Machine Learning Explainability and Transferability for Path Navigation

Adrian K. Agogino*, Ritchie Lee†, and Dimitra Giannakopoulou‡
NASA Ames Research Center, Moffett Field, CA 94035

Deep neural networks are powerful tools for machine perception. Unfortunately their decisions are difficult to explain due to the complexity and size of the networks. Previously we have alleviated this issue by using the representational portion of a deep neural network and combining it with a k-nearest neighbor (KNN) classifier. Through inspection of the decisions made by the KNN, we can directly see the training data responsible for the decisions, allowing us to determine the quality of the overall decision and the quality of the representational layer of the deep NN. While the technique worked well, it requires tens of thousands of latent vectors to be stored for classification. In addition, it lacks the ability to show how parts of an image influence the classification decision. Here we address these issues by 1) Using a radial basis function network (RBFN) in place of the KNN allowing far fewer images to be used in deployment and 2) Using an autoencoder network for explainability. In addition to these techniques, we examine the effects of transfer learning to determine that results are robust. All results are tested on a domain where an unmanned aerial vehicle (UAV) navigates a forest trail through a single camera.

I. Introduction

While tremendous progress has been made in autonomy in urban environments, many domains require the use of autonomy in non-urban environments such as navigating through forest trails. These environments differ considerably from urban environments not only in their physical form, but also in how much data, mapping, and infrastructure is available. In this paper we focus on navigating non-urban environments through the processing of images generated from an on-board camera and classified through a convolutional deep neural network. Convolutional neural networks perform well at image classification. However, their decision logic is opaque for humans. Moreover, they rely on training a large number of free parameters which, while contributing to impressively high performance on a particular dataset, may not necessarily transfer to data used in real world deployments. Finally, such neural networks are hard to verify through formal verification as it is hard to even formulate their requirements. These issues make trust in deep neural networks problematic, as: 1) They are difficult to verify, 2) They are hard to understand, and 3) It is hard to know that good performance achieved in training, testing, and validation will transfer into real-world deployment.

Our work focuses on addressing these issues by 1) Increasing the understanding of deep neural networks through explanations [1, 2], and 2) Evaluating robustness of performance results through transfer learning. The idea behind explainability is that 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. With explainability, we can determine if an algorithm is behaving reasonably on a particular dataset, but we would still like to have some measure of its expected robustness for new datasets. To do this, we introduce new datasets that are similar enough to the original dataset that we would expect similar performance if the classifier is robust, but different enough that the new dataset could represent a realistic transfer learning experience in a real world deployment.

Several approaches have been proposed in the literature for explaining deep neural networks, including visualizations [3], attributions [4], and class activation maps [5, 6]. The methods tend to work better for object localization, rather than more abstract tasks such as image-based navigation. In previous work [7], we developed a hybrid classification approach which: 1) Intercepts the outputs of an intermediate layer in a convolutional neural network, 2) Uses these intermediate vectors to compare images for classification and explanation. Intermediate vectors are known to form semantically relevant representations of the data and distances in the representation can be used for similarity [8]. More specifically, rather than making predictions by propagating through all layers of the neural network, we use the

*Research Scientist, Robust Software Engineering Group, adrian.k.agogino@nasa.gov
†Research Scientist, Robust Software Engineering Group, AIAA Senior Member, ritchie.lee@nasa.gov
‡Research Computer Scientist, Robust Software Engineering Group, dimitra.giannakopoulou@nasa.gov

intermediate representation to feed a k-nearest neighbor classifier based on the training dataset. Typically tens of thousands of intermediate representations are stored for the k-nearest neighbor classifier.

In this paper, we replace the k-nearest neighbor classifier with a radial basis function network (RBFN). This classifier differs in that instead of storing an entire training dataset, it uses a much smaller set of "basis" images that are representatative of the dataset. It then compares an image to be classified with a weighted distance to all the basis images. This technique allows for a far smaller number of images to be stored, which has two advantages: 1) Less storage is needed for the basis images as compared to nearest neighbor, which is especially critical when deployed on a small UAV, and 2) The small number of basis images allows for hand inspection of all images to determine their quality.

Although explanations in terms of images are intuitive for humans, it is hard to conceptualize what aspects of the images make them more or less similar to each other. To address this issue we explore an alternative approach to explainability, where disentangled representations are generated using a variational autoencoder network. This technique attempts to map individual values of the intermediate vector used by the classifier, to properties of the image such as texture and lighting. Our experiments are performed on a UAV forest trail navigation scenario described in [7]. In the scenario, the UAV determines its heading relative to the trail by classifying images from a front-facing camera as being left, right or center, relative to the trail. Based on these classifications, the UAV tries to follow the trail by correcting its heading if it is too far to the left or to the right of the trail.

The remainder of this paper is organized as follows. Section II reviews related work on explainability for machine learning. Section III describes the forest trail navigation scenario that our approach targets. We motivate and present our idea of hybrid classifiers in Section IV. Section V discusses results when the classifiers trained in one environment are tested on different environments. The new variations for hybrid classification and explanations that we explore are presented next: the RBFN in Section VI, and disentangled representations in Section VII. Finally, Section VIII closes the paper with observations and conclusions.

II. Related Work

Explainability in machine learning is an active area of research [2]. 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 [9], grammar-based decision trees (GBDTs) [10], decision sets [11], Bayesian rule lists [12], are inherently interpretable and can be used in this way. Another approach, called locally interpretable model-agnostic explanations (LIME) [13], 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 [14].

Deep neural network-based methods can make use of the available gradient information in the network. Attribution methods, such as integrated gradients [4] and layer-wise relevance propagation (LRP) [15], 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 [16]. Class activation map (CAM) [5] and gradient-weighted class activation map (Grad-CAM) [6] 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 [17] finds the image that maximally activates a particular neuron in the network, and activation atlases [3] produce two-dimensional visualizations of the representation space of the hidden layers. Representation learning models, such as β-variational autoencoder (β-VAE) [18] and information maximizing generative adversarial network (InfoGAN) [19], 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) [20] and mask R-CNN [21] return the predicted class and a bounding box that localizes the identified object; scene graphs extract explicit relationships between identified objects [22]; and instance segmentation [23] 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 similarity between images, we use the representational layer of a deep neural network. The similarities can then be used for classification in a k-nearest neighbor or radial basis function network classifier. Neural networks and k-nearest neighbor classifiers have been explored in previous work [24–26], most notably the work of Papernot and McDaniel on deep k-nearest neighbors [24]. 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 either the k-nearest neighbor or the radial basis function 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. Forest Trail UAV Navigation Scenario

Search and rescue is an important domain that is often outside of urban environments. A victim could be lost or injured in a large search space that can benefit from a UAV quickly searching many different areas. While there are many facets to search and rescue, in this paper we tackle the subdomain of 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 [27] and our experiments are based on the image datasets shared publicly by the authors [28]. 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 we used two different datasets. In the first one, image data was collected by hiking forest trails with head-mounted cameras [27]. Three cameras were used, one mounted facing forward and two mounted pointed off-center to the left and right. While hiking, 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. Our experiments use datasets 001 and 002 from [27], which have the best image quality. The combined dataset contains 25,868 images in total. To examine the effects of transfer learning, we created a new dataset using a single 360 degree camera. In this camera, a hiker takes a 360 degree video of a hiking trail. This video is post-processed to determine trail location with respect to the camera. Then three images are extracted from the 360 degree image frame corresponding to left, right and center. This second dataset contains a total of 40,536 images. In our experiments, all 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. Machine Learning Designed for Explainability

A. Background

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., $f(X) \triangleq [f(x_1), f(x_2), ..., f(x_n)]$.

A (feedforward) *deep neural network* f maps an input x to an output y via a sequence of L computational transformations, called *layers* [29]. 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 [29]. 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.

A *convolutional neural network* is a deep neural network with convolutional layers, which have been shown to perform very well on image processing tasks [29, 30]. 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 [29]. The exact mathematical details of these operators are not required to understand the discussion in this paper.

Even though convolutional neural networks can have very high performance in classifying images, it is practically impossible for humans to understand them. For this reason, in our previous work, we proposed a modification towards explainability. Our proposed framework uses a trained convolutional neural network for its representation layer, but substitutes the classification with an algorithm that is easier to explain to humans. In our previous work, we used the k-nearest neighbor classifier [7].

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 [31, 32]. 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 $\mathbf{d}$, we use 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 [8].

Our work applies on-the-fly data augmentation to train neural networks, in order 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%. The network is trained using the ADAM optimizer [33]. For nearest neighbors, we use the balltree algorithm from the *scikit-learn* library [34].

B. Approaches to Explainability and Trust

This paper explores three ways of improving our previous results: 1) We introduce a transfer learning dataset; the new non-urban environment is qualitatively similar to the previous dataset, but different enough to estimate the robustness of the neural network classifier during deployment; 2) We augment our previous hybrid network that used a KNN-classifier with an RBFN classifier that uses a reduced number of images for classification, allowing for reduced storage and inspection of the image set; and 3) We explore disentangled representations as a way of giving meaning to individual elements of the representational vector.

Previously we used a hybrid network with a k-nearest neighbor classifier to explain decisions in a forest trail scenario [7]. The results showed that this technique was effective in determining the relationship between a test image and the images in the dataset influencing the neural networks decisions. The approach also allowed us to easily see some problems with the dataset. While effective, the main downside to the KNN approach is that: 1) To perform classification all the intermediate representation vectors from the training set are needed, and 2) For explainability, all of the original images of the training set must be available. This large set of images poses two problems 1) A small vehicle such as a UAV may not have enough capacity to store the intermediate vectors needed for classification, and 2) It is difficult to manually inspect the large number of images used to ensure that there are no obvious problems with the dataset.

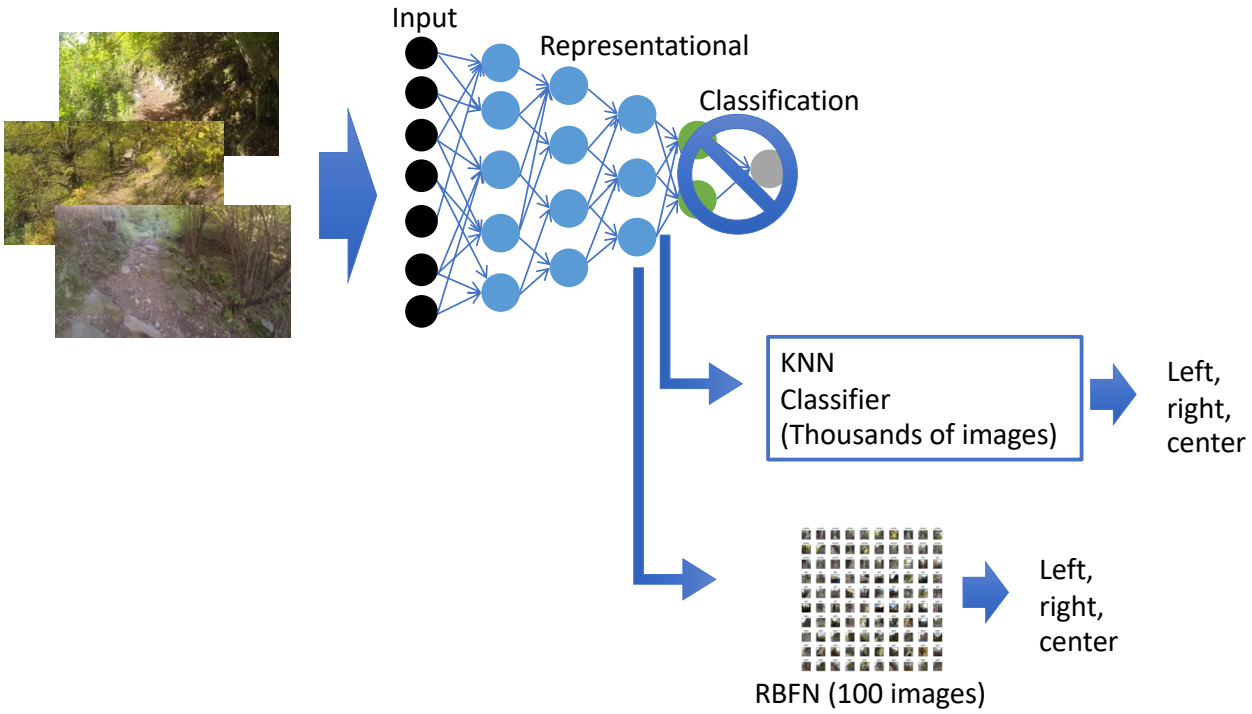

Fig. 1 Hybrid deep neural network using k-nearest neighbor or RBFN classification model.

To address these issues, we explore the use of an RBFN that uses 100 images for classification, instead of the tens of thousands used in the KNN. This smaller set of images is easier to store and can be hand inspected to make sure all the images are good training data. For the RBFN we use a similar *hybrid classification model* as in [7]. Similar to a KNN, an RBFN could be used directly on the image data. However, performance of an RFBN on raw image data would be poor due to the large dimensionality of the data and lack of invariants.

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 [17]. Consequently, similarly to the KNN classifier, the RBFN one uses the neural network representations at the penultimate layer as input.

The architecture of the hybrid classifier that we propose 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 [33]. 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 either a k-nearest neighbor classifier or a radial basis function network. For k-nearest neighbor, training simply involves saving the intermediate representations and their corresponding labels to an appropriate data structure. For RBFN, training involves choosing and storing a sampling of representations and then updating basis weights using the full dataset. To perform inference for an unseen image x_{test}, we first compute its neural network representation. Then we classify it using either the KNN or the RBFN classifier.

V. Transfer Learning

Transfer learning and generalization are two important properties in ascertaining the trust-worthiness of a machine learning system. While generalization involves how well the system performs on data coming from the same statistical distribution as the training data, transfer learning involves how well the system performs on data that is slightly different in nature than the training set. Often, transfer learning involves pushing the boundary on how different the testing data can be from the training data while still having the system perform well. Our approach is a bit different as we are

concerned with how trustworthy a system is when the test data is slightly different from the training data as inevitably happens in deployed systems. While it is desirable to have training data as close as possible to the actual data coming in from deployment, systems are typically deployed under different conditions and locations from where they were tested.

To analyze the effects of transfer learning we utilized five datasets:

- Dataset 1: Swiss Trail datasets 1-5
- Dataset 2: Small Bay Area trail dataset for training
- Dataset 3: Large Bay Area trail dataset for training
- Dataset 4: Large Bay Area trail dataset for testing
- Dataset 5: Bay Area trail data polluted with people in images

The first dataset comes from the Swiss trail data (see Section III). The other datasets come from the 360-degree videos taken on trails in the San Francisco Bay Area. Datasets 3 and 4 come from the same area, but different trails. Dataset 2 is a subset of Dataset 3, but is somewhat unrepresentative of the full dataset in terms of the texture of the vegetation. These datasets are chosen to examine a wide range of transfer conditions. As compared to the primary testing set (Dataset 4), Dataset 3 is the closest, followed by Dataset 2, with Dataset 1 being the furthest. In addition, Dataset 5 represents a problematic dataset.

As a baseline, we use a hybrid classifier with a representational layer and KNN trained on Dataset 3. The resulting classifier is tested on Dataset 4, and has a performance of 83%. This performance is slightly lower than our previous results without transfer learning, but still pretty decent since the testing data comes from a different trail than the training data. In our first experiment, we use a representational layer and KNN trained on the Swiss Trail data (Dataset 1) but tested on Bay Area data with Dataset 4. The resulting classification rate is 54%, showing that the classifier has difficulty transferring from the Swiss Trail data to the Bay Area data. We then perform a similar test with a representational layer and KNN trained on Dataset 2, and tested on Dataset 4. The resulting classifier has a classification rate of 69%, which is still relatively poor.

Our next experiment uses Dataset 5, which is polluted with images of the people taking the video. We build a classifier with a representational layer and KNN trained on Dataset 5, with some data withheld in order to also test the classifier on Dataset 5. When tested on the withheld data from Dataset 5, the classifier achieves a performance of 99.9%. This nearly perfect performance can be attributed to the classifier identifying which frame has a person in it, rather than doing the much harder problem of determining orientation by analysing the trail. However, when this classifier is tested on Dataset 4, it achieves significantly lower performance of 76%. This suggests that transfer learning can detect when a classifier has poor generalization.

VI. Explaining Predictions with RBFN Hybrid Classifier

As mentioned previously, we explore the RBFN classifier as a way of reducing storage of example images as well as allowing full inspection of the image set used for classification. Similarly to KNNs, RBFNs use distance functions to images to determine a test example's class. However, instead of using every image in a training set, RBFNs use a much smaller number of representational images as "bases" to perform classification (these bases do not even have to be actual images, though in this paper they always are). Also, unlike KNN, each basis image has a weight corresponding to how strongly that basis corresponds to each class. A basis image that is very representative of a class should have a large value of weight for that class. In contrast, a basis image that is unrepresentative of a class will have a negative weight corresponding to that class. These weights are typically determined through least squares optimization using a training set. In particular, we use a linear optimizer based on stochastic gradient descent from the *scikit-learn* library. To determine the class of a test image, the inverse of the distance of the image to a basis is computed and then multiplied by all the weights of the basis to form a class vector with respect to the basis. This is performed on all the basis images and all of the resulting class vectors are summed. The class is then determined by the class corresponding to the largest value of the summed vector.

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. Previously, we had shown that using a hybrid classifier with k-nearest neighbor can be helpful in explaining predictions and identifying problems with training data, both improving trust in the system [7]. Here, we show that a hybrid RBFN classifier is also helpful in explaining predictions, with the additional advantage that it has to use far fewer images.

Our first evaluation aims to determine if the hybrid RBFN classifier has acceptable classification rate. To verify this we run the classifier using the same forest trail data as we did before. We chose to use 100 basis images as this should be enough images to cover the space, but it is few enough images that each one could be individually inspected for quality.

Results show that indeed an RBFN network can be effective with 100 images used as bases. We found the classification rate of the original neural network to be 89.1% and the classification rate of the hybrid RBFN to be 85.7%. While not quite as good, it is close.

Figure 2 shows a sample of 100 images used as bases. These are the only images needed for classification and can be manually inspected for appropriateness. For instance, our original basis image set contained images where fingers from the camera person were blocking much of the image. These images were easily identified by inspection and removed. In addition, examining distances from a test image to basis images can be used for explanation and trust. In this example, an image of class "right" is evaluated with the classifier by measuring the distance of this image to all of the basis images. The classifier classifies this image correctly as "right", but also much more insight can be obtained by looking at the basis images after being sorted by distance as shown in Figure 2. The results show that the first five closest images are of the correct class and that seven out of the first ten are of the correct class. The result also shows that none of the ten most distant images are of the correct class "right". This gives us some confidence that the classifier is behaving reasonably, as it can be hand inspected that the distances to the basis images are on a reasonable trend.

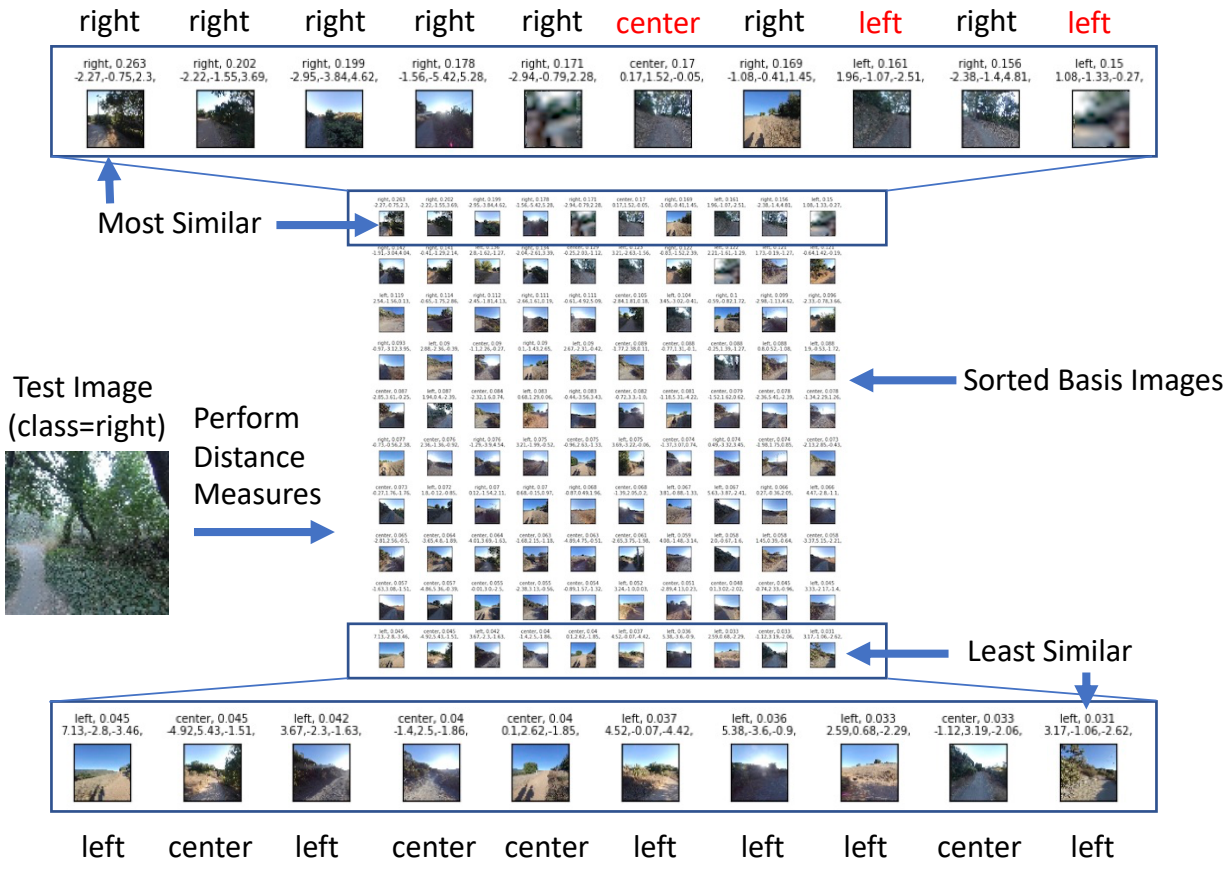

Fig. 2 Basis images for RBFN. 100 images can be used to make predictions instead of thousands used for KNN. Smaller number allows full inspection of images and reduced storage needs in deployment. Here an image of class "right" is classified by the RBFN. By examining the basis images sorted by distance to the test image, we can see that the closest images are indeed of the correct class and that the most distant images are of different classes.

VII. Explanations through Disentangled Representations

The hybrid classification model in Section IV-B is composed of a truncated neural network model that produces a representation and an interpretable classifier such as a KNN or RBFN that operates on the representation. While the interpretable classifier allows classification decisions to be explained through similar examples, the representation

on which the similarity is evaluated is not interpretable. Studies have shown that a deep neural network captures semantic information of the dataset in its latent representation [8]. The representation contains the information needed to drive classification. However, the representation itself is generally not intuitive. The main reason is that variations in individual dimensions of the representation do not generally correspond to independent and intuitive concepts, but instead are confounded and vary multiple attributes at once. Recently, various algorithms have been proposed to learn *disentangled representations*, where each neuron in the representation corresponds to an independent and fundamental source of variation in the data [18, 19, 35]. Existing work has shown that when applied on a dataset of faces, algorithms can find disentangled attributes such as azimuth, lighting, and skin color. Similarly, when applied on a dataset of chairs, the algorithms can tease out attributes such as viewing angle, size, and chair leg and back styles [18, 19, 36]. In this section, we explore the use of disentangled representation from a β-variational autoencoder (β-VAE) [18] in the hybrid classification model. Disentangled representation preserves the ability to find similar images, while potentially making the representation more intuitive.

A. β-Variational Autoencoder

The variational autoencoder (VAE) is an unsupervised learning method that learns a probabilistic latent representation of the data. The VAE consists of an encoder network q that maps an input image x to a representation distribution $q(z \mid x)$, and a decoder network that maps a representation $z \sim q(z \mid x)$ to a reconstructed image $\hat{x}$. The representation distribution is assumed to be a multivariate Gaussian with diagonal covariance matrix. Under this assumption, the output of the encoder network is a mean vector μ and variance vector σ^2. The parameters of the encoder and decoder networks are jointly trained to minimize loss using an optimizer such as ADAM [33]. The loss function consists of two terms. The first term aims to maximize the marginal likelihood of the observed data. This term is approximated by a reconstruction loss $\mathcal{L}$ that penalizes the difference between the input image and the reconstructed output image. Our experiments use (the negative of) structural similarity index measure (SSIM), which is a similarity metric for images that accounts for perceived image quality and features [37]. The second term of the VAE loss function minimizes the Kullback-Leibler (KL) divergence between the representation distribution and a prior distribution $p(z)$. The prior distribution is chosen to be a multivariate isotropic unit Gaussian.

The β-VAE loss function introduces a weight parameter $\beta \in \mathbb{R}$ on the second term as shown in Equation (1). The parameter is chosen to add additional weight (i.e., $\beta > 1$) on the prior term, encouraging the encoder to learn distributions more like the Gaussian prior.

$$\mathcal{L}_{\text{SSIM}}(x, \hat{x}) + \beta \cdot KL(q(z \mid x) \mid\mid p(z)) \tag{1}$$

It has been found that the dimensions of the representation become increasingly disentangled as β increases, thus learning increasingly independent fundamental variations in the data. However, there is a trade-off in β-VAEs in that increasing β also increases reconstruction loss, which means that the model doesn't represent the data as well. Variants of β-VAE with different formulations have been proposed to overcome this shortcoming. In this paper, we also explore total correlation variational autoencoder (β-TCVAE) on our task [35]. However, our experiments did not find significant differences in the results between β-TCVAE and β-VAE on our dataset, and thus we only present results from β-VAE.

B. A Hybrid Classification Model Based on Disentangled Representations

We explore a hybrid classification model that combines a disentangled representation from a β-VAE with a KNN classifier. First, we train (in an unsupervised manner) a β-VAE on the training data. The training produces an encoder network that maps an input image to its representation distribution, and a decoder network that maps a representation to a reconstructed image. Next, we try to isolate the effect of each dimension of the representation and assign it an intuitive label. We perturb—one dimension at a time—the representations of the training images, and generate images using the decoder network. A human visually inspects the images and manually assigns an intuitive label to each representation dimension based on the variations observed across the images. At inference time, an unseen image is passed through the encoder network to generate a representation, and then the KNN is used for classification. The nearest neighbor images from the KNN are presented to the user as an explanation as with the original hybrid classification model. However, the disentangled representation adds an additional interpretable element for the user. Because each dimension is associated with a different attribute, we can quantitatively compare attributes between query and neighbor images. Not only can the attributes be sorted by similarity, they can also be presented with intuitive descriptions. For example, the algorithm can say that the query image has a similar amount of foliage and visibility of the sky as the neighbor image, but a wider path.

C. Learning Disentangled Representations in the Forest Trail Dataset

We applied β-VAE to learn a disentangled representation of the Forest Trail dataset. Figure 3 shows a sweep of an example image varying the top five most relevant dimensions. To generate the visualization, we first take a test image and pass it through the encoder network to obtain the representation distribution mean μ and variance σ^2. The dimensions are sorted in decreasing order of relevance. Relevance is computed as the average standard deviation value from the encoder output for a dimension over the training set. Higher variation is regarded as more relevant. Each row of Figure 3 perturbs one dimension of μ while keeping the other dimensions constant. The perturbed vector is then passed through the decoder to produce an image. The center column is the reconstruction of the original image (without perturbation) while columns to the left and right show images with increasing levels of negative and positive perturbations, respectively. A good disentangled representation would show each row varying an independent attribute of the data.

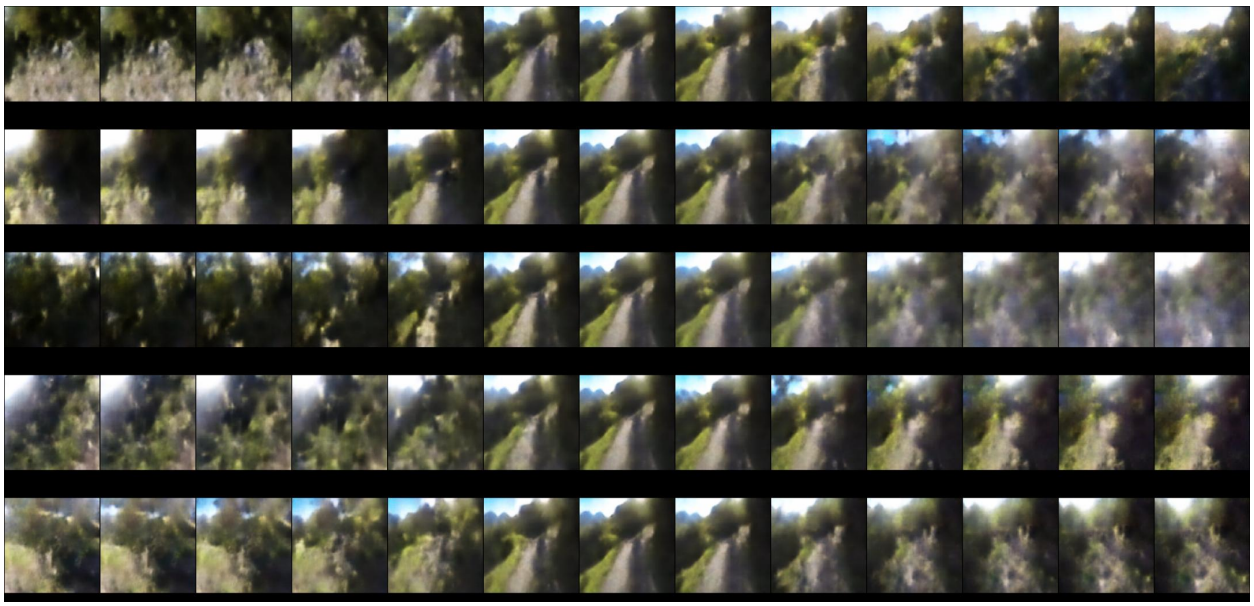

Fig. 3 Image sweep over the top five most relevant dimensions of the representation. Each row sweeps a single dimension starting from the most relevant dimension in the top row. The center column is the original reconstructed image.

Figure 3 shows variations as each dimension is perturbed. The interpretation can be somewhat subjective. In the first row, we see more and brighter sky as we move from left to right. We also see a widening of the path as we move to the left. In the second row, we generally see more foliage to the right and more path to the left. The third row shows more shadows on the left, and a brighter scene on right. The fourth row shows variation from a grassy path on the left to a more dirt-like path on the right. The fifth row shows a more grassy hill to the left and more sandy and rocky path to the right. Overall, while we see some general trends in the image sweeps, the disentanglement did not produce crisp intuitive attributes. Some variables were confounded, such as the path, sky, and shadowing in the first dimension. Some attributes also seemed to manifest differently with different input images. Because the interpretation was so subjective, it was unclear how to assign intuitive universal descriptions to the representation dimensions. We also explored using β-TCVAE and saw similar results. While disentangling representations teases out fundamental variations in the data, there is no guarantee that these variations correspond to intuitive concepts. We suspect that the primary reason for our results is that our dataset comprises a wide variation of scenes rather than single objects as in prior work.

We combined the β-VAE representation with a KNN and evaluated its prediction performance. We found that the hybrid classifier based on the disentangled representation had a 70.0% accuracy on the test set, compared to the hybrid classifier using the truncated convolutional neural network representation, which had an 89.8% accuracy. We suspect the reason is because the β-VAE is trained in an unsupervised manner and does not make use of the label information at all. As a result, the learned representation is not optimized for the actual classification task. Overall, our experiments with disentangled representations have yielded mixed results. We saw some indications of disentanglement in our visualizations. However, the dimensions remained confounded and did not reflect clear intuitive attributes. We also saw

a major drop in classification accuracy when using the β-VAE representation.

VIII. Conclusion

In this paper we augmented our previous results on increasing explainability for deep neural networks used for a a UAV forest navigation scenario in three different ways: 1) We added a radial basis function network to our hybrid classifier to allow for a dramatic reduction in images needed for classification and explanation, 2) We investigated using disentangled representations to give meaning to individual components of the intermediate representation, and 3) We applied our algorithms to a new transfer learning dataset to see the robustness of the classifier. The results showed that the radial basis function network is a viable alternative to our previous k-nearest neighbor network. The results for disentangled representations are mixed: while we saw some indications of disentanglement in our visualizations, the dimensions remained confounded and did not reflect clear intuitive attributes. Finally our transfer learning results showed that the deep neural network could transfer to different datasets taken in similar environments, but had issues transferring to datasets taken in a different environment.

Acknowledgments

This work was supported by the Autonomy Teaming & TRAjectories for Complex Trusted Operational Reliability (ATTRACTOR) Project under NASA Convergent Aeronautics Solutions (CAS), and the System-Wide Safety (SWS) Project under NASA Aeronautics Research Mission Directorate (ARMD) Airspace Operations and Safety Program (AOSP). We thank the ATTRACTOR team at NASA Langley for their support.

References

[1] 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.

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

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

[4] 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.

[5] 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.

[6] 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.

[7] Lee, R., Clarke, J., Agogino, A. K., and Giannakopoulou, D., "Improving Trust in Deep Neural Networks with Nearest Neighbors," *AIAA SciTech, Intelligent Systems Conference (IS)*, AIAA, 2020.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

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

[25] 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.

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

[27] 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.

[28] 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`.

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

[30] 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.

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

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

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

[34] 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.

[35] Chen, R. T., Li, X., Grosse, R. B., and Duvenaud, D. K., "Isolating Sources of Disentanglement in Variational Autoencoders," *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 2610–2620.

[36] Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A., "Understanding Disentangling in β-VAE," *arXiv preprint arXiv:1804.03599*, 2018.

[37] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, Vol. 13, No. 4, 2004, pp. 600–612.