in}}^{c_{out}}$. In Figure 2.5, we show how the graph demixing layer is designed to incorporate the self features, and global and local probabilities. For each point p_i at location $\{x_i, y_i, z_i\}$ along with feature $\mathbf{s}_i$, its neighborhood set is $\mathbf{s}_i^K$. We use a point p_0 at $\{x_0, y_0, z_0\}$ with $K = 8$ as an example, which means we find its 8 neighbors in a $2 \times 2 \times 2$ grid. The set of neighbors is denoted as $\mathbf{s}_i^K$, with K=8. The input data to the graph is the set $\mathbf{S} \in N \times \mathbb{R}^{D_{c_{in}}}$, where $\mathbf{S} = \{\mathbf{s}_i\}, i = 1, 2, ..., N$, and the output is $\mathbf{S}^{Demixing} \in N \times \mathbb{R}^{D_{c_{out}}}$. The demixing layer concatenates the input point set $\mathbf{S}$, and the neighborhood point set $\mathbf{S}^K$, which leads to $\mathbf{S}^{in} \in N \times 9 \times D_{c_{in}}$ features. A 2-D convolution layer has the parameters $\mathbf{W}_{c_{in}}^{c_{out}}$, the kernel size is 9 where each kernel in $\mathbf{W}_{c_{in}}^{c_{out}}$ is applied to the 9-dimensional features of input $\mathbf{S}^{in}$. This graph demixing layer design fuses the feature information of both the center point and its neighborhood points from the global and local feature embedding streams.

Graph Demixing Block As shown in Figure 2.4, given the input point set, the demixing block stacks two demixing layers with a skip link. The first demixing layer's receptive field is 8, and after the second demixing layer, the receptive field is increased up to 64 (8×8). We concatenate the first demixing layer's output using the skip link with the second demixing layer's output. By doing so, each point has

the receptive fields of its neighborhood in two scales which we name it as Multi-Receptive (MR) field graph demixing block (abbreviated as Graph Demixing-MR), that helps refine the point's semantic segmentation by using different levels of predictions of the surrounding points (we found this to be especially important for the building class since this class always has errors on boundaries). We also design a Multiscale Multi-Receptive (MSMR) field graph demixing block (abbreviated as Graph Demixing-MSMR) which is realized by utilizing the Sampling and Grouping (SG) and Feature Interpolation (FI) operations in Graph Demixing-MR to achieve multiscale operations (SG and FI are similar functions to downsampling and upsampling layers in convolutional network, which are from PointNet++). Because GLSNet++ utilizes different levels of contextual information of semantic label predictions from the global and local branches to refine point semantic prediction, it performs better than either GLSNet-parallel or GLSNet-cascade.

Table 2.1: Details of graph convolutional network Demixing layer and Demixing block architecture.

Module	**Layer**	**Kernel Size**	**# of input channels**	**# of output channels**	**# of Parameters**	**Total Parameters**
DL	Conv	9×1	c_{in}	c_{out}	$9 \times 1 \times c_{in} \times c_{out} + c_{out}$	
	BN	–	–	–	$2 \times c_{out}$	
Demixing-MR	DL1	9×1	17	64	$9 \times 1 \times 17 \times 64 + 64 + 2 \times 64$	66,310
	DL2	9×1	64+3	64	$9 \times 1 \times 67 \times 64 + 64 + 2 \times 64$	
	Conv	1×1	128	128	$1 \times 1 \times 128 \times 128 + 128$	
	BN	–	–	–	128+128	
	Classifier	1×1	128	6	$1 \times 1 \times 128 \times 6 + 6$	
Demixing-MSMR	DL1-s1	9×1	17	64	$9 \times 1 \times 17 \times 64 + 64 + 2 \times 64$	133,190
	DL2-s1	9×1	64+3	64	$9 \times 1 \times 67 \times 64 + 64 + 2 \times 64$	
	DL1-s2	9×1	17	64	$9 \times 1 \times 17 \times 64 + 64 + 2 \times 64$	
	DL2-s2	9×1	64+3	64	$9 \times 1 \times 67 \times 64 + 64 + 2 \times 64$	
	Conv	1×1	256	128	$1 \times 1 \times 256 \times 128 + 128$	
	BN	–	–	–	128+128	
	Classifier	1×1	128	6	$1 \times 1 \times 128 \times 6 + 6$	

2.4 Datasets and Implementations

2.4.1 LiDAR Datasets for Point Cloud Semantic Segmentation

We first train and test the proposed GLSNet using the IEEE GRSS Data Fusion Contest Urban Semantic 3D, Track 4 LiDAR dataset with ground truth; which we referred to as IEEE DFT4. This is a large scale airborne LiDAR point cloud classification dataset of IEEE 2019 GRSS Data Fusion Contest Track 4: 3D Point Cloud Classification Challenge, where given a LiDAR point cloud, the objective is to predict a semantic label for each 3D point. Since the task is to predict semantic labels for the individual points, it is also referred to as point cloud semantic segmentation. In IEEE Data Fusion Challenge, there are six classes: Ground (G), High Vegetation (HV), Building (B), Water (W), Elevated Road (ER), and Unlabeled (U). The Unlabeled category is not used in the performance evaluation as in IEEE DFT4. The information provided for the 3D dataset consists of $\{X, Y, Z, \textit{Intensity}, \textit{Return Number}\}$. There are 110 point clouds for training, 10 for validation, and another hidden 10 for testing on the server. The point clouds are from Omaha, NE and Jacksonville, FL. For details of this IEEE contest, please refer to [13, 11, 12].

We use an independent testing dataset consisting of LiDAR for Columbia, Missouri (COU). The high resolution Columbia point clouds were collected by the Boone County, Missouri local government in 2014 as part of their regular mapping process. The data was collected by a third party using a Leica ALS70 Aerial LiDAR sensor system. The nominal collection scenario called for the acquisition of nominal point spacing of 0.7 meter on the ground. In order to use the data to evaluate our proposed semantic segmentation networks, we crop two large tiles of original point cloud data. The cropped point clouds are named COUorbit28 and COUorbit30. Each tile covers $1,000m \times 1,000m$ areas. COUorbit28 had 4,244,969 points, and COUorbit30

had 4,368,678 points. in Figure 2.1, we colored the COU point cloud by their height, intensity and return number. Since the ground-truth of the COU point cloud segmentation were unavailable, a common scenario in real cases, we propose a new workflow to test and evaluate on these point clouds by using a system trained from a different point-cloud dataset, i.e., the IEEE 2019 challenge training set, and we describe the workflow below. Currently, the dataset is being annotated. We have released current annotation here for the two large point clouds with OSM ground-truth. Later, we will annotate more semantic classes, including Roads, Ground, Trees, and Water as we continuously update the ground truth labels over time.

2.4.2 Training GLSNet++

Global and Local Branches In Global Branch training, the key is to capture long range contextual and structural information. In order to effectively realize this, we employed large grid size to generate sparse point cloud which aggregates grid features. We use grid size of $5m \times 5m \times 1m$. If there were multiple points in a voxel grid, all the input feature vectors in each grid were averaged. The SSCN-U had 16 filters in the first layer and a total of 7 UPlanes, and added 16 more filters for each U-Plane which had 2 residual blocks. We trained global branch with the Adam optimizer with an initial learning rate 0.001 for 512 epochs. In Local Branch training, the key is to capture local tiles' fine contextual and structural information. Thus, we first quantized the whole volume of a point cloud with a grid size of 1m, and then split them into overlapping tiles. The local branch employing PointNet++ was trained with local tiles. We trained local branch using the Adam optimizer with an initial learning rate 0.001 for 200 epochs. For additional details of data augmentation in training the global and local branches, please refer to [1]. In [1], we only used 90% training data, and the remaining 10% for validation due to the fact that the official validation set was not accessible when we took part in the challenge. In this work, for

fair comparisons among different network structures, we retrained the local branch with 100% training data, and used the official validation set, while the global branch was directly from GLSNet trained model.

Graph Demixing Layer and Graph Demixing Block The demixing layer is composed of a convolution layer (Conv) with the number of input channel size c_{in}, the number of output channel size c_{out}, the kernel size 9×1 (parameter size $9 \times 1 \times c_{in} \times c_{out} + c_{out}$) and a Batch Normalization layer (BN) (parameter size c_{out} for beta and parameter size c_{out} for gamma in BN). Table 2.1, provides details of the Demixing Layer (DL), Demixing block-MR (Demixing-MR) and Demixing block-MSMR (Demixing-MSMR) architecture. In the Demixing-MR block, there is +3 in DL2 # of input channels because we concatenated the $\{X, Y, Z\}$ information in the layer to also encode the geometry information. The parameter sizes of Demixing-MR and Demixing-MSMR can be seen in Table 2.1, with around 0.067M and 0.133M, respectively.

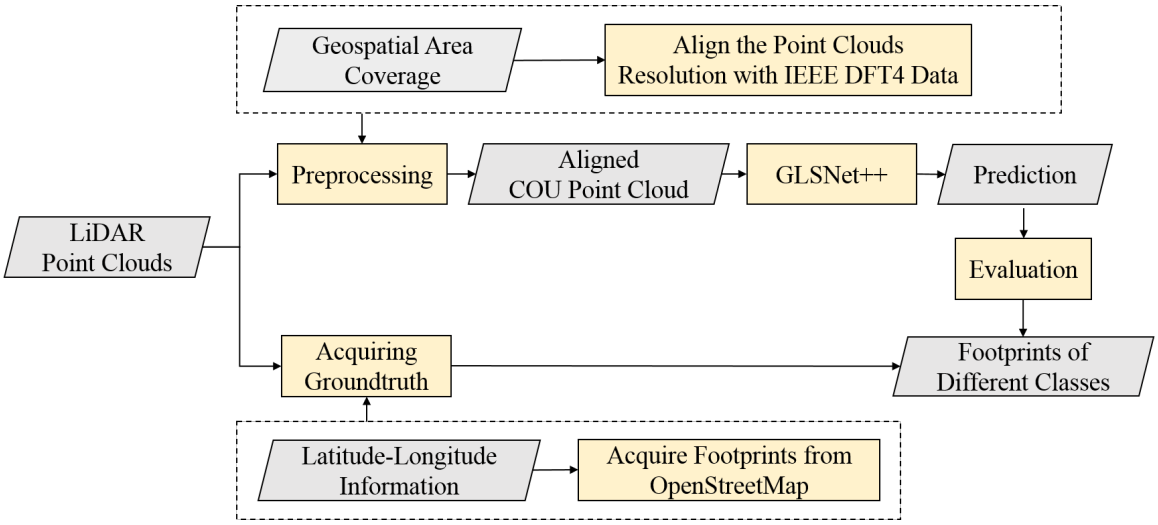


Figure 2.6: Workflow of testing and evaluation on unlabeled LiDAR point clouds.

GLSNet++ uses a two-stage training scheme. In the first stage, the global and local branches were trained independently. In the second stage, only the demixing block was trained to fuse the two branches' prediction results. We used the prepro-

Table 2.2: Class IoU for different percentage training data on 2019 IEEE DFT4 Test Set. †Results from [1].

Streams	Method	G	HV	B	W	ER	OA	mIOU
Single-Stream	**Global-branch (90%)†**	97.40	91.82	85.50	94.18	69.98	96.88	87.77
	Local-branch (90%)†	95.65	95.47	83.64	80.99	79.03	96.21	86.95
	Local-branch (100%)	96.73	95.12	86.19	93.21	76.08	96.41	89.47
Dual-Stream	**GLSNet-Parallel (90%)†**	97.56	93.94	87.56	95.64	77.90	97.37	90.52
	GLSNet-Parallel	97.73	93.83	88.02	96.44	76.47	97.47	90.50

Table 2.3: Class IoU for different percentage training data on 2019 IEEE DFT4 Test Set. †Results from [1].

Streams	Method	Parameter Size	G	HV	B	W	ER	OA	mIOU
Single-	**Global-branch†**	7.531M	97.40	91.82	85.50	94.18	69.98	96.88	87.77
	Local-branch	0.970M	96.73	95.12	86.19	93.21	76.08	96.41	89.47
Dual-	**GLSNet-Parallel**	8.50M†	97.56	93.94	87.56	95.64	77.90	97.37	90.52
Stream	**GLSNet-Parallel**	8.50M	97.73	93.83	88.02	96.44	76.47	97.47	90.50

Table 2.4: Semantic segmentation accuracy on 2019 IEEE DFT4 LiDAR segmentation benchmark using class IoUs comparing proposed architectures including single-stream and dual-stream methods. GLSNet++MSMR uses GLSNet-Parallel that combines Local- and Global-backbones (see Figure 2.2) followed by a Multiscale Multi-Receptive Demixing Block (see Figure 2.4).

Streams	Method	Parameter Size	G	HV	B	W	ER	OA	mIOU
Single-Stream	**Global-branch**	7.531M	97.40	91.82	85.50	94.18	69.98	96.88	87.77
	Local-branch	0.970M	96.73	95.12	86.19	93.21	76.08	96.41	89.47
Dual-Stream	**GLSNet-Parallel**	8.501M	97.73	93.83	88.02	96.44	76.47	97.47	90.50
	GLSNet-Cascade	8.501M	97.34	95.05	88.57	93.23	78.17	97.48	90.47
GLSNet++	**GLSNet++-MR**	8.634M	97.83	96.06	90.15	95.65	79.81	97.92	91.90
	GLSNet++-MSMR	8.634M	97.86	96.20	90.37	95.84	79.84	**97.97**	**92.02**

Table 2.5: Class IoU of GLSNet++-MR and GLSNet++-MSMR on 2019 IEEE DFT4 Test Set.

Method	Parameter Size	G	HV	B	W	ER	OA	mIOU
GLSNet-Parallel+PointNet++	0.970M	97.29	95.29	88.36	95.70	80.90	97.50	91.51
GLSNet++-MR	0.066M	97.83	96.06	90.15	95.65	79.81	97.92	91.90
GLSNet++-MSMR	0.133M	97.86	96.20	90.37	95.84	79.84	**97.97**	**92.02**

cessing codes provided by the baseline system of IEEE DFT4 [42] to process our data, which first quantized the point clouds by the cell size of 1 meter, and then partitioned

Table 2.6: Class IoU of GLSNet++-MR and GLSNet++-MSMR on 2019 IEEE DFT4 Test Set.

Method	Parameter Size	G	HV	B	W	ER	OA	mIOU
GLSNet-Parallel+PointNet++	970,218	97.29	95.29	88.36	95.70	80.90	97.50	91.51
GLSNet++-MR	66,310	97.83	96.06	90.15	95.65	79.81	97.92	91.90
GLSNet++-MSMR	133,190	97.86	96.20	90.37	95.84	79.84	**97.97**	**92.02**

Table 2.7: Ablation study for different grid resolutions and sampling point set sizes.

Method	Grid Size (m)	Sampling Size	G	HV	B	W	ER	OA	mIOU
GLSNet++-MR	1	8192	97.39	95.57	88.75	96.02	79.41	97.60	91.43
GLSNet++-MR	0.1	8192	97.62	95.85	89.56	95.31	80.95	97.77	91.86
GLSNet++-MR	0.1	16384	97.83	96.06	90.15	95.65	79.81	**97.92**	**91.90**
GCN2neighbor8	0.1	8192	97.38	94.87	88.12	96.13	79.60	**97.48**	**91.22**
GCN2neighbor16	0.1	8192	97.26	94.89	87.79	95.55	76.72	**97.39**	**90.44**
GCN7neighbor8	0.1	8192	97.21	94.76	87.81	95.58	75.16	**97.34**	**90.10**

Table 2.8: Class IoU recent architecture comparison for 3D semantic segmentation on 2019 IEEE DFT4 Test Set. * indicates that the performances are produced with the publicly available code.

Method	G	HV	B	W	ER	OA	mIOU
SSCN-U	97.40	91.82	85.50	94.18	69.98	96.88	87.77
PointNet++*	95.4	95.2	83.7	88.4	76.6	96.3	87.9
PointNet++(MSG)*	96.8	94.9	85.8	93.1	74.8	96.9	89.1
DGCNN*	96.4	96.2	86.3	96.2	48.7	96.9	84.8
PointCNN*	96.7	95.4	88.3	88.3	83.5	97.3	90.4
PointSIFT*	97.4	96.1	88.4	91.5	79.3	97.5	90.6
PointConv*	97.6	95.5	89.1	92.1	76.3	97.6	90.1
PointSIFT [23]*	–	–	–	–	–	97.55	91.02
UPNet (Rank-1)(EN) [40]	98.74	96.11	93.31	96.54	89.04	98.57	**94.55**
UPNet (Rank-1) [40]	97.87	95.79	89.98	94.94	86.48	97.94	93.01
ASNet (Rank-2) [41]	98.79	96.32	93.62	95.32	88.66	**98.62**	94.54
GLSNet [1]	97.56	93.94	87.56	95.64	77.90	97.37	90.52
GLSNet++	97.86	96.20	90.37	95.84	79.84	97.97	92.02

the point clouds to tiles where each tile had around 65,536 points. For details of the procedure, please refer to [42]. We also tried quantization with the cell size of 0.1 meter, which showed better performance in our ablation study. The demixing block was trained with the training set of IEEE DFT4 by using the Adam optimizer for

200 epochs, with the learning rate 0.001, momentum 0.9, and learning rate decay 0.7. We choose the model which has the highest overall accuracy on validation data as the best model, and report the best model performance on the held-out test set on the test server from IEEE DFT4. For testing on the COU point clouds, we use the model trained on IEEE DFT4, following the workflow described below in Subsection 2.4.3.

Table 2.9: Parameter Size and Model Size.

Methods	Model Size (MB)	OA	mIOU
SSCN	28.61	96.29	87.77
PointNet++	28.61	96.29	87.77
PointNet++(MSG)	3.69	96.3	87.9
DGCNN	9.07	97.3	84.8
PointCNN	43.91	97.5	90.4
PointConv	82.62	97.6	90.1
PointSIFT	51.6	97.5	90.6
DPNet	32.23	97.94	93.01
GLSNet++(Ours)	32.67	97.78	92.02

2.4.3 Testing and Evaluation

Labeled Point Clouds: We use the standard measures of mean Intersection over Union (mIoU) and overall accuracy (OA) for 3D point cloud classification as in [43].
Unlabeled Point Clouds: In real scenarios, it is often very difficult to obtain large-scale point cloud segmentation labels since manually label city-scale point clouds with millions of points is extremely time-consuming. Here, we describe our method for testing and evaluating on such unlabeled point clouds, as shown in the flowchart in Figure 2.6. Although we use the unlabeled COU point clouds as the example here, we believe that our method can be generally applied to testing on other unlabeled point cloud datasets as well. Before testing, we need to preprocess the COU point clouds since the LiDAR data were collected by LiDAR teams different from that of

the IEEE DFT4 which is used as our training set. The preprocessing procedure aligns the resolution of the testing point cloud (COU) to that of the training data (IEEE DFT4). For evaluation, the OpenStreetMap's building footprints are acquired as the groundtruth labels for use in evaluating the building segmentation performance. Note that there could be some misalignment between the footprints from OpenStreetMap with our collected point clouds since these were not collected at the same time, and some urban structures may have been changed during this time interval.

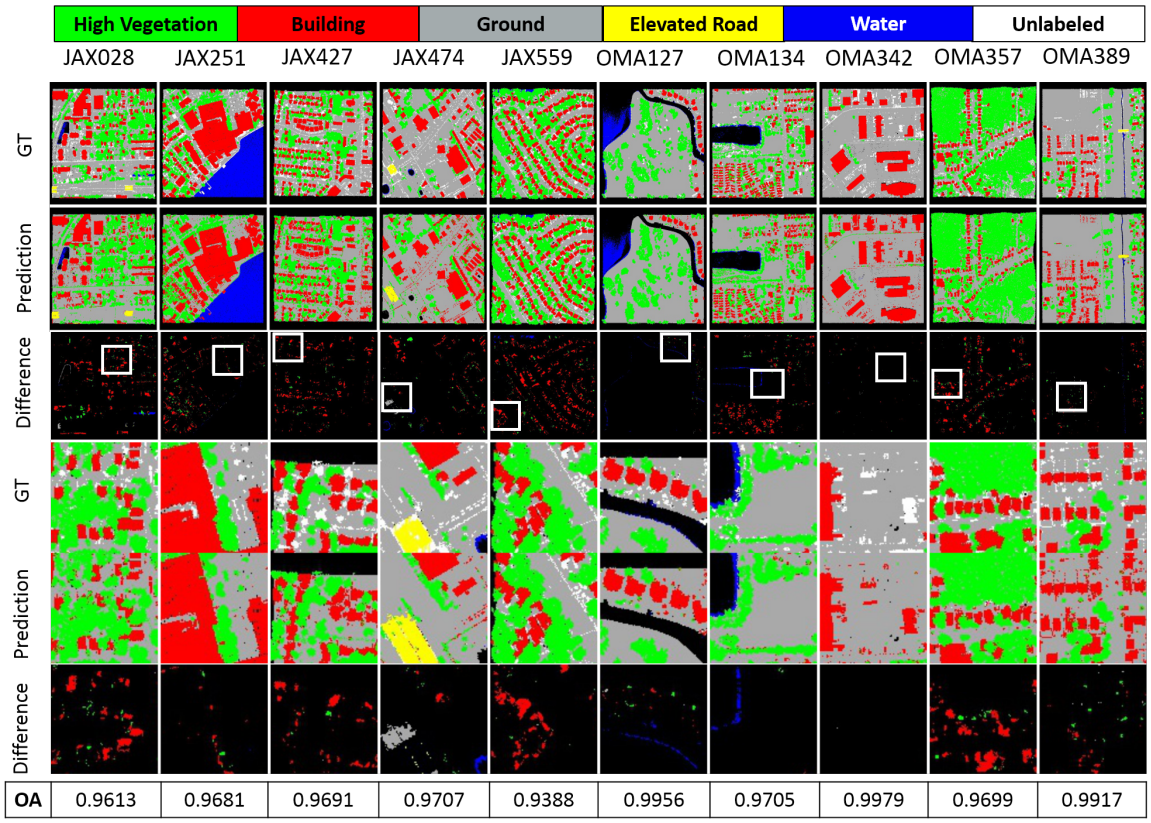


Figure 2.7: Visualization of GLSNet++ results for the ten validation LiDAR datasets from two cities (Jacksonville, FL and Omaha, NE) in the IEEE DFT4 (before demixing) with the overall accuracy (OA) for the full image shown in text at the bottom. Zoomed regions identified by white boxes (third row), illustrate the classification label differences at higher detail.

2.5 Experimental Results

We first performed ablation study of the three structures GLSNet-parallel, GLSNet-cascade and GLSNet++, and then compare our proposed GLSNet++ with the IEEE DFT4 challenge winners and four other state-of-the-art algorithms. Finally, we visualize GLSNet++ predicted semantic segmentation results on COU point clouds. For all the reported results, we abbreviate the class Ground as G, High Vegetation as HV, Building as B, Water as W, and Elevated Road as ER.

2.5.1 Ablation Study

In Table 2.3, the results from [1] all use only 90% training data. For fair comparison, in this current work, we retrain the local branch with 100% training data, and the results are shown in the case of single-stream as Local-branch (100%). Comparing with Local-branch (90%) from [1], the results is better since the training data is increased.

Dual-stream networks outperform single-stream networks: In the case of Dual-stream, GLSNet-parallel uses the newly trained Local-branch (100%) and the global branch is trained from [1]. Relative to Single-Stream, the performance of GLSNet-parallel is better, but the gain on OA over Local branch (100%) is small due to the simple max pooling fusion on the global-local branches. The performance of GLSNet- Cascade is comparable to GLSNet-Parallel in mIOU and OA, but for the individual classes, there are certain differences in accuracy. In general, the dual-stream network performs better than the single-stream networks.

GLSNet++ with demixing block is effective and efficient: 1) **Effectiveness** is shown in Table 2.6, where we compare different designs of global-local fusion in GLSNet++. The case of GLSNet-Parallel+PointNet++ represents the choice of using PointNet++ to replace our demixing block to fuse the two branches. It is observed

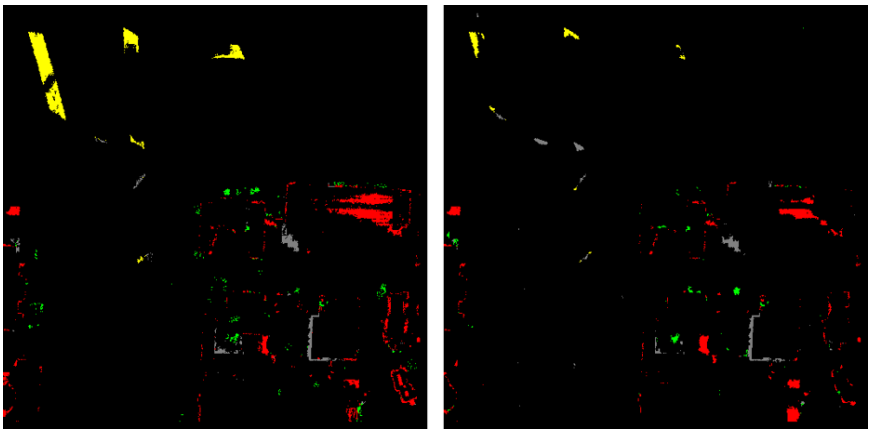


Figure 2.8: Visualization of Graph Demixing-MSMR. Left: Error map before demixing, mIOU: 85.80 OA: 96.90 Right: Error map after demixing, mIOU: 93.82 OA: 98.30.

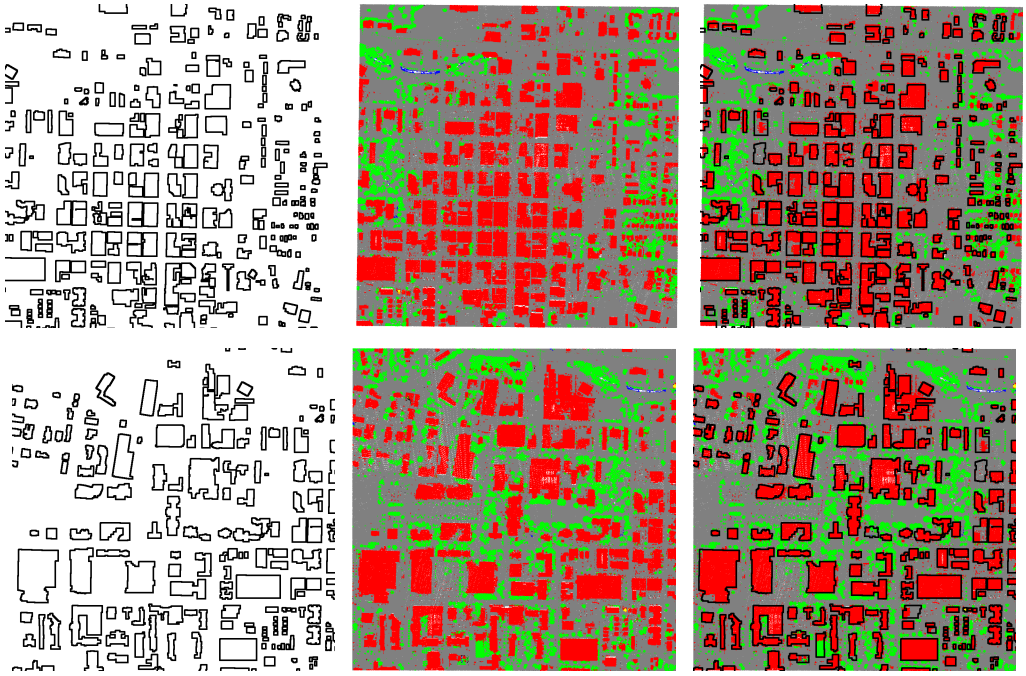


Figure 2.9: Evaluation of building segmentation (red), high vegetation (green) and ground (gray) for COUOrbit28 (first row) and COUOrbit30 (second row). From left to right: Building footprints from OpenStreetMap (OSM), GLSNet++ predicted class labels, predicted results overlaid with OSM building footprints. COU28Orbit accuracy is 82.9% and COU30Orbit is 90.1%.

that by using a simple demixing block, i.e, GLSNet++-MR, we can achieve performance comparable to using a deep PointNet++, which indicates that our demixing block is very effective for fusing the global-local branches. GLSNet++-MR, with its light-weight demixing block, achieves the mIOU performance of 91.90%, which

is 1.4% gain on GLSNet-Parallel and GLSNet-Cascade. 2) **Efficiency** based on the total parameter size of Demixing-MR being around 0.067M, which is about 7% of the parameter size of PointNet++. The total parameter size of Demixing-MSMR is around 0.133M which is about 14% of the parameter size of PointNet++. With the much lighter design, our GLSNet++ with Demixing-MR and Demixing-MSMR overperform the PointNet++ based fusion scheme by 1.5% in mIOU.

It is essential to analyze the influence of grid resolution and sampling size on GLSNet++ accuracy. *Grid resolution:* Grid size is the cell size used in the voxelization procedure for the local branch. For instance, if we use a cell size of 1 meter, then the voxelizated cell will be quantized with 1 meter before the whole point cloud is split into tiles as discussed in implementation of Section IV. In our ablation study, we tried grid sizes of 1 meter and 0.1 meter, where the latter can almost keep the original point cloud resolution of IEEE Data Fusion Challenge with its LiDAR density of 8 centimeter. *Sampling size:* Sampling size is the input size for the network, and tile prediction is voted by the prediction of sampled points. For achieving better performance, we tried two strategies: (a) using cell size of 0.1 m when quantizing the point clouds and (b) increasing the input point size to demixing block, which means that for each tile, we use more points to estimate the tile's semantic segmentation results since the results are interpolated from the preprocessed sets of points. In Table 2.7, we see that using a cell size 0.1m achieves better results than the size of 1.0m since the whole point cloud is better kept and the GLSNet++-MR can use finer details to demix the error areas. We also see that doubling the sampling size to 16,384 achieves better results because the network is provided with more points in training and also in inference.

2.5.2 Comparison with Recent Architectures

We compared our predictions on IEEE DFT4 held-out test as shown in Table 5.1. Our GLSNet++ outperforms PointNet++, SSCN-U, and PointSift, but is slighly lower than UPNet Rank-1 and ASNet Rank-2 deep network performance. UPNet Rank-1 (EN) is an ensemble network model with additional post-processing of the labeled point clouds. Our proposed GLSNet++ is very comparable to UPNet Rank-1 with ensemble classifier fusion.

2.5.3 Visualization of Error Maps and Graph Demixing Block

In order to analyze the effectiveness of the demixing block visually, we first visualize the GLSNet++ predictions in Figure 2.7. It is observed that the errors are mainly around the object boundaries. in Figure 2.8, we provide an example to show the error maps before and after the demixing block, respectively. It is clearly seen that the demixing block helps largely reduce the misclassification errors.

2.5.4 Generalization Performance on COU Point Cloud Dataset

We follow the workflow of Figure 2.6 to evaluate the performance of GLSNet++. From Figure 2.9, we observed that GLSNet++ produced very promising results on the Columbia data, since our building predictions are seen to have largely overlapped with the building footprints. It is worth noting that GLSNet++ was trained only on the IEEE GRSS DFT4 training set (point clouds for Jacksonville, Florida and Omaha, Nebraska), and the only additional information that we used in performing semantic segmentation on the Columbia, Missouri data was a scale factor (the latitude and longitude of point cloud tiles) to calibrate the resolution of the test point clouds relative to the training point clouds.

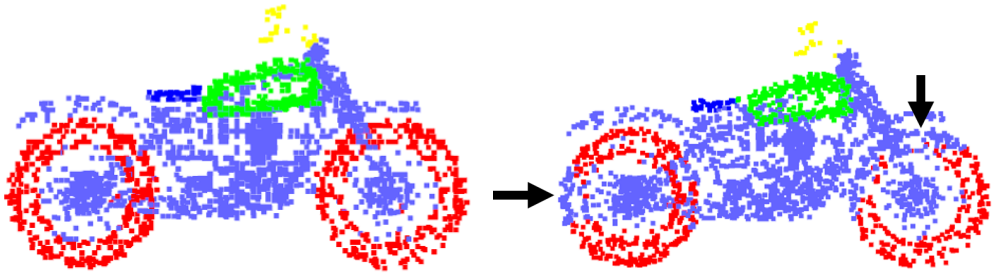
2.6 Chapter Conclusions

City-scale classification of 3D LiDAR data using only the range measurements can be accurately estimated using deep learning architectures. We show that the proposed GLSNet++ architecture with dual global and local feature embedding streams, using a SSCN-UNet and PointNet++ backbones respectively, outperforms the single stream architectures by up to 2.7 percent. Fusing the global and local feature embeddings using a novel lightweight graph convolutional network demixing block further improves the boundary accuracy by about 1.5%, resulting in an overall accuracy of 92% across five common urban classes even with a high degree of class imbalance. GLSNet++ using feature fusion is competitive with the best architecture that uses classifier fusion using an architecture ensemble method. The feature fusion approach enables better generalization to LiDAR data from unseen urban scenes and shows potential for large scale modeling, simulation and navigation applications.

Chapter 3

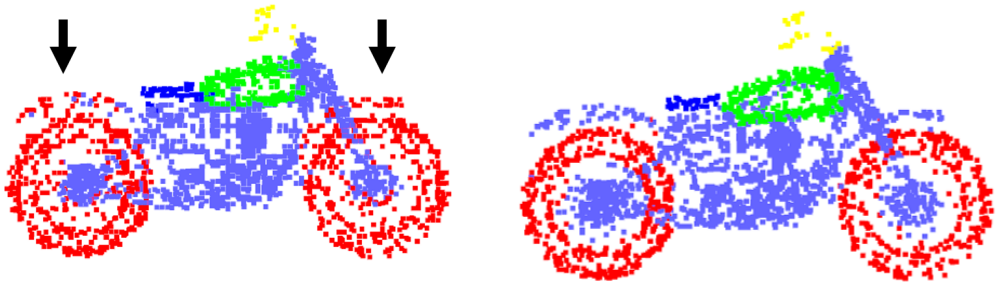
PointGrad: Point Gradient Operator for 3D Point Cloud Segmentation

Geometric shape and structure representation is a critical requirement for point cloud recognition and segmentation. Graph convolution networks offer a powerful approach to encode the geometric relationships between points using edge operators in a latent space projection. However, the commonly used $K-$nearest neighbor search tends to cluster groups of nearby points, losing larger-scale geometric information, besides, previous proposed graph convolution graph operators are not explicitly designed for directional gradient operation. We propose a new graph convolution operator, referred to as the Point Gradient (PointGrad) operator, that encodes point directional tensor gradients into a high-dimensional multiscale tensor space. PointGrad can encode different orders of partial derivatives, capturing multiscale geometric shape and curvature information. Using the PointGrad operator with graph convolutions on scattered irregular point sets captures the salient structural information in the point cloud across spatial and feature scale space, enabling efficient learning. By stacking PointGrad operators, we can construct families of deep PointGrad networks



(a) GroundTruth

(b) PointNet++ [21] (Acc87.11)



(c) DGCNN [33] (Acc88.52)

(d) Ours (Acc95.60)

Figure 3.1: Visualization of different methods on ShapeNet.

(PointGradNet). We demonstrate the effectiveness, efficiency and robustness of the PointGrad operator on several benchmark datasets.

3.1 Introduction

3.1.1 Motivation

3D point cloud semantic segmentation is a challenging task in computer vision, where points are irregular, sparse, and noisy. For point cloud understanding, structural information is vital. For example, in large-scale city point cloud semantic segmentation scenarios, structures of edges, facets, and planar structures are essential for differentiating building and high vegetations. Currently, how to efficiently and effectively capture structural and context information from 3D point clouds remains challenging.

In this work, we are motivated by the fact that points in various quantized direc-

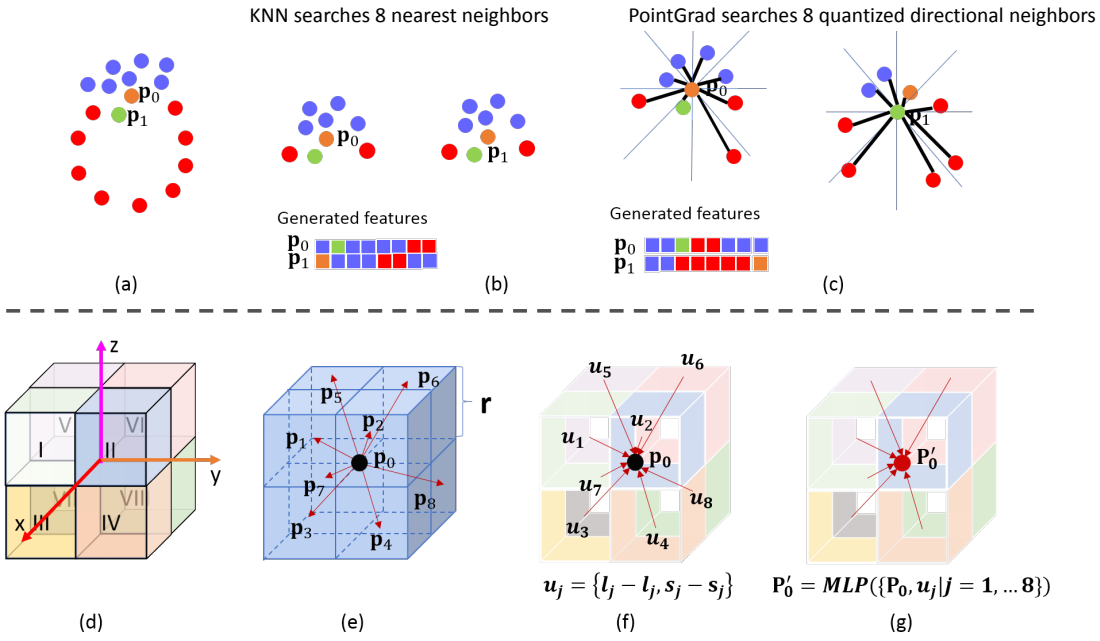
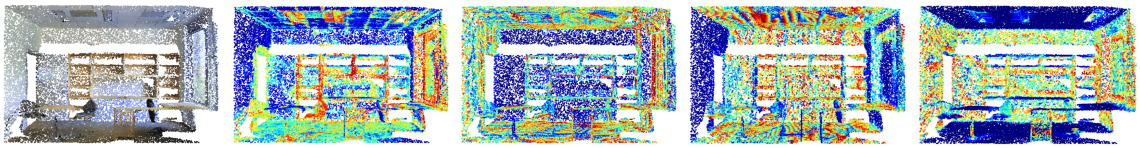


Figure 3.2: Visualization of the PointGrad operator: (a) Original point cloud object with two parts (blue and red) and two labeled points $\mathbf{p}_0, \mathbf{p}_1$. (b) KNN approach – the 8 nearest features based on spatial distance. The resulting features for $\mathbf{p}_0$ and $\mathbf{p}_1$ are shown. The generated features are ordered by distance from $\mathbf{p}_i$. (c) PointGrad approach – searches for NN within 8 octants in 3D (or sectors in the 2D example) for $\mathbf{p}_0$ and $\mathbf{p}_1$. The generated features are ordered by counter-clockwise directions from $\mathbf{p}_i$. (d) The 8 search regions visualized as octants. (e) Central reference point $\mathbf{p}_0$ and its 8 directional neighbors within a fixed radius, visualized using $2\times2\times2$ grid neighbors within radius r. The 8 direction vectors to NN in each octant are $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_8$. (f) PointGrad computes tensor gradients along each direction $u_j, j = 1, ..., 8$. (e) Using a multi-layer perceptron (MLP), we update the point tensor $\mathbf{p}_0$ to $\mathbf{p}'_0$ by encoding each directional gradient tensor.



(a) Point cloud (b) $\left\|\partial_{\{R,G,B\}} f\right\|$ (c) $\|\partial_x f\|$ (d) $\|\partial_y f\|$ (e) $\|\partial_z f\|$

Figure 3.3: Illustration of PointGrad tensor operator. A sample indoor scene point cloud from the S3DIS dataset. A 3D point cloud with point representation $f(x, y, z) = \{x, y, z, R, G, B\}$. The different tensor derivative operators: color vector derivative for image edge structures, partial derivative in each spatial direction for planar spatial structures.

tions preserve the geometry structure of point clouds, and directional gradients can describe the changes of points in a representation space that are highly relevant to point cloud understanding. We design a graph gradient operator, referred to as PointGrad, to encode the point directional gradient of different orders in high dimensional tensor space. Such an operator can capture the diverse scaled structural and contextual information in point clouds efficiently and effectively. In Figure 3.1, we present the motorbike part segmentation results of representative methods PointNet++ [21] and DGCNN [33]. The DGCNN designs a dynamic graph where the vertices in the graph are searched by the K-nearest neighbor(KNN) euclidean distance of the feature vector. Representative methods such as PointCNN [32], PointConv [44], PointwiseCNN [17] use K-nearest neighbor euclidean distance of spatial distance for local graph construction, and these methods design various network modules to encode information of the point clouds. When using KNN to search neighbors to construct the graph of feature space, the points of KNN will fall in the same quadrant sometimes, which leads to the loss of geometry information. PointSift proposes to search the points from directions in grids and designs an axis-aligned operator to encode the orientation information of points.

We propose PointGradient (PointGrad), which differs from these methods in two aspects: (1) Graph Construction: instead of using KNN search or search from directions in grids, PointGrad proposes to search in quantized directions. Therefore, searching directions in grids is one sub-case of quantized directional search. (2) Graph Computation: instead of complex network module design, PointGrad proposes to encode the quantized directional gradients into feature space. The PointGrad operator has two essential properties: (1) neighbors in search space are spread in various directions, which in our design, we propose quantized directional nearest neighbor search. (2) encoding features in each direction efficiently and effectively, which in our design, we propose the gradient feature of different orders to capture each direction space

structure and context representations effectively.

Our rationale of PointGrad is explained as the following: (1) graph construction: using quantized directional search (2) graph computation: encoding directional gradients. From the perspective of graph construction, we compare the quantized directions search with KNN search. In the first row of the Figure 3.2, we show that KNN searches eight nearest neighbors and using PointGrad quantized directional search eight directional nearest neighbors (each quantized quadrant covers 45°). In this way, PointGrad quantized directional search covers long-range structural and contextual information in various directions. In the first row of Figure 3.2, the KNN searched neighbors are clustered in the upper part, while the PointGrad directional searched neighbors are spread in various directional spaces. Besides, with PointGrad directional search, the generated features maps are more robust since the generated feature maps in KNN are arranged with the nearest distance, while in the PointGrad directional search, the feature maps are sorted in directional space order. In the second row of Figure 3.2, we show that the PointGrad will integrate features of each directional gradient in updating point features. We will discuss this process in section 3. From the perspective of graph computation, we illustrate the reason for designing directional gradients. For a simple case, the partial derivatives along the directions on RGB can easily delineate the edges of point clouds, since RGB colors change rapidly around edges, and therefore the gradients of RGB colors have high responses on edges. The partial derivatives along the x, y, z directions on RGB delineate the planes of point cloud, since RGB colors are consistent on the plane. Furthermore, noise effects such as slow variations in color, intensity, etc., can be reduced in the gradient space to allow the representation to concentrate more on the changes itself, where directional gradients of different orders can describe these representations in an effective way. In Figure 3.3, we show an example of a point cloud from S3DIS [43] which is a dataset composed of a set of indoor scene point clouds, as an example. We apply our

PointGrad module to compute $L2-norm$ of partial derivatives on the input space: x, y, z, RGB of 8 quadrant directions, and show the resulted features of the point cloud in (b), (c), (d), (e). It is evident that after partial derivative operators, we can easily detect x directional planes, y directional planes, and z directional planes since the partial derivatives are approximately zeros on these planes.

3.1.2 Contribution

Most existing methods attempt to learn the structural information by different graph convolutions. In our work, we explicitly extract directional gradients from 3D point cloud and design algorithms and network architectures accordingly. Through our proposed point directional derivative operators and network module PointGrad, we map 3D point clouds to the space of point-neighborhood directional gradients (PNDG) to efficiently encode the structures of edges, planer patches, curvature structures and etc, and contextural information of the spatial space and high dimensional tensor feature space in deep networks. We further embed PointGrad within three deep network structures to design structure-aware networks, and name the network family as PointGradNet. To perform the PNDG feature encoding in hierarchical feature space and different scales, we design several versions of PointGrad to encode gradient flow in alternative ways, forming three representative deep network structures within PointGradNet for 3D point clouds. We demonstrate the effectiveness, efficiency, and robustness of our directional derivative-based network representations in 3D point cloud semantic segmentation tasks.

In summary, we propose a network module PointGrad to encode point directional gradient features of different orders in 3D point clouds. This directional difference encoding has a strong power in representing crucial structural features such as edges and planar patches for point cloud segmentation. We design a PointGrad architecture family that embeds the PointGrad module in a set of networks to encode

multiple-order and multiple-scale point directional differences, and we designate it as a structure-aware point directional derivative network. We demonstrate the efficiency, effectiveness, and robustness of our novel PointGrad module on several point cloud semantic segmentation benchmarks. PointGrad achieves competitive performance consistently in various scales of point cloud segmentation tasks, including city-scale point cloud segmentation, indoor scene semantic segmentation, and object part segmentation, confirming that our proposed PointGrad can effectively capture structure information of different scales.

This chapter is organized as follows. Section 4.2 reviews the related work in point cloud segmentation; Section 3.3 describes our PointGrad module and PointGrad networks family in details; Section 4.5 presents quantitative results of our algorithm. We conclude this chapter in Section 4.6.

3.2 Related Work

In this section, we briefly review the deep learning methods on 3D point cloud segmentation. According to data types of point clouds, recent methods of deep learning on point clouds can be categorized as voxelization -based [25, 45, 46, 47, 48, 49], multi-view-based [50, 51, 52] and graph-based [20, 21, 53, 54, 55, 56, 34, 30, 31, 57].

Here we mainly discuss graph-based methods for point cloud segmentation since our method is related to graph-based methods. PointNet [20] proposes a network to learn on unordered point clouds by point-wise encoding and aggregation through global max-pooling. PointNet++ [21] proposes a hierarchical neural network to capture geometric details of point set. Kernel point convolution (KPConv) [36] proposes to directly operate on point clouds without intermediate representation. PartNet [35] proposes a top-down recursive decomposition strategy for shape segmentation. Superpoint Graphs (SPG) [34] adaptively partitions point clouds by contextual relationship

encoded superpoint graphs, and it embeds every superpoint with a shared PointNet. The semantic labels of the superpoints are predicted from the PointNet embedding of each current superpoint and its spatially neighboring superpoints. PointCNN [32] uses the layer of $\mathcal{X}$-Conv instead of multi-layer perceptron to exploit certain canonical ordering of points. More recently dynamic graph CNN (DGCNN) [33] suggests an alternative grouping method to ball query that is used in PointNet++ [21]: namely, Euclidean distance based KNN between feature vectors. PointSIFT [23] proposes an orientation-encoding unit by providing eight crucial orientation representations of points. Pointwise convolution neural network (PointwiseCNN) [17] proposes a pointwise convolution operator for each point, which bins its nearest neighbors into kernel cells before convolving with kernel weights. PCNN [58] module first queries the neighboring points and then bins them into grids, and multiple layers processing is performed on each grid to first get local grid features and then apply convolution. Existing methods of graph-based methods propose different methodologies to capture the point cloud information. In contrast to these efforts, our proposed PointGrad explicitly computes point directional gradients in an octant system with different orders to encode point cloud structure information. In Section 3.3.6, further details will be discussed to contrast of our PointGrad with the above discussed graph-based neural network methods.

3.3 Proposed PointGrad Operator

Given a set of 3D points, the task of semantic segmentation is to predict the semantic label of each point. In object part segmentation, the task is to segment different parts of a object. In indoor scene point cloud segmentation, the task is to segment furniture in the room. Our PointGrad is designed to be able to capture the different structures efficiently and effectively in these point clouds.

In this section, we will 1) define the directional derivatives 2) introduce the point directional derivative operations 3) describe the point directional derivative deep modules 4) present the PointGrad network family 5) discuss the detailed differences between PointGrad and other methods.

3.3.1 Directional Derivative

The directional derivative $\nabla_{(\mathbf{u}_m)} f(x_0, y_0, z_0)$ is defined as the rate at which the function $f(x, y, z)$ changes at a point (x_0, y_0, z_0) in the direction $\mathbf{u}_m$. It is a vector form of the conventional derivative, defined as,

$$\begin{aligned} \partial_{\mathbf{u}_m} f &= \lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{u}_m) - f(\mathbf{x})}{h} \\ \|\mathbf{u}_m\| &= \sqrt{u_x^2 + u_y^2 + u_z^2} = 1 \end{aligned} \tag{3.1}$$

where $\mathbf{u}_m$ is the unit directional vector.

Another representation for modeling directional changes, simpler than derivatives, is the finite directional difference:

$$\Delta_{h\mathbf{u}_m} f = f(\mathbf{x} + h\mathbf{u}_m) - f(\mathbf{x}) \tag{3.2}$$

where the finite directional difference is $h\mathbf{u}_m$. This is the basis for the PointGrad tensor gradient operator used in this work, since point sets are discrete sets and we use finite difference approximations in the rest of the chapter.

3.3.2 Point Directional Derivatives

For 3D point clouds, we define the point directional derivative operation. A point set with N points is defined as $\mathbf{P} = \{\mathbf{p}_1, ...\mathbf{p}_N\} \in \mathbb{R}^D$. For each point $\mathbf{p}_i = \{\mathbf{L}_{\mathbf{p}_i}, \mathbf{s}_{\mathbf{p}_i}\}$, $\mathbf{L}_{\mathbf{p}_i} = \{x, y, z\}$ is the 3D geometry coordinate of the point , and $\mathbf{s}_{\mathbf{p}_i}$ is the feature

vector of $\mathbf{p}_i$. For each point $\mathbf{p}_i$ with feature vector $\mathbf{s}_{\mathbf{p}_i}$, the form of $\mathbf{s}_{\mathbf{p}_i}$ depends on the input space.In LiDAR point cloud, $\mathbf{s}_{\mathbf{p}_i} = \{x, y, z, I, R\}$, where I is the LiDAR return intensity, and R is the LiDAR return number. In RGB-D point cloud, $\mathbf{s}_{\mathbf{p}_i} = \{x, y, z, R, G, B\}$ or $\mathbf{s}_{\mathbf{p}_i} = \{R, G, B\}$ depending on if the location $\{x, y, z\}$ is used as features; In networks, $\mathbf{s}_{\mathbf{p}_i}$ is the feature vector of $\mathbf{p}_i$ in certain layer.

We compute a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ representing local point cloud structure, where $\mathcal{V} = \{p_j, j = 1, ..., n\}$ and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ are vertices and edges, respectively. In the simplest case, we construct $\mathcal{G}$ as the K-nearest directional neighbor graph of $\mathbf{p}_m$ in $\mathbb{R}^D$. Since our module is designed to compute the point directional gradients in point cloud, the two properties of gradient, i.e., direction and magnitude, are both encoded in the graph by graph construction and edge function. We first introduce a K-nearest directional neighbor graph construction procedure which helps encoding the gradient direction of points, and we then introduce the edge function to encode the gradient magnitude of directional points.

The graph is constructed by the K-nearest directional points search. We define the K nearest directional points of the center point $\mathbf{p}_0$ as its K neighbors in the $G \times G \times G$ grid or quantized space. In $G \times G \times G$ grids, the set of these K directional neighboring points is $\mathcal{V}$ of the center point $\mathbf{p}_0$ ($K = G^3$). In quantized space as we show in Figure 3.2 first row (c), each nearest directional neighbors are searched in quantized directional space within radius r (In Figure 3.2 first row (c), the quantized space is 45 degrees in each quadrant). As an example, in Figure 3.2, we use $2 \times 2 \times 2$ grid with 8 directional derivatives within a radius r to illustrate our point directional derivative computation. Figure 3.2 (a) represents the 8 directions for each point. Each quadrant represents one direction. In Figure 3.2 (b), the center point is $\mathbf{p}_0$, and its 8 nearest directional neighbor set is $\mathcal{V} = \{\mathbf{p}_1, ..., \mathbf{p}_8\}$. If in one direction within radius r, there is more than one point in that direction octant, the point with the nearest euclidean distance to $\mathbf{p}_0$ is selected. If along a direction there exists no point

in the radius r, then $\mathbf{p}_0$ is duplicated as that directional neighbor, setting the gradient to 0 in this direction to represent an empty direction.

First-order Directional Derivative and Difference The first-order directional derivatives are used to describe feature tensor changes along the eight spatial directions. The point directional derivative operator in the direction $\mathbf{u}_m$ $(m = 1, 2, 3, ...8)$ is approximated by using finite differences for computational efficiency as,

$$\partial_{\mathbf{u}_m} f \approx \Delta_{h\mathbf{u}_m} = f(\mathbf{L}_{\mathbf{p}_m}) - f(\mathbf{L}_{\mathbf{p}_0}) = \mathbf{s}_{\mathbf{p}_m} - \mathbf{s}_{\mathbf{p}_0} \tag{3.3}$$

with $\mathbf{u}_m$ being the unit direction vector from $\mathbf{p}_0$ to $\mathbf{p}_m$, $\mathbf{L}_{\mathbf{p}_m} - \mathbf{L}_{\mathbf{p}_0} = h\mathbf{u}_m$ is the directional gradient vector from $\mathbf{p_0}$ to its neighboring points $\mathbf{p}_m$, within a specified ball of radius $r = \|\mathbf{u}_m\| = \sqrt{u_x^2 + u_y^2 + u_z^2}$. Note that the neighboring points in the set are scattered and irregularly spaced within this radius r ball centered at $\mathbf{p}_0$.

High-order Directional Derivative and Difference For the high order point directional derivative, we still consider its 8 directions. If the n^{th}-order $(n = 2, 3, ...)$ directional derivative is computed at the $\mathbf{u}_m$ direction, which is the same direction as its $(n-1)^{th}$ derivative, then we have

$$\begin{aligned} \Delta^{(n)}_{\mathbf{u}_m} &= \Delta^{(n-1)}_{\mathbf{u}_m} f(\mathbf{L}_{\mathbf{p}_0} + h\mathbf{u}_m) - \Delta^{(n-1)}_{\mathbf{u}_m} f(\mathbf{L}_{\mathbf{p}_0}) \\ &= \Delta^{(n-1)}_{\mathbf{u}_m} f(\mathbf{L}_{\mathbf{P_m}}) - \Delta_{\mathbf{u}^{(n-1)}_m} f(\mathbf{L}_{\mathbf{p}_0}) \end{aligned} \tag{3.4}$$

Otherwise, if the n^{th}-order directional derivative is computed along other directions, for example, $\mathbf{v}_k$ $(k = 1, 2, ...8), \mathbf{v}_k \neq \lambda\mathbf{u}_m$, then we have

$$\begin{aligned} \Delta^{(n)}_{\mathbf{u}_m\mathbf{v}_k} &= \Delta^{(n-1)}_{\mathbf{u}_m} f(\mathbf{L}_{\mathbf{p}_0} + t\mathbf{v}_k) - \Delta^{(n-1)}_{\mathbf{u}_m} f(\mathbf{L}_{\mathbf{p}_0}) \\ &= \Delta^{(n-1)}_{\mathbf{u}_m} f(\mathbf{L}_{\mathbf{p}_N}) - \Delta^{(n-1)}_{\mathbf{u}_m} f(\mathbf{L}_{\mathbf{p}_0}) \end{aligned} \tag{3.5}$$

3.3.3 PointGrad Module Design

Our PointGrad modules are designed to compute point directional derivatives in the neural network. We design several modules of PointGrad to compute different orders of directional derivatives for encoding different structural information. We denote the module that encodes the first-order point directional derivative as PointGrad-1, that encodes the second-order point directional derivative as PointGrad-2, ... , and that encodes n^{th}-order directional derivative as PointGrad-n. In this section, we first describe PointGrad-1, which is the basic module in the *PointGrad* Network family. The PointGrad-2 and PointGrad-n are designed by recursively applying PointGrad-1 in the network. We construct different types of networks by using the different versions of PointGrad, which form the PointGrad family discussed below.

First-order Directional Derivative PointGrad-1

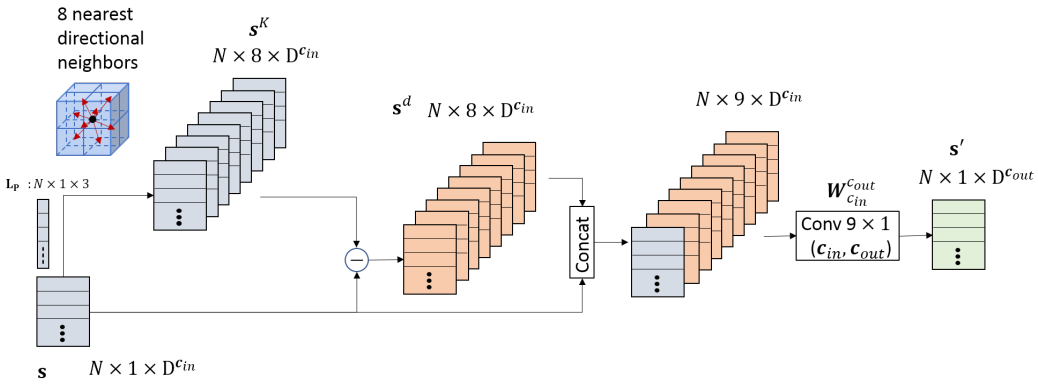


Figure 3.4: Design of PointGrad-1 for encoding first directional derivative with $\mathbf{p}_m$ being the center point.

We first describe the computation and implementation of first-order point directional derivative. For each layer in the network, the input point set is $\mathbf{P}$ and the output feature point set is denoted as $\mathbf{P}'$. For each point $\mathbf{p}_m$ in $\mathbf{P}$, with $\mathbf{p}_m = \{\mathbf{L}_{\mathbf{p}_m}, \mathbf{s}_{\mathbf{p}_m}\}$, we apply the PointGrad operator shown in Figure 3.4, then the output point is $\mathbf{p}'_m = \{\mathbf{L}_{\mathbf{p}_m}, \mathbf{s}'_{\mathbf{p}_m}\}$. We find the 8 nearest directional neighbors of $\mathbf{p}_m$ in $2 \times 2 \times 2$

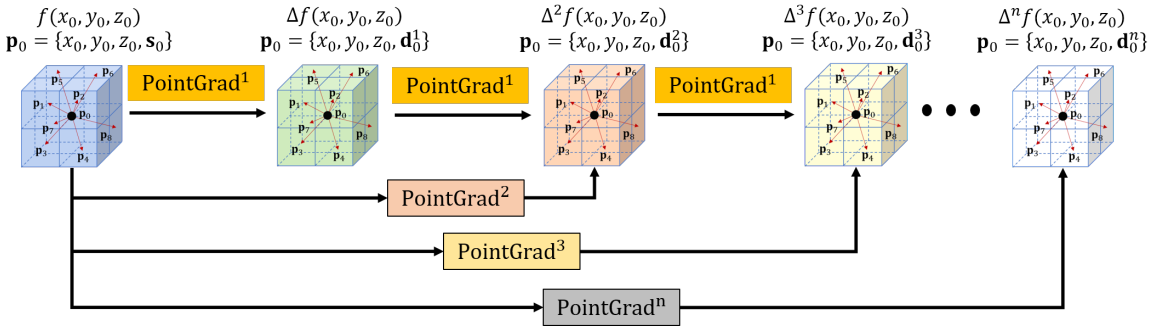


Figure 3.5: Design of n^{th} order PointGrad-n for encoding high-order gradients for modeling differential shape.

grid with the grid length r, with the set denoted as $\mathcal{V}_{\mathbf{p}_m}^K$, where $K = 8$. Each point in $\mathcal{V}_{\mathbf{p}_m}^K$ is represented as $\mathbf{p}_j = \{\mathbf{L}_{\mathbf{p}_j}, \mathbf{s}_{\mathbf{p}_j}\}, j = \{1, ..., 8\}$. Therefore, for the 8 directions, the directional difference in each direction $\mathbf{u}_j$ of the point $\mathbf{p}_m$ is $\Delta_{\mathbf{u}_j}(\mathbf{p}_m) = \mathbf{s}_{\mathbf{p}_j} - \mathbf{s}_{\mathbf{p}_m}, j = \{1, ..., K\}$. Let θ represent the parameters of PointGrad-1, and let f_θ be the mapping function that maps the input feature set $\mathbf{s}$ to the output feature set $\mathbf{s}'$. The updated feature set by PointGrad-1 is computed as,

$$\mathbf{s}' = f_\theta(\mathbf{s}) = \mathbf{W}_{c_{in}}^{c_{out}}([\mathbf{s}, \mathbf{s}^d]) \tag{3.6}$$

where $\mathbf{s}^d = \{\Delta_{\mathbf{u}_j}\} = \mathbf{s}_{\mathbf{p}_j} - \mathbf{s}_{\mathbf{p}_m}, j = \{1, ..., 8\}$, and $[\,\cdot\,]$ denotes the operation of concatenating the original feature $\mathbf{s}$ with the derivative feature $\mathbf{s}^d$. The output $\mathbf{s}'$ is computed by convolving the concatenated feature with $\mathbf{W}_{c_{in}}^{c_{out}}$, which is a set of 1×9 convolutional kernels with the input channel size c_{in} and output channel size c_{out}. $\mathbf{W}_{c_{in}}^{c_{out}}$ is learned to aggregate information from the original features and their derivatives. The implementation of PointGrad-1 is shown in Figure 3.4.

High-order Directional Derivative PointGrad-n

In addition to the first-order directional derivative, we also design modules to compute high-order directional derivatives for feature representation. We can deduce that high-order directional derivatives PointGrad-n can be implemented as in Figure 3.5, which

recursively applies PointGrad-1 to compute the high-order point feature directional derivatives. Another version of encoding mixture directional derivatives combines different orders of directional derivatives to enrich the representation of PointGrad. The usage of these modules will be demonstrated in our experiments.

3.3.4 Gradient Backpropagation in PointGrad

Here we compare the gradient computation in each layer of *PointGrad* with that of a regular neural net layer operation. For a regular layer, without PointGrad, the gradient is computed as,

$$\frac{\partial \mathbf{s}^{'}}{\delta \mathbf{W}_{c_{in}}^{c_{out}}} = \mathbf{s} \tag{3.7}$$

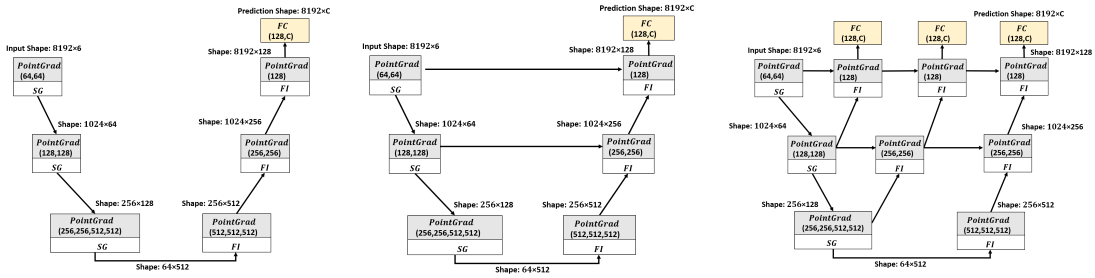
In PointGrad, the gradient is computed as,

$$\frac{\partial \mathbf{s}'}{\partial \mathbf{W}_{c_{in}}^{c_{out}}} = \mathbf{s} + \sum_{j=1,2,...K} \Delta_{\mathbf{u}_j} \tag{3.8}$$

Obviously, the operation of *PointGrad* is different from that of the regular PointNet layer, where beside of the input information $\mathbf{s}$, our *PointGrad* module additionally encodes the neighboring set directional difference $\sum_{j=1,2,...K} \Delta_{\mathbf{u}_j}$ into the backward gradient flow.

3.3.5 PointGradNet: PointGrad Network Family

In the last subsection, we have designed several types of PointGrad modules to compute gradient flow, where a sequence of PointGrad could encode higher-order directional derivatives for feature representations. Naturally, a deep neural network can be employed for this through stacking PointGrad modules. Therefore, we design a



(a) PointGrad-Deep (b) PointGrad-U (c) PointGrad-U++

Figure 3.6: PointGrad operator can be used in many different deep network families like the three shown above.

family of PointGrad Networks by exploring three representative gradient flows, 1) PointGrad-Deep: a deep network structure as in Figure 3.6 (a) that implements the PointGrad$-n$ by using the Figure 3.5 to encode high-order point directional derivative. 2) PointGrad-U: a deep network structure as in Figure 3.6 (b) that implements the mixture PointGrad$-n$ by using the concatenate different orders of PointGrad to encode the mixed order point directional derivative. 3) PointGrad-U++: a deep network structure as in Figure 3.6 (c), which is a dense version of PointGrad-U by using mixed orders of point directional gradients.

In integrating PointGrad with the three types of the networks, we borrow from PointNet++ [21] the sampling and grouping (SG) module and the feature interpolation (FI) module for downsampling and interpolation on point set, and adapt them for PointGrad. For details of SG and FI, please reference to [21]. To facilitate describing our network structures, we use () to represent the numbers of channels of the point features. By inserting the SG and FI modules into different locations of the network, the receptive fields are enlarged, and by inserting the FI module into different layers of the network, the downsampled points are interpolated back to the original point set size.

1) *PointGrad-Deep* is a deep network which implements the PointGrad$-n$ in combination with SG and FI. We stack the PointGrad to form the PointGrad$-n$ as a deep

Table 3.1: Comparison to existing graph convolution deep network architectures with graph edge operators.

Method	Neighbor Search	Graph Edge Function	Learnable Parameters	Feature Space Encoding
PointNet [20]	–	$f_\Theta(\mathbf{p}_i, \mathbf{p}_j) = f_\Theta(\mathbf{p}_i)$	Θ	Point feature manifold
PointNet++ [21]	KNN / Ball of {x,y,z}	$f_\Theta(\mathbf{p}_i, \mathbf{p}_j) = f_\Theta(\mathbf{p}_j)$	Θ	Neighbor feature mapping
MoNet [59]	KNN / Ball of $\{x, y, z\}$	$f_{\theta_m, w_n}(\mathbf{p}_i, \mathbf{p}_j) = \theta_m \cdot (\mathbf{p}_j \odot g_{w_n}(u(\mathbf{p}_i, \mathbf{p}_j)))$	θ_m,w_n	Adaptive Hadamard
PCNN [58]	KNN / Ball of $\{x, y, z\}$	$f_{\theta_m}(\mathbf{p}_i, \mathbf{p}_j) = \theta_m \cdot (\mathbf{p}_j \odot g(u(\mathbf{p}_i, \mathbf{p}_j)))$	θ_m	Fixed Hadamard
PointCNN [32]	KNN / Ball of $\{x, y, z\}$	$f_\Theta(\mathbf{p}_i, \mathbf{p}_j) = f_{\theta_c}(f_{M_\theta}(p_j) \times [f_{M_\delta}(\mathbf{p}_j), \mathbf{p}_i])$	M_θ,M_δ, θ_c	Coordinate mapping
PointConv [44]	KNN / Ball of $\{x, y, z\}$	$f_{M_d, M_\theta}(\mathbf{p}_i, \mathbf{p}_j) = (f_{M_d}(D_{\mathbf{p}_j}) \cdot \mathbf{p}_j) f_{M_\theta}(\mathbf{p}_j)$	M_d, M_θ	Point density mapping
DGCNN [33]	KNN of feature space distance	$f_\Theta(\mathbf{p}_i, \mathbf{p}_j)$	Θ	Adaptive feature mapping
PointSIFT [23]	Cube Directional search of $\{x, y, z\}$	$f_\Theta(\mathbf{p}_i, \mathbf{p}_j) = f_{M_\theta}(\mathbf{p}_i \oplus f_{\theta_o}(\mathbf{p}_j))$	M_θ,θ_o	Axis-aligned convolution
PointGrad	Quantized Directional search of $\{x, y, z\}$	$f_\Theta(\mathbf{p}_i, \mathbf{p}_j) = f_\Theta([\mathbf{p}_i, \mathbf{p}_j - \mathbf{p}_i])$	Θ	Tensor directional gradient operator

graph neural network. 2) *PointGrad-U* is a mixture point gradient network, by using skip links and concatenation operation to realize a mixture of directional derivatives. PointGrad-U only concatenates the gradient features at the corresponding levels so that different orders of derivatives at the same scale are added to the forward and backward flows in the network. We call it PointGrad-U because it is a U-shape network. 3) *PointGrad-U++* is designed to encode mixture-ordered point gradient in different scales in the network, as well as encode gradients of both shallow and deep features. We construct the PointGrad-U++. Besides, the PointGrad-U structure has the problem that there is a large gap of gradient crossing a link between encoder and decoder. By inserting a PointGrad module in each link between the encoder and decoder, the gradient gap can be reduced, and more orders of directional derivatives are involved in backward gradient flow to encode rich information. The above design is similar to UNet++ network structure, therefore we name it as PointGrad-U++. These three network designs of structures are quantitatively evaluated through experiments on 3D point cloud benchmarks, and the results are discussed below in Section 4.5.

3.3.6 Comparison with Existing Methods

PointGrad is related to graph convolutional neural networks. Here, we compare our PointGrad module design with several prototype graph CNNs for point cloud learning

including: (a) PointNet or PointNet++ layer operation, (b) MoNet [59] and Pointwise Convolutional Network (PCNN), PointCNN: Convolution on $\mathcal{X}$-Transformed Points, PointConv [44], (c) graph edge convolution in DGCNN [33], and (d) PointSIFT [23]. These methods are discussed from the perspectives of graph construction, edge function and learnable parameters. These major categories of approaches are summarized in Table 3.1 and discussed in further detail next.

For a point set, a directed graph is constructed as $G = (V, E)$, here we use the simplest case where the neighbor set $V = 1, ..., K + 1$ which is composed of a center point $\mathbf{p}_i$, and its K neighbor point $\mathbf{p}_j$. We define the updated $\mathbf{p}'_i$ to be the output of a graph convolution operation defined as,

$$\mathbf{p}'_i = f_\Theta(\mathbf{p}_i, \mathbf{p}_j) \tag{3.9}$$

(a) PointNet & PointNet++: PointNet can be regarded as a fixed weight graph representation for local point averaging or max-pooling operations, represented as the local graph operation, $f_\theta(\cdot)$. PointNet++ uses max-pooling to obtain local region information, which queries nearest neighbors based on geometric coordinates and uses multiple layers to encode local region information *excluding* $\mathbf{p}_i$, where the edge function is $f_\theta(\mathbf{p}_i, \mathbf{p}_j) = f_\theta(\mathbf{p}_j)$, and the information is aggregated by max operation, as discussed in [31].

(b) Recent graph networks such as MoNet [59], PCNN [32], PointCNN and PointConv, are focused on the design of different graph convolution-type operations.

- MoNet: the graph structure of MoNet in a local "pseudo-coordinate system" $\mathbf{u}$ is defined as a M-component Gaussian mixture model. The edge function is computed in $f_{\theta_m, w_n}(\mathbf{p}_i, \mathbf{p}_j) = \theta_m \cdot (\mathbf{p}_j \odot g_{w_n}(u(\mathbf{p}_i, \mathbf{p}_j)))$, where g is a Gaussian kernel and $\odot$ is the elementwise (Hadamard) product, and w_n are learnable parameters of the Gaussians (mean and co-variance), and θ_m are the learnable

filter coefficients.

- PCNN: PCNN can be regarded as a special case of MoNet with g as predefined Gaussian functions, and the learnable parameters w_n are removed, thus the edge function of PCNN is simplified as $f_{\theta_m}(\mathbf{p}_i, \mathbf{p}_j) = \theta_m \cdot (\mathbf{p}_j \odot g(u(\mathbf{p}_i, \mathbf{p}_j)))$.

- PointCNN: PointCNN proposes a pointwise convolution, and the edge function can be written as $f_\Theta(\mathbf{p}_i, \mathbf{p}_j) = f_{\theta_c}(f_{M_\theta}(\mathbf{p}_j) \times [f_{M_\delta}(\mathbf{p}_j), \mathbf{p}_i])$. It first lifts local coordinates of neighbor set into high dimensional features by f_{M_δ}, where f_{M_θ} is a learnable transform function of coordinates, $[\cdot]$ is the concatenate operation, and f_{θ_c} is the convolution operation.

- PointConv: PointConv proposes a density re-weighted convolution, as $f_{M_\theta, M_d}(\mathbf{p}_i, \mathbf{p}_j) = f_{w_d}(D_{\mathbf{p}_j})\mathbf{p}_j f_{M_\theta}(\mathbf{p}_j)$, where $D_{\mathbf{p}_j}$ is a kernelized density estimation function, f_{M_d} is a multi-layer perceptron (MLP) for adapting the density scale, and $f_{M_\theta(\mathbf{p}_j)}$ is a MLP with learnable weights M_θ for re-weighting the neighbor set density weighted point feature $f_{M_d}(D_{\mathbf{p}_j})\mathbf{p}_j$.

(c) DGCNN: Dynamic Graph CNN uses different graph edge functions for f_Θ. The main difference between DGCNN and the above methods is that the constructed DGCNN graph is dynamic because its neighbor sets are selected based on distances in feature space. Thus, as the feature space changes, so does the graph; while other methods construct graphs using distances based on point coordinates and so the graph structure remains fixed.

(d) PointSIFT [23] proposes to encode information of different orientations, and the edge function is defined as $f_\Theta(\mathbf{p}_i, \mathbf{p}_j) = f_{M_\theta}(\mathbf{p}_i \oplus f_{\theta_o}(\mathbf{p}_j))$, where f_{θ_o} is an orientation-encoding convolution along X, Y, Z, $\oplus$ is add or concatenation, and f_{M_θ} is an MLP operation, and M_θ is the set of multi-layer perceptron parameters.

(e) PointGrad: In comparison with all of the above operations, our approach has two major differences: (1) unlike the above local feature encoding methods, we

Table 3.2: Benchmarks for 3D point cloud semantic segmentation

Dataset	Data Type	Scenario	Task	# of Classes	Feature
ShapeNet [2]	3D Model	Object	Part Segmentation	50	{X,Y, Z}
S3DIS [60]	RGBD	Indoor Scene	Semantic Segmentation	13	{X,Y, Z, R, G, B}
IEEE DFT4 [12]	Airborne LiDAR	City-scale	Semantic Segmentation	5	{X,Y, Z, I, R}

are rooted in point directional difference, and our PointGrad explicitly computes the first-order directional difference feature to encode the structure information of planes, edges, or corners; (2) a sequence of *PointGrad*, i.e., PointGrad$-n$, essentially is a high order directional difference encoding module, and it can encode different orders of point directional difference information through different concatenating strategies. To the best of our knowledge, we are the first to propose a network module that can explicitly derive the point direction difference to encode structure information of a point set in a graph manner.

3.4 Experiments

3.4.1 Datasets and Implementations

We evaluate the proposed PointGrad network family on three tasks: 1) city LiDAR point cloud segmentation: capture large structure blocks such as buildings, trees, bridges, etc, 2) indoor scene segmentation: capture structures of indoor objects such as tables, chairs, etc., 3) Object part segmentation: capture small structures such as planes of table, legs of stools, etc. The details of these types of datasets are summarized in Table 3.2, and one example from each of these datasets are also shown in Figure 3.7. We choose these tasks from different scenarios to show that our algorithms can capture different scales of structure information including city blocks structures, indoor furniture structures, and object structure.

Large-scale city point cloud semantic segmentation 2019 IEEE Data Fusion Challenge Track4 Dataset is a city-scale LiDAR point cloud semantic segmentation dataset

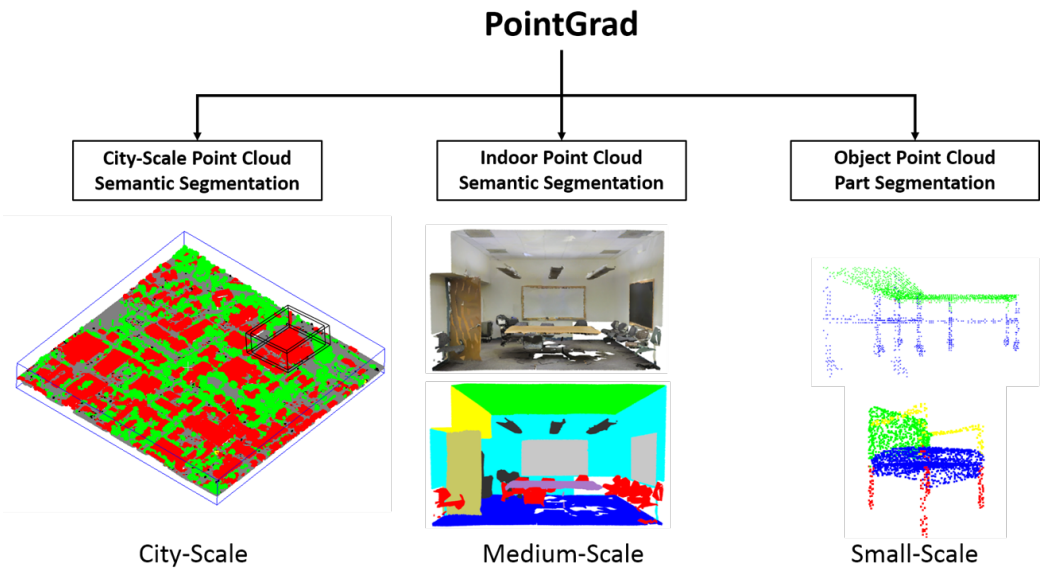


Figure 3.7: PointGrad on semantic segmentation benchmarks of different scales

which covers two cities: Omaha, NE and Jacksonville, FL. The range of LiDAR point clouds is around $500m \times 500m$ with LiDAR accuracy 80 centimeters. Each LiDAR point is provided 5-dimension input $\{X, Y, Z, Intensity, Return\ Number\}$, and labeled by six classes: Ground (G), High Vegetation (HV), Building (B), Water (W), and Elevated Road (ER). There are 110 point clouds for training, 10 validation point clouds, 10 point clouds for held-back testing on the test server.

Indoor scene segmentation is an important and challenging task in computer vision. We evaluate the proposed method on benchmark *S3DIS* [60]. The S3DIS dataset contains 3D scans of 271 rooms from 6 areas, with a total of 13 categories. Each point has RGB features. For experiments on S3DIS, we follow the training/testing split in [61] to measure the generalization ability of existing methods under training and testing setting of Area 5. The *ScanNet* dataset [62] contains 1,513 scenes captured by the Matterport 3D sensor. The points are annotated in 20 categories and one background class. We follow the official training/testing split 1201/312 scans from [62] in this work.

Shape part segmentation is a challenging task in fine-grained shape segmentation.

We evaluate the proposed method on *ShapeNet*[2] dataset which contains 16,881 shapes with 16 categories, and totally are segmented in 50 parts. We follow the data split in [21]. We random sample 2048 points as the input, and the one-hot encoding label of object class is concatenated. We report the mean IoU(mIoU) over all classes and all instances, respectively.

Our PointGrad is implemented in Tensorflow [63] framework. All the reported results of PointGradNet was trained in a total of 200 epochs using Adam optimizer [64] with an initial learning rate of 0.001. For experiments on IEEE DFT4, we followed the baseline pre-processing and post-processing procedure provided by the challenge baseline method [12], and we chose the best model on the validation set to test on the held-out test set. For experiments on ShapeNet, S3DIS, we follow the pre-processing procedure and evaluation of PointNet.

3.4.2 Ablation Study

Table 3.3: With/Without directional difference operation, Gradient (Direction + Magnitude)

Method	Encode Direction	Encode Magnitude	G	HV	B	W	ER	OA	mIOU
PointDiff		✓	96.99	95.00	86.41	89.27	79.55	97.09	89.45
PointRelation	✓		97.55	95.77	88.74	92.63	79.21	97.60	90.78
PointGrad	✓	✓	97.77	95.58	89.37	93.57	83.45	**97.76**	**91.95**

Encoding direction and magnitude both play important roles in semantic segmentation. The gradient is composed of two components: direction and magnitude of difference between two vectors. Our PointGrad is designed to encode these two components. To analyze the importance of encoding both direction and magnitude, we decouple the experiments into three variants: (1) PointDiff: instead of keeping the direction in tensor as in PointGrad, we randomly permute the direction neighborhood in each point neighbor set, and otherwise keep the difference operation the

Table 3.4: IoU accuracy for different variations of our proposed 3D PointGrad segmentation deep architecture on 2019 IEEE DFT4 (US3D) test dataset, for five classes: Ground (G), High Vegetation (HV), Building (B), Water (W), and Elevated Road (ER).

Method	**G**	**HV**	**B**	**W**	**ER**	**OA**	**mIOU**
PointGrad-Deep	97.60	95.07	88.49	92.40	83.35	97.56	91.38
PointGrad-U	97.99	95.50	90.12	93.66	84.81	97.90	92.41
PointGrad-U++	97.61	96.07	89.18	94.55	88.37	97.78	93.16

same with PointGrad. With this operation, the point direction is not consistent in each layer, which destroys the point directional property in point directional difference. (2) PointRelation: in this setting, we do not apply difference operation after obtaining the directional neighbor set, but directly apply 1×9 convolution on the neighbor set, which means that the difference property is destroyed in this setup. (3) PointGrad: this is our design of using both direction and magnitude of directional difference. We apply these three versions of feature encoding methods in Figure 3.6. The PointDiff version gets the lowest performance since it loses the feature orientation information for structure understanding. Due to the loss of orientation, even if the difference operation is applied, it could not encode higher-order difference information by stacking the layers together. Besides, it only encodes local region's information of neighbor set for each point in each layer without modeling the relation of points. The PointRelation version only encodes the orientation information but loses the difference information, and therefore it could not encode the magnitude of gradient. In contrast to PointDiff and PointRelation, PointGrad encodes both the direction and magnitude information of the directional differences, which yields the best performance.

Embeding PointGrad in deep network architectures is beneficial. We establish a family of PointGrad operators for different deep learning network architectures including Deep (no skipped connections), UNet, UNet++ structured networks using Point-

Table 3.5: Ablation study on shape part segmentation benchmark ShapeNet [2]

Method	OA	class mAcc	class mIOU	instance mIOU
PointGrad-Deep	93.64	87.05	81.37	83.03
PointGrad-U	94.11	88.05	82.46	83.60
PointGrad-U++	94.42	88.48	83.50	84.74

Grad module as discussed previously (see Figure 3.6. We verify how well the PointGrad operator works in different architectures across two tasks, including city-scale point cloud semantic segmentation dataset IEEE DFT4, and object part segmentation ShapeNet as in Table 3.4 and Table 3.5. With the skip connections, the PointGrad-U achieves higher performance than Point-Deep, and PointGrad-U++ achieves best performance since it benefits from multi-supervision training (PointGrad-U++ uses three losses for supervision) and adds more orders of point directional differences in network information aggregation.

3.4.3 Comparison to Existing Methods

Analysis on large-scale city point cloud segmentation. We first evaluate the proposed PointGrad on IEEE DFT4(US3D). The challenge for such a task is how to effectively encode large-scale differences of objects such as building vs. bridge, where a bridge is usually much larger than a building at a city scale. In Table 3.6, we compare our method with existing methods. In comparison with those graph-based methods, our PointGrad achieves the best performance. Comparing with the Rank-1 method DP-Net, our model surpasses their single model performance (their winning performance in the challenge was based on ensemble models and post-processing). We conjecture that the superior performance of our model over previous graph methods is due to the ability of PointGrad in explicitly computing the point directional differences, which is essential in encoding structure information. For example, the structures of buildings

Table 3.6: Class IoU accuracies for different 3D segmentation architectures on 2019 IEEE DFT4 (US3D) test dataset, for five classes: Ground (G), High Vegetation (HV), Building (B), Water (W), and Elevated Road (ER). * results are from [3]

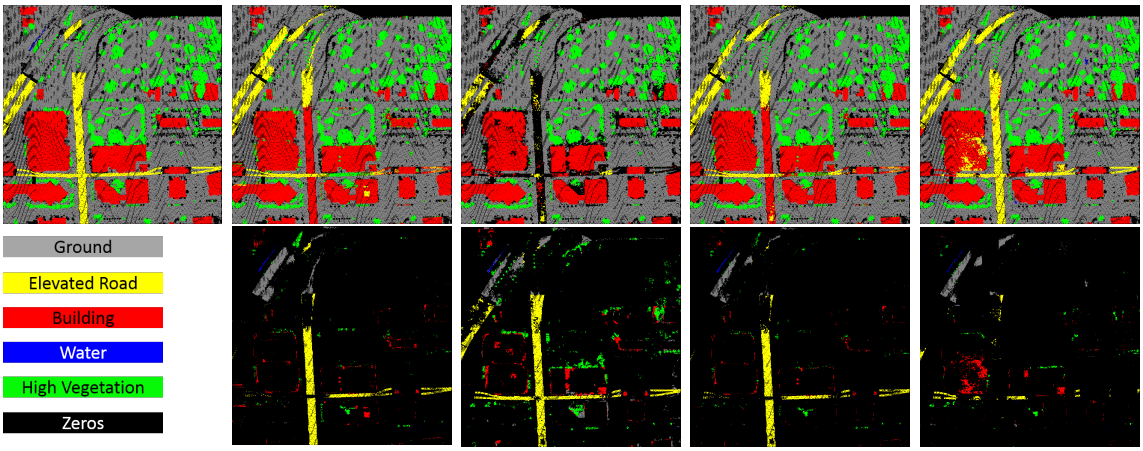
Method	**G**	**HV**	**B**	**W**	**ER**	**OA**	**mIOU**
SSCN-U	96.95	90.96	82.10	95.93	62.54	96.29	85.69
PointNet++*	95.4	95.2	83.7	88.4	76.6	96.3	87.9
PointNet++(MSG)*	96.8	94.9	85.8	93.1	74.8	96.9	89.1
DGCNN*	96.4	96.2	86.3	96.2	48.7	96.9	84.8
PointCNN*	96.7	95.4	88.3	88.3	83.5	97.3	90.4
PointSIFT*	97.4	96.1	88.4	91.5	79.3	97.5	90.6
PointConv*	97.6	95.5	89.1	92.1	76.3	97.6	90.1
GLSNet [1]	97.56	93.94	87.56	95.64	77.90	97.37	90.52
DPNet [40]	97.87	95.79	89.98	94.94	86.48	97.94	93.01
DPNet+grid map [40]	98.5	96.0	92.4	95.3	86.9	98.4	93.8
DPNet+grid map+model fusion [40]	98.74	96.11	93.31	96.54	89.04	98.57	**94.55**
PointGrad-U++	97.61	96.07	89.18	94.55	88.37	97.78	93.16

and trees can be more discriminate when observing gradient features: buildings are always associated with planes, and their gradient features would show many zeros, while trees display irregular structures, and their gradient features would be more random.

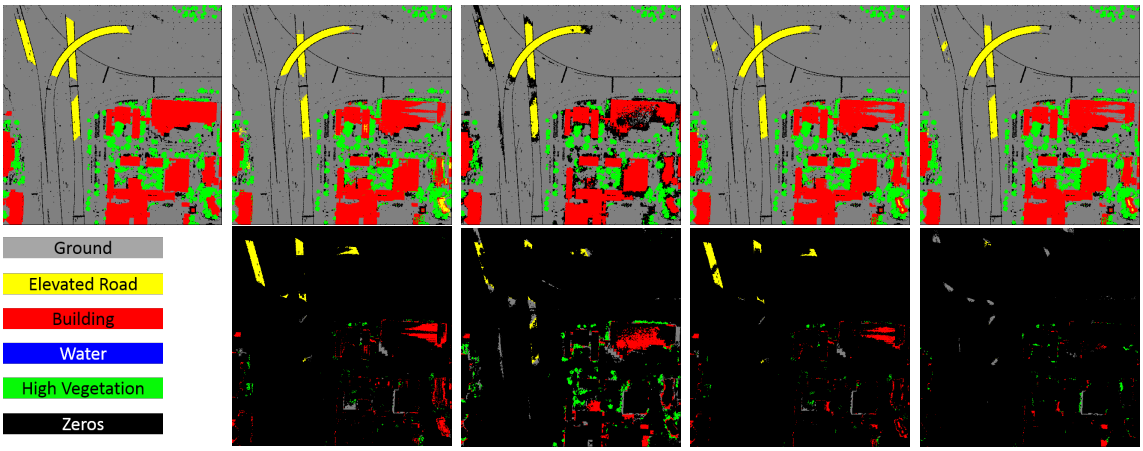
In Figure 3.8, we visualize the segmentation outcomes of the representative methods on DFT4 dataset. PointNet++ is the most commonly referenced methods in point cloud segmentation, and SSCN shows impressive performance of segmentation on several segmentation benchmarks. GLSNet [1] is a two-branch network for combining global and local structural and contextual information for large-scale point cloud segmentation. The error maps clearly show that our proposed PointGrad has fewer errors than the other methods, especially in bridge class.

Analysis on Indoor Scene Semantic Segmentation In Table 3.7 and Table 3.8, we compare our method with existing methods on indoor scene segmentation on S3DIS and ScanNet. Our PointGrad shows competitive performance on both datasets.

Analysis on Object Part Segmentation In Table 3.2, we compare our method with existing methods on ShapeNet, PointGrad shows its competitive performance on ob-



(a) JAX215 Groundtruth (b) PointNet++ (87.18) (c) SSCN (87.46) (d) GLSNet (88.55) (e) Ours (89.48)



(f) OMA243 Groundtruth (g) PointNet++ (91.93) (h) SSCN (89.97) (i) GLSNet (93.56) (j) Ours (95.19)

Figure 3.8: Visualization of different methods on IEEE DFT4 dataset.

Table 3.7: Comparisons with related work on S3DIS Area 5.

	OA	mIOU
PointNet [20]	–	48.98
SPGraph [34]	86.38	58.04
SegCloud [61]	85.91	58.27
PCCN [65]	–	58.27
PointCNN [32]	85.91	57.26
PointGrad (Ours)	86.17	56.95

Table 3.8: Comparisons with related work on ScanNet dataset.

	Publication	mIOU
OctNet [48]	CVPR17	18.1
ScanNet [62]	CVPR17	13.5
PointNet++ [21]	NIPS17	38.28
PointSIFT [23]	ArXiv18	41.5
RSNet [22]	CVPR18	39.35
TCDP [66]	CVPR18	40.9
PointGrad (Ours)		45.76

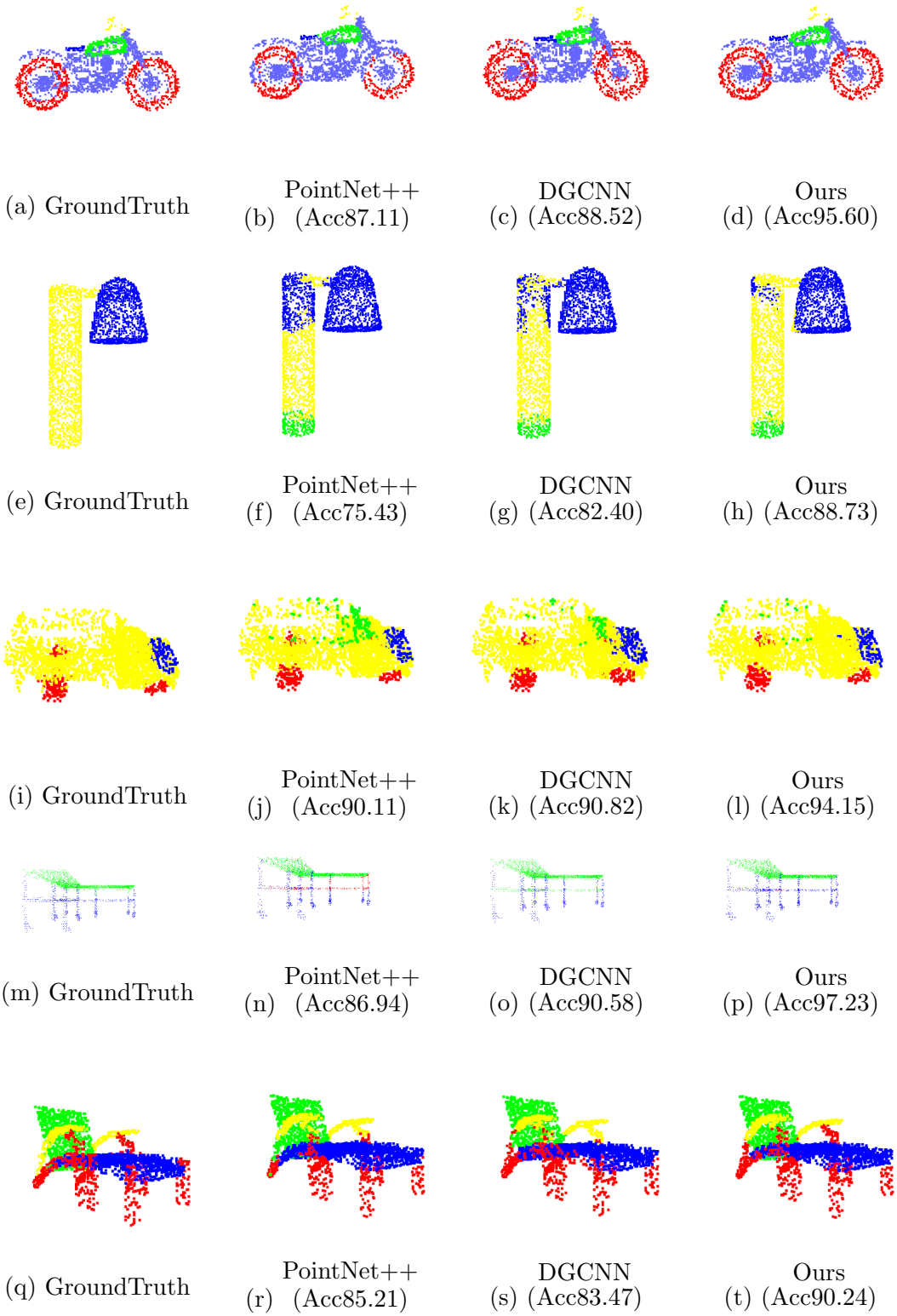
Table 3.9: Comparison to existing methods on ShapeNet

	ShapeNet	
Method	class mIOU	instance mIOU
ShapeNet [2]	-	81.4
Kd-Net [49]	77.3	82.3
PointNet [20]	80.4	83.7
RS-Net [22]	81.4	84.9
SCN [67]	81.8	84.6
PCNN [58]	81.8	85.1
SPLATNet [68]	82.0	84.6
KCNet [55]	82.2	84.7
DGCNN [33]	82.3	85.1
RS-CNN [69]	84.0	**86.2**
PointNet++ [21]	81.9	85.1
SyncCNN [54]	82.0	84.7
SO-Net [70]	80.8	84.6
SpiderCNN [19]	82.4	85.3
3DmFV-Net [71]	81.0	84.3
SSCN [15]	83.3	85.98
PointCNN [32]	**84.6**	86.14
SGPN [72]	82.8	85.8
PointConv [44]	82.8	85.7
PointGrad (Ours)	83.5	85.1

ject part segmentation. We visualize the PointGrad results as well as those of PointNet++ and DGCNN on ShapeNet in Figure 3.9 for visually comparison.

3.5 Chapter Conclusions

We propose a PointGrad operation to capture the structure information of point sets through encoding point directional gradient of different orders in tensor feature space. We also construct deep PointGrad network family by stacking the PointGrad module to empower the network feature representation. We demonstrate that such a design is effective and robust in capturing various scales of point cloud structure information on several typical semantic segmentation benchmarks. In the future, we plan to transfer the PointGrad into more tasks such as point cloud reconstruction, point cloud completion, etc., to benefit from the strong representation power of PointGrad.



(a) GroundTruth (b) PointNet++ (Acc87.11) (c) DGCNN (Acc88.52) (d) Ours (Acc95.60)

(e) GroundTruth (f) PointNet++ (Acc75.43) (g) DGCNN (Acc82.40) (h) Ours (Acc88.73)

(i) GroundTruth (j) PointNet++ (Acc90.11) (k) DGCNN (Acc90.82) (l) Ours (Acc94.15)

(m) GroundTruth (n) PointNet++ (Acc86.94) (o) DGCNN (Acc90.58) (p) Ours (Acc97.23)

(q) GroundTruth (r) PointNet++ (Acc85.21) (s) DGCNN (Acc83.47) (t) Ours (Acc90.24)

Figure 3.9: Visualization of different methods on ShapNet.

Chapter 4

MDXNet for Few-Shot Learning and Zero-Shot Learning

The human visual system has an ability for effortless generalization across visual categories and specialized domains. People are able to learn new classes and variations of existing classes from visual primitives (VPs) by using the semantic generalization of latent attributes. This cross-domain transfer learning likely occurs through multi-domain visual knowledge integration from widely seen, highly repetitive common everyday scenes and objects, encoded and extended for new visual tasks. This perceptual capability is highly successful even in extremely specialized domains consisting of many rare classes with few instances. In this work, we emulate the human visual system by leveraging multi-domain knowledge about visual primitives, like texture, object-appearance, and shape, learned from natural or universal image sources, and applying the composite knowledge to more specialized learning tasks in the application domain, such as aerial scene recognition. We use a cross-domain "few-shot" learning approach with just a few labeled application domain examples by leveraging knowledge from multiple VP domains. Unlike conventional few-shot learning, our method does not use any prior training data from the same application domain. We

propose the Multi-domain Explainable Latent Attribute Network (MDXNet), which first learns embeddings for perceptual VPs only from the outside domains to capture explainable perceptual VP latent attributes and then integrates them for application tasks. The proposed MDXNet is easily extended to zero-shot learning. Extensive experiments on different application domains demonstrate that our MDXNet provides a unified and effective approach for cross-domain few-shot learning and zero-shot learning tasks.

4.1 Introduction

Human visual system has an ability for effortless generalization across visual categories and specialized domains. People can learn new classes and variations of existing classes from visual primitives (VPs) by using the semantic generalization of latent attributes and transfer previously learned knowledge from one or more outside domains to a new domain by seeing only a few instances in the new domain. This cross-domain knowledge transfer occurs through information integration from widely seen, highly repetitive everyday scenes, and objects of multi-domains. This perceptual capability is highly successful even in extremely specialized domains consisting of many rare classes with few instances. For example, people can easily transfer the knowledge from routine recognition tasks on faces, common everyday objects, and vegetation to more specialized tasks of construction equipment recognition, aircrafts, biological species, or medical pathology slides, where the latter tasks often require expert visual recognition and knowledge.

AI has achieved a wide range of successes across different domains by learning from large-scale data. However, learning to recognize new classes with limited data is still challenging, as AI still falls short of human ability in knowledge association. Few-shot learning is an approach along this line that targets rapid generalization to

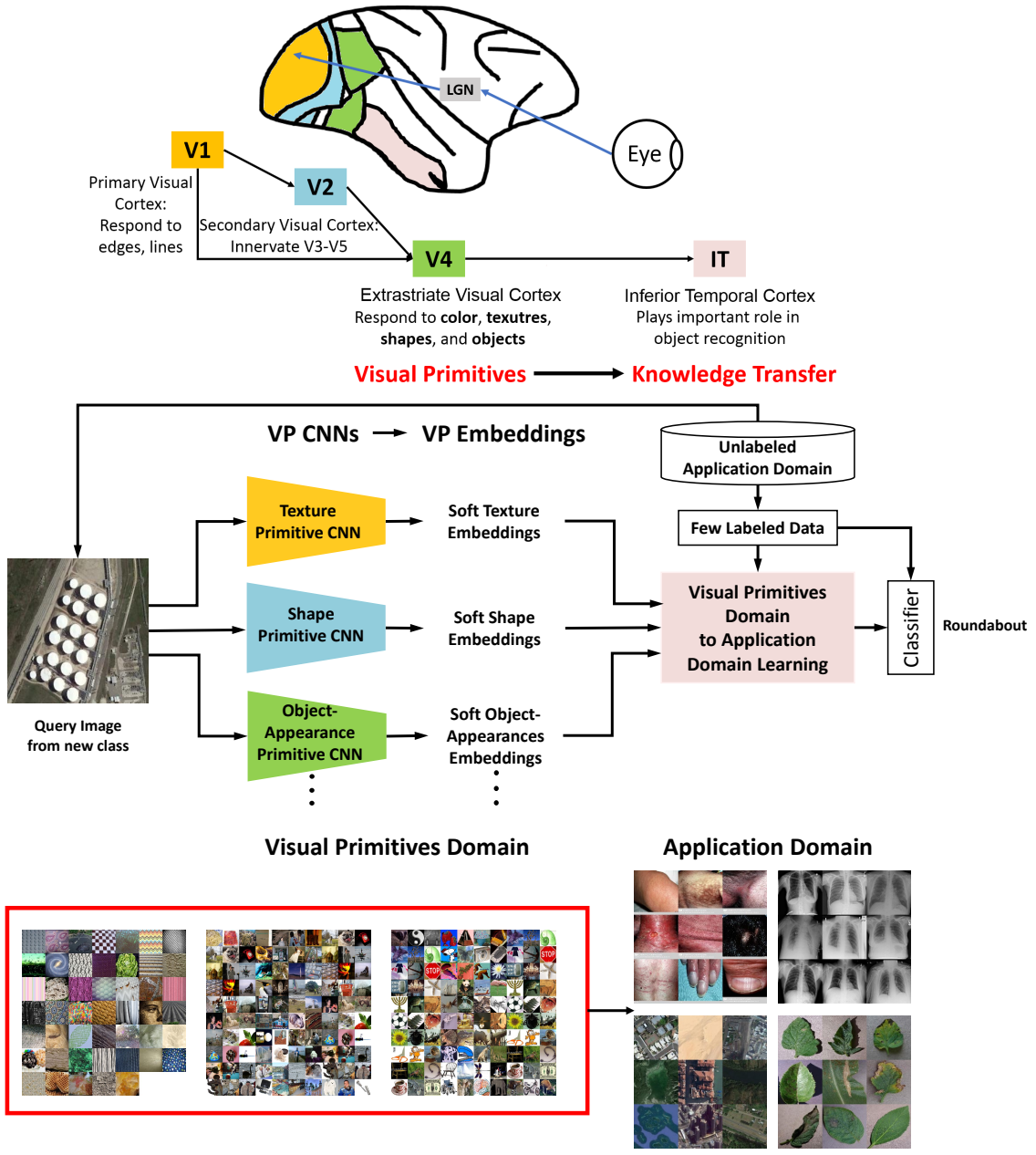


Figure 4.1: (a) Human visual system of object recognition [4]. (b) MDXNet visual primitive embedding and transfer learning network that can be applied to many application domains which is inspired and mimics human visual system.

new classes with limited supervision by incorporating prior knowledge.

The above observations inspire us to consider the following possibility: can we design a deep learning system for efficient cross-domain recognition tasks that emulates the Human Visual System (HVS)? This system transfers knowledge across different

Figure 4.2: Our MDXNet discovered top visual word embeddings from each VP domain (using ResNet18 backbones), for *storage tank, roundabout* and *snowberg* classes from the remote sensing scene classification dataset NWPU-RESISC45 [5].

domains through multi-domain visual knowledge integration from widely seen, highly repetitive common everyday scenes and objects, and encode and extend them for new visual tasks. Figure 4.1 (a) illustrates the human visual system's ventral stream, which is involved with the object and visual identification and recognition. Here we mainly show the primary visual cortex (V1), secondary visual cortex (V2), extrastriate visual cortex (V4), and inferior temporal cortex (IT), which are the primary cortices involved in the ventral stream. In HVS, V1 mainly processes edge and line information of images. After V1, the information goes through V2, which mainly divides information into different visual cortices V3-V5. V4 usually responds to color, texture, shapes, and objects, while some parts of V3 respond to orientation, depth, and motion (much remains unknown about V3). V5 usually responds to motion and stereo disparity [4]. Based on responses from V4, the IT cortex plays a vital role in object recognition by combining different responses from V4. **The textures, edges and shapes, appearances are basic primitives of HVS for recognizing new objects.**

Natural everyday images are rich in varieties of visual primitives, including tex-

ture, color, edge, shape, depth, etc. Although embedding features of these visual primitives can be learned by deep networks, such representation knowledge is often difficult to learn directly in a complex application domain, such as aerial images, and skin diseases, by using a single deep network. In such application domains, although visual primitive features are useful low-level descriptors, there may not be enough instances of these primitives in training data, especially when there are many classes (e.g. hundreds to thousands) with a wide range in the number of samples per class, including rare classes (e.g. from tens to thousands). In computer vision tasks, visual information has been mostly derived from natural images, but there exists visual information from other sources that may be informative but have not yet been utilized. Besides, unlike natural images where rich knowledge has been learned and available, in data sources such as aerial images of remote sensing, such knowledge is still unavailable or not adequately derived. For example, attributes such as color, texture, shape are informative descriptions of image objects, but they have not been derived for aerial scene image data.

In the current work, we propose to exploit the rich information in natural images for use in application domains that lack informative descriptions of image objects or scenes, mimicking human's ability of pattern or knowledge association. Specifically, we propose to extract embedding features of visual primitives from multiple outside-domain data sources and learn to fuse them for a new application domain through a few examples of the latter. Along this line, we design MDXNet, shown in Figure 4.3 (b). MDXnet is composed of two components: visual primitives learning component and visual primitive domain to new application domain learning component. In our system, visual primitive domain knowledge mimics HVS visual cortices of V1, V2, V4. and the domain adaptation process mimics the IT cortex. MDXNet, therefore, focuses on learning such visual primitives from natural images and leverage the knowledge thus learned to benefit recognition in new tasks.

4.1.1 Approach

As shown in Figure 4.1 (b), we propose a novel network architecture for extracting and integrating multiple latent attributes to tackle the few-shot learning problem for an application domain, where the latent attributes are from different knowledge domains that provide a rich set of visual primitives. This network design is based on our belief that the knowledge needed for recognizing classes in a new application domain can benefit from association with knowledge learned from multiple outside domains. The visual primitives that we propose to emulate human visual primitives are general in nature and they include: (i) **Texture VPs** for describing images with human interpretable texture attributes. (ii) **Shape VPs** for describing images with shape or edge attributes. (iii) **Object-Appearance VPs** for describing images with general object-appearance attributes.

The general applicability of the three types of VPs can be appreciated from the example in Figure 2 pertaining to aerial scene recognition. Although these three domains may not seem to be closely related to aerial images, visual primitives can be associated with aerial scene images very informatively. Figure 4.2 shows three aerial scene images, as well as the corresponding prediction label outputs of the three outside domains. It is seen that the aerial image of "Storage Tank" is associated with "Studded" and "Bumpy" in the texture domain and with "Accordion" and "Remote Control" in the object- appearance domain. The aerial image of "Roundabout" is associated with "Spiralled" and "Crystalline" in the texture domain and with "Electric fan" and "Hard Disk" in the object- appearance domain. Also, the aerial image "Snowberg" is associated with "Matted" and "Smeared" in the texture domain and with "Skunk" and "Valley" in the object-appearance domain. In the shape domain, the outlines or shapes of "Storage Tank" is similar to "iPod" and "Pencil Box," "Roundabout" is similar to "Snorkel" and Carousel, and "Snowberg" is similar to "Coral Reef" and "Ant". It is interesting to see that for each aerial scene, the pre-

dicted texture primitives appropriately describe its textural spatial layout and surface properties, the predicted object-appearance primitives properly provide its structural appearances, and the predicted shape primitives also adequately relate to its outlines or shapes. This example illustrates the capability of our approach in mimicking human knowledge association for new visual recognition tasks, with the prior knowledge derived from natural images of textures, objects, and shapes.

4.1.2 Contribution

Novelty Our first novelty lies in constructing a novel visual primitive space that can be used generally across different application domains, which is different from existing methods that require application domain datasets for training. Our second novelty is the design of the network architecture for MDXNet, which includes the selection of its backbone network and the out-of-domain training and domain-adaptive integration of the VPs. Our proposed MDXNet is the first approach to cross-domain few-shot learning that employs visual primitives motivated by human visual system to provide high-level descriptions of new images, where the VPs are learned from three domains of natural images and are describable in visual words at the *concept level*.

Furthermore, we provide experimental evidence using extensive benchmarks from various real world tasks. Our experimental results demonstrate that mimicking HVS in such a clean manner through texture, shape and object-appearances domain VPs can successfully transfer knowledge from natural images to other application domains, and effectively alleviate large domain discrepancy in cross-domain few-shot learning. In particular, our approach significantly outperforms state-of-the-art few shot learning methods on the real world remote sensing challenge dataset.

In summary, our contributions are three-fold. (1) We propose emulating HVS to leverage out-of-domain knowledge of three different types of visual primitives from natural images for new application domains. (2) We design an effective network archi-

tecture for visual latent attribute aggregation (3) We deliver state-of-the-art performance with good generalization on cross-domain few-shot and zero-shot benchmarks. Source codes of our methods will be released upon publication of this work.

The subsequent part of this chapter is organized as the following: section 4.2 reviews the related work in few-shot learning and zero-shot learning; section 4.3 describes in details our MDXNet method in cross-domain few-shot and zero-shot learning; section 4.4 describes datasets and implementation details; section 4.5 presents both quantitative and visualization results of our method; we conclude this work and discuss future directions in section 4.6.

4.2 Related Work

4.2.1 Few-Shot Learning

Conventional Few-Shot Learning (FSL)

Many methods have been proposed to tackle few-shot classification problems, such as *initialization based* (learning to fine-tune) [73], *distance metric based* (learning to compare) [74, 75, 76, 77, 78], *hallucination based* (learning to augment) [79, 80], and domain adaptation [81, 82, 83]. Some recent works combine semantic descriptions, e.g., word embeddings [84] or sentence descriptions, as additional information to help few-shot image recognition [85]. As discussed in the survey of FSL [86], existing works mainly deal with image recognition dataset [87, 75, 88, 89, 90, 76] and character recognition dataset [75, 91, 76, 92]. A new benchmark is proposed in [93] which is a dataset of datasets constructed from different sources for evaluating the generalization ability of few-shot classifiers on more natural and realistic tasks. Conventional few-shot learning methods have three major weaknesses: (1) only tackle the problem in the same target domain (i.e., training and testing classes are from the same domain); (2)

require lots of hand labeled data in the target domain; (3) have difficulty generalizing to new domains when the images are from non-natural image domains, such as aerial scenes and biomedical images.

Cross-Domain Few-Shot Learning

In cross-domain few-shot learning, base and novel classes are drawn from different domains, and the class label sets are disjoint. [88] proposes to utilize web data to help few-shot learning, but training and testing are still both natural and at the same scale. [94] proposes benchmarks of cross-domain few-shot learning. Recent works on cross-domain few-shot learning include analysis of existing meta-learning approaches in the cross-domain setting [95], specialized methods using feature transform to encourage learning representations with improved ability to generalize[96], and works studying cross-domain few-shot learning constrained to the setting of images of items in museum galleries [97]. [98] addresses the domain shift issue in one-shot learning. [99, 100, 96]also tackle the problem of domain adaptation where in the new domain only few data samples are available. Common to all these prior works is their limiting the cross-domain setting to the realm of natural images, which still retain a high degree of visual similarity, and do not capture the broader spectrum of image types encountered in practice, such as industrial, aerial, and medical images, where cross-domain few-shot learning techniques are in high demand.

While the above efforts do not provide insight on what knowledge should be focused on for few-shot learning, we believe in the importance of learning generally applicable knowledge of visual primitives to benefit the downstream cross-domain tasks. Existing cross-domain methods do not carefully choose source domain tasks, but they put more effort on domain adaptation. For example, based on mini-ImageNet knowledge, they verify on CUB, or simply learn jointly from several source tasks. In contrast, we believe that source tasks can introduce large domain discrepancy if

not carefully chosen, and extracting knowledge generally applicable across different domains will mitigate domain discrepancy and reduce the difficulty of domain adaptation.

4.2.2 Zero-Shot Learning

Semantic Space

Typical semantic information used in Zero-Shot Learning (ZSL) methods includes attribute [77, 101, 102, 103], word vector [104, 105], or text descriptor [106, 107, 108]. Some methods also explore multiple semantic descriptions, and different fusion methods of multiple semantic spaces, such as score-level fusion [109], recurrent neural network[110] or joint space learning [111, 112]. In contrast, instead of different semantic description space, our proposed approach explores multiple visual spaces with each corresponding to a intermediate semantic space.

Projection Learning

Existing methods also differ in projection learning of visual space and semantic space: (1) Regression from visual space to semantic space base on conventional regression model [113, 111] or deep networks [105, 114, 107, 108] (2) Semantic space to feature space [115, 116, 110] (3) Intermediate Space learning of visual and semantic features [117, 118, 119, 120, 108, 121, 122]. Hubness problem [123] is studied in high-dimensional data, and is explored in zero-shot learning [124, 125, 126]. To alleviate such problem, [110] proposes to map semantic space to visual space.

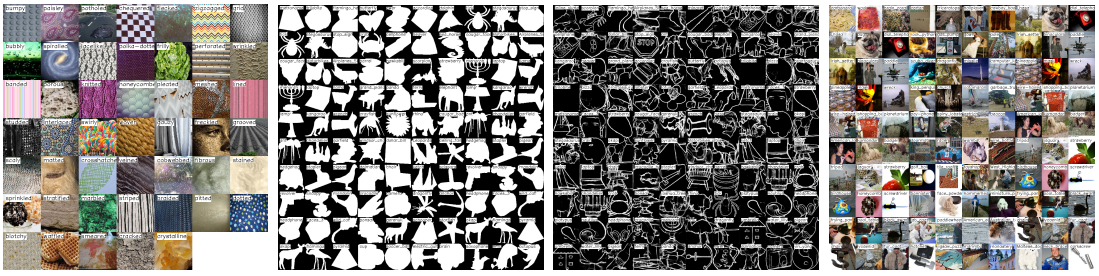
Our method differs from the existing methods in the above two respects: (1) Semantic Space: in the existing methods, semantic attributes are provided by testing benchmarks, and these semantic attributes are isolated among datasets because they are specifically annotated for each dataset. For example, for CUB dataset [127],

the attributes are defined for birds , such as "have red mouth", whereas for SUN dataset [128], the attributes are defined for scenes such as "has mountain","water". In contrast, our VPs in MDXNet can be shared among different domains of datasets since they describe generally applicable latent attributes. (2) Projection Learning: different to existing methods, our proposed MDXNet creates an intermediate space between visual space and semantic, which help alleviate domain discrepancy by using multiple visual spaces of visual primitives.

4.2.3 Domain Generalization and Domain Adaptation

Domain adaptation methods [129] target for reducing the domain shift between the source and target domains. Domain adaptation can be achieved by re-weighting the original samples [130] or by learning a classifier in the new domain [131] or using adversarial training at the feature-level to align the source and target distributions [132, 133]. Transfer learning has been investigated intensively [134, 135, 136, 137] on natural images to reduce domain shifts. [138, 139, 140, 141] focus on understanding transfer learning, but these methods often need large datasets. Most domain adaptation methods, however, target at adapting knowledge of the same category learned from the source to the target domain and therefore are less effective for handling novel categories as in the few-shot classification scenarios. These domain adaptation methods require access to the unlabeled images in the target domain at the training stage, which may not be feasible in many applications due to the difficulty of collecting abundant examples of rare categories (e.g., rare skin diseases, rare bird species).

Domain generalization [142] methods target for generalizing from a set of seen domains to an unseen domain without accessing instances from the unseen domain during the training stage. Existing methods proposed for tackling the domain generalization problem include extracting domain-invariant features from various seen domains [142, 143, 144], improving classifiers by fusing classifiers learned from seen



(a) Texture VPs (b) Shape VPs (c) Shape VPs (d) Object-Appearance VPs

Figure 4.3: VP design routine (a) DTD 47 human interpreteble attributes which are texture VPs visual words (b) General Object Shape (100 Classes) which are extracted from Caltech101 dataset, we extract edges from original image to generate the shape description dataset, as shown in (c), the class labels are shape VPs visual words. (d) ImageNet 100 out of 1000 classes are displayed here, 1000 classes labels are object-appearance VPs visual words.

domains [145, 146], decomposing the classifiers into domain-specific and domain-invariant components [147, 148], learning to augment the input data with adversarial learning [149, 150] and applying the learning-to-learn strategy to simulate the generalization process in the training stage [151, 152, 153].

Different from prior work, the goal of our work is to develop a method that is robust to domain shifts in cross-domain few-shot and zero-shot learning tasks through learning and integrating visual primitives that emulate the human visual system. In addition, our focus is on recognizing novel categories from unseen domains in few-shot and zero-shot learning settings.

4.3 MDXNet for Cross-Domain Few-Shot Learning and Zero-Shot Learning

We propose Multi-Domain Explainable Latent Attributes Network (MDXNet) for cross-domain few-shot learning and zero-shot learning tasks. The key idea behind the design of MDXNet is in line with the human visual system, i.e., the multi-domain knowledge of visual primitives extracted from natural images can be associated with

image patterns in various domains in a general way, and the multi-domain knowledge are complementary and should be leveraged for cross-domain learning tasks. We first introduce VPs in details, and then describe the designed structures of MDXNet for these two tasks.

4.3.1 Visual Primitives and VP Domain Knowledge Learning

First, because what we need are conceptual level representations, the representations should be deep features, not shallow or image operation-specific representations. Second, on top of the conceptual representations, we also need domain adaptation for cross-domain tasks. We propose to address these two issues by integrating domain generalization and domain adaptation. We do this for several reasons: (1) For the target domain, the domain-specific knowledge is usually unknown, and a general feature representation space could provide useful prior knowledge. A target domain only has a few labeled data, making domain adaptation very hard since few-shot learning can easily make the learning space biased. For the individual visual primitive (VP) domains, we train separate feature descriptors, $f_{\theta_{tex}}(\cdot)$, $f_{\theta_{obj}}(\cdot)$, $f_{\theta_{shape}}(\cdot)$, parameterized by the network weights, θ_{tex}, θ_{obj}, θ_{shape}, to learn the embedding vectors for texture, object-appearance, and shape VPs, respectively.

Texture VPs

Our proposed texture VPs describe texture patterns of images with texture latent attributes, and we learn the VPs by using the Describing Textures Dataset (DTD) [154] that annotates 47 human explainable texture attributes for natural images as shown in Figure 4.3 (a). This texture dataset provides common representations at the human conceptual level. Texture domain features describe not only the spatial arrangement but also the material surfaces. For example, in Figure 4.2, Storage

Tank is similar to object appearances of accordion and remote control, because of the contrast of light color white and black. However, in texture domain, it is described as studded because the surfaces of the storage tank are uneven. Roundabout is described as spiralled, not only because the spatial arrangement of roundabout is spiralled, but also because its surface appears to have a hole in the middle, which resembles spiralled and crystalline. For Snowberg, it is described as skunk in object appearance because of the white and green color, but it is also described as matted in texture due to the material surface similarity. DTD is the most appropriate dataset for texture primitives, which has multiple class labels per texture image and the most classes for daily texture descriptions and concept level texture labeling. Other texture benchmarks are too small [155, 156] or not having human interpretable texture descriptions [157, 158], or having only single class labels.

Texture Domain We train the texture feature extractor $f_{\theta_{tex}}(\cdot)$ and the Classifier $K_{tex}(\cdot|\mathbf{W}_b^{tex})$ (parametrized by $\mathbf{W}_b^{tex} \in \mathbb{R}^{\mathbf{D}_{tex} \times c_{tex}}$) to minimize the standard Binary Cross-Entropy Classification Loss L_{tex} for multi-label classification by using the examples in the texture dataset $\mathbf{x}_i^{tex} \in \mathbf{X}_b^{tex}$. We denote the embedding as $\mathbf{d}_{tex} \in \mathbb{R}^{\mathbf{D}_{tex}}$ and the output classes as c_{tex}. The classifier $K_{tex}(\cdot|\mathbf{W}_b^{tex})$ consists of a linear layer $(\mathbf{W}_b^{tex})^\top f_{\theta_{tex}}(\mathbf{x}_i^{tex})$ followed by a sigmoid activation function.

Shape VPs

Our proposed shape VPs provide shape outline descriptions learned from natural images. Choosing an appropriate source to provide shape prototypes is important. We choose Caltech101 dataset [159, 160] to extract shape visual primitives based on the following considerations: it contains many images to learn sufficient daily shape prototypes, and it facilitates learning to differentiate shape classes by shape representations. Humans can recognize objects from their silhouette in natural images. The silhouette or sketches are directly related to the edges and shapes of objects.

So training a classifier to recognize natural objects from their edge information is an attractive way to learn shape embedding from natural objects. Figure 4.3(b) shows the 100 silhouette images from 100 classes extracted from Caltech101, which include clock, sofa, car, hat, person, etc. As observed, they contain most daily shape prototypes such as triangle, rectangle, round, and thus they are good proxies for shape prototypes of the natural object classes. In order to efficiently extract shape information, edges are extracted from all images by a perceptual edge detection algorithm [161] as shown in Figure 4.3 (c), which provides shape and edge descriptions of images while ignoring irrelevant details. The input to shape representation learning are the edge extracted images. It is worth noting that Specific names for the shape classes are not critical, and only the embedding and clustering of different classes are important. For example, in Figure 4.2, we could name iPod, snorkel and coral ref as rectangular rows, arc contours, curvy borders, which may fit the target domain descriptions better, but they have no effect on the learned embedding other than our class labels.

Shape Domain We use the edge detection algorithm BDCNet [161] to generate edge detected images for use as $\mathbf{X}_b^{shape}$, which are derived from Caltech101 dataset 100 classes. The details of this dataset are described in Section 4. We train the shape feature extractor $f_{\theta_{shape}}(\cdot)$ and the Classifier $K_{shape}(\cdot|\mathbf{W}_b^{shape})$ (parametrized by $\mathbf{W}_b^{shape} \in \mathbb{R}^{\mathbf{D}_{shape} \times c_{shape}}$) by minimizing the standard Cross-Entropy Classification Loss L_{shape}. The embedding features of shape domain are denoted as $\mathbf{d}_{shape} \in \mathbb{R}^{\mathbf{D}_{shape}}$ and the output classes as c_{shape}. The classifier $K_{shape}(\cdot|\mathbf{W}_b^{shape})$ consists of a linear layer $(\mathbf{W}_b^{shape})^\top f_{\theta_{shape}}(\mathbf{x}_i^{shape})$ followed by a softmax function.

Object-appearance VPs

Our object-appearance VPs provide object-appearance latent attributes for natural images. To learn the representation, we use the large scale dataset ImageNet [162] that targets object category classification and detection for natural images. ImageNet

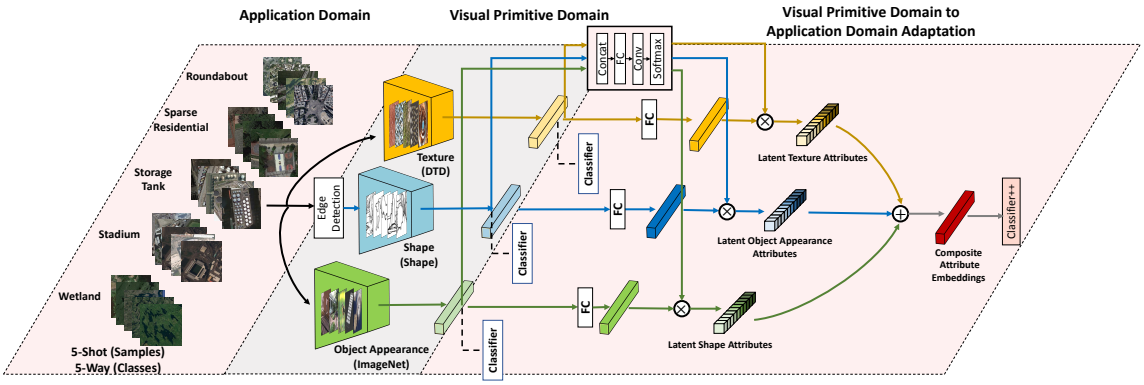


Figure 4.4: The proposed MDXNet framework showing VP embedding and transfer learning from outside the application domain.

can relate daily objects to those in many other target domains by data appearance representation. In Figure 4.3, we display 100 demo images from 1000 classes from ImageNet dataset.

Object Domain We directly use a feature extractor trained on ImageNet as our object domain feature extractor $f_{\theta_{obj}}(\cdot)$. We take the feature vector before the classifier as the object-appearance domain embedding, denoted as $\mathbf{d}_{obj} \in \mathbb{R}^{\mathbf{D_{obj}}}$.

In Figure 4.4, the classifiers that are connected through dashed lines to the feature extractors of the texture, object-appearance, and shape domains, respectively are only used during domain knowledge learning to generate the visual primitive embeddings. Afterwards, the feature extractors are all fixed for extracting the conceptual level VPs.

Justification for VPs

Here we mainly discuss the difference between VPs and semantic attributes. Early works studied semantic attributes in few-shot and zero-shot learning tasks. [163] proposes to perform feature selection for novel classes by using a prior learned features from familiar classes, and the features of novel classes are determined by their similarity to training examples. [164] also proposes to finding common features that can be shared across classes for multi-class and multi-view object detection. Early

works study semantic attributes in few-shot or 1-shot classification settings. [101, 165] are also based on traditional low-level attributes for recognizing new objects. [166] proposes zero-shot learning problem and tackles it with traditional low-level texture descriptions. Different to these early efforts, we model the out-of-domain data by visual primitives (VPs) since these learned representations are based on the early human visual system (HVS) processing pathways V1 and V2 for edge, color, shape, texture, motion, etc. Semantic attributes have been typically based on shallow networks and handcrafted operators (like Canny, LBP, HoG, SIFT, SURF, MSER, etc.) predating the recent work in deep learning [101].

The modern deep learning architectures with multiple parallel streams and network cascades that are successful at reaching human-level performance are closer to the complex multistage interconnected processing pathways in the HVS because human brains are multi-layers operations of different pathways and hence we choose to use the term visual primitives instead of semantic attributes. VPs provide a general visual conceptual presentation space which are generic across different tasks, instead of task-specific attributes. Our fine scaled representation is universal and the VP models (like local texture, local edges/ local shape, local parts-based decomposition/appearance) transfer well extremely different scales. For example, objects in aerial imagery has very different scales, local - one building with multiple windows repeating structures, global - city blocks with many similar buildings, VPs transfer appearance/parts- based repeating structures and spatial relationships knowledge between scales.

4.3.2 MDXNet in Cross-Domain Few-Shot Learning

Problem Description of Few-Shot Learning

Given abundant labeled base class data $\mathbf{X}_b$ and a small amount of labeled new class data $\mathbf{X}_n$, the goal of few-shot classification is to train classifiers for new classes by using the prior knowledge learned from $\mathbf{X}_b$. In conventional few-shot learning, $\mathbf{X}_b$ and $\mathbf{X}_n$ are from same domain, while in cross-domain few-shot learning, $\mathbf{X}_b$ and $\mathbf{X}_n$ are from different domains. In our method, $\mathbf{X}_b$ is composed of labeled data from the three domains of natural images, i.e., texture, object-appearance, and shape, denoted by $\mathbf{X}_b^{tex}$, $\mathbf{X}_b^{obj}$ and $\mathbf{X}_b^{shape}$, and the three domains are outside the application domain. Because we use base class data outside the application domain, we refer our method as cross-domain few-shot learning.

The input to MDXNet is an image from the new classes $\mathbf{x}_i \in \mathbf{X}_n$. The output is a composite knowledge feature vector $\mathbf{d}_n$. The vector $\mathbf{d}_n$ is learned with supervision by using a few labels of the new classes and on top of the visual primitive embeddings $\mathbf{d}_{tex}$, $\mathbf{d}_{obj}$ and $\mathbf{d}_{shape}$ that correspond to the three outside domains $\mathbf{X}_b^{tex}$, $\mathbf{X}_b^{obj}$ and $\mathbf{X}_b^{shape}$, respectively.

MDXNet is consisted of two stages: (1) Visual Primitive Domain Knowledge Learning: this stage is designed to learn visual primitive knowledge of the outside domains from natural images (learning $\mathbf{d}_{tex}$, $\mathbf{d}_{obj}$ and $\mathbf{d}_{shape}$). (2) Few-shot Learning: this stage is designed to associate knowledge of natural images with that of aerial images (learning $\mathbf{d}_n$). Figure 4.4 depicts the architecture of our proposed method.

It is seen from Figure 4.4 that in the stage of Domain Knowledge Learning, MDXNet performs domain feature extraction, and in the stage of Few-Shot Learning, MDXNet performs multi-latent attribute aggregation (MA) and classification. Based on latent attribute embeddings from multiple domains, we design a module to integrate them to form compositional attribute embeddings. Our rationale is that

each domain expert captures different information: texture domain captures low-level spatial arrangements features and surface properties, object domain adds higher-level structural appearance features, and shape domain provides perceptual shape or edge descriptions. These three domain features, when properly integrated, can work in synergy to improve the accuracy of different domains datasets.

Multi-Latent Attribute Aggregation Module (MLAA)

MLAA module is designed mainly for our domain latent feature aggregation. In Figure reffig:mlaa, it depicted the MLAA module algorithm. MLAA consists of two components: Domain Adapted Attention (denoted as DAA), Aligned Embedding Module(denoted as AE).

Domain Fusion

The input to MDXNet is an image from novel classes $\mathbf{x}_i \in \mathbf{X}_n$, After going through the multiple domain extractors , the three domain feature vectors are d_{tex}, d_{obj}, and d_{shape}. In our experiments, $d_{tex} \in R^{768}$, $d_{obj} \in R^{768}$, $d_{shape} \in R^{768}$. The compositional feature vector after is denoted as $d_n \in \mathbf{R}^D$, in out experiment, $d_n \in \mathbf{R}^{512}$. **Domain Adapted Attention Module** The combination of domain features should be adaptive to three domain feature vectors of each input, thus we design the Domain Adapted Attention module, which produces a vector for each domain embedding feature which is used to fuse domain feature vectors. $M \in \mathbf{R}^{512\times 3}$, $M = \{M_1, M_2, M_3\}$, each column of M is denoted as M_1, M_2, M_3. In Eq 4.1, $\Phi(\cdot)$ denotes {Convolutional, BatchNorm} operations, after we produce $h \in \mathbf{R}^{512}$, which is an attention vector. Since we have 3 domain, we want to obtain an attention vector for each domain, thus, we reshape h to a 1 channel vector $1 \times 512 \times 1$, and apply a convolutional layer and a batch norm layer to it. The parameter size for convolutional layer is only $3 \times 1 \times 1 \times 1$, thus, the output $\Phi(\widetilde{M}) \in \mathbf{R}^{3\times 512\times 1}$. We apply softmax to constrain the sum of attention vector

of three domains to be 1 on channel dimension, which means each row in M is 1 . In Eq 4.1, σ_s denotes softmax function.

$$
\begin{aligned}
\widetilde{M} &= \delta_f(d_{tex}, d_{obj}, d_{src}) \\
\delta_f &= W_h(d_{tex} + d_{obj} + d_{src}) + b_h \\
M &= \sigma_s(\Phi(\widetilde{M}) \\
d_n &= M_1 \cdot \widetilde{d}_{obj} + M_2 \cdot \widetilde{d}_{tex} + M_3 \cdot \widetilde{d}_{src}
\end{aligned}
\tag{4.1}
$$

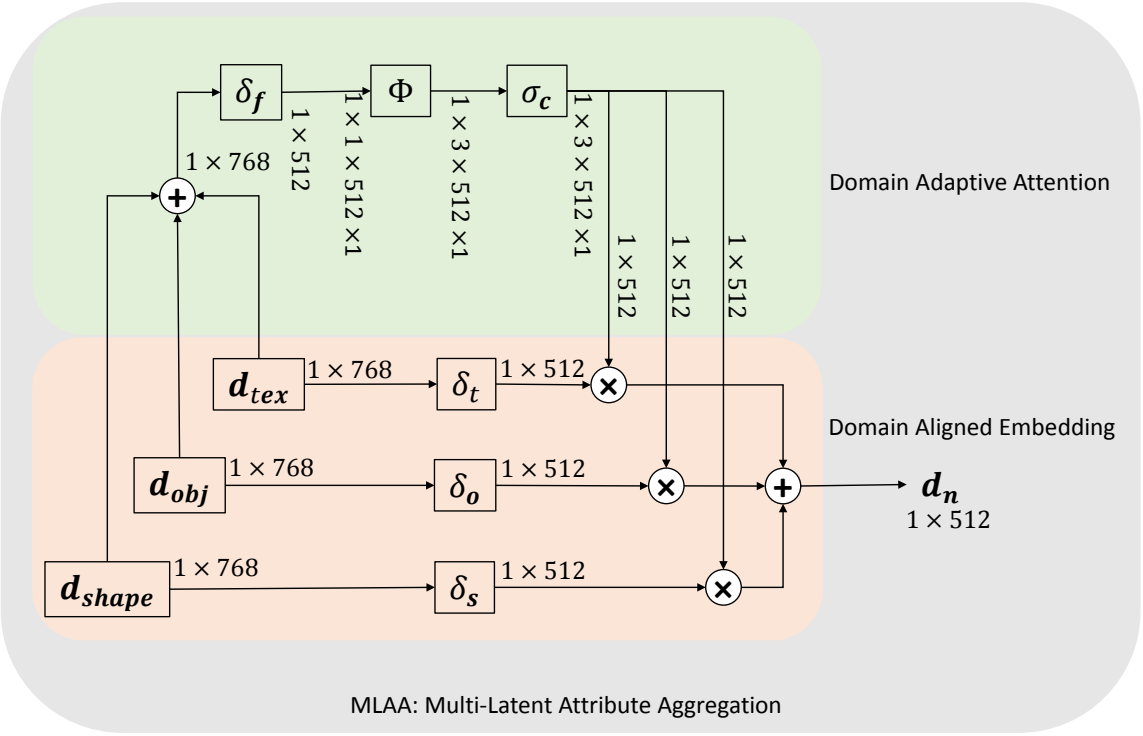


Figure 4.5: Multiple Latent Attribute Aggregation (MLAA) module that aggregates visual primitive embeddings to produce a fused domain aligned knowledge vector, d_n.

Domain Aligned Embedding

We design a knowledge fusion module as part of MDXNet for Multi-Latent Attribute Aggregation (MA) based on few-shot learning (see Figure 4.4), which provides a new architecture for fusing the interpretable VP embedding vectors learned as outside

domain prior knowledge. The MA module performs domain feature aggregation by using the aggregation function $\Phi(\cdot)$ to derive the composite attribute vector, $\mathbf{d}_n$, as defined in Eq 4.2, and this computation is defined in Eq 4.3, where $\widetilde{\mathbf{d}}_{tex}$, $\widetilde{\mathbf{d}}_{obj}$, and $\widetilde{\mathbf{d}}_{shape}$ are soft visual primitive embeddings obtained from affine transforms of $\mathbf{d}_{tex}$, $\mathbf{d}_{obj}$ and $\mathbf{d}_{shape}$, using the VP domain feature descriptors.

$$\mathbf{d_n} = \Phi(\mathbf{d}_{tex}, \mathbf{d}_{obj}, \mathbf{d}_{shape}). \tag{4.2}$$

$$\mathbf{d}_n = \widetilde{\mathbf{d}}_{tex} + \widetilde{\mathbf{d}}_{obj} + \widetilde{\mathbf{d}}_{shape} \tag{4.3}$$

The computation of $\widetilde{\mathbf{d}}_{tex}$, $\widetilde{\mathbf{d}}_{obj}$ and $\widetilde{\mathbf{d}}_{shape}$ are defined in Eq 4.4, where the domain-specific feature vectors $\mathbf{d}_{tex}$, $\mathbf{d}_{obj}$ and $\mathbf{d}_{shape}$ are projected to a shared embedding space to form $\mathbf{d}_n$. In the projection, we apply one fully connected layer to each domain feature vector, which is denoted as δ_t, δ_o and δ_s. After the projection, the three new domain feature vector become $\widetilde{\mathbf{d}}_{tex}$, $\widetilde{\mathbf{d}}_{obj}$ and $\widetilde{\mathbf{d}}_{shape}$.

$$\begin{aligned}
\widetilde{\mathbf{d}}_{tex} &= \delta_t(\mathbf{d}_{tex}), \delta_t = \mathbf{W}_h^{tex}\mathbf{d}_{tex} + b_{tex} \\
\widetilde{\mathbf{d}}_{obj} &= \delta_o(\mathbf{d}_{obj}), \delta_o = \mathbf{W}_h^{obj}\mathbf{d}_{obj} + b_{obj} \\
\widetilde{\mathbf{d}}_{shape} &= \delta_s(\mathbf{d}_{shape}), \delta_s = \mathbf{W}_h^{shape}\mathbf{d}_{shape} + b_{shape}
\end{aligned} \tag{4.4}$$

Classifier

The classifier of MDXNet is denoted as $K_n(\cdot|\mathbf{W}_n)$ (prameterized by $\mathbf{W}_n \in \mathbb{R}^{\mathbf{D}_n \times c_n}$). Different from the domain knowledge learning classifiers, we use the intra-class-variation reduced classifier as in [95] for few-shot learning.

We first compute the raw classification scores for a class k by using the cosine similarity operator s_k, as defined in Eq 4.5, and then apply the softmax operator. In

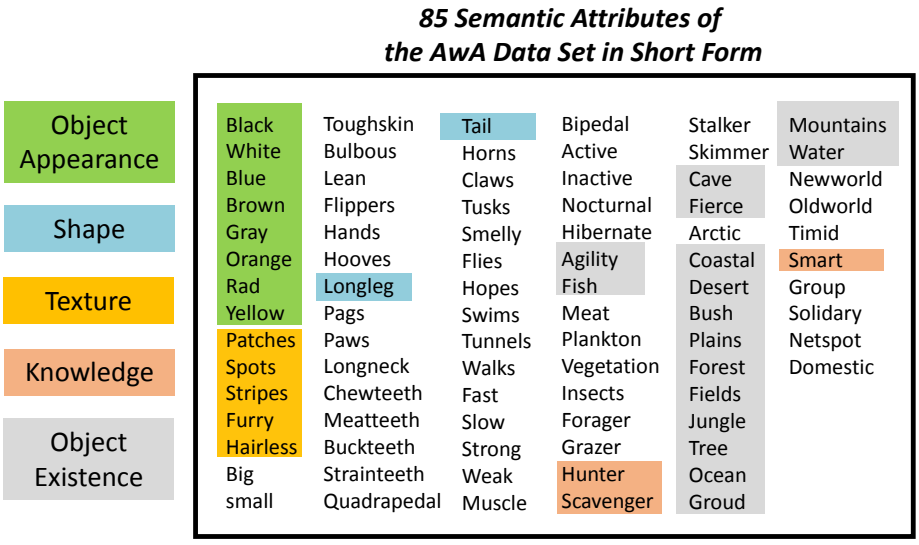


Figure 4.6: Zero-shot learning benchmark Animals With Attributes (AwA2) dataset with 85 semantic attributes with labels abbreviated (i.e. Fish for eating fish.

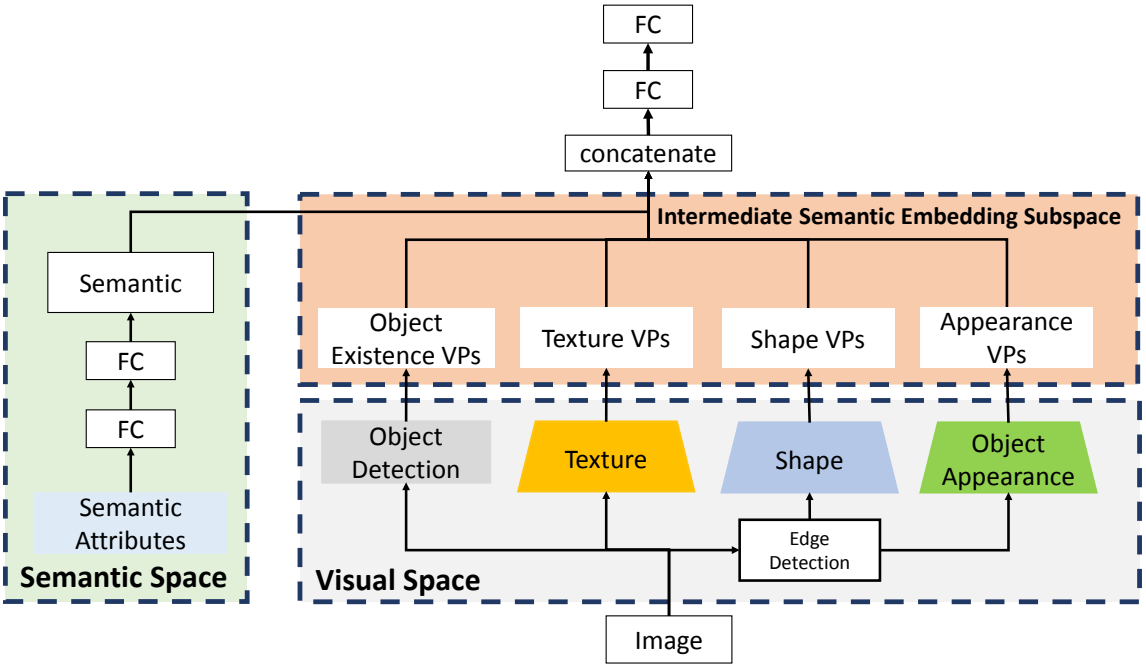


Figure 4.7: Extending MDXNet to zero-shot learning tasks.

the subsequent discussions, we denote this classifier as classifier++.

$$s_k = \tau \cdot cos(d_n, w_k^*) = \tau \cdot \bar{\mathbf{d}}_n^{\top} \bar{w}_k^* \tag{4.5}$$

where $w_k^* \in \mathbf{W}_n$, $\bar{\mathbf{d}}_n = \mathbf{d}_n / \|\mathbf{d}_n\|$, $\bar{w_k^*} = w_k^* / \|w_k^*\|$ are the L2-normalized vectors, and τ is a fixed scale factor for increasing the input range for softmax. In our experiments, we use $\tau = 2$.

In addition to using three outside domains for prior knowledge learning, we also consider using any two out of the three domains for this purpose, and in the degenerate case, we simply use one of the three outside domains to generate features for the application task.

4.3.3 MDXNet in Zero-Shot Learning

Problem Description of Zero-Shot Learning

Zero-shot learning is analogous to few-shot learning in that few shot images are given to define each class to recognize. However, instead of being given a support set with few-shot images for each of the training classes, it contains a semantic class embedding vector for each. The ZSL problem is formulated as given a labeled training set $\mathbf{D}^{train}$ with C^{seen} classes, and unlabeled test set $\mathbf{D}^{test}$ with C^{unseen} classes, and $C^{seen} \bigcap C^{unseen} = \varnothing$. $\mathbf{D}^{train} = \{(\mathbf{I}, \mathbf{S}^{C^{seen}}, \mathbf{Y}^{seen})\}_{(i=1,2,...N_{seen})}$ with N_{seen} samples, the image set is $\mathbf{I}$ and its class label set $\mathbf{Y}_{seen}$, with $\mathbf{S}^{C^{seen}}$ is its corresponding semantic representation vector. We use one instance $\mathbf{D}_i^{test}$ in $\mathbf{D}^{test}$ as an example. Given an image $\mathbf{I}_i$, its label semantic representation vector is $\mathbf{S}_i^{C_{unseen}}$, the goal of ZSL is to predict a class label y_i^s ($y_i^s \in C^{unseen}$).

MDXNet Creates an Intermediate Space between Semantic Space and Visual Space

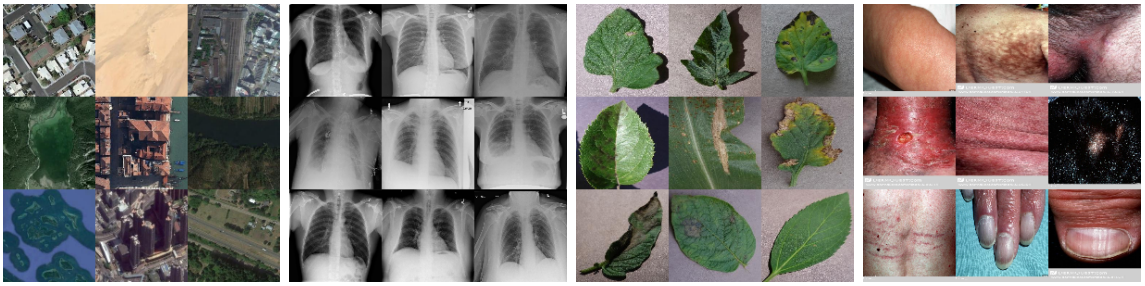
We modify the MDXNet to deal with the zero-shot learning problem, as shown in the Figure 4.7. as a different modality of semantic vectors is used for the support set (e.g. attribute vectors instead of images), the semantic embedding module is F_φ,

the visual embedding module is V_φ used for the image query set. Then joint learning network g_φ is applied for predicting label is $\hat{\mathbf{Y}}_i$ of each query input $\mathbf{I}_i$ will be:

$$\hat{\mathbf{Y}}_i = g_\varphi(F_\varphi(\mathbf{S}_i), V_\varphi(\mathbf{I}_i)) \tag{4.6}$$

$$V_\varphi(\mathbf{I}_i) = \left[f_{\theta_{tex}}(\mathbf{I}_i), f_{\theta_{shape}}(\mathbf{I}_i), f_{\theta_{obj}}(\mathbf{I}_i)\right] \tag{4.7}$$

As mentioned in related work, the semantic representation $\mathbf{S}_i$ can be various (for example, using word embeddings, semantic attributes). In our method, we project the visual representation space to different visual primitive domains. However, does our visual space provide enough information to mapping the visual space with the semantic space? We believe that the common embeddings used by most recent methods (embeddings from trained on ImageNet) are not enough for attribute semantic space representation. In Figure 4.6, we show that the 85 attributes from the ZSL benchmark Animals with Attributes (AWA). Different colored boxes represent different levels of information. The yellow box represents the attributes of texture. The light blue box represents the attributes of shape. The green box represents attributes that relate to shape. The gray box represents the information which can be observed from the background of the object since they are closely related to the living area or their closely related objects, and high-level attributes which are extremely hard to derive from images and should rely on knowledge from texts. Therefore, we believe that embedding various space of visual embedding is vital for zero-shot learning.



(a) NW45 (b) ChestX (c) CropDisease (d) SD198

Figure 4.8: Demo images from different application domains in few-shot learning

4.4 Datasets

We first describe the datasets we use to learn the visual primitive knowledge. We then describe the several types of application domains where we apply our approach: Aerial Scene Recognition, Alphabets Recognition, Skin Disease, and Plant Disease. We also describe the three types of zero-shot learning domains: animal recognition, bird recognition, and scene recognition.

4.4.1 Datasets for Visual Primitive Domains

Texture Domain The Describable Textures Dataset (DTD) [154] contains 5640 texture images. The images are annotated with the 47 attributes inspired from human perception. It contains both single label and multi-label annotations. **Object-Appearance Domain** ImageNet [162] data set is a well-known large scale visual learning dataset with 1000 classes for object recognition. **Shape Domain** We construct the Shape Primitive Dataset by processing images from Caltech101. It has the same 100 classes with 600 images per class. We generate the edge images by using the edge detection algorithm of [161].

4.4.2 Benchmarks for Application Domains

Few-Shot Learning Datasets

We evaluated our approach on several benchmarks of few-shot learning from different application domains: remote sensing image recognition NW45 [5], EuroSAT [167], plant disease CropDisease [168] ChestX-Ray [169, 167] and Skin Disease dataset SD198 [170]. We show 9 images from 9 classes of each dataset in Figure 4.8. NWPU-RESISC45 (NW45) [5] is a remote sensing image scene classification dataset which consists of 45 classes and each class includes 700 images with a size of 256×256 pixels in RGB. The dataset covered more than 100 countries and regions and extracted from Google Earth. NW45 is the aerial scene recognition dataset with the largest number of classes currently as far as we know. Existing methods in few-shot classification perform training and testing in the same data source. As there is no standard splits on aerial data in {Base, Validation, Test}. We, therefore, split the NW45 as our few-shot learning aerial scene datasets. The data split we used for {Base, Validation, Test} is {25 classes,10 classes,10 classes}. We made ten splits in this way, and denote them as NW45-split0,1,...9. In different splits, the classes in the Base, Validation and Test sets are different. NW45-split0 is constructed by first sorting the class names in alphabetical order and then taking the first 25, the next 10, and the last 10 classes for the Base, Validation, and Test Sets, respectively. The other 9 splits on NW45 are randomized splits of classes. SD198 [170] is a skin diseases dataset with 198 classes. [94] provide splits of ChestXRay, CropDisease, EuroSAT, thus we follow their experiments setting to report results. Omniglot [171] is an alphabet recognition dataset that consists of images from 1623 classes across 50 different alphabets. We follow the splits of {Base, Validation, Test} from [95]. Since we do not require visual knowledge data from the application or task domain, we apply our approach on the Test Set without using any samples from the Base and Validation Sets of the NW45

Domain	Method	20-way	25-way	30-way	35-way	40-way	45-way
Single-Domain	**Texture**	61.03 ± 0.33	56.61 ± 0.30	53.97 ± 0.26	51.65 ± 0.22	49.42 ± 0.19	47.34 ± 0.17
	Object-Appearance	67.44 ± 0.35	63.20 ± 0.32	60.88 ± 0.27	58.71 ± 0.23	56.62 ± 0.19	54.38 ± 0.17
	Shape	48.03 ± 0.33	43.21 ± 0.28	40.54 ± 0.24	37.91 ± 0.21	35.66 ± 0.17	33.66 ± 0.15
Dual-Domain	**MDXNet-TS**	68.24 ± 0.33	64.75 ± 0.28	62.39 ± 0.26	60.21 ± 0.22	58.39 ± 0.20	56.56 ± 0.17
	MDXNet-OS	68.82 ± 0.33	65.49 ± 0.30	63.21 ± 0.27	61.15 ± 0.23	59.41 ± 0.20	57.83 ± 0.17
	MDXNet-TO	71.00 ± 0.32	67.88 ± 0.29	65.45 ± 0.26	63.43 ± 0.22	61.48 ± 0.19	59.84 ± 0.17
Triple-Domain	**MDXNet**	$\mathbf{72.18 \pm 0.33}$	$\mathbf{68.97 \pm 0.29}$	$\mathbf{66.73 \pm 0.26}$	$\mathbf{64.67 \pm 0.22}$	$\mathbf{62.90 \pm 0.20}$	$\mathbf{61.13 \pm 0.17}$

Table 4.1: Evaluation on the effect of outside-domain features for 5-shot-N-way remote sensing image scene recognition on the NW45 Dataset. We abbreviate the three domains of Texture, Object-Appearance, and Shape as **T**, **O**, and **S**, respectively. We denote the dual-domain feature embeddings of Texture and Shape as **MDXNet-TS**, Object-Appearance and Shape as **MDXNet-OS**, and Texture and Object-Appearance as **MDXNet-TO**. **MDXNet** denotes the triple-domain embeddings of Texture, Object-Appearance and Shape.

and Omniglot data sets.

Zero-Shot Learning Datasets

Animals with Attributes (**AWA**) [165] consists of 30,745 images of 50 classes of animals. It has a fixed split for evaluation with 40 training classes and 10 test classes. The newly released AwA2 [172] consists of 37,322 images of 50 classes with 85 attributes. Caltech-UCSD Birds-200-2011 (**CUB**) [127] contains 11,788 images of 200 bird species with 150 seen classes and 50 disjoint unseen classes. **SUN** [128] is a scene recognition dataset with 102 attributes.

4.5 Experimental Results

4.5.1 Few-Shot Learning Analysis

All experiments are conducted under the standard setting of few-shot classification: K-Shot (samples) N-way (classes), with the support set being $K \times N$. We use 15 query images in each class, therefore the total number of query set is $15 \times N$. All the reported performance are accuracies averaged over 600 tests and with 95% confidence

intervals.

Ablation Study

Individual domain features are discriminative. In this study, We evaluate the performance of the embedding features derived from the individual domains of texture, object-appearance, and shape for aerial scene recognition. We train the classifier++ with each fixed domain feature extractor with $K \times N$ samples. Results are shown in the single-domain block (first three rows) in Table 4.1. We can see that the texture domain and object-appearance domain features both have good discriminative power for few-shot aerial scene recognition, and the shape domain feature, although less as effective, still has discriminative power. **Multiple-domain VP feature combination is synergetic**. In Table 4.1, we show the aerial scene classification results when combining features of two and three domains in the dual-domain block and the triple-domain block, respectively. It is observed that combining any two of the three domains improved recognition performance over that of the individual domains, confirming the effectiveness of the complementary features of each domain pairs. It is worth noting that adding texture domain to object-appearance domain or shape domain gave significant improvement. It is also observed that leveraging texture, object-appearances, and shape primitives gave the best performance. The effectiveness of multi-domain feature combination can also be observed on Omniglot dataset in Table 4.2. **The MLAA of MDXNet is effective**. Different from the commonly used direct concatenation of domain-specific feature vectors, in MDXNet, the MA module generates a composite prior knowledge embedding vector through an additive combination of the projected embedding features of the individual domains, and the projections are adaptively estimated during K-shot learning. This knowledge aggregation approach turned out to be more effective than direct feature concatenation (>1.5% -3.5% in classification accuracy gain).

Knowledge Domain	Method	5-way	20-way
Source	**MANN [173]**	94.9	–
	MatchingNet [75]	98.7	98.7
	ProtoNet [76]	99.7	98.9
	RelationNet [77]	99.8	**99.1**
	MAML [73]	**99.9** $\pm$ 0.1	98.9 $\pm$ 0.2
	CloserLook [95]	99.38 $\pm$ 0.10	–
	AdaCNN [174]	99.37 $\pm$ 0.28	98.43 $\pm$ 0.05
Outside Single-Domain	**Texture**	90.51 $\pm$ 0.45	76.65 $\pm$ 0.34
	Object-Appearance	89.14 $\pm$ 0.52	74.27 $\pm$ 0.37
	Shape	87.89 $\pm$ 0.57	71.90 $\pm$ 0.38
Outside Dual-Domain	**MDXNet-TS(ours)**	95.78 $\pm$ 0.27	93.21 $\pm$ 0.18
	MDXNet-OS(ours)	95.49 $\pm$ 0.28	92.75 $\pm$ 0.18
	MDXNet-TO(ours)	95.95 $\pm$ 0.27	92.67 $\pm$ 0.19
Outside Triple-Domain	**MDXNet(ours)**	96.39 $\pm$ 0.25	94.13 $\pm$ 0.16

Table 4.2: Comparison of MDXNet with state-of-the-art methods for 5-shot (sample) tasks on Omniglot with 5- and 20-way learning, where the texture domain feature extractor of MDXNet used InceptionS.

Comparison with State-of-the-Art Methods

We first evaluate MDXNet using single-, dual-, and triple-domain feature embeddings using the NW45 aerial scene classification dataset. We compare our results against five state-of-the-art few-shot learning methods, including MatchingNet [75], ProtoNet [76], MAML [73], RelationNet [77], and CloserLook [95]. These deep architectures are trained on the Base set of NW45. Thanks to the nice implementation codes from CloserLook [95], we can easily evaluate these methods on the aerial scene recognition dataset. For fair comparisons, all our experiments on the state-of-the-art methods use the ResNet18 backbone and the same hyperparaments as in [95]. For implementation details of these methods, please refer to [95]. For comparison with CloserLook [95], we use the Baseline++ method from the authors' paper.

Single-outside-domain knowledge is competitive with knowledge from the application domain. In Table 4.3, we observe that our single outside domain features, such as Texture Domain and Object-appearance Domain, result in competitive performance with state-of-the-art methods that are trained using samples from the application task domain Base set, which are in Source Knowledge Domain block. We

Knowledge Domain	Method	NW45 [5]	ChestX-Ray [169]	CropDisease [168]	EuroSAT [167]	SD198 [170]
Source	MatchingNet [75]	79.45 ± 1.91	22.40 ± 0.85	66.39 ± 0.78	64.45 ± 0.63	76.29 ± 0.62
	MAML [73]	75.26 ± 3.11	23.48 ± 0.96	78.05 ± 0.68	71.70 ± 0.72	–
	ProtoNet [76]	83.28 ± 1.43	24.05 ± 1.01	79.72 ± 0.67	73.29 ± 0.71	74.29 ± 0.62
	RelationNet [77]	79.08 ± 2.47	22.96 ± 0.88	68.99 ± 0.75	61.31 ± 0.72	74.72 ± 0.86
	MetaOpt [175]	–	22.53 ± 0.91	68.41 ± 0.73	64.44 ± 0.73	–
	CloserLook [95]	85.19 ± 1.50	25.97 ± 0.41	89.25 ± 0.51	79.08 ± 0.61	$\mathbf{75.55} \pm 0.85$
Outside Multiple	CFL-all[94]	–	26.74 ± 0.42	90.82 ± 0.48	81.29 ± 0.62	–
	CFL-IMS-f[94]	–	25.50 ± 0.45	90.66 ± 0.48	83.56 ± 0.59	–
Outside Single-Domain	Texture	83.12 ± 1.46	25.26 ± 0.47	85.79 ± 0.61	82.00 ± 0.61	65.72 ± 0.47
	Object-Appearance	87.00 ± 1.35	25.71 ± 0.46	90.45 ± 0.51	84.65 ± 0.55	71.62 ± 0.51
	Shape	74.65 ± 1.52	23.43 ± 0.41	81.47 ± 0.68	70.24 ± 0.62	55.38 ± 0.50
Outside Dual-Domain	MDXNet-TS(ours)	86.34 ± 1.32	26.19 ± 0.50	90.24 ± 0.52	83.46 ± 0.51	70.84 ± 0.45
	MDXNet-OS(ours)	86.36 ± 1.16	26.53 ± 0.55	91.28 ± 0.49	82.76 ± 0.51	72.31 ± 0.46
	MDXNet-TO(ours)	88.27 ± 1.20	$\mathbf{28.12 \pm 0.55}$	$\mathbf{93.86 \pm 0.43}$	$\mathbf{86.96 \pm 0.51}$	74.94 ± 0.48
Outside Triple-Domain	MDXNet(ours)	$\mathbf{88.62 \pm 1.19}$	27.43 ± 0.54	93.20 ± 0.45	86.31 ± 0.45	74.68 ± 0.44

Table 4.3: Comparison of MDXNet with state-of-the-art methods for 5-shot (sample) tasks with 5-way learning on different datasets.

conjecture two reasons for this outcome: (1) Knowledge learned by state-of-the-art methods from the application domain data is not rich enough due to insufficiently labeled source data, and (2) Our VP embedding knowledge learned from the domains outside the application task domain has robust generalization power. It is worth noting that this encouraging outcome supports our proposed concepts of associating the aerial scene classes with multiple VPs of other domains.

Dual-Domain MDXNet and Triple-Domain MDXNet are effective for few-shot learning in other task domains. We evaluate the performance of MDXNet on multiple few-shot benchmarks. As shown in Table 4.3, MDXNet shows its robustness and obtains consistently higher classification accuracy than two sets of state-of-the-art methods using the source domains and using multiple domains methods. On Omniglot, MDXNet using dual or triple outside-domain features is able to nearly match the state-of-the-art performance (see Table 4.2) without using any data from the application domain, while the existing methods under comparison all use extensively labeled training samples from Omniglot. Since the Omniglot dataset consists of alphabet characters which are already edge-like binary images, we directly used the input images for the shape domain without applying the edge detection operator.

	AWA2 [172]				CUB [127]				SUN [128]			
	ZSL	GZSL			ZSL	GZSL			ZSL	GZSL		
Method	T1	u	s	H	T1	u	s	H	T1	u	s	H
DAP [113]	46.1	0.0	84.7	0.0	40.0	1.7	67.9	3.3	39.9	4.2	25.1	7.2
IAP [113]	35.9	0.9	87.6	1.8	24.0	0.2	72.8	0.4	19.4	1.0	37.8	1.8
CONSE [176]	44.5	0.5	90.6	1.0	34.3	1.6	**72.2**	3.1	38.8	6.8	39.9	11.6
CMT [105]	37.9	8.7	89.0	15.9	34.6	4.7	60.1	8.7	39.9	8.7	28.0	13.3
SSE [177]	61.0	8.1	82.5	14.8	43.9	8.5	46.9	14.4	51.5	2.1	36.4	4.0
DEVISE [104]	59.7	17.1	74.7	27.8	52.0	23.8	53.0	32.8	56.5	16.9	27.4	20.9
SJE [111]	61.9	8.0	73.9	14.4	53.9	23.5	59.2	33.6	53.7	14.7	28.8	19.5
LATEM [178]	55.8	11.5	77.3	20.0	49.3	15.2	57.3	24.0	55.3	14.7	28.8	19.5
ESZSL [120]	58.6	5.9	77.8	11.0	53.9	12.6	63.8	21.0	54.5	11.0	27.9	15.8
ALE [179]	62.5	14.0	81.8	23.9	54.9	23.7	62.8	34.4	58.1	21.8	33.1	**26.3**
SYNC [117]	46.6	10.0	90.5	18.0	55.6	11.5	70.9	19.8	56.3	7.9	**43.3**	13.4
SAE [180]	54.1	1.1	82.2	2.2	33.3	7.8	57.9	29.2	40.3	8.8	18.0	11.8
DEM [110]	67.1	30.5	86.4	45.1	51.7	19.6	54.0	13.6	**61.9**	**20.5**	34.3	25.6
RN(CVPR18) [77]	64.2	30.0	93.4	**45.3**	55.6	38.1	61.1	47.0	48.89	11.32	20.93	14.69
f-CLSWGAN* [181]	–	–	–	–	54.4	47.4	47.6	47.5	–	–	–	–
SP-AEN [182]	58.5	23.3	90.9	37.1	55.4	34.7	70.6	46.6	59.2	24.9	38.6	30.3
CVC[‡] [183]	71.1	56.4	81.4	66.7	54.4	**57.7**	43.7	49.7	–	–	–	–
MDXNet(T+O)	68.53	29.54	91.49	44.66	**57.14**	38.40	**66.13**	**48.59**	46.53	9.93	13.03	18.95
MDXNet(T+O+S)	**70.68**	26.32	**93.51**	41.88	**59.64**	**40.60**	65.58	**50.16**	48.54	13.33	21.32	16.41

Table 4.4: Comparison of MDXNet with other Zero-Shot Learning (ZSL) methods using two sets of testing conditions: ZSL T1 accuracy (higher is better), Generalized Zero-Shot Learning (GZSL) with unseen classes (u), seen classes (s), and H (harmonic mean of u and s). All scores are in percent. [†]Generates additional data for training but the other models do not. [‡]Uses episode training or meta learning which mimics the testing set up, while the other models do not.

4.5.2 Zero-Shot Learning Analysis

We apply MDXNet in zero-shot learning tasks, and the results are shown in Table 4.4. With three domain VPS, our MDXNet achieves the state of art results in several benchmarks, which demonstrates that creating an intermediate subspace of visual words help zero-shot learning tasks.

4.5.3 Explainability of VPs

Visualization of VP Word Embeddings We visualize the embeddding space of domain knowledge learned from natural images with aerial image embeddings. The texture domain embeddings of aerial images are obtained by the texture domain feature extractor of InceptionS. In Figure 4.9, we first use PCA [184] to reduce the dimension of texture domain embeddings to 50, and then use t-SNE [185] to visualize

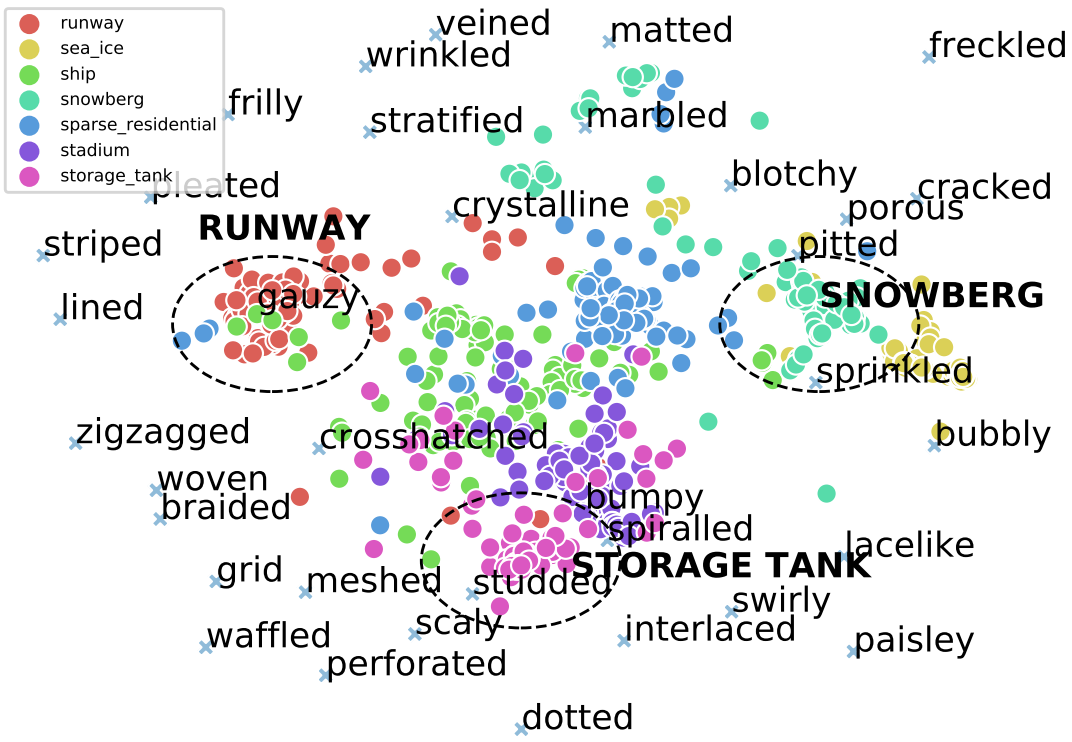


Figure 4.9: t-SNE visualization of VP feature embeddings for 7 of 45 classes from NW45 using MDXNet InceptionS backbone (best viewed in color mode). The lowercase text labels are for 35 of 47 DTD texture classes positioned at the feature embedding mean vectors. The three circles with uppercase texts identify the NW45 application domain classes: *runway, snowberg, storage tank*.

them in 2D space. For clear visualization, we plot the mean vectors of 35 texture classes in DTD (lowercase word labels). We visualize the embedding vectors of 100 samples of selected 7 classes in different colors from aerial images. We can clearly see that the texture domain embeddings of most samples of the class "Runway" from aerial images are near the mean vector of the class "Gauzy" from DTD due to their similar textures. We also find that the embeddings of the class "Snowberg"of aerial scene are near the texture classes "Pitted" and "Sprinkled", due to the visual similarity of these two texture classes with the aerial scene class. The visual similarities between the above discussed aerial scene and texture classes are illustrated by examples in Figure 4.10.

Figure 4.10: Representative VP words for the *runway* and *snowberg* NW45 classes from the t-SNE low-dimensional projection in Figure 4.9. See Figure 4.2 for the *storage tank* class VP words.

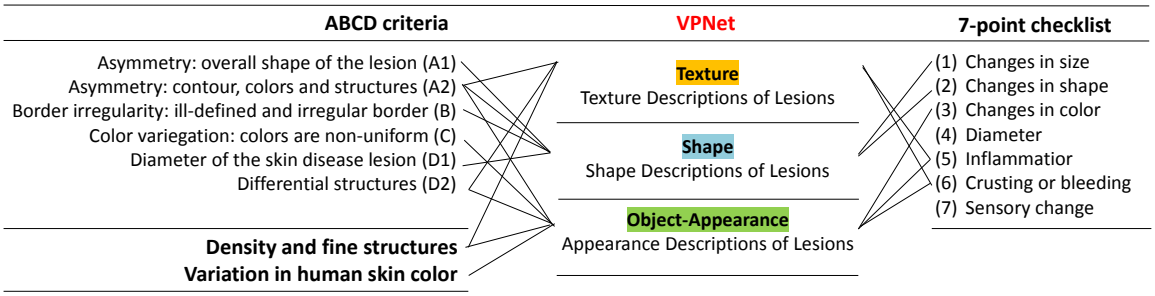


Figure 4.11: The mapping to visual primitives from dermatological criteria, ABCD criteria [6, 7], 7-checkpoint list [8], Density and fine structures [9] Variation in human skin color [10]

MDXNet is aligned with expert knowledge diagnosis of skin disease We analyze MDXNet explainability on skin disease diagnose problem. As shown in Figure 4.11, ABCD criteria [6, 7], 7-checkpoint list [8], Density and fine structures [9] Variation in human skin color [10] are four methods that skin disease experts used to diagnosis skin lesions. These rules can be mapped to our designed VP spaces.

4.5.4 Discussion

We propose VPs which benefit both domain adaptation and domain generalization tasks. We use VPs to transfer the internal representation of repetitive structures, symmetry, balance, spatial pattern/arrangement, spatial relationships learned in one universal/everyday/common domain to other specialized domains. By representing appropriate knowledge of VPs to mimic HVS and domain adaptation, we alleviate the domain shift problem, making MDXNet effective in cross-domain few-shot learning.

Our VPs create an intermediate space between seen domains and unseen domains for domain generalization tasks such as zero-shot learning.

Our MDXNet architecture has the following four advantages: (1) Good generalization and robustness: as the visual primitives cover multiple domain descriptions, they provide general-purpose embeddings and are not overly sensitive to training classes. (2) Low computation cost: as the visual primitives can be learned outside of an application domain, once they are learned, the composition of the visual primitives embeddings can be adapted to the application domain using only a few shot labeled images through the application-specific learning module with a small number of parameters, and thus is lightweight in computation. (3) Can use deep networks to extract visual knowledge since the outside domain has rich data available for learning different visual knowledge, in contrast to meta-learning based methods, which usually use shallow networks based on their design intuition of mimicking few- shot scenarios. (4) Amenable to introducing additional visual primitives. We demonstrate that each type of visual primitive embedding that we consider here is informative and effective, and the different types are complementary to each other.

We believe that the VPs space can be further optimized in the future since we believe each space can provide better prototypes for the target domain. Either we design an algorithm to compute good prototypes, or we increase prototypes in each domain to make them more generalized. In this work, we provide one way to construct the VPs space, and it is verified experimentally to deliver strong performances on both cross few-shot learning and zero-shot learning tasks.

4.6 Chapter Conclusions

We propose to mimic HVS by designing multiple types of VPs to leverage visual domain knowledge obtained from natural images to different application domains. Our

proposed network architecture MDXNet for VP learning and integration is effective on various application domains in both cross-domain few-shot learning and zero-shot learning recognition tasks. The VP embeddings provide a lexicon of visual words and text descriptions that correlate well with human perception and provide an explainable representation of sparsely labeled sample classes in new application domains. In future work, we plan to investigate effective methods for combining a wider range of domain knowledge from natural images and apply such knowledge to more cross-domain applications. We are also interested in the reverse direction of knowledge transfer, i.e., investigating whether visual knowledge gained from non-natural image sources can help natural image tasks such as recognition and segmentation.

Chapter 5

Domain Applications

We extended deep learning methods in various domains, including material science domain for predicting carbon nanotube forest attributes and mechanical properties, and biomedical domain for cell segmentation.

5.1 CNTNet: Carbon Nanotube Structure Recognition and Property Regression

5.1.1 Introduction

The timeline for materials discovery, development, and deployment is slow and resource intensive, requiring 10 to 20 years of research and development to bring a product to market [186]. The process also relies on human intuition and trial and error to traverse vast multi-variate synthesis domains. Integrating machine learning (ML) and artificial intelligence (AI) algorithms into the materials development cycle may drastically accelerate the process, reduce the role of human judgement, and alter the paradigm of materials research. Carbon nanotubes (CNTs) represent an illustrative example of the materials development cycle. CNTs have been at the

forefront of nanotechnology since their discovery in 1991 [187], yet their adoption into industrial applications remains sparse [188]. Their mechanical, thermal, and electrical properties far exceed those of conventional engineering materials [188], such that the technological potential of CNTs could be disruptive in applications ranging from structural materials [189, 190], digital electronics [191, 192, 193], flexible sensors [194, 195, 196], power transmission [197] and thermal devices [198, 199, 200], among others. As an example, CNTs exhibit an elastic modulus in excess of 1 TPa [201, 202] but are compliant enough to be tied into a knot and spun into yarns [203]. Their thermal conductivity exceeds 5,000 W/m-K [204] which is approximately 12 times greater than that of copper, at a mass density that is 4 times less. Single-walled CNTs have an electron mobility that is in excess of 100,000 cm^2/Vs [205] and can operate at frequencies greater than 1 GHz [206]. Scaling the physical properties of one CNT to large populations remains a significant challenge. When populations of CNTs are concurrently synthesized to form vertically oriented CNT forests, however, their ensemble properties are drastically reduced. Understanding and controlling the process-structure-property paradigm for CNT forests is a long-standing barrier to the widespread adoption of CNTs in many applications.

CNT forests are routinely synthesized using chemical vapor deposition (CVD) methods. The typical areal density of CNT forests ranges from an order of 10^9 CNTs/cm^2 for multi-walled CNT (MWNT) forests [207], to greater than 10^{13} CNTs/cm^2 [208] for single-walled CNT (SWNT) forests. The uncontrolled self-assembly of CNT forests generates an open-cell, foam-like structural morphology seen in the scanning electron microscope (SEM) images of Figure 5.1. The wavy, entangled morphology is thought to degrade the ensemble properties of CNT forests, sometimes by many orders of magnitude. As an example, the typical modulus of CNT forests is on the order of 10 to 100 MPa [209, 210, 211, 212, 213], a full four to five orders of magnitude less than the elastic modulus of individual CNTs. Similarly, the experimentally realized

thermal conductivity of CNT forests is on the order of 80 W/m-K [214], far less than the 5,000 W/m-K measured for an individual CNT. While the existing performance gap between predicted and observed properties is vast, it serves as an indicator that CNT forests may be designed to operate within a larger technological performance envelope if the CNT forest assembly processes could be better understood and controlled. Realizing properties within this envelope will require the development of new tools that can identify how CNT forest synthesis attributes correlate with forest structure and ensemble physical properties.

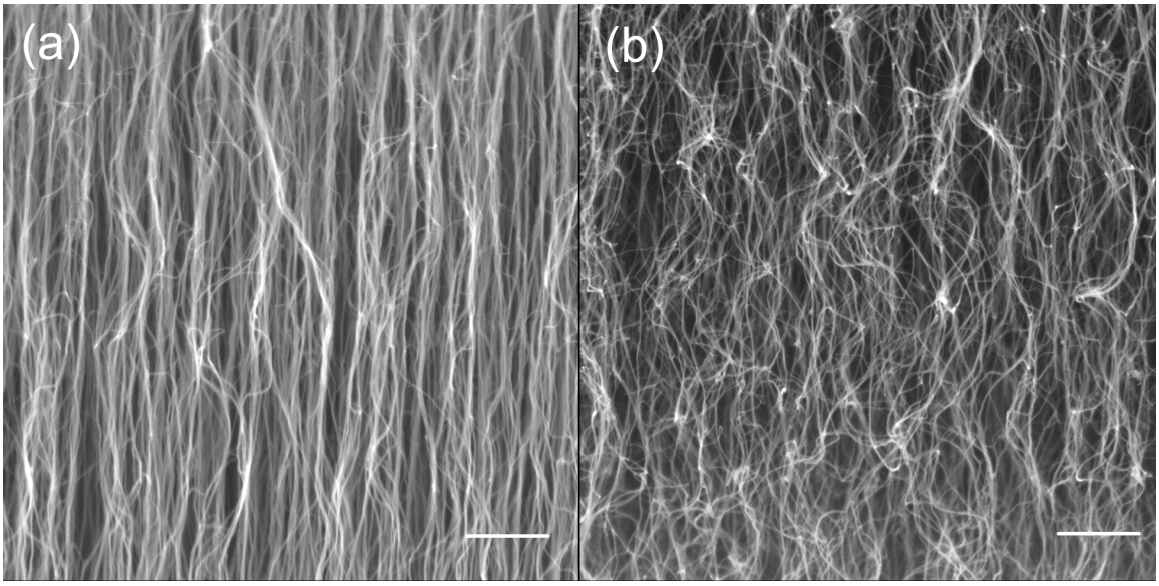


Figure 5.1: **SEM images of carbon nanotube forests.** The structural morphology of a CNT forests may exhibit relatively aligned CNTs or wavy and entangled CNTs depending on synthesis conditions. Scale bars represent 500 nm.

To efficiently navigate within a high-dimensional synthesis space, researchers have recently constructed a high-throughput, autonomous research robot to deterministically synthesize isolated single-walled CNTs (SWNTs). The Autonomous REsearch Systems (ARES) uses a Raman spectrometer laser to provide heat and simultaneously characterize growing SWNTs in-situ [215, 216]. The ARES system utilizes machine learning (ML) algorithms to learn from previous experimental results and uses an artificial intelligence (AI) planner to sequence experiments to reach an ex-

perimental objective. In a particularly compelling demonstration, the ARES system autonomously determined experimental parameter sets required to achieve specified SWNT growth rates in fewer than 100 experimental iterations [216]. The ARES approach shows that integrating AI and ML, with high-throughput experimentation can facilitate experimental parameter space navigation within a process-structure domain that is not well understood by researchers. The interactions between large populations of concurrently-growing CNTs in forests add complexity not present during the synthesis of isolated SWNTs, and ARES-style experimental campaigns interrogating CNT forests have not yet been reported but offer a promising approach.

In the current study, a time-resolved finite element method (FEM) CNT forest simulation tool [217, 218] is used as a high-throughput virtual laboratory to examine the synthesis-structure-property design loop of CNT forests. Images of each CNT forest morphology were obtained at the end of their simulated synthesis. A mechanical compression simulation was used to obtain mechanical properties [218, 219, 220]. Deep learning neural networks, trained and tested using FEM physics-based simulated images, were developed to predict the class label for the CNT synthesized images [221]. Subsequently, CNT forest physical and growth attributes like stiffness and buckling load are predicted using another ML random forest regression algorithm. A total of 63 unique CNT forest synthesis classes were established based on combinations of CNT diameter, population growth rate variability, and CNT areal density. From these classes, a combined pool of 22,106 FEM simulated synthesis and compression experiments were performed generating one image per experiment to create a pool of data for training and testing the ML models. *The physics-based FEM simulations provided precise ground-truth data that would otherwise be very difficult, if not impossible, to obtain using physical experimentation techniques.* The ML techniques achieved greater than 91% classification accuracy of physical and growth attributes, with an R^2-regression fit of 0.96 and 0.94 for buckling and stiffness predictions, re-

spectively. The image-based R^2 coefficient of determination, and root-mean-square error (RMSE) values matched or exceeded those obtained using a classification-based linear regression analysis in which all input physical attributes were precisely known *a priori* and no image data was considered. The results demonstrate that image-based ML algorithms using simulated SEM imagery are able to detect unique visual structural morphological characteristics of CNT forests to provide precise, individualized predictions.

5.1.2 Results

Simulating CNT Forest Synthesis and Mechanical Compression

The typical CVD parameters available for CNT forest synthesis include catalyst composition, catalyst thickness, buffer layer composition, buffer layer thickness, substrate temperature, gas temperature, catalyst conditioning, hydrocarbon gas composition, carrier gas composition, synthesis pressure, and synthesis time, among others. Each of these parameters can influence the resulting diameter distribution, areal density, growth rate, and CNT catalyst lifetime. For example, increasing the porosity of an alumina buffer layer (controlled by deposition methodology) was shown to drastically increase the growth rate, lifetime, and density of CNTs when using an Fe catalyst thin film [222]. Likewise, trace amounts of carbon can increase CNT number density by reducing oxidized catalyst nanoparticles [223, 224]. Because of the cost and time required to explore the available synthesis parameter space, researchers typically operate within a small parameter range that has produced acceptable CNT growth in the past. In fact, time and cost constraints are disincentives to thoroughly explore the vast parameter space. For the eleven synthesis parameters mentioned above, a full combinatorial experimental campaign consisting of just three quantized levels per parameter, without replicates, would consist of 177,147 experiments. At a relatively

aggressive pace of five experiments per day, this experimental campaign would require almost a hundred years to complete. Characterization of the resulting CNT forests would require additional time that may exceed the time required for synthesis.

Here, a mechanical finite element simulation was used as a high-throughput *virtual laboratory* to synthesize and mechanically compress thousands of unique CNT forests. Deep learning algorithms classified the CNT forest physical attributes and predicted the CNT forest mechanical properties based solely on the simulated CNT forest imagery. Deep learning algorithms typically require very large amount of training data which is not available in our case. The deep learning algorithms bootstrap onto the physics-based simulation to predict the complex synthesis-structure-property relationships of CNT forests, and can be viewed as an advanced data augmentation method to provide more training data, where data or ground-truth or both is very sparse like in molecular, cellular and tissue imaging [225, 226, 227, 228]. Diverse CNT forests could then be generated using combinations of seven CNT population densities, three CNT diameters, and three levels of population growth rate variability. The CNT linear density varied from 50 to 200 CNTs per 10 μm simulation span. These CNT densities correspond to 2.5x10^9 to 4x10^{10} CNT/cm^2, consistent with reports for multi-walled CNT forests [224]. Seven unique CNT density levels were selected to span this large span of observed results. The levels of CNT diameter (10, 16, and 22 nm) and CNT forest growth rate coefficients of variability (3, 6, and 9%) were also selected based on available data for multi-walled CNT forest synthesis data [229]. Each unique combination of these parameters was considered to represent a labeled class or category and used for supervised learning. We designate this complementary suite of digital AI/ML tools for CNT material discovery as *CNTNet*.

Each CNT forest simulation uses homogeneous CNT diameters selected from one of three values and a stochastic assignment of growth rate and orientation angle. Figure 5.2 displays a representative set of simulated CNT forest morphology images

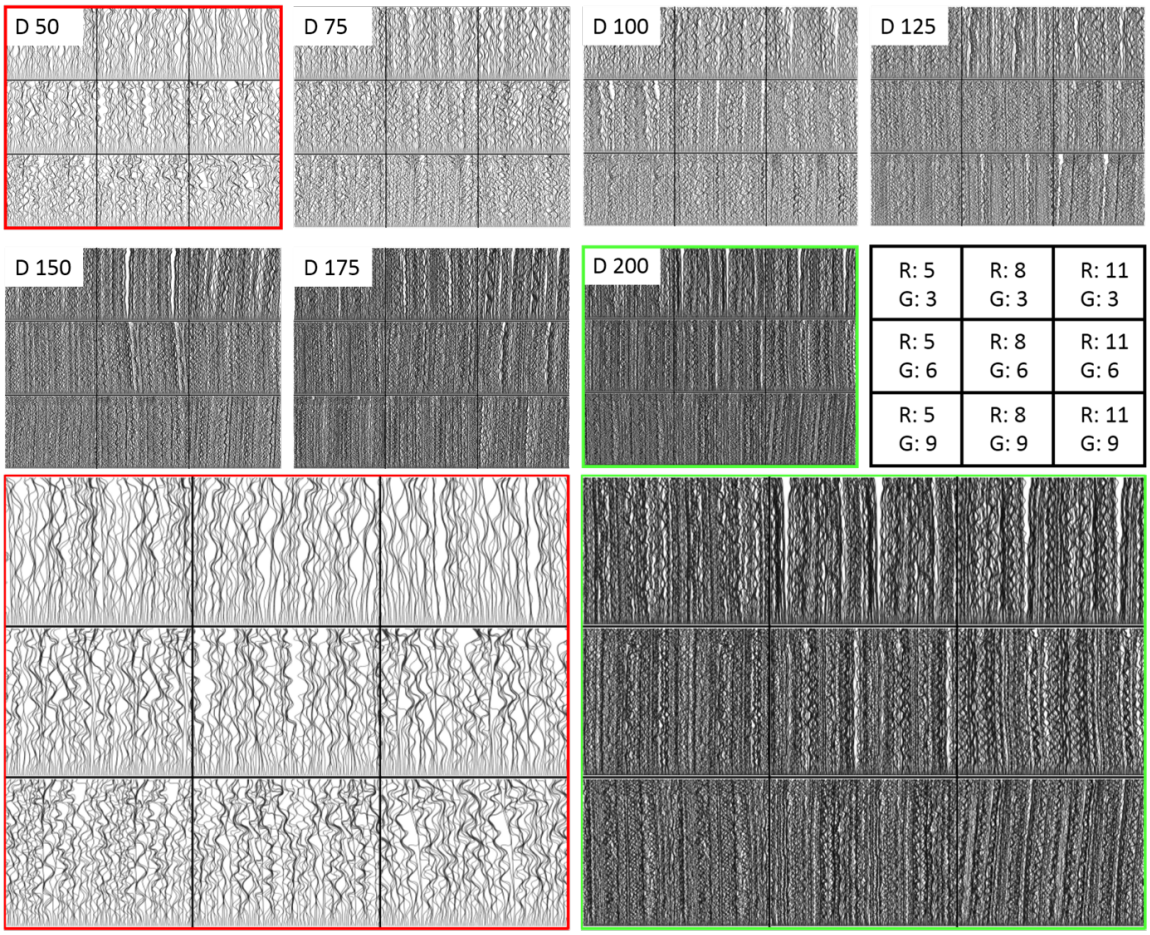


Figure 5.2: **Representative images from all 63 simulated CNT forest classes.** Each block of nine sub-types is grouped by CNT linear density, ranging between D50 to D200 (seven values), representing a density of 50 and 200 CNTs per 10 μm simulation span, respectively. Each block of 9 images is arranged in an order consistent with the key provided on the right side of the middle row of images. Here, R indicates the outer radius in nm (three values), and G is the growth rate coefficient of variation (three values). The image blocks for D50 (red border) and D200 (green border) are expanded in the third row to show greater image texture details. The horizontal domain is 10 μm for each simulation. Note that each image has a replicated texture block in the horizontal direction to demonstrate periodic boundary conditions.

from each of the 63 forest classes. While synthesis is confined to a two-dimensional plane, the resultant CNT waviness and bundling resemble physical CNT forests. In this way, the resultant CNT forest morphology approximates a slice from larger three-dimensional CNT forest. Each block of images is sorted based on the CNT density. The lowest and highest CNT densities, denoted as D50 and D200 (corresponding

to 50 and 200 CNTs per 10 μm span, respectively), are enlarged and shown in the third row of the figure. Some trends may readily be identified by the naked eye, particularly when observing the lowest and highest density examples. Increasing CNT linear density decreased the white space in each image, while the wavelength and amplitude of CNT waviness also decreased. An increased growth rate coefficient of variation within a forest decreased the vertical orientation of individual CNTs within a forest, leading to an increased CNT-CNT contact density and decreased wavelength along the length of the forest. Importantly, all CNTs were plotted with the same line thickness, regardless of CNT diameter. This decision was motivated by the likely resolution limits of an SEM, which at modest magnifications would be unable to distinguish between the small variations in CNT diameters presented here. As a result, all diameter classifications were inferred based upon variations in the CNT forest morphology rather than by direct diameter acquisition. Because the bending stiffness of hollow cylinders increases with the fourth power of diameter, the CNTs within larger-diameter CNT forests have less curvature than smaller-diameter CNT forests.

Each of the CNT forests used for classification were mechanically compressed to determine their buckling load and elastic stiffness. To the best of our knowledge, this study of 22,106 CNT forests represents the most extensive evaluation of CNT forest mechanics to date. A representative CNT forest undergoing compression, and the resultant force-displacement response, is shown in Figure 5.3. During compression, the CNT forest morphology deformed and buckled at the top and bottom-most regions of the CNT forest throughout the initial stages of compression, before densification. Coordinated CNT buckling that originated near the top surface of the CNT forest grew in extent with increased compressive strain, while a single buckle near the bottom of the forest persisted without obvious growth. Similar collective buckling behavior has been observed experimentally [209, 230, 211, 231]. The forest stiffness [N/m] was

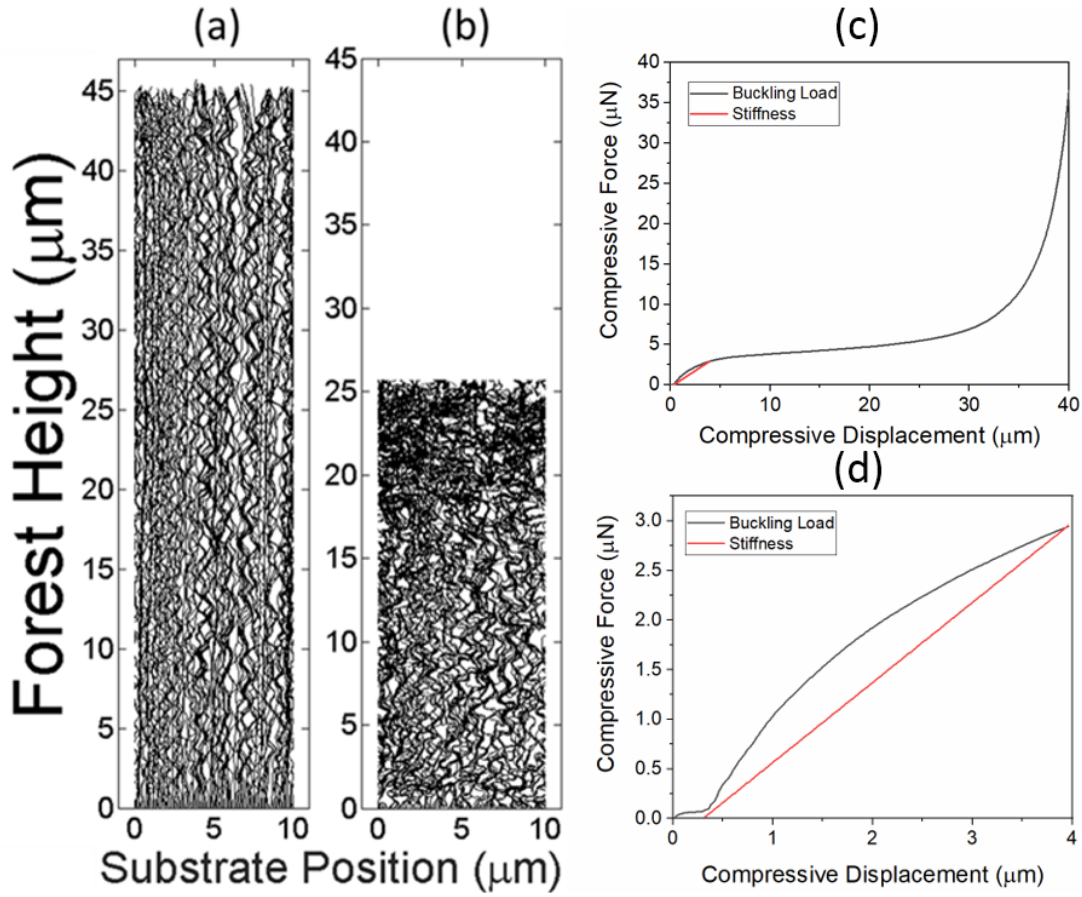


Figure 5.3: **Simulated CNT forest compression.** An expanded CNT forest compression demonstration showing the simulated CNT forest morphology (a) at the beginning of compression and (b) after 20 μm compression. (c) The full load-displacement curve shows the typical linear-elastic regime at low displacement, an extended plateau, and a densification regime. (d) A magnified view of the loading curve shown in (c) illustrates the evaluation of forest stiffness and buckling load.

determined from the tangent loading slope evaluated at a compressive displacement of 1.5 μm from the top surface of the forest. The initial 0.5 μm of compression after first contact deformed only the free ends of the tallest CNTs, as shown by the small and erratic loads at the onset of compression in Figure 5.3(c), and was not characteristic of the overall forest stiffness. At 1.5 μm compression, by contrast, contact was established with the entire span of the CNT forest. The buckling load was determined by imposing a line with a slope equal to the CNT forest stiffness and an offset of 0.4μm of compression. The buckling load was defined as the point at which

applied load and the offset stiffness line (shown in red in Figure 5.3(c)) intersected. The typical compression simulation was terminated after these characteristic metrics were obtained. An extended compression experiment is shown in Figure 5.3(d) showing the typical elastic, plateau, and densification regions that are characteristic of CNT forests. Nested box plots in Figure 5.4 display the stiffness and buckling load

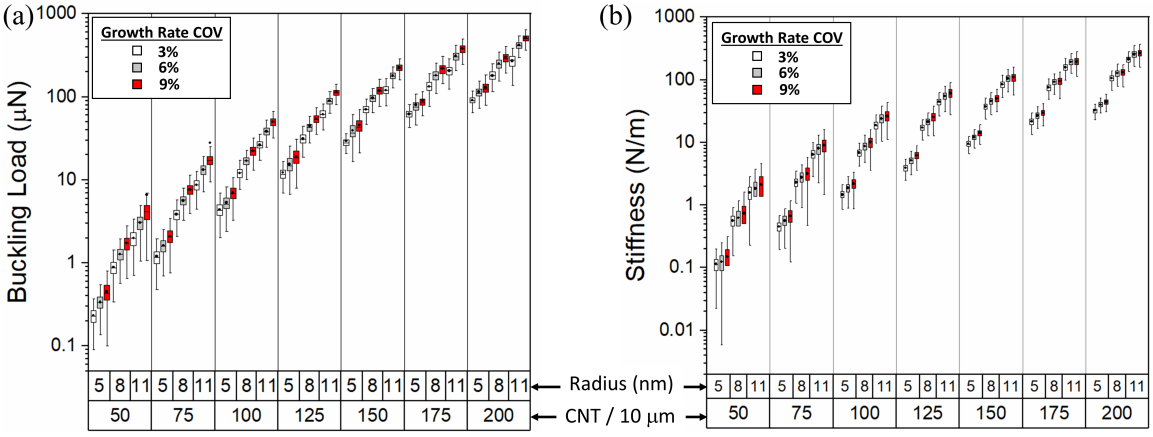


Figure 5.4: **Nested box plots of simulated CNT forest mechanical properties as a function of class.** The (a) buckling load and (b) elastic stiffness for 63 distinct classes of CNT forests vary by orders of magnitude within the simulated parameter space. The shaded box represents the upper and lower quantile, while the vertical whiskers represent 1.5 times the interquantile range. The bottom grouping, ranging from 50 to 200 represents the quantity of CNTs, the upper grouping represents CNT outer radius of 5, 8, and 11 nm.

representative of all CNT forests. Axes are plotted using a logarithmic scale to show the power-law scaling that exists between forest attributes and the corresponding mechanical response. Within each plot, the mean value is plotted as a solid dot, the shaded box represents the upper and lower quantile of the data, while the vertical whiskers represent 1.5 times the interquantile range. The dual x-axes are sorted by nesting CNT linear density and CNT outer radius in ascending order. Note that the inner CNT radius is 70% that of the outer radius in all simulations. The various population growth rate coefficients of variation are plotted by color within the box plots. Based on the box plots, the buckling load and stiffness are positively correlated with all CNT attributes, with a ranked sensitivity (from smallest to greatest) of pop-

ulation growth rate variation, diameter, and linear density. The variation of stiffness and buckling load spans greater than three orders of magnitude for the combination of parameters varied in the current study.

To determine the scaling relationships associated with each CNT forest attribute, and to serve as a baseline for later ML predictions, a linear regression analysis of CNT forest buckling load and stiffness was performed relative to the input CNT forest physical parameters. The analysis inherently presumed a foreknowledge of the CNT forest characteristics of CNT density, CNT diameter, and population growth rate variation and were not informed by CNT forest morphology images. We stress that the precise knowledge of these CNT forests attributes would be difficult or impossible to obtain with physical experiments, while simulation provided access to the exact values. The values within each CNT attribute class were normalized by the modal value for the regression analysis to ensure that each parameter varied by similar magnitude. A linear regression analysis based on the log_{10} parameter values was then conducted. The analysis showed that $E_f \sim \rho^{1.9}$, $E_f \sim D^{1.34}$, and $E_f \sim COV^{0.13}$, where E_f, ρ, D, and COV represent CNT forest stiffness, density, diameter, and population growth rate coefficient of variation, respectively. Similarly, the CNT forest buckling load, σ_y, scaled as $\sigma_y \sim \rho^{1.94}$, $\sigma_y \sim D^{1.01}$, and $\sigma_y \sim COV^{0.25}$.

Well-established scaling relationships exist for ideal open-cell foams that relate elastic modulus, E, yield strength, σ_y, and relative density, ρ. For ideal 2D open-cell foams and honeycombs, $E \sim \rho^2$ and $\sigma_y \sim \rho^2$, while isotropic 3D foams, including isotropic CNT foams [232], exhibit $E \sim \rho^3$ and $\sigma_y \sim \rho^3$ scaling relations. Alumina-coated CNT forests have exhibited scaling on the order of scale as $E \sim \rho^{2.8}$ and $\sigma_y \sim \rho^{2.9}$ [233], slightly less than that predicted by open cell foam predictions. In a similar fashion, our scaling factors of $E_f \sim \rho^{1.9}$ and $\sigma_y \sim \rho^{1.9}$ are slightly less than ideal value of 2 for these parameters with 2D open-cell foams, providing confidence in the simulation output. We anticipate that scaling the existing simulation to three

dimensions will reflect the scaling relationships of three-dimensional open-cell foams, as observed experimentally [233].

The predicted buckling load and stiffness resulting from the regression, plotted relative to their ground-truth values, may be found in supplementary information Figure S1 in our paper. Note that the linear regression model using the known values of physical attributes (without using image properties) provides a single prediction for each of the 63 classes. A statistical linear regression model produced an R^2 value of 0.94 for both buckling load and forest stiffness, indicating that buckling load and CNT forest stiffness was predicted relatively well conditioned upon knowledge of the CNT physical attributes. The linear regression model had a root-mean-squared error (RMSE) of 0.22 and 0.20 for stiffness and buckling load, respectively, for the same data. While this linear regression approach is straightforward when using synthetic simulation because all CNT forest simulation parameters are prescribed, for real-world experiments the determination of CNT diameter, density, and growth rate variation is time consuming, cost prohibitive, and may be unrealistic to obtain with physical experiments for even modest parametric studies. These difficulties motivate the physics-based simulation and image-based prediction capabilities afforded by the deep learning techniques.

CNTNet Classification and Prediction Results

Quantifying the complex and entangled CNT forest morphology after synthesis is itself a difficult task. Because of the relatively large SEM depth of field and because adjacent CNTs frequently bundle into ropes during synthesis, even rudimentary CNT forest measurements (such as diameter distribution and CNT areal density) are difficult using SEM analysis alone. Typical metrics to describe CNT forest alignment include the Herman Orientation Factor (HOF) obtained by small angle X-ray scattering (SAXS) [207, 234] or scanning electron microscopy (SEM) imagery [223], or by hand-

crafted image-based descriptors [212]. Of these, SAXS is perhaps the most descriptive quantitative technique for CNT forest characterization, as it provides measurements of CNT alignment, diameter distribution, and areal density. This technique requires a high-intensity synchrotron photon source to penetrate the forest, and as such, it is not readily available. The SAXS measurement also acquires population-averaged metrics that may be difficult to correlate to individual CNT-CNT interactions. Nevertheless, SEM-based morphology characterization is currently the most frequent methodology for CNT forest characterization. As such, we designed a deep learning (DL) framework to learn from physics-based simulated CNT forest morphology images that are similar in scale and resolution to SEM imagery.

Deep learning architectures are a powerful approach for pattern representation learning, and are being widely adopted in many fields including natural object image classification [235], aerial scene classification [236], feature tracking in wide area motion imagery [237], 3D point cloud classification [238], vessel segmentation [239], and malaria diagnosis [240]. Recent work evaluated using hand-crafted texture features like joint adaptive median binary patterns [241, 242] combined with random forest machine learning to identify the CNT class label for twelve classes [221]. In this work, we take advantage of recent developments in deep learning architectures to increase the number of CNT classes from 12 to 63, improve the classification accuracy, and add a machine learning approach for the prediction of CNT physical attributes and mechanical properties using simulated growth imagery. The CNTNet classifier provides a discrete class label output. We use the last feature layer, before the output layer, in the CNTNet classifier as the independent feature vector. We use random forest regression trees with the 4096-D deep image-based features learned by the deep neural network (DNN) classifier in order to provide continuous real-valued estimates of the dependent mechanical properties. By employing this scheme, mechanical property predictions benefit from the descriptive power of learned deep features combined

with the accuracy and robustness of random forest to noise and over-fitting by using an ensemble of diverse trees. As shown in Figure 5.5, the proposed CNTNet classification and regression framework consists of a pipeline of several modules. The CNT forest images generated using the physics-based growth simulation model are used as input to train a VGG-19 DNN [243] to predict CNT forest attributes (Structural Classification module) and mechanical properties (Property Regression module). The finite element simulation generates CNT forest structural morphology images, along with mechanical stiffness, and buckling load values. The CNTNet classifier accepts CNT forest images as input and generates predictions of the CNT class labels along with the feature space embedding for CNT forest attributes and the mechanical properties of stiffness and buckling load. In this way, the CNTNet capability spans the entire synthesis-structure-property paradigm for CNT forest synthesis and virtual screening.

Using CNTNet the overall classification accuracy for 63 classes was 91.0%, indicating that all three attributes of CNT density, radius, and growth rate coefficient of variation were correctly predicted with high accuracy on the test images. The feature descriptor embedding from the CNTNet classification module is then used for the CNTNet regression task to predict the CNT forest physical properties. Unlike the classification-based traditional linear regression model in the previous section, for the CNTNet random forest regression predictor, the image feature descriptor embedding alone was used as the model input, without any knowledge of CNT forest attributes provided to the decision trees. The CNTNet-predicted buckling load and elastic stiffness relative to the known ground-truth values are shown in Figure 5.6(c) and (d). Note that the solid line represents an ideal 1:1 correlation between CNTNet prediction and the ground-truth parameters from the physics-based simulation. For both stiffness and buckling load, the predicted mechanical properties lie close to the ground-truth values for ranges of values that vary by greater than three orders of

magnitude with an R^2 value of 0.96 for buckling load and 0.94 for stiffness, which is better than the R^2 values of 0.94 obtained for the classification-based traditional regression with more information known *a priori*. The RMSE values produced for the CNTNet-predicted stiffness and buckling load are 0.21 and 0.17, respectively. These values are an improvement over the respective RMSE values of 0.22 and 0.20 using the previously discussed statistical linear regression predictions which utilized the numerical values of CNT density, diameter, and growth rate variation as inputs.

The superior accuracy of the CNTNet regression predictions has practical implications. As stated previously, the structural characterization of CNT forests using methods such as SAXS is resource intensive and prohibitive for a high volume of samples. Image acquisition of CNT forests using SEM, however, is readily available to most researchers. The CNTNet results demonstrate that property predictions using image data may be superior to algorithms in which all CNT forest physical attributes are known a priori. Such a result might not be surprising when considering that image data contains embedded information that is unique to each CNT forest. Whereas the classification-based regression model based upon physical attributes generates a single-value prediction for all forests with matching attributes, image-based regression detects subtle differences within CNT forest morphology to generate unique predictions even for forests that share similar attributes. Moreover, the images themselves may contain sufficient information to infer the population-based physical attributes provided by methods such as SAXS, as demonstrated by the CNTNet classification results.

Data visualization can be used to understand the CNTNet classification and regression performance in a qualitative manner. The multi-class confusion matrix in Figure 5.6(a) shows the misclassification accuracy between class C_q and C_r arranged by CNT density, in the form of a matrix with color intensity mapping to the percentage of predicted values which fall within a known ground-truth class label. The axes of

the confusion matrix represent the predicted class label and the known ground-truth classification. The diagonal entries in the confusion matrix represent the respective class accuracies, while all other entries represent a misclassification. Marginal confusion matrices are provided as supplementary material in our paper which show the classification accuracy as a function of only CNT density, only CNT radius, or only population growth rate variation. In these matrices, the classification of a given parameter level is assumed to be correct even if the other independent parameters are not correctly predicted. The lowest prediction accuracy in each of these quantities was 99.97% for CNT density, 88.99% for CNT radius, and 97.88% for growth rate coefficient of variation. The CNT density was predicted with 100% accuracy for forests having between 50 to 150 CNTs per 10 μm span. The high degree of accuracy of the CNTNet classification module demonstrates that DL techniques can readily identify subtle differences in the wavy texture morphology of CNT forests arising from the relatively small changes in CNT attributes.

The t-Distributed Stochastic Neighbor Embedding (t-SNE) [185] can be used to visualize the class clusters projected in a low-dimenstional space suitable for qualitative understanding of the class separability and inter-class mixing. This is important for the CNT classification task due to the large number of 63 classes with subtle texture differences between some classes. t-SNE reduces the dimensionality of the 4096-D feature descriptor embedding vectors learned by the VGG-19 DL network to generate a 2-D plot as shown in Figure 5.6(b). A principal component analysis (PCA) [184] was first used to reduce the feature space dimensionality of the classification embeddings from 4096 to 50 before projecting their values to a 2D space using t-SNE. The 63 CNT classes are grouped into seven density classes shown in different colors. The seven CNT density classes are well-separated in the t-SNE subspace, indicating that CNTNet has learned a feature embedding manifold that is highly discriminatory. The t-SNE distributions sorted by diameter and growth rate variation are provided in the

supplementary material in our paper and show overlapping clusters that supports our confusion matrix analysis. The misclassification rate around 9% is due to the overlap in the 9 combinations of diameter and growth rate variation within each CNT density cluster as shown in Figure S5 in supplementary file of our paper.

5.1.3 Discussion

It is striking that image-based mechanical property prediction, matched or exceeded statistical predictions in which all CNT forest attributes were known with certainty. An image-based AI/ML approach using physics-based image synthesis is particularly powerful and applicable because the physical attributes of CNT forests are rarely, if ever, known with certainty in physical experiments. The CNTNet classification-regression framework demonstrates that the feature descriptor information embedding using CNT forest structural morphology images is sufficient to characterize the relevant synthesis-related CNT attributes (CNT density, CNT diameter, and CNT growth rate variation) and ensemble properties of CNT forests with high accuracy without any other external knowledge. We anticipate that these image-based computer vision techniques, upon further refinement, can be translated to and augmented with experimentally acquired SEM images of CNT forests to rapidly characterize their physical attributes with similar high accuracy. We acknowledge that the current simulation model may only capture the first order mechanisms of CNT forest self-assembly and mechanics, but we anticipate that data from emerging experimental techniques will continuously improve the fidelity of the simulation techniques.

An image analysis system designed to characterize SEM images of CNT forests must accommodate SEM-specific imaging characteristics including molecular structure variability, measurement noise and grayscale intensity variations through the depth of imaging. We are planning to develop a new module to generate photo-realistic images of simulated CNT forests that reflect SEM appearance characteristics.

The simulated CNT forests themselves will also be generated using CNT diameter distributions consistent with specific synthesis criteria. The CNTNet classifier and CNTNet regressor will then be retrained using more realistic synthetic CNT imagery augmented with labeled SEM images as these become available. Previous work on deep learning-based realistic synthetic data generation systems [244, 245], general adversarial networks [246, 247], and synthetic microscopy image generation approaches [248, 249] will inform our generation of more realistic synthetic CNT forest images, characterizing the generalization power of CNTNet and for translating the proposed pipeline to the analysis of SEM images.

High-throughput simulation is expected to play a major role in mapping the synthesis-structure-property relationships of CNT forests for virtual screening. The current study did not correlate the prescribed CNT synthesis attributes of diameter, areal density, and growth rate variation to specific process parameters such as catalyst composition, catalyst thickness, temperature, and gas flow rate; however, these correlative studies are being conducted, and limited empirical data exists within the literature. Precise determination of CNT catalyst kinetics is expected to accelerate in the coming years using in-situ transmission electron microscope and SEM synthesis observations. As more complete experimental data is obtained, the relevant physical relationships between process parameters and CNT synthesis attributes can be integrated to improve the physics in the simulation framework and generate increasingly accurate simulations with little additional computational resources. The exploration of the nearly inexhaustible CNT forest parameter space will accelerate while becoming significantly less cost prohibitive.

Future advancements of CNTNet lie in the incorporation of experimental and numerical simulation data as separate information sources, but coordinated data streams. SEM-based in-situ synthesis experiments are currently being conducted that will both validate the physics and kinetics of the finite element simulation and

will provide a source of CNT forest SEM image inputs. The in-situ experiments will provide an opportunity to approximate the ground truth and to precisely correlate simulation and experiment, which will allow evaluating the generalization capability of the CNTNet ML model. The implementation of 3D finite element code, analogous to the 2D code presented here, is ongoing and will produce CNT forest morphology imagery that is even more analogous to CNT forest SEM imagery. As quantification and understanding of increasingly complex CNT forest growth kinetics are understood and reported, simulation fidelity will continually increase. In the near future, we expect that a well-validated CNT forest simulation tool could be used to autonomously navigate the CNT forest synthesis parameter space to rapidly map the available combinations of CNT forest ensemble physical properties afforded by diverse CNT forests morphology, particularly those which have not been previously explored. These properties could then be confirmed and refined experimentally. Similarly, robust image-based characterization of CNT forest physical attributes will be enabled when a repository of well-characterized CNT forest SEM images becomes available. The tools developed and reported here are foundational to these efforts and establish that image-based DL can both classify CNT forests and predict their physical properties with high accuracy.

5.1.4 Methods

Physics-Based CNT Growth and Compression Simulation for Machine Learning

All finite element simulations were conducted using MATLAB 2018b software running a custom FEM code. CNT elements were treated as Euler-Bernoulli frame elements with a computational node at each end of the element, as described elsewhere [218]. The van der Waals interactions between adjacent CNT segments were simulated as

linear-elastic spring elements. As neighboring CNTs contact, the van der Waals interactions may lead to the formation of bundles or ropes of multiple CNTs that often persist throughout the height of forests. Likewise, contacting CNTs may bend and form a wavy morphology. Each unique morphology is shaped by the mechanical equilibrium established between the contacting CNTs. Mechanical equilibrium was computed at each time step for all nodes within the CNT population before initiating a new set of elements at the base of existing CNTs at discrete time steps to simulate CNT growth. To accommodate the new CNT elements, the nodes that had previously resided at the bottom of the forest were displaced by a distance corresponding to the growth rate of each CNT and at an angle consistent with the orientation angle of the CNT. The stiffness matrix is regenerated at the beginning of each time step to account for the new orientation of each element, the formation of new CNT-CNT contact points, and the addition of new elements to the system. A horizontal growth span of 10 μm was selected for all simulations, with periodic boundary conditions at the horizontal extremes of the simulation domain to mitigate potential edge effects [217]. The population-averaged growth rate was 60 nm per time step for all simulations. Diverse CNT forest populations were achieved by selecting among 7 CNT population densities (50 - 200 CNTs in increments of 25), 3 CNT outer radii (5, 8, 11 nm) and 3 values of CNT population growth rate coefficient of variation (3, 6, 9%), for a total of 63 distinct classes. The inner radius of each CNT was 70% that of the outer radius. Because the characteristics assigned to each CNT within a forest were chosen stochastically, each simulation produced a unique CNT forest, even within a common class. Each synthesis simulation was terminated once all CNTs in the population reached a height of 20 μm, and the growth image of the CNT forest morphology was saved for deep learning analysis.

Using the same simulation framework, the mechanical compression of each forest was simulated by vertically translating a rigid horizontal surface at 20 nm increments

per time step. Each CNT node in contact with the simulated moving boundary was pinned, and the base node of each CNT was fixed, simulating a rigid substrate. Mechanical equilibrium was computed for each displacement step, and the compressive load was determined by summing the vertical reaction forces of all CNTs in contact with the moving surface. This force was equal in magnitude to the summed force transmitted to the simulated substrate. The plastic deformation of constituent CNTs was considered by employing the Brazier instability mechanism for thin-walled cylinders in which the circular cross section of CNTs forms a flat kink [250]. For hollow CNTs, kinking of the original circular cross section reduces the bending stiffness, and the cylinder can support less of a bending moment. The critical kinking moment can be expressed as,

$$M_{kink} = 0.4683 \frac{EDt^2}{\sqrt{1-\nu^2}} \tag{5.1}$$

for a thin walled cylinder, where E is the CNT modulus, D is the CNT outer diameter, t is the CNT wall thickness, and ν is the Poisson's ratio (0.17 for graphite). This critical limit was experimentally validated for CNTs using in-situ TEM deflection and assuming a thickness of 2 walls where the maximum moment was generated [250]. The experimentally observed kinking moment was 25% greater than this approximation, as inner-most CNT walls acted to strengthen the deforming cross section. In our simulation, the full wall thickness of the CNT was considered when computing the kinking moment of an element. When an element exceeds the critical kinking moment, its moment of inertia is reduced by a factor of 10^5 to approximate the limitation of the kinked cross-section to support mechanical moments. The elastic modulus of each kinked segment is reduced by a factor of 10^3. These values are estimates and characterize the inability of a kinked CNT to support additional load or mechanical moment.

CNTNet: Deep Learning and ML Regression Using a Physics-Based Virtual Laboratory for Data Generation

The theoretical understanding of CNT morphology, structure and physical properties is limited, and the experimental generation of a large collection of real SEM imagery that is essential for training deep learning networks is unavailable due to difficult material synthesis, high cost, and high resource requirements [197]. Therefore, we investigated the feasibility that a deep learning architecture trained using *only simulated imagery* from the physics-based simulation virtual laboratory model could be used for virtual screening [251]. Success using a virtual laboratory screening method would provide confidence for pursuing a computational automated materials design approach [252] using SEM imagery from physical experiments to accelerate the guided search for optimal CNT materials with desirable properties.

We developed CNTNet a deep network that transfers the high-dimensional texture-based feature classification embeddings to predict CNT physical properties using machine learning-based random forest regression. CNTNet is composed of two modules as shown in Figure 5.5. The first *structure classification module* is the CNT image representation embedding deep architecture, which is learned using a classification task network. The second *property regression module* uses Random Forest regression (RF regression) [253, 254, 255, 256] for CNT physical property prediction using supervision from the physics-based simulation system. Random forests provide a unified framework for manifold learning [253], interpretability in the context of explainable AI [257], better robustness to adversarial noise, and randomization in RF has been shown to be a powerful way to learn distances between images of never-seen objects [258]. We use the feature space embedding learned during K-way classification to boost the prediction power of the *property regression module* CNTNet. Once the deep network is trained on the visual appearance-based structure classification task, all the deep learning parameters or weights θ_C are frozen, so that the embedding rep-

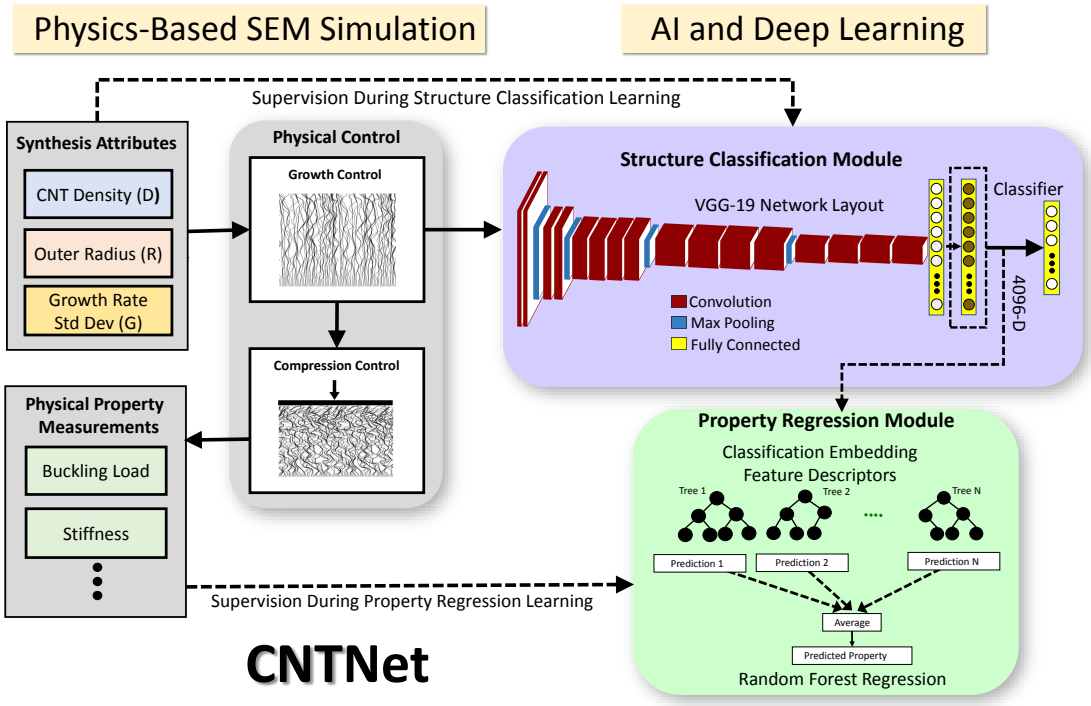


Figure 5.5: **Schematic of CNTNet modules.** CNTNet is a modular image-based machine learning and control framework for enabling artificial intelligence-driven classification of CNT forest synthesis attribute groups and predictions of CNT forest material properties. The top *structure classification module* uses the VGG-19 deep learning network for classifying CNT forests into one of 63 CNT classes based on combinations of synthesis attributes generating different CNT forest simulated SEM image classes. The structure classification network captures morphological texture properties and was trained using transfer learning with initialization using ImageNet weights. The bottom *property regression module* uses Random Forest regression trees to learn CNT forest buckling load and stiffness by mapping the 4096-D feature image representation descriptor using 2-D regression vector functions.

resentation learned through structure classification can be transferred to the property prediction module.

The CNTNet framework is motivated by the observation that CNT forest properties are strongly correlated with their class grouping (x-axes) and physical attributes of buckling load and stiffness, as depicted in Figure 5.4. Using the structure classification deep learning module, the CNTNet extracts a rich high-dimensional feature descriptor that encodes the prior knowledge of CNT physical attributes in an embedding manifold, that can then be used to learn additional physical properties in

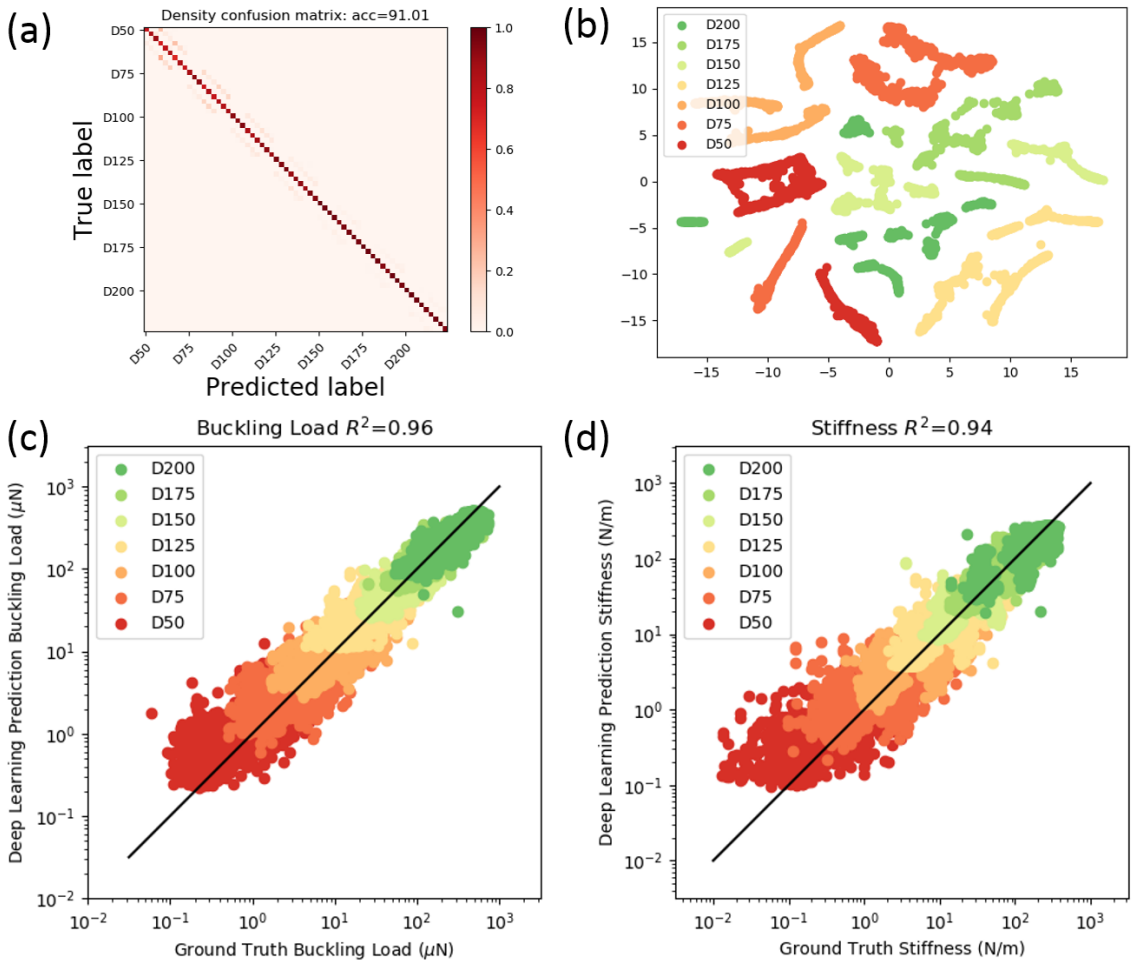


Figure 5.6: **CNTNet classification and regression accuracy.** (a) Confusion matrix for 63 CNT classes with the ordering based on seven CNT densities and nine sub-types. (b) t-SNE visualization of the 63 CNT forest classes grouped by the 7 CNT population densities for better visualization. The separation of the density groups using the VGG-19 feature embedding sub-space indicates that the CNT forest density is classified with high accuracy. CNTNet regression prediction for (c) buckling load and (d) stiffness compared to the simulated ground-truth values. Data are colored according to the CNT forest linear density.

the descriptor space, and leads to the observed strong performance of CNTNet for predicting the physical properties. The feature embedding provided by the VGG-19 network was used for both determining the CNT forest class label based on simulated growth attributes, and as input to a random forest regression estimator to predict the mechanical properties.

The *structure classification module* in CNTNet uses physics-based simulated CNT

forest structural imagery as input to predict the CNT attributes of CNT diameter, CNT population growth rate variation, and CNT density based on the 63 class groupings. Each class represents a unique (discrete) combination of these parameters. Given the morphology images $\mathbf{X}_i$ of CNT structures, $\mathbf{X}_i \in \mathbf{X}_{CNT}$, where $\mathbf{X}_{CNT}$ is the training set of CNT images. In this work, we used the VGG-19 CNN to learn the mapping function from CNT structure to CNT forest classes. We define $f_{\theta_C}(\cdot)$ as the mapping function from the input (image) space $\mathbf{X}_i$ to the class label space p_i, parameterized by the classification network weights, θ_C, which is then followed by the $Softmax(\cdot)$ non-linear operator, defined as:

$$p_i = Softmax(f_{\theta_C}(\mathbf{X}_i)), Softmax(z_{i,k}) = \frac{e^{z_{i,k}}}{\sum_{j=1}^{K} e^{z_{i,j}}}, \text{ for } j = 1, ...K \tag{5.2}$$

where $\mathbf{X}_i$ is the i^{th} input CNT image, and p_i is the associated CNTNet predicted class label for $\mathbf{X}_i$. After applying the softmax operation to, $z_{i,k}$, the output of the VGG-19 network estimates K real-valued class probabilities. Note that $Softmax(\cdot)$ is a differentiable version of the maximum operator. During training we minimize the cross-entropy loss J defined as,

$$J = -\frac{1}{N}\Sigma_{i=1}^{N}\ \Sigma_{k=1}^{K}\ y_{i,k}\log(p_{i,k}) \tag{5.3}$$

where $y_{i,k}$ is the element in y_i of k^{th} class, and y_i is the true label (1 for the target class and 0 for the other $(K-1)$ classes), $p_{i,k}$ is the predicted probability of label k for input $\mathbf{X}_i$, and N is the mini-batch size during training. In our experiments, we used VGG-19 [243] architecture as the deep learning backbone network, and Stochastic Gradient Decent (SGD) optimization to learn the optimal parameter space θ_c.

The classification embedding for the i^{th} input (image) $\mathbf{X}_i$ is defined as $\mathbf{d}_i^C \in \mathbb{R}^E$,

where E is the dimensionality of the classification embedding vector,

$$\mathbf{d}_i^C = f_{\theta_C}(\mathbf{X}_i) \tag{5.4}$$

where $f_{\theta_C}(\cdot)$ is the classification mapping to identify the CNT class type of the input image defined in Eq 5.2. Next, we learn $f_{\theta_R}(\cdot)$ the unknown mapping function from the CNT structure classification space to the CNT physical property space that is parameterized by the RF regression variables θ_R. We define the predicted CNT physical property as the output vector $\hat{\mathbf{y}}_i \in \mathbb{R}^2$,

$$\hat{\mathbf{y}}_i = f_{\theta_R}(\mathbf{d}_i^C) \tag{5.5}$$

where one output dimension predicts CNT buckling load, and the other output predicts stiffness. The reason we learn a single RF regression to predict both properties of buckling load and stiffness jointly is due to their physically coupled relationship that may lie in the same manifold embedding. A figure showing the correlation between buckling load and stiffness may be found in the supplementary file (Figure S3) in our paper. This approach is further strengthened by the general behavior of open-cell foams which consistently yield at an engineering strain of approximately 5% [259].

In our experiments, 60% of simulated images were used for training, 20% for validation, and 20% for testing. For the CNT classification label assignment task, the pyTorch deep learning framework was used to train a VGG-19 backbone [243] for image texture-based CNT structure categorization. For the physical property prediction task, we used the image texture embedding from the VGG-19 structural classification results to develop a random forest regression model using the Scikit-Learn software [260]. The size of our generated images was 907×725 pixels. During training, the images are resized to 256×256 pixels initially, followed by the random crop augmentation to extract sub-images that were 224×224 pixels. Later, during

validation and testing, we first resized images to 256×256 pixels, and then performed center cropping of each image to be the same 224×224 pixels.

The hyperparameters for training the VGG-19 deep learning network used a stochastic gradient optimization approach with an initial learning rate value of 10^{-3}, a weight decay of 10^{-4} and momentum of 0.9 for updating the gradient at each iteration, and a mini-batch-size of 32 for gradient estimation. Common image data augmentation methods were used to enrich the amount and variety of training data, including random cropping to extract subimages, and random flip with 0.5 probability. All experiments were conducted in 5-Fold cross-validation settings. We trained the deep network for 200 epochs and used the classification accuracy on the validation data to select the best model parameters. The validated VGG-19 learned network model was then evaluated using the testing data to avoid overfitting. The 4096-dimension feature vector embedding from the last fully connected layer in the VGG-19 classification model was used to train the random forest regressor decision trees.

The *random forest regression module* in CNTNet uses the RF approach which builds an ensemble of regression trees with randomized feature selection to learn the quantitative relationship between elastic stiffness and buckling load values. When training the regressor, the outputs are regressed to their $\log_e$ values. The Scikit-learn library [260] was used to learn the RF regression. In total, 1000 decision trees of varying depth were learned in the RF ensemble, all the 4096 features were considered at each node split, and with pure leaf nodes as the stopping condition during tree construction. The RF vector regressor is trained to predict two properties, buckling load and stiffness after a logarithmic transformation to handle the five order of magnitude range in property values (see Figure 5.6 and Supplementary Figure S3 in our paper).

5.1.5 Evaluation Metrics

To evaluate the performance of CNT forest classification, the standard overall accuracy (OA) metric was used,

$$OA = \frac{\sum_{k=1}^{K} C_{k,k}}{\sum_{q=1}^{K} \sum_{r=1}^{K} C_{q,r}} \tag{5.6}$$

where K is the number of classes, the $K \times K$ confusion matrix with diagonal terms being the correct classification probabilities and off-diagonal entries, $C_{q,r}$, are the misclassification probabilities between the ground-truth class q predicted to belong to class r.

To quantify the prediction of CNT forest mechanical properties, we use the coefficient of determination R^2 and Root Mean Square Error ($RMSE$) metrics. The R^2 score is computed as,

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{5.7}$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{5.8}$$

where y_i is the true CNT forest property value, $\hat{y}_i$ is the predicted CNT forest property value, and n is number of samples. The RMSE metric between the true and predicted property values is computed as,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{5.9}$$

5.2 DMNet for Cell Segmentation

Cell segmentation is an important task in biomedical domain. The challenges of segmenting cells include cells are small and low resolution, and many cells are clustered together. To tackle these challenges, we propose a dual-stream marker guided network (DMNet). Our network won the second place on ISBI 2021 sixth Cell Tracking Challenge (CTC, http://celltrackingchallenge.net/).

There are two networks in our algorithm, one network is designed for cell centroids (markers) detection, and the other is designed for mask detection. We use the HRNet [261] as the CNN model to learn the centroid localization and cell segmentation mask. Transform distance maps are computed in training and used for penalizing the boundary region for accurate cell segmentation. Centroids are used to locate the cells and segment the segmentation mask to multiple cells.

Table 5.1: Performance of DMNet on ISBI 2021 6th Cell Segmentation & Tracking Challenge.

Dataset	BF-C2DL -HSC	BF-C2DL -MuSC	DIC-C2DH -HeLa	Fluo-C2DL -MSC	Fluo-N2DH -GOWT1	Fluo-N2DL -HeLa	PhC-C2DH -U373	PhC-C2DL -PSC
OP_{CTB}	0.828	0.849	0.854	0.591	0.939	0.953	0.947	0.821
	6/10	**2/10**	6/19	12/26	**2/35**	**1/33**	**3/24**	4/26
SEG	0.699	0.742	0.802	0.522	0.931	0.923	0.923	0.708
	6/10	**2/10**	6/19	13/26	**1/35**	**1/33**	**3/24**	4/26
TRA	0.957	0.957	0.907	0.661	0.946	0.983	0.972	0.933
	4/10	4/10	9/19	12/26	7/35	10/33	11/24	8/26

The input images are pre-processed contrast adjustments with z-score distribution. For the training process, marker localization is trained with centroid supervision, and segmentation mask is supervised by silver truth of annotations. Both the marker localization network and cell segmentation network are trained on eight 2D datasets and five 3D datasets with an input size of 256×256 with distance penalty loss. During training, regular data augmentation strategies including rotation, flip, and scale from 0.8 to 1.5 are applied for each sample. During inference, both centroids and segmentation masks are generated, and then the morphology operations are used to split cells

guided by our generated centroids. We report the DMNet performance on CTC challenge benchmarks in Table 5.1. For details of evaluation metrics, please refer to the CTC challenge website and [262]. For detailed descriptions of our algorithm, please refer to MU-BA-US (http://celltrackingchallenge.net/participants/MU-Ba-US/). Our DMNet ranked top 3 in four datasets among the eight 2D cell segmentation datasets.

Chapter 6

Summary and Concluding Remarks

I have presented my work towards designing deep learning architectures for 2D and 3D challenges in scene perception.

For 3D point cloud understanding tasks, we develop several networks. We propose the novel dual-stream networks GLSNet and its extension GLSNet++ to capture multiscale structural information and handle large variations in object sizes typical of urban scenes. The proposed GLSNet++ incorporates a unique and effective graph convolutional network for demixing or unmixing voxel boundary regions composed of a mixture of object classes using spatial context-dependent feature fusion similar to conditional random fields. We propose PointGrad, a new graph convolution gradient operator. The PointGrad encodes point-based directional gradients into a high-dimensional multiscale tensor space to capture the salient structural information in the point cloud across spatial and feature scale space, enabling efficient learning. Integrating PointGrad with several deep network architectures demonstrates the the efficiency, effectiveness, and robustness of PointGrad for large-scale 3D point cloud segmentation, indoor scene segmentation, and object part segmentation.

For 2D recognition tasks, we propose MDXNet in emulation of the human visual system, which learns embeddings of texture visual primitives, shape visual primitives,

and object-appearance visual primitives from nature image domains to capture human explainable perceptual latent attributes and integrates them for tasks in multiple new application domains. The proposed MDXNet is successfully extended to tasks in zero-shot learning. We applied deep networks in multiple application domains, including the material science domain and biomedical domain.

All the above methods demonstrate favorable results compared with other state-of-the-art deep learning and few-shot learning methods. In the future, we plan to extend PointGrad to various 3D tasks such 3D point cloud denoising, 3D point cloud completion. Besides, the MDXNet can be extended to more application domains. We also plan to extend MDXNet to tackle the long-tail problem in AI.

Bibliography

[1] R. Bao, K. Palaniappan, Y. Zhao, G. Seetharaman, and W. Zeng. GLSNet: Global and local streams network for 3D point cloud classification. In *IEEE Applied Imagery Pattern Recognition Workshop*, pages 1–9, 2019.

[2] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics*, 35(6):1–12, 2016.

[3] Y. Lian, T. Feng, J. Zhou, M. Jia, A. Li, Z. Wu, L. Jiao, M. Brown, G. Hager, N. Yokoya, R. Hänsch, and B. L. Saux. Large-scale semantic 3-D reconstruction: Outcome of the 2019 IEEE GRSS data fusion contest—Part B. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:1158–1170, 2021.

[4] John P Frisby and James V Stone. *Seeing: The computational approach to biological vision*. The MIT Press, 2010.

[5] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[6] W Stolz. Abcd rule of dermatoscopy: a new practical method for early recognition of malignant melanoma. *Eur. J. Dermatol.*, 4:521–527, 1994.

[7] Naheed R Abbasi, Helen M Shaw, Darrell S Rigel, Robert J Friedman, William H McCarthy, Iman Osman, Alfred W Kopf, and David Polsky. Early diagnosis of cutaneous melanoma: revisiting the abcd criteria. *Jama*, 292(22):2771–2776, 2004.

[8] Giuseppe Argenziano, Gabriella Fabbrocini, Paolo Carli, Vincenzo De Giorgi, Elena Sammarco, and Mario Delfino. Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions: comparison of the abcd rule of dermatoscopy and a new 7-point checklist based on pattern analysis. *Archives of Dermatology*, 134(12):1563–1570, 1998.

[9] TB Fitzpatrick, JD Bernhard, TG Cropley, et al. The structure of skin lesions and fundamentals of diagnosis. *Dermatology in General Medicine*, 5:13–41, 1999.

[10] Gregory S Barsh. What controls variation in human skin color? *PLoS Biol*, 1(1):e27, 2003.

[11] Marc Bosch, Kevin Foster, Gordon Christie, Sean Wang, Gregory D Hager, and Myron Brown. Semantic stereo for incidental satellite images. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1524–1532. IEEE, 2019.

[12] Bertrand Le Saux, Naoto Yokoya, Ronny Hansch, Myron Brown, and Greg Hager. 2019 data fusion contest [technical committees]. *IEEE Geoscience and Remote Sensing Magazine*, 7(1):103–105, 2019.

[13] 2019 IEEE GRSS Data Fusion Contest. `http://www.grss-ieee.org/community/technical-committees/data-fusion/data-fusion-contest/`.

[14] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 424–432. Springer, 2016.

[15] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[16] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018.

[17] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.

[18] Fabian Groh, Patrick Wieschollek, and Hendrik PA Lensch. Flex-convolution. In *Asian Conference on Computer Vision*, pages 105–122. Springer, 2018.

[19] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In *European Conference on Computer Vision*, pages 87–102, 2018.

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[22] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3D segmentation of point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018.

[23] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.

[24] Min Zhang, Haoxuan You, Pranav Kadam, Shan Liu, and C-C Jay Kuo. PointHop: An explainable machine learning method for point cloud classification. *IEEE Transactions on Multimedia*, 22(7):1744–1755, 2020.

[25] Truc Le and Ye Duan. PointGrid: A deep network for 3D shape understanding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9204–9214, 2018.

[26] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[27] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):11, 2019.

[28] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120(14):145301, 2018.

[29] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[30] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019.

[31] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *IEEE International Conference on Computer Vision*, pages 9267–9276, 2019.

[32] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[33] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019.

[34] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[35] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. PartNet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9491–9500, 2019.

[36] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

[37] Alexandre Boulch. Generalizing discrete convolutions for unstructured point clouds. *arXiv preprint arXiv:1904.02375*, 2019.

[38] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[39] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[40] Yanchao Lian, Tuo Feng, and Jinliu Zhou. A dense PointNet++ architecture for 3D point cloud semantic segmentation. In *IEEE IGARSS*, pages 5061–5064, 2019.

[41] Meixia Jia, Aijin Li, and Zhaoyang Wu. A global Point-SIFT attention network for 3D point cloud semantic segmentation. In *IEEE IGARSS*, pages 5065–5068, 2019.

[42] 2019 IEEE GRSS Data Fusion Contest Baseline. `http://https://github.com/pubgeo/dfc2019/tree/master/track4/`.

[43] Timo Hackel, N. Savinov, L. Ladicky, Jan D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals Photogrammetry, Remote Sensing, Spatial Info. Sci.*, volume IV-1-W1, pages 91–98, 2017.

[44] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3D point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[45] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Transactions on Graphics*, 36(4):1–11, 2017.

[46] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[47] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[48] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[49] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[50] Truc Le, Giang Bui, and Ye Duan. A multi-view recurrent neural network for 3d mesh segmentation. *Computers and Graphics*, 66:103–112, 2017.

[51] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3779–3788, 2017.

[52] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *IEEE International Conference on Computer Vision*, pages 945–953, 2015.

[53] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[54] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. SyncSpecCNN: Synchronized spectral cnn for 3D shape segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2282–2290, 2017.

[55] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4548–4557, 2018.

[56] Xiaojuan Qi, Renjie Liao, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. 3D graph neural networks for RGBD semantic segmentation. In *IEEE International Conference on Computer Vision*, pages 5199–5208, 2017.

[57] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3693–3702, 2017.

[58] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.

[59] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[60] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[61] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3D point clouds. In *IEEE International Conference on Computer Vision*, pages 537–547, 2017.

[62] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[63] Sanjay Surendranath Girija. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *Software available from tensorflow. org*, 39(9), 2016.

[64] Diederik P Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[66] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.

[67] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018.

[68] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SplatNet: Sparse lattice networks for

point cloud processing. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[69] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[70] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[71] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3D point cloud classification and segmentation using 3D modified Fisher Vector representation for convolutional neural networks. *arXiv preprint arXiv:1711.08241*, 2017.

[72] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.

[73] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, volume 70, pages 1126–1135, 2017.

[74] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *International Conference on Machine Learning Workshop*, volume 2, 2015.

[75] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[76] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[77] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[78] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[79] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3018–3027, 2017.

[80] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7278–7286, 2018.

[81] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[82] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, volume 37, pages 1180–1189, 2015.

[83] Nanqing Dong and Eric P Xing. Domain adaption in one-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 573–588, 2018.

[84] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943, 2013.

[85] Eli Schwartz, Leonid Karlinsky, Rogerio Feris, Raja Giryes, and Alex M Bronstein. Baby steps towards few-shot learning with multiple semantics. *arXiv preprint arXiv:1906.01905*, 2019.

[86] Yaqing Wang and Quanming Yao. Few-shot learning: A survey. *arXiv preprint arXiv:1904.05046*, 2019.

[87] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems*, pages 2255–2265, 2017.

[88] Zhongwen Xu, Linchao Zhu, and Yi Yang. Few-shot object recognition from machine-labeled web images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1164–1172, 2017.

[89] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision*, pages 616–634. Springer, 2016.

[90] Yu-Xiong Wang and Martial Hebert. Learning from small sample sets by combining unsupervised meta-training with cnns. In *Advances in Neural Information Processing Systems*, pages 244–252, 2016.

[91] Mark Woodward and Chelsea Finn. Active one-shot learning. *arXiv preprint arXiv:1702.06559*, 2017.

[92] Yao-Hung Hubert Tsai, Liang-Kang Huang, and Ruslan Salakhutdinov. Learning robust visual-semantic embeddings. In *IEEE International Conference on Computer Vision*, pages 3591–3600, 2017.

[93] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[94] Yunhui Guo, Noel C Codella, Leonid Karlinsky, James V Codella, John R Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning. In *European Conference on Computer Vision*, pages 124–141. Springer, 2020.

[95] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[96] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. In *International Conference on Learning Representations*, 2020.

[97] Piotr Koniusz, Yusuf Tas, Hongguang Zhang, Mehrtash Harandi, Fatih Porikli, and Rui Zhang. Museum exhibit identification challenge for the supervised domain adaptation and beyond. In *European Conference on Computer Vision*, pages 788–804, 2018.

[98] Nanqing Dong and Eric P Xing. Domain adaption in one-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 573–588. Springer, 2018.

[99] Yunhui Guo, Noel CF Codella, Leonid Karlinsky, John R Smith, Tajana Rosing, and Rogerio Feris. A new benchmark for evaluation of cross-domain few-shot learning. *arXiv preprint arXiv:1912.07200*, 2019.

[100] Jiang Lu, Pinghua Gong, Jieping Ye, and Changshui Zhang. Learning from very few samples: A survey. *arXiv preprint arXiv:2009.02653*, 2020.

[101] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785, 2009.

[102] Vittorio Ferrari and Andrew Zisserman. Learning visual attributes. In *Advances in Neural Information Processing Systems*, pages 433–440, 2008.

[103] Devi Parikh and Kristen Grauman. Relative attributes. In *IEEE International Conference on Computer Vision*, pages 503–510, 2011.

[104] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.

[105] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943, 2013.

[106] Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, and Timothy M Hospedales. Actor-critic sequence training for image captioning. *arXiv preprint arXiv:1706.09601*, 2017.

[107] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–58, 2016.

[108] Jimmy Lei Ba, Kevin Swersky, Sanja Fidler, et al. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *IEEE International Conference on Computer Vision*, pages 4247–4255, 2015.

[109] Zhenyong Fu, Tao Xiang, Elyor Kodirov, and Shaogang Gong. Zero-shot object recognition by semantic manifold distance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2644, 2015.

[110] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2021–2030, 2017.

[111] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2927–2936, 2015.

[112] Wen Tang, Ashkan Panahi, and Hamid Krim. Joint concept matching based learning for zero-shot recognition. *arXiv preprint arXiv:1906.05879*, 2019.

[113] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2013.

[114] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[115] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Unsupervised domain adaptation for zero-shot learning. In *IEEE International Conference on Computer Vision*, pages 2452–2460, 2015.

[116] Seyed Mohsen Shojaee and Mahdieh Soleymani Baghshah. Semi-supervised zero-shot learning by a clustering-based approach. *arXiv preprint arXiv:1605.09016*, 2016.

[117] Soravit Changpinyo, Wei-Lun Chao, Boqing Gong, and Fei Sha. Synthesized classifiers for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5327–5336, 2016.

[118] Yao Lu. Unsupervised learning on neural network outputs: with application in zero-shot learning. *arXiv preprint arXiv:1506.00990*, 2015.

[119] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via joint latent similarity embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6034–6042, 2016.

[120] Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161, 2015.

[121] Yanwei Fu, Timothy M Hospedales, Tao Xiang, Zhenyong Fu, and Shaogang Gong. Transductive multi-view embedding for zero-shot recognition and annotation. In *European Conference on Computer Vision*, pages 584–599. Springer, 2014.

[122] Yongxin Yang and Timothy M Hospedales. A unified perspective on multi-domain and multi-task learning. *arXiv preprint arXiv:1412.7489*, 2014.

[123] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531, 2010.

[124] Yutaro Shigeto, Ikumi Suzuki, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. Ridge regression, hubness, and zero-shot learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–151. Springer, 2015.

[125] Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. Improving zero-shot learning by mitigating the hubness problem. *arXiv preprint arXiv:1412.6568*, 2014.

[126] Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Annual Meeting of the Association for Computational Linguistics*, pages 270–280, 2015.

[127] Catherine Wah, Steve Branson, Pietro Perona, and Serge Belongie. Multiclass recognition and part localization with humans in the loop. In *IEEE International Conference on Computer Vision*, pages 2524–2531, 2011.

[128] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The SUN attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81, 2014.

[129] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[130] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems*, 19:601–608, 2006.

[131] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[132] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[133] Han-Kai Hsu, Chun-Han Yao, Yi-Hsuan Tsai, Wei-Chih Hung, Hung-Yu Tseng, Maneesh Singh, and Ming-Hsuan Yang. Progressive domain adaptation for object detection. In *IEEE Winter Conference on Applications of Computer Vision*, pages 749–757, 2020.

[134] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[135] Sinno Jialin Pan, James T Kwok, Qiang Yang, et al. Transfer learning via dimensionality reduction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 8, pages 677–682, 2008.

[136] Jun Yang, Rong Yan, and Alexander G Hauptmann. Adapting svm classifiers to data with shifted distributions. In *IEEE International Conference on Data Mining Workshops*, pages 69–76, 2007.

[137] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3712–3722, 2018.

[138] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(9):1790–1802, 2015.

[139] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *IEEE International Conference on Computer Vision*, pages 843–852, 2017.

[140] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[141] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes ImageNet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[142] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. In *Advances in Neural Information Processing Systems*, pages 2178–2186, 2011.

[143] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[144] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18, 2013.

[145] Li Niu, Wen Li, and Dong Xu. Multi-view domain generalization for visual recognition. In *IEEE International Conference on Computer Vision*, pages 4193–4201, 2015.

[146] Li Niu, Wen Li, and Dong Xu. Visual recognition by learning from web data: A weakly supervised domain generalization approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2783, 2015.

[147] Aditya Khosla, Tinghui Zhou, Tomasz Malisiewicz, Alexei A Efros, and Antonio Torralba. Undoing the damage of dataset bias. In *European Conference on Computer Vision*, pages 158–171. Springer, 2012.

[148] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision*, pages 5542–5550, 2017.

[149] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018.

[150] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advances in Neural Information Processing Systems*, pages 5334–5344, 2018.

[151] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pages 998–1008, 2018.

[152] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[153] Yiying Li, Yongxin Yang, Wei Zhou, and Timothy M Hospedales. Feature-critic networks for heterogeneous domain generalization. *International Conference on Machine Learning*, 2019.

[154] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3606–3613, 2014.

[155] Barbara Caputo, Eric Hayman, and P Mallikarjuna. Class-specific material categorisation. In *IEEE International Conference on Computer Vision*, volume 2, pages 1597–1604, 2005.

[156] Lavanya Sharan, Ruth Rosenholtz, and Edward Adelson. Material perception: What can you see in a brief glance? *Journal of Vision*, 9(8):784–784, 2009.

[157] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3479–3487, 2015.

[158] Jia Xue, Hang Zhang, Kristin Dana, and Ko Nishino. Differential angular imaging for material recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 764–773, 2017.

[159] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE Conference on Computer Vision and Pattern Recognition Worshop*, pages 178–178, 2004.

[160] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[161] Jianzhong He, Shiliang Zhang, Ming Yang, Yanhu Shan, and Tiejun Huang. Bi-directional cascade network for perceptual edge detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3828–3837, 2019.

[162] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[163] Evgeniy Bart and Shimon Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 672–679. IEEE, 2005.

[164] Antonio Torralba, Kevin P Murphy, and William T Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007.

[165] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):453–465, 2013.

[166] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.

[167] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[168] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7:1419, 2016.

[169] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2097–2106, 2017.

[170] Xiaoxiao Sun, Jufeng Yang, Ming Sun, and Kai Wang. A benchmark for automatic visual classification of clinical skin disease images. In *European Conference on Computer Vision*, pages 206–222. Springer, 2016.

[171] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[172] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[173] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850, 2016.

[174] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. *arXiv preprint arXiv:1712.09926*, 2017.

[175] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 10657–10665, 2019.

[176] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *International Conference on Learning Representations*, 2013.

[177] Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via semantic similarity embedding. In *IEEE International Conference on Computer Vision*, pages 4166–4174, 2015.

[178] Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 69–77, 2016.

[179] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1425–1438, 2015.

[180] Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3174–3183, 2017.

[181] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5542–5551, 2018.

[182] Long Chen, Hanwang Zhang, Jun Xiao, Wei Liu, and Shih-Fu Chang. Zero-shot visual recognition using semantics-preserving adversarial embedding networks.

In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1043–1052, 2018.

[183] Kai Li, Martin Renqiang Min, and Yun Fu. Rethinking zero-shot learning: A conditional visual classification perspective. In *IEEE International Conference on Computer Vision*, pages 3583–3592, 2019.

[184] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[185] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[186] Elicia Maine and Elizabeth Garnsey. Commercializing generic technology: The case of advanced materials ventures. *Research Policy*, 35(3):375–393, 2006.

[187] Sumio Iijima. Helical microtubules of graphitic carbon. *Nature*, 354(6348):56, 1991.

[188] Michael F. L. De Volder, Sameh H. Tawfick, Ray H. Baughman, and A. John Hart. Carbon nanotubes: Present and future commercial applications. *Science*, 339(6119):535–539, 2013.

[189] Benjamin D Jensen, Jae-Woo Kim, Godfrey Sauti, Kristopher E Wise, Liang Dong, Haydn NG Wadley, Jin Gyu Park, Richard Liang, and Emilie J Siochi. Toward ultralight high-strength structural materials via collapsed carbon nanotube bonding. *Carbon*, 156:538–548, 2020.

[190] Brian L Wardle, Diego S Saito, Enrique J Garcia, A John Hart, Roberto Guzman de Villoria, and Eric A Verploegen. Fabrication and characterization of ultrahigh-volume-fraction aligned carbon nanotube–polymer composites. *Advanced Materials*, 20(14):2707–2714, 2008.

[191] Lijun Liu, Jie Han, Lin Xu, Jianshuo Zhou, Chenyi Zhao, Sujuan Ding, Huiwen Shi, Mengmeng Xiao, Li Ding, Ze Ma, et al. Aligned, high-density semiconducting carbon nanotube arrays for high-performance electronics. *Science*, 368(6493):850–856, 2020.

[192] Shaghayegh Shajari, Mehdi Mahmoodi, Mahmoud Rajabian, Kunal Karan, Uttandaraman Sundararaj, and Les Jozef Sudak. Highly sensitive and stretchable carbon nanotube/fluoroelastomer nanocomposite with a double-percolated network for wearable electronics. *Advanced Electronic Materials*, 6(2):1901067, 2020.

[193] Myounggu Park, Baratunde A Cola, Thomas Siegmund, Jun Xu, Matthew R Maschmann, Timothy S Fisher, and Hyonny Kim. Effects of a carbon nanotube layer on electrical contact resistance between copper substrates. *Nanotechnology*, 17(9):2294–2303, apr 2006.

[194] Matthew R Maschmann, Gregory J Ehlert, Benjamin T Dickinson, David M Phillips, Cody W Ray, Greg W Reich, and Jeffery W Baur. Bioinspired carbon nanotube fuzzy fiber hair sensor for air-flow detection. *Advanced Materials*, 26(20):3230–3234, 2014.

[195] Matthew R Maschmann, Ben Dickinson, Gregory J Ehlert, and Jeffery W Baur. Force sensitive carbon nanotube arrays for biologically inspired airflow sensing. *Smart Materials and Structures*, 21(9):094024, 2012.

[196] Gregory J Ehlert, Matthew R Maschmann, and Jeffery W Baur. Electromechanical behavior of aligned carbon nanotube arrays for bio-inspired fluid flow sensors. In *Active and Passive Smart Structures and Integrated Systems*, volume 7977, page 79771C, 2011.

[197] Natnael Behabtu, Colin C Young, Dmitri E Tsentalovich, Olga Kleinerman, Xuan Wang, Anson WK Ma, E Amram Bengio, Ron F ter Waarbeek, Jorrit J de Jong, Ron E Hoogerwerf, et al. Strong, light, multifunctional fibers of carbon nanotubes with ultrahigh conductivity. *Science*, 339:182–186, 2013.

[198] Baratunde A Cola, Jun Xu, Changrui Cheng, Xianfan Xu, Timothy S Fisher, and Hanping Hu. Photoacoustic characterization of carbon nanotube array thermal interfaces. *Journal of Applied Physics*, 101(5):054313, 2007.

[199] Baratunde A Cola, Jun Xu, and Timothy S Fisher. Contact mechanics and thermal conductance of carbon nanotube array interfaces. *International Journal of Heat and Mass Transfer*, 52(15-16):3490–3503, 2009.

[200] Baratunde A Cola, Xianfan Xu, and Timothy S Fisher. Increased real contact in thermal interfaces: A carbon nanotube/foil material. *Applied Physics Letters*, 90(9):093513, 2007.

[201] Qingzhong Zhao, Marco Buongiorno Nardelli, and Jerry Bernholc. Ultimate strength of carbon nanotubes: A theoretical study. *Physical Review B*, 65(14):144105, 2002.

[202] Rufan Zhang, Qian Wen, Weizhong Qian, Dang Sheng Su, Qiang Zhang, and Fei Wei. Superstrong ultralong carbon nanotubes for mechanical energy storage. *Advanced Materials*, 23(30):3387–3391, 2011.

[203] Mei Zhang, Ken R Atkinson, and Ray H Baughman. Multifunctional carbon nanotube yarns by downsizing an ancient technology. *Science*, 306(5700):1358–1361, 2004.

[204] Ya Feng, Taiki Inoue, Hua An, Rong Xiang, Shohei Chiashi, and Shigeo Maruyama. Quantitative study of bundle size effect on thermal conductiv-

ity of single-walled carbon nanotubes. *Applied Physics Letters*, 112(19):191904, 2018.

[205] T Dürkop, SA Getty, Enrique Cobas, and MS Fuhrer. Extraordinary mobility in semiconducting carbon nanotubes. *Nano letters*, 4(1):35–39, 2004.

[206] Donglai Zhong, Zhiyong Zhang, Li Ding, Jie Han, Mengmeng Xiao, Jia Si, Lin Xu, Chenguang Qiu, and Lian-Mao Peng. Gigahertz integrated circuits based on carbon nanotube films. *Nature Electronics*, 1(1):40–45, 2018.

[207] Mostafa Bedewy, Eric R Meshot, Michael J Reinker, and A John Hart. Population growth dynamics of carbon nanotubes. *ACS Nano*, 5(11):8974–8989, 2011.

[208] Guofang Zhong, Jamie H Warner, Martin Fouquet, Alex W Robertson, Bingan Chen, and John Robertson. Growth of ultrahigh density single-walled carbon nanotube forests by improved catalyst design. *ACS Nano*, 6(4):2893–2903, 2012.

[209] Matthew R Maschmann, Qiuhong Zhang, Feng Du, Liming Dai, and Jeffery Baur. Length dependent foam-like mechanical response of axially indented vertically oriented carbon nanotube arrays. *Carbon*, 49(2):386–397, 2011.

[210] Matthew R Maschmann, Gregory J Ehlert, Sei Jin Park, David Mollenhauer, Benji Maruyama, A John Hart, and Jeffery W Baur. Visualizing strain evolution and coordinated buckling within CNT arrays by in situ digital image correlation. *Advanced Functional Materials*, 22(22):4686–4695, 2012.

[211] Anyuan Cao, Pamela L Dickrell, W Gregory Sawyer, Mehrdad N Ghasemi-Nejhad, and Pulickel M Ajayan. Super-compressible foam-like carbon nanotube films. *Science*, 310(5752):1307–1310, 2005.

[212] Siddhartha Pathak, Ee J Lim, Parisa Pour Shahid Saeed Abadi, Samuel Graham, Baratunde A Cola, and Julia R Greer. Higher recovery and better energy dissipation at faster strain rates in carbon nanotube bundles: An in-situ study. *ACS Nano*, 6(3):2189–2197, 2012.

[213] Matthew R Maschmann, Qiuhong Zhang, Robert Wheeler, Feng Du, Liming Dai, and Jeffery Baur. In situ SEM observation of column-like and foam-like CNT array nano-indentation. *ACS Applied Materials & Interfaces*, 3(3):648–653, 2011.

[214] X Jack Hu, Antonio A Padilla, Jun Xu, Timothy S Fisher, and Kenneth E Goodson. 3-omega measurements of vertically oriented carbon nanotubes on silicon. *Journal of Heat Transfer*, 128(11):1109–1113, 2006.

[215] Rahul Rao, David Liptak, Tonya Cherukuri, Boris I Yakobson, and Benji Maruyama. In situ evidence for chirality-dependent growth rates of individual carbon nanotubes. *Nature Materials*, 11(3):213–216, 2012.

[216] Pavel Nikolaev, Daylond Hooper, Frederick Webber, Rahul Rao, Kevin Decker, Michael Krein, Jason Poleski, Rick Barto, and Benji Maruyama. Autonomy in materials research: A case study in carbon nanotube growth. *npj Computational Materials*, 2(1):1–6, 2016.

[217] Taher Hajilounezhad, Damola M Ajiboye, and Matthew R Maschmann. Evaluating the forces generated during carbon nanotube forest growth and self-assembly. *Materialia*, 7:100371, 2019.

[218] Matthew R Maschmann. Integrated simulation of active carbon nanotube forest growth and mechanical compression. *Carbon*, 86:26–37, 2015.

[219] Josef Brown, Taher Hajilounezhad, Nicholas T Dee, Sanha Kim, A John Hart, and Matthew R Maschmann. Delamination mechanics of carbon nanotube micropillars. *ACS Applied Materials & Interfaces*, 11(38):35221–35227, 2019.

[220] Ryan Hines, Taher Hajilounezhad, Cole Love-Baker, Gordon Koerner, and Matthew R Maschmann. Growth and mechanics of heterogeneous, 3D carbon nanotube forest microstructures formed by sequential selective-area synthesis. *ACS Applied Materials & Interfaces*, 12(15):17893–17900, 2020.

[221] Taher Hajilounezhad, Zakariya A Oraibi, Ramakrishna Surya, Filiz Bunyak, Matthew R Maschmann, Prasad Calyam, and Kannappan Palaniappan. Exploration of carbon nanotube forest synthesis-structure relationships using physics-based simulation and machine learning. In *IEEE Applied Imagery Pattern Recognition Workshop*, pages 1–8, 2019.

[222] Placidus B. Amama, Cary L. Pint, Seung Min Kim, Laura McJilton, Kurt G. Eyink, Eric A. Stach, Robert H. Hauge, and Benji Maruyama. Influence of alumina type on the evolution and activity of alumina-supported Fe catalysts in single-walled carbon nanotube carpet growth. *ACS Nano*, 4(2):895–904, 2010.

[223] Jennifer Carpena-Núñez, Jorge Anibal Boscoboinik, Sammy Saber, Rahul Rao, Jian-Qiang Zhong, Matthew R Maschmann, Piran R Kidambi, Nicholas T Dee, Dmitri N Zakharov, A John Hart, et al. Isolating the roles of hydrogen exposure and trace carbon contamination on the formation of active catalyst populations for carbon nanotube growth. *ACS Nano*, 13(8):8736–8748, 2019.

[224] Nicholas T Dee, Jinjing Li, Alvin Orbaek White, Christine Jacob, Wenbo Shi, Piran R Kidambi, Kehang Cui, Dmitri N Zakharov, Nina Z Janković, Mostafa Bedewy, et al. Carbon-assisted catalyst pretreatment enables straightforward synthesis of high-density carbon nanotube forests. *Carbon*, 153:196–205, 2019.

[225] Matthieu Lagardère, Ingrid Chamma, Emmanuel Bouilhol, Macha Nikolski, and Olivier Thoumine. FluoSim: simulator of single molecule dynamics for fluorescence live-cell and super-resolution imaging of membrane proteins. *Scientific Reports*, 10(1):1–14, 2020.

[226] Michal Kozubek. When deep learning meets cell image synthesis. *Cytometry. Part A: the journal of the International Society for Analytical Cytology*, 97(3):222–225, 2020.

[227] David Svoboda and Vladimir Ulman. MitoGen: a framework for generating 3D synthetic time-lapse sequences of cell populations in fluorescence microscopy. *IEEE Transactions on Medical Imaging*, 36(1):310–321, 2016.

[228] Akinyinka O Omigbodun, Frederic Noo, Michael McNitt-Gray, William Hsu, and Scott S Hsieh. The effects of physics-based data augmentation on the generalizability of deep neural networks: Demonstration on nodule false-positive reduction. *Medical Physics*, 46(10):4563–4574, 2019.

[229] Mostafa Bedewy and A John Hart. Mechanical coupling limits the density and quality of self-organized carbon nanotube growth. *Nanoscale*, 5(7):2928–2937, 2013.

[230] Matthew R. Maschmann, Gregory J. Ehlert, Sameh Tawfick, A. John Hart, and Jeffery W. Baur. Continuum analysis of carbon nanotube array buckling enabled by anisotropic elastic measurements and modeling. *Carbon*, 66:377–386, 2014.

[231] Shelby B Hutchens, Lee J Hall, and Julia R Greer. In situ mechanical testing reveals periodic buckle nucleation and propagation in carbon nanotube bundles. *Advanced Functional Materials*, 20(14):2338–2346, 2010.

[232] Marcus A. Worsley, Sergei O. Kucheyev, Joe H. Satcher, Alex V. Hamza, and Theodore F. Baumann. Mechanically robust and electrically conductive carbon nanotube foams. *Applied Physics Letters*, 94(7):073115, 2009.

[233] Anna Brieland-Shoultz, Sameh Tawfick, Sei Jin Park, Mostafa Bedewy, Matthew R Maschmann, Jeffery W Baur, and A John Hart. Scaling the stiffness, strength, and toughness of ceramic-coated nanotube foams into the structural regime. *Advanced Functional Materials*, 24(36):5728–5735, 2014.

[234] Mostafa Bedewy, Eric R Meshot, Haicheng Guo, Eric A Verploegen, Wei Lu, and A John Hart. Collective mechanism for the evolution and self-termination of vertically aligned carbon nanotube growth. *The Journal of Physical Chemistry C*, 113(48):20576–20582, 2009.

[235] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[236] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[237] K. Gao, H. AliAkbarpour, G. Seetharaman, and K. Palaniappan. DCT-based local descriptor for robust matching and feature tracking in wide area motion imagery. *IEEE Geoscience and Remote Sensing Letters*, pages 1–5, 2020.

[238] R. Bao, K. Palaniappan, Y. Zhao, G. Seetharaman, and W. Zeng. GLSNet: Global and local streams network for 3D point cloud classification. In *IEEE Applied Imagery Pattern Recognition Workshop*, pages 1–9, 2019.

[239] Y. M. Kassim, O. V. Glinskii, V. V. Glinsky, V. H. Huxley, G. Guidoboni, and K. Palaniappan. Deep U-Net regression and hand-crafted feature fusion

for accurate blood vessel segmentation. In *IEEE International Conference on Image Processing*, pages 1445–1449, Aug 2019.

[240] Yasmin M Kassim, Kannappan Palaniappan, Feng Yang, Mahdieh Poostchi, Nila Palaniappan, Richard J Maude, Sameer Antani, and Stefan Jaeger. Clustering-based dual deep learning architecture for detecting red blood cells in malaria diagnostic smears. *IEEE Journal of Biomedical and Health Informatics*, 25(5):1735–1746, 2020.

[241] Adel Hafiane, Kannappan Palaniappan, and Guna Seetharaman. Joint adaptive median binary patterns for texture classification. *Pattern Recognition*, 48(8):2609–2620, 2015.

[242] Adel Hafiane, Guna Seetharaman, Kannappan Palaniappan, and Bertrand Zavidovique. Rotationally invariant hashing of median binary patterns for texture classification. In *International Conference Image Analysis and Recognition*, pages 619–629. Springer, 2008.

[243] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[244] Rui Zheng, Lei Liu, Shulin Zhang, Chun Zheng, Filiz Bunyak, Ronald Xu, Bin Li, and Mingzhai Sun. Detection of exudates in fundus photographs with imbalanced learning using conditional generative adversarial network. *Biomedical Optics Express*, 9(10):4863–4878, 2018.

[245] Koundinya Nouduri, Ke Gao, Joshua Fraser, Shizeng Yao, Hadi AliAkbarpour, Filiz Bunyak, and Kannappan Palaniappan. Deep realistic novel view generation for city-scale aerial images. In *IEEE International Conference on Pattern Recognition*, pages 10561–10567, 2021.

[246] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[247] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[248] Stephan J Ihle, Andreas M Reichmuth, Sophie Girardin, Hana Han, Flurin Stauffer, Anne Bonnin, Marco Stampanoni, Karthik Pattisapu, János Vörös, and Csaba Forró. Unsupervised data to content transformation with histogram-matching cycle-consistent generative adversarial networks. *Nature Machine Intelligence*, 1(10):461–470, 2019.

[249] Patrick Trampert, Dmitri Rubinstein, Faysal Boughorbel, Christian Schlinkmann, Maria Luschkova, Philipp Slusallek, Tim Dahmen, and Stefan Sandfeld. Deep neural networks for analysis of microscopy images—synthetic data generation and adaptive sampling. *Crystals*, 11(3):258, 2021.

[250] K Jensen, W Mickelson, A Kis, and A Zettl. Buckling and kinking force measurements on individual multiwalled carbon nanotubes. *Physical Review B*, 76(19):195436, 2007.

[251] Kristofer G Reyes and Benji Maruyama. The machine learning revolution in materials? *MRS Bulletin*, 44(7):530–537, 2019.

[252] Stefano Curtarolo, Gus LW Hart, Marco Buongiorno Nardelli, Natalio Mingo, Stefano Sanvito, and Ohad Levy. The high-throughput highway to computational materials design. *Nature Materials*, 12(3):191–201, 2013.

[253] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold

learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2–3):81–227, 2012.

[254] Y. M. Kassim, V. B. S. Prasath, R. Pelapur, O. Glinskii, R. J. Maude, V. Glinsky, V. Huxley, and K. Palaniappan. Random forests for dura mater microvasculature segmentation using epifluorescence images. In *IEEE Engineering in Medicine and Biology Society Conference*, pages 2901–2904, 2016.

[255] V. B. S. Prasath, Y. M. Kassim, Z. A. Oraibi, J.-B. Guiriec, A. Hafiane, G. Seetharaman, and K. Palaniappan. HEp-2 cell classification and segmentation using motif texture patterns and spatial features with random forests. In *IEEE International Conference on Pattern Recognition*, pages 90–95, 2016.

[256] Zakariya A Oraibi, Hayder Yousif, Adel Hafiane, Guna Seetharaman, and Kannappan Palaniappan. Learning local and deep features for efficient cell image classification using random forests. In *IEEE International Conference on Image Processing*, pages 2446–2450, 2018.

[257] Hani Hagras. Toward human-understandable, explainable AI. *Computer*, 51(9):28–36, 2018.

[258] Frank Moosmann, Eric Nowak, and Frederic Jurie. Randomized clustering forests for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.

[259] Lorna J Gibson and Michael F Ashby. *Cellular Solids: Structure and Properties*. Cambridge University Press, 1999.

[260] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Vincent Weiss, Ron Dubourg, Alexandre Passos, and David Cournapeau. Scikit-learn:

Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[261] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[262] Pavel Matula, Martin Maška, Dmitry V Sorokin, Petr Matula, Carlos Ortiz-de Solórzano, and Michal Kozubek. Cell tracking accuracy measurement based on comparison of acyclic oriented graphs. *PloS One*, 10(12):e0144959, 2015.

VITA

Rina Bao was born in Hohhot, Inner Mongolia, China. She graduated with a B.E. Degree in Communication Engineering from the School of Electronic Information Engineering in Tianjin University in 2014. She studied towards her Ph.D. Degree in Computer Science at University of Missouri from 2014. She began working with Prof. Wenjun Zeng on pose estimation and object tracking research using deep learning. Afterward, she worked with Prof. Kannappan Palaniappan in CIVA Lab and closely collaborated with Prof. Yunxin Zhao. Her current research interests include image, video, and 3D point cloud analysis with machine learning methods.