To appear: Proc. of the 15th IEEE-RAS-EMBS International Conference on Rehabilitation Robotics (ICORR 2017), July 17-20, 2017, QEII Centre, London, UK, pp. 1-8. ©2017 IEEE

Representing High-Dimensional Data to Intelligent Prostheses and Other Wearable Assistive Robots: A First Comparison of Tile Coding and Selective Kanerva Coding

Jaden B. Travnik and Patrick M. Pilarski

***Abstract*— Prosthetic devices have advanced in their capabilities and in the number and type of sensors included in their design. As the space of sensorimotor data available to a conventional or machine learning prosthetic control system increases in dimensionality and complexity, it becomes increasingly important that this data be represented in a useful and computationally efficient way. Well structured sensory data allows prosthetic control systems to make informed, appropriate control decisions. In this study, we explore the impact that increased sensorimotor information has on current machine learning prosthetic control approaches. Specifically, we examine the effect that high-dimensional sensory data has on the computation time and prediction performance of a true-online temporal-difference learning prediction method as embedded within a resource-limited upper-limb prosthesis control system. We present results comparing tile coding, the dominant linear representation for real-time prosthetic machine learning, with a newly proposed modification to Kanerva coding that we call *selective Kanerva coding*. In addition to showing promising results for selective Kanerva coding, our results confirm potential limitations to tile coding as the number of sensory input dimensions increases. To our knowledge, this study is the first to explicitly examine representations for real-time machine learning prosthetic devices in general terms. This work therefore provides an important step towards forming an efficient prosthesis-eye view of the world, wherein prompt and accurate representations of high-dimensional data may be provided to machine learning control systems within artificial limbs and other assistive rehabilitation technologies.**

I. INTRODUCTION

Prosthetic limbs and other assistive rehabilitation technologies rely on their sensors to respond appropriately to the intentions and needs of their human users. In the specific case of clinically prescribed upper-limb prosthetic devices, the electromechanical sensor information available to a device's control system is typically limited to mechanical toggles or a small number of myoelectric (EMG) signals recorded from the tissue of the user's residual limb [1], [2]. These sensors provide enough information to design a prosthetic solution wherein a patient may be able to control one or two prosthetic actuators [3]. The use of machine learning approaches such as pattern recognition allow available sensors to be further leveraged to increase the number of functions controllable by a user [4]. There is now convincing evidence that machine learning control approaches such as pattern recognition can enable patients with amputations to sequentially control a device with a robotic elbow, wrist, and hand with multiple discrete grasp patterns—far more degrees of control than were previously possible with conventional control solutions [1], [4]. This increase in function can be attributed both to an increase in the number of sensors deployed within a prosthetic socket, and the efficient use and synthesis of the information provided by these sensors. The combination of sensorimotor signals into a useful summary of a system's state, termed a *state representation*, has become increasingly important to the performance of prosthetic devices, especially those that leverage pattern recognition, regression, or real-time machine learning [1].

This research was undertaken, in part, thanks to funding from the Canada Research Chairs program, the Canada Foundation for Innovation, and was supported by the Alberta Machine Intelligence Institute (Amii), Alberta Innovates – Technology Futures (AITF), the Government of Alberta, and the Natural Sciences and Engineering Research Council (NSERC).

J. B. Travnik and P. M. Pilarski are with the Department of Computing Science and the Department of Medicine, University of Alberta, Edmonton, AB T6G 2E1, Canada. Please direct correspondence to: travnik@ualberta.ca; pilarski@ualberta.ca

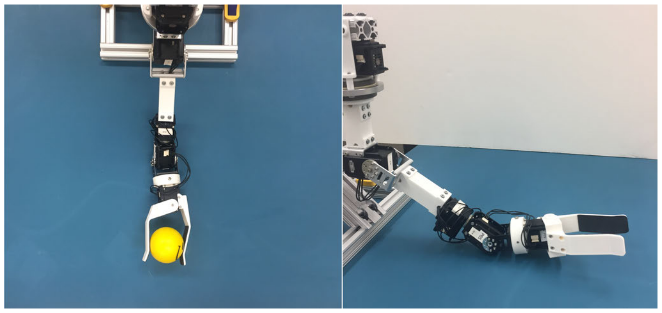

Fig. 1. The upper-limb research prosthesis used in this study (the Bento Arm [5]). The Bento Arm generates a continuous stream of position, velocity, load, voltage, and temperature sensor signals for each of its five actuators during its ongoing operation.

While sensors have played a critical role in increasing the capabilities of clinically deployed prostheses, pre-clinical research prostheses have also continued to evolved in terms of their sensorimotor space to try and meet the function, form, and feedback needs of users with amputations (c.f., [1], [2], [6], [7]). As one representative example, Fougner et al. have shown that the addition of sensors to help resolve residual limb position (e.g., accelerometers or inertial measurement units) can dramatically increase the performance of myoelectric pattern recognition as a subject with an amputation moves their limb through a range of common positions [8]. Further, even without the addition of new sensors to a prosthetic socket, the modern actuators and sensors within multi-joint prosthetic limbs can now generate a wealth of data of different frequencies, ranges, and modalities. If used carefully, these signals present a valuable window into the intent of a human user and their prosthesis's interactions with a changing, daily-life environment.

A. Towards Computationally Efficient Representations

A prosthesis must be able to approximate and react to its sensor inputs within a short window of time in order to remain effective [9]. If a representation can be formed for a control system in an efficient way, even in the face of high-dimensional sensor data, it can readily be *computed, stored, and used in real time* on the computationally limited hardware embedded within wearable prosthetic devices.

Previous work by our group has explored in detail the use of real-time machine learning methods for prosthetic control—we have to date performed a wide range of prosthetic prediction and control studies wherein subjects with and without amputations used different techniques from the field of reinforcement learning (RL) to operate a robotic arm, e.g., [10]–[16]. In all of these studies we exclusively relied on the linear representation method known as tile-coding (c.f., [17]) to provide our RL prediction and control systems with a view into the available space of prosthesis- and user-derived sensorimotor signals. These linear representations were chosen because they are highly efficient in both computation and data usage for a fixed (small) number of input signals. However, as prosthetic devices improve and the number of sensors available to the system increases, it remains unclear how standard linear representations like tile-coding are affected, and if more scalable representations exist which would be more aptly suited for use in prosthetic limbs and other wearable assistive rehabilitation technologies.

Finding a good representation of input signals in complex, high-dimensional, and continuous environments is a difficult problem faced not just in the adaptive control of prosthetic limbs, but in a wide range of domains where designers seek to deploy machine learning for real-world control tasks; notable examples range from speech recognition to self-driving vehicles. However, unlike enterprise scale applications with access to vast computational resources, the prosthetic setting requires that all computation be limited to run on small, power efficient processors that can be installed in a self-contained way within the chassis of a prosthetic device. The representation of high-dimensional continuous state information therefore needs to be done in a computationally efficient fashion. Furthermore, and specific to robotic applications, it is often impossible to exactly represent the state of a system to a control process; some approximation of the continuous state of the system must be used. One method approximating continuous signals and reducing high dimensional data is principle component analysis (PCA) which compresses high dimensional data to a much smaller set of salient features. However, recent studies have found that the implicit encoding of relevant variables PCA produces is not usable by a machine learner that requires representations of the interactions between sensorimotor data to perform well [18].

In this work we therefore contribute a first study on the effects of increasing the dimensionality of prosthetic sensory data in terms of computation time and prediction accuracy for linear tile-coding representations (the dominant function approximation approach used in RL), and propose a novel modification of a competitor coding approach that promises to scale accurately and efficiently as the number of sensory dimensions increases on an assistive device.

II. BACKGROUND

A. General Value Functions

An important idea from reinforcement learning is a mapping from the state of the agent to a prediction of an input signal, known as a value function. Most control problems in RL are concerned with the value function used to predict the desirability of being in a certain state. However, value functions can also be used to construct knowledge about sensorimotor interaction such as predicting when a bump sensor is activated or how a human wants to control a prosthetic limb [16], [19]. These are known as *general value functions* (GVFs) which can be used to ask questions about the sensorimotor data experienced by robotic systems [19]. An online GVF can be used to ask questions about the current behavior of the robotic system and can phrase these questions by using a representation of the experience and a termination function, $\gamma(S_t)$. When γ is set to a constant value between 0 and 1, it represents how many timesteps into the future to make a prediction as given by $timesteps = \frac{1}{1-\gamma}$. Thus with a constant $\gamma = 0.9$, an on-policy GVF will make a prediction that is 10 timesteps into the future.

B. True Online Temporal-Difference Learning

A central part of RL is the temporal-difference learning method known as TD(λ)[1]. With its low computational cost and good performance, it is frequently used when learning on-policy GVFs. However, TD(λ) is known to not maintain an exact equivalence with ideal mathematical outcomes for learned predictions—termed the *forward view*. Recently, van Seijen et al. proposed two small changes to the update rule of TD(λ), allowing true online temporal-difference learning methods to be constructed that do have algorithmic equivalence with the forward view [20]. These true online methods have been used to approximate value functions of sensorimotor interactions with superior performance over regular TD(λ) and Sarsa(λ) [20]. Algorithm 1 shows the true online algorithm as implemented in the present study.

C. Sparse Distributed Memory

Sparse Distributed Memory (SDM) is a mathematical model of human long-term memory [21]. It models how the distances between concepts in human minds is similar to the distances between points of a high-dimensional space where high-dimensional is at least in the hundreds. In the original formulation presented by Pentti Kanerva, a high-dimensional binary space could be represented with a set

[1] λ refers to the eligibility trace, a way of assigning credit for good and bad outcomes to states seen in the past [17].

Algorithm 1 True Online TD(λ)

```
Initialize $\theta$ arbitrarily
loop {over episodes}
    Initialize $\boldsymbol{e} = \mathbf{0}$
    Initialize $\boldsymbol{S}$
    $\hat{\boldsymbol{v}}_{\boldsymbol{S}} \leftarrow \boldsymbol{\theta}^{\intercal}\boldsymbol{\phi}(\boldsymbol{S}')$
    Repeat (for each step of episode):
        generate reward $\boldsymbol{R}$ and next state $\boldsymbol{S}'$ for $\boldsymbol{S}$
        $\hat{\boldsymbol{v}}_{\boldsymbol{S}'} \leftarrow \boldsymbol{\theta}^{\intercal}\boldsymbol{\phi}(\boldsymbol{S}')$
        $\boldsymbol{\delta} \leftarrow \boldsymbol{R} + \boldsymbol{\gamma}\hat{\boldsymbol{v}}_{\boldsymbol{S}'} - \hat{\boldsymbol{v}}_{\boldsymbol{S}}$
        $\boldsymbol{e} \leftarrow \boldsymbol{\gamma\lambda e} + \boldsymbol{\alpha}[\mathbf{1} - \boldsymbol{\gamma\lambda e}^{\intercal}\boldsymbol{\phi}(\boldsymbol{S})]\boldsymbol{\phi}(\boldsymbol{S})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{\delta e} + \boldsymbol{\alpha}[\hat{\boldsymbol{v}}_{\boldsymbol{S}} - \boldsymbol{\theta}^{\intercal}\boldsymbol{\phi}(\boldsymbol{S})]\boldsymbol{\phi}(\boldsymbol{S})$
        $\hat{\boldsymbol{v}}_{\boldsymbol{S}} \leftarrow \hat{\boldsymbol{v}}_{\boldsymbol{S}'}$
        $\boldsymbol{S} \leftarrow \boldsymbol{S}'$
    until $\boldsymbol{S}$ is terminal
end loop
```

of fixed randomly placed points known as *prototypes* [21]. Each prototype can be thought of as a memory element where it's address is its position in the space, and its value is data that can be read from and written to. Binary prototypes are declared similar if the hamming distance between their addresses is smaller than a given value.

In order to write to SDM given an address and a value, prototypes within a set radius of the specified address are updated with the value by adding it to each activated prototype's existing value (see Fig. 2). Although multiple values being stored in each prototype can corrupt the value stored there, reading from SDM is performed using the same radius method where the read value is then the average of the values from all activated prototypes. This read value is then a close approximation of data written previously to addresses near the specified address.

III. REPRESENTATION USING LINEAR METHODS

Approximating a value function using linear methods is considered a standard approach because of the ease of computing the gradient of the value function with respect to the learned parameters or weights (e.g., the weight vector θ in Alg. 1) [17]. This leads to a particularly simple form of the gradient-descent update with the guarantee that any convergence to a local optimum is a convergence to a global optimum. On top of their theoretical results, linear methods can excel in terms of computation and data efficiency but this depends critically on how the dimensions of states are represented in terms the representation's feature vector, ϕ. For instance, in continuous environments, it is natural for a single feature, $\phi(i)$, to represent a selected range of continuous values such as "joint A's angle is between 150 and 180 degrees." For complex tasks using linear methods, it is necessary to use features that represent combinations of state dimensions because linear methods, by their nature, are unable to represent interactions between their features.

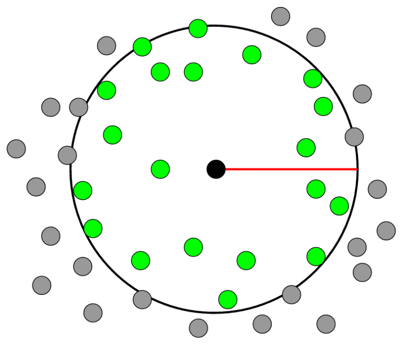

Fig. 2. A depiction of SDM prototypes being activated within a fixed radius of a specified address in the state space.

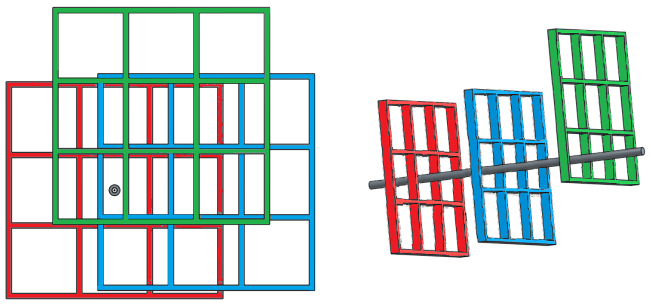

Fig. 3. 2D and 3D views of three overlapping tilings covering a 2D state space with three tiles per dimension. The number of features for this example is 27, where only three features are activated as given by the grey dot and line inside their tiles.

A. Tile Coding

Tile coding (TC) is a linear representation that is often chosen in RL for it's efficiency in online learning. Tile coding splits up the d-dimensional state space into m overlapping partitions called tilings and each tiling is split into a set of n tiles. Each tiling has an offset from each other (see Fig. 3), which leads to a better generalization [17]. Each tile represents a binary (0 or 1) feature that is activated when the state lies within the tile. Finer granularity can then be achieved by increasing the number of tiles, and thus the size of a tile, in each tiling but has the trade-off of generalization because states are less likely to activate the same tile.

A binary feature vector ϕ of length nm, can then be constructed by concatenating all n features for all m tilings. Since only m features are activated, tile coding has an advantage when choosing a constant step-size parameter, α, as the number of active features is independent of the state. For example, a reasonable step-size parameter might be $\alpha = 0.1m$.

In order to capture the interaction between dimensions, tilings must span the state space so that each tile represents a small d-dimensional hyper-cube. This leads to exponentially many features ($dim(\phi) = m * n^d$). For small dimensions found in common control tasks, tile coding provides an easily implementable representation with clear benefits. However, as the dimension of the task grows, so does the memory cost.

A common method of reducing the memory requirements is hashing, a pseudo-random compression of a large tiling into a reduced set of tiles of a given memory size. Hashing ensures that the memory requirements do not grow exponentially in the number of dimensions but still has some drawbacks. It is not clear how to choose a good memory size and some hand-tuning must be administered in practice. Additionally, the hashing function is a pseudo-random process which may have collisions leading to one feature being activated by two or more distinctly different sensory observations. Collisions may be accidentally beneficial if the collisions occur between sensory observations that have similarities and extend the generalization capabilities of the representation. However, the odds are usually in favor of a collision having negative affects. These tile collisions become more common as the memory size shrinks relative to the total number of tiles. Although it is possible to make "safe" hashing implementations of tile coding that take preventative measures when a collision is detected, these checks take additional time and require still more time to handle a collision when one has occurred.

B. Kanerva Coding

Kanerva coding is the application of sparse distributed memory as a representation in reinforcement learning (Alg. 2). Unlike tile coding, the number of features in Kanerva coding does not grow with the number of dimensions. This can lead to dramatic savings in necessary memory resources. Although a common issue when using Kanerva coding in RL is choosing an appropriate number of prototypes and activation radius, in contrast to other approximators, e.g., neural networks, the structural parameters of Kanerva coding can be easily changed without retraining the learned model. In fact, several researchers have shown improved performance by moving the prototypes around the representation space or by growing the representation using experience from the learner [22], [23]. These methods are effective but add extra memory and computational complexity to keep and examine different prototype statistics. Other work has shown that Kanerva coding may successfully be applied to continuous domains [24]. However, the computational cost of high-dimensional continuous states has not yet been explored. This is especially important in settings with limited computation and where the representation must be computed within a small amount of time.

Algorithm 2 Kanerva Coding

```
Initialize all $K$ prototypes in $P$ randomly
Choose an activation radius $r$ and distance function $d$ (e.g. Euclidean distance)
Given state $S$
    For $i$ = 0 to $K$-1
        $\phi_i \leftarrow 0$
        If $d(P_i, S) < r$
            $\phi_i \leftarrow 1$
    return $\phi$
```

C. Selective Kanerva Coding

We propose a method of finding nearby prototypes with minimal computation. The idea is to remove the activation radius and simply find the c closest prototypes to the state of the system using Hoare's quickselect which can be used to find the c shortest distances in an array of distances [25]. One way of choosing a good c is to choose a small ratio, η such that $c = \lfloor K\eta \rfloor$. Not only does this method, which we refer to here as *Selective Kanerva Coding* (SKC), still have the same $O(K)$ complexity as computing how many prototypes are within an activation radius, but it shares with tile coding the guarantee of the same number of activated features along with the associated benefits, e.g., in selecting learning step sizes. Utilized alongside True Online TD(λ), SKC promises to be an efficient, reliable representation for computing GVFs.

Algorithm 3 Selective Kanerva Coding

```
Uses $quickselect(D, c)$ which finds the $c$ smallest indicies in array $D$ of length $K$ in $O(K)$ complexity
Initialize all $K$ prototypes in $P$ randomly
Choose an $\eta$ such that $c = \lfloor K\eta \rfloor << K$ and distance function $d$ (e.g. Euclidean distance)
Given state $S$
    Initialize $D$ = $\mathbf{0}$
    For $i$ = 0 to $K - 1$
        $\phi_i \leftarrow 0$
        $D_i \leftarrow d(P_i, S)$
    $I \leftarrow quickselect(D, c)$
    For $i$ = 0 to $c - 1$
        $index \leftarrow I_i$
        $\phi_{index} \leftarrow 1$
    return $\phi$
```

A trivial extension of selective Kanerva coding, with an extra O(η) sorting operation, would further enable normalized real-valued features to be returned. In this setting, after applying quickselect to find the closest c prototypes, their features would be the normalized value of their proximity to the state, thus the closest prototype would have a feature equal to 1 and the furthest activated prototype would have a feature equal to 1/c. Further investigation into this real valued selective kanerva coding is needed, as it has great potential utility for assistive technologies.

IV. EXPERIMENT

A robotic arm designed to be worn as a research prosthesis (the Bento Arm of Dawson et al., [5], Fig. 1) was used to generate data for a prediction problem in order to compare the time and prediction performance of selective Kanerva coding and tile coding. This robot arm has shoulder and elbow joints, wrist rotation and flexion, as well as a gripper for a total of 5 degrees of freedom. Each joint contained sensors for position, velocity, load, and temperature which leads to 20 real valued sensor signals from all of the servos (as shown using different coloured traces in Fig. 1).

The arm was controlled in a sorting exercise where three different objects were picked up from the same location, carried, and then dropped at different locations assigned by their weight. A single trial consisted of the arm beginning at one end of its shoulder rotation, closing its gripper around an object, lifting its elbow while rotating its shoulder towards the drop off location at which point the elbow would descend, the gripper would release the object, and then the arm would return to the initial position. 30 trials were performed for each of the three objects, giving a total of 90 trials. Data from all servos was collected during each of these trials. Since each trial began and ended in the same location, the order in which each trial was presented could and was randomly shuffled. 5 long streams of continuous sensor information were created using this method, each containing over 16 minutes of sensor readings.

The goal of the prediction task was then to predict what angle the shoulder angle sensor would report over the next 10 timesteps ($\sim$0.3 seconds into the future). Predictions were made through the use of an on-policy general value function learned via true online temporal-difference learning (Alg. 1). In order to predict 10 timesteps into the future, $\gamma = 0.9$ was used in specifying the GVF. Learning rate and the eligibility trace parameters were empirically set to $\alpha = 0.001$ and $\lambda = 0.99$, respectively.

To examine the effect of the number of sensor dimensions on prediction performance and computation time, three different sensor combinations were used—representations were constructed with 2, 8, and 12 sensory input signals. In the 2-dimensional setting, only the elbow angle and load were given as part of the representation, whereas in the 8-dimensional setting, the angle, load, temperature, and velocity of both the elbow and the wrist rotator were given. Finally, the angle, load, temperature, and velocity of the elbow, the wrist rotator, and the wrist flexor were given in the 12-dimensional task.

Exactly 248 different combinations of tiles and tilings were used to generate a wide range of tile coding configurations and thus a wide range of features for each dimensional setting, ranging from 128 to 331776 features. One reasonable method of offsetting each tiling is to deterministically shift each tiling using a deterministic knight's-move pattern[2]. Although this knight's pattern is deterministic, by using the five different long streams of data generated by shuffling the trials, a more representative distribution of online learning performance was calculated.

For selective Kanerva coding, the number of prototypes ranged from 1000 to 20000 where each prototype was a point in the 2, 8, or 12 dimensional space. Euclidean distance was used to measure how close each prototype was to the observed state. A constant ratio $\eta = 0.025$ of features (rounded down) were active at anytime. For example, in the 1000 prototype case, the features of the 2 closest prototypes to the observed state were activated. In the 8000 prototype case, the features of the closest 200 prototypes to the observed state were activated. To ensure that different random distributions of the prototypes was accounted for, five different seeds were used to randomly distribute the prototypes.

[2]Similar to that in chess, the knight's pattern used here shifted the tilings over and up by an increasing amount for each new tiling, as per Miller and Glanz [26].

Thus each of the 248 configurations of tile coding and all five seeds of selective Kanerva coding were used for the 2, 8, and 12 dimensional settings using all five of the long streams of robot generated data. For each run, the prediction performance as well as the length of time it took to calculate active features, and the computation time per timestep, was recorded. To reveal the relation between the number of features that could be calculated and the calculation time per timestep, the number of features that could be calculated within 100 ms was also recorded.

V. RESULTS

Figure 4 compares the variation in both prediction error and computation time for tile coding and SKC as their number of features increases, up to a maximum computation time per step cutt-off of 100ms (selected in this study as the upper limit for real-time prosthesis control, c.f., [9]). After removing low-feature outliers from the tile coding data which were orders of magnitude worse than the trend seen in Fig. 4, the data indicates that both tile coding and SKC have improved performance as the number of features increases up until an optimum point at which the error increased or the representation took too long to compute. The maximum number of features calculated within 100 ms for the 12, 8, and 2 dimensional settings for SKC are represented as points I, II, and III, respectively. The mean number of features that can be calculated within 100 ms using tile coding had little variation on the log scale presented in Fig. 4, and is represented by a single point IV. The exact numbers these labeled points represent is shown in Tab. I.

The 100 ms processing limit had the effect that the 8 and 12 dimensional settings of tile coding did not improve beyond what the 2 dimensional setting was able to achieve, despite the possibility of improved prediction performance if more computation was allowed. TC was quite unlike SKC, which not only had similar performance trends across different number of dimensions but utilized the additional dimensions of the state space to have improvements in performance using the same number of prototypes and thus features. The best performance in terms of error across all conditions was found to be SKC at 8000 prototypes for all three different dimensionality conditions, with 12D SKC at 8000 features demonstrating the overall best performance on the entire experiment.

TABLE I
THE MAXIMUM NUMBER OF FEATURES THAT CAN BE CALCULATED WITHIN 100 MS ON A SINGLE-CORE PROCESSOR.

	Tile Coding		**Selective Kanerva Coding**	
Dimensions	Mean	Std	Mean	Std
2	247439	302	19333	236
8	244496	534	12960	198
12	245724	620	10883	165

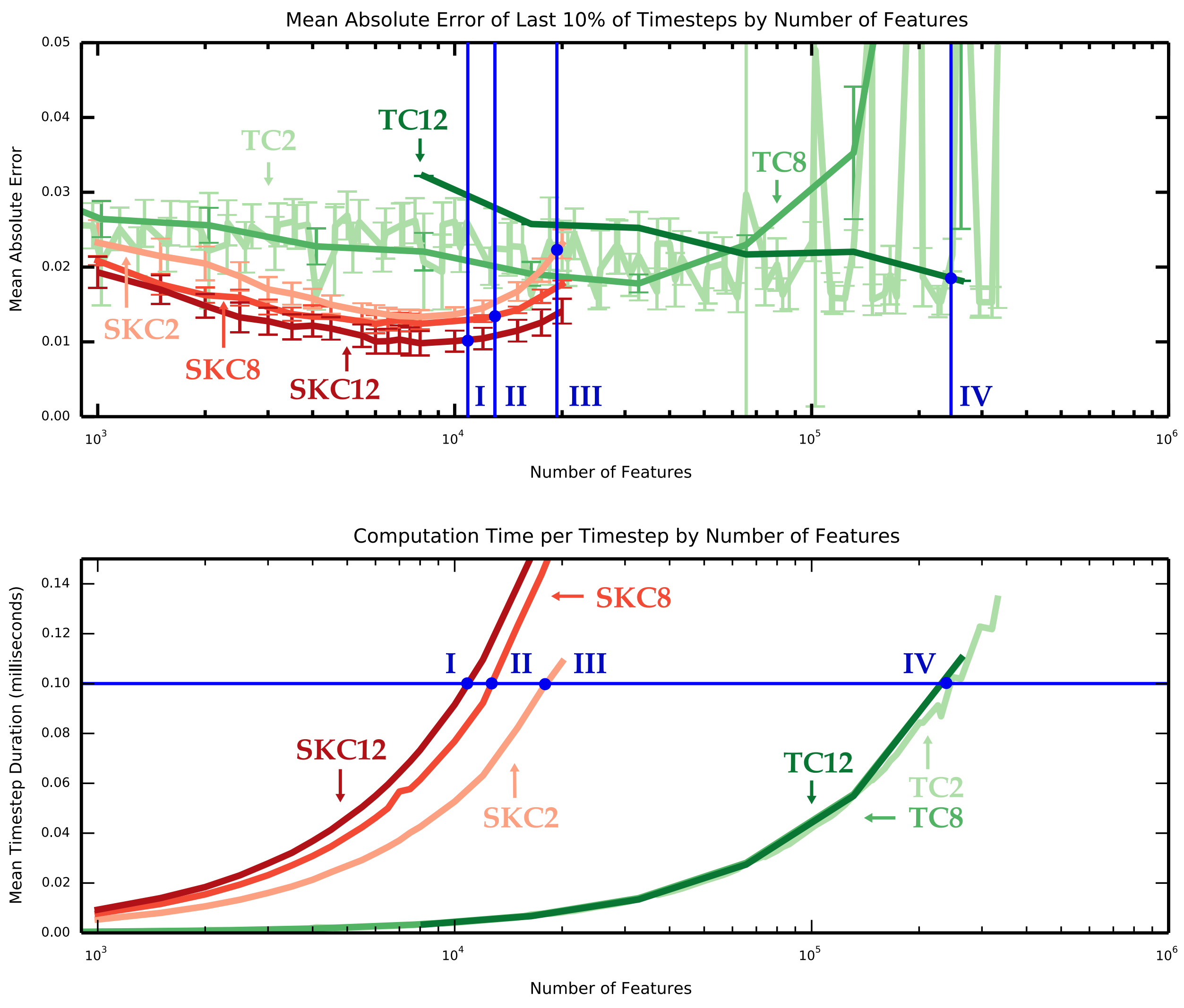

Fig. 4. Key result: comparison of prediction error and computation time for tile coding and SKC as their number of features increases, including (top) the mean absolute prediction error on the last 10% of the combined data for SKC and tile coding on 2, 8, and 12 dimensions; and (bottom) the mean computation time of the combined data for SKC and tile coding on 2, 8, and 12 dimensions. The maximum features computable in a 100ms timestep is shown in both plots by blue lines and the points I, II, II, and IV for SKC12, SKC8, SKC2, and TC2/8/12, respectively

Additionally, both tile coding and SKC had increased time step duration as the number of features increased. Not only was tile coding significantly more time efficient than SKC, but because there were tile coding configurations with the same number of features across different dimensions and each tile coding feature is calculated at the same speed, these different configurations had same computation time per timestep and thus overlapped as seen in Fig. 4 (bottom). As Euclidean distance was used to compute the distance between prototypes in SKC, the computation time per timestep increased with additional dimensions. This decreased the amount of features that be calculated within the 100 ms with increasingly more dimensions as seen in Tab. I.

Although SKC required significantly more time to calculate activated features, the extra time taken proved to have a stronger influence on prediction accuracy up until the optimal. After 8000 features, large numbers of additional features proved to be detrimental to SKC's performance which resulted in the error having a convex form with a minimum at 8000.

VI. DISCUSSION

Our results indicate that SKC is a representation that should be explored further within the context of prosthetic control, assistive or rehabilitation robotics, and other domains where high-dimensional continuous signals must be efficiently represented in real time to an adaptive or non-adaptive control process.

The five different random distributions of prototypes created for SKC did not lead to significant inter-distribution variations in performance (i.e., SKC had a consistently small standard error across prototypes distributions). This is an improvement over standard Kanerva coding, where the distribution of prototypes is known to play a significant role in the performance of the predictor. By only activating the features whose prototypes are the closest instead of within a radius, SKC invokes a limit on the distance between the furthest activated prototype and the state by the nature of distances in higher dimensional spaces. As the number of dimensions of a state space grows, the distances between random points in the space grows exponentially. In standard Kanerva coding, this growth along with the distribution of prototypes themselves increases the difficulty of appropriately setting the activation radius as it must grow exponentially as well. SKC, on the other hand, uses a notion similar to K-Nearest-Neighbors and locates the closest prototypes given that the majority of the prototypes (assuming $\eta < 0.5$) are further away. Following from the increasing distances between these prototypes, the set of features activated by SKC is appropriately flexible to changes in scale and dimension.

The error with respect to the number of features in SKC follows a convex curve which indicates that there is an optimal number of prototypes given an η. This decrease in performance as excessive prototypes are added to the representation requires further investigation. However, following from previous work where the addition and deletion of prototypes was explored, one could extend the present work by applying gradient descent methods to learn the optimal number of prototypes K and activation ratio η and where these prototypes should be positioned (e.g., via gradient derivations similar to those of Sutton et al. [27]). This would be an important result, as it could lead to a representation with fewer prototypes, and thus features, that still accurately represents a high dimensional state space. With fewer features, the representation can be constructed faster and the extra time can be devoted to making more predictions or to engaging more computation-heavy methods such as planning.

As the computation time increases with additional dimensions and features, there must exist an upper bound of how many dimensions and features can be represented on a single-core processor within a specified amount of time. The results indicate that SKC might be a representation that could provide accurate predictions until this upper bound is reached. Although this study explored the effect of higher dimensions within a limited time frame, further studies are needed consider the utility of different methods given variable time constraints and even higher dimensions.

Finally, it is natural to expect that to achieve the best performance on a prosthetic prediction or control task, the optimal number of or distribution of features in a SKC (or other) representation may be specific to each individual. That is, the best representation for a given prosthesis-user partnership may depend on the unique characteristics of an individual user's signals, behaviours, and the capabilities and operation of their prosthetic device. The best way to interpret signals for use in a machine learning or conventional control system may also change with time as the user's interactions with a device shift through experience and training. Exploring adaptive extensions to SKC and comparing them with other representation learning approaches to determine their viability on resource-constrained prosthetic control systems is therefore an important topic for future study.

VII. Conclusions

As the number of sensors available on prosthetic devices grows with their capabilities, an appropriate synthesis of sensorimotor signals into a useful representation becomes vital to the performance of these devices' machine learning control systems. If such a representation can remain computationally efficient, it can readily be used on the computationally limited systems residing within wearable prosthetic technology. This study explored how increasing the number of input signals affected performance and per step computation time of a true-online reinforcement learning system using both tile coding and a new, modified version of Kanerva coding that we term *selective Kanerva coding*.

The presented results reaffirm previous findings about tile coding's increasing computational requirements on high dimensional data. Our results further show that selective Kanerva coding has better accuracy on an upper-limb robotic prediction task than tile coding. We note that selective Kanerva coding takes additional time to compute a representation, but also show that not only are there significant gains in prediction performance with additional features but that there is a performance gradient for a fixed activation ratio, η. These findings suggest that selective Kanerva coding merits further study, and as such, this work contributes a significant step towards accurately representing the high-dimensional data of assistive technologies to a machine learning control system such as a reinforcement learning agent.

Acknowledgement

The authors thank their colleagues in the Bionic Limbs for Improved Natural Control Lab and the Reinforcement Learning and Artificial Intelligence Laboratory for a number of helpful discussions relating to this work.

References

[1] C. Castellini, P. Artemiadis, M. Wininger, et al., "Proceedings of the first workshop on peripheral machine interfaces: Going beyond traditional surface electromyography," *Frontiers in Neurorobotics*, vol. 8, no. 22, Aug. 2014. doi: 10.3389/fnbot.2014.00022.

[2] P. M. Pilarski, J. S. Hebert, "Upper and lower limb robotic prostheses," *in Robotic Assistive Technologies: Principles and Practice, Eds. P. Encarnacao and A. M. Cook*, pp. 99–144. Boca Raton, FL: CRC Press, 2017. ISBN: 978-1-4987-4572-7.

[3] P. Parker, K. Englehart, B. Hudgins, "Myoelectric signal processing for control of powered limb prostheses," *Journal of Electromyography and Kinesiology* vol. 16 no.6, pp. 541–548, 2006 doi:10.1016/j.jelekin.2006.08.006

[4] E. Scheme, K. Englehart, "Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use," *Journal of Rehabilitation Research and Development* vol. 48, no. 6, pp. 643–660, 2011.

[5] M. R. Dawson, C. Sherstan, J. P. Carey, et al., Development of the Bento Arm: An improved robotic arm for myoelectric training and research, *Proc. of MEC'14: Myoelectric Controls Symposium*, Fredericton, New Brunswick, August 18–22, 2014, pp. 60–64.

[6] D. J. Atkins, D. C. Y. Heard, W. H. Donovan, "Epidemiologic overview of individuals with upper-limb loss and their reported research priorities," *J. Prosthet Orthot.*, vol. 8, pp. 2–11, 1996.

[7] C. Antfolk, M. DAlonzo, B. Rosén, et al., “Sensory feedback in upper limb prosthetics,” *Expert Review of Medical Devices*, vol. 10, pp. 45–54, 2013.

[8] A. Fougner, E. Scheme, A. D. C. Chan, et al., “Resolving the limb position effect in myoelectric pattern recognition,” in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 6, pp. 644–651, Dec. 2011. doi: 10.1109/TNSRE.2011.2163529

[9] T. R. Farrell, R. F. Weir, “The optimal controller delay for myoelectric prostheses,” *IEEE Transactions on neural systems and rehabilitation engineering* vol. 15, no. 1, pp. 111–118, 2007.

[10] P. M. Pilarski, M. R. Dawson, T. Degris, et al., “Online human training of a myoelectric prosthesis controller via actor-critic reinforcement learning,” *Proceedings of the 2011 IEEE International Conference on Rehabilitation Robotics (ICORR)*, June 29-July 1, Zurich, Switzerland, pp. 134–140. doi:10.1109/ICORR.2011.5975338

[11] P. M. Pilarski, M. R. Dawson, T. Degris, et al., “Adaptive artificial limbs: A real-time approach to prediction and anticipation,” *IEEE Robotics and Automation Magazine* vol. 20, no. 1, pp. 53–64, 2013. doi: 10.1109/MRA.2012.2229948

[12] P. M. Pilarski, T. B. Dick, R. S. Sutton, “Real-time prediction learning for the simultaneous actuation of multiple prosthetic joints,” *Proceedings of the 13th IEEE International Conference on Rehabilitation Robotics (ICORR)*, June 24-26, 2013, Seattle, USA, pp. 1–8. doi: 10.1109/ICORR.2013.6650435

[13] A. L. Edwards, M. R. Dawson, J. S. Hebert, et al., “Application of real-time machine learning to myoelectric prosthesis control: A case series in adaptive switching,” *Prosthetics & Orthotics International*. vol. 40, no. 5, pp. 573–581, 2016.

[14] A. L. Edwards, J. S. Hebert, P. M. Pilarski, “Machine learning and unlearning to autonomously switch between the functions of a myoelectric arm,” *Proceedings of the 6th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob2016)*, June 26–29, 2016, Singapore, pp. 514-521.

[15] C. Sherstan, J. Modayil, P. M. Pilarski, “A collaborative approach to the simultaneous multi-joint control of a prosthetic Arm,” *Proceedings of the 14th IEEE/RAS-EMBS International Conference on Rehabilitation Robotics (ICORR)*, August 11-14, 2015, Singapore, pp. 13-18.

[16] A. L. Edwards, “Adaptive and Autonomous Switching: Shared Control of Powered Prosthetic Arms Using Reinforcement Learning,” M.Sc. Thesis, University of Alberta, 2016.

[17] R. S. Sutton, A. G. Barto. *Reinforcement learning: An introduction.* MIT Press, Cambridge, 1998.

[18] R. Legenstein, N. Wilbert, L. Wiskott, “Reinforcement learning on slow features of high-dimensional input streams,” *PLoS Comput Biol* vol. 6, no. 8, 2010. e1000894.

[19] R. S. Sutton, J. Modayil, M. Delp, et al., “Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction,” *Proc. the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2–6, Taipei, Taiwan, 2011, pp. 761–768.

[20] H. van Seijen, A. R. Mahmood, P. M. Pilarski, et al., “True online temporal-difference learning,” *Journal of Machine Learning Research* vol. 17, no. 14, pp. 1–40, 2016.

[21] P. Kanerva. *Sparse distributed memory.* MIT Press, Cambridge, 1988.

[22] W. Cheng, W. M. Meleis, “Adaptive Kanerva-based function approximation for multi-agent systems,” *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1361-1364, 2008.

[23] B. Ratitch, D. Precup, “Sparse distributed memories for on-line value-based reinforcement learning,” *European Conference on Machine Learning*. vol. 3201, Springer Berlin Heidelberg, 2004, pp. 347-358.

[24] B. Ratitch, et al., “Sparse distributed memories in reinforcement learning: Case studies,” *Proc. of the Workshop on Learning and Planning in Markov Processes-Advances and Challenges*, 2004, pp. 85-90.

[25] C. A. R. Hoare, “Algorithm 65: Find,” *Communications of the ACM* vol. 4, iss. 7, pp. 321-322, 1961.

[26] W. T. Miller, F. H. Glanz, “UNH CMAC version 2.1: The University of New Hampshire implementation of the Cerebellar Model Arithmetic Computer - CMAC,” Robotics Laboratory Technical Report, University of New Hampshire, Durham, New Hampshire, 1996.

[27] R. S. Sutton, et al., “Fast gradient-descent methods for temporal-difference learning with linear function approximation,” *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 993-1000.