Infra-Red Target Recognition using Realistic Training Images generated by modifying Latent Features of an Encoder-Decoder Network

Maliha Arif, student member, IEEE, Abhijit Mahalanobis, Fellow, IEEE

Abstract—Generating realistic images has been a challenging problem in computer vision, with many researchers focusing on novel methods and datasets to produce benchmark results. Our motivation for the same arises from the dearth of real training images for recognizing targets in infrared imagery. We propose an encoder-decoder architecture for generating realistic medium wave infrared images of targets at various azimuth angles, in day or night conditions, and at different ranges. Specifically, we use a CNN-based siamese autoencoder network that modifies the latent space embedding of a given input view to produce a novel output view. First, we train this network with a limited set of real images of the targets, and show that it can generate new and previously unseen views of the same. We show that the network operates in the non-linear feature subspace and learns the underlying manifold to develop a semantic understanding of the targets. We use the structural similarity index measure (SSIM) to quantify how the generated and real images of targets compare. Finally, we show classifiers trained with the generated images are able to recognize targets in real test images.

Index Terms—ATR Classification, view prediction, deep convolutional autoencoders, infrared imagery

I. INTRODUCTION

DATA collection and annotation has been a long-standing problem in machine learning, which is both expensive and time consuming. This is a particularly challenging issue for infrared (IR) images for which ground truth is scarcer than for RGB images. Motivated by this, we show that a limited set of infrared images of objects can be used to train encoder-decoder networks to generate additional new views which can then be used to train a classifier to recognize them. Figure 1 shows the overall proposed approach. We use the publicly available Automatic Target Recognition (ATR) Algorithm Development Image Database [1] in our experiments. Using a subset of the original images, we first train the view prediction network to create a large synthetically generated (but natural looking) training corpus. These predicted images are then used to train a VGG-16 classifier network to recognize targets in real infrared images. We assert that our approach learns the 3D representation of objects (as well as the underlying manifold), and quantify the performance of the proposed approach using both SSIM, as well as the overall performance of the classifier.

Broadly, view prediction algorithms can be classified into two categories[2]. The first category is geometry based where the weights of the neural network learn the geometric transformation of a pixel between two views. The second category is where view prediction is seen as a learning problem or feature extraction problem where the network is trained to understand the 3D representation of objects such that it can predict what a new view of an object will look like if it has seen other views of the object before. Existing works involve either using the first approach, (geometry based)[3], [4], [5] or the second...
approach, (learning based) [6]. We adopt the learning based approach and manipulate the latent space features to generate the desired output view.

There are various related works on novel or unseen view generation, most of which focus on using synthetic datasets. Tatarchenko et al. [6] and Dosovitskiy et al. [7] work is most similar to ours in view prediction in the nonlinear feature subspace. They make use of ShapeNet dataset [8] which contains over 8000 models of cars and chairs. Dosovitskiy uses a CNN based decoder architecture which creates a new view when given a viewpoint and class type such as a chair. The method involves using only latent space vector extracted from a training set and performing upsampling. However, it is restricted to generating images or interpolating between them using the training corpus only. Tatarchenko et al. [6] extends the problem of generating novel views for not just chairs but cars also. Further, the problem is made more appealing by ensuring that the class information is not given to the network as input, rather an encoder-decoder architecture is used to automatically extract this information. Further this work also emphasizes on learning 3D representations from view morphing. It uses multiple generated views and depth maps simultaneously to create a point cloud for the object. Our network is also based on an autoencoder similar to proposed in [6] but without considering the 3D depth aspect. We do not propose to explicitly learn the 3D information of our model as suggested in the referred work but rather we want to exploit the latent space embedding of autoencoders to learn the underlying semantic information and pixel relation between input and novel output views [9].

Other recent works include Zhu et al. [10] who propose a multiview perceptron learning algorithm that works on facial images and produces arbitrary viewpoints. It uses a random sampling technique for view representation using different sets of neurons for input view and then creates the desired view. As a result, the output images on Multi-PIE dataset used in training are sharp and clear with more robust features but dependent on random sampling of neurons and not on a single input image, which makes the method unsuitable for our purpose. Deep Convolutional Inverse Graphics Network, introduced by [11], make use of an encoder decoder architecture but involve manipulating the graphics code group in the latent space. Compared to [10], the quality of images is lower and restricted to smaller variations. Yang et al. [12] propose a recurrent neural autoencoder network which is trained end to end for generating multiple views using single images of faces in the Multi-PIE dataset. This however requires large amount of training data which is unavailable in our case. Further, it has limited ability to generate multiple views, and works only on discrete sets of views.

Another important direction for image generation which cannot be ignored is the use of adversarial networks by Goodfellow et al. [13]. GANs synthesize images based on some condition [14], this maybe labels [15], text vectors/embeddings [16] or images [17]. We did not choose GANs for our work mainly because the images are generated randomly and noise can be seen in the generated images [6].

II. DATA DESCRIPTION AND PREPROCESSING

The dataset we use in our experiments is publicly available and is called ‘DSIAC-ATR Algorithm Development Image database’ [1]. It was collected by US Army Night Vision and Electronic Sensors Directorate (NVESD) with the intent of supporting work in the infrared and ATR domain. This database contains both visible and MWIR (midwave infrared) images of military and civilian vehicles. The 207 GB of MWIR images include ground truth and meteorological data. There are 10 vehicle categories consisting of different visually appearing vehicles including SUVs, pickup trucks, battle tanks and anti-aircraft weapons. Each category or target has multiple videos collected during the day and during the night at different ranges (distance from the camera capturing the target). Each video shows the object moving in two 360° circles. As a result, for each target, there are images for every 1° in azimuth from 0° to 360°. The ranges lie between 900m-5000m. The objects of interest are relatively small compared to the overall image size of 640 x 512. Since these images represent thermal heat distribution on the surface of the objects, our work is very different from other view prediction research studies [6], [7], [2], [18], [19], [11] which were designed for color (RGB) imagery. Furthermore, earlier techniques have assumed unlimited samples to train the view synthesis network, and using 3D modeling software to generate images at any azimuth, elevation and range parameters. On the other hand, we are limited to the annotations supplied with the dataset, and parameters like range, azimuth, elevation, light intensity cannot be varied arbitrarily.

The original 16-bit data is converted from 'ARF' format to avi videos as was done in [20]. Individual frames are then extracted as jpeg images which are cropped to smaller 64 x 64 chips which contain the target. The dataset has 5 ranges; we work with objects that are at 1000m, 1500m and 2000m range. We create our dataset of views which are all 5° apart and contain images both from night time and day time at the specified ranges. We end up having approximately 50 images per object of different azimuth per range for a total of approximately 450 images in all. Our goal is to train an autoencoder such that it can generate new views of an object using a single input view. To achieve this we create pairs of training images with one serving as an input view (at a given orientation, time of day and range), while the second serves as the desired view of the same object but at a different orientation, time of day and range. We then split this subset of data into a training, validation and test set in a 60:20:20 ratio. We end up with 15k training pairs and 5k test pairs per range. We first train and test the autoencoder at a single range, and then extend the idea using all 3 ranges as further discussed under results and evaluation section ahead. The only preprocessing applied on the cropped images is normalizing them before feeding them into the network. Further, these are grayscale images and our networks both for view/state generation and classification have three channels as they assume input to be RGB. To resolve this problem, we copy each input on all three channels. This means that all the 3 channels have the same information.
Fig. 2. Architecture and heatmaps for activations of our deep CNN based State/View predictor autoencoder network. A 5x1 'state vector' is being injected in a separate 64 dimensional fully connected layer for feature fusion.

III. METHOD

Many generative models require the use of good input features for training and to produce quality results. Autoencoders have emerged as a powerful method for feature extraction and selection in the nonlinear feature subspace [21]. We experiment using a basic autoencoder and a siamese network, and discuss a method for manipulating the latent space to generate new views. We show that not only are the images visually realistic, but they can also be used for training classifiers, thereby enabling target recognition using very few original images.

A. Predictor Autoencoder (AE) Network

Our deep CNN autoencoder is based on [2], [6], [7], [11]. The autoencoder is symmetric having four convolutional layers in the encoder and same number of deconvolutional layers in the decoder. We use an image size of 64x64 for our network. The images are propagated through 4 convolutional layers each having 32,32,64 and 64 filters respectively. The filter size is 5x5 for the first layer and then 3x3 for the following layers. At the end of the encoder, we obtain an output of 4x4x64 dimensions which we flatten to create a 1024 long embedding. We then inject a 5x1 dimensions long state vector into a separate 64 dimensional fully connected layer. The state vector basically indicates the desired azimuth, the time of day (day or night) and the range. Specifically, the first and second elements are the sine and cosine of the time of data (which is 0

Fig. 3. Our proposed siamese network architecture and training technique using both vanilla autoencoder and predictor autoencoder.
for night and 1 for day). The third and fourth elements are the sine and cosine of the azimuth angle. The fifth element is the desired range (in meters) divided by 100. This state vector is injected into a separate FC layer to ensure that the independent parameters are preprocessed enough to enable the network to combine them with the input features in order to produce the output features. In an unsupervised setting, the 1024 long embedding of our input view (which may be of any azimuth, time and range) is concatenated with this independent feature vector. The resultant 1088 embedding is now jointly processed using two more 1024 long fully connected layers. The idea is to exploit the latent space embedding and change it using the injected parameters such that when the embedding passes the decoder part of the network, it creates the desired viewpoint. This can be observed in detail via Figure 2 which shows heatmaps for activations for each convolutional layer. The decoder part of our network consists of 4 upsampling layers which have the same number of filters per layer and filter footprint as the encoder. We use leaky relu activation function as it does not tend to degrade AE performance as compared to simple relu activation [21]. Relu activation function always outputs 0 for negative inputs and hence weakens the process of reconstructing output using input features. As a result, a detour is using leaky relu [22], [23]. For the last layer, we use hyperbolic tangent since it produces steeper slope and stronger gradients [24], [21].

We use average error or mean square loss function to train the AE networks.

\[
MSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - x_i)^2} \quad (1)
\]

where \(y_i\) is the ground truth and \(x_i\) is the predicted output.

**B. Siamese Autoencoder (AE) Network**

We also investigate a two stream network shown in Figure 3. Here, our goal is to guide the convergence of the latent space features, in addition to the MSE criteria for the generated image at the output. Block 1 is a vanilla autoencoder [25] which is first trained to compress and reconstruct each image. This block is trained separately, and is simply used to obtain the desired embeddings for the training image which serve as the output view later when training block 2. Block 2 shows a second autoencoder model, our Predictor AE, which we use to predict unseen views for our objects. However, unlike the first network, this one is trained with input/output training image pairs which represent different views of the same object. In addition to the MSE loss for the generated image, we also minimize the error between the latent space features of this network and the ideal ones produced by the network in block 1 in response to the image which represents the desired view. We also formulate an objective function, based on mean square error for both the final predicted images and latent space embeddings.

The loss function comprises of two terms; the first term aims to guide the latent space embedding to a set of values which best describe our desired viewpoint, while the second minimizes the MSE between the generated image and its real counterpart. The first term in the loss function \(L_e\) is given by

\[
L_e = \sum \|e_2 - e_1\|^2 \quad (2)
\]

where \(e_1\) and \(e_2\) are the encodings of the desired viewpoint and its ideal version, respectively. Note that the ideal embedding \(e_2\) is generated using our vanilla autoencoder, and \(e_1\) is the embedding which we obtain while training our predictor autoencoder network - block 1, as shown in Figure 3. The second term in our objective function \(L_o\) is given by

\[
L_o = \sum \|y_2 - y_1\|^2 \quad (3)
\]

where \(y_1\) and \(y_2\) are the predicted and ground truth images. It aims to reduce the loss between the generated image and its real ground truth image. Hence, the final reconstruction loss \(L_t\) is the sum of \(L_e\) and \(L_o\), i.e.

\[
L_t = L_e + L_o \quad (4)
\]

\[
L_t = \sum \|e_2 - e_1\|^2 + \sum \|y_2 - y_1\|^2 \quad (5)
\]

**C. Classifier architecture**

One of our principal goals is to show that a classifier trained with predicted images can effectively recognize real images. For this purpose, we employ a VGG-16 [26] network pretrained on the ImageNet dataset. Since our task involves only 10 targets, we fine-tune the model as follows. We freeze the weights for all convolutional blocks, but remove the last fully connected layer. We then add 2 additional fully connected layers with 1024 and 10 neurons respectively on top of this structure as shown in Figure 4. The last layer with 10 neurons uses softmax activation function. To prevent overfitting, a dropout layer is also added in between these two fully connected layers. We then fine-tune this network using our predicted images. We train the classifier using the categorical cross entropy loss function

\[
CE = -\log \left( \frac{e^{s_p}}{\sum_j e^{s_j}} \right) \quad (6)
\]
where \( S_p \) is the probability score for the positive class.

IV. Experiments

We use a keras implementation with tensorflow backend for all our experiments. The GPU we use is a NVIDIA GeForce GTX 1080 Ti which is relatively fast than its predecessors and gives us good processing speed for all the experiments.

A. Predictor Autoencoder (AE) Network

We train our basic autoencoder in the following manner. Our input data has azimuth angles with a step size of 5°. The parameters which control the state of the predicted image include azimuth, day or night and range. We choose to work with a batch size of 16 with the Adam optimizer [27]. We use this adaptive learning rate method and train our network with a higher learning rate of 1e-3 for initial few epochs and then decrease the learning rate to 1e-4 for later epochs until we reach convergence. This happens at around epoch 80. We use mean square error as the loss function and attain 0.0012 as our final error.

B. Siamese Autoencoder (AE) Network

For our siamese network, our training strategy involves training two networks separately. We first train the vanilla autoencoder, as shown in Figure 3, encode and decode the input image (without any changes to the same azimuth, time of day and range). We then use the embedded latent space features of this network to train our predictor network, block 2 in Figure 3. Here the input and output views are different from each other. The output represents an unseen state- azimuth and time of day according to the parameters defined. The desired parameters for the output view (i.e. azimuth, time of day, and range) are injected in the latent space embedding to allow the features to be modified and manifold learning to take place. We use Adam optimizer [27], an adaptive learning rate method along with learning rate schedules. We train our network, (Block 1 and Block 2) at a learning rate of 1e-3 for first few epochs. Once we see the loss has dipped to a low convergence point, we switch our learning rate to 1e-4. This slower learning enables both networks to reach complete convergence after 80 epochs. We extract \textit{ideal} embeddings from block 1 after it has been trained. These embeddings are then fed with the ground truth images for view 2 in block 2 of our proposed architecture. Our siamese network uses the loss function as defined in Section 3(B) to compute the final average error given in Table 1. We use a batch size of 64 and our input data has azimuth angles with a step size of 5° as already aforementioned.

<table>
<thead>
<tr>
<th>Model</th>
<th>Azimuth Step Size</th>
<th>Computed Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic AE network</td>
<td>5°</td>
<td>0.0012</td>
</tr>
<tr>
<td>Siamese network</td>
<td>5°</td>
<td>0.0009</td>
</tr>
</tbody>
</table>

TABLE I
A QUANTITATIVE COMPARISON - COMPUTED ERROR USING DIFFERENT APPROACHES

C. Classification model

Our goal is to show that predicted images (obtained using very few original training images) can be used to train a classifier to recognize targets in real images. We therefore train the VGG16 network with predicted images and test it with real images of the targets. The network is trained using a batch size of 16. Since we use pretrained weights, we fine-tune for few epochs to achieve convergence. We use adam optimizer with a learning rate of 1e-5. The only preprocessing involved in our images is normalizing them before passing them through the network.

V. Results and Evaluation

To evaluate our reconstructed images, we use SSIM which is defined as,

\[
SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{7}
\]

SSIM is a perception-based model that observes changes in structural information and perceives any image degradation therein. Structural information includes the inter-dependencies between the pixels and how strong this dependency is when the pixels are spatially close. These dependencies are a determining factor about the structure of the object in the visual scene. The SSIM formula includes luminance masking and contrast masking terms for better comparison which are combined to give the form as shown in equation 7 where \( \mu \) denotes average and \( \sigma \) denotes variance. We use python and its libraries to implement SSIM metric. We get the following results as shown in Table II, closer to 1 is better and depicts good reconstruction. Figure 5 and Figure 6 show corresponding visual results.

As a control experiment, we first train and test a classifier entirely with real images, the results of which are shown in Figure 7. Here, we attain an overall classification score of 96%. All classes perform extremely well except for ‘ZSU23’, which seems to be a difficult class.

In the next experiment, we train the network entirely with the predicted images for all classes, with training images generated every 5° in azimuth. We then test using only real-world images for all classes and obtain 88% overall classification accuracy. Figure 8 shows our obtained results. For targets 2S3, BMP2, BRDM2, BTR70, D20 and PICKUP, the classification scores are extremely good which is supported by their SSIM scores as shown in Table II. Target class ‘SUV’
gets slightly confused with 'PICKUP' and 'T72' whereas for target 'ZSU23', we obtain a low classification score since our generated images for that particular class have a very low SSIM. Figure 6 illustrates our predicted images for ZSU23.

While the previous experiments generated images at different azimuth angles, we now explore the ability to predict images at new ranges for which no original training images are available. For this experiment, we exclude the images of the BMP at a range of 2000m from the training of the predictor network. Furthermore, we also do not generate any images of the BMP at this range to train the classifier. The results of
training the classifier without any 2000m range images of the BMP is seen in Figure 9. The F1-score for BMP2 is low as compared to other classes because the classification network has not seen the 3rd range for BMP2 at all.

Following this, we used the same predictor network to generate new views of the BMP at a range of 2000m and included these predicted images in the training corpus for the classifier. We see obvious improvements in the F1-score for BMP2 in Figure 10 and the improvement in overall classification accuracy as compared to our control experiment in Figure 9. While the f1-score for the first experiment ranges from 0.29-0.48 for class BMP2, the f1-score in the second experiment ranges from 0.40 - 0.69 which is higher and shows much improvement.

1) t-SNE visualizations: Manifold representations are not only convenient for visualizing high dimensional data, but also provide insights about how different classes inter-relate in the data space. Thus, the manifold may be viewed as the high-dimensional surface on which the data points lie. A lower dimensional embedding of the manifold ensures that the geodesic relations between the data points are preserved. The problem of predicting new views from a few original views may be viewed as learning the behavior of the manifold, and the transition from one view to another as a movement along the surface of the same manifold. Thus, we interpret our results in terms of how the original and predicted views behave in the latent space. We verify that the predicted views and the ground truth images are 'close' on the underlying manifold. We use the t-Distributed Stochastic Neighbor Embedding (t-SNE) [28] method for visualizing the manifolds in the latent space of the autoencoders. The t-SNE method models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability. Figure 11(a) illustrates our intuition. We pick one object class, 2S3- a Self-Propelled Howitzer and plot its embeddings for ground truth and predicted view in the following manner. The ground truth image is passed through the network using 0-degree input feature vector. We extract two embeddings from this pass. We get the first 1024 dimension embedding soon after the encoder. We obtain the second embedding for ground truth after feature fusion of 0 degrees and just before the 1024 dimension embedding enters the decoder network. We fuse 0-degree input as we do not want any change in the view or orientation. Further, this allows a fair comparison with the predicted view. In a similar way, we also obtain 1024 long embeddings of predicted views before they enter the decoder network from our Predictor AE network. We make sure these predicted views are the same for which the ground truth images were passed through the Predictor AE network. Figure 11(a) shows the data points for the ground truth view and predicted view obtained via latent space embeddings before they enter the decoder network. Green data points represent ground truth views and red data points represent predicted views. We can observe how well the network learns the manifold and makes correct predictions in response to variation in view and time feature input. All ground truth and predicted data points overlap. To further affirm if our predictions are correct and that our network does perform manifold learning while predicting them, we also plot the embedding obtained for ground truth soon after encoder network. Figure 11(b) shows the t-SNE visualization. We can observe that these embeddings (denoted by green data points) make 2 clusters in between the overlapping predicted and ground truth data points, both of which were obtained using embeddings before they entered the decoder network and after feature fusion. This confirms that all three embeddings lie in the same feature subspace as evident from the manifold illustration.
Leonardo DRS for this research. We have performed a novel study of predicting unseen views or ‘states’ as we call it on infrared imagery. We perform novel state prediction as an image to image translation task. The use of a deep autoencoder for exploiting the latent space embeddings and predict novel states has shown promising results. Our proposed method has provided us an effective modeling of entities present in our dataset. The method is able to understand the difference in object shape and class without any reliance on color. Moreover, our experimental evaluations show the quality of our generated images and the effectiveness of our technique. We are able to classify these generated images relatively well and provide reasonable contribution to learning when training data is scarce.

VI. CONCLUSION
We have performed a novel study of predicting unseen views or ‘states’ as we call it on infrared imagery. We perform novel state prediction as an image to image translation task. The use of a deep autoencoder for exploiting the latent space embeddings and predict novel states has shown promising results. Our proposed method has provided us an effective modeling of entities present in our dataset. The method is able to understand the difference in object shape and class without any reliance on color. Moreover, our experimental evaluations show the quality of our generated images and the effectiveness of our technique. We are able to classify these generated images relatively well and provide reasonable contribution to learning when training data is scarce.

ACKNOWLEDGMENT
The authors would like to thank the generous support of Leonardo DRS for this research.

REFERENCES
9


Dr. Abhijit Mahalanobis is currently an Associate Professor in Center for Research in Computer Vision (CRCV) at the University of Central Florida (UCF). His primary research areas are in video/image processing for target detection and recognition, computer vision, and computational imaging. He has over 170 journal and conference publications in this area. He also holds four patents, co-authored a book on pattern recognition, contributed several book chapters, and edited special issues of several journals. Abhijit completed his B.S. degree with Honors at the University of California, Santa Barbara in 1984. He then joined the Carnegie Mellon University and received the MS, and Ph.D. degrees in 1985 and 1987, respectively. Prior to joining UCF, Abhijit was a Senior Fellow at Lockheed Martin in Orlando. He has also worked previously at Raytheon in Tucson, and was a faculty at the University of Arizona and the University of Maryland. Abhijit was elected a Fellow of SPIE in 1997, and a Fellow of OSA 2004 for his work on optical pattern recognition and automatic target recognition. He was elected Fellow of IEEE in 2015 for his work on the theory of correlation filters. He served as an associate editor for Applied Optics from 2004-2009. He was as an associate editor for the journal of the Pattern Recognition Society from 1994-2003. He served on OSA's Science and Engineering council in the capacity of Pattern Recognition Chair from 2001-2004, and as Technical Group Chair for Information Acquisition, Processing and Display on OSA's Board of Meetings from 2012-2015. He also serves on the organizing committees for the SPIE conferences, and OSA's annual and topical meetings. Abhijit was also recognized as the 2006 Scientist of the Year by Science Spectrum Magazine, a publication of the Career Communication Group, Inc.

Maliha Arif holds a B.E degree in Telecommunications Engineering with honours (Gold-Medal) from NED University of Engineering and Technology, Karachi, Pakistan (2016) and a M.S degree in Computer Science from the University of Central Florida, Orlando, USA (2019). She is currently working towards her Ph.D. at the Center for Research in Computer Vision (CRCV) at the University of Central Florida (UCF) under the supervision of Dr. Mahalanobis. Her research interests include computer vision, image generation and machine learning.