IEEE Open Journal of Systems Engineering

A Dataset Annotation System for Snowy Weather Road Surface Classification

MOHAMED KARAA[1], **HAKIM GHAZZAI**[2], **AND LOKMAN SBOUI**[1]

[1]Systems Engineering Department, Ecole de Technologie Supérieure (ÉTS), Montreal, Canada
[2]King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

CORRESPONDING AUTHOR: Hakim Ghazzai (e-mail: hakim.ghazzai@kaust.edu.sa).

ABSTRACT In this paper, we introduce an artificial intelligence-based annotation system for a dataset of snow-covered road images. We operate on a large dataset consisting of CCTV images and time and weather metadata. The dataset is fed to a series of data processing techniques to automatically assign each image one of four snow-cover categories aligned with snow removal operations. The processing pipeline includes feature learning using convolutional autoencoders and graph clustering using the Louvain community detection algorithm. The resulting dataset comprises over 41000 images automatically annotated in different weather and time settings. We train and test multiple deep learning models to validate the annotated dataset to classify snow-covered road images. We customize the models to consider the class distribution within the dataset. We achieve precision and recall scores of 97% using an EfficientNet model trained on separate day and night image datasets and using a class-weighted loss function.

INDEX TERMS Intelligent transportation systems, automated image annotation, road surface condition estimation, clustering, classification.

I. INTRODUCTION

One of the primary purposes of Intelligent transportation systems (ITS) mitigating the difficulties of mobility in adverse and unusual weather conditions [2], [3]. In this case, road safety and traffic management are promising applications. Artificial intelligence (AI) can be used to predict road hazards, anticipate accidents by monitoring vehicles' interior and exterior conditions, and optimize the traffic flow [4], [5]. One particular task that AI can perform is road surface condition (RSC) estimation. RSC is important during winter weather events such as rain and snowfall. In fact, many metropolitan areas in the northern hemisphere experience heavy snowfalls during the winter. In these severe weather conditions, vehicle collisions are more likely to happen as road surface becomes wet and slippery. Statistics from the Canadian National Collision Database mention that 30% of car accidents in 2020 occurred on snowy or icy roads. The U.S. Federal Highway Administration records nearly one thousand deaths in vehicle crashes during snowfall.

Multiple related studies were carried out in the context of RSC in snowy weather to predict the snow-cover type and depth. In [6], the authors combined both weather and surface conditions estimation to identify three weather conditions (clear, light snowfall, and heavy snowfall) and three surface covers (dry, wet, and snowy). They achieved a detection accuracy of 97% for weather conditions and 99% for surface conditions using ResNet18 architecture. However, this work only includes images of interstate webcams and does not discuss snow cover depth. Carrillo et al. compared the performance of different deep learning models on classifying images into bare pavement, partial snow cover, and full snow cover [7]. They achieved an accuracy of 91% using model trained on 14000 images collected from road weather information systems. They also discussed the effect of model fine-tuning and model size on the accuracy of RSC task. Finally, the authors combined the model outputs with other weather measurements (temperature, humidity, and pressure) into machine learning algorithms to boost performance. However, the work only focused on fine-tuning the fully connected layers of the pre-trained models which were trained on generic large datasets. This approach limits the networks to learn more robust features from road images that are fundamentally different from the pre-training datasets. Similarly, Pan et al. used pre-trained models to distinguish road surface conditions based on the percentage of the covered surface [8], achieving an accuracy of 97% on an other benchmark dataset of 33000 images. In this work, the authors acknowledge the need to fine-tune the models progressively to adapt more to the snowy road images. They

A part of this work has been accepted for presentation at the IEEE 18th Asia Pacific Conference on Circuits and Systems (IEEE APCCAS 2022) [1].

refined the fully connected layers parameters first and later a set of convolutional layers. In [9], the authors performed the classification of snowy roads cover using the model developed in [7]. The authors fine-tuned this model on a novel dataset of urban images unlike the previous works that only include rural areas. By refining classification head layers and convolutional layers progressively, the model reaches an F1-score of 98.4%.

The aforementioned studies employ labeled data to train models as a necessary component for supervised learning methods. In fact, unlabeled data is a major issue that faces the evolution of AI models, especially when manual annotation is difficult, as it requires more time as its size increases. Furthermore, manual labeling requires domain knowledge and reviews to generate accurate labels. Many research works tried to develop new learning algorithms to perform automatic image annotation tasks [10]–[12] with different degrees of supervision. Some of these works focused on unsupervised learning methods, specifically relying on autoencoders (AEs) and clustering to discover underlying representations within image datasets that separate different classes. In [13], Jianwei et al. proposed a method that combined image clustering and representation learning in training a convolutional neural network (CNN), such as the forward pass includes an agglomerative clustering step, and learning is performed in the backward pass. The authors propose a recurrent method where clusters are merged progressively till reaching the target number of clusters. In addition, they prove that the learned representations generalize well with other clustering algorithms. This can be beneficial for more complex clustering tasks. Similarly, Mathilde et al. [14] implemented an iterative clustering method that learns the CNN parameters and the cluster assignment of the extracted features. The algorithm takes in images as input, extracts features, performs k-means clustering on them, and iteratively retrains the network in a supervised way using assigned clusters as pseudo-labels compared to a classification head predictions. In another study, researchers presented deep autoencoder-based clustering (DAC) [15], a method that leverages an autoencoder to learn robust representations of input data and then uses the k-means algorithm to clusters these representations. The model is trained using a cluster-weighted mean squared error loss where the weights help separate distant features and aggregate similar ones. However, the method is validated only for a simple datasets such as MNIST that does not include complex or rich classes. In [16], researchers extracted embeddings learned by a convolutional variational autoencoder (VAE). The VAE is able to learn complete and continuous latent space representations that are suitable for clustering. Aljalbout et al. [17] provided comprehensive insights of deep clustering methods including autoencoder architectures, loss functions, clustering algorithms, and training strategies.

The described works on winter RSC detection used different image datasets representing different locations and classes. These datasets were specific to each work and were not published as benchmark datasets. This approach makes it hard to perform a fair comparison of the proposed models performance. For these reasons, it is necessary to have a benchmark dataset for RSC estimation in winter weather. In [18], we have proposed a similar image dataset for snow-covered roads in urban scenes. However, this large dataset lacks image labels. We have proposed a benchmark problem to automate the dataset annotation.

As mentioned earlier, the reviewed deep clustering methods used to explore classes within datasets are tested on general and less complex datasets such as MNIST, ImageNet, and YFCC100M, unlike the snow-covered road images that represent more complex scenes. Therefore, we adopt a system engineering approach to design an AI-based automated framework to annotate the snow-covered road dataset into four different categories. After defining the requirements, we have decomposed the framework into modules and steps, i.e., data pre-processing, feature extraction using CAEs, and unsupervised clustering. Afterwards, we perform a deep learning based classification to validate the content of datasets in categorizing unknown images.

The contributions of this paper can be summarized as follows:

- We solve the benchmark problem of snow-covered road image dataset annotation by proposing an AI-based annotation system.
- We leverage image processing and unsupervised learning techniques to build the annotation system and generate a labeled image dataset of snow-covered roads.
- We develop deep learning models to estimate snow level covering the roads from CCTV cameras.

The developed classification models are trained and tested on on the benchmark dataset[1]. We show that a 97% of accuracy, precision, and recall can be achieved using an EfficientNet model.

In the remainder of the paper, we first present an overview of the components of our developed system in Section II. Section III provides the steps of the annotation system. Section IV discusses the snow-covered road classification method. Next, we benchmark both the annotation and classification results in Section V. The paper is concluded in Section VI.

II. METHODOLOGY

A. OVERVIEW

In this work, we solve the benchmark problem proposed in [18]. We build an unsupervised learning-based system for snow-covered road image dataset annotation. The annotation system takes iteratively image sets from different cameras and explores clusters within these sets that correspond to four different snow cover classes. These classes are the following:

[1]The annotated dataset is temporary available at: https://github.com/mohamedkaraa/Snow-Covered-Roads-Dataset

This article has been accepted for publication in IEEE Open Journal of Systems Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSE.2024.3391326

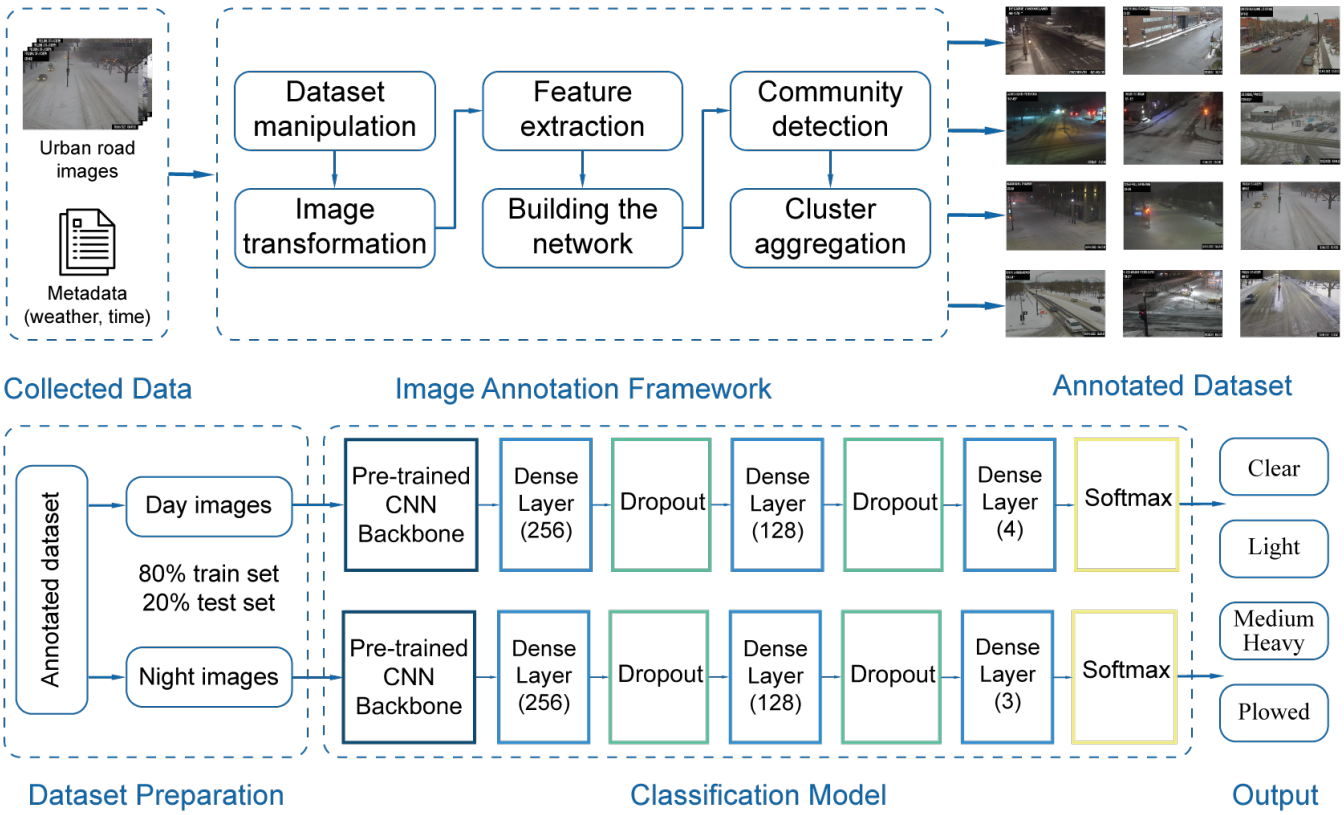


FIGURE 1: Overview of the annotation system and the classification model of snow-covered road images. The benchmark image dataset goes through an AI-based annotation system. The annotated dataset is used to train a CNN model for snow-covered road classification.

"clear surface", "light-covered surface", "medium-to-heavy-covered surface", and "plowed surface". The framework consists of a processing pipeline that includes transformations, feature extraction, and clustering. Afterwards, we train a deep learning classification model to test the effectiveness of the dataset. Fig. 1 shows a step-by-step overview of the annotation framework and the classification model.

Initially, we sample images from cameras having a constant view. Then, we split the images according to the time (day and night) and according to a macro-categories (clear and snowy surfaces) based on timestamps and weather conditions. This operation simplifies the annotation process. Next, we transform the images into a more representative format of the road surface cover. We enhance the contrast to accentuate details on the road surface. Then, we create binary images that describe snow cover in white and reduce redundant world details such as buildings, trees, and other static objects. We later use a Convolutional Autoencoder (CAE) architecture to extract features from the binary images. In the CAE's latent space, the input images dimensionality is reduced and the extracted features are more suited for clustering, i.e., representations of similar images are closer to each other and well separated from those of different ones. Finally, we use a community detection algorithm to cluster a graph that describes the extracted latent features and generates clusters corresponding to the defined labels. Hence, we obtain an annotated dataset that describes four snow-covered road classes in an urban context.

After the annotation step, we train different CNNs using the generated dataset to classify snow-covered road images. The purpose of the classification is to validate the usefulness of the dataset, and to automate the identification of road conditions. We notice that the models' performance is limited due to the imbalance of the dataset. We incorporate a weight term within the loss function to solve this problem, and train different models for day and night subsets.

III. AUTOMATED IMAGE DATASET ANNOTATION

This section details the process of the image annotation system. First, we perform a preprocessing step on the dataset hierarchy. Then, we apply an image transformation that better represents snow cover. Next, we use a CAE to extract latent features representing images. To generate the labels, we cluster the extracted features using a community detection method. We justify the technical choices based on visual observation or defined constraints of the clustering problem.

A. DATASET MANIPULATION

We simplify the dataset for clustering by performing preprocessing manipulations. First, we sample cameras that

kept a constant view during the image scraping. This step enables focusing on the variation of snow cover rather than the change of view. Images collected during the day and night periods show notable differences. Illumination in night images comes from artificial light, while day image illumination depends on weather conditions. In addition, day scenes display more traffic volumes. For these reasons, we divide the image sets into day and night times, using the metadata timestamps. A final manipulation would be to extract "clear surface" images using weather information. Clear roads coincide with periods of clear weather, absence of snowfall, and periods with no prior snowstorms. These conditions are determined based on the weather metadata and prior knowledge of the collection period. This final manipulation reduces the targets of the clustering to three classes. From this step onward, we iterate through the camera image sets and process them separately to achieve better annotation results.

B. IMAGE TRANSFORMATION

We perform a transformation step to guarantee a better representation of the images before using them as input for the clustering algorithm. This transformation helps reflect the snow cover, reduce irrelevant world details, and accentuate the relevant features. We convert color images into grayscale, then apply the contrast-limited adaptive histogram equalization (CLAHE) algorithm for contrast enhancement followed by images binarization. Fig. 2 depicts the result of such transformation, where white areas represent snow cover and black represents the bare road. The stripes represent the plowing traces.

C. FEATURE EXTRACTION

After making images more representative of snow cover, we would transform them into another feature space that is more suitable for clustering. In such space, the representations of similar images are close to each other and well separate from those of different images. To do so, we leverage a CAE to extract representations of the binary images in an unsupervised way. In fact, AEs are neural networks designed for unsupervised learning, with the goal of learning efficient representations or encoding of input data. AEs are composed of an encoder mapping the input data to a lower-dimension latent representation, and a decoder network that reconstructs the input data from that representation. CAEs are a variant of AEs that uses convolutional layers to handle images. Typically, input images are passed through convolutional layers in the encoder network to capture spatial features. In the latent space, the output of the encoder is a flattened to create a compressed representation of the input data that holds the most important features. Later, the latent vector is passed through the decoder gradually increasing the spatial dimensions in order to reconstruct the input image.

For each camera image set, we train the CAE using the binary images as input. The reconstruction is equivalent to a binary classification where each pixel is estimated as 0 or 1. Hence, we aim to minimize the binary cross-entropy loss defined as:

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i}), \quad (1)$$

where N is the output size of the CAE, y_i is the input image and $\hat{y}_i$ is the reconstructed image.

We propose an asymmetric CAE architecture that extracts relevant features characterizing the road's snow cover while reducing the dimensionality of the input images in half. The input layer has a shape of $480 \times 720 \times 1$ while the code size is 172800 ($120 \times 180 \times 8$ being the shape of the last encoder layer). Fig. 3 details the CAE architecture. Along with the convolutional layers, we introduce batch normalization layers to prevent the autoencoder from overfitting. We count 51313 total parameters, of which 50833 are trainable.

To understand the transformation undergone by the binary image data space, we use a t-SNE projection to visualize the initial data points (binary input images) and the extracted representations. T-SNE is a helpful dimensionality reduction technique for high-dimensionality data visualization. Fig. 4 shows an example of a single camera image set. In Fig. 4b, we can observe the formation of clusters and dense point regions separated from others, unlike in Fig. 4a, where data points have a random distribution. This shows that the CAE is able to transform the input images into a feature space more suitable for clustering. This observation is yet to be confirmed by a clustering algorithm.

D. CLUSTERING OF DIFFERENT SNOW LEVELS

Once we extract the features of snow-covered road images at each camera image set level, we need to cluster them into groups representing each class.

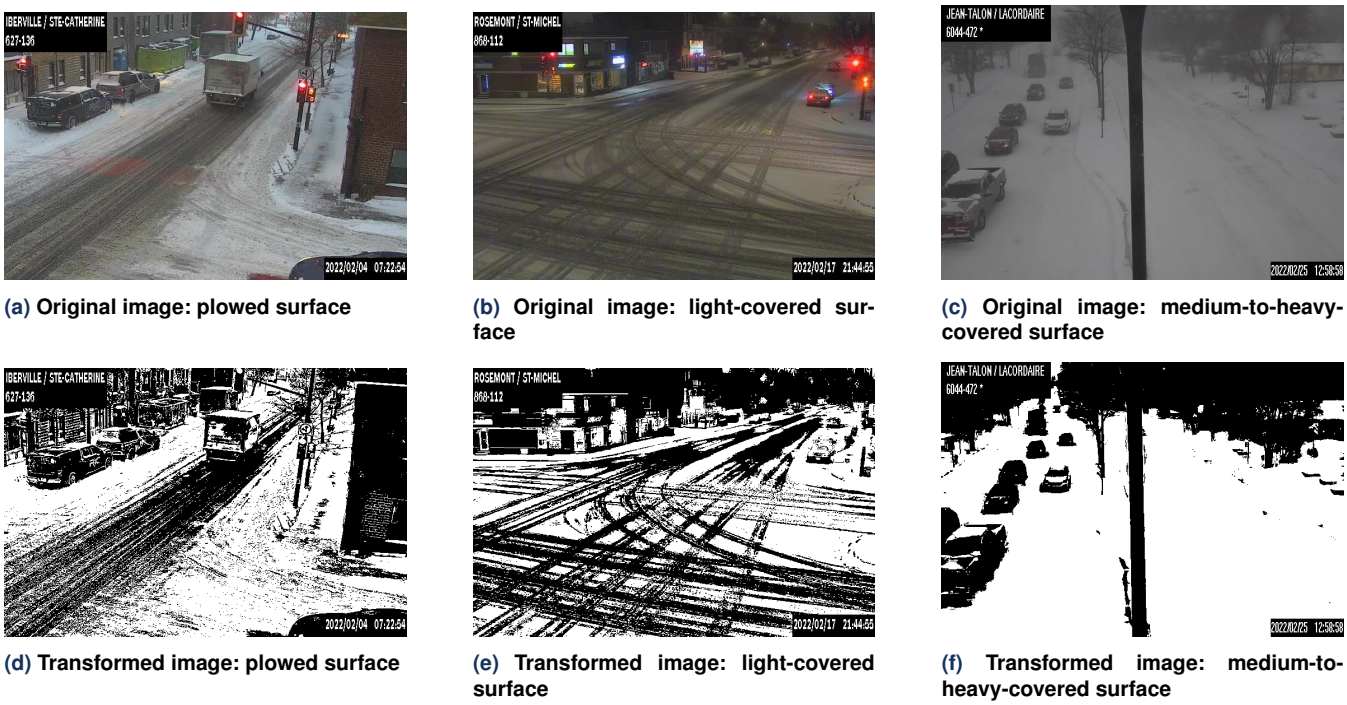
1) CONSTRAINTS

It is worth noting that the benchmark dataset imposes some constraints. As we treat each camera image set separately, we can not assume that all the target classes are present within this set. Hence, the number of clusters is unknown before clustering. Another constraint is the presence of outlier images that may not hold any information about the snow cover level, such as images from snow-covered cameras or cameras not pointing towards the road. These outliers should be detected and filtered out as their presence adds noise to data, i.e., erroneous annotation. Having such constraints, we can not use traditional clustering algorithms such as the k-means algorithm, as it requires prior knowledge of the number of clusters.

2) COMMUNITY DETECTION

To tackle the clustering with respect to the defined constraints, we propose using a community detection method to cluster the extracted representations. Community detection aims to identify groups of similar nodes within a weighted graph such as a community has densely connected nodes,

This article has been accepted for publication in IEEE Open Journal of Systems Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSE.2024.3391326



(a) Original image: plowed surface

(b) Original image: light-covered surface

(c) Original image: medium-to-heavy-covered surface

(d) Transformed image: plowed surface

(e) Transformed image: light-covered surface

(f) Transformed image: medium-to-heavy-covered surface

FIGURE 2: Examples of the transformation applied to sample images to better represent snow cover.

and different communities have sparsely linked nodes [19]. Community detection algorithms optimize the modularity metric that measures the quality of detected communities. In Section V.B, we will compare the performance of the proposed community detection method to a conventional clustering technique to validate our choice.

To detect communities (clusters) within the representations graph, we use the Louvain algorithm for community detection introduced by Blondel et al. [20]. The algorithm is based on modularity optimization, where each node is initially assigned to a community. Each node i is moved to the communities of its neighbors j. The node i is only assigned to the community of which the move leads to a positive gain in modularity. Resolution γ is an important parameter in the Louvain algorithm that affects the number of discovered partitions (communities). When $\gamma \to \infty$, the number of communities equals the number of nodes, and when $\gamma = 0$, only one community is discovered containing all the nodes.

In our case, we create a weighted graph representing each camera image set. A first intuition would be to have the latent representations, extracted by the CAE, as the graph's nodes and the Euclidean distance between the nodes as the vertice's weights. However, this approach generates a complete graph that would be hard to explore communities within. Instead, we create a simple graph linking each node to its k-nearest neighbor nodes. Fig. 5 explains the selection of the k parameter, where we plot the variation of modularity and the number of detected communities with respect to the graph resolution for different values of k. The figure shows the results obtained for the same previously used camera image set. However, similar behavior is observed for the other sets of images. This approach is generalized to the other sets as they have comparable instances number (as images are collected within the same period for all cameras) and also class instance number (i.e., same geographical locations that imply similar road condition distribution at a given moment). Smaller k values result in higher modularity values. For $k = 2$, we score higher modularity values, yet the generated graph is sparser and yields much more communities. For $k = 3$, there is more balance between modularity and the number of communities. Therefore, we set $k = 3$ for all the camera image sets.

After setting the k parameter, We perform the Louvain algorithm to cluster the latent representations extracted from CAE. For every camera image set, we select the resolution value from a value range such as it maximizes the modularity to generate the best possible partitioning for every single image set. Fig. 6 shows an example of discovered communities within a graph of latent representations. We observe that nodes within a community are densely connected while having sparse connections with other communities' nodes. We also notice that some communities (clusters) are closer to each other, which may suggest their similarity.

E. CLUSTER AGGREGATION

The proposed community detection method detects clusters often superior to the number of the desired clusters (target labels). The generated communities are granular partitions of the three snow levels. We explain the granularity by the fact

This article has been accepted for publication in IEEE Open Journal of Systems Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSE.2024.3391326

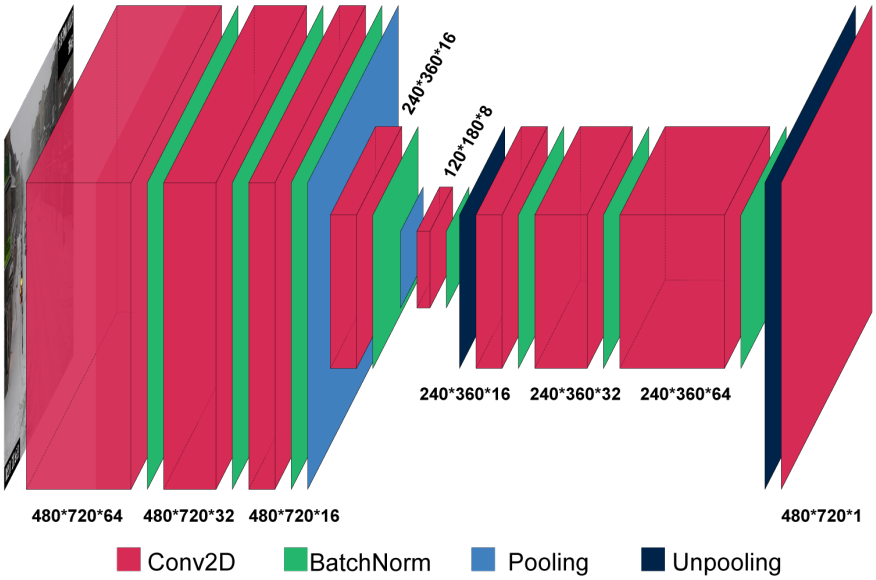
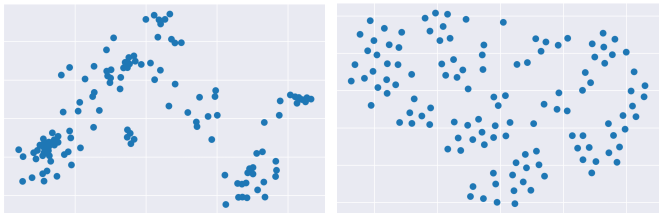


FIGURE 3: The proposed convolutional autoencoder architecture for the extraction of the representations from camera image sets.



(a) Initial projection of input binary images before applying CAE. **(b) Projection of image representation points after applying CAE.**

FIGURE 4: t-SNE 2D projections of binary images and their latent representations.

that one class could be present in different conditions within a camera image set, such as weather or illumination. Hence, we aggregate the clusters to match the defined targets. We assign a virtual center for every cluster by calculating the mean of the representation vectors of its members. We opt for this method because these centers would, to some extent, represent their clusters as they result from all the points.

The cluster aggregation is based on the similarity measure between the virtual centers. We calculate the cosine similarity for every pair of centers, defined as the cosine value of the angle between the two virtual center vectors. Then, we connect each cluster with the one having the most similar center to form larger clusters. At this point, the "light-covered surface" clusters are often identified as "plowed surface" or "medium-to-heavy-covered surface" clusters because of their visual similarity. We solve this problem using the timestamps associated with the images. Knowing the starting time of snowstorms, we extract clusters corresponding to the "light-covered surface" class, if they exist, before performing aggregation. The aggregation is performed only for the two remaining classes. Fig. 7 shows the resulting aggregated clusters (communities) projection for the sample example camera image set. The aggregation generates separate macro-clusters. We also notice the presence of points that might be misinterpreted at the cluster boundaries as other classes, due to their very similar features.

IV. SNOW-COVERED ROAD CLASSIFICATION

In this section, we exploit the automatically annotated dataset and put it into use to train a deep learning model for snow-covered road classification. The model should learn to accurately predict the road state based on training samples that present many challenges.

A. DATASET PREPARATION

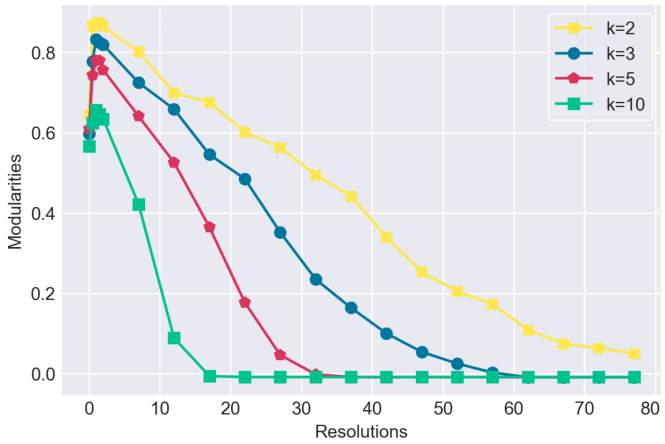
To train the model, we split the dataset into train and test sets with proportions of 80% and 20%, respectively. We randomly shuffle and split images of each class to keep the same proportions for all classes.

As we mentioned earlier, the "light-covered surface" class is the least represented during daytime as it has less occurrence than other states. To augment the number of instances of this class, we leverage the camera image sets that were unused in the annotation process, i.e., those with a changing view by sampling "light-covered surface" images using the timestamps during daytime.

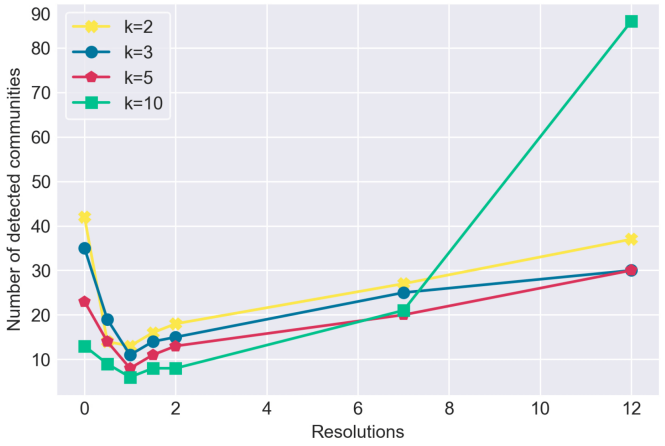
B. SNOW-COVERED ROAD CLASSIFICATION MODEL

To perform the classification, we use pre-trained CNNs, which we adapt to the snow-covered road context by excluding their top layers and replacing them with fully connected layers and an output size corresponding to target classes. The first CNN we use is a pre-trained ResNet50 [21] model that we replace its final layers with two consecutive fully

This article has been accepted for publication in IEEE Open Journal of Systems Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSE.2024.3391326



(a) Variation of the modularity measure with respect to the graph resolution for different values of k.



(b) Variation of the number of detected communities with respect to the graph resolution for different values of k.

FIGURE 5: Selection of the k parameter for building the image representations graph.

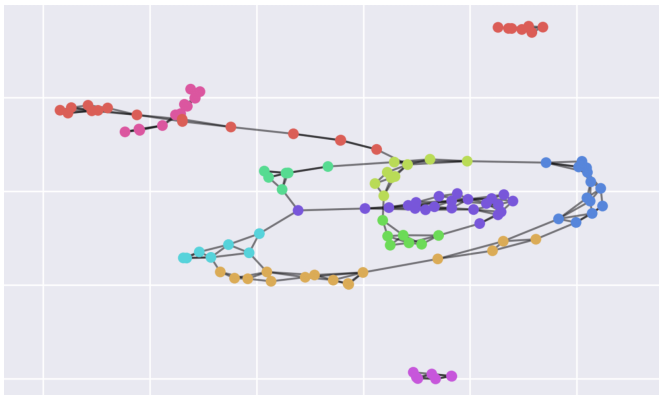


FIGURE 6: Discovered communities by the Louvain algorithm within the image representations graph.

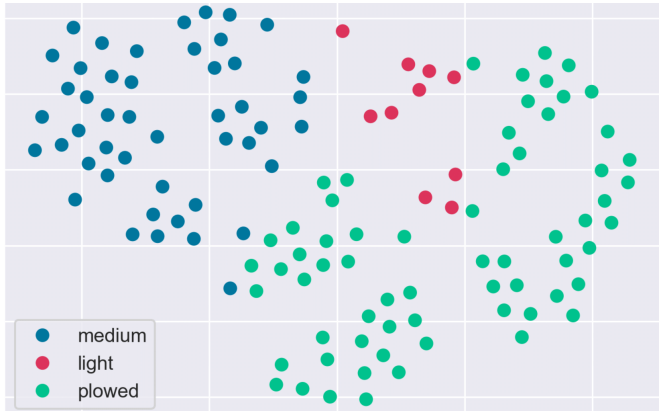


FIGURE 7: 2D Projection of aggregated clusters resulting from the community detection method.

connected layers and an output fully connected layer of size 4 to which we apply the softmax function. The second CNN is an EfficientNet [22] to which we apply the same modification as the ResNet to its network head. The ResNet50-based model has ∼74 million parameters, and the EfficientNet has ∼11 million parameters in total (EfficientNetB3 variant).

The four road cover classes are naturally not present in equal proportions. Hence, we expect having an imbalanced dataset, i.e., some classes have an instance count much higher than others. This might affect the classification model performance if not taken into consideration. In fact, it will be harder to predict minority classes when there are not enough training samples. We introduce class weights to the loss function while training to cope with this issue. Class weights are coefficients that help consider the skewed distribution of the classes while optimizing the loss function. The objective is to penalize the misclassification of the minority classes by setting a higher class weight. Usually, the majority classes are assigned lower class weights, though in our case, we keep them unweighted. For each class i, the class weight ω_i is given by:

$$\omega_i = \begin{cases} \frac{n}{N \times n_i} & \text{if } i \in [\text{light, plowed}] \\ 1 & \text{else.} \end{cases}, \tag{2}$$

where N is the number of classes, n is the total number of dataset instances, and n_i is the number of samples in class i. The loss function is expressed as:

$$\text{Loss} = -\sum_{i=1}^{N} \omega_i y_i \log(\hat{y}_i). \tag{3}$$

where y_i is the ground truth label, $\hat{y}_i$ is the predicted value and N is the number of classes.

V. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the annotation system by experimenting with different AE configurations, comparing the community detection component to another clustering method, and displaying the resulting dataset. We also perform an empirical study to test multiple classification models for different settings to validate the developed dataset and its practical application.

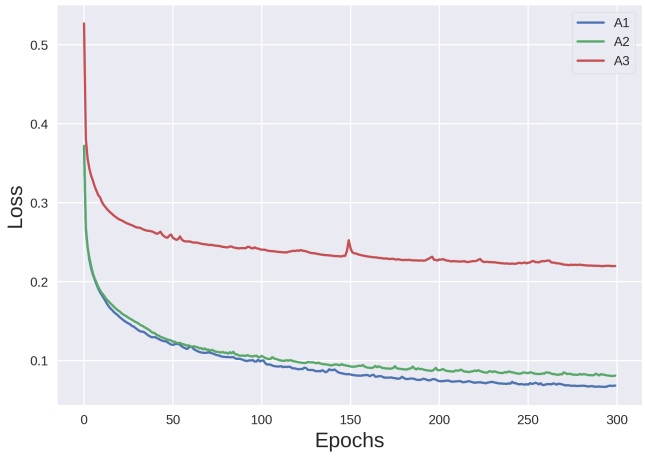


FIGURE 8: Evolution of the loss function for the three autoencoder architectures for a single camera image set.

A. ANNOTATION SYSTEM EVALUATION

1) AUTOENCODER ARCHITECTURE SELECTION

To study the performance of the CAE for dimensionality reduction and feature representation, we test different architectures based on their size. We denote d, the model's depth, as the number of pooling operations in the encoder and p as the total parameters number. We distinguish three architectures A1, A2, and A3 having respectively the following characteristics: $(d = 2, p \approx 51000)$, $(d = 2, p \approx 13000)$, and $(d = 3, p \approx 13000)$

We use a Tesla P100 GPU with 22Gb video memory for the experiments. We train each autoencoder for 300 epochs (per camera image set) using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 8 images. Fig. 8 shows the evolution of the binary cross-entropy loss function for the three AE architectures on a sample camera image set.

Table 1 presents the average performance of the architectures A1, A2, and A3. We split the image sets into two subsets (day and night) and measure the average accuracy and loss for each set and architecture. Accuracy for the CAE is defined as:

$$\text{Accuracy} = \frac{\text{number of correctly reconstructed pixels}}{\text{total number of image pixels}}, \quad (4)$$

as it consists of predicting pixel values of 0 and 1 for binary images. The model was trained for 249 camera image sets with a standard deviation of the loss function value of 0.024, 0.024, and 0.047 for autoencoders $A1$, $A2$, and $A3$ respectively. The architecture A1 demonstrates the highest accuracy and lowest error, with less depth and more parameters. We select this architecture to perform the next annotation steps. We proceed to hyperparameter tuning for the selected autoencoder $A1$, specifically the learning rate and batch size in order to select an optimal configuration for the annotation framework. The fine-tuning is performed on a randomly selected validation set of 20 camera image sets. The results are reported in Table 2.

TABLE 1: Comparison of the convolutional autoencoders performance averaged on all image sets.

	Metric \ Architecture	A1 ($d = 2$, $p \approx 51000$)	A2 ($d = 2$, $p \approx 13000$)	A3 ($d = 3$, $p \approx 13000$)
Day	Average accuracy	**0.97**	0.96	0.90
	Average loss	**0.05**	0.07	0.20
Night	Average accuracy	**0.97**	0.96	0.91
	Average loss	**0.06**	0.07	0.18

TABLE 2: Autoencoder learning rate (lr) and batch size (bs) tuning

Hyperparameters \ Metrics	Average BCE loss	Average MSE
(lr=10^{-2} , bs=4)	0.065	0.013
(lr= 10^{-3}, bs=4)	0.058	0.010
(lr=10^{-4} , bs=4)	0.079	0.017
(lr= 10^{-3}, bs=2)	0.045	0.007
(lr= 10^{-3}, bs=8)	**0.044**	**0.006**

2) CLUSTERING PERFORMANCE

a: METRICS

We define three metrics used to evaluate the clustering algorithms.

Modularity: a measure of the partitioning of a weighted graph introduced in [23]. Multiple community detection algorithms aim to unfold communities within graphs by optimizing the modularity measure. Modularity values range from -1 to 1, where values close to 1 indicate well-divided partitions. Modularity is given by:

$$\text{Modularity} = (1-t) + \frac{1}{2m}\sum_{i,j}[tA_{i,j} - \frac{k_i k_j}{2m}]\delta(c_i, c_j), \quad (5)$$

where t is a resolution parameter, A_{ij} represents the weight of the edge between i and j, $k_i = \Sigma_j A_{ij}$ is the sum of weights of edges attached to vertex i, c_i is the community to which vertex i is assigned, the δ-function $\delta(u, v)$ equals 1 if $u = v$ and 0 otherwise, and m is the total weight of the edges.

Coverage: The coverage of a graph partitioning C, as defined in [24], is the fraction of intra-cluster edges within the complete set of edges. It is defined as:

$$\text{Coverage} = \frac{m(C)}{m}, \quad (6)$$

where $m(C)$ is the number of intra-cluster edges and m is the number of edges in the graph. Larger coverage values occur for fewer clusters with higher density, implying better graph clustering.

Dunn index: It is an internal measure that depends on the clustered data itself. It determines the compactness of dis-

This article has been accepted for publication in IEEE Open Journal of Systems Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSE.2024.3391326

TABLE 3: Comparison of the community detection method and the density-based clustering method.

Metric \ Method	Community detection based method		Density-based clustering method	
	Day	Night	Day	Night
Modularity	**0.49**	**0.42**	0.40	0.41
Coverage	**0.97**	**0.99**	0.84	0.85
Dunn Index	**0.55**	**0.60**	0.47	0.44

covered clusters, i.e., having a small within-cluster variance, and their separation. A Higher Dunn index indicates better clustering. Dunn index is defined as:

$$DI_m = \frac{\min\limits_{1\leq i<j\leq m} \delta(C_i, C_j)}{\max\limits_{1\leq k\leq m} \Delta_k}, \tag{7}$$

where m is number of clusters, $\delta(C_i, C_j)$ is an inter-cluster distance metric, and Δ_k is the size or diameter of a cluster.

b: COMPARISON METHOD

We propose to compare the performance of the developed community detection algorithm to cluster our image dataset to one of the typical density-based clustering techniques. The latter is based on the assumption that a cluster is a dense region of data points, separated from other clusters by contiguous low density regions. We use the HDBSCAN algorithm [25], [26] to extract dense point distributions in the feature space. HDBSCAN is suitable for the problem as it does not require to define the number of clusters as input. In addition, HDBSCAN proved to outperform other similar methods such as DBSCAN and OPTICS.

HDBSCAN detects outlier points and does not assign them to any cluster. In our case, images representing the light-covered surface class, an underrepresented class, are often detected as outliers. To keep these images within the final output, we cluster the outliers only while reducing the minimum cluster size parameter. Finally, we filter points according to their cluster membership strength probability to obtain homogeneous clusters. We use the same aggregation step as in the community detection method.

We evaluate both clustering methods based on the defined metrics for the three snowy classes, using the same graph built for community detection. Table 3 showcases the benchmarking of the two methods. Our proposed approach achieves higher scores for modularity, coverage, and Dunn index for the totality of 249 camera image sets with standard deviation values of 0.069, 0.042, and 0.114 for the three metrics respectively. We can conclude that the community detection approach enables the generation of more cohesive clusters with a better ability to separate different classes.

In addition to comparison with HDBSCAN clustering, we compare our method to conventional clustering methods, namely agglomerative clustering and spectral methods. Unlike community detection and density-based clustering, these methods require a number of target clusters as an input parameter. We set this parameter to $k = 3$ corresponding to the number of target classes within the snow categories. We report the performance following the same previously used metrics on a validation set of 20 camera sets in Table 4. Our method shows superior performance compared to spectral clustering, and comparable to agglomerative clustering for the clustering metrics. However, when visualizing the resulting clusters, we observe that the "light-covered surface" class is not separated from the other classes. Adding to that the constraint of $k = 3$ forces the resulting clusters to have more assigned outliers.

TABLE 4: Comparison of the proposed community detection-based method and conventional clustering algorithms

Metric \ Method	Community detection	Agglomerative clustering	Spectral clustering
Modularity	**0.68**	0.54	-0.01
Coverage	**0.98**	0.94	0.35
Dunn Index	0.52	**0.61**	0.32

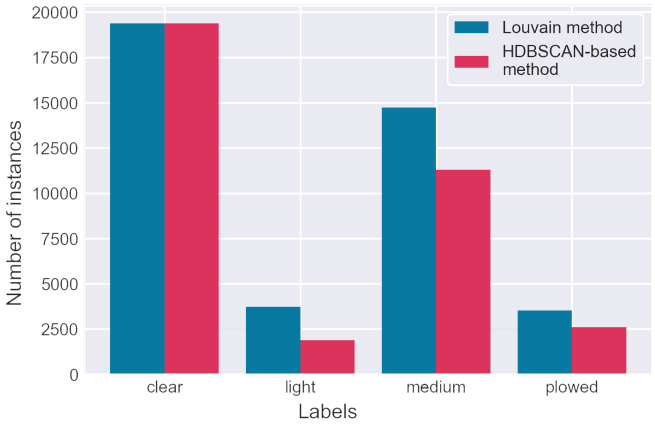


FIGURE 9: Distribution of the classes in the datasets annotated by both clustering methods.

3) FINAL ANNOTATED DATASET

The annotated dataset consists of 41346 images, distributed over describing four classes as follows:

- Clear surface: 17422 images.
- Light-covered surface: 3726 images.
- Medium-to-heavy-covered surface: 14725 images.
- Plowed surface: 3512 images.

TABLE 5: Classification results on the complete dataset with unweighted and weighted

Model	Accuracy (%)	Precision (%)	Recall (%)
ResNet50 (U)	83	86	81
ResNet50 (W)	**91**	**92**	**90**
EfficientNet (W)	90	91	90

Fig. 9 shows the class distribution for both clustering methods. The HDBSCAN-based clustering method demonstrates information loss due to its outlier detection component, especially for the "light-covered surface" class. It is also clear that the resulting dataset is imbalanced, where we have two majority classes and two minority classes. We performed manual test by randomly selecting 100 images from both minority classes and found out that 89% of the images have correct annotations.

B. SNOW-COVERED ROAD CLASSIFICATION EVALUATION

To classify snow-covered road images, we train and test multiple deep learning models using the previously mentioned dataset. In the first test, we use the ResNet model mentioned in Section IV.B and train it using the totality of the training dataset (33661 images) for 60 epochs. We use the Adam optimizer with an initial learning rate of 10^{-3} that we gradually decay to 5×10^{-4} and 10^{-4} after 15 and 30 epochs respectively. In this experiment, we compare the performance of the model trained with two distinct loss functions. The first one does not consider the dataset imbalance. We later use the loss function defined in (3) to train the model while taking into account the class imbalance.

Table 5 shows the model results on the test set after training on the totality of the train set using the two approaches. The class-weighted loss function approach noted as (W) delivers better performance than the model trained with the regular unweighted loss function noted as (U) for all accuracy, precision, and recall metrics. Nevertheless, the obtained scores are not satisfying and need improvement. The following models will use the class-weighted loss function when training to penalize the misclassification of minority classes. We proceed to fine-tuning the EfficientNet classifier hyperparameters, specifically learning rate (lr), batch size (bs), and weight decay (wd). We opt to fine-tune the model as it has fewer parameters than ResNet50 with comparable results. We report the resulting performance in Table 6 on the totality of the dataset.

In Table 7, we look into a more specific case of class distribution within the dataset. We notice that the "plowed surface" class is absent within night images due to the lack of samples representing it during the dataset collection. The second observation is that the models trained on the complete dataset perform poorly, specifically on "light-covered surface" images during the day. These images represent a third of the training set, in which some are mislabeled (limitations of the annotation system). For these reasons, we split the complete dataset into day and night datasets and then train the CNN models separately on each set using the same selected hyperparameters from Table 6. For each dataset, we compare two CNN architectures having as backbone a ResNet50 and EfficientNet. We modify the network head to fit three classes ("clear surface", "light-covered surface", and "medium-to-heavy-covered surface") for the night dataset and four classes ("clear surface", "light-covered surface", "medium-to-heavy-covered surface", and "plowed surface") for the day dataset. Table 7 demonstrates significant improvements reaching 97% on the test set for the three metric scores. After training with more specific datasets, the models can better distinguish the different classes. Finally, we decide to use the EfficientNet model since it less complex (fewer number of parameters) and allows faster training and testing times.

Fig. 10 showcases examples of correctly annotated images (rows 1 and 2) and mis-annotated images (row 3). The mis-annotated images are described as "true label" predicted as "false label".

TABLE 6: EfficientNet classification model hyperparameter fine-tuning

		Hyperparameters (lr, bs, wd)	Accuracy (%)	Precision (%)	Recall (%)
learning	rate	(10^{-3}, 4, 0)	0.419	0.235	0.419
		(5×10^{-4}, 4, 0)	0.425	0.238	0.425
		(10^{-4}, 4, 0)	0.815	0.823	0.815
batch	size	(10^{-4}, 8, 0)	0.892	0.892	0.892
		(10^{-4}, 16, 0)	0.869	0.864	0.869
weight	decay	(10^{-4}, 4, 10^{-5})	0.918	0.917	0.918
		(10^{-4}, 4, 2×10^{-5})	**0.923**	**0.921**	**0.923**

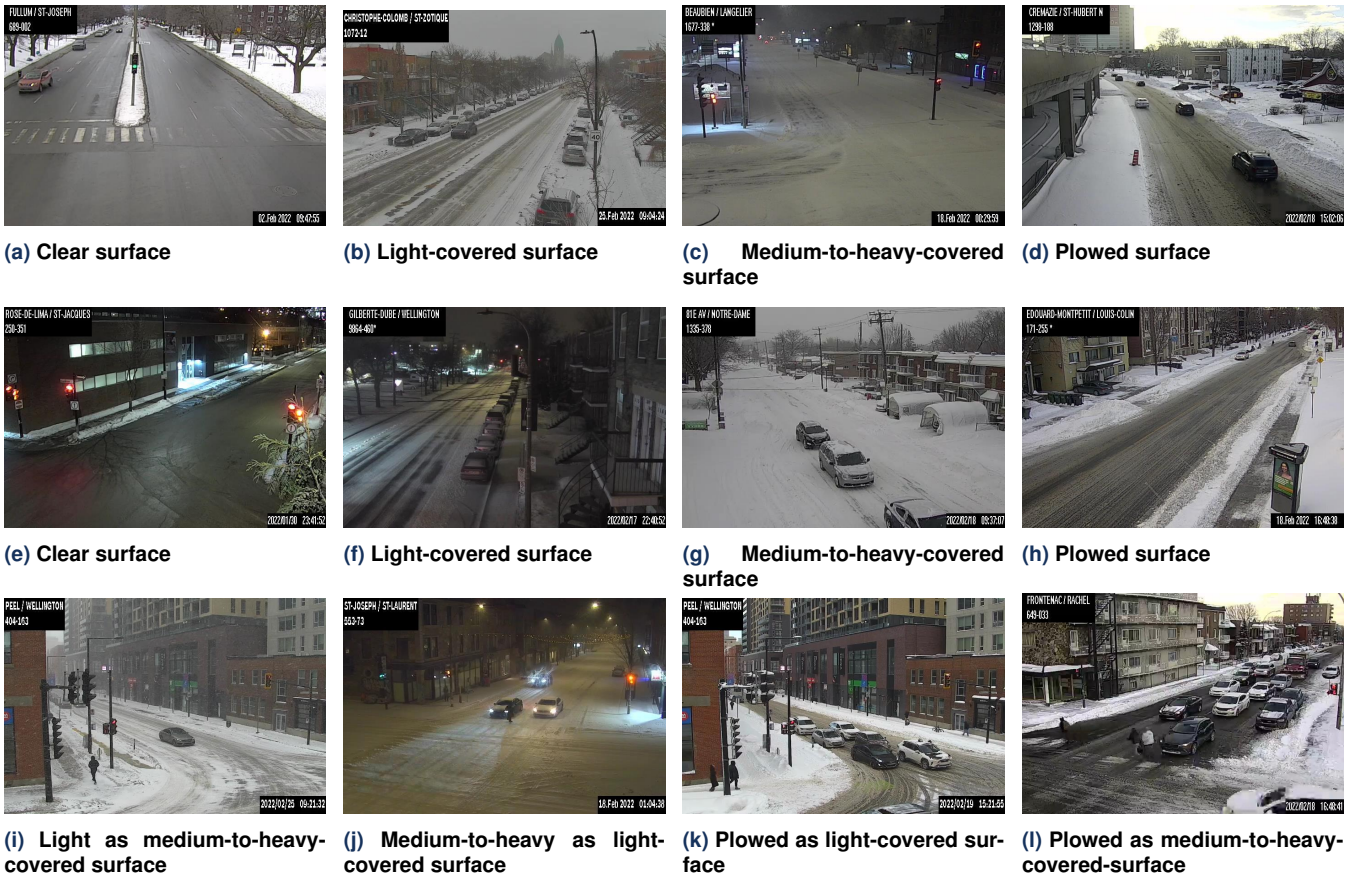
TABLE 7: Benchmark of the test classification models.

Dataset	Model	Accuracy (%)	Precision (%)	Recall (%)
Day	ResNet50 (W)	97.2	97.3	97.2
	EfficientNet (W)	**97.3**	**97.3**	**97.3**
Night	ResNet50 (W)	97.2	97.3	97.2
	EfficientNet (W)	**97.4**	**97.5**	**97.4**

C. FRAMEWORK COMPUTATIONAL COST

In this subsection, we report the computational cost of running the annotation framework and classification model in terms of execution time. We used a setup of 2 x AMD Milan 7413 @ 2.65 GHz CPU with 4 A100 GPUs (20Gb) for training, and 1 A100 GPU (20Gb) for inference. The

This article has been accepted for publication in IEEE Open Journal of Systems Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSE.2024.3391326



(a) Clear surface (b) Light-covered surface (c) Medium-to-heavy-covered surface (d) Plowed surface

(e) Clear surface (f) Light-covered surface (g) Medium-to-heavy-covered surface (h) Plowed surface

(i) Light as medium-to-heavy-covered surface (j) Medium-to-heavy as light-covered surface (k) Plowed as light-covered surface (l) Plowed as medium-to-heavy-covered-surface

FIGURE 10: Examples of correctly annotated images (row 1 and 2) and mis-annotated images (row 3).

TABLE 8: Average execution time (seconds) for the framework components (per camera image set for annotation blocks).

Component \ Mode	Training time (s)	Inference time (s)
Image transformation	19.79	
Feature extraction	502	0.8
Clustering (Louvain)	0.38	
Classification (EfficientNet)	6433	0.007

results are reported in Table 8 for each component of the framework.

VI. CONCLUSION & FUTURE WORK

In this paper, we have solved the benchmark problem of the annotation of a snow-covered road image dataset into four categories that represent snow depth. We have developed a generic system that automates the annotation of a benchmark dataset for snow-covered roads. The framework relies on image processing and unsupervised learning techniques. Images captured by each camera pass through a processing pipeline that includes feature extraction with a convolutional autoencoder, transformation into weighted graphs, and then clustering into similar groups using the Louvain community detection algorithm.

We have compared our clustering system to another one using a density-based algorithm and have achieved higher scores for graph-related metrics such as modularity and coverage. We have validated the application of the generated dataset by using it to classify snow-covered roads in a supervised manner. We have designed customized CNN models to fit the specificity of the dataset. We have trained the developed models while optimizing a class-weighted loss function to consider the imbalanced class distribution. We split the task into day and night times to increase the classification performance and trained separate models. We have achieved precision and recall scores of 97% on both day and night settings using an EfficientNet variant.

In future work, we plan to extract more complex cases, such as two-way roads with different states, and separate the medium and high snow level classes. We also plan to integrate the proposed classification model to solve real-world applications such as snow removal and winter public transport trip planning.

REFERENCES

[1] M. Karaa, H. Ghazzai, L. Sboui, H. Besbes, and Y. Massoud, "Unsupervised Image Dataset Annotation Framework for Snow Covered

Road Networks," in *2022 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, Nov. 2022, pp. 110–114.

[2] M. Aldibaja, R. Yanase, A. Kuramoto, T. H. Kim, K. Yoneda, and N. Suganuma, "Improving Lateral Autonomous Driving in Snow-Wet Environments Based on Road-Mark Reconstruction Using Principal Component Analysis," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 4, pp. 116–130, 2021.

[3] M. Hassaballah, M. A. Kenk, K. Muhammad, and S. Minaee, "Vehicle Detection and Tracking in Adverse Weather Using a Deep Learning Framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4230–4242, Jul. 2021.

[4] J. Fior and L. Cagliero, "Estimating the incidence of adverse weather effects on road traffic safety using time series embeddings," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, Jul. 2021, pp. 402–407.

[5] B. Hallmark and J. Dong, "Examining the Effects of Winter Road Maintenance Operations on Traffic Safety through Visual Analytics," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Sep. 2020, pp. 1–6.

[6] M. N. Khan and M. M. Ahmed, "Weather and surface condition detection based on road-side webcams: Application of pre-trained Convolutional Neural Network," *International Journal of Transportation Science and Technology*, vol. 11, no. 3, pp. 468–483, Sep. 2022.

[7] J. Carrillo, M. Crowley, G. Pan, and L. Fu, "Design of Efficient Deep Learning models for Determining Road Surface Condition from Roadside Camera Images and Weather Data," in *2019 TAC-ITS Canada Joint Conference*, 2019.

[8] G. Pan, L. Fu, R. Yu, and M. Muresan, "Evaluation of Alternative Pre-trained Convolutional Neural Networks for Winter Road Surface Condition Monitoring," in *2019 5th International Conference on Transportation Information and Safety (ICTIS)*. IEEE, Jul. 2019, pp. 614–620.

[9] Q. Xie and T. J. Kwon, "Development of a Highly Transferable Urban Winter Road Surface Classification Model: A Deep Learning Approach," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2676, no. 10, pp. 445–459, Oct. 2022. [Online]. Available: http://journals.sagepub.com/doi/10.1177/03611981221090235

[10] M. M. Adnan, M. S. M. Rahim, A. R. Khan, T. Saba, S. M. Fati, and S. A. Bahaj, "An Improved Automatic Image Annotation Approach Using Convolutional Neural Network-Slantlet Transform," *IEEE Access*, vol. 10, pp. 7520–7532, 2022.

[11] F. Lotfi, M. Jamzad, and H. Beigy, "Automatic Image Annotation Using Quantization Reweighting Function and Graph Neural Networks," in *Service-Oriented Computing – ICSOC 2021 Workshops*. Springer International Publishing, 2022, vol. 13236, pp. 46–60, series Title: Lecture Notes in Computer Science.

[12] S.-M. Schröder, R. Kiko, and R. Koch, "MorphoCluster: Efficient Annotation of Plankton Images by Clustering," *Sensors*, vol. 20, no. 11, p. 3060, May 2020.

[13] J. Yang, D. Parikh, and D. Batra, "Joint Unsupervised Learning of Deep Representations and Image Clusters," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016, pp. 5147–5156.

[14] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep Clustering for Unsupervised Learning of Visual Features," in *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, vol. 11218, pp. 139–156, series Title: Lecture Notes in Computer Science.

[15] S. Lu and R. Li, "DAC–Deep Autoencoder-Based Clustering: A General Deep Learning Framework of Representation Learning," in *Intelligent Systems and Applications*. Springer International Publishing, 2022, vol. 294, pp. 205–216, series Title: Lecture Notes in Networks and Systems.

[16] V. Prasad, D. Das, and B. Bhowmick, "Variational Clustering: Leveraging Variational Autoencoders for Image Clustering," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2020, pp. 1–10.

[17] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with Deep Learning: Taxonomy and New Methods," 2018, publisher: arXiv Version Number: 2.

[18] M. Karaa, H. Ghazzai, and L. Sboui, "A benchmark dataset for snow-covered roads in urban scenes," *IEEE Open Journal on Systems Engineering*, vol. this volume, 2023.

[19] D. Jin, Z. Yu, P. Jiao, S. Pan, D. He, J. Wu, P. Yu, and W. Zhang, "A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2021.

[20] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, Oct. 2008.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2016, pp. 770–778.

[22] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114.

[23] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, Feb. 2004.

[24] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on Graph Clustering Algorithms," in *Algorithms - ESA 2003*. Springer Berlin Heidelberg, 2003, vol. 2832, pp. 568–579, series Title: Lecture Notes in Computer Science.

[25] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," in *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2013, vol. 7819, pp. 160–172, series Title: Lecture Notes in Computer Science.

[26] L. McInnes and J. Healy, "Accelerated Hierarchical Density Based Clustering," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, Nov. 2017, pp. 33–42.