Tightly Integrated Deep Learning and Symbolic Programming on a Single Neuromorphic Chip

Bryan P. Dawson, Jamie K. Infantolino, Manuel M. Vindiola, and John V. Monaco
U.S. Army Research Laboratory
Aberdeen Proving Ground, MD 21005, USA
{bryan.p.dawson.ctr, jamie.k.infantolino.civ, manuel.m.vindiola.civ, john.v.monaco2.civ}@mail.mil

***Abstract*—This work integrates deep learning and symbolic programming paradigms into a unified method for deploying applications to a neuromorphic system. The approach removes the need for coordination among disjoint co-processors by embedding both types entirely on a neuromorphic processor. This integration provides a flexible approach for using each technique where it performs best. A single neuromorphic solution can seamlessly deploy neural networks for classifying sensor-driven noisy data obtained from the environment alongside programmed symbolic logic to processes the input from the networks. We present a concrete implementation of the proposed framework using the TrueNorth neuromorphic processor to play blackjack using a pre-programmed optimal strategy algorithm combined with a neural network trained to classify card images as input. Future extensions of this approach will develop a symbolic neuromorphic compiler for automatically creating networks from a symbolic programming language.**

I. Introduction

Symbolic processing in neural networks remains a challenge. This gap reflects a disparity between connectionist and symbolic models with regard to the central paradox in cognition, namely that humans are capable of both pattern recognition and symbolic processing [**?**]. Although connectionist models, such as deep neural networks, have seen great success in pattern recognition tasks, methods to perform symbolic processing tasks in the same architecture are currently limited.

There exists practical, in addition to theoretical, motivation to perform symbolic processing and deep learning on a single neural architecture. As special-purpose neural processing units (NPU) have begun to emerge, the need for symbolic processing on the same architecture has increased. So far, NPUs have been co-located with von Neumann style (general-purpose) central processing units (CPU) and memory, in which case symbolic processing can be performed by the CPU. The ability of the NPU to perform symbolic processing tasks would alleviate reliance on and communication with the CPU, towards an all-neuromorphic computer. Such a computer carries the benefits of neuromorphic computing, including low power consumption through event-driving computation and massive parallelism.

There are primarily two methods to configure a neural network to perform a symbolic processing task. The first is a *bottom-up* approach whereby general rules are extracted from a set of training examples. This can be performed using machine learning techniques, such as by training a fully differentiable network with external memory [**?**]. The differentiable neural computer (DNC) can solve complex structured tasks that typically require some form of symbolic processing and external memory, such as finding the shortest path in a graph and solving block puzzles. The DNC is also capable of learning algorithmic tasks, such as copying and sorting, using input/output examples.

The second approach to symbolic processing is *top-down*, in which a neural network is systematically configured according to a set of general rules known *a priori*. Several previous works have taken this approach. The NEural Language (NEL) provides a framework to deterministically configure a neural network equivalent to a procedure given in a high-level programming language [**?**]. Neuron activations encode scalar and boolean values as well as lists and stacks, for which a set of primitive functions are used to read and write to. In JaNNeT, neural network compilation is performed using cellular encoding as an intermediary representation, requiring four different transfer functions and asynchronous global dynamics [**?**]. As the only spiking architecture, STICK is a computation framework that uses spike time intervals to encode values and composition of networks to encode functions [**?**].

While a machine learning (bottom-up) approach could potentially be used to learn a complicated algorithm from training examples, such as natural language parsing and inference [**?**], there are some scenarios in which a top-down approach might be preferred. It is generally difficult to learn the special or rare cases of a general rule from training examples alone, as the frequent cases often overwhelm the infrequent ones. Rare training examples may be treated as noise, in which case the learning procedure fails to capture the complete set of rules. If the general rule is tractable and known *a priori*, it may be more practical to instead configure the network directly to encode this information.

This work follows a top-down approach, in which a spiking neural network (SNN) is configured to perform a symbolic processing task, and this network is integrated with a pattern recognition network on a single neuromorphic architecture. Using blackjack as a motivating scenario, a neuromorphic agent that combines deep learning and symbolic processing is described. Blackjack was chosen for its simplicity and the existence of an optimal strategy that remains tractable, as opposed to other games such as chess or Go. The optimal strategy can be described succinctly by a set of rules and programmed in a symbolic language, however there do not yet exist SNN constructs to achieve the same well-defined behavior. We assume that the agent interacts through a visual interface; thus it must recognize cards and make game play decisions based on the card types. Using raw images as input, the complete solution has two components: image classification and game play strategy. Image classification is performed using an existing deep learning neuromorphic solution. Configuration of a SNN for the game strategy is the focus of this work.

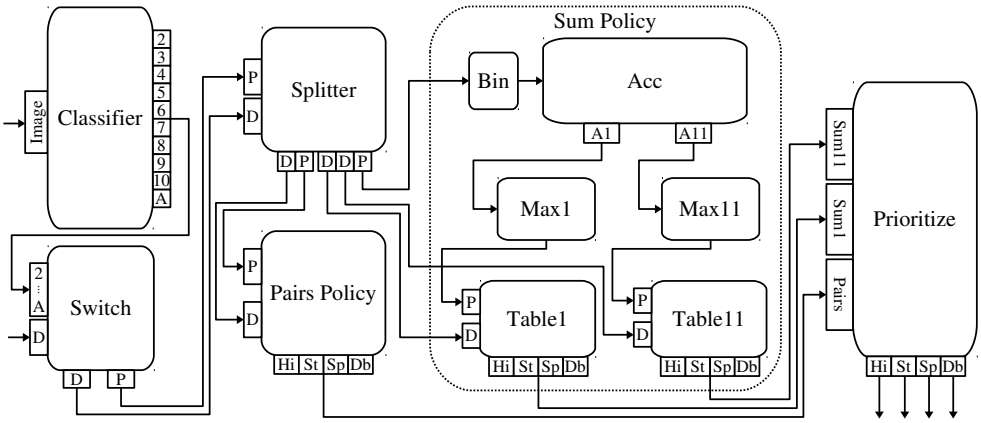


Fig. 1: Architecture overview.

The blackjack agent is implemented on the TrueNorth neuromorphic architecture and deployed to the IBM Neurosynaptic System (NS1e). The NS1e is a hardware implementation of the TrueNorth, capable of simulating 1 million leaky integrate-and-fire (LIF) neurons at 1 KHz while consuming less than 100 milliwatts [**?**]. TrueNorth implements a digital spiking neuron model with over 20 independent parameters [**?**]. There are primarily two modes of programming TrueNorth which reflect the two approaches described above: a machine learning pipeline for object classification [**?**] and a compositional network programming language, the Corelet Programming Environment (CPE) [**?**]. A TrueNorth *corelet* is an abstract building block in which network functionality is abstracted away from input/output specification. In this way, networks that perform complicated functions can be composed using corelets that perform simpler functions.

II. Neuromorphic Blackjack Agent

Blackjack is a card game in which a player competes against the dealer. The objective of the player is to obtain a card total not over 21 that is larger than the dealer's card total. On each hand, the player is dealt two cards and chooses one of four actions as long as the card total has not exceeded the limit: *hit* (receive an additional card), *stand* (receive no more cards), *split* (create two separate hands using two cards with the same value), and *double* (receive only one additional card and then stand).

Basic blackjack strategy, i.e., the policy that minimizes losses to the dealer, depends on the player card total and the dealer's upcard, which is visible to the player at the beginning of each hand [**?**]. The basic strategy can be broken up into two separate lookup tables: the first is a function of card pairs (doubles and A2-A10) and dealer's upcard. The second is a function of the sum of player card values and the dealer's upcard. The second table is only used if none of the entries in the first table are matched.

The neuromorphic blackjack agent makes decisions using the card images, recognized by the pattern recognition network, and optimal game policy, represented symbolically. Gameplay proceeds as follows. At the beginning of the hand, the agent sequentially examines all of it's cards followed by the dealer's upcard. The agent's cards are stored in a memory that accumulates the hand total. After the dealer's card is presented, the agent makes a decision to take one of the four actions. The

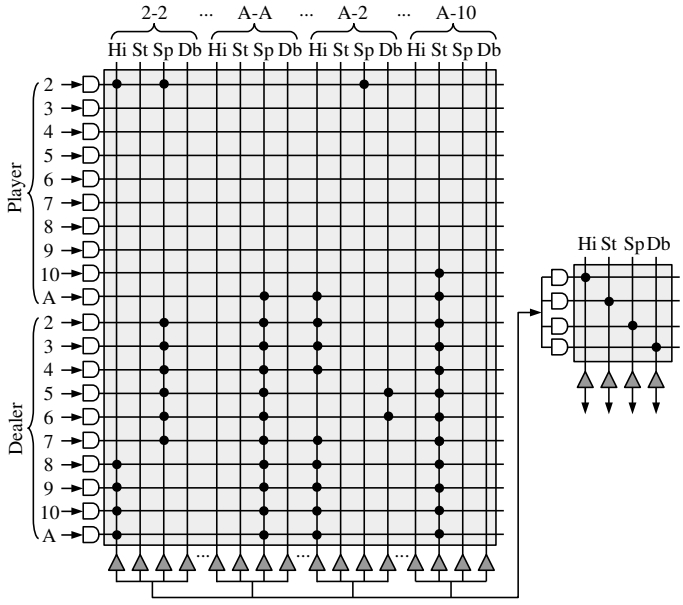


Fig. 2: Pairs policy corelet.

first lookup table (pairs policy) is only used if the agent has two cards matching one of the pairs in the table: two cards with the same value or an Ace and non-Ace card. Otherwise, the sum policy is used. The agent design reflects this, shown in Figure 1.

Card image classification is performed by the *Classifier* corelet which encapsulates a low-precision convolutional neural network (CNN) with trinary ($\{-1, 0, 1\}$) weights and topology that reflects TrueNorth architectural constraints, occupying 719,662 neurons (2812 TrueNorth cores) [**?**]. Images are converted to a spiking representation using a layer of convolutional filters (the transduction layer) with high-precision input and binary output. Neurons are stateless, performing a single classification in a single timestep, 1 ms on the NS1e.

Optimal game policy is implemented by a hierarchical network of 323 neurons. The network is constructed from the bottom up, where each component encapsulates some well-defined behavior in the game logic.

Since all images pass through a single classifier, and dealer

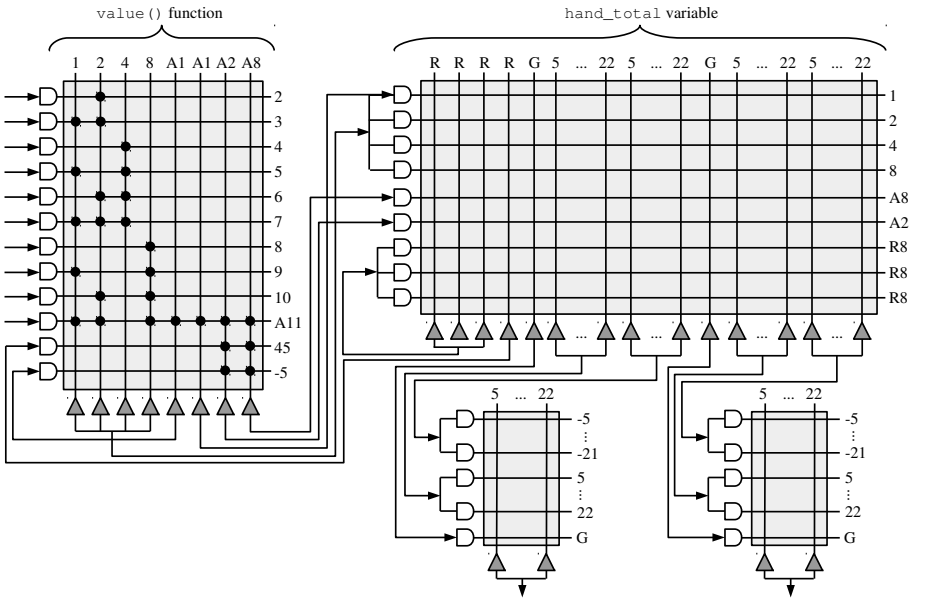
```
input cards
hand_total = 0
action = "bust"

// Loop until the dealer
// card is presented
card = next(cards)
while(type(card) != "dealer")
    hand_total += value(card)
    card = next(cards)
dealer_value = value(care)

// Lookup action by hand
// total and dealer card
if (hand_total==5
      && dealer_value==2)
    action = "hit"
…
else if (hand_total==20
          && dealer_value==Ace)
    action = "stand"

output action
```

(a) Sum policy pseudocode.



(b) Sum policy corelets.

Fig. 3: Sum policy corelets and pseudocode.

cards are handled differently than player cards, branching must occur within the symbolic processing network. On a parallel architecture, such as TrueNorth, this is primarily achieved through recurrent inhibitory connections. The *Switch* corelet routes the recognized card values to either the player (P) or dealer (D) pipelines. Due to a fan-out constraint on TrueNorth, in which neurons can have up to 256 postsynaptic connections all residing on a single core, a *Splitter* corelet sends classification decisions to each of the corelets implementing the basic game strategies. The *Pairs policy* corelet performs a table lookup using the player and dealer card values, producing no output if no match is found. The *Sum policy* corelet maintains a variable that represents the player's hand total, which is used in the second lookup table.

Lookup in the pairs policy table is performed by spikes transmitted to axons that correspond to player and dealer card values, shown in Figure 2. Neurons are grouped by the pair type, e.g., A-A if two Aces are seen by the agent, and each group contains one of four actions. Since only one pair can be activated in a single step, an element-wise *or* operation is applied to each action across the 10 groups to obtain 4 neuron outputs which correspond to the 4 game actions.

Neurons in the pairs policy corelet are configured to spike after having received 3 consecutive input spikes. Thus, the leak is set to -1, synaptic weights set to 2, and threshold set to 3. Synaptic connections, shown in Figure 2, ensure that only one action neuron spikes when the dealer card is observed after the pair cards. For example, consider the card sequence $\{2,2,10\}$ for player, player, and dealer cards, respectively. For the first two player cards, spikes are transmitted to the '2' player card axon, raising membrane potential of the two connected neurons to 2. Since the dealer card is 10, a spike is then sent to the '10' dealer axon, causing only the 'Sp' neuron to spike. The agent then decides to split.

The pairs policy corelet highlights a fundamental difference between sequential and parallel computations. In a sequential approach, if a condition is not met then the corresponding instruction block is never executed. However on a parallel architecture, such as a NPU, all instruction blocks are evaluated simultaneously. Therefore, an explicit mechanism is needed to suppress the effects of instruction blocks that are not applicable. In the blackjack agent, this is implemented by a recurrent inhibitory connection which disables output from the pairs policy corelet after more than 2 player cards are observed, after which the sum policy corelet takes precedence.

The sum policy corelet is composed of six corelets that perform different functions to implement the pseudocode shown in Figure 3a. Card values are sequentially presented, and a total of the hand card values is maintained. This total, along with the dealer card value, is used to determine the optimal action. The *Bin* and *Acc* corelets, shown in Figure 3b, work together to maintain a sum of card values in the current hand. Since Aces can be treated as either 1 or 11, this total is maintained for each scenario, reflected by the parallel processing pipelines following the Acc corelet.

The Bin corelet performs a decimal to binary conversion, representing each card value as a sum of powers of two, corresponding to the axon types of Acc. Each card value activates a number of neurons in the Acc corelet equal to the card value. This is performed for both the Ace 1 and Ace 11 scenarios. Following the Acc corelet, the *Max1* and *Max11* corelets compute the maximum active neuron by inhibiting neurons less than the max. The maximum neuron is passed to the *Table1* and *Table11* corelets, which lookup the optimal decision using the card values and the dealer's upcard, similar to the Pairs policy corelet.

Output from both policies is sent to the *Priority* corelet that determines the decision priority. The pairs policy takes precedence, and if it outputs a spike, this inhibits any decision from the sum policy tables. Otherwise the sum policy output is

used. The Table11 decision takes priority unless the hand total is over 21, in which case the decision from Table1 is used, whereby an Ace would be treated as value 1 instead of 11. If the hand total remains over 21 when treating the Ace as a 1, then the round is lost and no decision is output.

III. Results

Image recognition and game policy performances are evaluated separately since these are separate components in the blackjack agent. Performance of each component is given by classification accuracy (ACC).

The CNN must learn to recognize the card value (10 classes total) and not the card itself. Training is performed using 10k card images scaled to 32x32 and augmented with affine transformations, as shown in Figure 4. Target labels are given by the card value (2-10 and Ace, where all face cards are 10). The CNN is trained for 10k epochs with 0.2 learning rate. The resulting network achieves 99% ACC on the training dataset and 97% ACC on a separate test dataset with 10k card images.

Game policy ACC is given by the ability of the policy network to output the optimal action given card value as input, where optimal actions are defined by the basic blackjack strategy tables. Since the symbolic processing approach encodes exactly the optimal strategy, it achieves perfect accuracy. For comparison, we also evaluate a machine learning approach.

A 3-layer fully-connected high-precision neural network is trained using simulated blackjack hands. For the training dataset, 100k blackjack hands with the optimal actions are generated. Out of the 100k generated hands, there are 10,201 unique training instances, i.e., unique player and dealer card value combinations. In this dataset, the 4 target classes are extremely unbalanced, having the following ratios of each action: 0.809 (stand), 0.167 (hit), 0.021 (split), and 0.003 (double). To compensate for the unbalanced classes, each sample is weighted in training by $1 - r_c$, where r_c is the ratio of class c.

The player and dealer hands are encoded as a length-20 vector, where each element corresponds to the normalized card count of each card value. The fully connected network has an input layer of size 20 (corresponding to the card count of player and dealer cards), hidden layer of size 200, and output layer of size 4 (corresponding to the 4 actions). The network is trained for 10k epochs using Adam optimization with 0.01 learning rate and cross-entropy loss function.

The resulting network achieves 99.59% accuracy on the training dataset. Perfect accuracy is difficult to achieve using a machine learning approach due to the unbalanced classes and presence of rare samples. This is verified by examining the action ratios in the 42 missed training samples: 0 (stand), 0.071 (hit), 0.476 (split), and 0.452 (double).

To test the generalization ability of the policy network, an additional 10k simulated hands are generated. Of these 10k hands, there are 255 unique player/dealer card values which are not seen in the training dataset. This set of 255 hands contains 227 stand and 27 hit actions. Ability to generalize to unseen card value combinations is evaluated by performance on the 255 novel hands. The network has a classification accuracy of 99.61% (254 out of 255 correctly classified).

IV. Conclusion and Future Work

This work introduced a symbolic processing approach to basic blackjack strategy, implementing a blackjack agent in spiking neurons. The all-neuromorphic solution integrates symbolic processing for decision making with a spiking CNN for image recognition. In cases where optimal decisions are captured by a small finite set of rules, symbolic processing is preferable to a machine learning approach, whereby rare samples and unbalanced class distributions are difficult to capture. Encoding the rules directly ensures the decision-making component of the agent has known behavior over all possible inputs, leaving errors only to the image recognition.

Fig. 4: Example card images.

Several constructs were introduced in order to implement some symbolic processing paradigms in spiking neurons. These include: resetting to an initial state when no inputs are received as a form of loop iteration; recurrent inhibitory connections as a form of control flow and branching; and a stateful population code as a form of discrete memory for integer variables.

The solution has been handcrafted utilizing a unique combination of spiking networks. Future work will generalize the above constructs with an ultimate goal of building a neuromorphic compiler. Such a tool would allow seamless integration of deep learning with the symbolic processing tasks required for neuromorphic device operation, such as input/output management, memory allocation, and other rule-based tasks in which well-defined behavior is warranted.

References

[1] P. Smolensky, G. Legendre, and Y. Miyata, "Principles for an integrated connectionist/symbolic theory of higher cognition," 1992.

[2] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[3] H. T. Siegelmann, "Neural programming language," in *AAAI*, 1994, pp. 877–882.

[4] F. Gruau, J.-Y. Ratajszczak, and G. Wiber, "A neural compiler," *Theoretical Computer Science*, vol. 141, no. 1, pp. 1–52, 1995.

[5] X. Lagorce and R. Benosman, "Stick: Spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony," *Neural computation*, 2015.

[6] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[7] A. S. Cassidy *et al.*, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–10.

[8] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *arXiv preprint arXiv:1603.08270*, 2016.

[9] ——, "Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–10.

[10] R. R. Baldwin, W. E. Cantey, H. Maisel, and J. P. McDermott, "The optimum strategy in blackjack," *Journal of the American Statistical Association*, vol. 51, no. 275, pp. 429–439, 1956.