Learning Geometric Equivalence Between Patterns Using Embedding Neural Networks

Olga Moskvyak
School of Electrical Engineering and Computer Science
Queensland University of Technology
Brisbane, Australia
Email: olgamoskvyak@gmail.com

Frederic Maire
School of Electrical Engineering and Computer Science
Queensland University of Technology
Brisbane, Australia
Email: f.maire@qut.edu.au

***Abstract*—Despite impressive results in object classification, verification and recognition, most deep neural network based recognition systems become brittle when the view point of the camera changes dramatically. Robustness to geometric transformations is highly desirable for applications like wild life monitoring where there is no control on the pose of the objects of interest. The images of different objects viewed from various observation points define equivalence classes where by definition two images are said to be equivalent if they are views from the same object. These equivalence classes can be learned via embeddings that map the input images to vectors of real numbers. During training, equivalent images are mapped to vectors that get pulled closer together, whereas if the images are not equivalent their associated vectors get pulled apart. In this work, we present an effective deep neural network model for learning the homographic equivalence between patterns. The long term aim of this research is to develop more robust manta ray recognizers. Manta rays bear unique natural spot patterns on their bellies. Visual identification based on these patterns from underwater images enables a better understanding of habitat use by monitoring individuals within populations.**

We test our model on a dataset of artificially generated patterns that resemble natural patterning. Our experiments demonstrate that the proposed architecture is able to discriminate between patterns subjected to large homographic transformations.

I. Introduction

Common machine learning tasks like classification and recognition involve learning an appearance model. These tasks can be interpreted and even reduced to the problem of learning manifolds from a training set [1]. Useful appearance models create an invariant representation of the objects of interest under a range of conditions [2]. A good representation should combine invariance and discriminability. For example, in facial recognition where the task is to compare two images and determine whether they show the same person, the output of the system should be invariant to the pose of the heads. More generally, the category of an object contained in an image should be invariant to viewpoint changes.

The motivation for our work is the lack of fully automated identification systems for manta rays. The techniques developed for such systems can potentially be applied to other marine species that bear a unique pattern on their surface. The task of recognizing manta rays is challenging because of the heterogeneity of photographic conditions and equipment used in acquiring manta ray photo ID images like those in Figures 1 and 2. Many of those pictures are submitted by recreational divers. For those pictures, the camera parameters are generally not known. Manta rays are born with natural spot patterning that stays unchanged throughout their life and can be used like human fingerprints to identify individuals [3]. These individual patterns enable scientists to study populations, track long range movements and record habitat use.

Fig. 1. Manta rays bear natural patterning across their belly. These unique spots enable the identification of individuals.

The state of the art in manta ray recognition is the system proposed in [3]. Their software requires user input to manually align and normalize the 2D orientation of the ray within the image. Moreover, the user has to select a rectangular region of interest containing the spot pattern. The images have also to be of high quality. In practice, marine biologists still prefer to use a decision tree that they run manually.

The local max-pooling layers in Convolutional Neural Networks (CNN) allow a network to be somewhat spatially invariant to the position of features. As max-pooling is typically executed on windows only a few pixels wide, many layers of max-pooling and convolutions, as well as a lot of training examples are required to achieve a more global spatial invariance [4].

A recent hybrid approach introduces a new learnable mod-

Fig. 2. The same manta as in Figure 1. A non affine transformation is required to align the belly patterns of the two images.

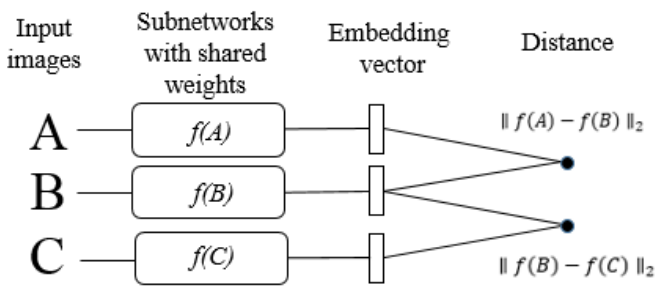

Fig. 3. Triplet network architecture. Three input images A, B and C are fed to three identical subnetworks that share the same weights. Embedding vectors $f(A)$, $f(B)$ and $f(C)$ are computed for each image. The Euclidian distance between these embeddings provides a proxy for the similarity between images. The whole network is trained by minimizing a loss function that penalizes large distances between the embeddings of images from the same class, and penalizes small distances between the embeddings of images from different classes.

ule, the *Spatial Transformer*, which explicitly allows the spatial manipulation of data within the network [5]. This differentiable module can be inserted into existing convolutional architectures, giving neural networks the ability to actively spatially transform feature maps, conditional on the feature map itself. However, this approach requires the user to specify the parametric form of the transformation. We prefer a less engineered approach with an end-to-end solution that is agnostic to the type of transformation to be learned.

In order to develop robust algorithms for recognizing manta spot patterns, we consider the problem of recognizing artificially generated patterns subjected to projective transformations that simulate changes in the camera view point. Artificial data allow us to experiment with a large amount of patterns and compare different network architectures to select the most suitable for learning geometric equivalence. In this paper, we focus on the *verification problem*. That is, we try to guess whether two images are from the same equivalence class or not.

Our main contribution is the demonstration that a relatively simple class of neural networks can distinguish between patterns under a wide range of viewing conditions. We present a deep neural network that can learn to recognize homographically equivalent patterns. We explore the suitability of different architectures.

The paper is structured as follows. Section II reviews related work in face verification and applications that use Siamese and Triplet network architectures. Section III discusses Siamese and Triplet network architecture and the loss functions used for training. Section IV describes the design of our experiments, and provides quantitative results. Section V concludes the paper.

II. Related Work

Our approach to learning geometric equivalence is inspired by some of the solutions created for the *face verification problem*. That is the problem of deciding whether a given pair of face images correspond to the same person or not. The verification task is different from the traditional image classification problem where all classes are known in advance and there are a sufficient number of training examples for each class. In the verification problem the number of classes (e.g. distinct individuals) is not specified in advance. The verification task is generally decomposed into two main steps [6], [7]:

1) non-linear mapping of the input vector to a lower dimensional embedding vector space,
2) computation of the distance between embedding vectors to decide whether the two inputs are from the same equivalence class.

During training, the model learns a similarity metric that minimizes distances between representatives of the same class (e.g. faces of the same person in various conditions and facial expressions) and maximizes distances between representatives of distinct classes (e.g. faces of different people). The trained network is then used to compare previously unseen query images with images stored in a database.

Face verification has received great attention in recent years and a range of effective techniques have been proposed for identifying faces in various conditions [8]–[10]. Convolutional neural networks are used to learn a mapping from a face image space to an Euclidean space of a smaller dimension.

The distance between the learned embedding vectors correspond to a face similarity measure [8]. Hierarchical non-linear transformations have been proposed to map face images into a feature space [7]. Deep CNN can also be trained to extract features on which a joint Bayesian metric enables the estimation of the likelihood that two images are of the same person [11]. FaceNet system [9] implements a deep CNN that projects face images into a compact Euclidean space where again the distance between projections relates to face similarity.

Siamese and Triplet networks are two popular network architectures for the verification task. A Siamese network accepts a pair of images as input and feed the images to two identical subnetworks that share the same weights [6]. In [12], Siamese and Triplet networks are used to learn image descriptors. Siamese networks have been used for tracking arbitrary objects in videos by comparing the initial appearance of an object with occurrences in subsequent frames [13]. Using Siamese architecture with a convolutional network in its core, strong results have been achieved in one-shot image recognition [14] where a prediction is made based only on a single example of a new class.

Inspired by the Siamese architecture, a Triplet network accepts a triplet of images which consists of an anchor image, a positive image (an example of the same class) and a negative image (an example of a different class). The loss used to optimize a Triplet network aims to minimize the distance between the anchor image and the positive image and maximize the distance between the anchor image and the negative image [9]. Triplet networks are used to learn a compact representation for the image retrieval task. The triplet loss is utilized to optimize a ranking objective [15]. Triplet networks have been successfully applied for learning a fine-grained similarity metric directly from images [16]. The task of person re-identification across cameras has been explored using a triplet of networks where a shared CNN model learns both global full-body and local body-parts features and a triplet loss function separates different classes with a predefined threshold [17].

In this paper we demonstrate that the Siamese and the Triplet network architectures are suitable for the task of learning the equivalence of patterns subjected to homographic transformations.

III. Learning equivalence classes

In this section, we compare two candidate neural network architectures, namely the Siamese and Triplet networks, to assess their suitability for learning equivalence modulo homographic transformations. Both architectures use a trainable subnetwork $f(P)$ that maps each image P to an embedding vector $f(P) \in R^d$ where d is a dimension chosen by the designer of the network. We also explore different convolutional architectures and loss functions for the shared subnetwork $f(P)$ on a dataset of artificially generated patterns.

A. Siamese network architecture

A Siamese network consists of two identical subnetworks that share the same weights followed by a distance calculation layer [6]. The input of a Siamese network is a pair of images (P_i, P_j) and a label y_{ij}. If the two images are deemed from the same equivalence classes, the pair is called *a positive pair*, and the target value is $y_{ij} = 0$. Whereas for a pair of images from different equivalence classes, the pair is called *a negative pair*, and the target value is $y_{ij} = 1$. The target value y_{ij} can be interpreted as the desired distance between the embedding vectors. The input images P_i, P_j are fed to the twin subnetworks to produce two vector representations $f(P_i), f(P_j)$ that are used to calculate a proxy distance. The training of a Siamese network is done on a collection of positive and negative pairs. Learning is performed by optimizing a contrastive loss function originally proposed in [18]. The aim is to minimize the distance between a pair of images from the same equivalence class while maximizing the distance between a pair of different equivalence classes. Let us define the function D between two input images P and S as the Euclidean distance between their embeddings in a feature space R^d. That is, $D(P, S) = \|f(P) - f(S)\|_2$. The contrastive loss function is defined as follows;

$$\mathcal{L}_c(P_i, P_j) = \begin{cases} \frac{1}{2} D(P_i, P_j)^2, & \text{if } y_{ij} = 0 \\ \frac{1}{2} \max(0, m - D(P_i, P_j))^2, & \text{if } y_{ij} = 1, \end{cases}$$

where (P_i, P_j) is an input pair, y_{ij} is the target value. The distance $D(P, S) = \|f(P) - f(S)\|_2$ helps pull closer together the embeddings $f(P)$ and $f(S)$ when P and S come from the same equivalence class. The margin $m > 0$ determines how far the embeddings of a negative pair are pushed apart. The loss functions are plotted in Figure 4.

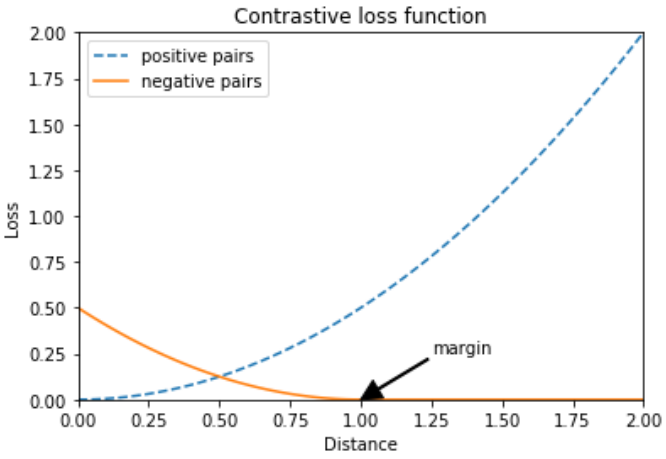

Fig. 4. Contrastive loss function $\mathcal{L}_c$ against the distance $D(P_i, P_j)$. The blue dashed line represents the loss function for positive pairs. The red solid line is the loss function applied for negative pairs. A negative pair contributes to the loss if its associated distance is greater than a predefined margin m. The contrastive loss function enforces a margin between the embeddings of the negative pairs.

The cross-entropy loss $\mathcal{L}_{ce}(P_i, P_j)$ can also be used for optimizing Siamese networks. This loss on a set of N training pair examples, is computed with the expression below

$$\frac{1}{N} \sum_{ij} y_{ij} \log D(P_i, P_j) + (1 - y_{ij}) \log(1 - D(P_i, P_j))$$

where the label is $y_{ij} = 0$ for a pair of images (P_i, P_j) from the same equivalence class and $y_{ij} = 1$ for a pair from different equivalence classes.

B. Triplet network architecture

Inspired by [12], [16], [19], the learning of an embedding can be approached by constructing an architecture that consists

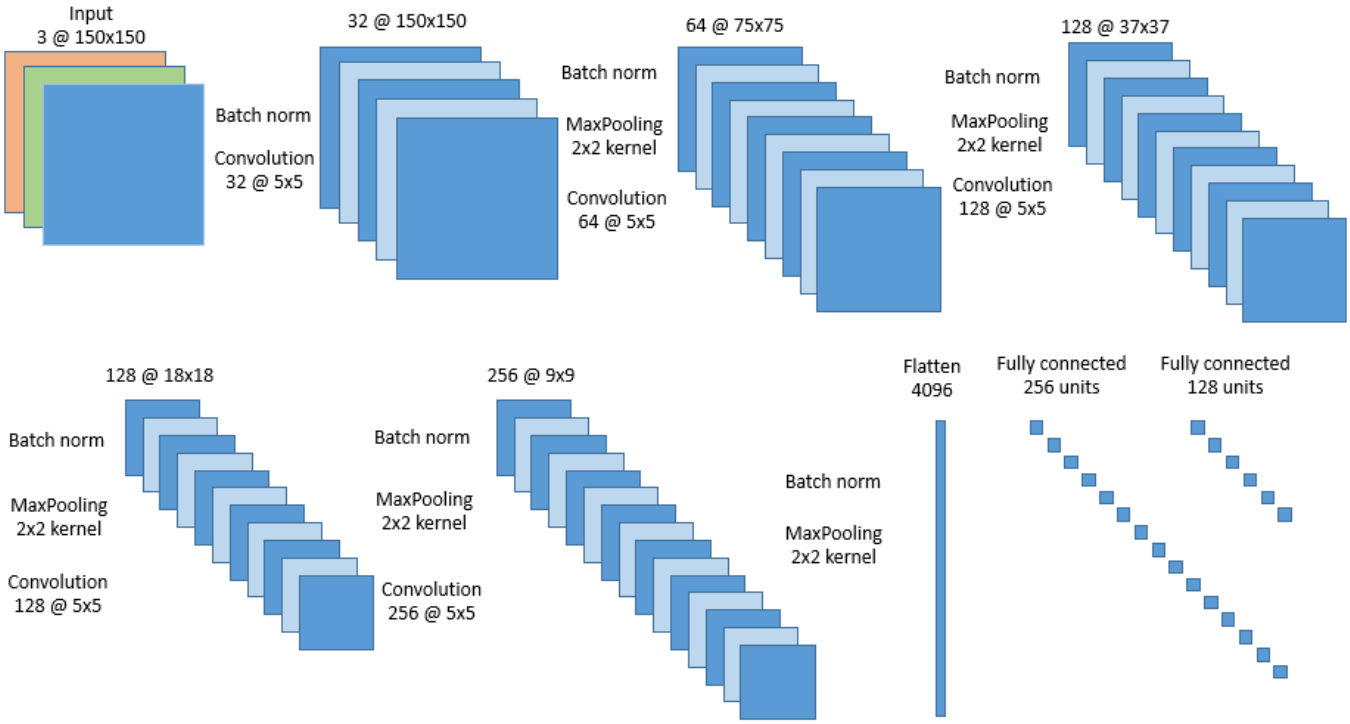

Fig. 5. The architecture of the shared subnetwork that produces an embedding vector f(P) for an input image P. Input image P has size 150*150 pixels and 3 colour channels. Five convolutional layers with 5×5 filters are followed by max pooling layers with kernels 2×2. The output of the last convolutional layer is flattened before being passed to two fully connected layers that output an embedding vector of dimension 128.

of three identical networks that share the same set of weights (see Figure 3) and accepts a triplet of images as its input: an arbitrary image, an image from the same equivalence class and an image from a different equivalence class. In our case each training example is a triplet of images (P^a, P^+, P^-) where P^a is a random pattern viewed from some 3D vantage point, P^+ is the same pattern viewed from a different vantage point and P^- is a distinct pattern viewed from another random vantage point.

Each image of the input triplet (P^a, P^+, P^-) is mapped to R^d by calculating the triplet $(f(P^a), f(P^+), f(P^-))$ and the Euclidean distances are calculated for the positive pair and the negative pair. That is, $D(P_i^a, P_i^+) = \|f(P^a) - f(P^+)\|_2$ and $D(P_i^+, P_i^-) = \|f(P^+) - f(P^-)\|_2$ respectively.

Training is performed by minimizing the hinge loss function. This process ensures that the distances between positive pairs are smaller than the distances between negative pairs with a chosen margin m [16]. In other words, we want to minimize the following loss function:

$$\mathcal{L}_h = \sum_{i=1}^{N} \max(0, m + D(P_i^a, P_i^+)^2 - D(P_i^+, P_i^-)^2)$$

where N is the number of training triplets.

The second loss function we experimented with is the mean squared error between the target label values and the predicted distances:

$$\mathcal{L}_m = \frac{1}{N} \sum_{i=1}^{N} (D(P_i^a, P_i^+) - y^+)^2 + (D(P_i^+, P_i^-) - y^-)^2$$

where y_i^+ is the target value for the positive pair (P_i, P_i^+) and y_i^- is the target value for the negative pair (P_i^+, P_i^-).

C. Shared subnetwork architecture

We chose a standard CNN architecture to generate the embedding $f(P)$ of the input pattern P motivated by the successes of this architecture in object recognition and verification [20]. The network architecture is a stack of normalization layers, convolutional layers and max-pooling layers as illustrated in Figure 5. The proposed CNN starts with a batch normalization layer which is also applied after each convolutional layer. The benefits of this type of layer include regularization capabilities and increased learning speed [21]. Batch normalization is applied to each training mini-batch. Use of batch normalization reduced the need of Dropout layers and increase generalization capabilities of the model.

Convolutional filters work as local feature detectors while max pooling layers make the model more robust to small translations [22]. A deep architecture with several layers of convolutional filters, normalization and max-pooling layers acts as a robust local feature extractor. Kernel weights matrices and bias vectors for convolutional layers are initialized with Xavier uniform initializer [23] where weights are drawn from

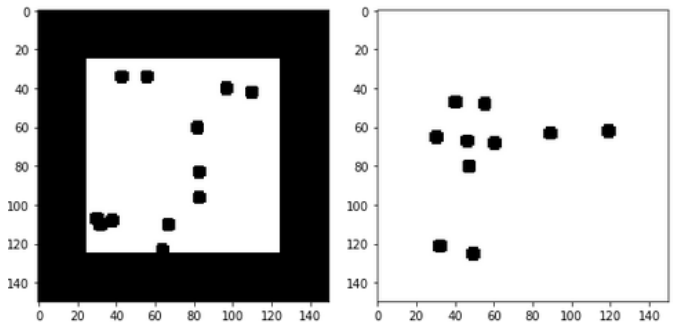

Fig. 6. Examples of generated patterns. For the first dataset we generated seed patterns as a set of black disks on a white square with a black background. For the second dataset, the seed patterns were a set of black disks on white background.

Pattern	P_k	P_k	P_s
Transform	T_l	T_p	T_q
Label	y = (0, 1)		

Pattern	P_k	P_s	P_k
Transform	T_l	T_p	T_q
Label	y = (1, 0)		

Fig. 7. Schema for triplets generation with seed patterns and transformations. Two types of triplets are generated with equivalent pattern at the second or at the third place for symmetry.

a uniform distribution with ranges calculated based on the number of input and output units. All layers in the shared CNN are followed by a rectified linear activation function.

The output of the last max pooling layer is flattened and two fully-connected layers compute non-linear transformations on local features to produce an embedding in a vector space of dimension d. The distance between representation vectors is calculated and passed to the sigmoid activation function $\frac{1}{1+e^{-x}}$ to restrict the output to the interval $[0, 1]$.

IV. Experiments

Our experiments were designed to test whether the Siamese and Triplet network architectures are able to recognize patterns subjected to large complex geometric transformations like homographies.

A. Design of experiments

1) Seed patterns and transformations: Consider a collection of seed patterns $\mathcal{P} = \{P_1, \ldots, P_n\}$ where each image P_i is a unique pattern of black disks on a white square as illustrated in Figure 6. Seed images represent a canonical view of a pattern from a camera placed in front of the pattern. When the camera moves, the projection of the pattern on the camera plane will be related to the canonical view by a homography.

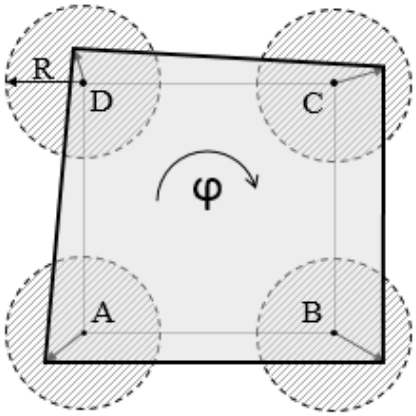

Fig. 8. A projective transformation is completely characterized by the mapping between the vertices of two quadrangles. In order to generate random homographies, we start with the four vertices A, B, C, D of a square, then randomly pick matching points in a neighbourhood of radius R to create a quadrangle. Finally, we rotate the quadrangle by arbitrary angle φ.

We consider also a set of random homographic transformations $\mathcal{T} = \{T_1, \ldots, T_m\}$. Each projective transformation T_j warps a seed pattern P_i to generate a new pattern. We say that two images are *equivalent* or *belong to the same equivalence class* if they are generated from the same seed pattern using projective transformations. We will call a pair of images from the same equivalence class a *positive pair* and a pair of images from different equivalence classes *a negative pair*.

2) Training pairs: Siamese networks require a collection of positive and negative pairs of images for training. A positive pair $(T_p(P_k), T_l(P_k))$ consists of two different $(p \neq l)$ projective transformations T_p, T_l of the same seed pattern P_k. A negative pair $(T_p(P_k), T_l(P_q))$ is generated from two distinct $(k \neq q)$ seed patterns P_k, P_q warped by two projective transformation. A positive pair is assigned the label $y = 0$ and a negative pair has the label $y = 1$.

3) Triplets of images: a Triplet model takes as input a triplet of images. To generate a triplet, we randomly choose a pattern P_k from a set of seed images $\mathcal{P}$ and apply a random projective transformation $T_l \in \mathcal{T}$ to this image to obtain an anchor image $P^a = T_l(P_k)$. Then we apply another projective transformation T_p to the same seed image P_k to generate an image $P^+ = T_p(P_k)$. The third image P^- of a triplet is generated by selecting a distinct $(s \neq k)$ seed image $P_s \in \mathcal{P}$ and applying a random projective transformation $T_q \in \mathcal{T}$. Images P^+ and P^- are swapped randomly in a triplet.

Each triplet is assigned a 2D-tuple label y that corresponds to the *discrete metric* between patterns in a positive pair (P^a, P^+) and a negative pair (P^+, P^-). Triplet (P^a, P^+, P^-) is assigned the label $y = (0, 1)$, and triplet (P^a, P^-, P^+) is given the label $y = (1, 0)$ as illustrated in Figure 7. The neural network tries to position the embeddings of equivalent pattern images in such a way that their distance is close to zero, whereas the embeddings of pattern images that are not equivalent are positioned such that their distance

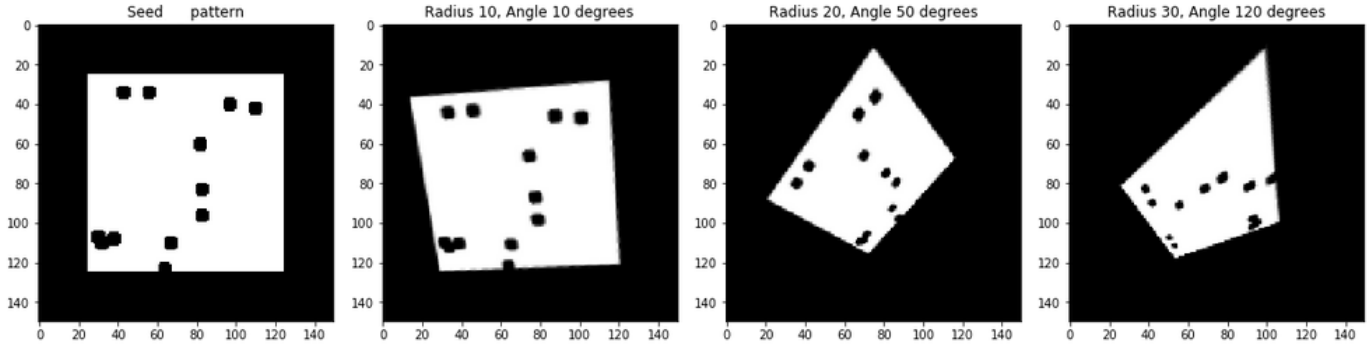

Fig. 9. Example of three different projective transformations with increasing deformations of a seed pattern. Small deformations are obtained with small radius and angle of rotation. Higher values of the radius and the rotation angle generate homographies that cause more dramatic distortions to the seed pattern.

is close to one.

For the sake of completeness, we tested whether the same network architecture is able to learn *transformation equivalence*. Our experiments show that the network is able to learn whether two images of patterns presented are views from the same view point or not. To carry out the experiment, a set of triplets (P^a, P^+, P^-) is generated to provide a model with information about the equivalence of transformations, where anchor image P^a is some seed pattern P_l warped by projective transformation T_k, element P^+ is a seed pattern P_p warped by the same projection T_k and element P^- is a distinct projection T_s $(k \neq s)$ applied to a pattern P_q.

B. Dataset generation

1) Seed patterns: Two separate datasets of seed patterns are generated for training (2000 patterns) and validation (200 patterns). Validation set is used for tuning architectural parameters of a model and monitoring performance during training to prevent overfitting. A separate test set (2000 patterns) is created to evaluate and compare architectures and it ensures that the performance evaluation is done on patterns that the network has not seen during training.

Each seed pattern has the following characteristics: image size 150×150 pixels, a central white square of size 100×100 pixels, 10 randomly placed black disks with a diameter of 5 pixels. A white square is placed at the center of the image with a black background and 50 pixels margin around it to allow display of a projected pattern. We also experimented with patterns with no visible border (a set of black disks on a white background like in Figure 6). The proposed network architectures were able to learn the equivalence of such patterns. Unsurprisingly, the neural networks could not learn transformation equivalence when the patterns had a complete white background.

We use the same number of disks in the patterns to prevent the network from learning to count the number of disks in a pattern to classify it. We also tested the proposed networks on datasets with both a constant and diverse number of disks. The results were not sensitive to the number of disks. Moreover, we experimented with different pattern shapes including triangles and rectangles and found out that, once trained, a network was able to distinguish between equivalent patterns independently of the shape of the dots.

2) Projective transformations: Separate training and validation sets of homographies were generated with 2000 and 200 transformations respectively. One of the fundamental theorems of projective transformations [24] states that a projective transformation T_i is completely characterized by the images $T_i(A)$, $T_i(B)$, $T_i(C)$, $T_i(D)$ of four points A, B, C, D no three of each are collinear.

In our experiment we selected the four vertices of a square $A = (0, 0)$, $B = (0, 100)$, $C = (100, 100)$, $D = (100, 0)$ and randomly moved them in a neighbourhood of radius R (see Figure 8). The moved points are then rotated around the center of the image by a random angle φ. Smaller values for radius R and angle φ generate homographic transformations that cause less deformations to seed patterns. These images are easier to classify for the networks. Increasing radius R generates projective transformations that apply heavier distortions to seed patterns and are more difficult for the networks to learn. Refer to Figure 9 for some examples.

C. Training

Siamese and Triplet network architectures are trained on sets of images that are generated by applying projective transformations to seed patterns. Training is performed on saved datasets of pairs and triplets that allows fair comparison of different architectures and loss functions on the same set of images. Generating good training examples that present a variety of positive and negative pairs to the network is important. However, there should be a balance between informative examples and swamping training with too challenging data.

Experiments show that proposed networks learn faster if they are trained in stages. That is, when they are first trained on easier examples and then on more difficult sets of images. We trained networks first on a set of examples with a radius $R = 15$ and an angle of rotation up to $\varphi = 90$, then increased the radius to $R = 25$ and the angle to $\varphi = 180$ both clockwise and anticlockwise. To ensure a fair comparison between training performance of Siamese and Triplet networks, the number of pairs and triplets is calculated as follows: the total number K of images is organized in $\frac{K}{2}$ pairs and $\frac{K}{3}$ triplets. Thus,

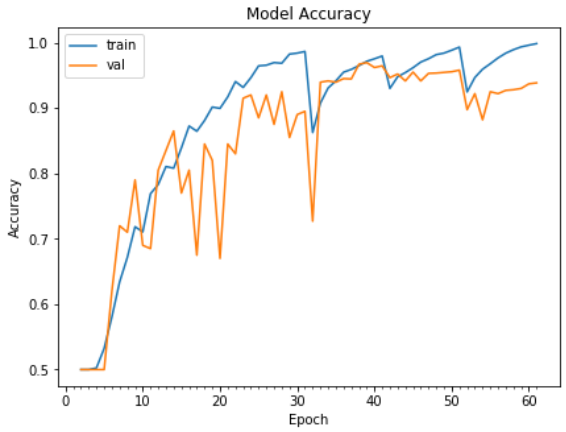

Fig. 10. Training plot of a Siamese network trained with a cross-entropy loss function. Model accuracy on training set of 24,000 pairs (blue line) and validation set of 2,400 pairs (red line) against 60 epochs. Dips in the training curve are due to the introduction of more challenging training examples.

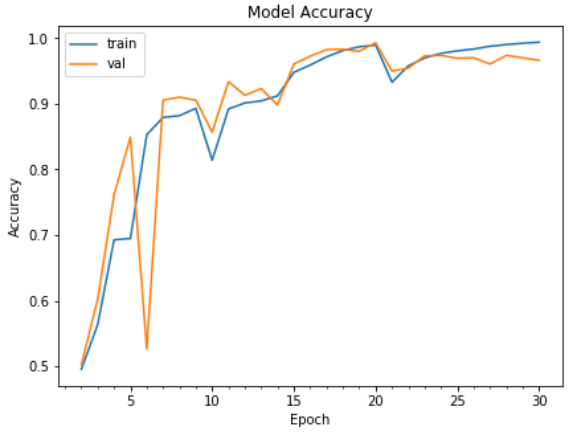

Fig. 11. Training plot of a Triplet network with a hinge loss function. Model accuracy on training set of 16,000 triplets (blue line) and validation set of 1,600 triplets (red line) over 30 epochs. A dip in a graph at epoch 21 corresponds to the introduction of more challenging training examples.

Siamese networks are trained in two stages with 24,000 pairs in each one and Triplet networks are trained in two stages with 16,000 triplets in each one. The same number of pairs and triplets are generated for learning transformation equivalence.

Both types of networks are trained from scratch using Adam, an algorithm for gradient-based optimization of objective function that works well in practice [25]. Fully connected layers use weight decay ($L2$ regularization) rate of 0.01 and bias vectors are initialized to zeros. Hinge loss and contrastive loss functions are used with a margin $m = 1$. The training plots showing training and validation accuracy against epochs for Siamese network optimized with a cross-entropy loss function and a Triplet network with hinge loss are presented in Figure 10 and Figure 11 respectively.

D. Results

Different network configurations have been explored. The most suitable architecture for a shared subnetwork is presented

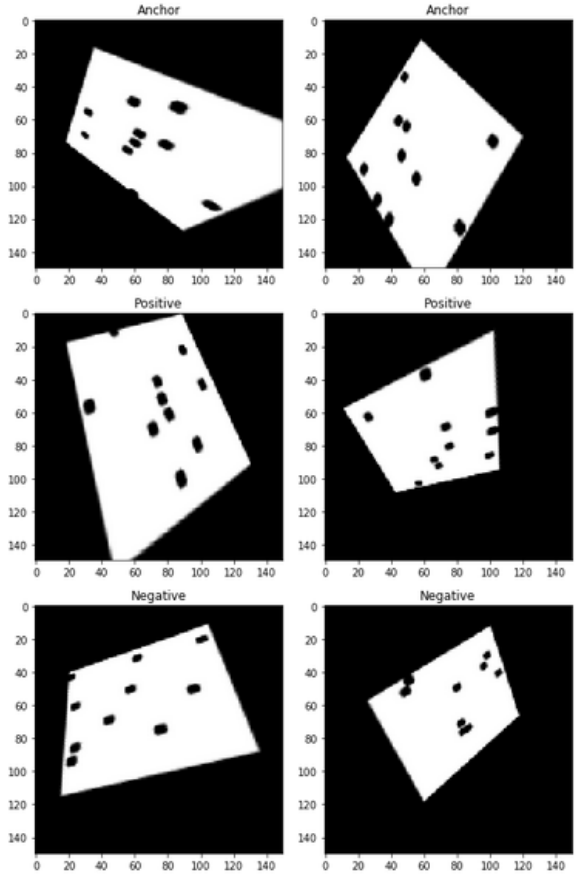

Fig. 12. A challenging correctly classified triplet is in the left column and incorrectly classified triplet is in the right column. Triplets are challenging because the angle of rotation between the anchor image and the positive image is over 90 degrees and patterns are warped significantly.

in Figure 5. The minimum number of convolutional layers that work out pattern equivalence has been determined empirically and is equal to five. Convolutional layers have 5×5 filters, max-pooling layers have kernel 2×2 and the embedding vector space has dimension $d = 128$.

We evaluated the performance of the two network architectures, Siamese and Triplet, trained with different loss functions on separate test pairs and triplets. Performance is measured using accuracy metric with a threshold of 0.5. Triplet (P^a, P^+, P^-) is identified correctly if a positive pair (P^a, P^+) has the distance less than 0.5 and a negative pair (P^+, P^-) has the distance greater or equal to 0.5. Predictions of Siamese network on positive and negative pairs are evaluated in the same way.

Architectures and loss functions comparison results are presented in Table I. The best result is shown by a Triplet network with a hinge loss function that demonstrates 97.14% accuracy while a Triplet network trained with a mean squared error loss function reaches only 89.69% on a test set of 10,000 triplets. Figure 12 shows challenging triplets from a test set where the angle of rotation between an anchor and a positive image is more than 90 degrees and patterns

TABLE I
ARCHITECTURE COMPARISON

Architecture	Loss function	# Test items	Accuracy
Siamese	Cross-entropy	15,000 pairs	93.06%
Siamese	Contrastive	15,000 pairs	84.17%
Triplet	Hinge	10,000 triplets	97.14%
Triplet	Mean squared error	10,000 triplets	89.69%

are warped significantly. The example in the left column is classified correctly while the example in the right column was predicted incorrectly by a Triplet network trained with a hinge loss function. Siamese architecture trained with a cross entropy loss function performs better (93.06% accuracy) than the one trained with a contrastive loss function (84.17% accuracy) compared on a test set of 15,000 pairs. For the task of learning transformation equivalence both architectures showed accuracy over 95%.

V. CONCLUSION

Our study clearly demonstrates that Siamese and Triplet networks are suitable network architectures for learning homographic equivalence. That is, we can train these neural networks to predict whether the patterns in two images can be matched by a projective transformation. This result opens new avenues for building automatic manta rays recognizers. In future work, we will use a foreground extraction algorithm to create a mask of the manta. Segmentation algorithms like GrabCut [26] can do this with minimal user interaction. A collection of masked color images of mantas will be fed to a Siamese or Triplet network for training. To determine whether a new image of a manta corresponds to a known manta, we will compute the distance of the embedding vector of the masked version of the query image to the embedding vectors of the known mantas. The embedding vectors of the known mantas can be cached to speed up the search. A practical system would return the 5 closest manta rays to the query image.

REFERENCES

[1] J. Ding, Y. Tang, W. Liu, Y. Huang, and K. Huang, "Tracking by local structural manifold learning in a new ssir particle filter," *Neurocomputing*, vol. 161, pp. 277 – 289, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215001769

[2] S. Reed, K. Sohn, Y. Zhang, and H. Lee, "Learning to disentangle factors of variation with manifold interaction," in *International Conference on Machine Learning*, 2014, pp. 1431–1439.

[3] C. Town, A. Marshall, and N. Sethasathien, "Manta matcher: automated photographic identification of manta rays using keypoint features," *Ecology and evolution*, vol. 3, no. 7, pp. 1902–1914, 2013.

[4] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[5] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *CoRR*, vol. abs/1506.02025, 2015. [Online]. Available: http://arxiv.org/abs/1506.02025

[6] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[7] J. Hu, J. Lu, and Y.-P. Tan, "Discriminative deep metric learning for face verification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1875–1882.

[8] O. M. Parkhi, A. Vedaldi, A. Zisserman *et al.*, "Deep face recognition." in *BMVC*, vol. 1, no. 3, 2015, p. 6.

[9] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[10] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, 2014, pp. 1988–1996.

[11] J.-C. Chen, V. M. Patel, and R. Chellappa, "Unconstrained face verification using deep cnn features," in *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. IEEE, 2016, pp. 1–9.

[12] B. Kumar, G. Carneiro, I. Reid *et al.*, "Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5385–5394.

[13] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 850–865.

[14] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, vol. 2, 2015.

[15] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," *arXiv preprint arXiv:1604.01325*, 2016.

[16] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[17] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person reidentification by multi-channel parts-based cnn with improved triplet loss function," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1335–1344.

[18] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2. IEEE, 2006, pp. 1735–1742.

[19] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," *arXiv preprint arXiv:1412.6622*, 2014.

[20] G. E. Hinton, "Learning multiple layers of representation," *Trends in cognitive sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[23] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[24] P. S. Modenov and A. Parkhomenko, *Projective Transformations: Geometric Transformations*. Academic Press, 2014.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": Interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015720