Hybrid system identification using switching density networks

Michael Burke
School of Informatics
University of Edinburgh
michael.burke@ed.ac.uk

Yordan Hristov
School of Informatics
University of Edinburgh
yordan.hristov@ed.ac.uk

Subramanian Ramamoorthy
School of Informatics
University of Edinburgh
s.ramamoorthy@ed.ac.uk

Abstract:
Behaviour cloning is a commonly used strategy for imitation learning and can be extremely effective in constrained domains. However, in cases where the dynamics of an environment may be state dependent and varying, behaviour cloning places a burden on model capacity and the number of demonstrations required. This paper introduces switching density networks, which rely on a categorical reparametrisation for hybrid system identification. This results in a network comprising a classification layer that is followed by a regression layer. We use switching density networks to predict the parameters of hybrid control laws, which are toggled by a switching layer to produce different controller outputs, when conditioned on an input state. This work shows how switching density networks can be used for hybrid system identification in a variety of tasks, successfully identifying the key joint angle goals that make up manipulation tasks, while simultaneously learning image-based goal classifiers and regression networks that predict joint angles from images. We also show that they can cluster the phase space of an inverted pendulum, identifying the balance, spin and pump controllers required to solve this task. Switching density networks can be difficult to train, but we introduce a cross entropy regularisation loss that stabilises training.

Keywords: Behaviour cloning, switching density networks, hybrid systems

1 Introduction

Behaviour cloning is a commonly used technique in learning from demonstration or imitation learning. Here, demonstrations of successful behaviours are used to train models that replicate the demonstrated behaviour [1, 2]. Supervised learning problems such as these are often formulated as classification or regression tasks. The former typically assumes that some prior categorisation has been done, possibly through an initial clustering phase, while the latter tends to ignore these aspects and focuses only on predicting some real-valued output. However, many learning problems require that a hierarchical model be learned, where some latent symbolic aspect of a problem maps to a real valued observation.

This is particularly true in robotics applications, where hierarchical learning is often key to robust, generalisable control, and learned skills are typically required to be decomposable for re-use in other applications. Historically, this problem has been addressed in a multi-stage process. For example, in the context of learning from demonstration, behaviours or low-level skills are often first identified through a clustering process, before being grounded through some learning process [3, 4]. Hybrid systems [5] offer a rich mechanism for expressing behaviours, but are typically obtained by hand. More recently, differentiable parametrisations have been exploited for gradient-based inference in hierarchical latent variable models [6, 7]. This has paved the way to incorporating valuable structure into end-to-end learning models. The incorporation of structure into neural models, while still subject to debate, has gained in popularity recently [8, 9, 10], motivated as a mechanism for interpretable learning and as a means of improving performance. Interpretable robot behaviour is key if learning robots are to be trusted in practical settings.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

This work introduces switching density networks (SDN) for switching control law identification. Switching density networks are a form of mixture density network [11] leveraging Gumbel-Softmax [7] gating to learn to generate densities based on a binary encoded bottleneck layer. This forces the network to perform an intermediate clustering phase prior to making output predictions, which results in more interpretable, hierarchical models that allow for further reasoning. Unlike traditional hybrid system identification, which typically occurs in simple state spaces [12], switching density networks allow for hybrid system identification with richer sensor data such as images.

This interpretability is a particularly useful property in robotics, and can be used to learn compositional control strategies from demonstration. In our experiments, SDNs are used to predict the controller parameters and reference states for a family of proportional-integral-derivative (PID) control laws. Hybrid systems of these control laws are applied ubiquitously across domains [13] including process control and automotive applications, so this family covers a broad class of applications. Experimental results show that switching density networks are able to identify the robot joint angle goals that constitute a demonstration sequence and can successfully learning visual grounding of these goals. Moreover, we show that SDNs learning PID control law families can identify the state-space regions required for pumping, spinning and balancing an inverted pendulum.

Training such SDNs can be challenging, and we show that they are indeed vulnerable to mode collapse. This work introduces a cross-entropy batch regularisation loss that remedies this and allows for reliable training.

Formally, our goal is to identify hybrid systems or state space models conditioned on a discrete switching process, that is, to find a mapping from observation $\mathbf{z}_t$ to a state of interest, $\mathbf{x}_t$, in a stochastic dynamical system conditioned on a discrete latent variable i_t,

$$q(\mathbf{x}_t|\mathbf{z}_t) = p(\mathbf{x}_t|\mathbf{z}_t, i_t) \tag{1}$$

$$i_t \sim p(i_t|i_{t-1}). \tag{2}$$

State space models of this form are particularly powerful, as they can be used to express complex motions and behaviours using interpretable sub-components In the context of sensorimotor control for robotics, let $\mathbf{z}_t$ denote sensor data captured at time t and $\mathbf{x}_t$ the pose or configuration of a robot at time t. i_t is a discrete indicator variable controlling the transition between robot behaviour states or environment dynamics.

2 Related work

Numerous models and approaches [14] have been developed to address the learning problem formulated above. Gaussian mixture models fit using expectation maximisation [15] are widely used for clustering, while their switching state space analog, Gaussian emission hidden Markov models have a long history of application in sequence learning. Although typically fit using the Baum-Welch algorithm [16] (a form of expectation maximisation), variational approaches have also been proposed for a broader class of switching state space models [17].

Learning for switching state space models can also be considered from a changepoint detection perspective, and a range of numerical inference techniques have been used to detect changepoints in sequential data [18]. More recently, variational and gradient-based inference strategies for Bayesian learning have proved useful in hierarchical modelling [6, 7] and variational auto-encoding [19].

Hierarchical modelling is an effective means of incorporating structure into a learning problem, so as to avoid sample inefficient learning and improve generalisation through abstraction. Work on options learning [20, 21] and skill identification [22, 23] has paid significant attention to hierarchical learning, but has been a particular challenge for visuomotor control.

Our work is inspired by sequential composition theories in robotics [24], where tasks are solved by moving between sub-controllers lying within the domains of one another. Here, we seek to identify the sub-controllers required for a given task in an end-to-end fashion, from demonstration sequences. Learning from demonstration (LfD) [25] is widely acknowledged as a particularly useful paradigm for robot programming. Significant progress has been made in LfD, moving beyond the direct replication of motions to produce more robust approaches [26] through the introduction of more general schemes for modelling motion like dynamic motion primitives [27], linear dynamical attractor systems [28], sparse online Gaussian processes [29, 30] or conditionally linear Gaussian

models [31, 3] that can be used for trajectory optimisation. It is important to note that each of these systems behaves as a hybrid system, decomposing a state space into specific regions, and learning appropriate dynamics for each region.

More recently, trajectory optimisation approaches have been extended to incorporate end-to-end learning, demonstrating robust task level visuomotor control [32] through guided policy search, or using deep dynamic motion primitives [33]. End-to-end learning has allowed for the use of domain transfer to facilitate one-shot learning [34] from human video demonstrations, and for the use of reinforcement learning to learn optimised control policies [35, 36]. Unfortunately, end-to-end learning approaches typically lack interpretability and are difficult to verify without policy distillation [37]. Burke et al. [38] fit a sequence of proportional control laws to end-to-end model demonstrations using particle filters, in an attempt to obtain a more interpretable control system, but this approach is vulnerable to performance loss if important properties of the network fail to be inferred. In contrast, this paper shows that it is possible to learn switching proportional control laws in an end-to-end fashion, by emdedding this structure into the learning process.

In computer vision, spatial transformers [39] and capsule networks [40] embed learnable structured transformations in an attempt to better capture the relational properties of image attributes in convolutional neural networks. Without this structure, convolution neural networks can learn jumbled image representations [41]. This work shows that mixture density networks suffer from a similar problem, which switching density networks address. Switching density networks are conceptually similar to the stochastic neural network architecture proposed by Florensa et al. [42], which uses a switching structure to learn reusible skills in a reinforcement learning setting. Our work differs by considering the use of switching structures for parameter prediction for state space models, thereby incorporating known controller structure into the learning process in a lightly supervised manner.

Switching density networks are closely related to mixture density networks [11], a family of neural network constructed using K output distributions. In the Gaussian mixture case, MDNs fit a weighted combination of Gaussian distributions,

$$q(\mathbf{x}_t|\mathbf{z}_t) = \sum_{i=1}^{K} \pi_i(\mathbf{z}_t)\mathcal{N}(\mathbf{x}_t|\mu(\mathbf{z}_t), \mathbf{\Sigma}(\mathbf{z}_t)), \tag{3}$$

using mean $\mu(\mathbf{z}_t)$, variance $\mathbf{\Sigma}(\mathbf{z}_t)$ and normalised weight parameters $\pi_i(\mathbf{z}_t)$, which are predicted using a neural network. Unfortunately, there is no direct link between weight components and mean or variance components, so mixture density networks often learn seemingly arbitrary connections. We illustrate this experimentally in Section 4.2, showing that an MDN trained to predict manipulator joint angles will use only a single mixture component for completely different joint angle predictions, somewhat unintuitively learning to change the mean and variance parameters instead of toggling between mixture components. This occurs because no structure forces mixture consistency in the network.

3 Switching density networks for hybrid control

Switching state space models are typically learned using a multi-stage process. For example, Gaussian mixture models could be fit to robot state measurements, and perception networks trained to predict hidden states from image observations. Switching density networks attempt to learn hybrid systems like this jointly in an end-to-end fashion. More formally, given a hybrid system of $i = 1 \rightarrow N$ dynamical systems, each with parameters θ_i,

$$\dot{\mathbf{x}}_t = q(\mathbf{x}_t; i, \theta_i), \tag{4}$$

we train a SDN to predict parameters θ_i, maximising the log likelihood of the distribution $\mathcal{N}(\dot{\mathbf{x}}_t|q(\mathbf{x}_t; i, \theta_i), \Sigma_i)$, where Σ_i denotes the measurement uncertainty. Figure 1 shows an example SDN architecture. A SDN is similar to a mixture density network, which typically consists of a neural architecture that predicts K different sets of distribution parameters, along with a set of discrete weights K. However, unlike mixture models, switching models only predict a single output distribution, which is conditioned on a K-dimensional one-hot encoded discrete latent variable. The final layer of a SDN is a fully connected layer with no bias parameters. When combined with the bottleneck switching layer, this produces a switching state as output.

Gradient-based learning with discrete latent variables is challenging, as backpropagation is unsuitable for non-differentiable layers. The Gumbel-softmax distribution [7] approximates a categorical distribution using a temperature (τ) controlled softmax function,

$$y_i = \frac{\exp\left((\log(\pi_i) + g_i)/\tau\right)}{\sum_{j=1}^{K} \exp\left((\log(\pi_i) + g_i)/\tau\right)} \text{ for } i = 1 \dots K, \tag{5}$$

with logits π_i and i.i.d samples g_i drawn from a Gumbel(0,1) distribution. As the temperature τ tends to 0, samples from the Gumbel-softmax distribution tend towards a one-hot encoded binary vector. As a result, neural model training using temperature annealing allows for backpropagation to be used to learn models with discrete latent parametrisations.

The Gumbel-softmax reparametrisation allows switching density networks to be trained with a discrete bottleneck layer, using stochastic gradient descent to minimise the negative log likelihood of a state given a network prediction conditioned on an input observation.

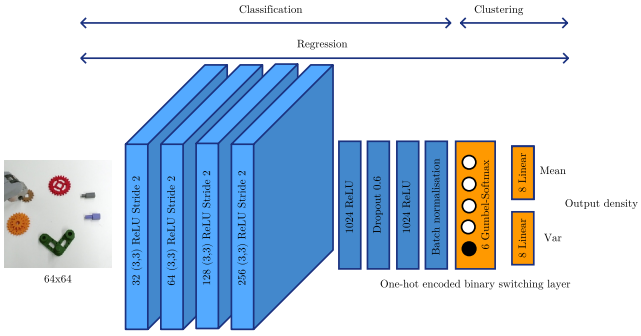


Figure 1: Switching density networks are discrete latent variable models that switch between output densities. This architecture is used for reaching controller identification in 8 DOF joint angle space.

In contrast to typical hybrid system identification problems, where system dynamics are identified, in a behaviour cloning setting we only observe the *closed-loop* or controlled response of the system of interest, so need to infer switching controller models. As a particularly useful example, we choose to express the behaviour of a robot using a generative switching model comprising a sequence of PID controllers, motivated by the proportional control formulation of Burke et al. [38]. PID control laws produce controller actions $\mathbf{u}_k$,

$$\mathbf{u}_k = K_p(\mathbf{x}_k - \mu) + K_i \sum_{l=1}^{L} (\mathbf{x}_{k-l} - \mu) + K_d \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta_t}, \tag{6}$$

using three error terms comprising a proportional gain K_p acting on the error between state $\mathbf{x}_k$ at time step k and a desired reference point μ, an integral gain K_i acting on the cumulative error between state $\mathbf{x}_k$ and a desired reference point μ over a window of time steps l, and a derivative gain K_d acting on the change in state, over a time difference Δ_t.

In a behaviour cloning setting, we typically observe state-action pairs and train models to regress the appropriate action for a given state. However, if we assume that actions should be produced by proportional-integral-control laws, we can reformulate the regression problem as one of predicting the controller gains and reference points that generate observed actions. In complex systems, it may be the case that we require multiple PID control laws in different states and are required to switch between control laws. In this case, and assuming Gaussian observation noise, we can model the controller action,

$$\begin{aligned}\mathbf{u}_k \sim K_p(\mathbf{z}_t, i_t)\left[\mathbf{x}_t - \mu(\mathbf{z}_t, i_t)\right] + K_i(\mathbf{z}_t, i_t) \sum_{l=1}^{L} [\mathbf{x}_{k-l} - \mu(\mathbf{z}_t, i_t)] + \\ K_d(\mathbf{z}_t, i_t)\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta_t} + \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}(\mathbf{z}_t, i_t)),\end{aligned} \tag{7}$$

using a hybrid system of i distinct control law parameters and reference points conditioned on some sensor data. Here, each controller is parametrised by a set of gains, $K_p(\mathbf{z}_t, i_t), K_i(\mathbf{z}_t, i_t), K_d(\mathbf{z}_t, i_t)$, and goal configuration states, $\mu(\mathbf{z}_t, i_t)$. Controller action is measured subject to uncertainty, $\mathbf{\Sigma}(\mathbf{z}_t, i_t)$. Sequencing a number of these controllers allows a robot to transition through the set of states required to solve many manipulation and navigation tasks.

Given this model and a demonstration sequence of state and observation pairs, our goal is to learn to identify K suitable sub-controllers that make up the demonstrated robot behaviour. For the Gaussian proportional controller model of (7), we train switching density networks using the log-likelihood ($\mathcal{LL}$) objective:

$$\theta^* = \text{argmin}_\theta - \mathcal{LL}(\mathbf{u}_k | K_p(\mathbf{z}_t;\theta), K_d(\mathbf{z}_t;\theta), K_i(\mathbf{z}_t;\theta), \mu(\mathbf{z}_t;\theta), \mathbf{\Sigma}(\mathbf{z}_t;\theta)), \tag{8}$$

In many cases, there may be multiple possible gains and reference points that produce an observed controller action. In practice, we address this by only predicting a subset of the controller parameters, through sensible weight initialisation in the final layer of the SDN and by relying on persistent excitation in the demonstration sequence.

Unfortunately, training a switching density network frequently results in mode collapse. Here, all probability mass becomes concentrated in a single class, and the network merely learns to regress the mean state. We remedy this by using an additional cross-entropy loss term, which serves as a batch-level regulariser. Here, we assume that all categories are likely to occur at a similar rate in a given batch, and minimise the cross-entropy between the average Gumbel-softmax distribution over a batch $\hat{y}_i$ and a uniform distribution,

$$\theta^* = \text{argmin}_\theta - \mathcal{LL}(\mathbf{u}_k | K_p(\mathbf{z}_t;\theta), K_d(\mathbf{z}_t;\theta), K_i(\mathbf{z}_t;\theta), \mu(\mathbf{z}_t;\theta), \mathbf{\Sigma}(\mathbf{z}_t;\theta)) - \sum_{i=1}^{K} \frac{1}{K} \log(\hat{y}_i). \tag{9}$$

This ensures that on average no individual category in the switching layer becomes dominant, and helps to counter mode collapse.

4 Experimental Results

We evaluate SDN controller identification in two distinct domains. The first, an inverted pendulum, is a canonical hybrid continuous control problem, while the second, a set of visuomotor manipulation tasks, illustrates the applicability of SDNs to higher dimensional input and output spaces. For both experiments, we specify the exact number of controllers to be identified. In practice, should this number not be known, more controller switches could be specified, to allow for redundancy.

4.1 Balancing an inverted pendulum

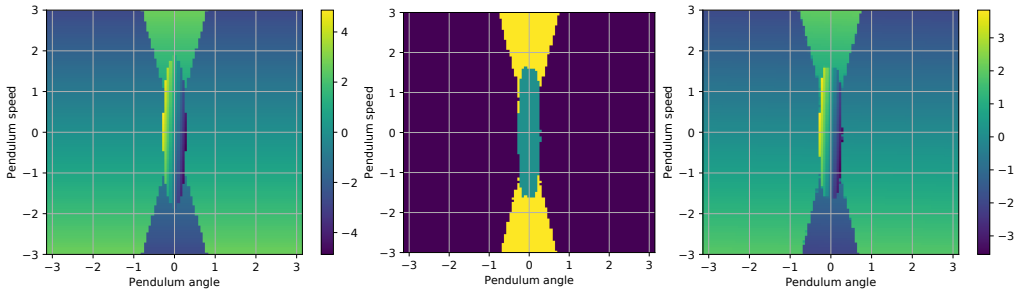
We demonstrate the use of SDN PID controller laws on a simulated, under-actuated inverted pendulum. Closed-loop control of an inverted pendulum can be accomplished using a hybrid system with three key modes [43], that derive from fundamental properties of the physics of this system. In the pump mode, energy is injected to the system, such that it can be swung up. In the spin mode, energy is removed from the system through damping action. Finally, when the pendulum is near vertical, a balancing controller can be used to maintain the pendulum in an upright position. Control in each mode can be provided by a proportional control law, with the energy in the system used to determine which control law to apply.

We train a SDN (3 fully connected layers of 16 neurons, and 3 switching states) to predict the proportional controller gains (no reference points are needed as the pendulum goal state is known) for the three controllers described above, using 10,000 state-action pairs provided by the original hybrid controller [43]. Figure 2 shows the controller response in different regions of the state space, with the three controller regions clearly visible. Importantly, the figure shows that SDNs learn to identify the regions in which each of these controllers should be applied, in addition to the required sub-controller parameters. Direct behaviour cloning using a neural network is still effective, but does not provide the same level of interpretability as the learned hybrid control system.

Table 1: Pendulum experiments

	Average reward
Hybrid controller	-0.812 ± 0.514
SDN PID	-0.857 ± 0.621
Fully connected	-0.850 ± 0.538

The latent structure can be used to interpret and reason about the underlying dynamics and physics of the controlled system and environment. For example, by analysing the regions in which sub-controllers operate (Figure 2), along with the inferred controller parameters, we can see that the pendulum requires positive feedback control in the blue region, indicating the presence of a system with a stable equilibrium point, and negative feedback control in the blue-green region, indicating an unstable equilibrium point.



(a) Hybrid controller (b) Sub-controller regions (c) SDN response

Figure 2: Controller responses for inverted pendulum learned using the SDN closely match those of the ground truth hybrid controller used for demonstration. Importantly, the SDN correctly identifies the regions in state space in which each controller should be applied.

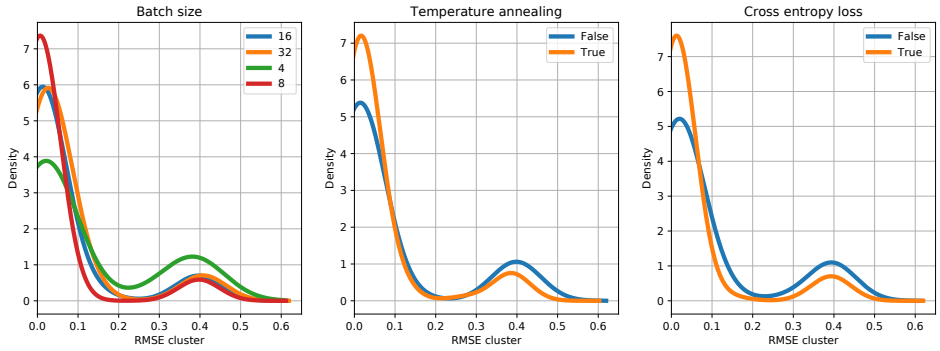


Figure 3: Kernel density estimates of root mean square error between true controller goals and PR2 goal predictions highlight the value of cross entropy regularisation and temperature annealing.

Table 1 shows the average reward ($\theta^2 + 0.1\dot{\theta}^2 + 0.001u^2$, where $(\theta, \dot{\theta})$ denotes pendulum angle and velocity, and u the control) obtained over 1,000 randomly initialised tests using the ground truth hybrid controller, and controllers learned using a SDN. Behaviour cloning using a baseline, fully connected network with no structure performs similarly to a SDN, but the latter is more interpretable, as specific control laws can be linked to regions in state space, allowing for more detailed controller analysis. Importantly, the inferred control law parameters are close to the ground truth values, indicating that the SDN has successfully identified the underlying control laws demonstrated.

4.2 Identifying PR2 manipulation controllers

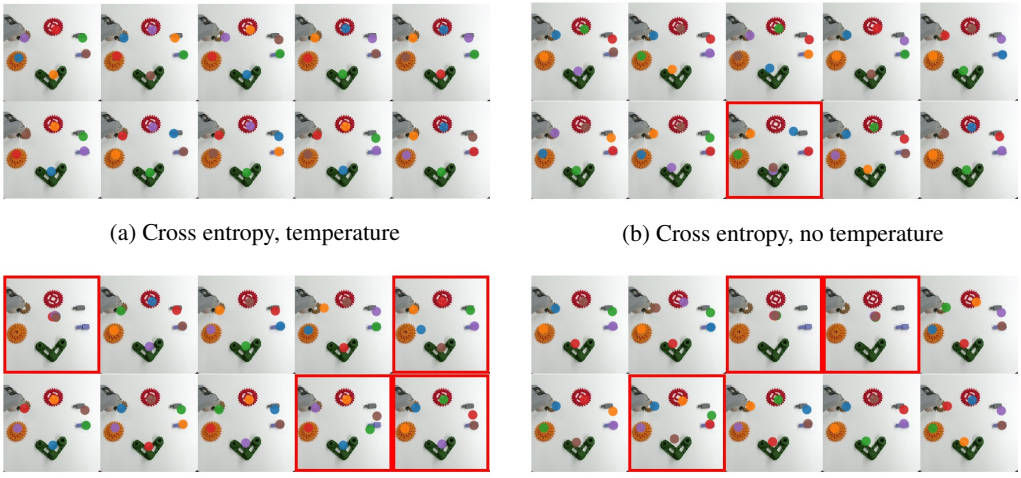
We also evaluate the use of switching density networks (Architecture in Figure 1) for controller goal identification using an inspection task with a PR2 robot. Here, the PR2 is required to repeatedly reach to a series of components. We hard coded this behaviour and collected approximately 2,000 images and corresponding joint angle (8 dimensions) and velocity measurements, while the PR2 repeated this process 10 times. Our goal is to learn to identify the sub-controllers that make up this sequence, and train a model to predict these controller parameters from image observations. We split this set into two, with the first 1,000 frames used for training and the remainder for testing.

Table 2: RMSE

SDN	0.8°
CNN	1.47°
MDN	3.95°

Here, we assume that the controller gains are known, and only learn to predict controller reference points. Figure 3 shows the distribution over the root mean square error in predicted joint angle goals for each frame in the test set. Results are provided for varying batch sizes, using varying Gumbel-softmax temperature annealing schedules, and with or without the proposed cross-entropy loss. Experiments were repeated 10 times for each parameter setting combination.

It is clear that both temperature annealing and the cross-entropy loss are required for stable training. Batch size is a proxy for the rate of temperature annealing, since temperature was annealed at each epoch step, but also affects the cross-entropy loss term. If the batch size is too low, the assumptions governing the cross entropy regularisation loss are less likely to be true. This effect is clearly visible



(a) Cross entropy, temperature

(b) Cross entropy, no temperature

(c) No cross entropy, temperature

(d) No cross entropy, no temperature

Figure 4: Projected joint angle goals (dots) identified using the SDN highlight the importance of cross entropy regularisation. Failed goal identification is indicated using a red border.

for the batch size of 4. Figure 4 shows the projection of the detected joint angle goals into the image plane for experimental runs using the various parameter configurations. When trained with both a cross-entropy loss and temperature annealing the SDN successfully identifies the inspection goals comprising this task. Without the cross-entropy loss, the network is vulnerable to mode collapse, where predicted goals regress to the mean.

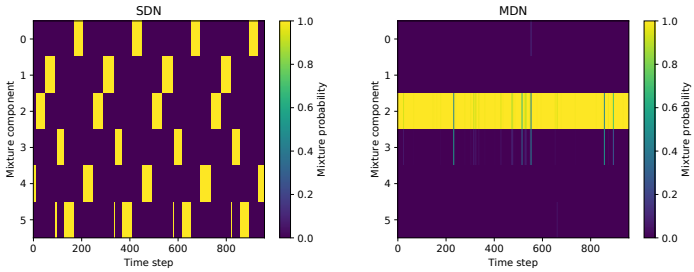


Figure 5: Unlike the SDN, MDNs do not necessarily use different mixture components for similar regressions.

It is important to note the difference between switching and mixture density networks. The former enforces a hierarchical latent model structure, while the latter places no constraints on the mapping between mixture components and distribution outputs. This can be observed when the distribution of the mixture weights is shown for the test sequence for switching and mixture density networks, when both are trained to predict controller goals. SDNs implicitly learn the latent task behaviour, while MDNs simply regress using arbitrary paths through the network. Table 2 shows the root mean square errors over all joints for the best performing parameter settings on the inspection task. The MDN performs substantially worse than the SDN, which captures the inherent switching structure of the inspection task. In general, larger batch sizes improve MDN results, as does temperature annealing, but cross entropy regularisation has little effect. Figure 5 shows the trace of mixture component densities, which highlights the fact that SDNs learn intrinsically meaningful grounded predictions, while MDNs rely on an arbitrary mapping between input and output.

Figure 6 shows the projected controller rollouts obtained using the SDN. It is clear that the SDN has identified the appropriate switching points and control strategies need to replicate the inspection task. This is particularly useful for learning from demonstration, as this options discovery allows for the inclusion of higher level reasoning about the inspection plan being followed, and the identification of implicit constraints or search patterns followed by the demonstrator.

Although seemingly simple, even contact rich behaviours can be expressed using PID control laws. For example, when we applied a SDN to a kinesthetically demonstrated suitcase opening task, we discovered two primary controllers (Figure 7), one moving beneath the case lid, and a second that opened it by moving to a goal state above the case. Importantly, the SDN allows for the use of vision to determine which controller to apply.

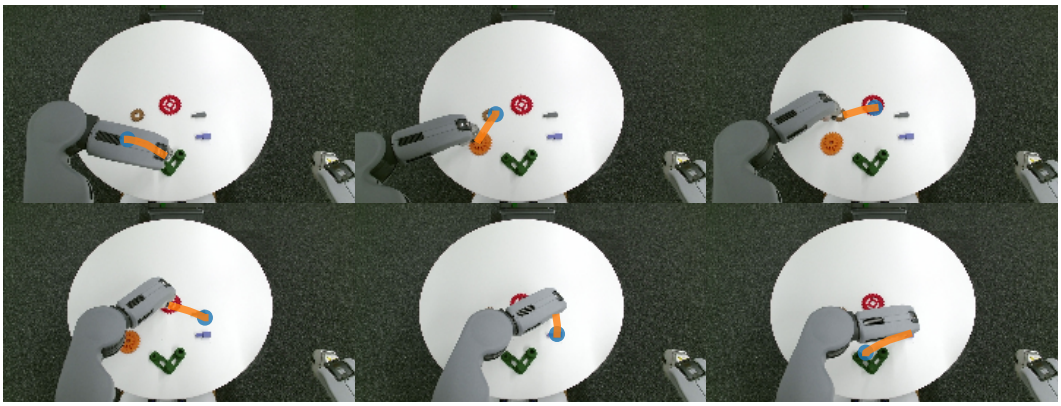


Figure 6: Inferred controller rollouts (joint angles projected into the image plane) are obtained by predicting the controller goal state for the given image using the SDN, and then using the associated PID controller to generate a trajectory.

Empirically, we have found that fully connected networks were easier to train (faster convergence) and perform slightly better than SDNs on tasks that can be solved using a smooth non-linear controller (pendulum), but that SDN's are easier to train and provide substantial improvements over direct CNNs on tasks with a clear switching structure (inspection task). Our hypothesis is thus that SDNs are best suited to hybrid systems, which cover a broad set of processes. However, we believe that the increased training difficulty in the first case is made up for by the substantial interpretability gains that can be obtained by discretising a process using an SDN.

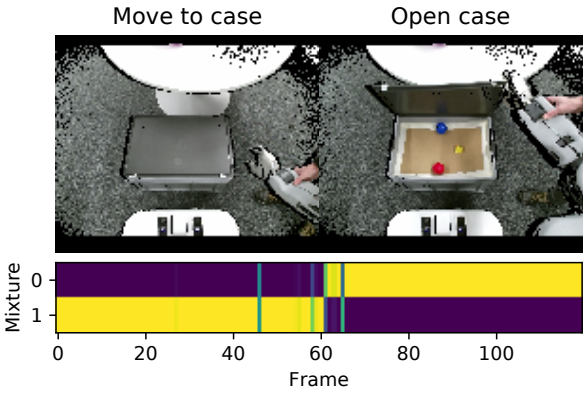


Figure 7: Two controllers are identified for a kinesthetically demonstrated suitcase opening task.

5 Conclusion

This work has introduced an approach for end-to-end hybrid system identification using switching density networks and generalised state-space control laws, in the context of behaviour cloning. Switching density networks are harder to train than their fully connected counterparts, but this work has shown empirically that the addition of a cross-entropy regularisation term stabilises training. Hybrid systems are frequently used in robotics, and PID controllers are a trusted and well understood control paradigm, used widely across domains and disciplines. Although demonstrated using PID control laws, the proposed approach allows for hybrid system identification using other controller families. Importantly, jointly inferring sub-controllers and the states in which they are applied allows for reasoning about implicit constraints and the physical properties of systems and environments.

This work has shown that SDNs can be used to perform goal identification in a visuomotor manipulation tasks and inverted pendulum controller discovery, identifying compositional control strategies that can be directly used for explainable control. We have contrasted these with MDNs and fully connected neural networks, which are unable to learn interpretable latent representations. Moreover, hybrid system identification using SDNs allows for the re-use of learned policies in downstream tasks, through hierarchical reinforcement learning or options scheduling. Future work will involve exploring options scheduling with sub-controllers learned using SDNs.

Acknowledgments

This work is supported by funding from the Turing Institute, as part of the Safe AI for surgical assistance project. We are particularly grateful to the Edinburgh RAD group for valuable discussions and recommendations.

References

[1] D. A. Pomerleau. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation*, 3(1):88–97, March 1991.

[2] J. A. D. Bagnell. An Invitation to Imitation. Technical Report CMU-RI-TR-15-08, Carnegie Mellon University, Pittsburgh, PA, March 2015.

[3] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, 2014.

[4] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *IJRR*, 34(2):131–157, 2015.

[5] R. Goebel, R. G. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems*, 29(2):28–93, 2009.

[6] D. Kingma and M. Welling. Efficient Gradient-Based Inference through Transformations between Bayes Nets and Neural Nets. In *ICML*, volume 32, pages 1782–1790, 2014.

[7] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.

[8] S. Penkov and S. Ramamoorthy. Learning Programmatically Structured Representations with Perceptor Gradients. In *ICLR*, 2019.

[9] P. Karkus, X. Ma, D. Hsu, L. P. Kaelbling, W. S. Lee, and T. Lozano-Pérez. Differentiable Algorithm Networks for Composable Robot Learning. *RSS*, 2019.

[10] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and P. Battaglia. CompILE: Compositional Imitation Learning and Execution. In *ICML*, pages 3418–3428, 2019.

[11] C. M. Bishop. Mixture density networks. Technical report, Citeseer, 1994.

[12] S. Paoletti, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal. Identification of hybrid systems a tutorial. *European journal of control*, 13(2-3):242–260, 2007.

[13] J. Lunze and F. Lamnabhi-Lagarrigue. *Handbook of hybrid systems control: theory, tools, applications*. Cambridge University Press, 2009.

[14] R. Murray-Smith and T. A. J. (Eds.). *Multiple Model Approaches to Modelling and Control*. Taylor and Francis, London, 1997.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

[16] L. R. Rabiner. An introduction to hidden Markov models. *IEEE ASSP magazine*, 3(1):4–16, 1986.

[17] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural computation*, 12(4):831–864, 2000.

[18] J. O. Ruanaidh, W. J. Fitzgerald, and K. J. Pope. Recursive Bayesian location of a discontinuity in time series. In *ICASSP*, volume 4. IEEE, 1994.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[20] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[21] G. Konidaris and A. G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *NIPS*, pages 1015–1023, 2009.

[22] S. Niekum and A. G. Barto. Clustering via Dirichlet Process Mixture Models for Portable Skill Discovery. In *NIPS*, pages 1818–1826, 2011.

[23] P. Ranchod, B. Rosman, and G. Konidaris. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *IROS*, pages 471–477. IEEE, 2015.

[24] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential Composition of Dynamically Dexterous Robot Behaviors. *IJRR*, 18(6):534–555, 1999.

[25] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009. ISSN 0921-8890.

[26] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20, 1997.

[27] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *ICRA*, pages 763–768. IEEE, 2009.

[28] K. R. Dixon and P. K. Khosla. Trajectory representation using sequenced linear dynamical systems. In *ICRA*, volume 4, pages 3925–3930 Vol.4, April 2004.

[29] J. Butterfield, S. Osentoski, G. Jay, and O. C. Jenkins. Learning from demonstration using a multi-valued function regressor for time-series data. In *Humanoids*, pages 328–333, Dec 2010.

[30] D. H. Grollman and O. C. Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *ICRA*, pages 3315–3320. IEEE, 2008.

[31] S. Chiappa and J. R. Peters. Movement extraction by detecting dynamics switches and repetitions. In *NIPS*, pages 388–396, 2010.

[32] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 17(1):1334–1373, 2016.

[33] A. Pervez, Y. Mao, and D. Lee. Learning deep movement primitives using convolutional neural networks. In *Humanoids*, pages 191–197, Nov 2017.

[34] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine. One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning. In *RSS*, Pittsburgh, Pennsylvania, June 2018.

[35] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *RSS*, Pittsburgh, Pennsylvania, June 2018.

[36] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramr, R. Hadsell, N. de Freitas, and N. Heess. Reinforcement and Imitation Learning for Diverse Visuomotor Skills. In *RSS*, Pittsburgh, Pennsylvania, June 2018.

[37] O. Bastani, Y. Pu, and A. Solar-Lezama. Verifiable Reinforcement Learning via Policy Extraction. In *NIPS*, 2018.

[38] M. Burke, S. Penkov, and S. Ramamoorthy. From explanation to synthesis: Compositional program induction for learning from demonstration. *RSS*, 2019.

[39] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial Transformer Networks. In *NIPS*, 2015.

[40] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *NIPS*, pages 3856–3866, 2017.

[41] G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *ICANN*, pages 44–51. Springer, 2011.

[42] C. Florensa, Y. Duan, and P. Abbeel. Stochastic Neural Networks for Hierarchical Reinforcement Learning. *ICLR*, 2017.

[43] B. Kuipers and S. Ramamoorthy. Qualitative Modeling and Heterogeneous Control of Global System Behavior. In *Hybrid Systems: Computation and Control*, pages 294–307, 2002.