2022 IEEE International Conference on
Development and Learning (ICDL)

Feedback-Driven Incremental Imitation Learning Using Sequential VAE

1st Gabriela Sejnova
Czech Institute of Informatics, Robotics, and Cybernetics
Czech Technical University
Prague, Czech Republic
gabriela.sejnova@cvut.cz

2nd Karla Stepanova
Czech Institute of Informatics, Robotics, and Cybernetics
Czech Technical University
Prague, Czech Republic
karla.stepanova@cvut.cz

***Abstract*—Variational Autoencoders (VAEs) have attracted a lot of attention from the machine learning community in recent years. The usage of VAEs in learning by demonstration and robotics is still very restricted due to the need for effective learning from only a few examples and due to the difficult evaluation of the reconstruction quality. In this paper, we utilize the current models of conditional variational autoencoders for the purpose of teaching a robot simple actions from demonstration in an incremental fashion. We in detail evaluate various training approaches and define parameters that are important for enabling high-quality samples and reconstructions. The quality of the generated samples in different stages of learning is evaluated both quantitatively and qualitatively on the humanoid robot Pepper. We show that the robot can reach a reasonable quality of generated actions already after 20 observed samples.**

***Index Terms*—imitation learning, VAE, humanoid robots, incremental learning**

I. Introduction

During the past years, Variational Autoencoders (VAEs) [1] have been the subject of intense research in many different application areas. Among the most common are image reconstruction and generation [2], object classification [3] and recently also action recognition and reproduction [4].

However, implementing VAEs for tasks such as learning by demonstration in real-world robotic scenarios is limited as there are usually only a few training examples available. Moreover, such models are expected to be able to learn new tasks incrementally; e.g., they need to have an inbuilt mechanism that would avoid catastrophic forgetting. Another problem is how to evaluate and compare the quality of the generated samples other than based on empirical observation.

In this paper, we explore and demonstrate whether and how can the recent VAE models be implemented for few-shot learning of robotic actions based on human demonstrations. We choose the humanoid robot Pepper for action reproduction as its arms (specifically shoulders and elbows) can easily mimic human motion.

In our experiments, we first demonstrate which parameters of the VAE network have the largest impact on the quantitative and qualitative results. To evaluate the qualitative results, we propose two metrics based on the statistical features of the original and reconstructed data. Next, we show how many samples are needed for a VAE model to learn a single task only. Finally, we compare two possible approaches toward incremental learning of multiple tasks by a single model, making use of the generative nature of VAEs.

The code and the dataset are available at https://github.com/imitrob/robot_action_imitation_VAE.git

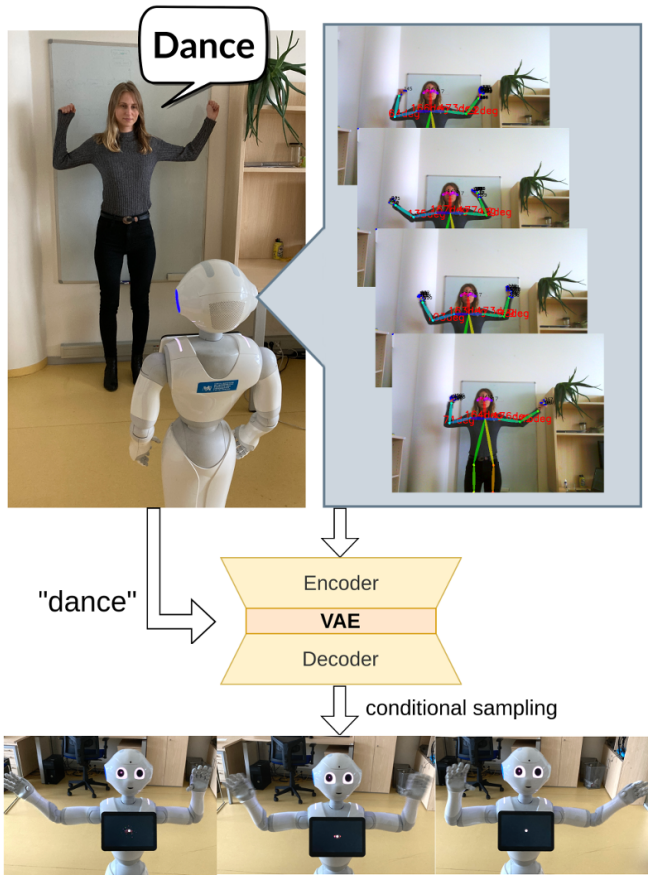

Fig. 1. Overview of our few-shot incremental imitation learning scenario with the humanoid robot Pepper. We demonstrate the tasks and their labels to the robot, extract the sequences of joint positions from RGB images and iteratively train the conditional Variational Autoencoder (VAE). After each update, we can observe the learning progress using conditional sampling and demonstration of the generated actions. If the action quality needs to be improved, more examples can be provided.

The work was supported by the Czech Science Foundation (project no. GA21-31000S) and by CTU Student Grant Agency (reg. no.SGS21/184/OHK3/3T/37).

II. Related Work

Our work explores how the current incremental VAE approaches might be used for teaching a humanoid robot simple actions from few human demonstrations. The related work on incremental and few-shot learning, as well as on the usage of generative models for learning action sequences from demonstration, is discussed in this section.

A. Incremental Learning with VAE

Incremental (or continual) learning deals with the situation when not all the data are available in one go. Either new data for already known classes might be coming, or even new classes of the data can appear on the input. Here we differentiate between the case when the task-id is provided during the inference time (so-called task-incremental learning) and the case when the trained model has to discriminate between all the classes seen in the previous tasks without reference to task-id (so-called class-incremental learning).

The typical approaches to incremental (or continual) learning for generative models are to store a subset of past data or use a generative replay of past data by sampling from the latent space [5], [6]. The former has high storage demands (as well as might have privacy/safety issues), and the latter is more computationally expensive. A different way how to deal with newly coming data is to penalize weights in the network while learning new tasks/classes to avoid catastrophic forgetting of the network [7]. However, the problem is that the difference in weights is not clearly corresponding to the difference in outputs. In [8], the learning rate adapts according to the uncertainty defined in the probability distribution of the weights in networks. In [9], they find optimal additive perturbation to the prior distribution induced by the encoder for individual tasks.

There are recent proposals that show that class-incremental (few-shot) learning might be enabled by training separate VAEs for individual classes and then mixing them together [10], [11]. This approach can be used in the cases of unbalanced data and also avoid catastrophic forgetting. However, one drawback is its poor scalability with a growing number of classes, another downside is that such an approach does not allow for the generation of semantical transitions between classes (which is one of the desired features of VAEs).

Label-conditioning of the VAE enables easier separation of individual classes in the common latent space and therefore the sampling is very convenient. An example of label-conditioned VAE is ACTOR, an action-conditioned Transformer-based architecture, that enables to encode and decode a sequence of human motions [4]. The disadvantage is again that we cannot generate semantical transitions across classes or use the model for classification. The ACTOR architecture was tested on the action dataset in non-incremental scenarios.

The above-mentioned methods of incremental learning for VAEs have typically been tested on toy datasets such as MNIST [12] and not for incremental learning of actions. In this paper, we explore the ways how the ACTOR [4] architecture might be utilized for few-shot incremental learning. We compare 3 of these approaches (storing past data, resampling from the distribution, and training separate VAEs for individual classes) and explore their behaviour based on the model setting and data itself.

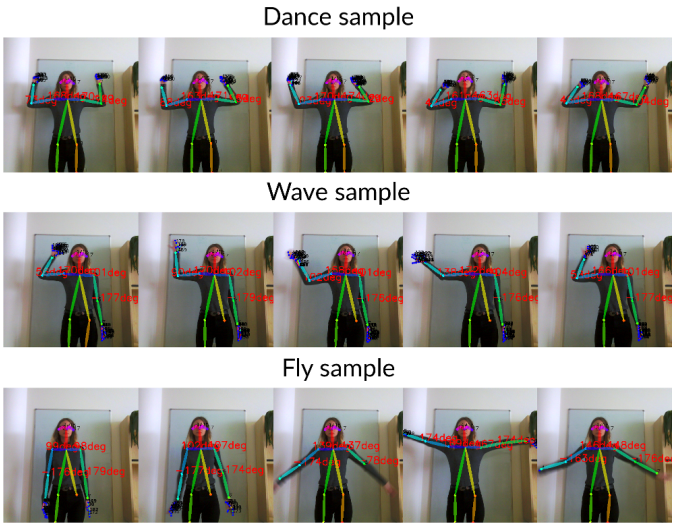

Fig. 2. Examples of the three action classes in our dataset: *dance*, *wave* and *fly*. We analyse the images with OpenPose [13] and extract angles of the shoulders and elbows.

B. Generative models for Learning from demonstration

A survey [14] covers the latest research on deep learning and reinforcement learning methods used for teaching a robot to perform highly complex tasks. Supervised approaches struggle with cascading failures when the agent trajectory diverges from the demonstration (dubbed "behavioural cloning"). Considering that we have enough data, deep generative models seem to be a promising approach in imitation learning for motor control. However, typically only the next state of the robot is predicted (e.g., [15]). This approach is vulnerable to cascading failures when we aim to generate a longer trajectory. Another approach for few-shot learning is Generative Adversarial Imitation Learning (GAIL), which lets the agent interact with the environment during training, and therefore can learn more robust controllers from fewer demonstrations, but might fail to produce adequately diverse samples (dubbed "mode collapse"). In [16] and [17], the benefits of both VAE and GAN are combined to produce robust policies capturing diverse behaviour. Noseworthy et al. [18] present a Task-Conditioned Variational Autoencoder (TC- VAE) that enables specification of parameters that are important for the given task (or motion primitive) while the remaining parameters are learnt from demonstrations. The approach is demonstrated by learning individual movement primitives (very simple task trajectories) that are represented by the positions of the end-effector.

Many of the above-mentioned models consider the availability of all the training data in advance, predict only the next step of the trajectory, and consider simple motions without periodicity. Compared to such approaches,

we consider the case, when the new data are coming continuously. We, therefore, evaluate the performance of incremental VAE methods. Furthermore, we consider more complex movements, that include also periodicity and try to learn (and generate) the whole sequence of the data.

III. Experimental setup

In our experiments, we consider an imitation learning scenario where the SoftBank Robotics' humanoid robot Pepper learns to repeat arm motion performed by a human demonstrator. We use the robot's inbuilt RGB camera to record videos of the observed humans and analyse their joint angles using OpenPose [13].

In the demonstrated scenario, we focus on arm motion, i.e. we detect the shoulder and elbow joint angles (which can be directly mapped to the robot's joints). We also focus on "planar" motion, that is the demonstrated actions only include arm motion in the plane parallel to the camera view plane. This restriction enables us to estimate the joint angles from 2D images only, which ensures the training data is maximally precise and balanced and does not introduce any bias to the model comparison. The camera frame rate was 10 fps, the demonstrated actions took around 3 seconds. The recorded actions are thus sequences of 27-33 poses, each extracted from the raw RGB image using the OpenPose library.

For the training, we use a conditional VAE with transformer encoder and decoder networks inspired by the ACTOR model from Petrovich et al. [4]. Although we use the same network design and condition the decoder on the action label, we do not feed the action into the encoder network as the authors since we did not find rapid improvement in such cases for our dataset.

The inputs are the whole sequences of actions, the variable lengths are handled with padding and additional masks that inform the transformer network on the original sequence length. We train the model to minimize the evidence lower bound (ELBO) as follows:

$$\mathcal{L}_{ELBO}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z}, \boldsymbol{c})] - \mathrm{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)||p(\boldsymbol{z})) \quad (1)$$

where KL is the Kullback-Leibler divergence between the encoder's distribution $q_\theta(z|x)$ and $p(z)$, while $\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}_i)}$ is the reconstruction term conditioned on class c.

The neural network parameters (such as the number of layers and heads in the Transformers) and optimizer were varied for performance comparison and are described in Section VI.

IV. Dataset description

Although this work focuses on the possible usage of a VAE model in real-time incremental learning, we still collect a dataset to be able to compare several approaches systematically on the same data (the data is presented to the model in very small batches exactly as in the actual real-time scenario).

For the comparison of various learning approaches, we collect a class-annotated training dataset composed of 3 action classes: *wave*, *dance* and *fly*. There are 20 samples for each class (60 altogether) and another 10 samples per class are used for the test set.

Each training sample is a timeseries of 27-33 poses, where each pose is represented by 4 angles. The form is thus $4 \times n$: $[\theta_{sl_1}, \theta_{el_1}, \theta_{sr_1}, \theta_{er_1}, ..., \theta_{sl_n}, \theta_{el_n}, \theta_{sr_n}, \theta_{er_n}]$, where $\theta_{sl}, \theta_{el}, \theta_{sr}, \theta_{er}$, are left shoulder angle, left elbow angle, right shoulder angle, and right elbow angle, respectively. The angles are in degrees and, for training, they are normalized between -1 and 1. The actions are all periodic motion (waving, dancing, pretending to fly) and the number of repetitions is fixed across samples, so that the variability is only in the speed and trajectory deviations. You can see examples of the training actions in Fig. 2.

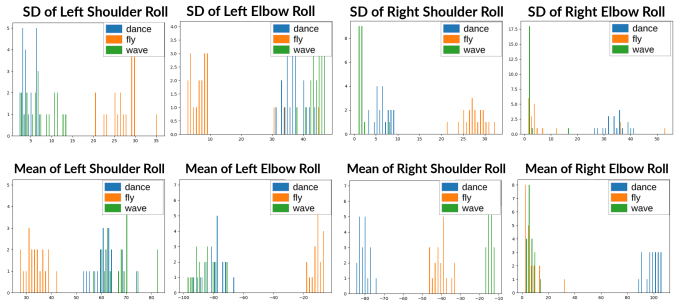

Fig. 3. Histograms of standard deviations (SD) and means of the training data for each action class. There are 20 data samples for each action and the actions are represented as sequences of joint poses. The means and SDs are thus calculated for each action sample and each of the four joints. We use the histograms to compare and estimate the quality of VAE sampled/reconstructed actions (see Subsection V-B).

V. Experiments

We consider a task-incremental learning scenario in which the human demonstrator shows examples for one class at a time and observes the learning progress through class-conditioned sampling when the robot directly performs the desired actions. Since the class name (e.g. *wave*, *fly* or *dance*) is provided along with the sample, it is possible to either train a single VAE model conditioned on the class or to initiate a new VAE instance each time a new class type appears and train it on the corresponding data.

Both approaches have advantages and disadvantages: training a separate model for each action type enables us to finetune the training parameters (such as latent vector size, learning rate, network parameters etc.) and the model cannot suffer from catastrophic forgetting since it is class-specific. In comparison, a single multi-class VAE is a more scalable option for a larger number of classes and the probabilistic latent space enables sampling of transitions between two or more different class types (provided the latent space is regularised). However, extra mechanisms need to be added to avoid forgetting the previously learned classes - for example, the commonly used generative replay [5], [6].

In our set of experiments, we compare the two above-mentioned approaches and investigate the conditions that need to be met to achieve the desired results (measured in terms of plausibility of generated actions, see Subsection V-B).

A. Experiment Overview

We compare two main approaches: training a new VAE for each new class and training one VAE for all classes presented in sequential order. In order to achieve optimal results, we first compared the training hyperparameters and observed their impact on the test loss and task accuracy. The order of experiments is thus following:

- Comparison of hyperparameters - we perform a hyperparameter grid search over both VAE-specific and learning-specific parameters. From the former, we compare the dimensionality of the latent space and parameters of the transformer encoder and decoder networks (number of attention heads and layers). From the latter, we compare the selected optimizers, batch size and number of iterations on each batch.
- Task-incremental learning of three individual class-specific VAEs using the selected hyperparameters.
- Task-incremental learning of a single VAE first without generative replay, then using two generative replay approaches differing in the time at which samples of the previous actions are added.

For the generative replay, we save the weights of the trained model each time before updating on samples from a new class. We then use these saved models to generate samples of the class they were last trained on, select only the samples which meet our sample accuracy metrics (see Subsection V-B) and mix them together with the currently trained class. You can see the different mixing methods and their impact in Section VI.

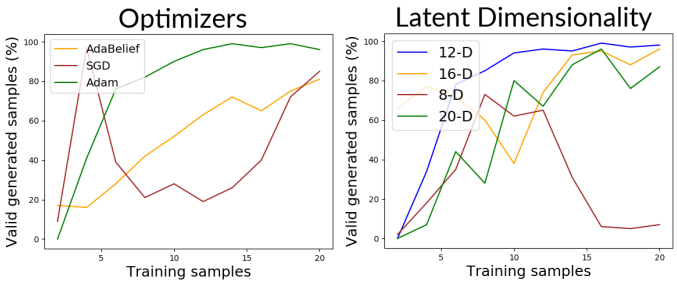

Fig. 4. Impact of the used optimizer (left) and latent space dimensionality (right) on the final quality of the generated samples. For clarity, we show Robustness for the VAE model trained on the *dance* class only, although the trend was similar for other classes as well.

B. Evaluation metrics

Since we are interested in qualitative results rather than quantitative, we create our own metrics that measure the statistical features of the output actions such as the mean and standard deviation (SD) over the whole sequence and for each joint. The mean informs us whether the model learned the default joint positions for the given action, SD reflects whether the model learned to reconstruct the periodic motion.

You can see the histograms of means and SDs for all joints in Fig. 3. Based on these ground truth features, we specify for each class $c_{i=\{1,2,3\}}$, joint $t_{i=\{1,\dots,4\}}$ and statistical feature $f_{i=\mu,\sigma}$ an interval $[\alpha_{low}, \alpha_{high}]_{c_i,t_i,f_i}$ within which the action's mean or SD should fit to be considered as valid. During evaluation, we retrieve N actions from the model and evaluate what percentage of samples s_j would fit with their features within all the intervals $[\alpha_{low}, \alpha_{high}]_{c_i,t,f}$ for the given class i. We refer to the resulting metrics as *Sample Accuracy* A_{c_i}:

$$\mathrm{A}_{c_i} = \frac{1}{N} \sum_{j=1}^{N} \mathbb{1}[\alpha_{low} \leq s_j \leq \alpha_{high}]_{t,f,c_i} \qquad (2)$$

Similarly as in [18], we use our Sample Accuracy to measure the *Class Success* and *Robustness* for the trained model:

- *Class Success* measures the Sample Accuracy for reconstructions of the testing data ($N = 10$ samples per class), e.g. how many of the 10 reconstructed samples were valid.
- *Robustness* stands for the Sample Accuracy of random actions generated from the learned prior conditioned on the action/class label (we use $N = 100$ samples per class).

Note that *Robustness* is more important than *Class Success* for the imitation learning scenario as it shows the percentage of plausible actions that the robot makes during inference. To make the *Robustness* metric more informative, we further distinguish the Sample Accuracy into three levels of precision: *difficult*, allowing zero tolerance for deviation of some of the sample features from their predefined intervals, *medium*, allowing 15 % deviation of the sample features from the predefined ground truth intervals, and *easy*, allowing 25 % deviation from the predefined intervals. Note that even in the *easy* scenario, the samples could be still empirically recognized.

VI. Results

A. Comparison of hyperparameters

To find the most stable and the best-performing model, we perform a hyperparameter grid search. In the VAE architecture, we vary the latent vector size (dimensionality of the latent space) and the hyperparameters of the Transformer encoder and decoder networks (number of attention heads and number of layers). In the training scheme, we vary the optimizer, batch size and number of iterations on each batch.

1) VAE parameters: Based on our hyperparameter grid search, we find that there is a narrow range of the latent vector sizes that produce the optimal results and the model fails to learn when the dimensionality of the latent space is too big or too small (see Fig. 4). We obtain the best results with 12-D latent vectors for the class-specific VAEs and 16-D latent vectors for the multi-class VAE scenario.

Next, we observe that the Transformer network hyperparameters such as the number of layers and attention heads can have a large influence on the model's capability to capture variance/periodicity in the time series. For example, with the 8 attention heads and 2 layers, the model cannot reconstruct the variance across the time series (e.g., the robot found the right

TABLE I
COMPARISON OF CLASS SUCCESS AND ROBUSTNESS METRICS FOR A VAE TRAINED ON THE DANCE CLASS ONLY, WITH BATCH SIZE 1 AND 5. WE SHOW THE RESULTS AFTER $S_{5,10,15,20}$ TRAINING SAMPLES

Batch Size	Metric	S_5	S_{10}	S_{15}	S_{20}
1	Class Success	0.0	0.17	0.1	1.0
	Robustness	0.0	0.78	0.94	0.97
5	Class Success	0.0	0.0	0.13	0.18
	Robustness	0.0	0.03	0.47	0.79

position but was not moving). A reduced number of layers (1) and heads (4) enables the model to generate the sequences including the periodic motion.

2) Learning parameters: We compared the convergence of three different optimizers in the case of an incremental learning scenario with few samples. We compared the commonly used Adam optimizer [19], Stochastic Gradient Descent (SGD), and the recently proposed AdaBelief [20]. The AdaBelief optimizer was outperformed by the Adam optimizer for all the training scenarios. Furthermore, Adam provided more stable results across various parameter setups compared to SGD and we thus used it for the subsequent experiments (see the comparison in Fig. 4).

Next, we tried multiple training scenarios in terms of batch size (number of samples presented at each iteration) and number of iterations for each batch. We varied the batch size between 1 and 10 samples. Although larger batches lead to slightly lower final loss, we found smaller accuracy of the generated samples probably due to overfitting (see Table I). The best results in terms of Robustness were thus obtained with batch size 1.

Last, we varied the number of weight updates for each sample (corresponding to the number of epochs). Since we cannot iterate over the whole dataset in the online scenario, we update the weights on each presented sample N times. We found that the highest sample accuracy can be obtained for our scenario with $5 \leq N \leq 10$.

Based on the hyperparameter grid search, we used for all the subsequent experiments a VAE architecture with 1 hidden layer and 4 attention heads, 12-D latent vectors for class-specific VAE and 16-D latent vectors for the multi-class VAE. The training was done with Adam optimizer and batch size 1, while each sample was presented to the network 5 times (for the class-specific VAEs) or 10 times (for the multi-class VAE).

B. Class-specific VAEs

Using the optimal learning hyperparameters, we trained a specialised VAE for each class to see how many samples are needed for optimal results. As you can see in Fig. 5, each class has different convergence and thus requires a different minimum number of samples for optimal accuracy. For example, in the case of the *dance* action, around 12 samples are needed to get 100 % accuracy for the *easy* threshold of Sample Accuracy, while the whole set of 20 samples is needed to achieve 100 % in the case of the *fly* action.

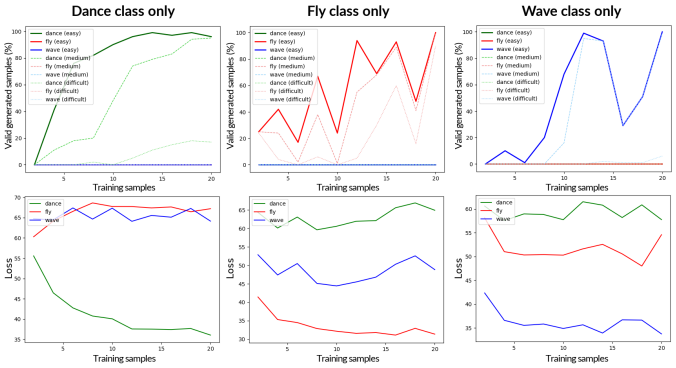

Fig. 5. Overview of the single-task VAEs based on their specialization (*dance*, *fly* or *wave*) and the provided number of training samples (x-axis, 1-20). The top row shows the Robustness for each model, e.g. the percentage of conditional samples generated by the model that have the desired statistical features (see Subsection V-B for details). We show three different curves (*easy*, *medium*, *difficult*) based on how tight the range for the desired features is. In the bottom row, we show the test loss calculated on the testset of 10 actions for each task.

Overall, we show that the few-shot incremental learning of a single class is possible using the sequential VAE. However, the training can fail completely with slightly altered hyperparameters and it is thus always necessary to perform a grid search. Furthermore, although our data are more complex than the typically used data for action classification from few samples, they are still quite simple and with minimal noise compared to real human demonstrations. Tuning a model for very complex data could be much more challenging in the few-shot scenario.

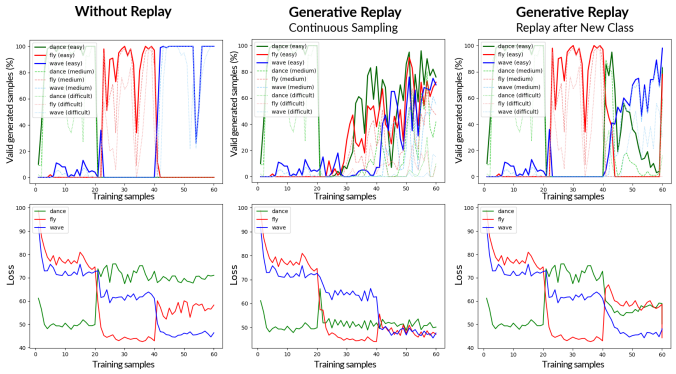

Fig. 6. Comparison of task-incremental learning scenarios using a class-conditioned sequential VAE. We show the Robustness (upper row, see Section V-B for metric details) and loss (lower row) for each of the classes with increasing number of presented training samples. In the Generative Replay scenarios, weights of the model are saved when a new class appears and are used for sample generation in later stages. In the Continuous Sampling scenario (centre), the samples from previous tasks are continuously mixed with the new samples, while in the rightmost approach, the model is retrained on mixed samples from previous tasks only after it finishes learning the current class.

C. Multi-class incremental VAE

We compare three possible scenarios of how the individual classes might be learned within one class-conditioned VAE:

1) Naïve approach: we present samples for one task at a time and observe the Sample Accuracy during training.

2) Generative replay with continuous sampling: we save the weights of the model at the moment a new class is presented and use it to generate samples for further replay. We continuously mix the samples from previous tasks with the new training data.

3) Generative replay with replay after new class: We generate samples in the same way as in 2), but we first retrain the model on the new class only and then, after the learning of the class is finished, we retrain the model with a mixture of samples from all the classes learned by that time.

For the results, see Fig. 6. The naïve model forgets the previous task right after it is presented with a new one. However, you can see that each task reaches high Sample Accuracy much faster compared to the individual task-specific VAEs in Fig. 5. This indicates that the model can make use of the previously learned tasks and does not need to start learning from scratch. In terms of loss and accuracy of the generated actions, both presented incremental scenarios with generative replay achieved similar results in the final stage of learning. However, the generative replay with continuous sampling is able to provide better results while a new task is presented. Please note that in Fig. 6, both the loss and Sample Accuracy improve rapidly in the last step of the Generative Replay after the new task as the model is retrained on samples from all three previously learned tasks.

In terms of loss and accuracy of the generated actions, both presented incremental scenarios with generative replay achieved similar results in the final stage of learning. Although the final performance is slightly lower than in the case of the individual class-specific VAEs, such a model can be potentially used for traversing over the latent space and generating novel types of actions based on the combination of the known ones.

VII. Conclusions and discussion

In this paper, we explore the possible approaches toward task-incremental few-shot learning of actions from human demonstrations using a conditional VAE. Firstly, we show whether and how it is possible to train a VAE on as few as 20 examples per class so that it can generate semantically accurate actions based on the given task label. Secondly, we compare two task-incremental learning approaches based on generative replay that enable learning of a potentially unlimited number of classes within a single VAE model (see Fig 6).

We train and evaluate the model on real-world data using a humanoid robot Pepper and sequences of human joint positions retrieved from RGB images using the OpenPose library [13]. In our future work, we plan to extend this architecture to more complex tasks such as object manipulation annotated with natural language descriptions rather than simple task-ids. If successful, the proposed approach could enable us to teach robots new skills in a more natural way similar to how we teach infants - by showing them examples and observing their progress as they try to repeat our actions.

References

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[2] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," *Advances in neural information processing systems*, vol. 30, 2017.

[3] C.-C. Lin, Y. Hung, R. Feris, and L. He, "Video instance segmentation tracking with a modified vae architecture," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 147–13 157.

[4] M. Petrovich, M. J. Black, and G. Varol, "Action-conditioned 3d human motion synthesis with transformer vae," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 985–10 995.

[5] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *Advances in neural information processing systems*, vol. 30, 2017.

[6] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell, "Continual unsupervised representation learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[8] S. Ebrahimi, M. Elhoseiny, T. Darrell, and M. Rohrbach, "Uncertainty-guided continual learning with bayesian neural networks," *arXiv preprint arXiv:1906.02425*, 2019.

[9] E. Egorov, A. Kuzina, and E. Burnaev, "Boovae: Boosting approach for continual learning of vae," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[10] D. C. Mocanu and E. Mocanu, "One-shot learning using mixture of variational autoencoders: a generalization learning approach," *arXiv preprint arXiv:1804.07645*, 2018.

[11] G. M. van de Ven, Z. Li, and A. S. Tolias, "Class-incremental learning with generative classifiers," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021, pp. 3606–3615.

[12] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[13] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[14] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, vol. 21, no. 4, p. 1278, 2021.

[15] M. Zambelli, A. Cully, and Y. Demiris, "Multimodal representation models for prediction and control from partial information," *Robotics and Autonomous Systems*, vol. 123, p. 103312, 2020.

[16] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess, "Robust imitation of diverse behaviors," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[17] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3758–3765.

[18] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy, "Task-conditioned variational autoencoders for learning movement primitives," in *Conference on robot learning*. PMLR, 2020, pp. 933–944.

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

[20] J. Zhuang, T. Tang, Y. Ding, S. C. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan, "Adabelief optimizer: Adapting stepsizes by the belief in observed gradients," *Advances in neural information processing systems*, vol. 33, pp. 18 795–18 806, 2020.