ReLU early in training: The function is activated around the nonlinearity, but does not reach the saturating region that occurs with larger activations. However, later training also takes advantage of the saturating region. In this manner, the same activation function can be used in two ways: presumably to keep the gradients from vanishing early, and to commit to decisions later. This result challenges the common approach in deep learning design, and demonstrates the power of neuroevolution in metalearning good designs.

In sum, surrogate optimization techniques make it possible to scale up neuroevolution metalearning; in doing so, it is possible to identify principles that would be difficult for human designers to discover.

11.3.6 Synergies

Perhaps the most important future direction in evolutionary metalearning is to discover and utilize synergies between the different aspects of the learning system design. For instance, the best performance was reached by optimization activation functions for the specific architecture; It might be possible to optimize the architecture simultaneously to emphasize this effect.

Simply running evolution on all these design aspects simultaneously is unlikely to work; the search space would be prohibitively large. Similarly, adding more outer loops to the existing process (where supervised learning is the inner loop and metalearning is the outer loop) is likely prohibitive as well. However, it might be possible to alternate evolution of different aspects. Better yet, techniques from bilevel (or multilevel) optimization could be useful—the idea is to avoid full inner-outer loop structure, but instead use e.g. surrogate models to evaluate outer loop innovations (Sinha et al. 2014; Liang and Miikkulainen 2015).

A practical approach is to simply add constraints, and search in a smaller space. A first such step was already taken in the EPBT system (J. Liang et al. 2021), which combines hyperparameter tuning, loss-function optimization, and population-based training (PBT) into a single loop. That is, hyperparameters and loss functions are evolved at the same time as the networks are being trained. Hyperparameter tuning is limited to those that do not change the structure of the networks (e.g. learning rate schedules) so that they can be continuously trained, even when the hyperparameters change. Similarly, loss-function optimization is limited to TaylorGLO coefficients (J. Liang et al. 2021) that can be changed while training is going on. Even so, the simultaneous evolution and learning was deceptive, and needed to be augmented with two mechanisms: quality-diversity heuristic for managing the population and knowledge distillation to prevent overfitting. The resulting method worked well on optimizing ResNet and WideResnet architectures in CIFAR-10 and SVHN, but also illustrates the challenges in taking advantage of synergies of metalearning methods.

Similarly, promising results were obtained in an experiment that compared human design with evolutionary metalearning (Miikkulainen, Meyerson, et al. 2021). Using the same datasets and initial model architectures, similar computational resources, and similar development time, a team of data scientists and an evolutionary metalearning approach developed models for age estimation in facial images (Figure 11.6). The evolutionary metalearning approach, LEAF-ENN, included optimization of loss functions (limited to linear combinations of MSE and CE), learning hyperparameters, architecture hyperparameters, and data augmentation methods. Evolution discovered several useful principles that the data scientists were not aware of: Focusing data augmentation to regions that mattered most, and

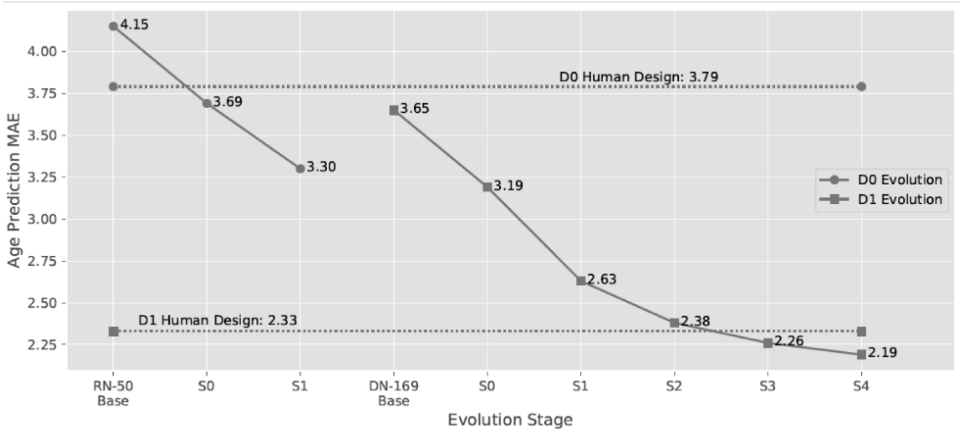


Figure 11.6: **Utilizing Metalearning Synergies to Beat Human Designers.** In this natural experiment, human experts and metalearning were both working at the same time to improve the accuracy of age estimation from facial images. In two datasets (D0 and D1), LEAF-ENN evolutionary metalearning (Miikkulainen, Meyerson, et al. 2021) was able to discover models that performed better than those simultaneously designed by human machine learning engineers. The humans optimized the ResNet-50 architecture for D0 and EfficientNet-B8 for D1. The evolutionary runs progressed in stages: In D0, ResNet-50 (S0) was expanded to Densenet 169 (S1); in D1, DenseNet-169 (S0) was expanded to DenseNet-201 (S1) and trained longer (S2), then expanded to EfficientNet-B6 (S3), and ensembling (S4). At the same time, evolution optimized learning and architecture hyperparameters, data-augmentation methods, and combinations of loss functions. The approach discovers and utilizes synergies between design aspects that are difficult for humans to utilize. The final accuracy, MSE of 2.19 years, is better than typical human accuracy in age estimation (3-4 years). (Figure from Miikkulainen, Meyerson, et al. 2021)

utilizing flips only horizontally across the face; utilizing different loss functions at different times during learning; relying mostly on the output level blocks of the base models. With both datasets, the eventual accuracy of the metalearned models was significantly better than that of the models developed by the data scientists. This result demonstrates the main value of evolutionary metalearning: It can result in models that are optimized beyond human ability to do so.

11.4 Neuroevolution of Neuromorphic Systems

Neuromorphic computing, i.e. spiking neural networks designed to be implemented in hardware, is a promising new area for neuroevolution. Such networks need to be energy efficient, and therefore compact and complex, with many design parameters that need to be optimized and customized. This general area is reviewed in this section, several examples are given, and future opportunities are outlined.

11.4.1 Neuromorphic computation

Neuromorphic computation, a field focusing on hardware implementation of neural networks, is a burgeoning field with a long history Catherine D. Schuman et al. 2017; James et al. 2017. There are several motivations: neuromorphic circuits offer parallel computation that results in real-time performance, they can be fault tolerant, such systems may learn online, and they can be used to evaluate hypotheses in neuroscience. However, energy efficiency has gradually emerged as the main goal over the years. Most of the implementations are based on spiking neurons, as opposed to neurons that are activated with continuous values representing firing rates. Such spikes require very little power, resulting in energy savings of several orders of magnitude. As computation and AI move to the edge, i.e. sensors and actuators in the field, power becomes a primary constraint on computation, and neuromorphic designs offer a possible solution.

Although the full power of neuromorphic computing is still a way off, substantial hardware designs have already been manufactured that demonstrate its potential. IBM's TrueNorth (Akopyan et al. 2015) is one and Intel's Loihi (Davies et al. 2018) another, both with 1M spiking neurons. It is therefore possible to generate neuromorphic methods and have them run on these actual physical devices. However, the field is much more broad, and many methods are proposed for a wide variety of conceptual devices. What makes the field particularly interesting is that the resulting neural network architectures and algorithms are often new and different, and not just hardware approximations of existing simulated neural networks such as backpropagation on a three-layer feedforward network. In that sense, neuromorphic computing is driving innovation in neural networks.

Biology is the source for many such ideas in that many neuromorphic designs are inspired by neuroscience. Some of them are also plausible, intended to capture principles of biology closely enough to test hypotheses about it. For instance, spiking neurons can be implemented at the level of Hodgkin-Huxley equations, i.e. the electrochemical balance of compartments in the neural membrane. Such implementations allow studying single neuron computation well. Other models like the Izhikevich neuron aim to replicate the bursting and spiking behavior with simpler computation. The leaky-integrate-and-fire model simplifies them further into integrating the spikes in each synapse over time (with decay), and firing when a threshold is exceeded.

Learning in spiking networks is often based on spike-timing-dependent plasticity (STDP). If a postsynaptic neuron fires shortly after the presynaptic neuron, it is possible that the presynaptic firing caused the postsynaptic firing, and the connection is strengthened. Conversely, if the postsynaptic neuron fires shortly before the presynaptic neuron, the connection is weakened. In this sense, STDP is a time-based refinement of the Hebbian learning principle, i.e. that neurons that fire together wire together.

Note that STDP is an unsupervised learning method: there are no targets or gradients, but simply an adaptation principle that applies to each connection independently. To make learning more goal directed, learning mechanisms that approximate backpropagation have also been proposed. A practical approach along these lines is to first train a standard simulated firing-rate backpropagation network off-line, and then convert the resulting network into a spiking neural network equivalent (Lu and Sengupta 2022). Such implementations can achieve power savings, however, they do not take into account or utilize any further properties of hardware systems such as delays and timing.

Thus, LIF neurons with an STDP learning rule are the most common implementation of neuromorphic architectures. It has low energy requirements and is event driving, and is thus suitable for many architectures and applications. The designs include hardware-constrained circuits such as those of provided by TrueNorth and Loihi, brain-inspired circuits, feedforward neural networks, and convolutional networks.

Interestingly, reservoir computing architectures have emerged as a popular design as well, as a way to extend neuromorphic computing to time-varying problems. A reservoir is a recurrent network that generates a time-varying signal that can then be processed with a feedforward network, making it possible to recognize time series, or generate time-varying behavior such as locomotion. The reservoir is initialized with random neurons and connection weights, and they are not modified, making them particularly useful for neuromorphic computation, for instance through a memristor implementation.

The designs are often evaluated with standard machine learning tasks. However, the ultimate applications range from vision and sensing to robotics and control. While it may be possible to achieve better performance through e.g. deep learning, some of such tasks need to be performed in physical devices at the edge with little power available. For instance, visual and auditory signal detection, brain-machine interfaces, and central pattern generators for locomotion may be such applications in the future.

Because neuromorphic designs are unique and varied, there is a great opportunity to optimize them through neuroevolution, as will be discussed next.

11.4.2 Evolutionary optimization

Neuromorphic designs include many dimensions that can be optimized towards several different objectives. For instance, the synaptic efficacy, activation decay, firing threshold, refractory period, and transmission delay of LIF neurons can be adjusted; the connectivity of the network can be changed, and the timing and extent of plasticity modified. Performance in the task is one objective, energy consumption, size, and complexity of the network others.

Optimization of neuromorphic designs is thus a compelling application for neuroevolution. First, gradients are often difficult to obtain with neuromorphic architectures and in domains where they would be applied. Neuroevolution does not depend on gradients, and it can therefore be used to implement supervised learning. It can therefore be used to extend neuromorphic computing to many engineering applications. Second, while many applications can be built with deep-learning designs, they are too large to be effectively deployed at the edge. Neuroevolution often results in compact designs that are space and energy-efficient. Third, it is possible to optimize the designs towards multiple objectives simultaneously, including performance, energy consumption, size, complexity, and specific hardware restrictions. Fourth, evolution can be extended to include hardware design as well, leading to the co-design of the hardware and the algorithms that run on it. Fifth, while such optimization is compute-intensive, it can be done offline, taking advantage of existing hardware simulators.

Many approaches to neuromorphic neuroevolution have been proposed, targeting different aspects of hardware design. For instance, the Evolutionary Optimization of Neuromorphic Systems (EONS; C. D. Schuman et al. 2020) framework, the idea is to evolve a flexible structure of nodes and edges, as well as many of their parameters such as the connection weights, the time delay on the connections and neurons, activation thresholds, and leak rate.

The system starts with a randomly initialized population represented as lists of nodes with IDs and parameters; as usual, each generation of individuals is evaluated in the task, and crossover and mutation applied to selected parents. The method is thus similar to NEAT but includes many more parameters that are specific to neuromorphic hardware. Note EONS is also generic and can be adjusted to different kind of hardware. Evolution is simple enough so that it can be implemented in hardware at the edge, but usually it is done off-line using a hardware simulator.

EONS has been tested on several standard benchmarks. For instance, in classification tasks from the UCI database it resulted in simpler and more accurate solutions than standard neuromorphic designs. Evolution also adapted the solutions to hardware constraints such as the number of bits used to encode the weights. With a secondary objective to minimize the number of nodes and connections, in addition to accuracy, it produced a range of tradeoffs. Such experiments thus demonstrate the viability of hardware/algorithm co-design.

11.4.3 Examples

A particularly interesting application of EONS is to optimize reservoir architectures. Although reservoir networks usually have a fixed structure and weights, and learning is only done on the feedforward network that receives input from the reservoir, evolution can be used to optimize the reservoir itself. Such optimization may include tuning its hyper-parameters, connectivity, and even the weights. This optimization can be done before the learning in the feedforward network, the feedforward network can be evolved directly at the same time, or the trained performance of the feedforward network can be used as fitness for reservoir evolution (Reynolds, Plank, and Schuman 2019; Iranmehr et al. 2019). Note that even though these optimizations were developed for neuromorphic computing, they apply to firing-rate versions of reservoir networks as well.

Evolutionary optimization of reservoir networks was shown to result in better performance than e.g. the usual grid search for good designs. A particularly illustrative application was to classify radar pulse sequences in order to identify movements of free electrons in the ionosphere. The performance was close to other machine learning methods; the low-power implementation may make it possible to deploy actual physical solutions even in satellites.

Along the lines of building better detectors, radiation anomaly detection is a similar potential killer app for neuromorphic computing (Ghawaly et al. 2022; Ghawaly et al. 2023). As part of nuclear nonproliferation research, the challenge is to detect hidden gamma-ray sources in an urban environment. This is a difficult task because the detection needs to be done by moving through the normal accessible environment, and background radiation varies significantly. Potential sources need to be detected as anomalies in the observed levels that are very noisy, triggering an alarm for further study. As usual in such tasks, the true positive rate needs to be increased while keeping the false alarm rate as low as possible.

The task is well defined, with ANSI standards for acceptable detection levels for different types of radiation, as well as standard datasets through which performance can be evaluated. The best current approaches are based on machine learning: In a recent competition by US Department of Energy, nine of the ten best methods were based on neural networks and similar methods (Department of Energy 2019). However, such methods consume a lot of energy, which limits their applicability in the field. Neuromorphic computing is a viable alternative, offering real-time detection with much less energy usage.

In a series of experiments, EONS was set to design a network for this task. As usual, EONS optimizes the topology and weights of the network, but also several hyperparameters such as the encoding for the spikes, the delays on neurons and connections, neuron leakage, spiking thresholds, and short-term memory between inferences. A threshold on the spiking rate was used to trigger alarms, adjusted to an acceptable false-alarm rate. The resulting designs had a sensitivity of about half of a compute-intensive PCA-based spectral analysis method; thus, the energy savings still come with a cost. However, they met several ANSI standards and performed better than a common kσ baseline method, suggesting that it may already be possible to deploy them in conditions where energy is at a premium. Most interestingly, the best designs leveraged both spatial and temporal features in the signal, taking advantage of short-term memory. Also, while the leakage rate was not important, spike encoding mattered, with the number of spikes generated being the most powerful. Such insights are useful in neuromorphic computing in particular because they can drive co-design of the hardware, suggesting what elements are most useful to implement.

While low energy consumption is important in sensing, it can also be crucial for actuators at the edge. For instance for autonomous cars, computing consumes 40 to 80% of the power required for the control system (Baxter et al. 2018). Neuromorphic computing could reduce this requirement significantly, thus extending battery life. This idea was tested in the F1Tenth system, which is a 1/10 scale simulation and physical implementation of a Formula One race car (Figure 11.7; C. Schuman et al. 2022).

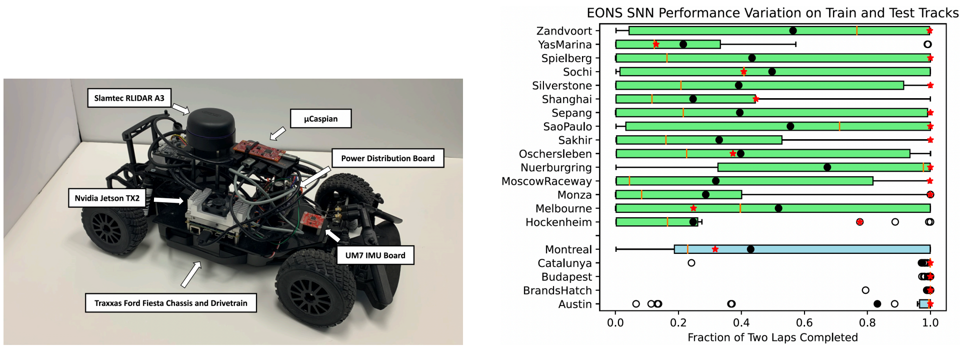
Compared to imitation learning based on hand-designed waypoints, neuroevolution resulted in architectures that performed better, although they took longer to train. This improvement was due to discovering customized structure in the network; without it, the results were not as good. Interestingly, the discovered network structures were also smaller than the best hand-designed ones for imitation learning and evolution without structure optimization. Since smaller networks are easier to deploy at the edge, with less energy and space needed, neuroevolution again provides solutions that make physical hardware implementations more realistic.

As a proof of concept, the evolved controllers were implemented on a circuit board on a physical car and tested on a physical track setting. While the performance dropped some, as is usual in transfer from simulation to the physical world, the driving was largely successful, demonstrating actual neuromorphic control at the edge.

11.4.4 Future directions

Neuromorphic neuroevolution is a relatively new opportunity. The motivation from energy consumption is compelling, and there are several encouraging results, but the performance still needs to be improved and killer applications identified and implemented. However, there are several ways in which it can be further developed and improved, which makes it an interesting area for neuroevolution in the future.

While neural architecture search at the level of deep learning has become rather difficult, due to extremely large networks and a few dominant architectures, the demands of neuromorphic computing are almost exactly the opposite. The networks need to be small, often recurrent, and customized. There are many hyperparameters beyond the standard neural



(*a*) F1TENTH physical car (*b*) Performance on simulated tracks

Figure 11.7: **Evolving a neuromorphic race car controller.** Neuromorphic control can reduce the energy consumption of both sensing and actuation, which is crucial in applications at the edge, such as self-driving cars. (*a*) The physical platform was a F1TENTH robotic vehicle, intended to represent 1/10 of a Formula One race car. The controller was implemented on the μCaspian neuromorphic development board. (*b*) Performance of the neuroevolved controller on various simulated race tracks. The bottom five were used for training and the two 15 for testing. Performance was measured in x-axis as the fraction of two laps completed. The box plots show the distribution of the best networks found in 30 evolution runs; the red star is the network with the best average performance. Some tracks are more difficult than others, but evolution discovered networks that performed well on all of them, and the best network on nine of the 15. When transferred to a real-world track (not shown), performance was not as good as in simulation, but still demonstrated a practical implementation of a neuromorphic controller at the edge. Figures from C. Schuman et al. 2022

network ones, such as delays, leakage, thresholds, spike encoding, and short-term memory. The designs are constrained by restrictions and properties of the actual hardware where they will eventually run.

As a result, there are many opportunities for neuroevolution. As with deep neuroevolution, the overall topology, i.e. neurons and their connectivity, is important, but also because the networks are compact, the connection weights can be optimized directly. The hyperparameters make the optimization problem complex but also provide an opportunity for further improvement and customization. New learning mechanisms may be developed through neuroevolution, improving upon STDP and perhaps providing practical methods for online supervised learning. Information about not only spike timing across an individual synapse may be used, but also timing across multiple synapses and their history. There may be opportunities to leverage imperfections and other properties of physical devices, and even interactions between them like coupling.

Perhaps the most exciting opportunity is the co-design of neuromorphic architectures and hardware. It may be possible to establish a cooperative coevolutionary mechanism that modifies both aspects simultaneously, resulting in an optimal fit not unlike the brain and behavior coevolution discussed in Section 14.5. There are several constraints on both sides on size, communication, and complexity, but they can possibly be incorporated into the search and evaluation mechanisms. As a result, entirely new architectures and algorithms may be discovered, and customized to the task to be solved. Such an approach may indeed prove crucial in moving more computing to the edge in the future.

11.5 Chapter Review Questions

1. **Complex System Design:** What are the main advantages of using evolutionary optimization for designing complex systems, such as VLSI circuits or neural networks, compared to traditional human-driven approaches?
2. **Bilevel Neuroevolution:** How does bilevel neuroevolution enhance the performance of neural networks? Why is surrogate modeling crucial in this process?
3. **Loss Function Optimization:** Discuss how evolutionary techniques discovered the ”Baikal Loss” function, and its impact on regularization and robustness in neural networks.
4. **Activation Functions:** Explain the role of activation functions in neural network performance and how evolutionary approaches like Pangaea can customize activation functions for specific architectures and tasks.
5. **Data Augmentation:** Describe how evolutionary optimization can be applied to data augmentation. Provide examples of transformations discovered during such processes.
6. **Learning Methods:** What are the key findings of the AutoML-Zero system? How does it demonstrate the potential of evolutionary approaches in discovering fundamental learning algorithms?
7. **Synergies in Metalearning:** Why is it challenging to optimize multiple aspects of neural network design simultaneously? How can these challenges be addressed in evolutionary metalearning to outperform human-designed models?
8. **Neuromorphic Computation:** What are the key advantages of neuromorphic computing, particularly in the context of energy efficiency and edge applications? How do spiking neural networks differ from traditional neural networks in achieving these goals?
9. **Evolutionary Optimization in Neuromorphic Systems:** How does the Evolutionary Optimization of Neuromorphic Systems (EONS) framework adapt standard neuroevolution methods for neuromorphic hardware? What unique parameters does it optimize compared to traditional neural networks?
10. **Applications and Future Directions:** Discuss how neuromorphic neuroevolution has been applied in tasks such as reservoir optimization, radiation anomaly detection, and autonomous vehicle control. What are some future opportunities and challenges in combining hardware and algorithm co-design in neuromorphic systems?