Improving Trust in Deep Neural Networks with Nearest Neighbors

Ritchie Lee*
Stinger Ghaffarian Technologies, NASA Ames Research Center, Moffett Field, CA 94035

Justin Clarke†
University of Massachusetts Amherst, 140 Governor's Dr., Amherst, MA 01003

Adrian K. Agogino‡, Dimitra Giannakopoulou§
NASA Ames Research Center, Moffett Field, CA 94035

Deep neural networks are used increasingly for perception and decision-making in UAVs. For example, they can be used to recognize objects from images and decide what actions the vehicle should take. While deep neural networks can perform very well at complex tasks, their decisions may be unintuitive to a human operator. When a human disagrees with a neural network prediction, due to the black box nature of deep neural networks, it can be unclear whether the system knows something the human does not or whether the system is malfunctioning. This uncertainty is problematic when it comes to ensuring safety. As a result, it is important to develop technologies for explaining neural network decisions for trust and safety. This paper explores a modification to the deep neural network classification layer to produce both a predicted label and an explanation to support its prediction. Specifically, at test time, we replace the final output layer of the neural network classifier by a k-nearest neighbor classifier. The nearest neighbor classifier produces 1) a predicted label through voting and 2) the nearest neighbors involved in the prediction, which represent the most similar examples from the training dataset. Because prediction and explanation are derived from the same underlying process, this approach guarantees that the explanations are always relevant to the predictions. We demonstrate the approach on a convolutional neural network for a UAV image classification task. We perform experiments using a forest trail image dataset and show empirically that the hybrid classifier can produce intuitive explanations without loss of predictive performance compared to the original neural network. We also show how the approach can be used to help identify potential issues in the network and training process.

I. Introduction

Deep neural networks have demonstrated impressive ability to solve complex perception and decision-making tasks including image classification [1], speech recognition [2], playing video games [3, 4], and playing board games [5]. Consequently, there is strong interest in applying deep neural networks to perception and decision-making tasks in self-driving cars [6] and unmanned aerial vehicles (UAVs) [7, 8]. Among other benefits, using deep neural networks and computer vision for perception reduces hardware cost by allowing cameras to replace more expensive sensors as the primary sensing modality. Deep neural networks also hold the promise to make better high-level decision-making than existing approaches.

However, robustness and trust remain key challenges that face deep neural networks, especially in safety-critical applications where failures can result in loss of life and property. At its core, the challenge is that the manner in which the neural network model derives its output is unintuitive to a human. That is, while a human can follow the mechanics of how a neural network derives its output, the computation process does not provide the necessary intuition for a human to have a deep physical understanding. As a result, deep neural networks are largely treated as black boxes.

*Researcher, Robust Software Engineering Group, AIAA Senior Member, ritchie.lee@nasa.gov
†Graduate Student, College of Information and Computer Sciences, jclarke@cs.umass.edu
‡Research Scientist, Robust Software Engineering Group, adrian.k.agogino@nasa.gov
§Research Computer Scientist, Robust Software Engineering Group, dimitra.giannakopoulou@nasa.gov

Viewing the model as a black box is problematic for various reasons. During development, it hinders the ability of the engineers to thoroughly understand the system, notice poorly trained models, provide safety guarantees, and effectively debug issues. During deployment, human operators that monitor the system can have difficulty distinguishing between correct and incorrect behavior. Human monitoring is important for safety-critical systems given the gravity of failures and that the system cannot be made perfect. Moreover, many studies have shown the existence of adversarial examples in deep neural networks, where small, often imperceptible, perturbations in the input can lead to drastic mispredictions with very high confidence [9]. These studies show that neural networks can not only be quite brittle but also fail without warning.

Explainability has been proposed as a key component to solving these issues [10, 11]. The idea is to have machine learning models return not only a prediction, but also some accompanying explanation that justifies its prediction to a human. Based on the explanation, the human can decide whether to trust the model prediction. A variety of approaches have been proposed to explain deep neural networks, including visualizations [12], attributions [13], and class activation maps [14, 15]. However, whether these methods produce explanations adequate to convince a human is quite subjective and highly task-dependent. Many methods, due to their origins, tend to work better for image recognition tasks where an object can be localized in the image rather than more abstract tasks such as predicting action directly from image.

In this paper, we take the approach of explanation by example, where, given an unseen example, the most similar examples in the training dataset are used as explanation. Our approach relies on intercepting the neuron outputs of an intermediate layer in the network. These intermediate vectors are known to form semantically relevant representations of the data and distances in the representation can be used for similarity [16]. Rather than making predictions by propagating through all layers of the neural network, we use the intermediate representation to feed a k-nearest neighbor classifier based on the training dataset. The k-nearest neighbor classifier uses the most similar examples i.e., the *nearest neighbors*, in the training set to vote for the prediction. The prediction process is made transparent to the user by returning the nearest neighbors involved. Because the nearest neighbors correspond to concrete examples in the dataset, they are semantically meaningful when presented to a human.

To make our discussion more concrete, we study an image classification task for a UAV navigating in a forest. In the scenario, the task is to decide, based on image input, the discretized heading of the UAV relative to a navigable forest trail. The forest trail can be in front, to the right, or left of the UAV. We study the utility of our approach via a number of experiments in this scenario. We show empirically that the predictive performance of the hybrid classifier is comparable to the original neural network classifier and we show that the nearest neighbors can make viable and meaningful explanations to a human. Furthermore, we show how our approach has helped identify certain issues in the training process and highlight interesting examples of the neural network classifier.

The remainder of this paper is organized as follows. Section II describes related work in explainability for machine learning. Section III reviews convolutional neural networks and k-nearest neighbor classifiers, which are fundamental concepts used in the paper. The section also describes the UAV forest trail navigation task on which we base our experiments. Section IV describes the construction and training of the hybrid classifier. Section V describes an adjacency issue encountered in our initial experiments and a train-test splitting procedure used to address it. The section also contains a sequential analysis of the forest trail data. Finally, Section VI presents the results of applying our hybrid classifier to analyze the forest trail dataset.

II. Related Work

Explainability in machine learning is an active area of research [11]. A variety of approaches have been proposed. A general approach, applicable to black box classifiers, is to learn an interpretable model that approximates the input-output behavior of the black box model. The interpretable model is then used to interface with the human. Rule-based models, such as decision trees [17], grammar-based decision tree (GBDT) [18], decision sets [19], Bayesian rule lists [20], are inherently interpretable and can be used in this way. Another approach, called locally interpretable model-agnostic explanations (LIME) [21], learns a locally valid linear classifier centered about an individual data point. Explainability has also been explored for planning and control, where sequences of states and actions need to be considered [22].

Deep neural network-based methods can make use of the available gradient information in the network. Attribution methods, such as integrated gradients [13] and layer-wise relevance propagation (LRP) [23], produce saliency maps that highlight the input dimensions that contributed most to the prediction. However, because saliency is computed per input dimension and spatial correlations are not captured, the resulting saliency maps can be very discontinuous, especially with high-dimensional inputs such as images. Saliency methods can also be unreliable as they are not invariant to simple transformations of the data [24]. Class activation map (CAM) [14] and gradient-weighted class activation

map (Grad-CAM) [15] use the spatial information from the convolutional layers of convolutional neural networks to produce more locally smooth heatmaps highlighting the most relevant parts of the input space. These methods have been shown to work well for object recognition tasks, where they can approximately localize the objects within an image. However, it is unclear how helpful they can be for more abstract tasks where there may not be a single object to be identified, as is the case in our forest trail application.

Visualization is another approach to understanding deep neural networks, which has been heavily investigated for image data. Neuron activation maximization [25] finds the image that maximally activates a particular neuron in the network, and activation atlases [12] produce two-dimensional visualizations of the representation space of the hidden layers. Representation learning models, such as β-variational autoencoder (β-VAE) [26] and information maximizing generative adversarial network (InfoGAN) [27], learn generative models with disentangled neurons that can generate novel examples where key characteristics of the image can be controlled through specific neurons.

Scene understanding considers images that can contain many objects and where objects need not necessarily occupy a large portion of the image. These approaches don't directly produce explanations, but can give other information to aid the user. For example, fast region-based convolutional neural network (R-CNN) [28] and mask R-CNN [29] return the predicted class and a bounding box that localizes the identified object; scene graphs extract explicit relationships between identified objects [30]; and instance segmentation [31] provides pixel-level segmentation of the image providing precise outlines of objects. These methods generally require additional detailed annotations of the images.

Our work takes a different approach to explainability where, for a test image, we provide the most similar examples from the training set. To compute the nearest neighbors, we create a hybrid model that combines a deep neural network with a k-nearest neighbor classifier. Neural networks and k-nearest neighbor classifiers have been explored in previous work [32–34], most notably the work of Papernot and McDaniel on deep k-nearest neighbors [32]. In deep k-nearest neighbors, nearest neighbors are computed at all intermediate layers and the final prediction is generated via a hypothesis testing procedure. Our work takes a considerably simpler approach feeding the representation from the penultimate layer into the k-nearest neighbor classifier. This simpler approach is arguably more intuitive due to the simpler decision-making process. But more importantly, explanations make the prediction process completely transparent to the user, since predictions and explanations are derived through the same underlying process.

III. Background

A. Notation

The following notation is adopted throughout the paper. A dataset D is a sequence of n input-label pairs $[(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)]$, where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{L}$, and $\mathbb{L}$ is the set of possible discrete class labels. For convenience, we define X to denote the sequence of inputs $[x_1, x_2, ..., x_n]$ and Y to denote the sequence of labels $[y_1, y_2, ..., y_n]$. Function operators on single inputs are extended with broadcast semantics such that they operate element-wise when presented with multiple inputs, e.g., $f(X) \triangleq [f(x_1), f(x_2), ..., f(x_n)]$.

B. Deep Neural Network Classifier

A (feedforward) deep neural network f maps an input x to an output y via a sequence of L computational transformations, called *layers* [35]. The data is transformed into increasingly more abstract *representations* of the data as it passes through the layers of the network. The input of a layer indexed by ℓ (where $\ell \in 0..L-1$) is the output of the previous layer at $\ell - 1$. A layer consists of smaller computation units, called *neurons*, that compute one dimension of the layer's output. The non-linear transformations at each layer f_ℓ are parameterized by *weights* w_ℓ, which are learned during the training process using backpropagation and stochastic gradient descent. A common non-linear function, or *activation function*, is the rectified linear unit (ReLU), which computes $max(0, u)$ for a scalar input u [35]. The final output layer of a classification network typically consists of a dense fully-connected layer with a *softmax* activation function given by

$$\sigma_j(u) = \frac{e^{u_j}}{\sum_{p=1}^{m} e^{u_p}}$$

where j is the neuron index and $u \in \mathbb{R}^m$. The softmax function produces the predicted probabilities of each class as the final output of the neural network. A loss function based on the cross-entropy between predicted probabilities and ground truth is used to generate the error signal for training. Overall, given an input x, the neural network performs the

following computation to predict a class y:

$$y = f(x) = f_{L-1}(f_{L-2}(...f_0(x)))$$

where the weights at each layer are not denoted for brevity.

Convolutional neural network. A convolutional neural network is a deep neural network with convolutional layers, which have been shown to perform very well on image processing tasks [35, 36]. Convolutional layers learn translation-invariant filters over its input via a weight sharing architecture. It is common for convolutional neural networks to alternate convolutional layers with *maxpooling* layers, which output the value of the maximum element from a multidimensional input. For a detailed presentation of convolutional neural networks, we refer to the reader to [35]. The exact mathematical details of these operators are not required to understand the discussion in this paper.

C. k-Nearest Neighbor Classifier

The k-nearest neighbor classifier is a non-parametric classifier that votes on the predicted label using the labels of the k closest examples in the training set [37, 38]. More precisely, let $\mathbf{d}$ be a distance function such that $\mathbf{d} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$. Then, for a test point $x \in \mathbb{R}^d$, the k-nearest neighbor classifier first finds the k nearest neighbors that minimize $\mathbf{d}(x, x_i)$ for $x_i \in X$, and then returns the mode of the labels of the k nearest neighbors. Ties are broken by selecting the label of the closest neighbor amongst the ties. "Training" a k-nearest neighbor model with dataset D consists primarily of storing D in a convenient data structure to enable fast distance searches at prediction time.

For the distance function $\mathbf{d}$, this paper uses cosine distance $\mathbf{d}_{cos}$ given by

$$\mathbf{d}_{cos}(u, v) = 1 - \frac{u \cdot v}{||u||_2||v||_2}$$

where $u \in \mathbb{R}^d$, $v \in \mathbb{R}^d$, and $|| \cdot ||_2$ is the Euclidean norm. Cosine distance is often used for comparing neural network representations [16].

D. Forest Trail UAV Navigation Scenario

We analyze a UAV navigation scenario, where the UAV follows a navigable trail in a forest environment based on image input. The scenario follows that described in [39] and our experiments are based on the image datasets shared publicly by the authors [40]. The scenario involves a UAV navigating a forest trail based on images from a monocular camera mounted at the front of the UAV. At each time step, the convolutional neural network uses the image input to predict the UAV's current heading relative to the general direction of the forest trail. The output of the neural network is a discrete class label: left, center, or right. If the output is center, for example, then the UAV is aligned with the forest trail and should move straight ahead to follow the forest trail. If the output is left, then the UAV's heading is pointed left of the forest trail, and the UAV should first turn right to align itself with the forest trail. In other words, the neural network classifier does not attempt to explicitly identify the trail or objects in the image, but rather directly predicts (the negative of) the control action to take.

To train the convolutional neural network, the authors collected image data by hiking forest trails with head-mounted cameras [39]. Three cameras were used, one mounted facing forward and two mounted pointed off-center to the left and right. As he hiked, the hiker tried to keep the forward camera aligned with the trail even as the trail curved. The data consists of three sequences of images, one from each camera and collected at 10 Hertz. The camera from which the image originates—left, center, or right—acts as the class label for the neural network training.

The experiments in this paper focus on datasets 001 and 002, which have the best image quality. The combined dataset contains 25,868 images in total. The images are resized to 101 by 101 pixels to speed up training. Each image is labeled 0 for left, 1 for center, and 2 for right.

IV. Dissecting the Neural Network for Trust

We explore a *hybrid classification model* that combines the representation of a deep neural network with a k-nearest neighbor classifier. The k-nearest neighbor is a natural choice as a classifier because notions of similarity and voting are very intuitive, making its decision-making process very understandable by humans. Additionally, the neighbors that participate in the voting can be presented to the user to explain the decision. However, the k-nearest neighbor performs poorly when directly applied to high-dimensional input data, such as images, because distance metrics are less meaningful in the high-dimensional space. Deep neural networks are naturally very effective at mapping

high-dimensional inputs to lower-dimensional and more semantically meaningful representations. These representations are found as the outputs of the intermediate layers, where deeper layers tend to represent higher level and more abstract features [25]. Consequently, we use the neural network representation at the penultimate layer to feed the k-nearest neighbor classifier.

The architecture of the hybrid classifier that we propose at train and test time is illustrated in Figure 1. To construct the hybrid classifier, we first train a convolutional neural network classifier f that maps an input image x to a class label y, such that $y = f(x)$. We learn the weights of the neural network using the training set D_{train} and standard backpropagation training methods, such as Adam [41]. Now, we remove the final output (dense and softmax) layer and consider the representation at the penultimate layer. We denote the new neural network $\hat{f}$, which maps an input image x to a representation z, such that $z = \hat{f}(x)$. The representations of the training set $Z_{train} = \hat{f}(X_{train})$ and their corresponding ground truth labels Y_{train} are used to train a k-nearest neighbor classifier g.

To perform inference for an unseen image x_{test}, we first evaluate its neural network representation $z_{test} = \hat{f}(x_{test})$, then evaluate the k-nearest neighbor classifier on the representation to obtain the class prediction, $y_{test} = g(z_{test})$. Our experiments in this paper set $k = 3$ and use cosine distance as a measure of neighbor similarity.

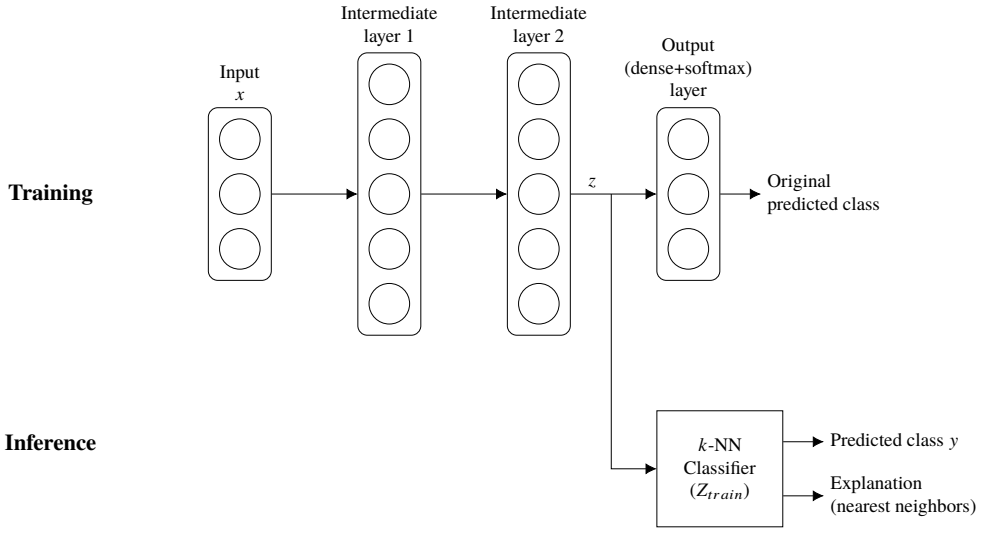

Fig. 1 Hybrid deep neural network and k-nearest neighbor classification model.

V. Training with Sequential Image Data

Our initial experiments divided the training and testing sets by uniformly sampling images from the dataset without regard for their sequential order. We obtained close to perfect accuracy scores and used our hybrid classifier to provide nearest neighbor explanations. We were surprised to discover that each test image had nearest neighbors in the training set that were almost identical to it. This observation quickly led us to realize that our test and training sets contained nearly identical images, which originated from adjacent frames in the original dataset. Figure 2 shows some examples of these images where the test image and the nearest training image look nearly identical. However, the images are in fact different and their minute differences can be seen by looking at the numerical difference in their pixel values. An example of these differences is shown in Figure 3.

Deep neural network training assumes that the training set represents independent and identically distributed samples of the underlying distribution [35]. However, this assumption is violated in our setting because images from frames of a video are sequentially correlated. By presenting the nearest neighbors as explanation to the user, our hybrid classifier exposed this adjacency issue and made it easy to spot. Images that are taken close together in time can appear very similar, especially at times when the hiker moves slowly or is stationary. For this reason, it is important to pay special attention when creating training and testing sets from the available data. A naive splitting of the data would result in a testing set where almost all the images have an almost identical image in the training set. This phenomenon would artificially inflate testing accuracy and mislead users of the model's true generalization ability.

Fig. 2 Examples of adjacent frames identified using nearest neighbors. The images and their nearest neighbors appear visually nearly identical. However, they are not identical as differences exist in their pixel values.

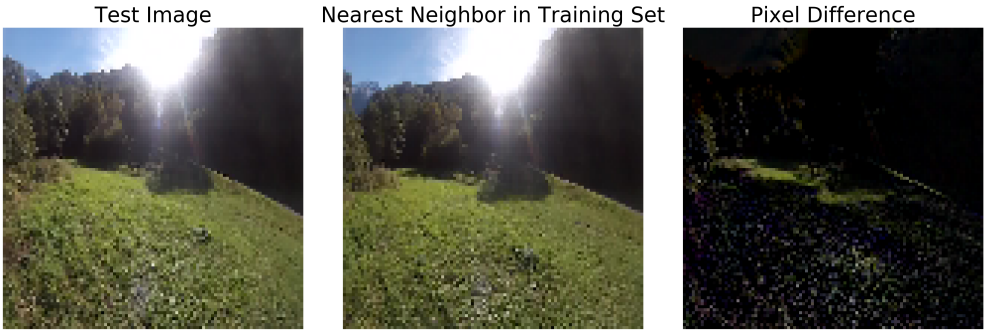

Fig. 3 Example of adjacent frames and their pixel differences.

To address this issue, when selecting images for training and testing sets, it is critical to ensure that no images in the testing set are too close in time to an image in the training set. We begin by sorting the data chronologically and then partitioning the data into chunks of size N. We randomly select approximately 60% of the chunks to be the training set and the remaining 40% of the chunks to be the testing set. To ensure a gap between all images in the testing and training sets, we discard the first and last $N/4$ images and retain only the middle $N/2$ images for the chunks in the testing set. This procedure is illustrated in Figure 4. We empirically selected $N = 64$, which ensures a gap of 16 images between train and test images. Following this construction method, the training set contains 15,488 images and the testing set contains 5,184 images. Performance scores in this paper are evaluated on testing sets generated using this method.

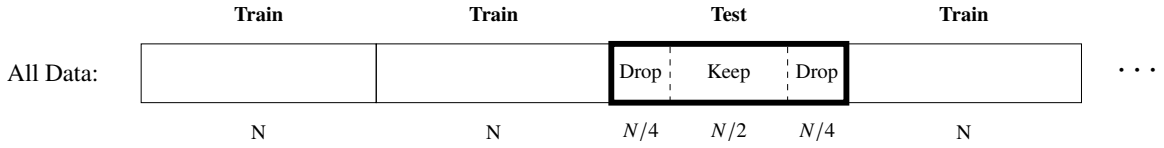

Fig. 4 Train-test split procedure.

Since the image data is ordered, it is possible to determine approximately which sections of the data are more difficult for the model to learn and generalize. That is, as the hiker traverses different trail environments, some portions of the hike will be more challenging to classify for a deep neural network than others. We quantify these variations via a leave-one-out cross-validation study. First, we split the image data into 10 partitions. For each partition, we train a classifier using data from all other partitions, then evaluate the accuracy for the current partition. By testing the generalization accuracy on each partition in turn, we obtain generalization scores for each section of the forest trail data. Figure 5 shows the result of the cross-validation experiment. The final section of the trail had significantly lower accuracy, which indicates that the final section may be different and more difficult to predict than other sections of the trail.

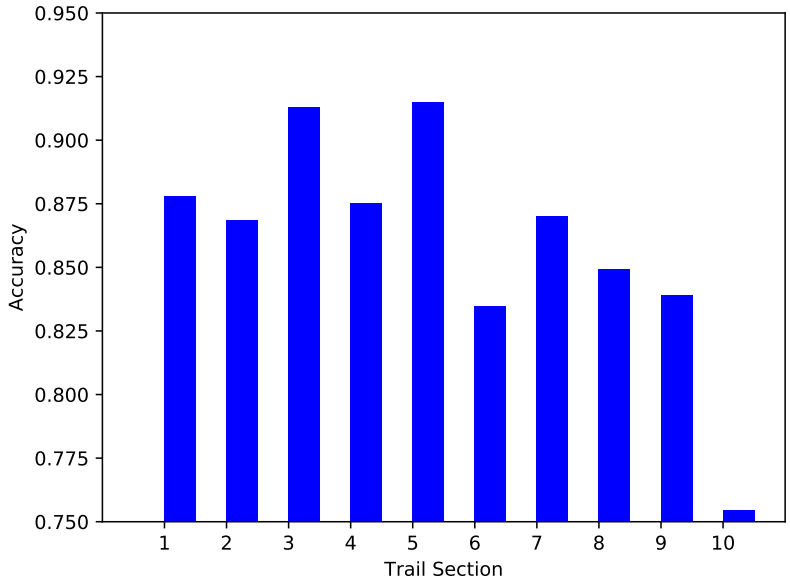

Fig. 5 Generalization accuracy by section from leave-one-out cross-validation.

VI. Application to Forest Trail UAV Navigation Scenario

In this section, we discuss observations made from the application of our approach to the Forest Trail UAV Navigation Scenario discussed in Section III-D. During training of the neural network, we apply on-the-fly data augmentation to help generalization of the trained model. Specifically, we apply a set of small random transforms to the image prior to presenting the image for training: rotation, up to ±18°; shear, up to ±20%; and zoom, up to ±20%. We use a batch size of $N = 64$ (see Section V) and train using the Adam optimizer [41]. Appendix A shows a summary of the neural network used for the experiments. For nearest neighbors, we use the balltree algorithm from the *scikit-learn* library [42].

We depict results obtained from our hybrid classifier as follows (see Figure 6, Figure 9, and Appendix B, for example). The leftmost column shows images from the testing set. The ground truth label and predicted label are indicated above each test image. The color of the test image title indicates whether the ground truth label agrees with the predicted label. Green indicates a correct classification, whereas red indicates a misclassification. The remaining columns show the three nearest neighbors from the training set. The ground truth label and distance to the test image are indicated above each nearest neighbor image. The color of the nearest neighbor title indicates whether the ground truth label of the nearest neighbor matches that of the predicted class. Green indicates that they match whereas red indicates a mismatch.

A. Explaining Predictions

The primary motivation for our approach is to be able to generate explanations that can help a human understand and gain confidence in the model's predicted output. We show that nearest neighbors present viable intuitive explanations for a user. Figure 6 shows various classification examples from the hybrid classifier. The hybrid classifier has high prediction accuracy, so the vast majority of classifications are correct. We also found that in most cases, the nearest neighbors voted unanimously. These properties are seen in Figure 6. We present an analysis of the nearest neighbor voting categories in Section VI-C.

Figure 6 shows examples of classifications and explanations from each class. In each case, we see that the test image and the nearest neighbors are visually very similar. They show scenes with similar foliage, rocks, geometries, and lighting. The visual similarity helps a human intuitively connect the nearest neighbor labels with the predicted label of the classifier, thus increasing the confidence of the model prediction. Conversely, a lack of visual similarity can be used as a basis for distrusting the predicted label.

B. Performance Evaluation

We compare the classification performance of the hybrid nearest neighbor classifier to the original neural network classifier by evaluating the accuracy of the two models on the testing set. The results are shown in Table 1. We find that the hybrid classifier has a slightly higher accuracy than the original neural network, despite adding the ability to explain predictions. In our experiments, we have observed comparable classification performance between the two models with minor variations between training runs.

Table 1 A Comparison of Prediction Accuracy on the Testing Set.

Model	Accuracy
Original network	89.1%
Hybrid classifier	89.8%

C. Nearest Neighbor Voting in Hybrid Classifier

The hybrid classifier uses nearest neighbor voting to make classification predictions. Our experiments use votes from the three nearest neighbors and choose the mode as the prediction. We break three-way ties using the closer neighbor. Because we have three class labels and three nearest neighbors, there are a number of ways the vote can result. Intuitively, we consider unanimous voting, where all the neighbors agree, as the most confident prediction. We can also have the converse where all three neighbors disagree. We consider these predictions as the least confident. We also consider the case where the first two neighbors disagree as this is a strong sign of contention in the voting.

Fig. 6 Hybrid model predictions and nearest neighbor explanations. Each row presents a test image followed by its three nearest neighbors in the training set. Ground truth label, denoted *gt*, and predicted label, denoted *pred*, are indicated for each test image. Ground truth label and neighbor distance, denoted *d*, are indicated for each neighbor. Labels are 0 for left, 1 for center, and 2 for right.

Figure 7 shows the counts of these three categories for the testing set broken down by classification correctness. We see that when the classifier makes a correct prediction, the voting is unanimous the vast majority of the time. Correct classifications rarely result from disagreements between the first two neighbors or disagreements between all three neighbors. On the other hand, the majority of misclassifications are also unanimous. However, the ratio of unanimous to other categories is not as high. Unfortunately, these results generally imply that confidence cannot be used as a reliable indicator of prediction correctness. Appendix B shows some examples of classifications from each voting category.

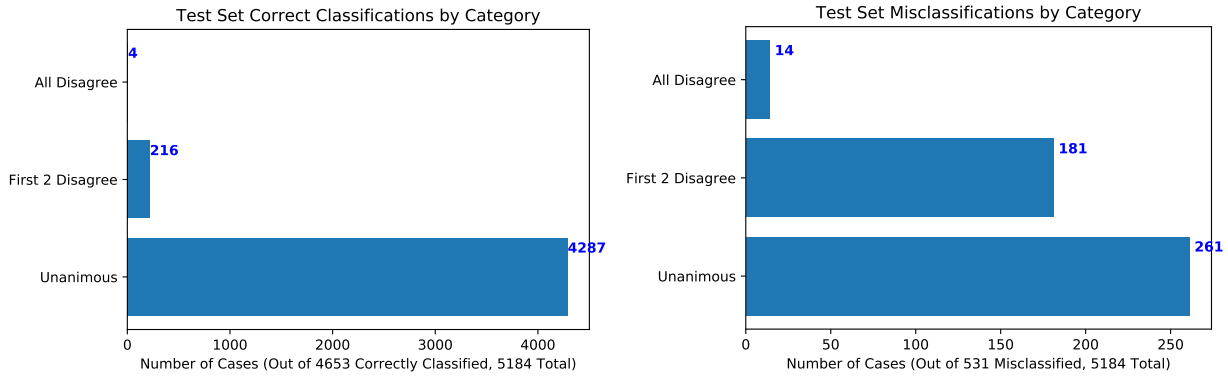

Fig. 7 Number of correctly classified and misclassified cases by voting category on the testing set.

D. Selected Examples

Dataset Issues. While the method of collecting images during a hike using head-mounted cameras is convenient as it alleviates the need to manually label the data, we note that the dataset does have a number of issues. These issues can cause confusion in the training as well as the explanation. Figure 8 shows some examples of problematic images in the dataset. First, a number of images do not show a clearly defined path, which makes the assigned labels seem arbitrary. Even a human cannot definitively label these images. Some images show obstructions by trees or bushes in the entire view. Other images show unobstructed, equally plausible terrain in all directions. Some images show an obstruction in the center and open paths on either side. There are also some rare instances that seem mislabeled altogether. For example, the first image in Figure 8 is labeled center, but the center of the image is clearly obstructed by trees. A second issue is lighting and shadowing. Some images are washed out in strong sunlight while others are largely covered in shadow. Finally, there are also some images polluted by non-stationary objects such as humans and vehicles.

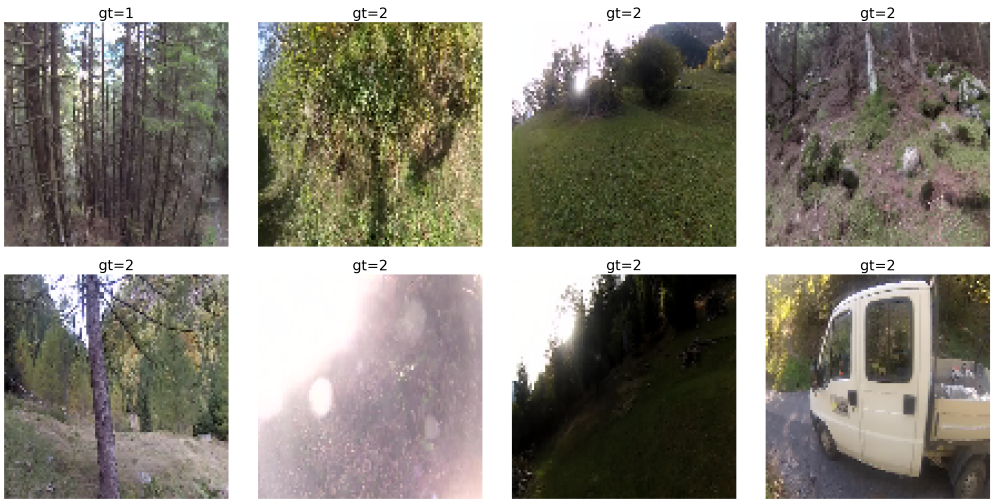

Fig. 8 Some problematic examples in the dataset.

Interesting Prediction Examples. Figure 9 presents six interesting examples of predictions from our experiments, one per row. The first column shows the test image and the remaining columns show the nearest neighbors from the training set starting with the first nearest neighbor in the second column.

Fig. 9 Some interesting classification examples.

In the first and second examples (rows 1 and 2), the classifier thinks, based on color and texture, that the path in the test image and the rock walls in the neighbor images look very similar. However, the features in these images should not be confused with each other because they imply very different UAV actions—the UAV should fly towards a path but turn away from a wall! A human would likely be able to differentiate between the two without problem. Part of the issue here is that the classifier does not understand depth. As a result, it may be useful in future work to consider integrating depth estimation techniques [43].

In the third and fourth examples (rows 3 and 4), we see that lighting and shadowing can play a major role in the classification. In example 3, the neighbors all have darker areas near the bottom of the images and light skies near the top of the images. This lighting effect plays a strong role in determining similarity by the neural network, whereas a human might conclude that the third neighbor is very dissimilar to the test image due to the mountainous landscape. In example 4, the test image contains a shadow of a tree projected onto the path. The classifier views the shadow projected on the ground as similar to a silhouette of a tree against the sky. A human would likely distinguish between the two cases. The last two examples (rows 5 and 6) are cases where the similarity distance provided by the neural network representation is unintuitive to a human. For example, in row 5, the second neighbor seems out-of-place and the third neighbor seems more visually similar to the test image than the second neighbor. In example 6, it looks like the second and third neighbors should be considered more similar to the test image than the first neighbor due to the configuration of the path and trees.

VII. Conclusion

In this paper, we considered the problem of explaining the outputs of a convolutional neural network for a UAV forest navigation scenario. We explored a hybrid classifier that, for inference, replaces the final output layer of a convolutional neural network with a k-nearest neighbor classifier trained on the training set. This modification enables the model to not only make class predictions but also provide the most similar examples in the training set to the user. We argued that explanations by example are viable and intuitive. Additionally, our experiments showed empirically that the hybrid model gives comparable classification performance as the original neural network. This approach helped us detect an adjacency issue in the way we created our testing set. Furthermore, we looked at how the nearest neighbors voted and found that confidence could not be used a reliable indicator for prediction correctness. Lastly, we discovered some interesting classification examples where the similarity measure based on the neural network representation can give unintuitive results. Overall, nearest neighbors increase transparency in the process of classification. Developers can select specific cases to be flagged for closer inspection in testing and deployment, e.g., inspect cases where voting is not unanimous.

Acknowledgments

This work was performed in the context of the Autonomy Teaming & TRAjectories for Complex Trusted Operational Reliability (ATTRACTOR) Project under NASA Convergent Aeronautics Solutions (CAS). We thank the ATTRACTOR team at NASA Langley for their support.

References

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

[2] Graves, A., Mohamed, A.-r., and Hinton, G., "Speech Recognition with Deep Recurrent Neural Networks," *International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2013, pp. 6645–6649.

[3] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[4] Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al., "Human-Level Performance in 3D Multiplayer Games with Population-Based Reinforcement Learning," *Science*, Vol. 364, No. 6443, 2019, pp. 859–865.

[5] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al., "Mastering the Game of Go without Human Knowledge," *Nature*, Vol. 550, No. 7676, 2017, p. 354.

[6] Bouton, M., Nakhaei, A., Fujimura, K., and Kochenderfer, M. J., "Safe Reinforcement Learning with Scene Decomposition for Navigating Complex Urban Environments," *Intelligent Vehicles Symposium (IV)*, 2019.

[7] Julian, K. D., Kochenderfer, M. J., and Owen, M. P., "Deep Neural Network Compression for Aircraft Collision Avoidance Systems," *AIAA Journal on Guidance, Control, and Dynamics*, Vol. 42, No. 3, 2019, pp. 598–608.

[8] Mern, J., Julian, K. D., Tompa, R. E., and Kochenderfer, M. J., "Visual Depth Mapping from Monocular Images using Recurrent Convolutional Neural Networks," *AIAA SciTech, Intelligent Systems Conference (IS)*, 2019.

[9] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R., "Intriguing Properties of Neural Networks," *arXiv preprint arXiv:1312.6199*, 2013.

[10] Alexandrov, N. M., and Allen, B. D., "Autonomy Teaming & TRAjectories for Complex Trusted Operational Reliability (ATTRACTOR)," NASA ARMD TACP/Convergent Aeronautics Solutions Project Showcase, Oral Presentation, Sep 2018.

[11] Gunning, D., "Explainable Artificial Intelligence (XAI)," *Defense Advanced Research Projects Agency (DARPA), Web*, 2017.

[12] Carter, S., Armstrong, Z., Schubert, L., Johnson, I., and Olah, C., "Activation Atlas," *Distill*, 2019.

[13] Sundararajan, M., Taly, A., and Yan, Q., "Axiomatic Attribution for Deep Networks," *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 3319–3328.

[14] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A., "Learning Deep Features for Discriminative Localization," *Conference on Computer Vision and Pattern Recognition*, IEEE, 2016, pp. 2921–2929.

[15] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," *International Conference on Computer Vision*, IEEE, 2017, pp. 618–626.

[16] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J., "Distributed Representations of Words and Phrases and Their Compositionality," *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 3111–3119.

[17] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A., *Classification and Regression Trees*, CRC Press, 1984.

[18] Lee, R., Kochenderfer, M. J., Mengshoel, O. J., and Silbermann, J., "Interpretable Categorization of Heterogeneous Time Series Data," *International Conference on Data Mining (SDM)*, SIAM, 2018.

[19] Lakkaraju, H., Bach, S., and Leskovec, J., "Interpretable Decision Sets: A Joint Framework for Description and Prediction," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2016, pp. 1675–1684.

[20] Letham, B., Rudin, C., McCormick, T. H., Madigan, D., et al., "Interpretable Classifiers using Rules and Bayesian Analysis: Building a Better Stroke Prediction Model," *Annals of Applied Statistics*, Vol. 9, No. 3, 2015, pp. 1350–1371.

[21] Ribeiro, M. T., Singh, S., and Guestrin, C., "Why Should I Trust You?: Explaining the Predictions of Any Classifier," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1135–1144.

[22] Agogino, A. K., Lee, R., and Giannakopoulou, D., "Challenges of Explaining Real-Time Planning," *ICAPS Workshop on Explainable Planning (XAIP)*, 2019.

[23] Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., and Samek, W., "Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers," *International Conference on Artificial Neural Networks*, Springer, 2016, pp. 63–71.

[24] Kindermans, P.-J., Hooker, S., Adebayo, J., Alber, M., Schütt, K. T., Dähne, S., Erhan, D., and Kim, B., "The (Un) Reliability of Saliency Methods," *stat*, Vol. 1050, 2017, p. 2.

[25] Erhan, D., Bengio, Y., Courville, A., and Vincent, P., "Visualizing Higher-Layer Features of a Deep Network," Tech. Rep. 3, University of Montreal, 2009.

[26] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A., "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," *International Conference on Learning Representations*, Vol. 3, 2017.

[27] Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P., "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets," *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 2172–2180.

[28] Girshick, R., "Fast R-CNN," *International Conference on Computer Vision*, IEEE, 2015, pp. 1440–1448.

[29] He, K., Gkioxari, G., Dollár, P., and Girshick, R., "Mask R-CNN," *International Conference on Computer Vision*, IEEE, 2017, pp. 2961–2969.

[30] Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D., Bernstein, M., and Fei-Fei, L., "Image Retrieval using Scene Graphs," *Conference on Computer Vision and Pattern Recognition*, IEEE, 2015, pp. 3668–3678.

[31] Dai, J., He, K., and Sun, J., "Instance-Aware Semantic Segmentation via Multi-Task Network Cascades," *Conference on Computer Vision and Pattern Recognition*, IEEE, 2016, pp. 3150–3158.

[32] Papernot, N., and McDaniel, P., "Deep k-Nearest Neighbors: Towards Confident, Interpretable, and Robust Deep Learning," *arXiv preprint arXiv:1803.04765*, 2018.

[33] Jiang, H., Kim, B., Guan, M., and Gupta, M., "To Trust or Not to Trust a Classifier," *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 5541–5552.

[34] Wallace, E., Feng, S., and Boyd-Graber, J., "Interpreting Neural Networks with Nearest Neighbors," *arXiv preprint arXiv:1809.02847*, 2018.

[35] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT press, 2016.

[36] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[37] Murphy, K. P., *Machine Learning: A Probabilistic Perspective*, MIT Press, Cambridge, MA, 2012.

[38] Bishop, C. M., *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[39] Giusti, A., Guzzi, J., Ciresan, D., He, F.-L., Rodriguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., Scaramuzza, D., and Gambardella, L., "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, 2016.

[40] Giusti, A., Guzzi, J., Ciresan, D., He, F.-L., Rodriguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., Scaramuzza, D., and Gambardella, L., "On the Visual Perception of Forest Trails," , 2019. URL `https://people.idsia.ch/~giusti/forest/web`.

[41] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-Learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[43] Pillai, S., Ambruş, R., and Gaidon, A., "Superdepth: Self-Supervised, Super-Resolved Monocular Depth Estimation," *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9250–9256.

Appendix A: Neural Network Summary

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 99, 99, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 47, 47, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 23, 23, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 32)          9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 200)               102600
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 603
=================================================================
Total params: 131,843
Trainable params: 131,843
Non-trainable params: 0
```

Appendix B: Classifier Prediction Examples by Voting Category

Fig. 10 Examples of misclassifications where the nearest neighbors voted unanimously. Red title on the test image indicates a misclassification.

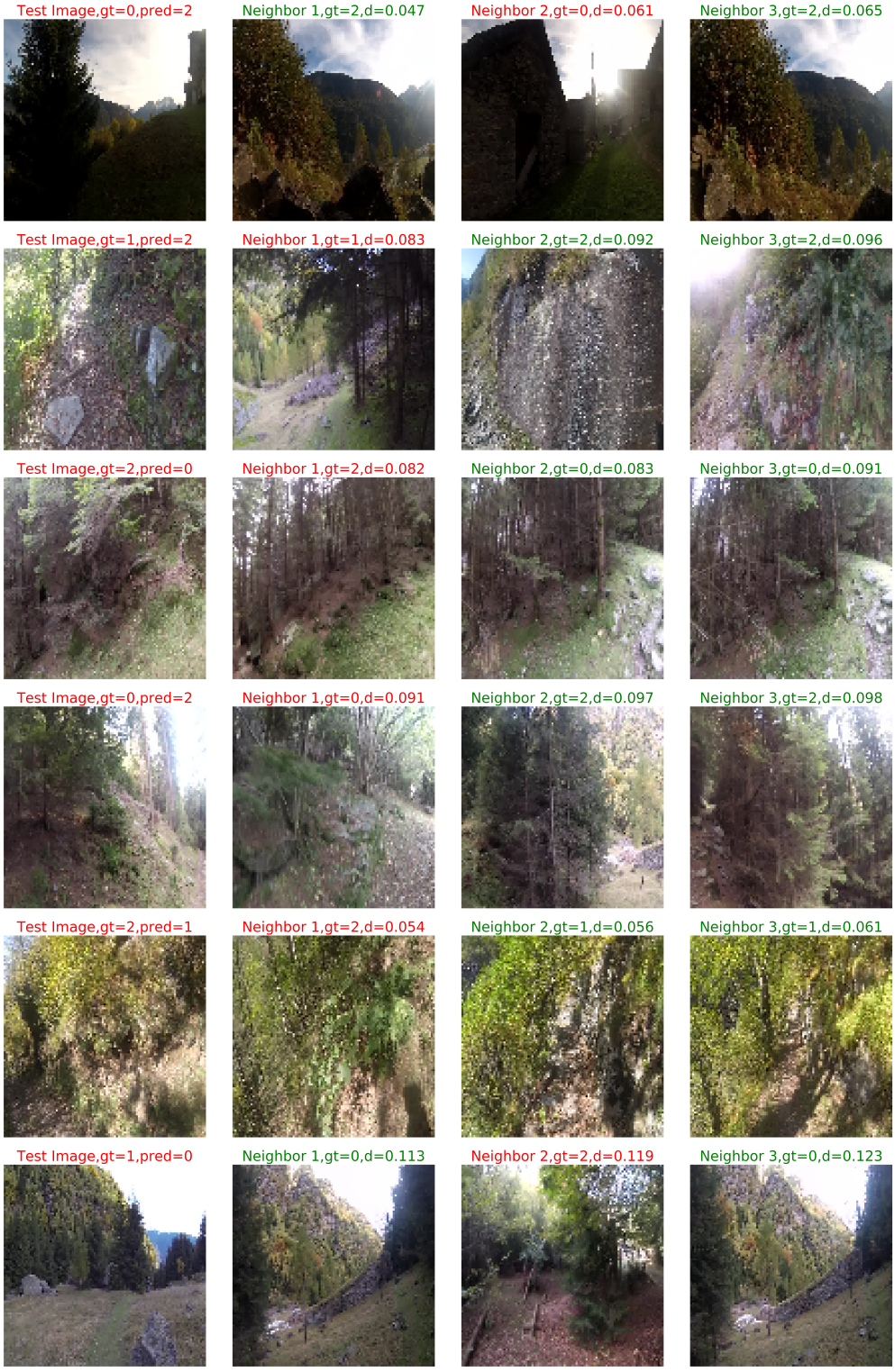

Fig. 11 Examples of misclassifications where the first two nearest neighbors disagree.

Fig. 12 Examples of misclassifications where all the nearest neighbors disagree. In this case, the nearest neighbor classifier predicts the label of the first nearest neighbor.

Fig. 13 Examples of correct classifications where the first two nearest neighbors disagree.

Fig. 14 Examples of correct classifications where all the nearest neighbors disagree. In this case, the nearest neighbor classifier predicts the label of the first nearest neighbor.