Supplementary: "Incremental Learning for Animal Pose Estimation using RBF k-DPP"

Gaurav Kumar Nayak*[1]
gauravnayak@iisc.ac.in
Het Shah*[12]
divhet163@gmail.com
Anirban Chakraborty[1]
anirban@iisc.ac.in

[1] Indian Institute of Science
Bangalore, India

[2] BITS Pilani
Goa, India

1 Detailed Diagrammatic View of our Approach

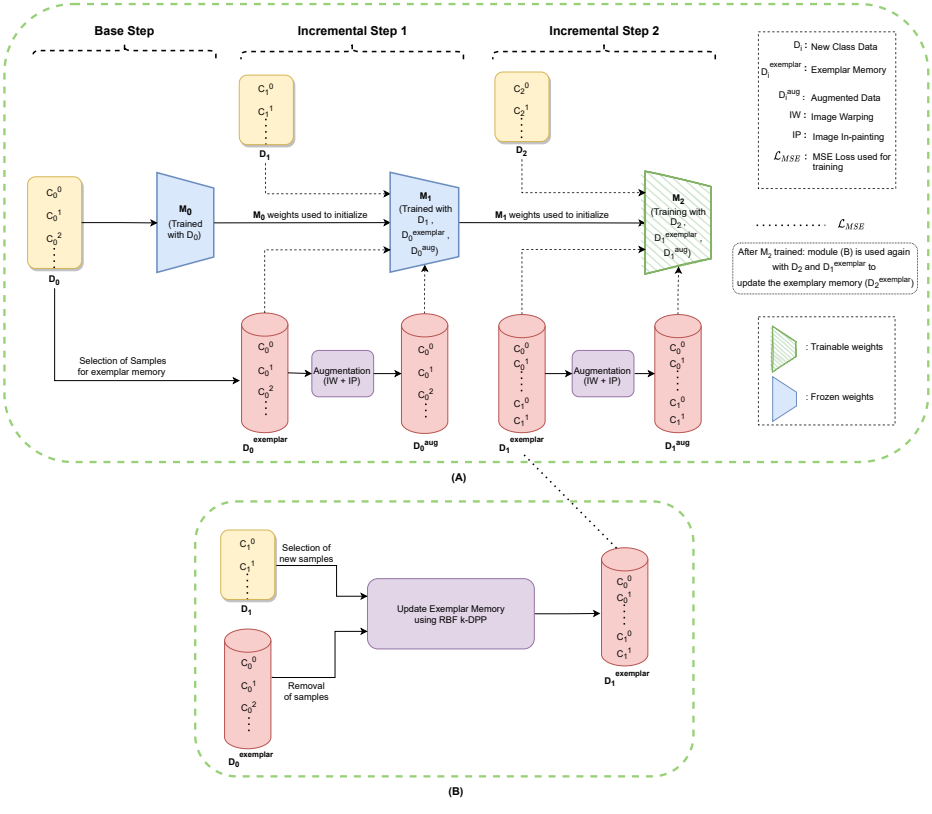


Figure 1: (A) Complete overview of our approach. We depict training process at the second incremental step. The model $\boldsymbol{M_2}$ is initialized with the weights of $\boldsymbol{M_1}$ and its weights are kept trainable. Augmentation module is explained in detail in Figure 2. (B) To update the exemplar memory, we remove samples from $D_0^{exemplar}$, and select new samples from new class data, D_1. Thus, $D_1^{exemplar}$ is obtained after this update step which is used along with D_1^{aug} and D_2 to train $\boldsymbol{M_2}$.

* Equal contribution.

2 Image Warping Augmentation

Figure 2, describes the image warping and in-painting operations which are used as an augmentation technique and are applied on the input images of animal pose data. The limbs of horse in Figure 2 (c) are rotated by a small angle using Thin Plate Splines (TPS Module). We then perform image in-painting to fill in the pixels with no values.

The proposed augmentation when applied to samples from the exemplar memory helps in creating diverse poses. This eventually mitigates class imbalance between new class data and exemplar memory during the training at each incremental step. Figure 3 demonstrates samples generated using this augmentation technique.

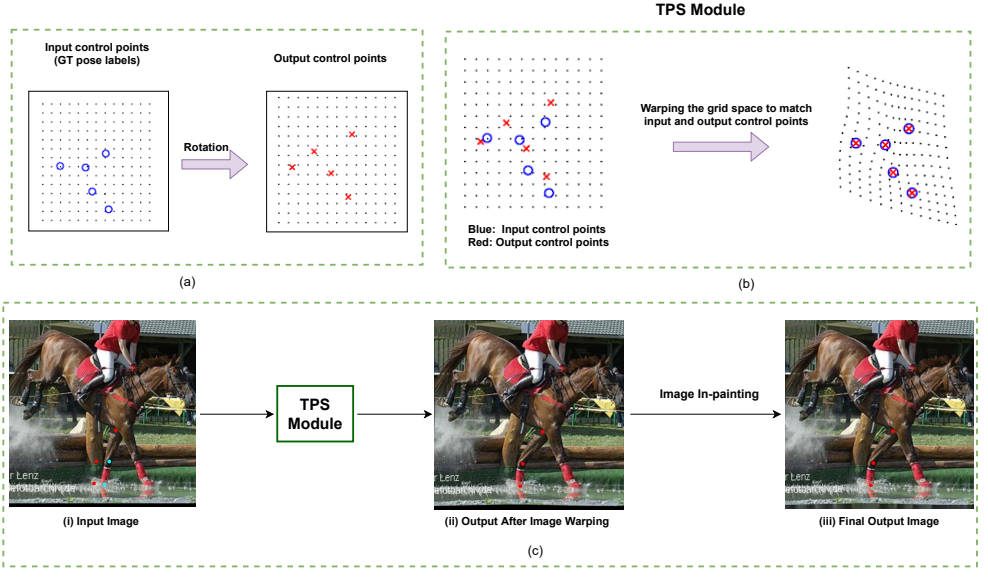


Figure 2: Overview of Image warping and in-painting augmentation. (a) Ground truth pose labels are taken as Input control points, we rotate these keypoints by a small angle under skeletal constraints for each animal to get the Output control points, (b) TPS warps the image grid space to match the input and output control points, (c) We show steps of the proposed augmentation technique (i) For demonstration we only rotate the left frontal leg of the horse, cyan dots represent the original keypoints and red dots represent the rotated keypoints, (ii) After TPS warping, we get the rotated left front limb, (iii) Final output image is generated after applying image in-painting.

(a) Before Augmentation (b) After Augmentation

Figure 3: The images in (a) are before augmentation, images in (b) are after image warping and in-painting augmentation applied to respective images in (a).

3 Proof for "RBF is positive semi-definite"

The Radial Basis Function, $F_{RBF}(x,y)$, when x, y are real numbers, is given as

$$F_{RBF}(x,y) = exp(-\frac{\|x-y\|^2}{2\sigma^2}) \tag{1}$$

Without the loss of generality, we can assume that $\sigma = 1$. We can further, write $F_{RBF}(x,y) = h(x-y)$, where

$$\begin{aligned} h(t) &= exp(-\frac{t^2}{2}) \\ &= exp(0(it) + 1/2(1)^2(it)^2), \end{aligned} \tag{2}$$

here i denotes iota.

Let's assume that Z is a random variable where $Z \sim G(0,1)$, G is the Gaussian distribution. and we know that moment generating function for Gaussian distribution can be written as,

$$M_Z(x) = exp(\mu x + (\sigma^2 x^2)/2) \tag{3}$$

From Eq. 2 and Eq. 3 we get,

$$\begin{aligned} h(t) &= M_Z(it) \\ &= E[e^{itZ}] \end{aligned} \tag{4}$$

Any $m \times m$ matrix A is positive semi-definite if,

$$\boldsymbol{v}^T A \boldsymbol{v} \geq 0, \quad \forall \mathbf{v} \in \mathbb{R}^m \tag{5}$$

Thus, for real numbers $x_1, x_2, ...x_n$ and $a_1, a_2, ...a_n$, a quadratic form of F_{RBF}, would be

$$\sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k F_{RBF}(x_j, x_k) = \sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k h(x_j - x_k) \tag{6}$$

Thus it is sufficient to prove that,

$$\sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k h(x_j - x_k) \geq 0 \tag{7}$$

Therefore,

$$\begin{aligned} \sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k h(x_j - x_k) &= \sum_{j=1}^{n}\sum_{k=1}^{n} a_j a_k E[e^{i(x_j - x_k)Z}] \\ &= E[\sum_{j=1}^{n}\sum_{k=1}^{n} a_j e^{ix_j Z} a_k e^{-ix_k Z}] \\ &= E[|\sum_{j=1}^{n} a_j e^{ix_j Z}|^2] \geq 0, \end{aligned} \tag{8}$$

This entails that the Radial Basis Function F_{RBF} is positive semi-definite, and thus a kernel. Without loss of generality, the same proof can be extended when x and y are vectors.

4 Baselines

Adapted-iCaRL: At each incremental step i, we first concatenate the exemplar memory and the new class(es) training data,

$$\mathcal{D} \leftarrow \mathcal{D}_{exemplar} \bigcup \mathcal{D}_{new} \tag{9}$$

where $\mathcal{D}_{exemplar}$ consists of old classes, i.e. $C_0 \bigcup C_1 ... \bigcup C_{i-1}$ and $\mathcal{D}_{new}$ consists of the new classes which the model has to learn, i.e. C_i. Thus, $\mathcal{D}$ consists samples from the entire set, $C_0 \bigcup C_1 ... \bigcup C_i$

We use the following loss function for the training procedure.

$$\mathcal{L} = \alpha * \Sigma_{(x_j,y_j) \in D_{exemplar}} MSE(M_{i-1}(x_j), M_i(x_j)) + (1-\alpha) * \Sigma_{(x_k,y_k) \in D_{new}} MSE(M_i(x_k), y_k) \tag{10}$$

where α is an hyperparameter in the loss function, we set $\alpha = 0.5$ for all the experiments.

We use the herding strategy to sample the exemplar memory, similar to what was used in the original implementation of iCaRL.

Adapted-EEIL: At each incremental step i, we first train the model similar to Adapted-iCaRL's training step, which comprises of the data from exemplar memory and the new class(es) data. After this step, we perform additional balanced finetuning, as done in EEIL [2]. This balanced finetuning is performed on a training subset containing equal number of samples for each class. This is done by sampling n samples for the new class data, $\mathcal{D}'_{new}$, by using the Herding strategy. We use the model in the training step performed before balanced finetuning step, M'_i, as the teacher network for the Knowledge Distillation loss term. The weights of this model are frozen and it's predicted heatmaps are used for loss calculations. The updated loss term used for this step is given as,

$$\mathcal{L}_{old} = \alpha * \Sigma_{(x_j,y_j) \in D_{exemplar}} MSE(M'_i(x_j), M_i(x_j)) + (1-\alpha) * \Sigma_{(x_j,y_j) \in D_{exemplar}} MSE(M_i(x_j), y_k) \tag{11}$$

$$\mathcal{L}_{new} = \alpha * \Sigma_{(x_k,y_k) \in D'_{new}} MSE(M'_i(x_k), M_i(x_k)) + (1-\alpha) * \Sigma_{(x_k,y_k) \in D'_{new}} MSE(M_i(x_k), y_k) \tag{12}$$

$$\mathcal{L} = \mathcal{L}_{old} + \mathcal{L}_{new} \tag{13}$$

where α is an hyperparameter in the loss function, we set $\alpha = 0.5$ for all the experiments.

After the balanced finetuning step, we update the exemplar memory by removing samples from the exemplar memory, and adding samples for the new class.

5 Hyperparameter details

A list of hyperparameters used in this work is provided in Table 1.

Hyperparameters	Value
Input Image Size	512x512
Output Heatmap Size	128x128
Base Model Training Epochs	30
Base Model Optimizer	Adam
Base Model Learning Rate	0.0001
Batch size for training Base Model	13
Incremental Model Training Epochs	20
Incremental Model Optimizer	Adam
Incremental Model Learning Rate	0.0001
Batch Size for training Incremental Model	5
Balanced Finetuning Training Epochs (Adapted EEIL)	5
Balanced Finetuning Optimizer (Adapted EEIL)	Adam
Balanced Finetuning Learning Rate (Adapted EEIL)	0.00001
Batch Size for Balanced Finetuning (Adapted EEIL and iCaRL)	5
α (Adapted EEIL and iCaRL)	0.5

Table 1: Hyperparameter details used in this work.

6 Additional Experiments

We perform additional experiments on a different setup, i.e. growing memory case, where the number of samples per class remains fixed. We perform experiments in such a scenario to further demonstrate the efficacy of our approach. We restrict ourselves to 10% of samples for each class. The results are shown in the Table 2. We observe that our proposed DPP w/ clustering performs significantly better than the Random and Herding strategy baselines. Further, our proposed RBF k-DPP ($\gamma = 50$) improves the performance on DPP w/ clustering.

Approach	Incremental Steps	
	1	2
Oracle	0.8478	0.8457
Herding	0.7513	0.6626
Random	0.7982	0.7259
DPP w/ clustering (**Ours**)	0.8245	0.7613
RBF k-DPP ($\gamma = 50$) (**Ours**)	**0.8291**	**0.7799**

Table 2: PcK@0.05 results for Growing memory (fixed number of samples per class), base classes are {'cat', 'dog', 'cow'} and 'horse' and 'sheep' are added at the incremental steps. 10% of **each** class data is added to the memory.

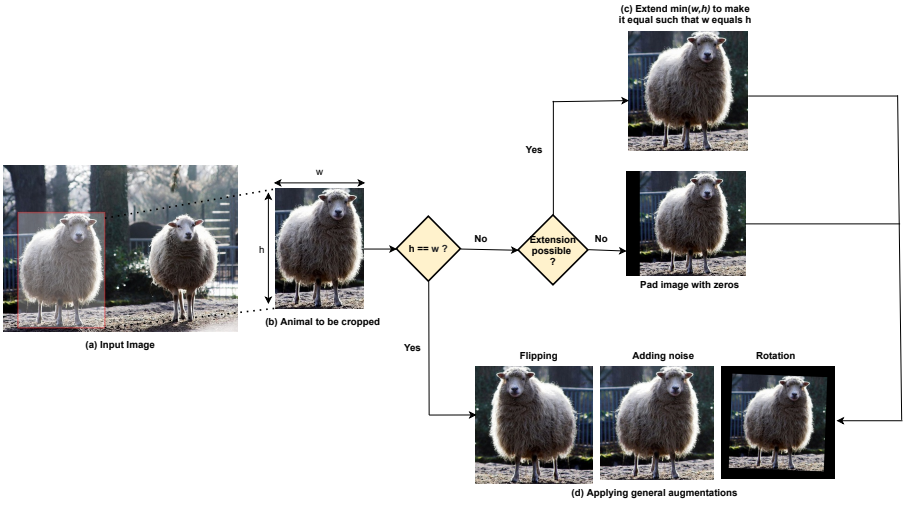


Figure 4: Details of image data pre-processing pipeline. (a) Input Image, (b) we select only the object of interest (animals in our case) as an input to the model, (c) if the selected object of interest has rectangular bounding box we directly extend it or pad zeros to make it a square shaped box and then crop it out, (d) we finally perform various augmentations like flipping, adding noise, rotation and combinations of these.

7 Dataset Pre-Processing

Given an image, which may contain multiple animals in it, we crop out each animal using the ground-truth bounding boxes provided with the dataset. These bounding boxes can be rectangular in shape. In order to explicitly convert them to a square bounding box, we extend the smaller side of the rectangle to make it a square. However there can be an edge case, where while extending the bounding box we may exceed the image boundary region. To overcome this problem, we pad the image with zeros and then extend the smaller side of rectangle to make it a square. After getting a square shaped crop of the animal image, we resize it to a fixed image size across all the input images. We further augment the data by *Flipping, adding Gaussian Noise, Rotating the images by a small random angle*, and a combination of these augmentation strategies. These augmentations helps to increase the training set size and act as a regularizer to reduce overfitting in our pose estimation model. An overview of the data pre-processing pipeline is provided in Figure 4.

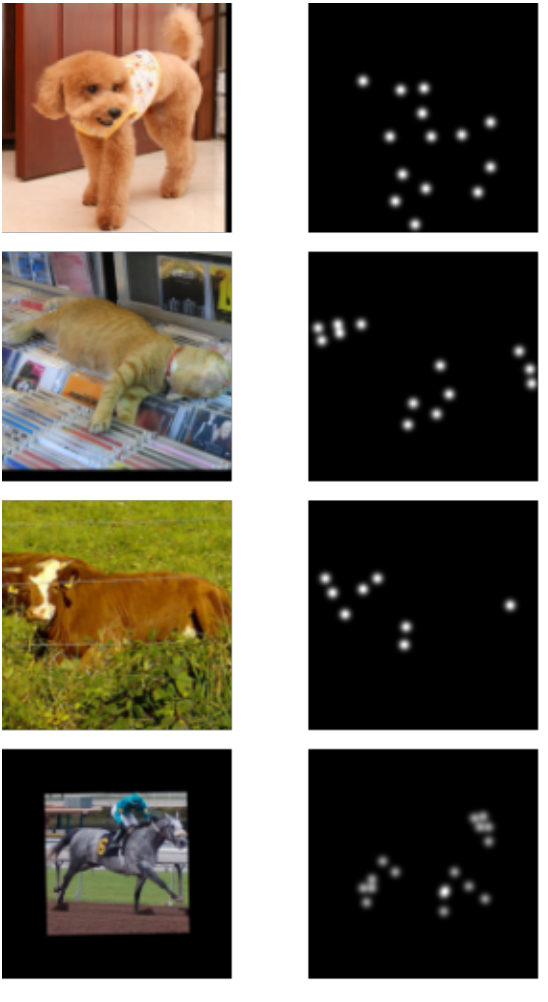
8 Visualization

We provide visualization of some pre-processed samples and their ground-truth keypoints labelled in Figure 5. Red points in the figure show the ground-truth keypoint label. There are total 17 keypoints labelled for each image, namely two ‘Eyes’, two ‘Earbases’, ‘Nose’, four ‘Elbows’, four ‘Knees’ and four ‘Paws’.

As explained in the main draft, the ground-truth keypoints are converted to Gaussian heatmaps to assist in training the pose estimation model. To generate the Gaussian heatmaps we center each keypoint on the spatial coordinates of the keypoint. A visualization of the input image and the **summation** of the heatmaps for all the keypoints is provided in the Figure 6.

(a) (b) (c)

Figure 5: Visualization of animals and their keypoints labelled.



(a) (b)

Figure 6: The images in (a) are the input images for various classes of animals from Animal-Pose Dataset [0]. The images in (b) are **Summation** of all the 17 heatmaps of the keypoints

References

[1] J. Cao, H. Tang, H. Fang, X. Shen, Y. Tai, and C. Lu. Cross-domain adaptation for animal pose estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9497–9506, Los Alamitos, CA, USA, 2019. IEEE Computer Society. doi: 10.1109/ICCV.2019.00959. URL https://doi.ieeecomputersociety.org/10.1109/ICCV.2019.00959.

[2] Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 241–257, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01258-8.