2021 Third International Conference on Transdisciplinary AI (TransAI) | 978-1-6654-3412-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/TransAI51903.2021.00032

Classification and Feature Extraction for Hydraulic Structures Data Using Advanced CNN Architectures

Sameerah Talafha
School of Computing
Southern Illinois University, USA
Email: sameerah.talafha@siu.edu

Di Wu
Geography and Environmental Resources
Southern Illinois University, USA
Email: di.wu@siu.edu

Banafsheh Rekabdar
School of Computing
Southern Illinois University, USA
Email: brekabdar@cs.siu.edu

Ruopu Li
Geography and Environmental Resources
Southern Illinois University, USA
Email: ruopu.li@siu.edu

Guangxing Wang
Geography and Environmental Resources
Southern Illinois University, USA
Email: gxwang@siu.edu

***Abstract*—An efficient feature selection method can significantly boost results in classification problems. Despite ongoing improvement, hand-designed methods often fail to extract features capturing high- and mid-level representations at effective levels. In machine learning (Deep Learning), recent developments have improved upon these hand-designed methods by utilizing automatic extraction of features. Specifically, Convolutional Neural Networks (CNNs) are a highly successful technique for image classification which can automatically extract features, with ongoing learning and classification of these features. The purpose of this study is to detect hydraulic structures (i.e., bridges and culverts) that are important to overland flow modeling and environmental applications. The dataset used in this work is a relatively small dataset derived from 1-m LiDAR-derived Digital Elevation Models (DEMs) and National Agriculture Imagery Program (NAIP) aerial imagery. The classes for our experiment consist of two groups: the ones with a bridge/culvert being present are considered "True", and those without a bridge/culvert are considered "False". In this paper, we use advanced CNN techniques, including Siamese Neural Networks (SNNs), Capsule Networks (CapsNets), and Graph Convolutional Networks (GCNs), to classify samples with similar topographic and spectral characteristics, an objective which is challenging utilizing traditional machine learning techniques, such as Support Vector Machine (SVM), Gaussian Classifier (GC), and Gaussian Mixture Model (GMM). The advanced CNN-based approaches combined with data pre-processing techniques (e.g., data augmenting) produced superior results. These approaches provide efficient, cost-effective, and innovative solutions to the identification of hydraulic structures.**

I. Introduction

Digital Elevation Models (DEMs) utilizing grid data have been developed to facilitate terrain modeling and analysis using computers. Such gridded DEMs automatically derive thematic maps for calculating terrain attributes, including concavity, convexity, aspect, and slope. This efficient and expansive functioning of DEMs has engendered interest in exciting applications. For example, DEMs have proved useful in maintaining geographical databases, assessing damage from natural disasters like floods and earthquakes, and supporting crucial military training and other [15]. The uses of DEMs, however, present unique challenges in some applications. For example, DEMs, especially those high-resolution DEMs, are only available as Digital Surface Models (DSMs), which often represent the land surface elevation without the consideration of potential underground structures. This is especially the case in hydrological modeling in which drainage flowlines may falsely terminate at the hydraulic structures (i.e., road culverts and bridges) near interactions of roads and streams [15]. Presently, the dataset of hydraulic structures is either largely missing or available in variable quality, which affects proper use of high-resolution DEMs for many hydrological applications [10]. In many cases, laborious on-screen digitization using aerial photos has been used to identify hydraulic structures or improve their locational accuracy. Thus, it is imperative to develop an efficient approach for identifying hydraulic structures.

Traditional machine learning classification tasks can be time consuming and burdensome [19]. First, a feature must be selected from the imagery(using hand-picked or analytical methods), and then a classifier must be used to detect different group types in the image. Analyzing DEM samples with such a method to extract good features/attributes can, then, be a formidable task. The use of CNNs [19] obviates the necessity of prior feature selection by assimilating useful features through extraction and classification automatically, into a single framework. Feature selection is automatically assigned to the network which, for a given classification task, optimally selects the most relevant features. CNN is also flexible and works quite well on images' datasets. However, CNNs rely on data to perform well [23]. Before investing the time and money needed to collect data and move it into a system, stakeholders (particularly in a business context) want to see and understand how such a deep learning system will work. This necessitates alternative methods. This paper investigated the potential of different advanced deep CNN models, such as SNNs [23], CapsNets [5], [1], and GCN [11], towards classifying images with or without a culvert/bridge as "True" or "False" in a given DEM sample. These CNN models build more accurate prediction of culvert existence,

978-1-6654-3412-6/21/$31.00 ©2021 IEEE
DOI 10.1109/TransAI51903.2021.00032

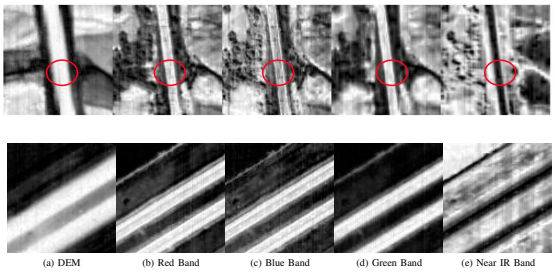


Fig. 1: Five bands of the two samples in our dataset, where (up) is the True sample group where the culvert exists, and (down) is the False sample group where the culvert does not exist. The red circles in the truth sample in (up) point to the position of the culvert/bridge in each band of the image. [20]

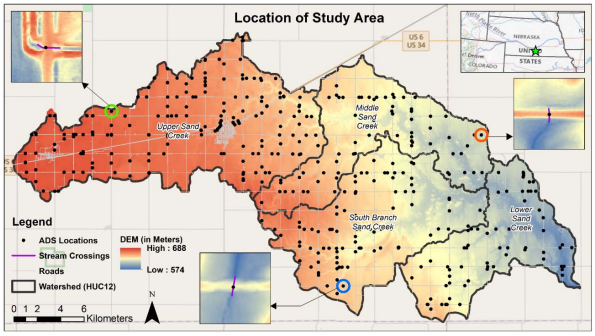


Fig. 2: The site of the study area is color-coded to indicate elevation information, the location of anthropogenic drainage structures (ADS), stream crossings, and roads. The green, orange and blue circles show the three zoomed-in locations of the ADS at stream crossings.

TABLE I: List of datasets with details and sources

Dataset	Description	Source
Aerial orthophoto	1m aerial imagery from the National Agriculture Imagery Program (NAIP)	USDA's Farm Service Agency [20]
Medium resolution DEM	10-m resolution DEM	U.S. Geological Survey [22]
High-resolution digital elevation model (HRDEM)	LiDAR-derived HRDEM with 1m spatial resolution	3D Elevation Program [21]
Road network	Polylines representing road centerlines	U.S. Census Bureau TIGER/Line [2]

a first step crucial in developing auto-detection algorithm on large images. The key contributions in this work are two folds:

1) **Data Augmentation**: We explore Generative Adverserial Networks based data augmentation technique, yielding better results for classification of DEM samples when compared with classical data augmentation techniques.
2) **Classification using Advanced Deep Neural Network Architectures**: We explore SNNs (CNN based), CapsNet and GCN archictures for classifiying DEM samples. To the best of our knowledge, no prior work exists to classify DEM samples with aforementioned architectures.

The rest of the paper is organized as follows: section II reviews related works and introduce the necessary background about machine learning/deep learning models applied on hydraulic structures datasets, section III describes the study area and dataset , section IV represents our proposed model, section II details experimental design, the evaluation, and results, and section VI concludes the paper with remarks on the achieved results.

II. Related Works

Recently, many hydraulic-related applications have adopted deep learning models. Jakovljevic et al. [7] developed the Neural Network (NN) and deep learning-based method to classify point cloud and ground point filtering. The model succeeded to improve the process of ground classification of LiDAR and Unmanned Aerial Vehicle (UAV) data, leading to produce DEM with the desired accuracy for flood risk mapping. This study's LiDAR and UAV data are huge and allowed the model to extract 6 million points from intensity data and local and global geometric features. Jiang et al. [8] used a multi-scale CNN model to deal with the complicated features of urban topography and to rebuild high-resolution urban DEMs. In this work, the data, approximately 17 million DEM samples, were collected from London, one of the biggest cities in the world. Kabir et al. [9] proposed an effective hydraulic model based on CNN architecture to predict rapidly flood volume and inundation maps. It showed that a fast and robust model for the real-time problem is crucial for estimating the multidimensional social and economic effects and giving credible predictions to improve societal resilience to flooding. Further, the results showed that the model is quite accurate in extracting flooded cells as pointed out by different quantitative assessment matrices. Most of the deep learning hydraulic CNN models proposed need a large dataset to achieve decent results. In our work, we expose limitations of the standard deep CNN model, such as the overfitting problem [19] when applied to the classification task by using our small dataset. Further, since CNNs poorly encode the different representations of pose and orientation, traditional CNNs could only make predictions on an image if the original image on which they were trained and the test image were almost perfectly aligned. This problem in CNN is what Professor Hinton called the Invariance vs. Equivariance problem [5]. We investigate using advanced deep CNN-based models, such as SNNs CapsNets, and GCNs as an alternative to standard CNNs.

III. Study Areas and Dataset

In this paper, the datasets used include the LiDAR-derived topographic DEMs, Farm Service Agency (FSA) 4-band digital orthophotos, and other supplemental geospatial datasets (e.g., administrative boundaries, National Hydrogra-

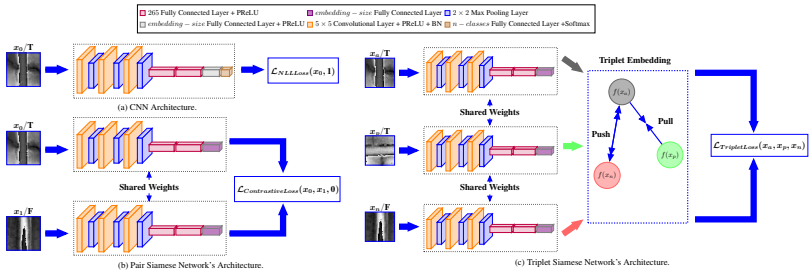


Fig. 3: (a) An example showing how Softmax NLLLoss works to identify the label of the DEM sample. (b) An example showing how ContrastiveLoss works to distinguish the dissimilarity between the pair of the embedded images belonging to the different classes. (c) An example showing how TripletLoss works to pull the embedded images of the same label closer together, and push the embedded images of the different labels further apart.

phy Dataset) located in northeastern Nebraska, USA, with a dimension of 100×100 pixels (see Fig. 1). The datasets along with its description are displayed in Table I. The watershed contains four HUC-12 subwatersheds, namely South Sand Creek Branch, Lower Sand Creek, Middle Sand Creek, and Upper Sand Creek, with a total surface area of approximately $552\ km^2$. The dataset contains 1968 images containing at least one culvert (True) and 1968 that contain no culverts (False). To develop and test the efficacy of methods in broader geographic contexts, we selected the Sand Creek watershed in the northeast Nebraska of the United States to conduct our experiments Fig. 2. The landscape is characterized by intensive agriculture, relatively level topography, and a dense network of roads. The embankments of these roads complicate flow patterns and significantly fragment the agro-hydrologic landscape, making these areas representative of regions in which human structures have led to broadly segmented agro-hydrologic systems. In these areas, then, development of hydrologic drainage networks that are accurate and fully connected is critical. In all experiments, to evaluate the models (the traditional machine learning and advanced CNNs' models), we used K-Fold Cross-Validation , where we divided the data into 10 folds, each being (10%) of the full dataset. To update iterative network weights based on training data, in all the advanced CNNs' models Adam optimizer was utilized through an inceptive learning rate of 10^{-3}.

IV. Proposed Method

Most traditional machine learning models are unsuccessful to extract useful features from images with complex data structures. In binary classification, classifiers like SVM [4], GC [17], and GMM [14] are task-specific algorithms utilizing the location of hyperplanes or decision boundaries to determine if a data point is to be positively or negatively labeled. On the other hand, CNN model is based on learning patterns and representations found in the given data (understanding the data patterns). It extracts the relevant features from the input using the convolution operation, and then selects the essential features that improve the performance for various image classification problems. Recently, some rather exciting advancements in CNNs have been reported, such as using beneficial types of optimizers, activation functions, loss functions, and regularizers [19]. Furthermore, a notable improvement has been achieved in representing CNNs' architectures by exploiting spatial and channel information, controlling in depth and width of architecture, and processing multi-path information, which in turn led to the emergence of sophisticated CNNs. This section will discuss the effectiveness of the three significant CNN architectures to solve our classification problem.

A. Siamese Learning Pair/Triplet Mining

In the modern Deep learning era, traditional CNNs are sufficient for most classification problems. Unfortunately, in real-world datasets, CNNs need a large dataset to perform well. Recently, SNNs have become increasingly popular in deep learning research and applications because of their ability to learn from relatively small data. For this reason, we explore using SNNs to deal with the lack of our training set. The main advantages of using SNNs are [23]: (1.) they make training with a small dataset possible by using a one-shot learning strategy. (2.) they do classification based on semantic similarity by creating embeddings for inputs derived from the deeper layers in CNN, where the similar embeddings belong to the same class. (3.) they perform better than ensemble classifiers (e.g. Gradient Boosting Machine (GBM) and Random Forest (RF) classifier), where using two or more parallel CNN networks can do much more efficiently than simple average over many traditional classifiers. SNN can be used as a classification model. SNNs generate images' embeddings that have a meaningful Euclidean relationship using CNNs, where images of the same class will be eventually embedded close together, or at least closer than images from another class. SNN is comprised of a similar chain of the layers of CNN, but in this case the output layer lacks a Softmax function (see Fig. 3). In addition, since the training of SNNs involve pairwise or triplet learning, cross-entropy loss [27] interpreted as a negative log-likelihood (**NLLLoss**) cannot be used in this case.

In our experiment, we use two types of learning to train SNNs: (1.) learning with offline pair/triplet mining [3], and

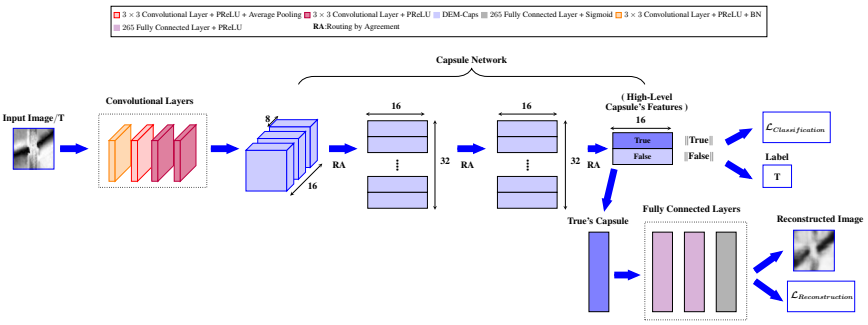


Fig. 4: The Proposed DEM-CAPS Architecture. Convolutional layers are used to form the capsules, and the decision is made based on the agreement among these capsules. The 'True' capsule in the DEM-CAPs is represented by the 'deep blue' indication. Based on the integration of three layers, this can efficiently predict the input.

(2.) learning with online pair/triplet mining [28]. An offline pair mining model takes a pair of images and produces their embeddings using the two parallel identical CNNs. The Contrastive loss function (**ContrastiveLoss**) is used to learn the offline pair mining model as follows:

$$\mathcal{L}(x_0, x_1, y) = \frac{1}{2} y \|f(x_0) - f(x_1)\|_2^2 + \frac{1}{2}(1-y) \\ max(0, m - \|f(x_0) - f(x_1)\|_2{}^2 , \quad (1)$$

where $f(x_0)$ and $f(x_1)$ are the representation of an embedding's pair for the images x_0 and x_1, y is a binary flag equal to 0 for a negative sample pair (from the same class) and to 1 for a positive sample pair (from different classes), m is a margin that often equals 0.2, and the distance $\|.\|_2$ is the Euclidean distance. While the traditional loss aims to predict class labels, the contrastive loss aims to intuit within intervals of inputs Euclidean distances when those inputs are projected onto a hyperspace. An offline triplet mining model takes three inputs which we call a triplet. These include an anchor x_a, a positive x_p demonstrating an equivalent class as anchor, and a negative x_n demonstrating a different class than the anchor. At training time, each input of this triplet will be fed to its own CNN branch to be embedded. These embeddings $f(.)$ are passed to what is referred to as triplet loss [3]. This triplet loss aims to learn the embeddings in order to produce a mariginal value by which the anchor is rendered more similar to the positive example than to the negative. Triplet loss (**TripletLoss**) is given as follow:

$$\mathcal{L}(x_a, x_p, x_n) = max(0, m + \|f(x_a) - f(x_p)\|_2^2 \\ - \|f(x_a) - f(x_n)\|_2^2) . \quad (2)$$

The big plus of the offline pair/triplet strategy is that it is easy to understand, it is supervised, and it is relatively straightforward to implement. However, the SCNN's performance is highly dependent on the creation of these pairs/triplets; if pairs/triplets are too easy, there is nothing to learn, and if they are too difficult the embedding might collapse into a single point. Therefore, The target here is to create as many semi-hard pairs/triplets as possible. Moreover, each image that is provided to the network is applied only once for the computation of contrastive/triplet loss for pairs/triplets. This computation, then, is somewhat wasted; once the embedding is generated, it should be reused for many pairs/triplets. In the offline mining technique, the whole dataset is converted into pairs/triplets before training. On the contrary, in online mining, during a training phase impromptu pairs/triplets are generated from data which is processed in the mini-batch of fed data . In this case, if we feed our classification network with K images per P classes, we can process up to $(KP-1)*(KP)/2$ pairs and $P*K*(K-1)/2*((P-1)*K)$ triplets, compared to $KP/2$ pairs and $KP/3$ triplets in offline implementation. This technique gives more pairs/triplets for a single batch of inputs. It is, therefore, much more efficient. Additionally, the selected pairs/triplets can be considered moderate sample (semi-hard), since they are the hardest within a small subset (mini-batch) of the data, which is exactly what is best for learning with the contrastive/triple loss model. As a general rule, online mining ought to be performed wherever possible, since it allows for much faster training.

B. Capsule Networks (CapsNets)

Despite CNNs' architectures success in many classification problems, they have various conceptual drawbacks: (1.) the average-pooling and max-pooling layer in CNNs waste away the position information about some entity that the network seeks to recognize, (2.) they require much data to learn efficiently, and (3.) they do not take into consideration spatial hierarchical relation between simple and complex objects in the image. CapsNet, with CNN capabilities and without its shortcomings, was conceptualized by Geoffrey Hinton [5]. It processes visual information in much the same way as the brain, which means it can maintain hierarchical spatial relationships; theoretically, this architecture may learn faster and use fewer samples per class. Recently, Sabour et al. [16] introduced the applicable implementation of CapsNet by extended it with an iterative routing-by-agreement algorithm to classify and segment multiple digits within an image. What makes CapsNet different from a CNN is this routing-by-agreement, which further gives it the ability to identify the

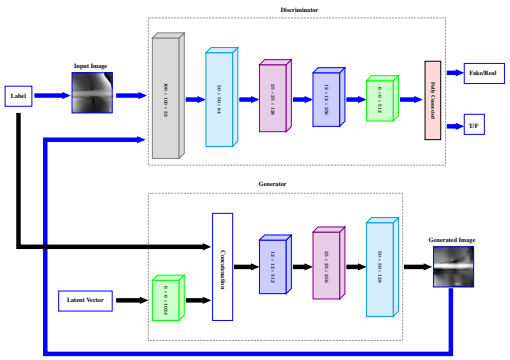


Fig. 5: DEM-AC-GAN complete Architecture with generator and discriminator.

spatial relations. In a CapsNet, convolution layers extract low-level features from the image and forward them to capsule layers containing vectors of equal dimension, where the first capsule layer shape must fit the output from the previous convolution layer. The orientation of the input vector s_j is kept, while length is changed within 0 and 1 using the non-linear activation function "squashing", with length of this vector representing the probability of the presence of an entity. As the equation, (3) shows that the capsule j represents the squashing function for the input vector s_j (before squashing), and the output vector v_j. For a specific entity, the attributes in the vector v_j generate varying aspects, for example the length, texture, position, and scale of that entity.

$$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|'}, \tag{3}$$

Similar to a traditional CNN, the input s_j is a weighted sum through all multiplications between the coupling coefficients c_{ij} computed by the iterative "Routing by Agreement" process and prediction vectors $\hat{u}_{j|i}$, where each prediction vector is created through multiplication of the output u_i from a specific capsule in the previous layer by a trainable weight matrix W_{ij}, as follows [13]:

$$\hat{u}_{j|i} = W_{ij}u_i, \tag{4}$$

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_i k)}, \tag{5}$$

$$s_j = \sum_i c_{ij}\hat{u}_{j|i}, \tag{6}$$

From capsule i along with capsules in the previous layer j, coupling coefficients are computed by the Softmax of b_{ij} pointing out the probability that capsule i should be coupled to capsule j,as follows:

$$b_{ij} = b_{ij} + \hat{u}_{j|i}.v_j, \tag{7}$$

1) The Proposed DME-CAPS : Fig. 4 displays the architecture of the DEM-CAPS here proposed. DEM-CAPS consists of three stages; (1.) feature extraction (2.) classification and (3.) reconstruction. DEM-CAPS was composed 4 convolutional layers, 3 capsule layers, along with three layers fully connected. DEM-CAPS is fed a DEM sample, as input. 4 convolution layers extract (low-level) image features. After that, these features will pass to the capsule layers to extract the high-level features performing the routing process, while the last capsule layer represents the instantiation parameters for the "True" class and the "False" class.

Finally, we calculated the length (L2 Norm) of each class (True and False) capsule to predict the class. As mentioned above, since the length of the capsule implements the probability of the entity which exists, if the label of the image is True, DME-CAPS's goal is to make True's capsule vector longer and make False's capsule vector shorter. Consequently, the combined losses of the two-class capsules dictates classification loss, defined as follows [13]:

$$\begin{aligned}\mathcal{L}_{classification} = \sum_k (T_k max(0, m^+ - \|v_k\|)^2 + \\ \lambda(1-T_k)max(0, \|v_k\| - m^-)^2)\end{aligned} \tag{8}$$

where T_k is 1 whenever the class k is existent and 0 otherwise. Terms m^+, m^-, and λ are the hyper parameters of the model.

We use the decoding network in the reconstruction stage. It has 3 layers which are fully connected, the first two-layer with PReLU activation function and the last with Sigmoid. Most essentially, to make sure there is no equivariance of mapping between an original image and its high-level capsule features, DME-CAPS, through value of Mean Squared Error (MSE) between a reconstructed image and the original, corresponds features of the class to which the last capsule layer belongs, using the decoding network to reconstruct the image from this last capsule layer by corresponding the features of the class to which it belongs. The reconstruction loss is given as follows [13]:

$$\mathcal{L}_{reconstruction} = \mu MSE(I, FC_{decoding}(v_k)) \tag{9}$$

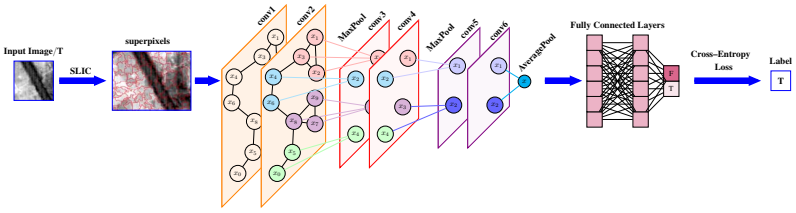


Fig. 6: The Proposed DEM-GCN Architacture.

The less the value of the reconstruction loss, the more the reconstructed image $FC_{decoding}(v_k)$ that is the output of fully connected layers is similar to I, and the better decoding network is able to conserve information required to rebuild an image within DEM-CAPS.

2) Data Augmentation: To avoid overfitting, classification problems often need large quantities of data, so there is the additional issue of inhibition of performance due to a shortage of training data. To offset this, substantial work has gone into maximum extraction of information by utilizing as much data collected through different data augmentation techniques as possible. Unless training data contains sufficient examples at different rotations, classical data augmentation techniques such as rotation, scaling, cropping, reflection, and translation, of training images, are not easily learnable by CNNs. In addition, these techniques produce only limited quantities of surrogate data. Recognizing the possibility of producing a wider set of accessions, we use a generative model to produce data augmentation, which organically moves beyond the bounds of classical techniques. A robust and efficient collection of networks, GANs [18], produce generate novel, credible images from originals both labeled and unlabeled, and thereby provide a germane solution to deep learning problems. To expand training datasets, various methods of GAN application have recently been proposed to generate new training images for classification problems. In part designed according to image Auxiliary Classifier Generative Adversarial Networks (AC-GAN) [25], this model develops the ability to generalize data items and produce additional data within that class by first deriving data from a particular source domain. We built out our AC-GAN, which is called DEM-AC-GAN (see Figure 5). The discriminator here is a CNN architecture fed DEM samples (the real image from the dataset, and the fake one created by the generator). The discriminator has two output layers; the first one predicts the label of class and the second output dictates whether an image is real or fake. The generator is fed by a latent vector and class label as input. It has deconvolution layers that reverse the operations of the convolutional layers of the discriminator to create an output image (fake image) passed to the discriminator.

C. Graph Convolutional Networks (GCNs)

Traditional CNN architectures can only work convolution ordered square image regions with fixed size and weights, so they cannot globally adjust to the distinguished local regions with different geometric appearances and object distributions. Lately, various advanced types of Graph Neural Networks (GNNs) have been created, with Graph Convolutional Networks (GCNs) being an important example [12]. Similar to CNN, GCNs extract features through exploration of local graph neighboring nodes. Sometimes considered a more generalized kind of CNN, GCNs essential different because while CNNs are designed to work on (Euclidean) structured data, GCNs are designed to work on irregular (non-Euclidean) structured data, in which connections in graph nodes differ and the nodes are un-ordered. To solve our classification problem, we implemented DEM's classification on embedded graphs by first utilizing Simple Linear Iterative Clustering (SLIC) [26] or the Quickshift algorithm [24]to segment the image into superpixels, converting this representation into a graph and inputting this to our GCN (DEM-GCN). The convolutions conducted on these graphs simultaneously aggregate features while gradually refining the input graphs. The superpixels that have the same label will be perfectly clustered together in the embedding space. Eventually, the classification output will be created by the well-trained network.

1) The Proposed DEM-GCN: The proposed DEM-GCN consists of graph convolutional layers and Eigen Pooling [12] used to extract features of the input graph and then create a vector representation for the classification process. Fig. 6 shows a binary graph classification for which the proposed model DEM-GCN was designed. The illustration shows the graph finally becoming a single supernode, after being 3 times coarsened. Node features make up the input. To the graph signal, there is then applied two convolutional layers through which the input is passed. As defined in the coarsened graph, the resulting node features are pooled to fewer nodes. Twice more this procedure (two convolution layers followed by the pooling layer) is repeated, until the pooling of the graph results in a signal node. Based on the value of this node, the prediction will be made through use of the cross-entropy loss in the last layer. Several convolutional layers stacked in the DEM-GCN can be written as follows :

$$F^{i+1} = \text{PReLU}(\tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}F^{i}W^{i}). \tag{10}$$

The output of the i–th convolutional layer is $F^{i+1} \in \mathbb{R}^{N\times d_{i+1}}$, where $i > 0$. $F^0 = I$ indicates to the input node features created by the superboxel model (SLIC or Quickshift). All convolutional layers C are gathered to the final node impersonations which the DEM-GCN model has learned, and

the output is implemented as the matrix F^C. We used the spectral pooling (averaging or maximizing) operator based on graph Fourier transform through utilization of the Eigenvectors of the Laplacian matrix of each subgraph [12] to aggregate into higher-level grapsh impersonations the node information, eventually aggregating it further into the single supernode. The pooling operator with l–th pooling operator Θ_l depended on graph Fourier transform is given as follow:

$$X_l = \Theta_l^{\mathrm{T}} X \tag{11}$$

where Θ_l is the pooling operator consisting of all the l–th eigenvectors from all the K subgraphs $\left[\bar{u_1}^{(1)}, ..., \bar{u_l}^{(K)}\right]$. Each $\bar{u_l}^{(k)}$ is up–sampled version of the eigenvector $u_l{}^{(k)}$ in the Laplacian matrix $L^{(k)}$ of the subgraph $\mathcal{G}^{(k)}$ is given as follow [12]:

$$\bar{u_l}^{(k)} = C^{(k)} u_l^{(k)}, \quad l = 1 \ , ..., \ N_k. \tag{12}$$

where $C^{(k)}$ is the up–sampling operator used to up–sample the eigenvectors into the entire graph. For more information regarding to Eigenvector-based pooling strategy, we refer readers to [12].

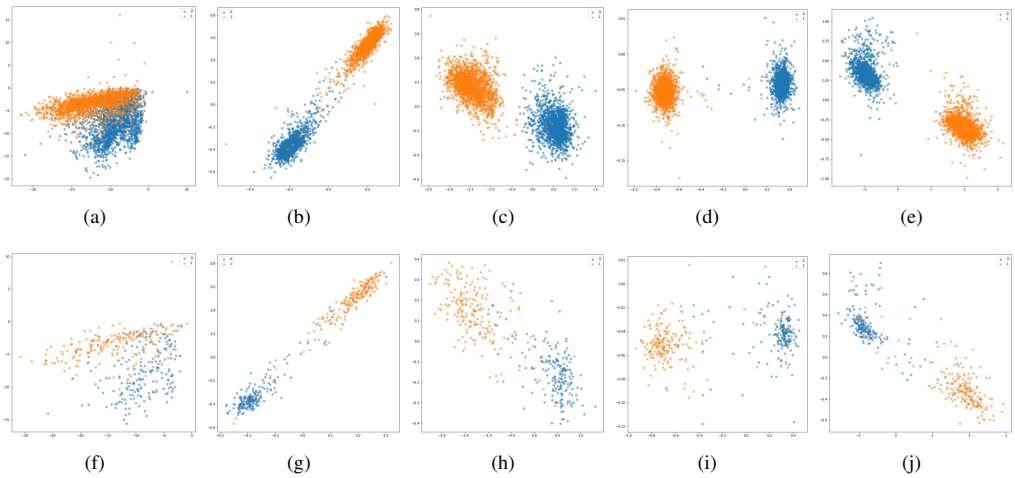
V. Results and Discussion

Traditional learning techniques such as SVM, GC and GMM are have several advantages but are not as performant (see Table II) as the deep learning models such as SNNs, CapsNets, and GCNs we proposed to solve our classification problem. These models have initiated a surge in deep learning due to their ability to function with small datasets. SNNs successfully solve over-fitting problems which hinder the performance of deep learning approaches to the train small dataset. To extract the most important features, SNNs' algorithms utilize parallel functioning of two or more CNNs working simultaneously, which vastly improved the classification accuracy. SNNs can be trained using either online or offline pair/triplet mining. Before beginning online mining, we studied how extreme distances effect performance, as well offline effects of neighbored patches training.We then analyzed, for online versus offline mining, extreme cases of embedding distance, examining the impact of batch semi-hard pair/triplet mining, easy positive, and neighborhood component analysis loss. Fig. 7 show results of two-dimensional embeddings taken from the penultimate layer of networks trained online and offline for our embeddings' network (Embedding-net). Figure 3 depicts the SNN architectures used together with their embeddings. In our experiments, we found that using the Embedding-net network that contains a batch normalization layer following each convolution layer improves the performance of all SNNs. Though enhancing the model's depth increases performance, it is challenging to train the neural network model with many layers due to the sensitivity of spontaneous conceptual weights and learning algorithm configuration. For each respective mini-batch, using batch normalization layers helps the deep neural network model standardize inputs to a specific layer, which not only changes the learning process but also significantly lessens the required intervals needed to adequately train the model. The classification results obtained by the SNNs' models were evaluated using the loss function for each type of SNNs and accuracy. To compute the accuracy in the SNNs' models, we extended the resulting embedding with a Sigmoid output layer for categorical predictions. The loss and accuracy plots of the four SNNs' models are illustrated in Fig. 8. These results suggest the online mining models perform the best compared to other SNN models.

The other advanced CNN model used is CapsNet. By utilizing inherent spatial relationships, such capsule networks more accurately summarize what we perceive by simulating the task of understanding changes in an image. To function with the small number of DEM samples available to us, we proposed the use of DEM-CAPS. While CNNs must be trained and deployed on large datasets, DEM-CAPS generalizes well with small datasets, making it conducive to our classification problem. Though it is possible through the use of the data augmentation technique to compensate for possible loss of performance resulting from lack of data, if our model is limited to train and test on the scant available domain data, there will be a noticeable difference in terms of performance.

It is difficult using traditional data augmentation methods alone (such as scaling, cropping, flipping, padding, etc.) to obtain a suitable number of appropriate images. We used a data augmentation method within GAN (DEM-AC-GAN). DEM-AC-GAN generated 1000 images of True labels and 1200 images of False labels. Figure 9 shows the example of generated samples by DEM-AC-GAN based on the ground truth images for both classes. The results show the ability of DEM-GAN to generate high-quality images, which we can now consider as the new samples in our dataset. The training and testing loss which is the classification loss and the reconstruction loss for the training and testing dataset, classification accuracy, and confusion matrix graphs for our classification problem obtained without/ with data augmentation are depicted in Fig. 11. The DEM-CAPS without augmentation techniques accurately identified 370 images from 396 testing images, while identifying 378 after applying the data augmentation technique. Fig. 10 shows an example of reconstructed images from the test dataset in both cases without/ with augmentation. Although the DEM-CAPS model performs creditably well on our small dataset, when we increased the number of the training images using the data augmentation technique, the DEM-CAPS became better able to reveal compelling features that can distinguish the two different labels of the DEM samples.

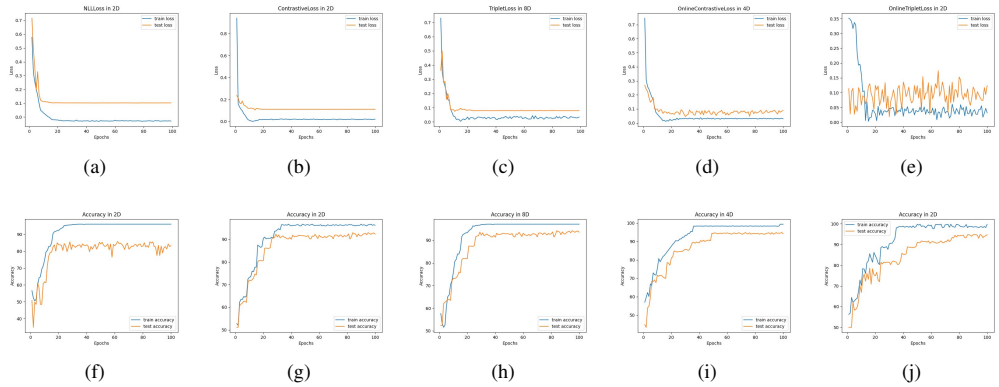
Finally, we used GCN with superpixel segmentation. For graphs, convolution was defined utilizing the graph Fourier transform, which in turn was defined as projection on the eigenvalues of the Laplacian. Fig. 12 shows the classification losses, classification accuracies, and confusion matrixes for the DEM-GCN model with the two types of superpixel segmentation models (SLIC and Quickshift). Based on the results, the performance of DEM-GCN achieved is poor relative to the other two advanced convolutional models, which indicates



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

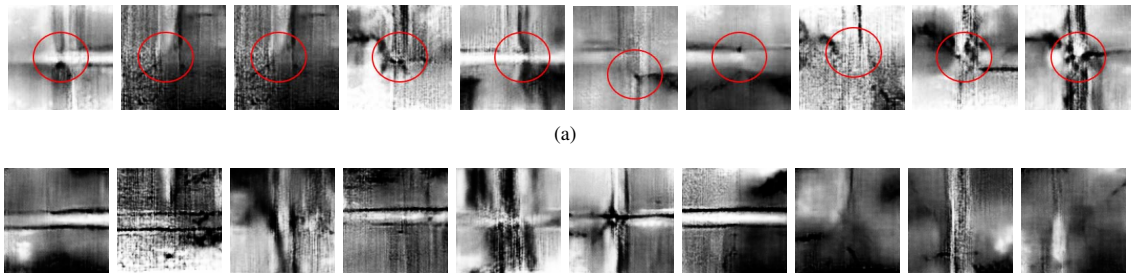
Fig. 7: Analysis of the performance of SNNs with Embedding-net in offline and online pair/triplet mining for the DEM dataset, where (up) is the training dataset, and (down) is the test datset. The graph results are structured as follows; (a)/(f): slandered CNN with Softmax layer, (b)/(g): SNN with offline pair mining, (c)/(h): SNN with offline triplet mining, (d)/(i): SNN with online pair mining, and (e)/(f): SNN with online triplet mining.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

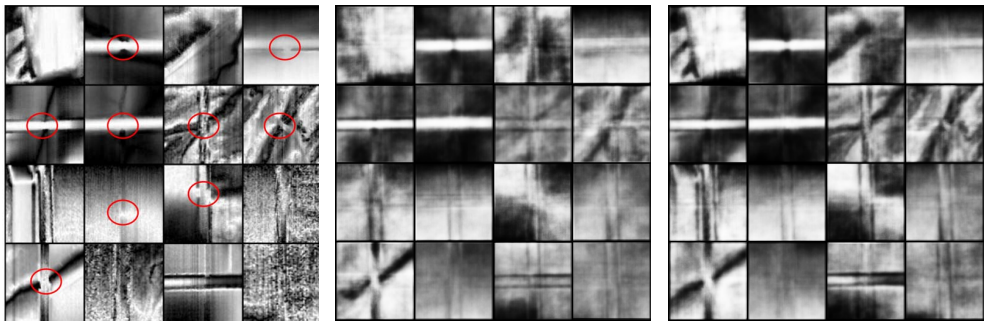
Fig. 8: The performance of the loss and accuracy of the SNNs for the best dimension of Embedding-net in each model , where (up) is the SNNs' losses, and (down) is the SNNs' accuracies. The graph results are structured as follows; (a)/(f): slandered CNN with Softmax layer, (b)/(g): SNN with offline pair mining, (c)/(h): SNN with offline triplet mining, (d)/(i): SNN with online pair mining, and (e)/(f): SNN with online triplet mining.



(a)

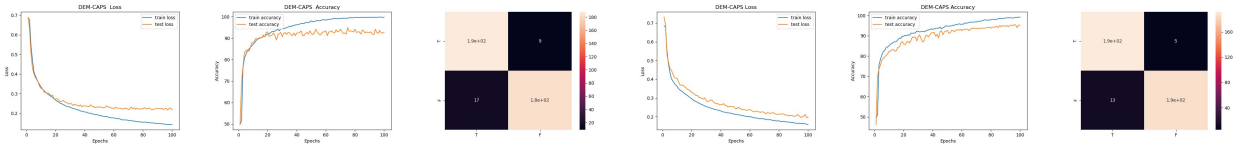
(b)

Fig. 9: (up): Generated images by DEM-AC-GAN (True label), (down): Generated images by DEM-AC-GAN (False label) .



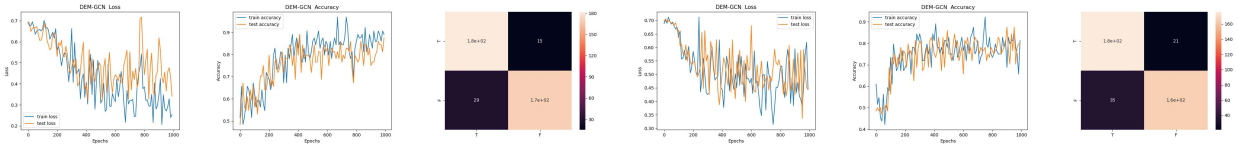
(a) (b) (c)

Fig. 10: Reconstructed DEM samples. (a): original DEM samples, (b): Reconstructed DEM samples without augmentation, and (c) Reconstructed DEM samples with augmentation.



(a) (b) (c) (d) (e) (f)

Fig. 11: (a)/(d): Training and Testing loss graphs for a fold (without data augmentation/with data augmentation), (b)/(e): Training and Testing Accuracy graphs for a fold (without data augmentation/with data augmentation), and (c)/(f): Confusion matrix of our binary classification results for testing dataset (without data augmentation/with data augmentation).



(a) (b) (c) (d) (e) (f)

Fig. 12: (a)/(d): Training and testing loss graphs for a fold in case using SLIC/ Quickshift segmentation, (b)/(e): Training and testing accuracy graphs for a fold in case using SLIC/ Quickshift segmentation, and (c): Confusion matrix of our binary classification results for testing dataset in case using SLIC/ Quickshift segmentation.

the GCN is not suitable to extract optimal features to classify the DEM samples. We invariably obtained unstable GCNs' training, despite using different methods to tune the parameters. Graph-structured data introduces complex noise, despite–compared with other formats of data, increased conservation of relational data. GCN also uses max-pooling, or simple average pooling, with loss of each node's characteristics as well as the topology between nodes. This further confirms the effectiveness of CapsNets, which do not include pooling layers in their architecture. Another way of interpreting the low performance of GCN in this context is that GCNs lose a large amount of information in the early stage during the graph convolution step, and only the final aggregation output is used for predicting the label. Various comparisons were also made between DEM-CAPS and the other traditional classifiers: all results showed that the DEM-CAPS is the best classification model for our dataset. Furthermore, the training and testing stability of DEM-CAPS is better compared to the other models. Table.II shows the summarized of all the models applied to solve our classification problem.

VI. Conclusion and Future Direction

Our binary classification problem was addressed by finding the optimal method to detect the label type of the DEM samples, where those with bridge/culvert points are considered "True", and those without bridge/culvert points are considered "False". One of the key issues for classification problems is the overfitting that frequently occurs with small datasets. Both deep learning and machine learning techniques have been implemented in this paper to deal with this problem and improve classification accuracy. We found that the proposed DME-CAPS with data augmentation achieved better accuracy

TABLE II: The accuracy results of traditional machine learning models vs advanced CNNs' architectures. In the traditional machine learning models, we used the Radial Basis Function (RBF) kernel SVM with(C and γ) parameters, GC with α that represents the smoothing parameter, and GMM with (α and K), where K is the number of Gaussian distributions' components used.

Traditional Machine Learning Models		Advanced CNNs' Architectures			
Model	Accuracy	DEM-SNN		DEM-Caps	
SVM($\gamma = 0.001$, C=10, without PCA)	70%	Model	Accuracy	Model	Accuracy
SVM($\gamma = 0.0001$, C=10, without PCA)	87%	Standard CNN with Softmax layer	82%	DME-CAPS without augmentation	93.4%
SVM($\gamma = 0.0001$, C=10, PCA=14)	85%	Standard CNN with Sigmoid layer	90%	DME-CAPS with augmentation	95.3%
GC/unsmoothed ($\alpha = 1$, PCA=14)	85%	SNN with offline pair mining	91%	DEM-GCN	
GC/smoothed ($\alpha = 0.9$, PCA=12)	83%	SNN with offline triplet mining	92%	Model	Accuracy
GMM ($\alpha = 0.9$, PCA=29, K=1)	82%	SNN with online pair mining	94.3%	DEM-GCN with SLIC	80%
GMM ($\alpha = 0.9$, PCA=29,K=2)	89%	SNN with online triplet mining	94.5%	DEM-GCN with Quickshift	83%

compared to the other models. In the future, we aim to study the effect of self-attention mechanism [6], which integrates the concept of relevance by focusing only on the relevant aspect (detect the culvert/bridge area in our case) of the given input, on our proposed models and its efficacy on DEM.

ACKNOWLEDGEMENT

This work is supported by a grant awarded by the National Science Foundation (1951741). We thank Sourav Bhadra for his data and modeling support.

REFERENCES

[1] Sourav Bhadra, Ruopu Li, Di Wu, Guangxing Wang, and Banafsheh Rekabdar. Assessing the impacts of anthropogenic drainage structures on hydrologic connectivity using high-resolution digital elevation models. *Transactions in GIS*, 2019.

[2] Bureau, U. S. C. . TIGER/Line Shapefiles. retrieved from:. https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html, 2014. Accessed: 2010-09-30.

[3] Sounak Dey, Anjan Dutta, J Ignacio Toledo, Suman K Ghosh, Josep Lladós, and Umapada Pal. Signet: Convolutional siamese network for writer independent offline signature verification. *arXiv preprint arXiv:1707.02131*, 2017.

[4] Terrence S Furey, Nello Cristianini, Nigel Duffy, David W Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.

[5] Geoffrey E Hinton, Sara Sabour, and Nicholas Frosst. Matrix capsules with em routing. In *International conference on learning representations*, 2018.

[6] Assaf Hoogi, Brian Wilcox, Yachee Gupta, and Daniel L Rubin. Self-attention capsule networks for image classification. *arXiv preprint arXiv:1904.12483*, 2019.

[7] Gordana Jakovljevic, Miro Govedarica, Flor Alvarez-Taboada, and Vladimir Pajic. Accuracy assessment of deep learning based classification of lidar and uav points clouds for dtm creation and flood risk mapping. *Geosciences*, 9(7):323, 2019.

[8] Ling Jiang, Yang Hu, Xilin Xia, Qiuhua Liang, Andrea Soltoggio, and Syed Rezwan Kabir. A multi-scale mapping approach based on a deep learning cnn model for reconstructing high-resolution urban dems. *Water*, 12(5):1369, 2020.

[9] Syed Kabir, Sandhya Patidar, Xilin Xia, Qiuhua Liang, Jeffrey Neal, and Gareth Pender. A deep convolutional neural network model for rapid prediction of fluvial flood inundation. *Journal of Hydrology*, 590:125481, 2020.

[10] Ruopu Li, Zhenghong Tang, Xu Li, and Jessie Winter. Drainage structure datasets and effects on lidar-derived surface flow modeling. *ISPRS International Journal of Geo-Information*, 2(4):1136–1152, 2013.

[11] Qichao Liu, Liang Xiao, Jingxiang Yang, and Zhihui Wei. Cnn-enhanced graph convolutional network with pixel-and superpixel-level feature fusion for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 2020.

[12] Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 723–731, 2019.

[13] Kai Qiao, Chi Zhang, Linyuan Wang, Jian Chen, Lei Zeng, Li Tong, and Bin Yan. Accurate reconstruction of image stimuli from human functional magnetic resonance imaging based on the decoding model with capsule network architecture. *Frontiers in neuroinformatics*, 12:62, 2018.

[14] Carl Edward Rasmussen et al. The infinite gaussian mixture model. In *NIPS*, volume 12, pages 554–560, 1999.

[15] Santosh Rijal, Guangxing Wang, Philip B Woodford, Heidi R Howard, JM Shawn Hutchinson, Stacy Hutchinson, Justin Schoof, Tonny J Oyana, Ruopu Li, and Logan O Park. Detection of gullies in fort riley military installation using lidar derived high resolution dem. *Journal of Terramechanics*, 77:15–22, 2018.

[16] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.

[17] Francesca Strappini, Elad Gilboa, Sabrina Pitzalis, Kendrick Kay, Mark McAvoy, Arye Nehorai, and Abraham Z Snyder. Adaptive smoothing based on gaussian processes regression increases the sensitivity and specificity of fmri data. *Human brain mapping*, 38(3):1438–1459, 2017.

[18] Sameerah Talafha, Banafsheh Rekabdar, Chinwe Pamela Ekenna, and Christos Mousas. Attentional adversarial variational video generation via decomposing motion and content. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 45–52. IEEE, 2020.

[19] Meng Tang, Federico Perazzi, Abdelaziz Djelouah, Ismail Ben Ayed, Christopher Schroers, and Yuri Boykov. On regularized losses for weakly-supervised cnn segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 507–522, 2018.

[20] USDA-FSA. National Geospatial Data Asset (NGDA) NAIP Imagery.retrieved from:. https://www.fsa.usda.gov/programs-and-services/aerial-photography/imagery-programs/naip-imagery/, 2015. Accessed: 2010-09-30.

[21] USGS. 3D Elevation Program-3DEP. retrieved from:. https://www.usgs.gov/core-science-systems/ngp/3dep/, 2015. Accessed: 2010-09-30.

[22] USGS. 1/3rd arc-second Digital Elevation Models (DEMs) - USGS National Map 3DEP Downloadable Data Collection. retrieved from:. https://www.sciencebase.gov/catalog/item/4f70aa9fe4b058caae3f8de5/, 2017. Accessed: 2010-09-30.

[23] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. Gated siamese convolutional neural network architecture for human re-identification. In *European conference on computer vision*, pages 791–808. Springer, 2016.

[24] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In *European conference on computer vision*, pages 705–718. Springer, 2008.

[25] Abdul Waheed, Muskan Goyal, Deepak Gupta, Ashish Khanna, Fadi Al-Turjman, and Plácido Rogerio Pinheiro. Covidgan: data augmentation using auxiliary classifier gan for improved covid-19 detection. *Ieee Access*, 8:91916–91923, 2020.

[26] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[27] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018.

[28] Xinxin Zhou, Zhaohui Zhang, Lizhi Wang, and Pengwei Wang. A model based on siamese neural network for online transaction fraud detection. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019.