SUPPLEMENTARY INFORMATION "EFFICIENT RECURRENT ARCHITECTURES THROUGH ACTIVITY SPARSITY AND SPARSE BACK-PROPAGATION THROUGH TIME"

Anand Subramoney[1], **Khaleelulla Khan Nazeer**[2], **Mark Schöne**[2], **Christian Mayr**[2,3], **David Kappel**[1]
[1] Institute for Neural Computation, Ruhr University Bochum, Germany
[2] Faculty of Electrical and Computer Engineering, Technische Universität Dresden, Dresden, Germany
[3] Centre for Tactile Internet with Human-in-the-Loop (CeTI)
{anand.subramoney,david.kappel}@ini.rub.de
{khaleelulla.khan_nazeer,mark.schoene,christian.mayr}@tu-dresden.de

A DERIVATION OF CONTINUOUS TIME VERSION OF GRU

In this section we derive the continuous-time version of the GRU model. Note that our definition of the GRU differs from the original version, presented in (Cho et al., 2014), by inverting the role of the $\mathbf{u}^{\langle t \rangle}$ and $1 - \mathbf{u}^{\langle t \rangle}$ terms in the updates equations for $\mathbf{y}^{\langle t \rangle}$ (a change in sign). This substitution does not change the behavior of the model but simplifies the notation in the continuous-time version of the model.

We first rewrite the dynamics of a layer of GRU units at time step t from Eq. (1) of the main text, separating out the input and recurrent weights:

$$\mathbf{u}^{\langle t \rangle} = \sigma\left(\mathbf{U}_u \mathbf{x}^{\langle t \rangle} + \mathbf{V}_u \mathbf{y}^{\langle t-1 \rangle} + \mathbf{b}_u\right), \quad \mathbf{r}^{\langle t \rangle} = \sigma\left(\mathbf{U}_r \mathbf{x}^{\langle t \rangle} + \mathbf{V}_r \mathbf{y}^{\langle t-1 \rangle} + \mathbf{b}_r\right),$$
$$\mathbf{z}^{\langle t \rangle} = g\left(\mathbf{U}_z \mathbf{x}^{\langle t \rangle} + \mathbf{V}_z \left(\mathbf{r}^{\langle t \rangle} \odot \mathbf{y}^{\langle t-1 \rangle}\right) + \mathbf{b}_z\right), \quad \mathbf{y}^{\langle t \rangle} = \mathbf{u}^{\langle t \rangle} \odot \mathbf{z}^{\langle t \rangle} + (1 - \mathbf{u}^{\langle t \rangle}) \odot \mathbf{y}^{\langle t-1 \rangle}, \tag{S1}$$

we can write this as

$$\mathbf{y}^{\langle t \rangle} - \mathbf{y}^{\langle t-1 \rangle} = -\mathbf{u}^{\langle t \rangle} \odot \mathbf{y}^{\langle t-1 \rangle} + \mathbf{u}^{\langle t \rangle} \odot \mathbf{z}^{\langle t \rangle}. \tag{S2}$$

Note that $\mathbf{u}$ here is equivalent to $\tilde{\mathbf{u}} = 1 - \mathbf{u}$ used in the standard GRU model. Eq. (S2) is in the form of a forward Euler discretization of a continuous time dynamical system. Defining $\mathbf{y}(t) \equiv \mathbf{y}^{\langle t-1 \rangle}$, we get $\mathbf{r}(t) \equiv \mathbf{r}^{\langle t \rangle}, \mathbf{u}(t) \equiv \mathbf{u}^{\langle t \rangle}, \mathbf{z}(t) \equiv \mathbf{z}^{\langle t \rangle}$. Let Δt define an arbitrary time step. Then Eq. (S2) becomes:

$$\mathbf{y}(t + \Delta t) - \mathbf{y}(t) = -\mathbf{u}(t) \odot (\mathbf{y}(t) + \mathbf{z}(t))\, \Delta t \tag{S3}$$

Dividing by Δt and taking limit $\Delta t \to 0$, we get:

$$\dot{\mathbf{y}}(t) = -\mathbf{u}(t) \odot (\mathbf{y}(t) - \mathbf{z}(t)), \tag{S4}$$

where $\dot{\mathbf{y}}(t) \equiv \frac{d\mathbf{y}(t)}{dt}$ is the time derivative of $\mathbf{y}(t)$.

B FULL DETAILS OF THE CONTINUOUS TIME EGRU

In this section we establish the continuous time version of the EGRU model. To describe the event generating mechanism and state dynamics it is convenient to express the dynamical system equations in therms of the activations $\mathbf{a}_{\mathrm{X}}$.

We first rewrite Eqs. (3) & (4) of the main text, as:

$$f_{a_{\mathrm{X}}} \equiv \tau_s \dot{\mathbf{a}}_{\mathrm{X}} + \mathbf{a}_{\mathrm{X}} + \mathbf{b}_{\mathrm{X}} = \mathbf{0}, \quad \mathrm{X} \in \{u,\ r,\ z\} \tag{S5}$$
$$f_c \equiv \tau_m \dot{\mathbf{c}}(t) + \mathbf{u}(t) \odot (\mathbf{c}(t) - \mathbf{z}(t)) = \tau_m \dot{\mathbf{c}}(t) - F(t, \mathbf{a}_u, \mathbf{a}_r, \mathbf{a}_z, \mathbf{c}) = 0. \tag{S6}$$

We write the event transitions for $\mathbf{c}$ at network event $e_k \in \mathbf{e}$, $e_k = (s_k, n_k)$, where s_k are the continuous (real-valued) event times, and n_k denotes which unit got activated, and using the superscript $.^-$ ($.^+$) to the quantity just before (after) the event, as:

$$c^-_{n_k}(s_k) = \vartheta_{n_k}, \qquad c^+_{n_k}(s_k) = 0, \qquad c^+_m(s_k) = c^-_m(s_k). \tag{S7}$$

where $m \neq n_k$ denotes all the units connected to unit n_k that are not activated. At the time of this event, the activations $a_{\mathrm{x},m}$ ($\mathrm{x} \in \{u, r, x\}$) experiences a jump in its state value, given by:

$$a^+_{u,m}(s_k) = a^-_{u,m}(s_k) + v_{u,mn_k} \times c^-_{n_k}(s_k)\,, \tag{S8}$$
$$a^+_{r,m}(s_k) = a^-_{r,m}(s_k) + v_{r,mn_k} \times c^-_{n_k}(s_k)\,, \tag{S9}$$
$$a^+_{z,m}(s_k) = a^-_{z,m}(s_k) + v_{z,mn_k} \times r_{n_k} \times c^-_{n_k}(s_k)\,, \tag{S10}$$
$$a^+_{\mathrm{x},n_k}(s_k) = a^-_{\mathrm{x},n_k}(s_k)\,. \tag{S11}$$

External inputs also come in as events $\tilde{e}_k \in \tilde{\mathbf{e}}$, $\tilde{e_k} = (s_k, i_k)$, where s_k are the continuous (real-valued) event times, and i_k denotes the index of the input component that got activated. Only the activations $a_{\mathrm{x},l}$ for the l-th unit experience a transition/jump on incoming external input events, as follows:

$$a^+_{\mathrm{x},l}(s_k) = a^-_{\mathrm{x},l}(s_k) + u_{\mathrm{x},ln_k} \times x_{i_k}(s_k)\,, \tag{S12}$$

where $x_{i_k}(s_k) = (\mathbf{x}(s_k))_{i_k}$ is the i_k-th component of the input $\mathbf{x}$ at time s_k. The internal state $\mathbf{c}$ remains the same on the external input event. That is, $c^+_l = c^-_l$.

C Details for Proof of Proposition 2

Using basic matrix algebra, it can be shown that both $\frac{\partial F}{\partial \mathbf{c}}$ and $\frac{\partial F}{\partial \mathbf{a}_\mathrm{x}}$ simplify to a diagonal matrix due to the independence of a notional unit i from unit j in the forward dynamics in Eqns. (3), (4). Therefore, Eqn. (8) can be written as the following for unit i:

$$\frac{\partial \dot{c}_i}{\partial c_i}\lambda_{c,i} - \tau_m \dot{\lambda}_{c,i} = 0\,, \qquad \lambda_{a_\mathrm{x},i} + \frac{\partial \dot{c}_i}{\partial a_{\mathrm{x},i}}\lambda_{c,i} - \tau_s \dot{\lambda}_{a_\mathrm{x},i} = 0\,, \tag{S13}$$

where $\dot{c}_i = (F)_i$, the ith element of F.

D Derivation of Event-Based Learning Rule in Continuous Time

In this section we derive the event-based updates for the network weights. The update questions yield different results for the recurrent weights ($\mathbf{V}_\mathrm{x}$), biases ($\mathbf{b}_\mathrm{x}$) and input weights ($\mathbf{U}_\mathrm{x}$), which are derived in the remainder of this section. To increase readability important terms are highlighted in color.

D.1 Gradient updates for the recurrent weights $\mathbf{V}_\mathrm{x}$: Proof of Theorem 1

We first split the integral Eq. (7) across events as:

$$\mathcal{L} = \sum_{k=0}^{N} \int_{s_k}^{s_{k+1}} \left[\ell_c(\mathbf{c}(t), t) + \boldsymbol{\lambda}_c \cdot f_c + \sum_{\mathrm{x}\in\{u,r,z\}} \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot f_{a_\mathrm{x}} \right] dt\,. \tag{S14}$$

Then taking the derivative of the full loss function, we get:

$$\frac{d\mathcal{L}}{dv_{ji}} = \frac{d}{dv_{ji}} \left\{ \sum_{k=0}^{N} \int_{s_k}^{s_{k+1}} \left[\ell_c(\mathbf{c}(t), t) + \boldsymbol{\lambda}_c \cdot f_c + \sum_{\mathrm{x}\in\{u,r,z\}} \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot f_{a_\mathrm{x}} \right] dt \right\}\,. \tag{S15}$$

By application of Leibniz integral rule we get,

$$\frac{d}{dv_{ji}} \int_{s_k}^{s_{k+1}} \ell_c(\mathbf{c}(t), t) dt = \ell_c(\mathbf{c}, s_{k+1}) \frac{ds_{k+1}}{dv_{ji}} - \ell_c(\mathbf{c}, s_k) \frac{ds_k}{dv_{ji}} + \int_{s_k}^{s_{k+1}} \frac{\partial \ell_c}{\partial \mathbf{c}} \cdot \frac{\partial \mathbf{c}}{\partial v_{ji}} dt\,. \tag{S16}$$

and

$$\frac{d}{dv_{ji}} \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_c \cdot f_c \, dt \tag{S17}$$

$$= \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_c \cdot \frac{df_c}{dv_{ji}} \, dt \; = \; \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_c \cdot \left\{ \tau_m \frac{d}{dt} \frac{\partial \mathbf{c}}{\partial v_{ji}} + \frac{\partial F}{\partial v_{ji}} \right\} dt \tag{S18}$$

$$= \tau_m \left[\boldsymbol{\lambda}_c \cdot \frac{\partial \mathbf{c}}{\partial v_{ji}} \right]_{s_k}^{s_{k+1}} \tag{S19}$$

$$- \tau_m \int_{s_k}^{s_{k+1}} \left\{ \dot{\boldsymbol{\lambda}}_c \cdot \frac{\partial \mathbf{c}}{\partial v_{ji}} + \boldsymbol{\lambda}_c \cdot \left(\left(\frac{\partial F}{\partial \mathbf{c}} \right)^T \frac{\partial \mathbf{c}}{\partial v_{ji}} + \sum_{\mathrm{x} \in \{u,r,z\}} \left(\frac{\partial F}{\partial \mathbf{a}_\mathrm{x}} \right)^T \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} \right) \right\} dt \, , \tag{S20}$$

where we first apply Gronwall's theorem Gronwall (1919), then integration by parts, and M^T denotes the transpose of matrix M. $\ell_c(\mathbf{c}(t), t)$ is the instantaneous loss evaluated at time t. Similarly,

$$\frac{d}{dv_{ji}} \int_{s_k}^{s_{k+1}} \sum_{\mathrm{x} \in \{u,r,z\}} \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot f_{a_\mathrm{x}} \, dt = \sum_{\mathrm{x} \in \{u,r,z\}} \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot \left\{ \tau_s \frac{d}{dt} \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} + \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} \right\} dt \tag{S21}$$

$$= \tau_s \left[\boldsymbol{\lambda}_{a_\mathrm{x}} \cdot \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} \right]_{s_k}^{s_{k+1}} - \tau_s \int_{s_k}^{s_{k+1}} \left\{ \dot{\boldsymbol{\lambda}}_{a_\mathrm{x}} \cdot \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} + \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} \right\} dt \, , \tag{S22}$$

since $\frac{\partial \mathbf{b}}{\partial v_{ji}} = 0$.

Substituting these values into Eq. (S15), and setting the coefficients of terms with $\frac{\partial \mathbf{c}}{\partial v_{ji}}$ and $\frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}}$ to zero (using the fact that we can choose the adjoint variables freely due to f_c and f_{a_x} being everywhere zero by definition), we get the dynamics of the adjoint variable described in Eq. (8). The adjoint variable is usually integrated backwards in time starting from $t = T$, also due to its dependence on the loss values. The initial conditions for the adjoint variables is defined as $\boldsymbol{\lambda}_c = \boldsymbol{\lambda}_{a_\mathrm{x}} = \mathbf{0}$.

Setting the coefficients of terms with $\frac{\partial \mathbf{c}}{\partial v_{ji}}$ and $\frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}}$ to zero allows us to write the parameter updates as:

$$\frac{d\mathcal{L}}{dv_{ji}} = \sum_{k=0}^{N} \left\{ \left(l_c^- - l_c^+ \right) \frac{ds}{dv_{ji}} + \tau_s \sum_{\mathrm{x}} \left(\boldsymbol{\lambda}_{a_\mathrm{x}}^- \cdot \frac{\partial \mathbf{a}_\mathrm{x}^-}{\partial v_{ji}} - \boldsymbol{\lambda}_{a_\mathrm{x}}^+ \cdot \frac{\partial \mathbf{a}_\mathrm{x}^+}{\partial v_{ji}} \right) + \tau_m \left(\boldsymbol{\lambda}_c^- \cdot \frac{\partial \mathbf{c}^-}{\partial v_{ji}} - \boldsymbol{\lambda}_c^+ \cdot \frac{\partial \mathbf{c}^+}{\partial v_{ji}} \right) \right\} \tag{S23}$$

$$= \sum_{k=0}^{N} \xi_{\mathrm{x},ijk} \tag{S24}$$

To define the required jumps at event times for the adjoint variables, we start with finding the relationship between $\frac{\partial \mathbf{c}^-}{\partial v_{ji}}$ and $\frac{\partial \mathbf{c}^+}{\partial v_{ji}}$. Eqs. (S7) define s_k as a differentiable function of v_{ji} under the condition $\dot{c}_{n_k}^- \neq 0$ and $\dot{c}_{n_k}^+ \neq 0$ due to the implicit function theorem (Wunderlich and Pehle, 2021; Yang et al., 2014).

$$c_{n_k}^- - \vartheta_{n_k} = 0 \tag{S25}$$

$$\frac{\partial c_{n_k}^-}{\partial v_{ji}} + \frac{dc_{n_k}^-}{ds} \frac{\partial s}{\partial v_{ji}} = 0 \tag{S26}$$

$$\frac{\partial c_{n_k}^-}{\partial v_{ji}} + \dot{c}_{n_k}^- \frac{\partial s}{\partial v_{ji}} = 0 \tag{S27}$$

$$\frac{\partial s}{\partial v_{ji}} = \frac{-1}{\dot{c}_{n_k}^-} \frac{\partial c_{n_k}^-}{\partial v_{ji}} \, , \tag{S28}$$

where we write $\frac{dc^-_{n_k}}{ds} \equiv \dot{c}^-_{n_k}$ and $\dot{c}^-_{n_k} \neq 0$. Similarly,

$$c^+_{n_k} = 0 \tag{S29}$$

$$\frac{\partial c^+_{n_k}}{\partial v_{ji}} + \dot{c}^+_{n_k}\frac{\partial s}{\partial v_{ji}} = 0 \tag{S30}$$

which allows us to write

$$\frac{\partial c^+_{n_k}}{\partial v_{ji}} = \frac{\dot{c}^+_{n_k}}{\dot{c}^-_{n_k}}\frac{\partial c^-_{n_k}}{\partial v_{ji}} \tag{S31}$$

Similarly, starting from $c^+_m = c^-_m$, we can derive

$$\frac{\partial c^+_m}{\partial v_{ji}} = \frac{\partial c^-_m}{\partial v_{ji}} - \frac{1}{\dot{c}^-_{n_k}}\frac{\partial c^-_{n_k}}{\partial v_{ji}}\left(\dot{c}^-_m - \dot{c}^+_m\right) \tag{S32}$$

For the activations $\mathbf{a}_{\mathrm{X}}$, we use Eqs. (S8)–(S11) to derive the relationships between $\frac{\partial a_{\mathrm{X}}}{\partial v_{ji}}^+$ and $\frac{\partial a_{\mathrm{X}}}{\partial v_{ji}}^-$. Thus, we have:

$$\frac{\partial a^+_{\mathrm{X},m}}{\partial v_{ji}} = \frac{\partial a^-_{\mathrm{X},m}}{\partial v_{ji}} - \frac{1}{\tau_s}\frac{v_{mn_k}\, r^-_{\mathrm{X},n_k} c^-_{n_k}}{\dot{c}^-_{n_k}}\frac{\partial c^-_{n_k}}{\partial v_{ji}} + \delta_{in_k}\delta_{jm}c^-_{n_k} + c^-_{n_k}v_{mn_k}\frac{\partial r^-_{\mathrm{X},n_k}}{\partial v_{ji}} - c^-_{n_k}v_{mn_k}\frac{\dot{r}^-_{\mathrm{X},n_k}}{\dot{c}^-_{n_k}}\frac{\partial c^-_{n_k}}{\partial v_{ji}} \tag{S33}$$

$$\frac{\partial a^+_{\mathrm{X},n_k}}{\partial v_{ji}} = \frac{\partial a^-_{\mathrm{X},n_k}}{\partial v_{ji}} \tag{S34}$$

where $\mathbf{r}_{\mathrm{X}} = \mathbf{0}$ if $\mathrm{X} \in \{u, r\}$ and $\mathbf{r}_{\mathrm{X}} = \mathbf{r}$ if $\mathrm{X} = \{z\}$.

Substituting Eqs. (S31), (S32), (S34), (S33) into Eq. (S23), we get:

$$\xi_{\mathrm{X},ijk} = \Bigg\{ \frac{\partial c^-_{n_k}}{\partial v_{ji}} \Bigg(\frac{-1}{\dot{c}^-_{n_k}}\left(\ell^+_c - \ell^-_c\right) + \tau_m \left(\lambda^-_{c,n_k} - \frac{\dot{c}^+_{n_k}}{\dot{c}^-_{n_k}}\lambda^+_{c,n_k} \right) + \tau_m \frac{1}{\dot{c}^-_{n_k}} \sum_{m\neq n_k} \lambda^+_{c,m}\left(\dot{c}^-_m - \dot{c}^+_m\right) \tag{S35}$$

$$+ \sum_{\mathrm{X}} \frac{r^-_{\mathrm{X},n_k} c^-_{n_k}}{\dot{c}^-_{n_k}} \sum_{m\neq n_k} v_{mn_k}\lambda^+_{a_{\mathrm{X}},m} + \tau_s \sum_{\mathrm{X}} \frac{\dot{r}^-_{\mathrm{X},n_k} c^-_{n_k}}{\dot{c}^-_{n_k}} \sum_{m\neq n_k} v_{mn_k}\lambda^+_{a_{\mathrm{X}},m} \Bigg) \tag{S36}$$

$$+\tau_m \sum_{m\neq n_k} \frac{\partial c^-_m}{\partial v_{ji}}\left(\lambda^-_{c,m} - \lambda^+_{c,m}\right) \tag{S37}$$

$$\tau_s \sum_{\mathrm{X}} \frac{\partial a^-_{\mathrm{X},n_k}}{\partial v_{ji}} \left(\left(\lambda^-_{a_{\mathrm{X}},n_k} - \lambda^+_{a_{\mathrm{X}},n_k}\right) - c^-_{n_k} G'(a^-_{\mathrm{X},n_k}) \sum_{m\neq n_k} v_{mn_k}\lambda a_{\mathrm{X}}, m^+ \right) \tag{S38}$$

$$\tau_s \sum_{\mathrm{X}} \sum_{m\neq n_k} \frac{\partial a^-_{\mathrm{X},m}}{\partial v_{ji}}\left(\lambda^-_{a_{\mathrm{X}},m} - \lambda^+_{a_{\mathrm{X}},m}\right) \tag{S39}$$

$$-\tau_s \delta_{in_k} r^-_{\mathrm{X},n_k} c^-_{n_k} \sum_{m\neq n_k} \delta_{jm}\lambda^+_{a_{\mathrm{X}},m} \Bigg\} \tag{S40}$$

where we use $\mathbf{r}_{\mathrm{X}} = G(\mathbf{a}_{\mathrm{X}})$ to denote $G(\mathbf{a}_r) = \mathbf{r}$ and $G(\mathbf{a}_z) = G(\mathbf{a}_u) = \mathbf{1}$, δ_{ab} is the kronecker delta defined as:

$$\delta_{ab} = \begin{cases} 1 & \text{if} \quad a = b, \\ 0 & \text{otherwise} \end{cases} \tag{S41}$$

Setting the coefficients of $\frac{\partial c_i^-}{\partial v_{ji}}$ and $\frac{\partial a_{\mathrm{X}}^-}{\partial v_{ji}}$ to 0 (again, using our ability to choose the adjoint variables freely), we can get both $\xi_{\mathrm{X},ijk}$ and the transitions for the adjoint variables.

For the parameter updates we get:

$$\xi_{ijk} = -\tau_s \delta_{in_k} r_{\mathrm{X},n_k}^- c_{n_k}^- \sum_{m \neq n_k} \delta_{jm} \lambda_{a_{\mathrm{X}},m}^+ \tag{S42}$$

$$= -\tau_s r_{\mathrm{X},i}^- c_i^- \lambda_{a_{\mathrm{X}},j}^+ \,. \tag{S43}$$

Thus we can write:

$$\Delta w_{\mathrm{X},ij} \;=\; \frac{\partial}{\partial w_{\mathrm{X},ij}} \mathcal{L}(\mathbf{W}) \;=\; \sum_k \xi_{\mathrm{X},ijk} \,. \tag{S44}$$

The corresponding value of $\xi_{\mathrm{X},ijk} = (\boldsymbol{\xi}_{\mathrm{X},k})_{ij}$ is given by the following formula, written in vector form for succinctness:

$$\boldsymbol{\xi}_{\mathrm{X},k} \;=\; -\tau_s \left(\mathbf{r}_{\mathrm{X}}^-(s_k) \odot \mathbf{c}^-(s_k)\right) \otimes \boldsymbol{\lambda}_{a_{\mathrm{X}}}^+(s_k) \,, \tag{S45}$$

The jumps/transitions of the adjoint variables are:

$$\lambda_{a_{\mathrm{X}},m}^+ = \lambda_{a_{\mathrm{X}},m}^- \tag{S46}$$

$$\lambda_{a_{\mathrm{X}},n_k}^+ = \lambda_{a_{\mathrm{X}},n_k}^- - c_{n_k}^- G'(a_{\mathrm{X},n_k}) \sum_{m \neq n_k} v_{mn_k} \lambda_{a_{\mathrm{X}},m}^+ \tag{S47}$$

$$\lambda_{c,m}^+ = \lambda_{c,m}^- \tag{S48}$$

$$\begin{aligned} \tau_m \dot{c}_{n_k}^+ \lambda_{c,n_k}^+ = &-(\ell_c^+ - \ell_c^-) + \tau_m \dot{c}_{n_k}^- \lambda_{c,n_k}^- + \tau_m \sum_{m \neq n_k} \lambda_{c,m}^+ (\dot{c}_m^- - \dot{c}_m^+) \\ &+ \tau_s c_{n_k}^- \sum_{\mathrm{X}} \left(\dot{r}_{\mathrm{X},n_k}^- + \frac{r_{\mathrm{X},n_k}^-}{\tau_s} \right) \sum_{m \neq n_k} v_{mn_k} \lambda_{a_{\mathrm{X}},m}^+ \,, \end{aligned} \tag{S49}$$

where $(\ell_c^+ - \ell_c^-)$ denotes the jumps in the instantaneous loss around event time s_k. Thus, all the quantities on the right hand side of Eq. (S23) can be calculated from known quantities.

D.2 Gradient updates for biases $\mathbf{b}_{\mathrm{X}}$

Proceeding similarly for the biases $\mathbf{b}_{\mathrm{X}}$ for each of $\mathrm{X} \in \{u, r, z\}$ (dropping the subscript X for simplicity):

$$\frac{d\mathcal{L}}{db_i} = \frac{d}{db_i} \left\{ \sum_{k=0}^{N} \int_{s_k}^{s_{k+1}} \left[\ell_c(\mathbf{c}(t), t) + \boldsymbol{\lambda}_c \cdot f_c + \sum_{\mathrm{X} \in \{u,r,z\}} \boldsymbol{\lambda}_{a_{\mathrm{X}}} \cdot f_{a_{\mathrm{X}}} \right] dt \right\} \,. \tag{S50}$$

the $\xi_{\mathrm{X},ik}^{\mathrm{bias}}$ term can be shown to be:

$$\xi_{\mathrm{X},ik}^{\mathrm{bias}} = \int_{s_k}^{s_{k+1}} \lambda_{a_{\mathrm{X}},i} \, dt \tag{S51}$$

with

$$\frac{d\mathcal{L}}{db_i} = \sum_{k=0}^{N} \xi_{\mathrm{X},ik}^{\mathrm{bias}} \,. \tag{S52}$$

D.3 Gradient updates for input weights $\mathbf{U}_{\mathrm{X}}$

Proceeding similarly for the input weights $\mathbf{U}_{\mathrm{X}}$ for each of $\mathrm{X} \in \{u, r, z\}$ (dropping the subscript X for simplicity):

$$\frac{d\mathcal{L}}{du_{jx}} = \frac{d}{du_{jx}} \left\{ \sum_{k=0}^{N} \int_{s_k}^{s_{k+1}} \left[\ell_c(\mathbf{c}(t), t) + \boldsymbol{\lambda}_c \cdot f_c + \sum_{\mathrm{X} \in \{u,r,z\}} \boldsymbol{\lambda}_{a_{\mathrm{X}}} \cdot f_{a_{\mathrm{X}}} \right] dt \right\} \,. \tag{S53}$$

the $\xi^{\text{input}}_{\text{x},jxk}$ term can be shown to be:

$$\xi^{\text{input}}_{\text{x},jxk} = -\tau_s \lambda^{+}_{a_{\text{x}},j} x_x \tag{S54}$$

with

$$\frac{d\mathcal{L}}{du_{jx}} = \sum_{k=0}^{N} \xi^{\text{input}}_{\text{x},jxk} \ . \tag{S55}$$

E Details of experiments

E.1 DVS128 Gesture recognition

In this experiment we use Tonic library (Lenz et al., 2021) to prepare the dataset. The recordings in the dataset are sliced by time without any overlap to produce samples of length 1.7 seconds. The data is denoised with a filter time of 10ms and normalised to [0;1] before being fed to the model. The positive and negative polarity events are represented by 2 separate channels. Our model consists of a preprocessing layer which performs downscaling and flattening transformations, followed by two RNN layers. Both RNN layers have the same number of hidden dimensions. Finally, a fully connected layer of size 11 performs the classification. All the weights were initialised using Xavier uniform distribution, while the biases were initialised using a uniform distribution. The unit thresholds were initialised using a normal distribution with mean 0 and standard deviation of $\sqrt{2}$, but was transformed to their absolute value after every update. We use cross-entropy loss and Adam optimizer with default parameters (0.001 learning rate, $\beta_1 = 0.9$, $\beta_2 = 0.999$). The learning rate is scaled by 80% every 100 epochs.

We use additional loss to regularize the output and increase sparsity of the network. The applied regularization losses are shown in Eq. (S57). L_{reg} is applied indirectly to the active outputs and L_{act} is applied on the auxiliary internal state $c_i^{\langle t \rangle}$, the threshold parameter ϑ_i is detached from the graph in the second equation so the loss only affects the internal state. We set the regularization weights w_{reg} and w_v to 0.01 and 0.05 respectively.

Fig. S2(a) shows comparison of training curves for LSTM, GRU and EGRU, mean activity of the EGRU network is also shown, the network achieved 80%+ sparsity without significant drop in accuracy. The activities of LSTM and GRU are not shown in Fig S2(a) since they are always 100%. In our experiments we calculate sparsity of these networks as average number of activations close to zero with an absolute tolerance of 1×10^{-8}, however in Fig. S2(b) we show that even if we increase the absolute tolerance to 1×10^{-3}, the sparsity of these networks is still an order of magnitude lower than EGRU. The analyse the activity of the individual units of EGRU network in Fig. S3 with an histogram of unit activity for the entire test dataset. The activity of the units shown in x-axis is normalised to the sequence length. As expected from the overall activity sparsity shown in Table. 1, most of the units have low activity with some dead units.

Hyperparameters were chosen by conducting a grid search over the number of units (32 - 2048), number of layers (1 - 4) and values of regularization weights. Learning rate and optimizer was chosen from initial experiments. Since batch size did not have any significant effect on training, we chose a batch size that maximizes GPU utilization. Models with CNN feature extractors are trained with slightly different hyper-parameters than the pure RNNs. These hyperparameters are chosen by a Bayesian search. This includes hidden to hidden dropout p_h of 0.4 for CNN+EGRU(256) and 0.08 for CNN+EGRU(795). The batch size used in this case is 40 to ease data augmentation. The learning rate is set initially to 0.001 and then scaled by 80% every 60 epochs. The input channels are combined into a single channel by averaging over the channel dimension.

$$L_{reg} = w_{reg} \left(\frac{1}{N} \frac{1}{n_{\text{units}}} \sum_{n=1}^{N} \sum_{1}^{n_{\text{units}}} H\left(c_i^{\langle t \rangle} - \vartheta_i \right) \quad - 0.05 \right) \tag{S56}$$

$$L_{act} = w_v \left(\frac{1}{N} \frac{1}{n_{\text{units}}} \sum_{n=1}^{N} \sum_{i=1}^{n_{\text{units}}} c_i - (\vartheta_i - 0.05) \right) \tag{S57}$$

where N spans mini-batch.

architecture (# units)	para-meters	effective MAC (mean±std)	accu-racy (%) (mean±std)	activity sparsity (%) (mean±std)	backward sparsity (%) at epoch 100 (mean±std)
LSTM (867)	16.3M	14.2M	87.9±1.0	0	-
GRU (1024)	15.7M	10.6M	88.1±0.8	0	-
GRU (1024)+DA	15.7M	10.6M	94.8±0.3	0	-
EGRU (512)	5.5M	1.2M±0.1M	86.0±1.2	76.1±5.9	45.7±0.7
EGRU (1024)	15.7M	3.1M±0.4M	87.7±2.1	79.8±3.3	54.4±1.2
EGRU (1024+DA)	15.7M	2.7M±0.3M	95.9±0.7	84.2±3.0	52.8±3.3
EGRU (1024)*	110.1M	105.2M±441.3K	85.7±0.9	77.3±7.0	64.6±1.1
CNN+GRU (136)+DA*	1.7M+0.4M**	0.3M†	97.15±0.2	0	-
CNN+**EGRU** (256)+DA*	1.7M+1.0M**	0.5M±12.4,K†	96.8±0.3	73.4±2.1	55.3±1.3
CNN+**EGRU** (795)+DA*	1.7M+3.1M**	1.6M±34.3K†	97.3±0.4	77.6±1.8	72.7±1.0

Table S1: Model performance over 5 runs for the DVS Gesture recognition task. Effective number of MAC operations as described in section 3.3. * indicates network with 128×128 input size, all other networks have scaled input as explained in Section 5.1. ** Indicated parameters are split between CNN and RNN. † Only RNN MAC operations. CNN adds an additional 79M MAC operations, however since these are not affected by activity sparsity, we exclude them from this table for brevity.

layer	channels	output shape
Input	1	128x128
Convolution	64	31x31
ReLU	64	31x31
Pooling	64	15x15
Convolution	192	15x15
ReLU	192	15x15
Pooling	192	7x7
Convolution	384	7x7
ReLU	384	7x7
Pooling	384	3x3
Convolution	256	3x3
ReLU	256	3x3
Pooling	256	1x1
Convolution	256	1x1
ReLU	256	1x1
Fully connected	512	1x1
ReLU	512	1x1

Table S2: Details of CNN layers for the feature extraction head used in CNN+EGRU models

Model Hidden units	EGRU 512/1024	CNN+EGRU 795	CNN+EGRU 256	CNN+GRU 136
Layers	2	1	2	1
Learning rate	0.001	0.001	0.001	0.001
Learning rate decay	0.8	0.874	0.8	0.89
Learning rate decay epochs	100	56	100	70
Batch size	256	40	40	40
Dropout p_l	0	0.632	0	0.239
DropConnect p_h	0	0.081	0.4	0.074
Zoneout p_z	0	0.2	0	0
Activity regularization	0.01	0.01	0.01	-
Surrogate gradient ϵ	1	0.588	1	-
Threshold init μ	0	-0.246	0	-

Table S3: Detailed hyper-parameters of our best models for DVS Gesture recognition.

E.1.1 ABLATION STUDY

We performed ablation studies, showing the performance of the EGRU models with variation of the gating mechanism. All models in this study are a variation of our **EGRU**(1024) model. The results of these experiments are presented in Table S4. By using a scalar threshold ϑ where all units share a same threshold parameter we find that the accuracy drops by 2% but the the activity sparsity is increased to 90%.

Next, we evaluate a model with 'hard reset' where the auxiliary internal state $c_i^{\langle t \rangle}$ is set to 0 every time an event is emitted by an unit. We observe a drop in accuracy possibly because the hard reset loses information when the internal state has gone above threshold at at any particular simulation time step, which may happen due to the limitations on precision in discrete time simulations with a fixed time grid. This drop in performance might be significant for applications which require high temporal resolution, which necessitates the term $- y_i^{\langle t-1 \rangle}$ in Eq. (2). Model is also evaluated with 'no reset' where the term $- y_i^{\langle t-1 \rangle}$ is removed from Eq. (2) which results in slightly lower accuracy and sparsity.

E.2 SEQUENTIAL MNIST

All the weights were initialised using Xavier uniform distribution, while the biases were initialised using a uniform distribution. The unit thresholds were initialised using a normal distribution with mean 0 and standard deviation of $\sqrt{2}$, but was transformed to be between 0 and 1 by passing through a standard sigmoid/logistic function after every update. We used a batch size of 500 for sMNIST and 300 for psMNIST. In all the experiments, we trained the network with Adam with default parameters (0.001 learning rate, $\beta_1 = 0.9$, $\beta_2 = 0.999$) on a cross-entropy loss function. We used gradient clipping with a max gradient norm of 0.25. We trained models for 200 epochs for sMNIST and 700 epochs for psMNIST. The model trained on psMNIST used DropConnect Wan et al. (2013) with $p = 0.4$. The outputs of all the units were convolved with an exponential filter with time constant of 10 time units i.e. with $e^{\frac{-1}{10}}$ to calculate an output trace. The value of this trace at the last time step was used to predict the class through a softmax function.

For sMNIST, hyper-parameters were chosen by performing a search over batch sizes (50-1000), learning rates $(10^-3, 10^-4)$, use of output trace, activity regularisation. An extensive Bayesian search was conducted using Weights & Biases (Biewald, 2020) to optimize hyperparameters of EGRU with 590 hidden units on psMNIST on NVIDIA V100 GPUs. The initialisation method of the thresholds were also tweaked – currently we use a normal initialisation with a sigmoid projection into the $[0, 1]$ range, but we experimented with projecting it with an absolute value followed by clipping, which proved unstable.

E.3 PTB LANGUAGE MODELING

Our experimental setup largely follows Merity et al. (2017). In particular, we download and preprocess PennTreebank (Marcus et al., 1993) and WikiText-2 (Merity et al., 2016) with their published code [1]. Words are projected to an d_{emb}-dimensional dense vector by a linear transformation, followed by three RNN layers without skip connections. The first two RNN layers feature the same hidden dimension, while the hidden dimension of the last RNN layer equals the word vector embedding dimension. As common in language modeling, we apply cross entropy loss and use weight tying Inan et al. (2017); Press and Wolf (2017).

E.3.1 TRAINING DETAILS AND HYPERPARAMETER OPTIMIZATION

Our activity sparsity mechanism introduces two new hyperparameters ϵ and μ. First, the shape of the surrogate gradient

$$\frac{\mathrm{d}H}{\mathrm{d}c} = \lambda \ \max\left(1 - |c|/\epsilon\right) \tag{S58}$$

is defined by ϵ, which thus determines the backward sparsity. Second, the initialization of the rectifyer thresholds φ determines both inference and BPTT sparsity at initialization. We choose to reparameterize thresholds with a sigmoid function to limit their domain to the interval $[0, 1]$. With τ_i drawn from a normal distribution $\tau_i \sim \mathcal{N}(\mu, \sigma\sqrt{2})$, the thresholds are initialized as $\varphi_i = 1/(1 + \exp(-\tau_i))$, where τ_i are the trainable parameters and μ is the new hyperparameter. See Figure S7 for the resulting distribution of initial thresholds. Table S7 shows the sensitivity of model

[1] https://github.com/salesforce/awd-lstm-lm

performance w.r.t. these parameters. Non-trainable thresholds are also considered in Table S7. We observe that language modeling benefits from initialization near 0. Trainable thresholds slightly outperform non-trainable thresholds. The gap depends on the initialization and is fairly small for the best initialization strategies. This is not very much surprising as the model is able to counteract constant thresholds with bias terms in the GRU equations.

We apply most of the regularization strategies of Merity et al. (2017), except for (temporal) activity regularization. Backpropagation through time is conducted with a variable sequence length. With 95% probability, the sequence length is drawn from $\mathcal{N}(s, 5)$, and with 5% probability the sequence length is drawn from $\mathcal{N}(s/2, 5)$, where s is a tuned hyperparameter. We apply variational dropout Gal and Ghahramani (2016) to the vocabulary with probability p_{voc}, to the word embedding vectors with probability p_{emb} as well as to each layer output with probability p_l. DropConnect Wan et al. (2013) was applied to the hidden-to-hidden weight matricies with probability p_h. We experimented with both Adam Kingma and Ba (2015) and NT-AvSGD Merity et al. (2017) optimization procedures. While Adam lead to competitive results for all models, GRU based models did not converge to competitive results using NT-AvSGD. When optimized with SGD based optimizers, both GRU and EGRU fell behind Adam optimized models. Momentum was set to 0 as reported in Melis et al. (2018). Gradient clipping was applied to all models, where the magnitude of clipped gradients only made very small differences in results. While gradient clipping of 0.25 was used for GRU, we used 2.0 for EGRU.

We apply a cosine-annealing learning rate schedule, where the first $n/2$ epochs were trained at constant learning rate λ, and a cosine decay from λ to $0.1 \cdot \lambda$ was applied for the remaining $n/2$ epochs. All EGRU models were trained for 2500 epochs.

An extensive Bayesian search was conducted using Weights & Biases (Biewald, 2020) to optimize hyperparameters of GRU with 1350 hidden units and EGRU with 1350 and 2000 hidden units on Penn Treebank for about 65 GPU days on NVIDIA A100 GPUs. The surrogate gradient parameter ϵ and the initialization of the thresholds φ_i are treated as hyperparameters of this model. Due to our constrained computational resources, we used the same hyperparameters on WikiText-2.

We found the word embedding dimension $d_{\text{emb}} = 400$ set by (Merity et al., 2017) to be a good fit for GRU. For EGRU, we observed much larger dimensions around $d_{\text{emb}} = 800$ to outperform smaller dimensions. This increases the number of parameters of the word-embedding layer by about a factor of 2. Language models need to compare the output embedding vector via dot product with the embedding vectors of the dictionary. Since EGRU outputs only positive values, we hypothesise that extra parameters are required to cancel terms in the dot-product. See table S6 for detailed hyperparameters of the best models.

E.4 Model compression through activity pruning

We present a simple model compression heuristic based on activity. Starting from a trained model, we remove the least active r_i % of the units of layer i. Since we observed different levels of activity in the layers, we work with different combinations of compression rates r_i per layer. Figure S6 shows how model performance and sparsity depend on the model compression. Surprisingly, we observe very similar sparsity levels across the evaluated compressed models.

F Dataset licenses

Penn Treebank Marcus et al. (1993) is subject to the Linguistic Data Consortium User Agreement for Non-Members[2].

> LDC Not-For-Profit members, government members and nonmember licensees may use LDC data for noncommercial linguistic research and education only. For-profit organizations who are or were LDC members may conduct commercial technology development with LDC data received when the organization was an LDC for-profit member unless use of that data is otherwise restricted by a corpus-specific license agreement. Not-for-profit members, government members and nonmembers, including nonmember for-profit organizations, cannot use LDC data to develop or test products for commercialization, nor can they use LDC data in any commercial product or for any commercial purpose.

[2] https://www.ldc.upenn.edu/data-management/using/licensing

model	accuracy (%)	activity sparsity (%)
Full **EGRU** (1024)	90.2	82.5
without regularization	89.3	76.5
scalar ϑ	88.3	90.8
hard clear	87.2	90.0
no clear	88.9	80.7

Table S4: Performance of the **EGRU** (1024) model for the ablation study performed on the DVS gesture task as described in Section E.1.1.

architecture (# units)	parameters	effective MAC (mean±std)	test accuracy (%) (mean±std)	activity sparsity (%) (mean±std)	backwards sparsity (%) at epochs 20/50/100 (mean±std)
GRU (512)	791K	795K	98.6±0.2	-	-
GRU (590)	1.049M	1.054M	98.7±0.1	-	-
EGRU (512)	790K	(147±7)K	87.2±3.0	82.1±0.9	22.2±2.8/24.9±0.7/ 28.7± 1.1
EGRU (590)	1.048M	(210±51)K	95.5±1.6	80.5±4.9	24.9±6.8 / 26.1±5.9 / 25.6±1.7

Table S5: Model performance over 4 runs for sequential MNIST task. Test scores are given as percentage accuracy, where higher is better.

	Model	GRU	**EGRU**	**EGRU**
	Hidden units	1350	1350	2000
PTB	Test ppl (best)	66.3	58.7	58.8
	Val perplexity (best)	68.7	59.5	59.6
	Val perplexity (mean ± std)	68.9 ± 0.1	59.7 ± 0.1	60.0 ± 0.5
	Forward sparsity (test)	0.0 %	(79.9 ± 0.1) %	(85.3 ± 0.9) %
	Backward sparsity (train)	0.0 %	(46.0 ± 0.3) %	(40.5 ± 3.8) %
	Effective MACs (RNN + emb)	(21.9 + 5.6) M	(6.8 + 3.1) M	(9.7 + 3.0) M
WT2	Test perplexity (best)	71.8	70.6	68.9
	Val perplexity (best)	75.7	73.9	71.5
	Val perplexity (mean ± std)	75.9 ± 0.1	74.0 ± 0.1	75.7 ± 6.5
	Forward sparsity (test)	0.0 %	(77.0 ± 0.1) %	(84.6 ± 2.9) %
	Backward sparsity (train)	0.0 %	(43.8 ± 0.3) %	(36.1 ± 8.7) %
	Effective MACs (RNN + emb)	(21.9 + 16.9) M	(7.4 + 10.1) M	(10.6 + 9.5) M
	Learning rate	4.62×10^{-4}	4.44×10^{-4}	4.94×10^{-4}
	Batch size	96	64	128
	Sequence length s	34	68	67
	Embedding dimension d_{emb}	563	788	786
	Dropout p_h	0.506	0.679	0.621
	Dropout p_l	0.474	0.264	0.241
	Dropout p_{emb}	0.729	0.707	0.765
	Dropout p_{voc}	0.093	0.055	0.149
	Weight decay	4.60×10^{-6}	9.01×10^{-6}	6.69×10^{-6}
	Activity regularization	2.766	0	0
	Temporal Activity regularization	0.29	0	0
	Surrogate gradient ϵ	-	0.459	0.425
	Threshold init μ	-	-3.769	-3.496

Table S6: Detailed results and parameters for our best models. Mean and standard deviations are calculated over 5 runs with different random seeds. Effective MAC operations consider the layer-wise sparsity in the forward pass. Activity sparsity is given for the trained model to resemble inference sparsity. Backward sparsity is averaged over the whole training. Model parameters were optimized on Penn Treebank and transfered to WikiText-2.

Following Merity et al. (2017), we download Penn Treebank data from http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz.

The DVS128 Gesture Dataset (Amir et al., 2017) is released under the Creative Commons Attribution 4.0 license and can be retrieved from: https://research.ibm.com/interactive/dvsgesture/. We used Tonic library (Lenz et al., 2021) for Pytorch to preprocess data and to apply transformations.

The sequential MNIST task (Le et al., 2015) is based on the MNIST dataset first introduced in (LeCun et al., 1998), available from: http://yann.lecun.com/exdb/mnist/.

G HARDWARE AND SOFTWARE DETAILS

We implement EGRU as a modification of Haste GRU (Nanavati, 2020) and observe slightly shorter wallclock times than PyTorch's (Paszke et al., 2019) GRU implementation.

Most of our experiments were run on NVIDIA A100 GPUs. Some initial hyper-parameter searches were conducted on NVIDIA V100 and Quadro RTX 5000 GPUs. We used about 45,000 computational hours in total for training and hyper-parameter searches. All models and experiments were implemented in PyTorch. For the continuous time EGRU model, we also used the torchdiffeq (Chen et al., 2018) library.

The machines used for the DVS128 gesture recognition task and for the PTB language modeling task feature 8x NVIDIA A100-SXM4 (40GB) GPUs, 2x AMD EPYC CPUs 7352 with 24 cores each, and 1TB RAM on each compute node. For each run, we only use a single GPU, and a fraction of the cores and memory available on the node to run multiple experiments in parallel. The nodes operate Red Hat Enterprise Linux Server (release 7.9).

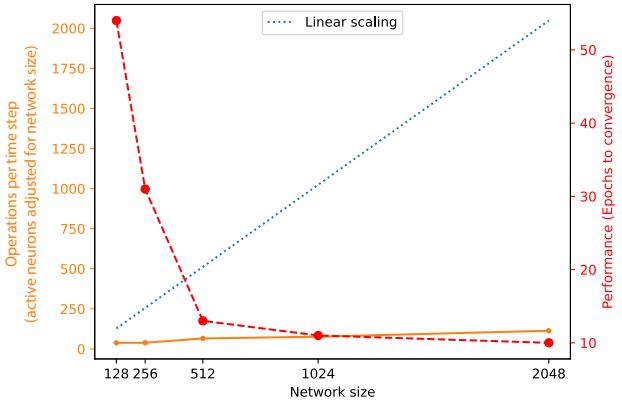
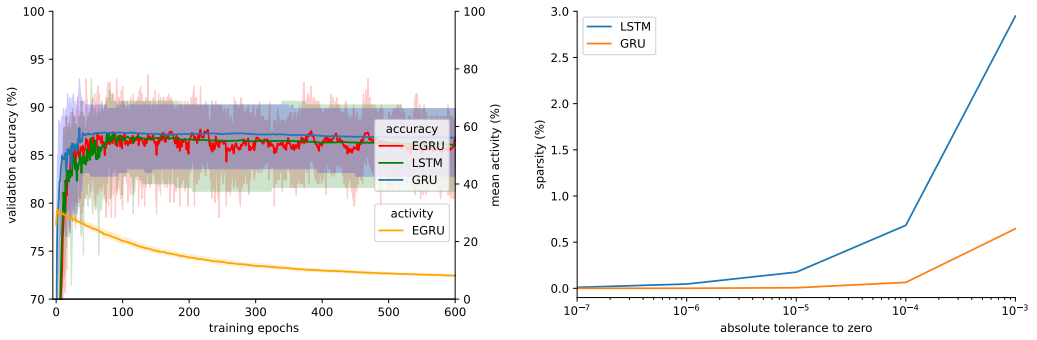


Figure S1: Illustration of the scaling properties of the EGRU on a 14×14 sequential MNIST task (1 run per network size). As the size of the network increases, the network converges faster. Increasing the network size 10x increases the speed of convergence 5x, while increasing the total amount of computation per sample only 2x. The total amount of computation is adjusted for network size. The smaller subsampled 14x14 sMNIST task was chosen here for reasons of computational limitations.



(a) **(b)**

Figure S2: **(a)** mean training curves over 5 runs for DVS gesture task. **(b)** activity sparsity of LSTM and GRU for DVS gesture task over 1 run across various values of absolute tolerance to zero.

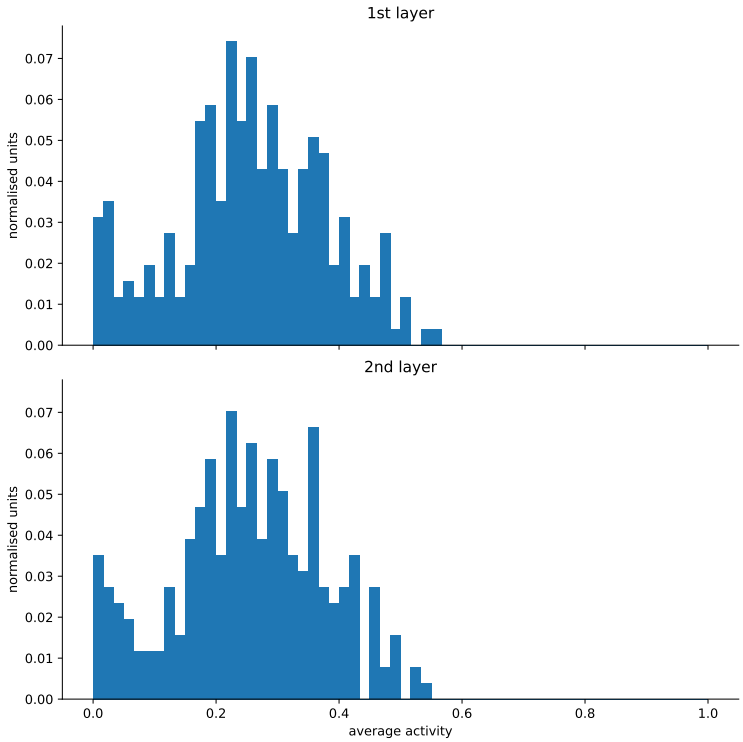
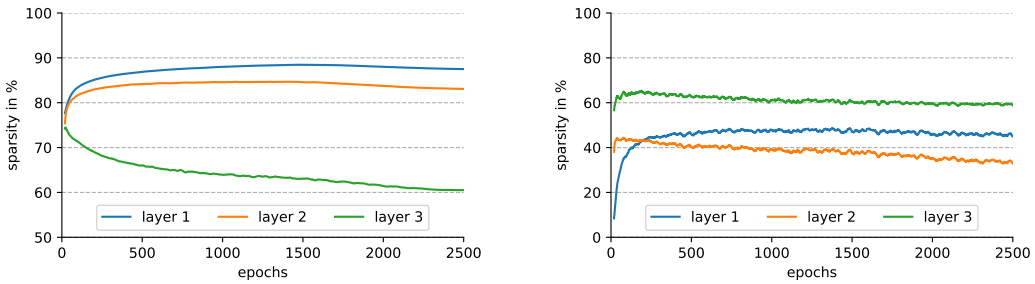


Figure S3: Histogram of normalised unit activity for fully trained 2 layer EGRU network with 256 units performing DVS gesture task. Activity is strongly skewed towards low values. There are no units always active, however some units are inactive for the entire test dataset.



(a) **(b)**

Figure S4: EGRU with 1350 hidden units on the Penn Treebank language modeling task with pseudo-derivative $\epsilon = 0.45$. **(a)** layer-wise forward sparsity **(b)** layer-wise backward sparsity

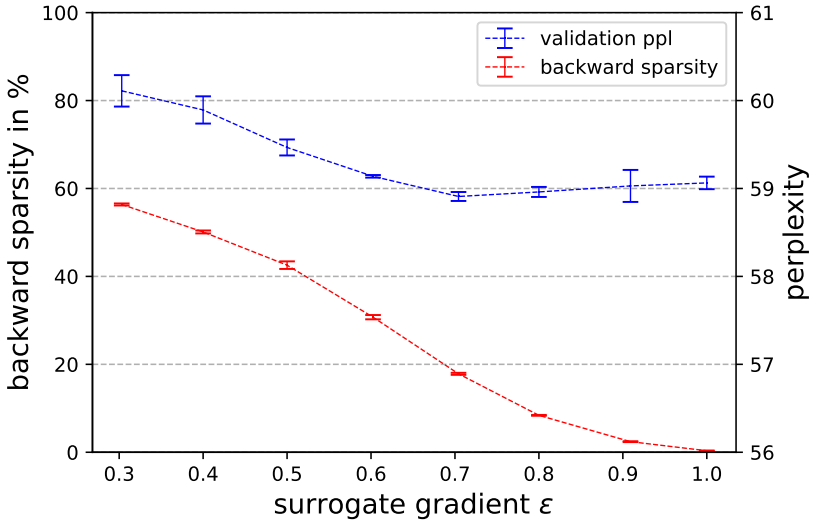


Figure S5: Backward sparsity and corresponding perplexity for a 3-layer EGRU with 1350 hidden units on the Penn Treebank language modeling task with varying pseudo-derivative support ϵ. Standard deviations are calculated over three runs with different random seeds.

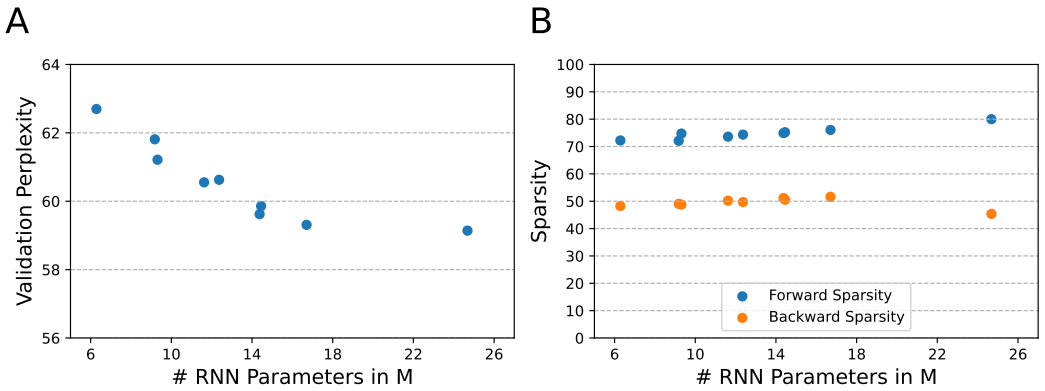


Figure S6: We apply different levels of layer-wise model compression according to Sec. E.4. **A** Model performance **B** Forward and backward sparsity

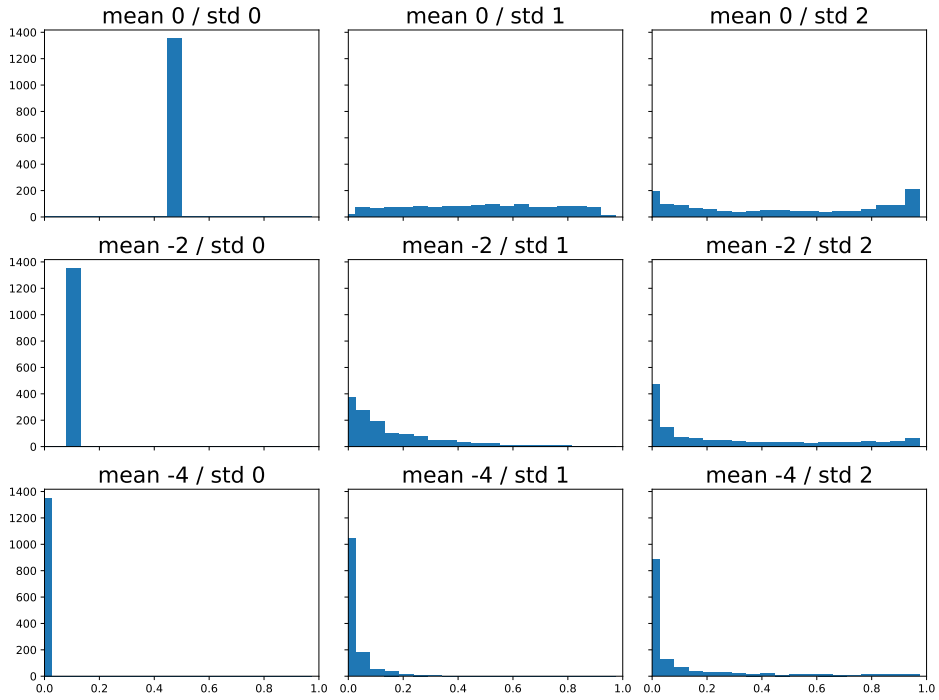


Figure S7: Visualization of threshold initializations for different parameters μ and σ. The thresholds are reparameterized with a sigmoid function to limit their domain to the interval $[0, 1]$. With τ_i drawn from a normal distribution $\tau_i \sim \mathcal{N}(\mu, \sigma\sqrt{2})$, the thresholds are initialized as $\varphi_i = 1/(1+\exp(-\tau_i))$, where τ_i are the trainable parameters

non-trainable thresholds trainable thresholds		threshold initialization std 0.0	 1.0	 2.0
threshold initialization mean	-4.0	59.7 ± 0.1 59.2 ± 0.1	59.5 ± 0.1 59.4 ± 0.2	60.1 ± 0.1 60.0 ± 0.2
	-2.0	682.7 682.7	63.0 ± 0.2 61.8 ± 0.2	63.9 ± 0.2 62.6 ± 0.2
	0.0	682.7 682.7	292.5 ± 337.9 288.8 ± 341.1	73.8 ± 0.6 69.8 ± 0.1

Table S7: Performance on Penn Treebank word-level language modeling for different values of the initialization parameters μ and σ (see E.3). See table S7 for the corresponding distribution of thresholds Scores are given as perplexity averaged over three runs, where lower is better. Gray values correspond to models with non-trainable thresholds.

REFERENCES

A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.

L. Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6572–6583. Curran Associates, Inc., 2018.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/076a0c97d09cf1a0ec3e19c7f2529f2b-Paper.pdf.

T. H. Gronwall. Note on the derivatives with respect to a parameter of the solutions of a system of differential equations. *Annals of Mathematics*, pages 292–296, 1919.

H. Inan, K. Khosravi, and R. Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1aPbsFle.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Q. V. Le, N. Jaitly, and G. E. Hinton. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units, Apr. 2015. URL http://arxiv.org/abs/1504.00941.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

G. Lenz, K. Chaney, S. B. Shrestha, O. Oubari, S. Picaud, and G. Zarrella. Tonic: event-based datasets and transformations., July 2021. URL https://doi.org/10.5281/zenodo.5079802. Documentation available under https://tonic.readthedocs.io.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, jun 1993. ISSN 0891-2017.

G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=ByJHuTgA-.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

S. Merity, N. S. Keskar, and R. Socher. Regularizing and Optimizing LSTM Language Models. *arXiv:1708.02182 [cs]*, Aug. 2017.

S. Nanavati. Haste: a fast, simple, and open rnn library. https://github.com/lmnt-com/haste/, Jan 2020.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

O. Press and L. Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain, Apr. 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025.

L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/wan13.html.

T. C. Wunderlich and C. Pehle. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, 11(1):12829, June 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-91786-z. URL https://www.nature.com/articles/s41598-021-91786-z.

W. Yang, D. Yang, and Y. Fan. A proof of a key formula in the error-backpropagation learning algorithm for multiple spiking neural networks. In *International Symposium on Neural Networks*, pages 19–26. Springer, 2014.