Using a Recurrent Kernel Learning Machine for Small-Sample Image Classification

M. Cudic
Department of Electrical and Computer Engineering
University of Florida
Gainsville, FL U.S.A

***Abstract*— Many machine learning algorithms, like Convolutional Neural Networks (CNNs), have excelled in image processing tasks; however, they have many practical limitations. For one, these systems require large datasets that accurately represent the sample distribution in order to optimize performance. Secondly, they are unable to transfer previously learned knowledge when evaluating data from slightly different sample distributions. To overcome these drawbacks, we propose a recursive kernel based approach for image processing using the Kernel Adaptive Autoregressive Moving Average algorithm (KAARMA). KAARMA minimizes the amount of training data required by using the Reproducing Kernel Hilbert Space to build inference into the system. The recursive nature of KAARMA additionally allows the system to better learn the spatial correlations in the images through one-shot or near one-shot learning. We demonstrate KAARMA's superiority for small-sample image classification using the JAFFE Face Dataset and the UCI hand written digit dataset.**

Keywords—Image Classification, Recurrent Systems, Kernel Machines

I. Introduction

As a result of technological advancements and increased accessibility of digital cameras, images have become a common method of recording and storing information from the natural world. Oncologists, for example, use microscopy images to characterize cancerous tissues to better diagnose patients [1]. Astronomers additionally use images from telescopes to discover new celestial objects and document the topology of the universe [2]. Although humans can easily analyze such images to extract useful information, there is a growing necessity to develop and integrate technologies that automate this analysis.

Such image processing has been a large focus of study for computer and engineering researchers and many algorithms have been proposed. Most notably, Convolutional Neural Networks (CNNs) have become the standard image processing algorithm used due to their success on a variety of image-based tasks [3], [4], [5]. However, CNNs and most alternative image processing algorithms have practical limitations as their success is heavily dependent on the size of the image dataset and the distribution of image samples.

Ideally all image processing algorithms should be trained on large enough datasets that accurately represent the distribution of samples in the dataspace and tested on datasets generated from the same distribution. These requirements are only attainable when large collaborative research efforts are put into place to collect ample amount of data for the given problem. Even then, there are some image-based tasks where the size of the dataset is limited by the rarity of the events being captured. Secondly, algorithms like CNNs are computationally intensive and require infrastructure that may be inaccessible to researchers outside of the computer science and engineering fields.

Therefore, we propose a kernel-based recursive learning machine for small-sample image classification using the Kernel Adaptive Autoregressive Moving Average algorithm (KAARMA) [6]. By segmenting the image into patches and sequentially mapping these patches to the Reproducing Kernel Hilbert Space, inference is successfully built into the KAARMA system. The recursive nature of KAARMA additionally allows the algorithm to learn adaptive convolutional filters dependent on the information seen previously in the image. As a result, KAARMA is successful in classifying images with low number of samples through one-shot learning or near one-shot learning and offers a robust light-weight alternative for image processing. This algorithm was tested on the JAFFE face dataset and the UCI handwritten dataset for various number of training samples and performed better than popular alternative algorithms tested – K-Nearest Neighbors (K-NN), Support Vector Machines (SVMs), and CNNs.

II. Related Works

A. Small-sample Image Classificaiton

In recent times, CNNs have been the most popular algorithm for image processing. CNNs, like Artificial Neural Networks (ANNs), rely on several feedforward layers of non-linear processing units to learn highly complex solutions. In addition to ANNs, these systems utilize convolutional layers at the beginning of every architecture and are thus more robust to spatial variations in images [3]. Although CNNs have been shown to have superior performance on a plethora of image-based tasks when properly trained, their performance is not guaranteed.

Every layer in a CNN consists of a large number adjustable paraments that must be optimized. For tasks that require highly complex solutions, this parameter total can easily reach up to 60 million [4]. Thus, it is imperative to partition the available data samples into a training dataset and validation dataset to prevent overfitting and properly optimize the system. There is no defined standard to how the available data should be split, but training and validation datasets should ideally contain enough samples to approximate the data distribution. For small-sample classification, this requirement will never be fully met and generating a validation dataset further reduces the number of samples used to train the system.

More so, CNNs only learn a mapping between the training samples and their respective classifications. If CNNs were to be tested on slightly different data distributions from the training dataset, then they would be unable to infer correct class memberships. This is especially problematic for small-sample classification where images are highly variable. Thus, CNNs are successful when large datasets are available and when computation time is not an issue. More practical solutions are necessary for when these objectives are unattainable.

Other simpler techniques that perform well with limited data have also been used for image classification. The K-Nearest Neighbor algorithm (K-NN), for example, is a supervised non-parametric algorithm that classifies new samples based on the k closest samples in the training dataset [7]. K-NNs rely only on the local information of the image in relation to previously seen samples and offers non-linear decision boundaries for multi-class classification. However, K-NNs have no control over the features used for classification as K-NNs use a distance metric on the training data to classify new samples. Thus, if no preprocessing of the image is used, K-NNs are highly susceptible to the dominate features expressed in the image even if these features are irrelevant.

Support Vector Machines (SVMs) are another popular supervised algorithm that have been successful on classification tasks [8], [9]. Using the kernel trick, SVMs can learn non-linear decision boundaries that maximize the margins used for classification. This results in the system learning to extract relevant features and produces more generalized decision boundaries unlike K-NNs. SVMs, however, are similar to K-NNs in that they do not contain enough complexity to extract resolved features when given a finite amount data. Most importantly, both SVMs and K-NNs treat the input data as static random variables and are not optimized to take advantage of the spatial correlations present throughout the entire image.

B. Recurrent Networks for Images

There has been a growing number of literature that uses recurrent based approaches for image processing as well. Although images have no inherent temporal information, images can be segmented into patches and ordered into a sequence. This allows recurrent systems to learn the global information of an image by first analyzing the local correlations between image patches. As the dynamics of the image patches change, then recurrent systems are best equipped in adjusting to the new set of statistics and successfully model the trajectory of the sequence.

Pixel RCNN, for example, uses this approach by training a recurrent convolutional neural network to reconstruct occluded images pixel by pixel [10]. By using a Long Shot-Term Memory architecture, Pixel RCNN can model the local information of the image, like texture or color, while understanding the global context of the pixel in question, like whether the pixel belongs to an animal's head or leg. Similarly, the RNN-CNN is another recurrent convolutional neural network optimized for image classification [11]. Unlike CNNs, this system is able to adaptively focus its attention to different parts of the image depending on other objects that have been identified. As a result, better contextual understanding is used for accurate multi-object classification.

Although recurrent systems have been successful in building some inference into image processing, these systems are extremely computational expensive for several reasons. Firstly, the gradient of the error must be taken with respect to time which requires multiple error calculations and weight updates. Secondly, most recurrent systems require seeing an image sequence repeatedly to reach a fully optimal solution. This drawback in conjunction with backpropagating the gradient through time exponentially increases the time complexity for training. If, however, image sequences can be inputted into a recursive system a limited number of times and approach one shot learning, then we can utilize the inference abilities of recurrent systems and negate expensive computations.

III. Methodology

A. The KAARMA Algorithm

The performance of the KAARMA algorithm on small-sample image classification is the focus of this paper. Thus, we begin by first summarizing the KAARMA algorithm in the context of Kernel Adaptive Filters (KAFs). For a more complete description of the KAARMA algorithm, please refer to Li and Pincipe [6].

KAFs have risen to prominence due to their superior performance over alternative adaptive filters [12]. By mapping the input vector $\mathbf{u}$ where $\mathbf{u} \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ into the RKHS, the input is projected into a potentially infinite dimensional feature space $\mathbb{F}$. Let us define this mapping from $\mathrm{U} \rightarrow \mathrm{F}$ as $\varphi(\mathbf{u})$. Linear optimization techniques can then be used to learn the non-linear associations between the $\mathbf{u}$ and the desired value $\mathrm{y} \in \mathbb{R}^{n_y}$. This parametric model can be thought of as a dot product in the RKHS as defined as:

$$\mathrm{y} \approx \hat{\mathrm{y}} = \hat{\mathrm{f}}(\mathbf{u}) = \mathbf{\Omega}^{\mathrm{T}}\varphi(\mathbf{u}) \quad (1)$$

where $\mathbf{\Omega}^{\mathrm{T}}$ is the weight vector in the RKHS. The function $\hat{f}(\boldsymbol{u})$ can also be universally approximated using the kernel trick and the Representors theorem as shown below:

$$\hat{\mathrm{f}}(\mathbf{u}) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \alpha_{\mathrm{i}} \kappa(\mathbf{u}_{\mathrm{i}}, \mathbf{u}) \quad (2)$$

where $\mathbf{u}_{\mathrm{i}}$ is a previously seen sample, $\mathbf{u}$ is the current sample in question, N is the number of samples in the training set, $\kappa(\cdot,\cdot)$ is a positive semi-definite Mercer kernel, and α_i is the kernel's associated weight that is calculated based on the instantaneous error at time i. The most commonly used kernel, and the kernel that will be used throughout this paper, is the gaussian kernel as defined below:

$$\kappa_{\varepsilon}(\mathbf{u}, \mathbf{u}') = e^{-\varepsilon\|\mathbf{u}-\mathbf{u}'\|^2} \quad (3)$$

where $\varepsilon > 0$ and dictates the size of the kernel. Gaussian kernels not only provide an infinite dimensional non-linear mapping of the input space, but they also build inference into KAFs by creating a continuous and statically significant representation of the samples in RKHS. Different levels of importance can also be given to neighboring samples and outliers by changing the size of ε. Thus, ε can be adjusted to capture and generalize the

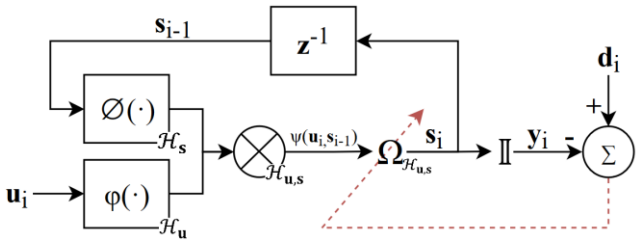

Figure 1: A summary of the KAARMA state-space model

statistics of the training data irrespective of the number of samples available. This capability is unique to gaussian kernel based methods and proves to be valuable when dealing with only a few samples.

As we can see in equation 1, only one set of weights are applied over the entire image dataset when using a KAF. Given that there are enough training samples present, then this is theoretically sufficient to approximate any arbitrarily complex solution. However, image classification is a non-trivial task and KAFs must train on many images to properly model the global information over an entire image. In order to successfully classify images with a low number of samples, more complex kernel methods are required that can extract resolved features and piece them together to reconstruct the global information present. This is what KAARMA sets out to do.

In its essence, KAARMA is a kernel based state-space model. Thus, let us first define the continuous non-linear state-transition function g(·, ·) and an observation function h(·) as:

$$\mathbf{x}_i = \mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i) \quad (4)$$
$$y_i = \mathbf{h}(x_i) \triangleq \mathbf{h} \circ \mathbf{g}(\mathbf{s}_{i-1}, \mathbf{u}_i) \quad (5)$$

where $\mathbf{x}_i \in \mathbb{R}^{n_x}$, the state vector is $\mathbf{s}_i \triangleq [\mathbf{x}_i, y_i]^T \in \mathbb{R}^{n_s}$, and ∘ is the composition operator between g(·, ·) and h(·). Since all computation will be done in the RKHS, both the input vector $\mathbf{u}_i$ and the state vector $\mathbf{s}_i$ are mapped into separate RKHSs defined as $\varphi(\mathbf{u}_i) \in \boldsymbol{\mathcal{H}}_{\boldsymbol{u}}$ and $\emptyset(\mathbf{s}_i) \in \boldsymbol{\mathcal{H}}_{\boldsymbol{s}}$ respectively. Given that the joint RKHS $\boldsymbol{\mathcal{H}}_{\boldsymbol{u,s}} \triangleq \boldsymbol{\mathcal{H}}_{\boldsymbol{u}} \otimes \boldsymbol{\mathcal{H}}_{\boldsymbol{s}}$ with ⊗ being a tensor-product kernel, the state space model can be represented by a set of weights as defined below:

$$\boldsymbol{\Omega}_{\mathcal{H}_{u,s}} \triangleq \begin{bmatrix} \mathbf{g}(\cdot,\cdot) \\ \mathbf{h}(\cdot) \end{bmatrix} \quad (6)$$

Thus, the kernel state-space model becomes:

$$\mathbf{s}_i = \boldsymbol{\Omega}^T_{\mathcal{H}_{u,s}} \psi(\mathbf{s}_{i-1}, \mathbf{u}_i) \quad (7)$$
$$y_i = \mathbb{I}\mathbf{s}_i \quad (8)$$

where $\psi(\boldsymbol{s}_{i-1}, \boldsymbol{u}_i) \triangleq \varphi(\mathbf{u}_i) \otimes \emptyset(\mathbf{s}_i) \in \boldsymbol{\mathcal{H}}_{\boldsymbol{u,s}}$ and $\mathbb{I} \triangleq \begin{bmatrix} \mathbf{0} & \mathbf{I}_{n_y} \end{bmatrix}$ is used to derive the desired signal y_i with I_{n_y} being an n_y by n_y identity matrix. Furthermore, the tensor product kernel can be defined as:

$$\begin{aligned} \langle \psi(\mathbf{s},\mathbf{u}), \psi(\mathbf{s}',\mathbf{u}') \rangle_{\mathcal{H}_{u,s}} &= \kappa_{u,s}(\mathbf{s},\mathbf{u},\mathbf{s}',\mathbf{u}') \\ &= (\kappa_u \otimes \kappa_s)(\mathbf{s},\mathbf{u},\mathbf{s}',\mathbf{u}) \\ &= \kappa_s(\mathbf{s},\mathbf{s}') \cdot \kappa_u(\mathbf{u},\mathbf{u}') \end{aligned} \quad (8)$$

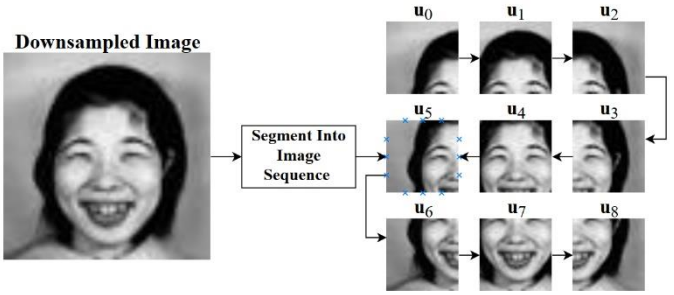

Figure 2: Illustration of how image samples were segmented and ordered. Note that there are overlapping pixels in each image patch. This was done to smoothen the trajectory learned by the KAARMA algorithm. Downsampling was also used to spread information around the image and reduce the time complexity of training. Patches of 16x16 and 4x4 were used for the JAFFE and UCI digit dataset respectively.

We can now use these equations to recursively train the KAARMA system by back-propagating the error through time. A summary of KAARMA is shown in figure 1.

Once the system is fully optimized, KAARMA operates as a recurrent network that uses information in the past to influence future mappings. Thus, as new samples of a sequence are introduced, the system adapts its states to learn the dynamics of a sequence while following a trajectory until it reaches a steady state solution.

Unlike KAFs where one set of weights are applied over a signal, KAARMA maps a given sample to different parts of the RKHS depending on the current state. This results to effectively having several unique non-linear projections that are optimized at analyzing the local information of the signal. KAARMA has been shown to be successful for modeling time-based signals like speech waveforms; however, we can further extend KAARMA to images if we feed several snapshots of an image as a sequence. In this way, we effectively learn the spatial information present in an image by convolving more resolved kernel based filters. Better features can now be extracted allowing for more complex solutions to be learned. Inference is also built into the system for optimal performance when training with a low number of samples.

Finally, because this is a kernel base method, training samples are used to construct the RKHS and thus memorized when introduced into the system. This results in drastically reducing the time complexity associated with recurrent systems as a sequence can be shown once, or close to once, if the trajectory of the sequence is stable. This results in near one-shot learning algorithm that is optimal for small-sample image classification.

B. The Experiment

The KAARMA algorithm was tested on two datasets, the Japanese Female Facial Expression (JAFFE) dataset and the University of California Irvine Pen-Based Recognition of Handwritten Digits (UCI digit) dataset [13], [14].

The JAFFE dataset consists of 213 images of 10 Japanese face models displaying 7 different emotions (neutral, happiness, sadness, surprise, fear, anger and disgust). Due to the limited number of data and the various emotions expressed, this dataset proves to be quite challenging for standard image processing algorithms. Systems must be able to both ignore dominate features associated with emotion and potentially identify faces displaying emotions that have never been seen before. All

images are 256x256 gray scale images that were scaled between 0 and 1 [13]. To spread some of the information around the image and reduce computation time, the images were also down sampled by 16 to create 64x64 images.

The UCI digit dataset, unlike the JAFFE dataset, contains high frequency information as the images are of hand written digits. There are roughly 11,000 500x500 gray scale images in total [14]. The images were further down sampled to be 8x8 images and all pixels were scaled between 0 and 1.

Because KAARMA requires a sequence to be inputted into the system, saccading patches mimicking convolutions were created as demonstrated in figure 2. While testing the KAARMA system, we did notice that the order of the sequence played a significant role in determining the system's performance. If patches were sequenced in non-intuitive ways that jumped randomly around the image, KAARMA was unable to learn the trajectories of the image. Thus, we settled for the ordering pattern seen in figure 2 as this ensured that all prior and subsequent patches shared some information to smoothen the sequence's trajectory. The expected output after every patch was a binary class membership vector with a 1 designating the class that the sample belonged to.

KAARMA additionally has multiple hyperparameters that need to be optimized, such as the step size, state kernel size, and input kernel size. These parameters were found through an extensive hyperparameter search and cross-validating on the testing set. Although the optimal values directly depend on the number of samples used, they roughly remain in the same neighborhood and were used as starting points for further experiment-specific hyperparameter tuning. For the JAFFE dataset when training with 50 images, a step size of .025, an input kernel size of .5 and a state kernel size of 2.0 were found to be optimal. Likewise, a step size of .05, an input kernel size of 1.0 and a state kernel size of 6.0 were used when training on 100 images from the UCI digit dataset. It should also be noted that only 2 state parameters were used for all experiments. This is another hyperparameter that can be adjusted to create more complex solutions, but 2 states were sufficient for our tasks. All training was also done using the mean squared error cost function.

To test the performance of KAARMA, we generated various amounts of training data and evaluated the classification performance on a fixed amount of testing data (100 JAFFE images and 500 UCI digit images) Since the systems were trained on a low number of samples, the system's performance on testing was highly variable. All results were thus reported as the average performance over 20 different randomly generated datasets.

The KAARMA algorithm was validated against several popular image classification algorithms – K-NNs, SVMs and CNNs. These systems were optimized as well based on previous literature and cross-validation. The parameter selections and architecture topologies are as follows. The K-NN algorithm used 3 nearest neighbors to evaluate new samples. In addition, the SVMs were trained using a radial basis kernel with a kernel coefficient of 0.0002 and .001 for the JAFFE dataset and the UCI digit dataset respectively. The SVMs were then trained using the "one-versus-rest" method for classification. Finally, the CNNs required the most tweaking as they have the most hypermeters to choose from. The typology for these systems were based off a

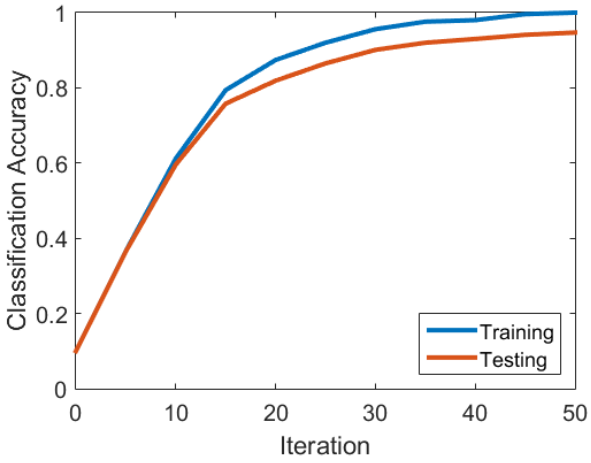

Figure 3: Classification accuracy on training and testing vs. the iteration in the training process for the JAFFE dataset. Only 5 images per individual (50 images in total) were used for training and the graph was obtained by taking the average results over 10 random generations of data.

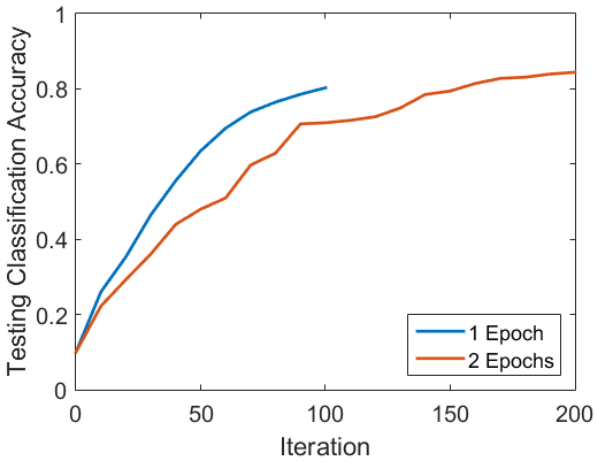

Figure 4: Testing classification accuracy on vs. the iteration in the training process for the UCI digit dataset for various epochs. Only 10 images per digit (100 images in total) were used for training and the graph was obtained by taking the average results over 10 random generations of data.

light-weight architecture used for MNIST classification provided by Keras. For the JAFFE dataset, the CNN used contained 8 3x3 convolutional layers in the first layer proceeded by 16 3x3 convolutional layers, a 2x2 max pooling, a 64 unit dense layer with a .25 dropout rate, and a final softmax layer. The architecture for the UCI digit dataset is much more compact due to the smaller image sizes. The CNN for this data set is structured as follows: 4 3x3 convolutional layers, 8 3x3 convolutional layers, 32-neuron dense layer, and a final softmax layer. All processing units were ReLu functions due to their computational efficiency and superior performance. Adadelta optimization was also used as the optimization technique and training was terminated through early stopping after 10 epochs of no improvements in classification accuracy in the validation dataset. Finally, the validation dataset was created by using 20% of all available data for training.

Number of Samples Per Individual	K-Nearest Neighbors	Support Vector Machines	Convolutional Neural Networks	KAARMA
3	87.30	90.75	70.94	92.5
5	92.60	94.29	85.05	94.89
10	97.90	95.70	94.50	97.80

Table 1: The average final testing classification accuracy for the JAFFE dataset using various algorithms (K-Nearest Neighbors, Support Vector Machines, Convolutional Neural Networks, and KAARMA). The classification accuracy averages were calculated over 20 separate runs and datasets were randomly generated for each run.

Number of Samples Per Digit	K-Nearest Neighbors	Support Vector Machines	Convolutional Neural Networks	KAARMA
5	88.41	84.8	68.59	81.13
10	89.33	85.18	71.95	85.4
20	93.84	86.52	86.36	89.37
40	96.56	87.49	90.59	91.83

Table 2: The average final testing classification accuracy for the UCI digit dataset using various algorithms (K-Nearest Neighbors, Support Vector Machines, Convolutional Neural Networks, and KAARMA). The classification accuracy averages were calculated over 20 separate runs and datasets were randomly generated for each run.

IV. Results

To first demonstrate the performance of KAARMA for image classification, the average learning curves for the training and testing accuracies on the JAFFE dataset can be seen in figure 3. As we can see, the system perfectly identifies all individuals in the training dataset after all 50 images are used. Due to the arbitrarily complex solutions that kernel based methods can learn, there is a high risk of overfitting to the training data entirely. However, as we can see by the testing performance, the system is able to generalize well if the proper kernel size is used. More so, the plateau in performance on the testing dataset after all 50 training images demonstrates KAARMA's ability for one-shot learning. Not only does one-shot learning tremendously save computational complexity when training recurrent systems, but it also offers a clear stopping point for training. This eliminates any need for a validation dataset and allows for more samples to be used for training to create more generalized solutions.

One-shot learning, however, may be insufficient to train KAARMA to learn very complex decision boundaries. This can be seen in figure 4 which shows the testing learning curves on the UCI digit dataset when KAARMA is trained over 1 and 2 epochs. As we can see, KAARMA is unable to obtain high classification accuracies with 1 epoch. This is because large step sizes are required to ensure convergence over 1 epoch which limits the precision of the decision boundaries learned. By increasing the number of epochs used for training, we decrease the step size and learn an RKHS mapping that better represents all the training data. Although showing every image to KAARMA twice is not considered one shot learning, it still close to one shot learning and greatly reduces the training that is necessary.

The final average testing performances over 20 simulations of data for the JAFFE dataset and the UCI digit dataset are shown in Table 1 and Table 2 respectively. We can see that KAARMA performs best on the JAFFE dataset for the reasons discussed in the paper. The various emotions expressed in the JAFFE images provide dominate and extraneous features that algorithms like SVMs and KNNs have trouble ignoring. K-NNs, however, perform better than KAARMA on the UCI digit dataset with KAARMA performing second best overall. This the neighborhood information of a UCI image is sufficient to determine its class. If this dataset is less controlled and contained more variability in handwriting, then we should expect KAARMA to outperform K-NNs. Finally, KAARMA's superior

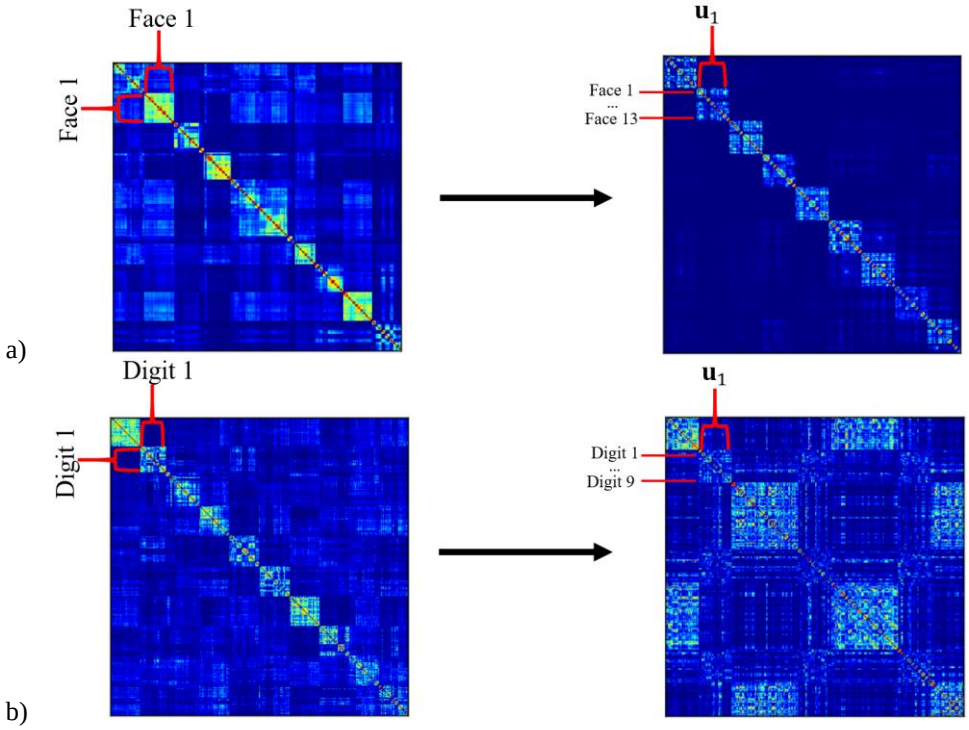

Figure 5: a) and b) show the gram matrices for the JAFFE and UCI digit dataset respectively. The Gram matrices on the left are based on the entire image and organized by class. The Gram matrices on the right are based on the patches in the image sequence. They are additionally organized by the patch's order in the sequence and then by the class the sequence belongs to.

performance to CNNs demonstrate the robustness of the algorithm for low frequency and high frequency images.

We can additionally analyze why KAARMA performs well on these datasets by plotting the gram matrices for the images and image sequences as shown in Figure 4a and 4b. In figure 4a specifically, we directly see the benefit of segmenting the image into a sequence as this build a sparse RKHS with more internal structure. This then allows the KAARMA algorithm to better extract local information in the image by using more resolved filters. Figure 4b also elucidates why multiple epochs must be used to train KAARMA for the UCI image dataset. Some parts of the image sequence overlap in the RKHS irrespective of the class. This creates class confusion for the KAARMA system and more precise fine tuning is required to determine the optimal decision boundary. These overlapping features can be eliminated from the image sequence entirely use pre-processing. However, this is out of the scope for this paper as we wanted to evaluate KAARMA's ability to analyze raw images.

V. Conclusion

KAARMA is a recurrent kernel based system that was originally designed to model temporal information. However, we demonstrate that KAARMA can be further extended to learn spatial correlations along an image for small-sample image classification.

KAARMA offers many advantages over alternative algorithms. By mapping parts of the sequence into the RKHS, KAARMA is able to learn infinitely complex solutions and give statistical significance to every sample seen. This allows KAARMA to better infer classifications for images that may not be well represented in the training set. KAARMA can also be thought of as an adaptive convolutional network. Patches containing local information are inputted into the system as if KAARMA were saccading through the image. However, the system adapts its filters after every patch in order to model the global trajectory of the sequence. This allows KAARMA to extract more resolved features than alternative kernel based algorithms and therefore develop more complex decision boundaries. Finally, every sample that is introduced to KAARMA is used to construct the RKHS. Due to the scarcity of data for small-sample classification, this results in KAARMA memorizing every training sample and operating as a one-shot, or near one-shot, learning algorithm.

We demonstrate the superiority of the KAARMA for small-sample image classification using the JAFFE dataset and the UCI pen hand written datasets. Not only did KAARMA perform better than most alternative algorithms, but it proved to be robust to low-frequency and high frequency images.

More research should be done to test the limits of the KAARMA algorithm for different types of images. It also might be of interest to see how KAARMA performs on other image-based tasks, such as image segmentation. Finally, we believe that KAARMA can be further optimized using different cost functions. MSE was used to train KAARMA due to its simplicity and previous literature. However, MSE only captures the second moment of the error and is heavily prone to outliers. Entropy based approaches could potentially improve KAARMA for one-shot learning by smoothening the learning curve and utilizing all training samples to its maximum effect.

VI. References

[1] Charron, M., Beyer, T., Bohnen, N.N., Kinahan, P.E., Dachille, M., Jerin, J., Nutt, R., Meltzer, C.C., Villemagne, V. and Townsend, D.W., 2000. Image analysis in patients with cancer studied with a combined PET and CT scanner. Clinical nuclear medicine, 25(11), pp.905-910.

[2] Böker, T., Laine, S., van der Marel, R.P., Sarzi, M., Rix, H.W., Ho, L.C. and Shields, J.C., 2002. A Hubble Space Telescope census of nuclear star clusters in late-type spiral galaxies. I. Observations and image analysis. The Astronomical Journal, 123(3), p.1389.

[3] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[4] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[5] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[6] Li, K. and Príncipe, J.C., 2016. The kernel adaptive autoregressive-moving-average algorithm. IEEE transactions on neural networks and learning systems, 27(2), pp.334-346.

[7] Boiman, O., Shechtman, E. and Irani, M., 2008, June. In defense of nearest-neighbor based image classification. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on (pp. 1-8). IEEE.

[8] Foody, G.M. and Mathur, A., 2006. The use of small training sets containing mixed pixels for accurate hard image classification: Training on mixed spectral responses for classification by a SVM. Remote Sensing of Environment, 103(2), pp.179-189.

[9] Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L. and Huang, T., 2011, June. Large-scale image classification: fast feature extraction and svm training. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on (pp. 1689-1696). IEEE.

[10] Oord, A.V.D., Kalchbrenner, N. and Kavukcuoglu, K., 2016. Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759.

[11] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C. and Xu, W., 2016, June. Cnn-rnn: A unified framework for multi-label image classification. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on (pp. 2285-2294). IEEE.

[12] Liu, W., Park, I. and Principe, J.C., 2009. An information theoretic approach of designing sparse kernel adaptive filters. IEEE Transactions on Neural Networks, 20(12), pp.1950-1961.

[13] Lyons, M., Akamatsu, S., Kamachi, M. and Gyoba, J., 1998, April. Coding facial expressions with gabor wavelets. In Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on (pp. 200-205). IEEE.

[14] Alpaydin, E. and Alimoglu, F., 1998. UCI pen-based recognition of handwritten digits data set.