AutoSnow: A framework for generating synthetic winter images to test object detection systems used in autonomous vehicles

Amardeep Sarang
assarang@uwaterloo.ca

Abstract

Autonomous vehicles require very large amounts of testing data to be deemed safe. However, most popular autonomous vehicle data sets are recorded largely under good weather conditions and testing data for winter weather conditions is underrepresented. To address this problem, we present a framework called AutoSnow for generating synthetic winter image data from clear weather datasets by using simple python tools like NumPy and OpenCV. The AutoSnow project is open source and it can be accessed at this Github repo[1]. AutoSnow can efficiently transform images to generate 8 unique winter weather conditions. We use AutoSnow to generate synthetic winter datasets and use these datasets to test object detection systems, similar to those used in autonomous driving, that use models trained on clear weather data. Our experiments showed that the accuracy of these models is reduced dramatically when tested on synthetic winter data, which is similar to the results seen when clear weather models are tested on real winter data. Lastly, AutoSnow is also used to generate synthetic winter testing data to retrain an object detection model in an attempt to make it more robust to real winter data.

1 Introduction

Autonomous vehicles (AV) have been a lofty goal of the computer science and A.I. community for quite a while now, but this dream is still far from being a reality. Along with the challenge of developing these systems, the task of testing them and certifying them for public use is also immense. These systems need to be tested and operate safely under millions of different condition combinations. Some estimates put the amount of testing required at the equivalent of 10^8 to 10^9 km of real-world testing. Despite the need for such testing, there are common conditions faced by millions that are underrepresented in real-world testing and testing datasets [20]. One such sub-set of conditions is those encountered in winter driving. Many popularly used autonomous vehicle data sets are recorded largely under good weather conditions and even those which specifically try to cover all possible environmental influences, only contain very few images showing snowy roads and none showing falling snow [20]. This lack of data is likely because collecting data in extreme winter conditions is inherently risky to testers and is thus avoided. However, despite these challenges, access to these datasets will be required for us to test the safe operation of AVs in these conditions.

To address these shortcomings, in this paper we explored techniques to test object detection models used in autonomous vehicles on synthetically generated winter weather data. Where we focused on using automatic techniques like image transformations and overlaying image filters to create a dataset mimicking winter conditions from clear weather datasets. We will specifically look to create images that reflect common winter conditions like light fluffy snow, fast streaking snow like one may see when driving at high speeds, near white-out blizzard conditions, and ice/frost build-up on the camera.

1.1 Contribution 1: Synthetic winter dataset generation

To do so as part of this paper we present a framework called AutoSnow for generating synthetic winter data from clear weather datasets by using simple python tools like NumPy and OpenCV as our first major contribution. AutoSnow can transform images to generate 8 unique winter weather conditions efficiently. Where each transformation algorithum aims to represent a condition that provides a unique challenge to machine learning models. The project is open source and can be accessed at this Github repo.

1.2 Object detection

Object detection is the process of identifying the class of an object and identifying its position in an image. We investigate object detection models specifically since these are the components of an autonomous driving system that are most likely to fail since they need to detect objects that are normally only a small part of the image and would thus be more sensitive to noise caused by winter conditions. There are also not many datasets focusing on object detection for wintertime implementation thus there has not been much work to identify how object detection models may fail in these conditions [21]. Having the ability to identify and make object detection robust to failures is critical as failures of object detection in autonomous vehicles have been linked to fatal crashes, like in 2016 when a Tesla's camera failed to recognize the white truck against a bright sky causing its autopilot system to crash into it, killing the driver [3].

[1] https://github.com/AmardeepSarang/AutoSnow-A-synthetic-winter-image-generator-framework

1.3 Contribution 2: Testing an object detection model with synthetic winter data

To address this need, the second contribution of this paper is to investigate the effect of winter data on object detection models used in autonomous vehicles using the synthetically generated winter data created using AutoSnow. This allows the testing of these models by automatically creating testing data that can be used to safely test these autonomous driving systems without the need for risky data collection in dangerous winter conditions since it can simply be generated using clear weather data.

We test how the accuracy of an object detection model previously trained on a clear weather dataset would be affected by a dataset generated using each synthetic weather condition. This model is then also tested with a real winter object detection data set to see how the accuracy is affected and how closely this matches the accuracy when tested with synthetic winter data. Our evaluation shows that the model when trained on only clear weather data performs similarly poorly on both the real and synthetic winter weather data. Having multiple different synthetic conditions also allows us to discover that the model accuracy is affected uniquely across different winter weather conditions, object sizes, and object classes.

1.4 Contribution 3: Testing object detection model with synthetic winter data

It may not be enough to simply test a model with new data to verify its inadequacy but one may also then want to retrain the model to be more robust to such data in the future. Training autonomous vehicle object detection models in winter conditions require large amounts of data similar to that used for testing, and due to many of the same reasons this data is only sparsely available. Therefore, for our final contribution, we study if the generated winter training datasets made using AutoSnow can be used to better train an object detection model to handle real adverse weather conditions. This experiment is conducted by using synthetic winter data to train a model and then compare its performance on real winter data with a model only trained with clear weather data, to see if the performance can be improved.

1.5 Research questions

To guide the contributions of this paper we shall start by asking the following research questions:

> **RQ1:** *Can we create synthetic images that closely reflect images seen by an autonomous vehicle's cameras during incremental winter weather? What methods can be used to transform clear weather images into those reflecting winter conditions? Can these transformations be applied efficiently to large datasets?*
> **RQ2:** *What are the effects on the accuracy of an object detection model, trained with clear weather data, when the model is tested on synthetically generated winter data? Do the test results (accuracy) using the synthetic winter data reflect the model's test results on real winter data?*
> **RQ3:** *Does training the model on synthetically generated winter data help improve its performance on real winter data, when compared to the model trained with clear weather data?*

In the next section, we will highlight other works that have conducted similar studies. Following this, we outline the method used for our own experiments in section 3. This will include details of the datasets and object detection framework used. We then provide a detailed description of the synthetic winter condition transformations in AutoSnow and how they are created. The section also includes the experimental procedure used to answer the 3 research questions. Section 4 presents the results of these experiments and uses them to answer the research questions. Finally, we will discuss some of the limitations of the experiments in section 5 and conclude the paper in section 6.

2 Related works

One of the main contributions of our paper is to transform images to simulate a condition not original to that image and see the effect it has on a machine learning model. These experiments have been done in other works where they experiment with generating images with differing brightness, changing contrast, translation, and fog effect to test a steering prediction model [18]. However, this particular work does not include snow in its experiment. Some of the closest works to ours which do include a snow effect have also benchmarked existing object detection models like Faster R-CNN with snowy images to explore how performance is affected [5, 12], but these do so with only a single snow overlay which is added onto all their images. While other works have attempted to improve winter weather performance by developing a model specifically tailored to snowy conditions with a real winter data set [21]. These techniques lack realism due to there being no randomness in where the snowflakes appear as they are in the same location across all images. Also, the filter they use only represents a single type of snowfall, and although the authors of these works showed that this particular snowfall condition reduced their model's accuracy or can be used to train a new model, this cannot provide an accurate representation of how other types of snowfall would affect these models. For this, more realistic and multiple types of snowfalls must be tested.

Some works have also tackled the problem of creating highly realistic test snowy data in OpenGL [20] or in specialized simulations that can make highly realistic ice build-up on surfaces [8]. These works do address some problems by making the snow effects more realistic but they also introduce their own set of problems. [20] only provides images

from a single simulated city which means it may not generalize to other regions, and [8] method is intended to create a single highly realistic image but could not be efficiently scaled to creating an entire test dataset where thousands of images must be generated.

Our project offers an approach that addresses problems from both sides of this spectrum. AutoSnow can have a greater variety of multiple different types of snow conditions and have random variation within each condition so each image has a unique placement of snowflakes which trains the model for greater generalizability on more realistic looking conditions. It also allows the snow effects to be efficiently applied to any existing image dataset, making it easy to test with different regional datasets. Also, to the best of our knowledge, we will be the first to compare the test results of synthetic winter image data with that of real winter data, and whether training a model on synthetic winter image data will make it more robust when facing real winter data. This will be an important contribution as it will indicate how well synthetic winter data mimics real data in model training and testing.

3 Method

This section will discuss the details of the method we used to conduct our experiment. We will start by discussing the datasets and models we used followed by a discussion outlining the procedure we followed to address our research questions. This will include details of how we implemented our AutoSnow framework to create 8 unique winter conditions as well as the reasoning for including these conditions.

3.1 Data

For the study, we used two datasets. The first was a clear weather dataset which we used to train the baseline clear model. A subset of this dataset would also be used as the clear weather test dataset, as well as forming the dataset used for transforming into winter data. The popular KITTI 2D object detection dataset was used for this purpose [4]. It consisted of 7481 training images and 7518 test images, comprising a total of 80256 labeled objects [4]. However, we could only make use of the training images since test data labels were not publicly available and we needed them to perform our own validation, separate from the KITTI benchmark. To do so we split the original 7481 training images into 5237 images for training and, 2244 images for testing as well as transforming into winter data. Additional modifications to the labels were made using a publicly available script [2] that was used to reformat them to work with the YOLOv5 framework format of $\left(classLabel, \frac{X_{center}}{image\ width}, \frac{Y_{center}}{image\ height}, \frac{box\ width}{image\ width}, \frac{height}{image\ height}\right)$ [7, 13]. This script was also modified to ensure that only objects labeled as “Pedestrian”, “Cyclist”, “Car”, and “Truck” classes were kept in the dataset since these were the classes that were also common to our winter dataset.

Our second dataset was the Canadian Adverse Driving Conditions Dataset (CADCD) which is a winter weather driving dataset made by the University of Waterloo [16]. This dataset was used as our real winter dataset and would be used to test our clear weather and synthetic winter models on real winter conditions. From this model, we used the 1592 images which were documented to have been collected during medium to extreme falling snow. These images had object annotations that were in a 3D format, we used a script from past work [6] to convert them into 2D labels and then converted them to the YOLOv5 framework label format discussed above.

3.2 YOLOv5 object detection model

To create and train our models we used the YOLOv5 object detection framework since it was straightforward to use [7] while allowing for easy training and utilization of models trained with different data. This is because instead of training a model from scratch YOLOV5 would start from the general-purpose YOLOv5s model trained on the COCO dataset [10]. Then we further finetune this model by training it with our dataset. This method delivers faster training while still producing reasonably accurate results.

3.3 Training clear weather model and testing to get baselines

Our first step was to train our clear weather model on the 5237 images from our KITTI clear weather dataset. The model was trained for 22.5 hours, completing 15 epochs. After training, we tested this model on the clear weather test dataset. To do so we ran a script that would generate predictions of the object labels and their bounding boxes (positions in the images) for each of the 2244 clear weather test images. We then input the prediction label and ground truth labels to an Object Detection Metrics Toolkit [13] which then output that the model achieved an accuracy of AP = 0.469 and mAP = 0.787 on our clear weather test data. A more detailed discussion of the metrics used will be given in section 4. We then performed a similar test to the clear weather baseline by having the model generate predictions for the 1592 winter images. The accuracy for this test was AP = 0.02 and mAP = 0.04.

3.4 Creating a synthetic winter dataset

The next step of the project was to create the AutoSnow framework itself. This framework uses transformations created using python to create images that approximate winter conditions by applying the transformation algorithms to clear weather images. Using efficient python libraries like OpenCV and Numpy allowed us to apply these transformations to each image in a manner allowing for sufficient randomization while allowing the method to be efficient enough to be used to transform large datasets with many thousands of images.

For this work we designed the framework with transformation algorithms that could create 8 unique winter conditions: clear ice buildup on the camera, heavy ice buildup on the camera, light ice buildup on the camera, white-out falling snow, fine falling snow of heavy intensity, fine falling snow of medium intensity, fluffy falling snow, and streaking snow. An example of each of the listed synthetically generated conditions is shown in figure 1. We selected these conditions based on seeing them in the winter dataset and based on conditions that have been seen in real-life experience, where each condition provides a unique set of challenges that the model is tested for. Each transformation algorithm simulates one or more of the conditions. We will now discuss each of the algorithms in detail showing the steps to create each of the winter weather conditions.

3.4.1 Clear ice buildup. The first of these algorithms transforms an image to simulate clear ice build-up on a camera or windshield as one may see after a freezing rain/ice storm. An example of this transformation is shown in figure 1b. This condition provides a unique challenge to the object detection model because in the image rough object shapes can be seen through clear ice but they appear distorted as light is randomly refracted and reflected causing light to take a slightly different path. This creates a distortion effect similar to frosted glass which the model must overcome to detect the object.

The transformation algorithm for this condition creates a new image by replacing each pixel from the original image with another pixel within a N pixel radius to create distortion and random ice refraction effect. To change the level of distortion the hyperparameter N can be increased or decreased, creating a more distorted image with a higher value. This randomization of the pixels simulates the path taken by light through the ice and thus creates a realistic effect.

3.4.2 Heavy and light ice buildup. The next algorithm aims to simulate ice buildup on the car's cameras like one may see after a wet snowstorm mixed with ice. In this condition, chunks of ice and snow accumulate to block parts of the camera. This condition and the condition above are unlikely given most autonomous vehicles have heating elements to clear this type of ice buildup from cameras and sensors. Despite this, we include these conditions to test if a model could handle these conditions should those clearing mechanisms fail. In such a case this condition would be challenging for an object detection model since large continuous parts of the camera would be blocked, meaning the accuracy would be affected by views of objects being partly or fully covered up. Such conditions would also produce blurry images due to the camera being out of focus because of ice/snow chunks on the camera.

Simulating such ice formation conditions on glass in a very realistic way would be a highly complex task that would require a deep understanding of how the crystals behave on a molecular level as noted in a thesis work on the topic [8]. Our goal was not to fully simulate snow/ice formation but rather mimic the look of it on a camera. To achieve this, we required a way of generating random dark and light regions that look similar to ice buildup. To do so we used Perlin noise which is a technique for creating varying intensity noise in 2D natural-looking clusters [14, 15]. The transformation algorithm for this condition first used a library to generate Perlin noise in the shape of the image. Then it created a new ice layer image which is colored based on the intensity of the Perlin noise by setting a threshold dictating which intensity gets the darker ice colors. Next, the ice layer is merged with the original image where areas below the lower threshold will keep the original image pixels, while other areas will be colored according to the ice layer. Finally, the resulting image is blurred to add the camera out-of-focus effect. We can manipulate parameters of transformation, like coloring thresholds to make ice build-up heavier or lighter which can be seen in the sample images for this effect in figure 1c and 1d.

3.4.3 Fluffy snow and streaking snow generation with random shape. AutoSnow also has an algorithm to simulate conditions with large sparse falling snowflakes. One of these conditions which we refer to as "fluffy" falling snow occurs when wet snow clumps together into large snowflakes. This results in the larger "fluffy" snowflakes that are farther apart when they fall. This algorithm will also generate a second condition of streaking snow which occurs when falling snow appears as long thin lines moving across an image due to the snow moving very fast relative to the camera. These conditions will challenge the model by producing large areas of white noise (snowflakes) that are farther apart. This means that many objects in the images are largely visible with only a few large snowflakes in front of the object, but this could still be tricky if a large snowflake covers up a small distant object. This may cause the model to be able to detect most objects accurately but fail on a few objects in a few images

The algorithm to generate these effects was inspired by the Automold image augmentations library [17]. It did not have any method to generate falling snow but it did have a method for generating rain which used randomly generated lines to mimic streaks of falling rain which we used to generate streaking snow by creating the random white lines. We used a similar method for fluffy snow by substituting the thin lines for randomly generated ovals and rectangles. Our algorithm differed from Automold in the technique used to blend the snowflake shapes into the image. Automold simply used a blur filter from OpenCV to do so but this cause the drops to become semi-transparent, this is desirable for rain but we needed our snowflakes to maintain their white opaque color. To achieve this effect, we first generated our snow streaks on a black image and blurred them to create fuzzy edges. Then we created a method to overlay the snow into the image

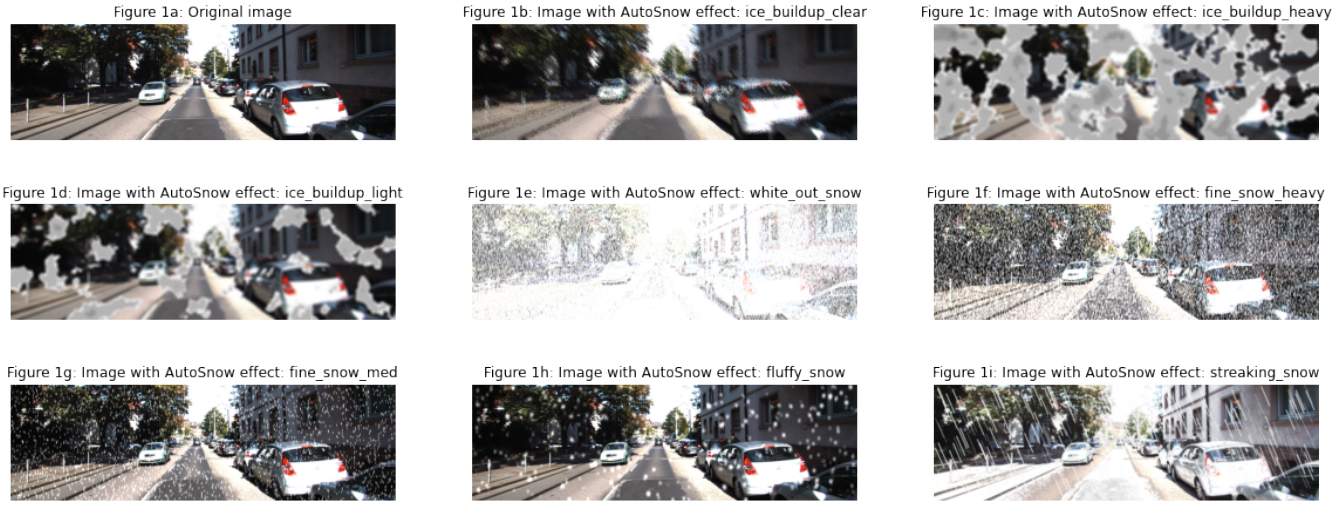

Figure 1. Shows an example image being transformed using the AutoSnow algorithm into images reflecting 8 winter weather conditions (top to bottom, left to right): clear ice buildup on the camera, heavy ice buildup on the camera, light ice buildup on the camera, white-out falling snow, fine falling snow heavy intensity, fine falling snow of medium intensity, fluffy falling snow, and streaking snow.

which preserved the white pixels of the snowflake layer while blurring the flake edges based on the color intensity. Lastly for the streaking snow condition we used Automold's method for generating snow on the ground. This is a fairly simple method that makes all pixels below a certain threshold bright white. It does not give much control over where the snow accumulates but it gives a satisfactory effect of there being some ground snow in addition to falling snow. Using these methods, we created the fluffy and streaking snow conditions which can be seen in figure 1h and 1i.

3.4.4 Fine snow generation. The final algorithm in our AutoSnow method is one to create fine powder-like dry falling snow conditions at varying intensities. This type of snow shows up in images as many small particles which are normally more numerous and closer together than the snowflakes in wet snow. Although the individual snow particles are smaller, they are much more numerous thus many snow particles may eclipse many more pixels of an individual object and add much more noise that the model must handle.

The technique for this algorithm was originally from a photoshop tutorial on YouTube [1] to create fine falling snow particles on images. The method consisted of first generating a layer of noise (random white or black pixels). Then the noise layer was scaled up and then we cropped to the original image size so that each individual noise pixel would appear bigger and thus represent a larger snowflake. Finally, the method used motion blurring to blur the pixels in the noise layer to make the fine snow particles appear fuzzy before blending the fine snow layer onto the original image. We closely followed this Photoshop method but the main challenge to following this method was translating the techniques from photoshop to python and OpenCV, for example, the level adjustment feature from photoshop was not in OpenCV so we had to divert from the tutorial to find a different way of controlling the density of the snow using noise probability. We controlled the intensity of noise (and thus snow intensity) by assigning a probability distribution to white and black pixels ie. $P(white) = threshold$ and $P(black) = 1 - threshold$ where a higher probability of white pixels resulted in more noise/snowflakes. Python also can't create image layers so we instead created a new image to generate the noise and then blended it using the same method as we did for fluffy snow. Using this method, we were able to create a very convincing effect and we used it to generate white-out snow, fine snow of heavy intensity, and fine snow of medium intensity by varying the size of the snowflakes by scaling the noise, or by adjusting the noise intensity to make the snowfall denser. For the white-out snow we also used Automold's method for generating snow on the ground. Examples of these generated conditions can be seen in figure 1e, 1f, and 1g.

3.5 Creating synthetic winter data and testing using AutoSnow methods

Using the above methods, we created datasets for the 8 unique winter conditions: clear ice buildup, heavy ice buildup, light ice buildup, white-out snow, fine snow heavy, fine snow medium, fluffy snow, and streaking snow. These datasets were generated by running the AutoSnow algorithms for creating each condition on the 2244 images in our KITTI

testing set. We also created a mixed set that randomly selected a weather condition to apply to each image. We then tested each of the synthetically generated datasets with the clear weather model by running the prediction script to generate prediction labels for each dataset. We used the same ground truth labels from the clear weather test set since these were the images that were transformed to create the winter datasets and thus the objects would remain in the same position. Then we evaluate the accuracy of the model using predicted labels, ground truth labels, and the Object Detection Metrics Toolkit [13] similarly to how we tested the baseline datasets. We then analyzed how the dataset with each condition effect applied affected the accuracy of the model which will be presented in section 4.

3.5.1 Synthetic winter data model. The final part of our experiment methodology is to test if we can use AutoSnow to generate a test dataset of synthetic winter data and use it to retrain the model by training on this dataset. We first created the training dataset by randomly selecting a weather condition transformation to apply to each of the 5237 images in the KITTI test dataset. This created training data featuring a mix of all conditions. It should be noted that this was done instead of creating a dataset where there would be 8 copies of each image with each weather condition applied as this would have created a dataset that was far too large to train in a reasonable amount of time for this project. We did add clear weather (untransformed) copies of the 5237 images to the new dataset along with labels to ensure that it could still identify clear weather objects. A similar procedure was followed for creating a validation set by transforming the KITTI testing dataset. This resulted in a dataset of 10474 training images and 4488 testing images.

Using this dataset, we then trained our model similarly to how we trained the baseline model by finetuning training the YOLOv5s starting model. We trained the model on the dataset for as long as our hardware would allow until the accuracy on the valuation dataset was similar to the accuracy we reached for the valuation of the clear weather model. This training took longer since there was more data to train on so we ran the training for 46 hours and completed 18 epochs. At which point we got an accuracy of AP = 0.430 and mAP = 0.731 on the valuation dataset. We then tested the model with the clear weather and real winter weather test sets with the same procedure as our other tests, to see if the model created with the synthetic data would have improved performance on real winter data while still being capable of detection in clear weather data. We present these and other results from our experimental method in the next section, as well as discussing how they can be used to answer the questions asked at the start of the paper.

4 Evaluation

In this section we evaluate the accuracy measurements for each of the two models created when they were tested we the clear, real winter, and synthetic winter datasets. By doing so we will answer the research questions.

4.1 Object evaluation metrics

Before we can discuss the results of testing the models, we will briefly explain the metrics used to measure the accuracy of the models. For evaluating the accuracy, we used the Object Detection Metrics Toolkit developed by Padilla et al. [13] this tool kit could generate many accuracy metrics used in object detection benchmarks like COCO [10]. Each of these metrics differs slightly in how they are calculated and by comparing multiple different metrics we were able to gain deeper insight into the effects of the different weather conditions. Therefore we provide an overview of the metrics used while omitting the technicalities of implementing the metrics for which we refer the reader to the paper accompanying the toolkit [13].

4.1.1 Intersection over union (IOU) and object bounding boxes. In object detection, we have labels and bounding boxes around each object. The ground truth bounding boxes b_{gt} represents a true bounding box for an object as defined in the testing data labels. Then when the model is used to generate predictions, a prediction bounding box b_p is generated for each object the model can identify based on where the model believes the object is located. For a pair of ground truth and prediction labels, the way that we determine if they are similar is to calculate the intersection over union (IOU). This is a measurement of how much the prediction box overlaps the ground truth box and is calculated as $IOU = \frac{area\ of\ overlap}{area\ of\ union} = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}}$. Then based on an IOU threshold (percentage of area needing to overlap) set at the start of a calculation a prediction is classified as either true positive (TP): A correct detection of a ground-truth bounding box by having an IOU above the threshold, a false positive (FP): An incorrect detection of a non-existing object or a misplaced prediction box of an existing object by having an IOU below the threshold, or false negative (FN): An undetected ground-truth bounding box with no matching prediction [13].

4.1.2 Precision and recall. Using the TP, FP, and FN the precision and recall are calculated. Precision is the ability of the model to only identify relevant or ground truth objects and is calculated using the equation $Pr = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{S} FP_n} = \frac{\sum_{n=1}^{S} TP_n}{all\ detections}$, where S is the number of correct detections there would be. Meaning the precision will decrease if the model detects objects not present in ground truth.

Recall is the ability of the model to identify all relevant or ground truth objects and is calculated using the equation $Rc = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{S} FN_n} = \frac{\sum_{n=1}^{S} TP_n}{all\ ground\ truths}$, where S is the number of correct detections there would be. Meaning the recall will decrease if the model did not detect all objects present in ground truth.

4.1.3 Average Precision. Precision and recall can then be plotted as a precision-recall curve. We can then calculate the area under this curve and use it in our first accuracy metric: average precision (AP). We do so by taking the Riemann integral of Pr. This gives us an overall metric that incorporates precision and recall which can be used to compare across tests. In its standard form, all the object classes are included in the same AP calculation which gets repeated for IOU thresholds from 0.5 up to 0.95, increasing in 0.05 increments. Then all these AP calculations at each IOU threshold are averaged to get an overall AP metric.

4.1.4 Mean average precision. The second metric we considered is mean average precision (mAP). This metric differs from AP since instead of calculating the precision, recall, and average precision at the same time it calculates them separately at an IOU threshold of 0.5 for each class and then averages them. This makes this metric less strict than normal AP. It is calculated using the equation $mAP = \frac{1}{C} \sum_{i=1}^{C} AP_i$, where C it's the number of classes.

4.1.5 Other AP metrics. The toolkit also allows us to calculate variations of AP. AP@50 and AP@75 are versions of the AP calculation that only get calculated at IOU thresholds of 0.50 and 0.75 respectively. These are useful for seeing how AP may drop off when being more strict with the overlap needed. We also can calculate AP while only considering certain sized objects in the image, APsmall: only considers ground truth objects with a pixel area $< 32^2$ pixels in the image, APmedium: only considers ground truth objects with a pixel $32^2 <$ area $< 96^2$ pixels in the image, and APlarge: only considers ground truth objects with a pixel area $> 96^2$ pixels in the image. These metrics were used to give insights into how the effect on accuracy from different weather conditions might differ for smaller and farther away objects vs larger or closer ones. Lastly, we also calculated accuracy for each class individually which showed if any of the classes were affected differently in each weather condition.

4.2 RQ1 results and discussion

From the accuracy metrics collected from our experiments, we will now provide answers to our research questions. The first and most basic question we asked was weather it was possible to create a method that could transform clear weather images into images reflecting winter weather conditions. We showed that this can be done with the algorithms in the AutoSnow framework described above. However, we cannot make any provable claims of how similar these synthetic images are to real winter conditions as this is a subjective question and although attempts were made to make the synthetic images look realistic, to verify this empirically would require a survey of human participants to judge the realism of the synthetic winter images, which is beyond the scope of the study. We can instead use the accuracy of the clear weather model on the real winter and synthetic winter datasets as a proxy, where we can consider the synthetic data similar in the eyes of a model if it achieves a similar accuracy for both datasets. Using this measure we see in figure 2 that some of the synthetic datasets like the clear ice buildup, light ice buildup and heavy fine snow are similar to the real winter data as they all have very poor accuracy in terms of the mAP metric, and similarly so for AP. The real winter data getting a poor accuracy of AP 0.02 and an mAP of 0.04 is also similar to all the synthetic conditions overall as they had an average accuracy of AP 0.11 and an average mAP of 0.19.

The last part of this question was whether the data could be generated efficiently. This was an important guiding metric of the framework as a system where it took several minutes to transform a single clear weather image into one reflecting winter conditions would not have been practical for use on datasets with thousands of images. To avoid this bottleneck we built AutoSnow in an efficient manner. This resulted in almost all 8 transformations being efficient due to being created using vectorized methods. These vectorized transformations could transform 16.3 images per second or 1000 images per minute when tested on a laptop with an Intel Core i7-9750H CPU @ 2.60GHz and 16 GB of ram, meaning they would only take a few hours to transform even a large object detection dataset like COCO with over 330k images [10]. Only the heavy and light ice buildup transformations were not able to be streamlined due to making use of the Perlin noise library which was not vectorizable. Due to this these transformations take significantly longer at 22 minutes per 1k images making them only suitable for smaller datasets like ours.

4.3 RQ2 results and discussion

To answer our second research question we tested the clear weather model with each of the synthetic winter conditions. We then compared these results with the clear weather baseline test to see how the model's accuracy was affected by the synthetic data. For the clear weather baseline, where the clear weather data was tested on the clear weather model, we got an accuracy of 0.468 for the AP metric, and 0.78 for the mAP metric. When we then tested on the synthetic winter condition where despite using the same images just with the winter transformations applied we saw an overall decrease in accuracy. The AP values ranged from 0.004 to 0.42, depending on the effect used with an average AP of 0.11, and mAP values ranged from 0.002 to 0.72, with an average mAP

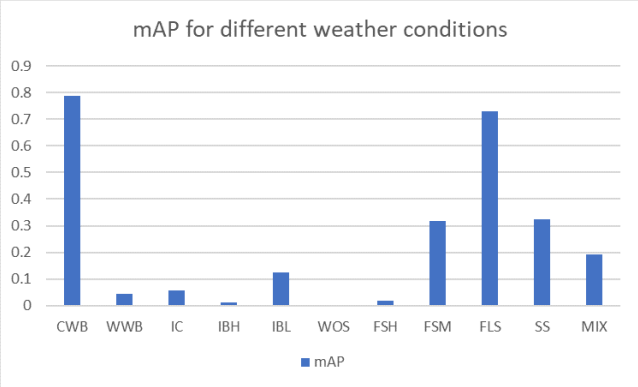

Figure 2. The mAP of the clear weather model when tested on Clear weather baseline (CWB), Winter weather baseline (WWB), AutoSnow ice buildup clear (IC), AutoSnow ice buildup heavy (IBH), AutoSnow ice buildup light (IBL), AutoSnow white out snow (WOS), AutoSnow fine snow heavy (FSH), AutoSnow fine snow med (FSM), AutoSnow fluffy snow (FLS), AutoSnow streaking snow (SS), and AutoSnow mixed (MIX) datasets.

of 0.19. We also were able to see that this drop in accuracy, when tested on synthetic data, was similarly poor to when we tested the clear weather model on the real winter weather baseline data, where an accuracy of only 0.02 for the AP metric, and 0.04 for the mAP was recorded. This result shows that at least for some of the synthetic weather conditions tested the accuracy on the clear weather model was similar to when it was tested with real winter data.

4.3.1 Comparing synthetic weather conditions' effects on a clear weather model. Despite the overall lowered accuracy when testing on synthetic conditions, we see in figure 2 that not all conditions affect the accuracy equally. The model accuracy was less affected by the condition that introduces fluffy snow, still having a mAP of 0.72, similarly for streaking snow with a mAP of 0.32, and medium intensity fine snow with a mAP of 0.31. In general, the model was better at handling conditions that produced larger but fewer snowflakes. Meanwhile, we see that conditions that created many smaller snowflakes or, blurred and distorted the entire image like in the ice build-up condition have much worse accuracy with whiteout snow being the condition with the worse accuracy of 0.002 mAP and the others being similarly poor. This indicates that this model is not suited to conditions that distort many pixels or introduce large amounts of noise in the form of snowflakes, with overall model performance worsening when more pixels are affected by the condition.

4.3.2 Comparing affects of object size and weather on a clear weather model. Using the APsmall, APmedium, and APlarge metrics discussed above also allowed us to study how the synthetic conditions affected accuracy at a finer level and study if the overall accuracy decrease seen changed for different sized objects. The intuition behind this deeper study was that the model may be able to make out large closeup objects better even in winter conditions due to larger objects having more pixel data thus making them easier to make out despite the added noise. This intuition proved true as it was found that across all weather conditions (clear, synthetic, real winter) model performance decreased as the objects took up smaller positions of the image as seen in figure 3 but in conditions like clear and light ice buildup, the model was still able to detect larger objects with higher accuracy. When applying the clear ice effect a reduction of 98% was seen compared with the clear baseline in the AP metric when only being applied to small objects (APsmall), but only a reduction of 72% was seen for large objects (APlarge). This shows that in conditions like clear and light ice buildup the model's ability to detect the larger objects did not degrade as much in these conditions as it did for smaller objects.

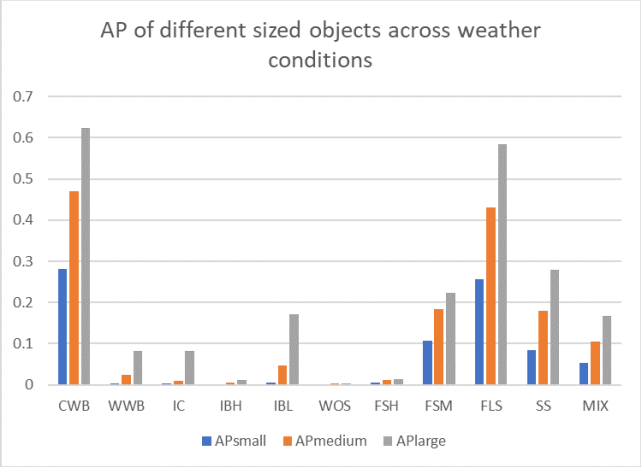

Figure 3. The AP of the clear weather model for different object sizes when tested on Clear weather baseline (CWB), Winter weather baseline (WWB), AutoSnow ice buildup clear (IC), AutoSnow ice buildup heavy (IBH), AutoSnow ice buildup light (IBL), AutoSnow white out snow (WOS), AutoSnow fine snow heavy (FSH), AutoSnow fine snow med (FSM), AutoSnow fluffy snow (FLS), AutoSnow streaking snow (SS), and AutoSnow mixed (MIX) datasets.

4.3.3 Effect of weather conditions on per class accuracy. We also conducted a similar analysis using the per class AP metric to see if the model's ability to detect each class of object is affected differently by different weather conditions. In the clear weather baseline, the model recognized trucks more accurately with an AP of 0.91, followed by cars, cyclists, and pedestrians with APs of 0.86, 0.71, and 0.65 respectively. Since the synthetic dataset used the same images as the baseline for their transformation, they should have the same sets of objects and thus the same accuracy

distribution. However, this pattern did not hold for all the synthetic data. In all synthetic winter conditions, except fluffy snow which closely matched the baseline, the accuracy of the cyclist and trucks were the lowest with pedestrians and cars having the highest. For example, the AP when light ice build-up was applied was 0.02, 0.31, 0.04, 0.11 for trucks, cars, cyclists, and pedestrians respectively. This indicated that certain weather conditions affected the model's ability to detect certain classes of objects differently and was not proportional to the original clear weather accuracy distribution.

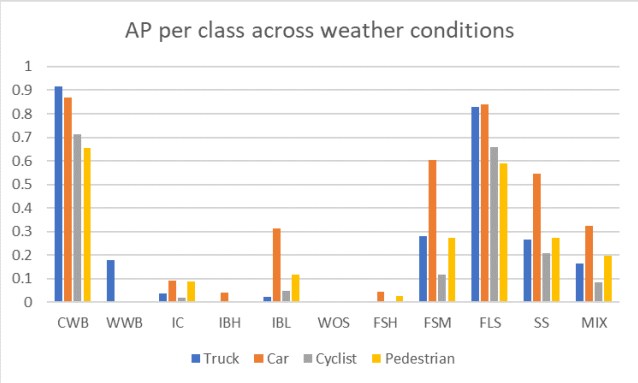

Figure 4. The AP of the clear weather model for different object classes when tested on Clear weather baseline (CWB), Winter weather baseline (WWB), AutoSnow ice buildup clear (IC), AutoSnow ice buildup heavy (IBH), AutoSnow ice buildup light (IBL), AutoSnow white out snow (WOS), AutoSnow fine snow heavy (FSH), AutoSnow fine snow med (FSM), AutoSnow fluffy snow (FLS), AutoSnow streaking snow (SS), and AutoSnow mixed (MIX) datasets.

4.4 RQ3 results and discussion

For the last research question, we retrained the model with a mix of different winter synthetic conditions and clear weather data as discussed in the method section to determine if this new model could provide better performance on a real winter dataset. We first tested the new model on the clear weather and synthetic winter data where we found that the new model performed similarly well on the clear weather test set achieving an AP of 0.48 and a mAP of 0.79 compared to an AP of 0.46 and a mAP of 0.78 when tested with the clear weather model. The retrained winter model also showed significant improvement on the mixed synthetic winter data improving from a mAP of 0.19 to 0.65. This was expected since the model was trained on clear and mixed synthetic winter data, but it still provided a "sanity check" and showed that the model can maintain clear weather performance while also being trained on synthetic winter data to improve its accuracy on it. When the winter model was tested on the real winter data, however, no significant improvement in the accuracy was observed the accuracy was a bit worse as can be seen in figure 5. When tested with the winter model AP was 0.01 and mAP was 0.03 compared to an AP of 0.02 and a mAP of 0.04 when tested with the clear weather model. There was also no improvement for any of the other accuracy metrics. This result suggests that even though synthetically generated winter data is useful for testing the model in a way similar to real winter data, at least for this particular YOLO model that we tested, using the synthetic winter data for training the model does not improve its performance on real winter data.

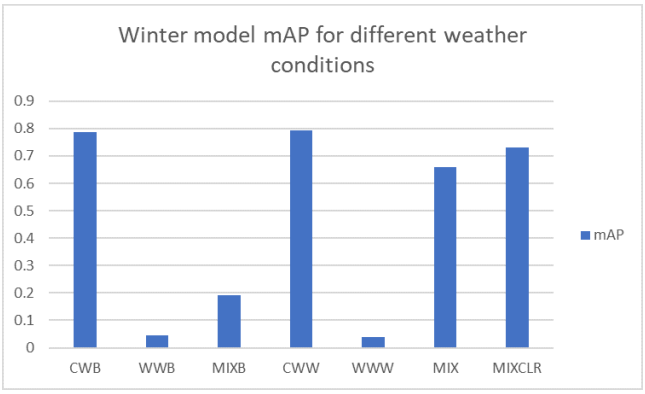

Figure 5. The mAP of the winter weather model when tested on, Clear weather baseline (CWW), Winter weather baseline (WWW), AutoSnow mixed (MIX), Auto snow and clear (MIXCLR) datasets. Along with the mAP of the clear weather model when tested with Clear weather baseline (CWB), Winter weather baseline (WWB), and AutoSnow mixed (MIXB) datasets.

5 Limitations and threats to validity

We now discuss reasons for the above negative result for RQ3 as well as other limitations of our method. The biggest limitation of our current method was that the YOLOv5s is a fairly small model which may have limited its ability to adapt well and transfer the learning from the synthetic dataset to the real winter dataset. We used this model knowing these limitations exists due to time and computational limitations but we did not anticipate it would be quite so limited. To improve on this limitation in future work it may be useful to use a more advanced network like Faster R-CNN [12].

Another reason for the model not being able to generalize well to the winter dataset could be the regional differences between the two datasets. The winter dataset (CADCD) is recorded in Waterloo, Canada, whereas the clear weather (KITTI) and synthetic winter weather dataset made from it was based on image data from Karlsruhe, Germany. This was not expected to have a large impact since the object detection classes were the same. Cars in KITTI were mostly small

hatchback cars whereas the north American CADCD had more SUVs. Also, we noticed that the images from the older (2014) KITTI dataset were often over-exposed and lower resolution compared to the more recent (2020) sharper images of the CADCD. These differences may have also been a factor in the model's reduced performance on the winter data meaning the reduction cannot be purely attributed to the winter weather. Future work should strive to more strictly control these variables by using clear and winter datasets that are more similar to each other in terms of region and collection technique.

Future work should also repeat this study on object detection tasks that continue across many video frames, instead of the single-frame object detection covered in the paper. Testing a video-based object detection model would be more representative of real autonomous systems since it could be better at handling objects being temporarily occluded by snowflakes or ice on the camera like in [11] by using signals from multiple video frames.

Lastly, we discuss the limitations of the synthetic winter condition generation techniques of AutoSnow itself. The techniques all apply layers of falling snow or ice on top of the original image and do not alter the image beyond simple blur or distortion effects and although this makes the method highly efficient it also has the disadvantage of not being able to change individual aspects of the image itself like the sky or trees. This leads to some unrealistic scenarios like having full green leaves on trees and clear blue skies while it is snowing heavily. In addition, we also do not detect where grass or static surfaces are in the image meaning we cannot add realistic accumulated snow along with the falling snow. Some snow effects do have snow on the ground using Automold but this effect was fairly limited in that it only worked by changing saturation levels with no real control of which area turned white.

A second limitation of only generating a flat overlay effect is that although the snow is realistic looking in the foreground, we are not able to create very convincing snow effects further in the background as we are not able to simulate the effect of the depth (distance from the camera) further in the background of the images.

To fix these limitations, in future iterations of AutoSnow we will address these issues by adding techniques that can alter certain parts of the image individually. One way to do this would be to use a CNN to identify different aspects of the image and then replace those aspects with a more winter-appropriate version. For example, we could use the methods presented in [19] to replace clear weather skies with one appropriately cloudy for falling snow. We could also use a similar method to identify grass and add ground snow to those regions more accurately.

The second limitation could also be addressed using depth mapping techniques [9] which would allow us to identify background regions and apply a more suitable falling snow effect to that region. With these updates, it is possible that we could create more realistic synthetic images and improve on some of the experimental results in this paper.

6 Conclusion

In this paper, we have presented a framework called AutoSnow that uses simple python tools to transform clear weather images into images that simulate challenging real winter weather conditions. We used AutoSnow to create synthetic winter datasets which reflected 8 winter conditions and used these datasets to test object detection systems, similar to those used in autonomous driving, that used models trained on clear weather data. Our experiments showed that the accuracy of these models reduced dramatically when tested on synthetic winter data which was similar to the results seen when tested on real winter data. We also showed that the model accuracy was affected differently by different winter weather conditions, object sizes, and classes of objects. The model was not able to handle conditions that created many smaller snowflakes or, blurred and distorted the entire image like in the ice build-up condition, and also struggled to detect smaller, farther away objects in these conditions. We also attempt to make the object detection model more robust on real winter data by training on synthetic winter data.

We hope the work done in creating the AutoSnow framework streamlines the testing of object detection models used in autonomous vehicles by making it easier to create winter data and showing the need to further test and improve these models in extreme winter conditions. It paves the way for more work in this area by addressing some of the limitations of collecting winter data and we believe it will lead to more robust models that could help in creating safe autonomous driving systems for winter conditions.

References

[1] 2017. Photoshop Snow Effect: Add falling snow to your photos! https://www.youtube.com/watch?v=ufGa2OkmKcQ

[2] 2018. KITTI for YOLO. https://github.com/oberger4711/kitti_for_yolo.

[3] Neal E Boudette. 2017. Tesla's self-driving system cleared in deadly crash. https://www.nytimes.com/2017/01/19/business/tesla-model-s-autopilot-fatal-crash.html

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261* (2019).

[6] Matt Hildebrand. 2020. faster rcnn pytorch multimodal. https://github.com/mathild7/faster_rcnn_pytorch_multimodal.

[7] Glenn Jocher. 2022. YOLOv5. https://github.com/ultralytics/yolov5c.

[8] Theodore Won-Hyung Kim. 2006. *Physically-based simulation of ice formation.* The University of North Carolina at Chapel Hill.

[9] Jae-Han Lee, Minhyeok Heo, Kyung-Rae Kim, and Chang-Su Kim. 2018. Single-image depth estimation based on fourier domain analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 330–339.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[11] Dianting Liu, Mei-Ling Shyu, Qiusha Zhu, and Shu-Ching Chen. 2011. Moving object detection under object occlusion situations in video sequences. In *2011 IEEE International Symposium on Multimedia*. IEEE, 271–278.

[12] Claudio Michaelis, Benjamin Mitzkus, Robert Geirhos, Evgenia Rusak, Oliver Bringmann, Alexander S Ecker, Matthias Bethge, and Wieland Brendel. 2019. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484* (2019).

[13] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. 2021. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics* 10, 3 (2021). https://doi.org/10.3390/electronics10030279

[14] Ken Perlin. 1985. An Image Synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*. Association for Computing Machinery, New York, NY, USA, 287–296. https://doi.org/10.1145/325334.325247

[15] Ken Perlin. 2002. Improving Noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (San Antonio, Texas) *(SIGGRAPH '02)*. Association for Computing Machinery, New York, NY, USA, 681–682. https://doi.org/10.1145/566570.566636

[16] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. 2020. Canadian Adverse Driving Conditions dataset. *The International Journal of Robotics Research* 40, 4-5 (Dec 2020), 681–690. https://doi.org/10.1177/0278364920979368

[17] Ujjwal Saxena. 2022. Automold–Road-Augmentation-Library. https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library.

[18] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. 303–314.

[19] Yi-Hsuan Tsai, Xiaohui Shen, Zhe Lin, Kalyan Sunkavalli, and Ming-Hsuan Yang. 2016. Sky is not the limit: semantic-aware sky replacement. *ACM Trans. Graph.* 35, 4 (2016), 149–1.

[20] Alexander Von Bernuth, Georg Volk, and Oliver Bringmann. 2019. Simulating photo-realistic snow and fog on existing images for enhanced CNN training and evaluation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 41–46.

[21] Rauf Yagfarov, Vladislav Ostankovich, and Salimzhan Gafurov. 2019. Augmentation-based object detection for winter time applications. In *2019 3rd School on Dynamics of Complex Networks and their Application in Intellectual Robotics (DCNAIR)*. IEEE, 178–180.