Supplementary information “EGRU: Event-based GRU for activity-sparse inference and learning”

Anonymous Author(s)
Affiliation
Address
`email`

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 [3], 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:

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

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:

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

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

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

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:

$$\begin{aligned} f_{a_{\mathrm{X}}}\quad &\equiv\quad \tau_s\,\dot{\mathbf{a}}_{\mathrm{X}}+\mathbf{a}_{\mathrm{X}}\;+\mathbf{b}_{\mathrm{X}}\;=\;\mathbf{0}\,,\qquad \mathrm{X}\in\{u,\;r,\;z\} &\text{(S5)}\\ f_c\quad &\equiv\quad \tau_m\,\dot{\mathbf{c}}(t)+\mathbf{u}(t)\odot\left(\mathbf{c}(t)-\mathbf{z}(t)\right)\;=\;\tau_m\,\dot{\mathbf{c}}(t)-F\left(t,\mathbf{a}_u,\mathbf{a}_r,\mathbf{a}_z,\mathbf{c}\right)=0\,. &\text{(S6)}\end{aligned}$$

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:

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

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

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 Derivation of event-based learning rule for EGRU

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.

C.1 Gradient updates for the recurrent weights $\mathbf{V}_{\mathrm{x}}$

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

$$\mathcal{L} = \sum_{k=0}^{N} \int_{s_k}^{s_{k+1}} \left[\ell_c(\mathbf{c}(t), t) + \boldsymbol{\lambda}_c \cdot f_c + \sum_{\mathrm{x}\in\{u,r,z\}} \boldsymbol{\lambda}_{a_{\mathrm{x}}} \cdot f_{a_{\mathrm{x}}} \right] dt\,. \tag{S13}$$

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

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

By application of Leibniz integral rule we get,

$$\frac{d}{dv_{ji}} \int_{s_k}^{s_{k+1}} \ell_c(\mathbf{c}(t), t) dt = \ell_c(\mathbf{c}, s_{k+1}) \frac{ds_{k+1}}{dv_{ji}} - \ell_c(\mathbf{c}, s_k) \frac{ds_k}{dv_{ji}} + \int_{s_k}^{s_{k+1}} \frac{\partial \ell_c}{\partial \mathbf{c}} \cdot \frac{\partial \mathbf{c}}{\partial v_{ji}} dt\,. \tag{S15}$$

and

$$\frac{d}{dv_{ji}} \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_c \cdot f_c \, dt \tag{S16}$$

$$= \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_c \cdot \frac{df_c}{dv_{ji}} dt \;=\; \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_c \cdot \left\{ \tau_m \frac{d}{dt} \frac{\partial \mathbf{c}}{\partial v_{ji}} + \frac{\partial F}{\partial v_{ji}} \right\} dt \tag{S17}$$

$$= \tau_m \left[\boldsymbol{\lambda}_c \cdot \frac{\partial \mathbf{c}}{\partial v_{ji}} \right]_{s_k}^{s_{k+1}} \tag{S18}$$

$$- \tau_m \int_{s_k}^{s_{k+1}} \left\{ \dot{\boldsymbol{\lambda}}_c \cdot \frac{\partial \mathbf{c}}{\partial v_{ji}} + \boldsymbol{\lambda}_c \cdot \left(\left(\frac{\partial F}{\partial \mathbf{c}} \right)^T \frac{\partial \mathbf{c}}{\partial v_{ji}} + \sum_{\mathrm{x}\in\{u,r,z\}} \left(\frac{\partial F}{\partial \mathbf{a}_{\mathrm{x}}} \right)^T \frac{\partial \mathbf{a}_{\mathrm{x}}}{\partial v_{ji}} \right) \right\} dt\,, \tag{S19}$$

where we first apply Gronwall's theorem [5], 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,

$$\frac{d}{dv_{ji}} \int_{s_k}^{s_{k+1}} \sum_{\mathrm{x} \in \{u,r,z\}} \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot f_{a_\mathrm{x}} \, dt = \sum_{\mathrm{x} \in \{u,r,z\}} \int_{s_k}^{s_{k+1}} \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot \left\{ \tau_s \frac{d}{dt} \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} + \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} \right\} dt \tag{S20}$$

$$= \tau_s \left[\boldsymbol{\lambda}_{a_\mathrm{x}} \cdot \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} \right]_{s_k}^{s_{k+1}} - \tau_s \int_{s_k}^{s_{k+1}} \left\{ \dot{\boldsymbol{\lambda}}_{a_\mathrm{x}} \cdot \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} + \boldsymbol{\lambda}_{a_\mathrm{x}} \cdot \frac{\partial \mathbf{a}_\mathrm{x}}{\partial v_{ji}} \right\} dt \, , \tag{S21}$$

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

Substituting these values into Eq. (S14), 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 (See Fig. S1). 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:

$$\frac{d\mathcal{L}}{dv_{ji}} = \sum_{k=0}^{N} \left\{ \left(l_c^- - l_c^+ \right) \frac{ds}{dv_{ji}} + \tau_s \sum_{\mathrm{x}} \left(\boldsymbol{\lambda}_{a_\mathrm{x}}^- \cdot \frac{\partial \mathbf{a}_\mathrm{x}^-}{\partial v_{ji}} - \boldsymbol{\lambda}_{a_\mathrm{x}}^+ \cdot \frac{\partial \mathbf{a}_\mathrm{x}^+}{\partial v_{ji}} \right) + \tau_m \left(\boldsymbol{\lambda}_c^- \cdot \frac{\partial \mathbf{c}^-}{\partial v_{ji}} - \boldsymbol{\lambda}_c^+ \cdot \frac{\partial \mathbf{c}^+}{\partial v_{ji}} \right) \right\} \tag{S22}$$

$$= \sum_{k=0}^{N} \xi_{\mathrm{x},ijk} \tag{S23}$$

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 [16, 17].

$$c_{n_k}^- - \vartheta_{n_k} = 0 \tag{S24}$$

$$\frac{\partial c_{n_k}^-}{\partial v_{ji}} + \frac{dc_{n_k}^-}{ds} \frac{\partial s}{\partial v_{ji}} = 0 \tag{S25}$$

$$\frac{\partial c_{n_k}^-}{\partial v_{ji}} + \dot{c}_{n_k}^- \frac{\partial s}{\partial v_{ji}} = 0 \tag{S26}$$

$$\frac{\partial s}{\partial v_{ji}} = \frac{-1}{\dot{c}_{n_k}^-} \frac{\partial c_{n_k}^-}{\partial v_{ji}} \, , \tag{S27}$$

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

$$c_{n_k}^+ = 0 \tag{S28}$$

$$\frac{\partial c_{n_k}^+}{\partial v_{ji}} + \dot{c}_{n_k}^+ \frac{\partial s}{\partial v_{ji}} = 0 \tag{S29}$$

which allows us to write

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

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

$$\frac{\partial c_m^+}{\partial v_{ji}} = \frac{\partial c_m^-}{\partial v_{ji}} - \frac{1}{\dot{c}_{n_k}^-} \frac{\partial c_{n_k}^-}{\partial v_{ji}} \left(\dot{c}_m^- - \dot{c}_m^+ \right) \tag{S31}$$

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:

$$\frac{\partial a^{+}_{\mathrm{X},m}}{\partial v_{ji}} = \frac{\partial a^{-}_{\mathrm{X},m}}{\partial v_{ji}} - \frac{1}{\tau_s}\frac{v_{mn_k}\, r^{-}_{\mathrm{X},n_k} c^{-}_{n_k}}{\dot{c}^{-}_{n_k}}\frac{\partial c^{-}_{n_k}}{\partial v_{ji}} + \delta_{in_k}\delta_{jm}c^{-}_{n_k} + c^{-}_{n_k}v_{mn_k}\frac{\partial r^{-}_{\mathrm{X},n_k}}{\partial v_{ji}} - c^{-}_{n_k}v_{mn_k}\frac{\dot{r}^{-}_{\mathrm{X},n_k}}{\dot{c}^{-}_{n_k}}\frac{\partial c^{-}_{n_k}}{\partial v_{ji}} \tag{S32}$$

$$\frac{\partial a^{+}_{\mathrm{X},n_k}}{\partial v_{ji}} = \frac{\partial a^{-}_{\mathrm{X},n_k}}{\partial v_{ji}} \tag{S33}$$

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. (S30),(S31),(S33), (S32) into Eq. (S22), we get:

$$\xi_{\mathrm{X},ijk} = \left\{ \frac{\partial c^{-}_{n_k}}{\partial v_{ji}} \left(\frac{-1}{\dot{c}^{-}_{n_k}}\left(\ell^{+}_{c} - \ell^{-}_{c}\right) + \tau_m\left(\lambda^{-}_{c,n_k} - \frac{\dot{c}^{+}_{n_k}}{\dot{c}^{-}_{n_k}}\lambda^{+}_{c,n_k}\right) + \tau_m\frac{1}{\dot{c}^{-}_{n_k}}\sum_{m\neq n_k}\lambda^{+}_{c,m}\left(\dot{c}^{-}_{m} - \dot{c}^{+}_{m}\right) \right.\right. \tag{S34}$$

$$\left.+\sum_{\mathrm{X}}\frac{r^{-}_{\mathrm{X},n_k}c^{-}_{n_k}}{\dot{c}^{-}_{n_k}}\sum_{m\neq n_k}v_{mn_k}\lambda^{+}_{a_{\mathrm{X}},m} + \tau_s\sum_{\mathrm{X}}\frac{\dot{r}^{-}_{\mathrm{X},n_k}c^{-}_{n_k}}{\dot{c}^{-}_{n_k}}\sum_{m\neq n_k}v_{mn_k}\lambda^{+}_{a_{\mathrm{X}},m}\right) \tag{S35}$$

$$+\tau_m\sum_{m\neq n_k}\frac{\partial c^{-}_{m}}{\partial v_{ji}}\left(\lambda^{-}_{c,m} - \lambda^{+}_{c,m}\right) \tag{S36}$$

$$\tau_s\sum_{\mathrm{X}}\frac{\partial a^{-}_{\mathrm{X},n_k}}{\partial v_{ji}}\left(\left(\lambda^{-}_{a_{\mathrm{X}},n_k} - \lambda^{+}_{a_{\mathrm{X}},n_k}\right) - c^{-}_{n_k}G'(a^{-}_{\mathrm{X},n_k})\sum_{m\neq n_k}v_{mn_k}\lambda a_{\mathrm{X}}, m^{+}\right) \tag{S37}$$

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

$$\left.-\tau_s\delta_{in_k}r^{-}_{\mathrm{X},n_k}c^{-}_{n_k}\sum_{m\neq n_k}\delta_{jm}\lambda^{+}_{a_{\mathrm{X}},m}\right\} \tag{S39}$$

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:

$$\delta_{ab} = \begin{cases} 1 & \text{if} \quad a = b, \\ 0 & \text{otherwise} \end{cases} \tag{S40}$$

Setting the coefficients of $\frac{\partial c^{-}}{\partial v_{ji}}$ 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:

$$\xi_{ijk} = -\tau_s\delta_{in_k}r^{-}_{\mathrm{X},n_k}c^{-}_{n_k}\sum_{m\neq n_k}\delta_{jm}\lambda^{+}_{a_{\mathrm{X}},m} \tag{S41}$$

$$= -\tau_s r^{-}_{\mathrm{X},i}c^{-}_{i}\lambda^{+}_{a_{\mathrm{X}},j}\,. \tag{S42}$$

The jumps/transitions of the adjoint variables are:

$$\lambda^+_{a_{\mathrm{x}},m} = \lambda^-_{a_{\mathrm{x}},m} \tag{S43}$$

$$\lambda^+_{a_{\mathrm{x}},n_k} = \lambda^-_{a_{\mathrm{x}},n_k} - c^-_{n_k} G'(a_{\mathrm{x},n_k}) \sum_{m \neq n_k} v_{mn_k} \lambda^+_{a_{\mathrm{x}},m} \tag{S44}$$

$$\lambda^+_{c,m} = \lambda^-_{c,m} \tag{S45}$$

$$\begin{aligned} \tau_m \dot{c}^+_{n_k} \lambda^+_{c,n_k} &= -(\ell^+_c - \ell^-_c) + \tau_m \dot{c}^-_{n_k} \lambda^-_{c,n_k} + \tau_m \sum_{m \neq n_k} \lambda^+_{c,m} (\dot{c}^-_m - \dot{c}^+_m) \\ &\quad + \tau_s c^-_{n_k} \sum_{\mathrm{x}} \left(\dot{r}^-_{\mathrm{x},n_k} + \frac{r^-_{\mathrm{x},n_k}}{\tau_s} \right) \sum_{m \neq n_k} v_{mn_k} \lambda^+_{a_{\mathrm{x}},m} \, , \end{aligned} \tag{S46}$$

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. (S22) can be calculated from known quantities.

C.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):

$$\frac{d\mathcal{L}}{db_i} = \frac{d}{db_i} \left\{ \sum_{k=0}^{N} \int_{s_k}^{s_{k+1}} \left[\ell_c(\mathbf{c}(t), t) + \boldsymbol{\lambda}_c \cdot f_c + \sum_{\mathrm{x} \in \{u,r,z\}} \boldsymbol{\lambda}_{a_{\mathrm{x}}} \cdot f_{a_{\mathrm{x}}} \right] dt \right\} . \tag{S47}$$

the $\xi^{\text{bias}}_{\mathrm{x},ik}$ term can be shown to be:

$$\xi^{\text{bias}}_{\mathrm{x},ik} = \int_{s_k}^{s_{k+1}} \lambda_{a_{\mathrm{x}},i} \, dt \tag{S48}$$

with

$$\frac{d\mathcal{L}}{db_i} = \sum_{k=0}^{N} \xi^{\text{bias}}_{\mathrm{x},ik} \, . \tag{S49}$$

C.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):

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

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

$$\xi^{\text{input}}_{\mathrm{x},jxk} = -\tau_s \lambda^+_{a_{\mathrm{x}},j} x_x \tag{S51}$$

with

$$\frac{d\mathcal{L}}{du_{jx}} = \sum_{k=0}^{N} \xi^{\text{input}}_{\mathrm{x},jxk} \, . \tag{S52}$$

D Details of experiments

D.1 DVS128 Gesture recognition

In this experiment we use Tonic library [10] 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. (S54). 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. S3(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 S3(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. S3(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.

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.

$$L_{reg} = w_{reg} \left(\frac{1}{N} \frac{1}{n_{\text{units}}} \sum_{n=1}^{N} \sum_{1}^{n_{\text{units}}} H\left(c_i^{\langle t \rangle} - \vartheta_i \right) \quad - 0.05 \right) \tag{S53}$$

$$L_{act} = w_v \left(\frac{1}{N} \frac{1}{n_{\text{units}}} \sum_{n=1}^{N} \sum_{i=1}^{n_{\text{units}}} c_i - (\vartheta_i - 0.05) \right) \tag{S54}$$

where N spans mini-batch.

D.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.

D.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. In all the experiments, we used a batch size of 500, and 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. All models were trained for 200 epochs. 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.

input size (# bits)	network size (# units)	delay (time)	iterations to convergence (mean ± std)
2	2	2	15.0 ± 8.9
2	2	3	10.0 ± 5.3
2	5	2	16.0 ± 13.3
3	32	2	17 ± 11.0
3	32	5	28.3 ± 23.8

Table S1: Model performance of continuous time EGRU trained using hybrid continuous/discrete adjoint method for different parameter values. Performance reported as number of iterations to reach perfect bitwise accuracy (100%) on this task. Mean and standard deviation over 3 runs are reported.

delay (steps)	GRU	**EGRU**
2	87.6 ± 1.7	240.3 ± 22.7
4	128.0 ± 11.4	451.0 ± 99.8
8	321.0 ± 76.9	841.6 ± 132.5
16	1034.6 ± 210.0	1755.0 ± 158.3

Table S2: Comparing memory capacity of the GRU and the discrete-time EGRU on the delay copy task. The binary pattern used is 3 bits wide and 2 bits long, given to a network with 256 units in all rows with a batch size of 1000. Performance reported as number of iterations to reach 98% bitwise accuracy on this task. Mean and standard deviation over 3 runs are reported.

architecture (# units)	parameters	effective MAC (mean±std)	accuracy (%) (mean±std)	activity sparsity (%) (mean±std)	backward sparsity (%) at epoch 100 (mean±std)
LSTM (867)	16.28M	20.97M	87.9±1.0	0	-
GRU (1024)	15.75M	15.73M	88.1±0.8	0	-
EGRU (512)	5.51M	4.31M±93.14K	86.0±1.2	76.1±5.9	45.7±0.7
EGRU (1024)	15.75M	10.71M±206.05K	87.7±2.1	79.8±3.3	54.4±1.2
EGRU (1024)*	110.12M	105.24M±441.30K	85.7±0.9	77.3±7.0	64.6±1.1

Table S3: Model performance over 5 runs for the DVS Gesture recognition task. Effective number of MAC operations as described in section 3.5. * indicates network with 128 × 128 input size, all other networks have scaled input as explained in Section 4.2.

model	accuracy (%)	activity sparsity (%)
Full **EGRU**(1024)	90.2	82.5
without regularization	89.3	76.5
scalar ϑ	88.3	90.8
hard reset	87.2	90.0
no reset	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 D.1.1.

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. 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.

D.3 PTB Language modeling

Our experimental setup largely follows [13]. In particular, we download and preprocess PennTreebank with their published code [1]. Words are projected to a 400-dimensional dense vector by a linear transformation, followed by three RNN layers. The first two RNN layers feature the same hidden dimension, while the hidden dimension of the last RNN layer always equals the word vector embedding dimension. As common in language modeling, we apply cross entropy loss and use weight tying [6, 14].

Initialization of weights and thresholds is the same as for sequential MNIST (see D.2). We apply the regularization strategies of [13], where we found that (temporal) activity regularization only benefits LSTM and GRU. Backpropagation through time is conducted with a variable sequence length. With 95% probability, the sequence length is drawn from $\mathcal{N}(70, 5)$, and with 5% probability the sequence length is drawn from $\mathcal{N}(35, 5)$. We apply variational dropout [4] to the vocabulary with probability $p = 0.1$, to the word embedding vectors with probability $p = 0.4$ as well as to each layer output with probability p_l. DropConnect [15] was applied to the hidden-to-hidden weight matricies with probability p_h. While vocabulary dropout and embedding dropout where fixed for all models, we tuned layer-to-layer and hidden-to-hidden dropout rates for each model individually. We experimented with both Adam [7] and NT-AvSGD [13] optimization procedures. While Adam lead to competitive results for all models, only LSTM models converged using NT-AvSGD. When optimized with SGD based optimizers, both GRU and EGRU fell behind Adam optimized models. Thus, we optimized all models with Adam, and set momentum to 0 as reported in [12]. 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 LSTM and GRU, we used 2.0 for EGRUD. We trained our models for 1000 total epochs. GRU and LSTM achieved their best results with a reduce-on-plateau learning rate schedule, where we multiplied the learning rate with a factor of 0.2 if the loss did not improve for 100 epochs. EGRU worked best with a cosine-annealing learning rate schedule, where the first 500 epochs were trained at constant learning rate λ. Then a cosine decay from λ to $0.1 \cdot \lambda$ was applied for the remaining 500 epochs. A partial grid parameter search was conducted exclusively on the PTB training and validation set. See table S6 for detailed hyperparameters of the best models.

Our experiments exhibit both forward and backward sparsity also for the challenging task of language modeling. For this task, we don't apply any explicit loss terms to improve sparsity. Figure S4 shows how sparsity evolves during the training process. It is evident that forward sparsity naturally evolves from EGRU, even without explicit sparsity regularization. The sparsity of the backward pass is governed by the width parameter ϵ of the pseudo-derivative. We experimented with different values of ϵ on PTB. As shown in figure S5 backward sparsity increases with smaller values of ϵ as expected. We also observe that ϵ acts as a regularizer. The larger EGRU models with 2000 and 2700 hidden units, tends to overfit on PTB if $\epsilon = 1$. Figure S5 shows the dependence of performance measured in perplexity and backward sparsity on the pseudo-derivative support width ϵ.

E Dataset licenses

Penn Treebank [11] 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

---

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

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

> to develop or test products for commercialization, nor can they use LDC data in any commercial product or for any commercial purpose.

Following [13], we download Penn Treebank data from `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`.

The DVS128 Gesture Dataset [1] 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 [10] for Pytorch to preprocess data and to apply transformations.

The sequential MNIST task [8] is based on the MNIST dataset first introduced in [9], available from: `http://yann.lecun.com/exdb/mnist/`.

F Hardware and software details

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 12,000 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 [2] 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).

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 Hidden units	LSTM 1150	GRU 1350	**EGRU** 1350	**EGRU** 2000	**EGRU** 2700
Test ppl (best)	58.9	68.8	64.5	63.7	63.5
Val ppl (best)	61.0	71.2	67.4	66.5	66.4
Val ppl (mean±std)	61.2 ± 0.2	71.7 ± 0.2	67.6 ± 0.1	66.6 ± 0.2	66.7 ± 0.2
Forward sparsity	0 %	0 %	(87.9 ± 0.1) %	(90.43 ± 0.04) %	(93.2 ± 0.1) %
Forward MACs	24.2 M	24.2 M	4.7 M	6.6 M	8.1 M
Learning rate	0.003	0.001	0.003	0.003	0.003
Batch Size	40	80	80	80	80
Dropout p_h	0.5	0.3	0.5	0.5	0.5
Dropout p_l	0.4	0.3	0.3	0.4	0.4
Weight decay	1.2×10^{-6}	1×10^{-5}	1.2×10^{-6}	1.2×10^{-6}	1.2×10^{-6}
AR	2	0	0	0	0
TAR	1	0	0	0	0
Pseudo-derivative ϵ	-	-	0.8	0.8	0.6

Table S6: PTB language modeling: Best parameters for the trained models. Mean and uncertainty are calculated from multiple runs with different random seeds. 10 runs for LSTM, GRU, and EGRU-2700, 5 runs for EGRU-1350, 3 runs for EGRU-2000. Activity regularization (AR) and temporal activity regularization (TAR) [13] only reduced the accuracy of GRU and EGRU based models.

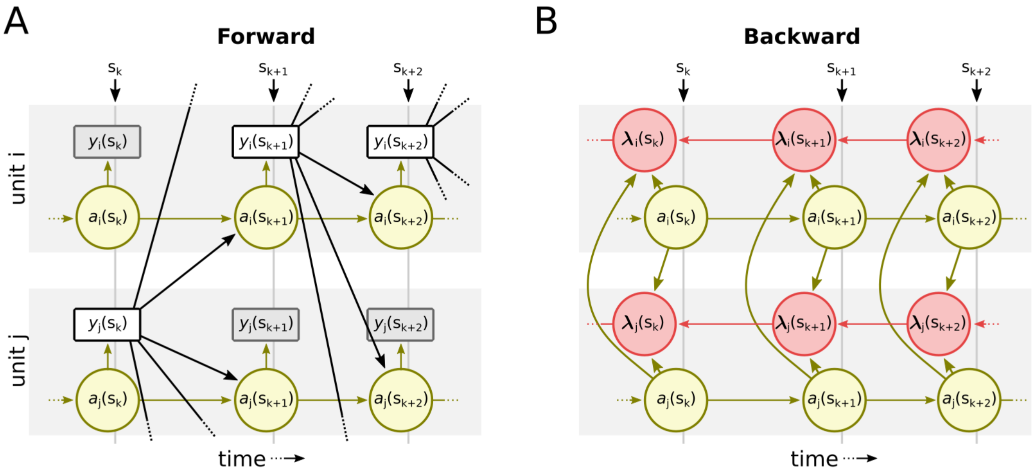


Figure S1: Illustrate the event-based state dynamics for two EGRU units (i and j) (A) Forward dynamics. Information only propagates from units that generate an event. (B) Backward dynamics.

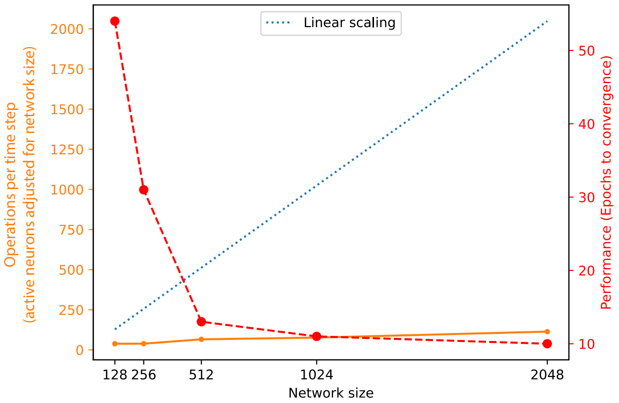
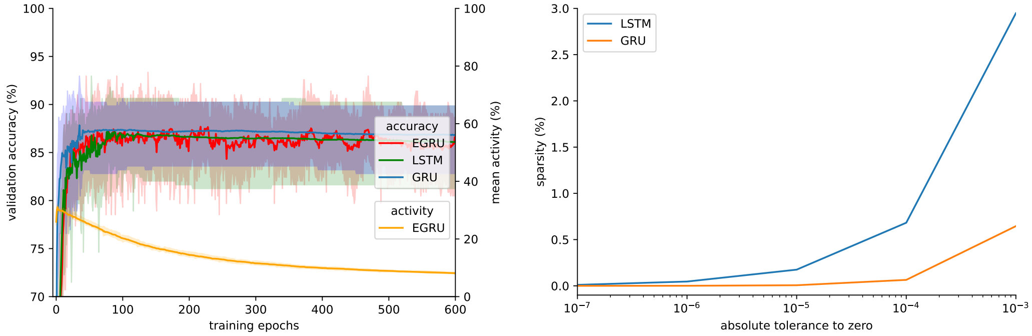


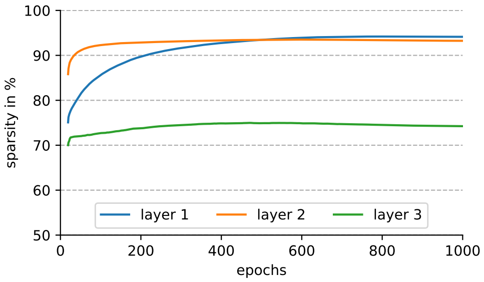
Figure S2: Illustration of the scaling properties of the EGRU on a 14×14 sequential MNIST task (1 run per network size). As the size of the network increases, the network converges faster. Increasing the network size 10x increases the speed of convergence 5x, while increasing the total amount of computation 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.

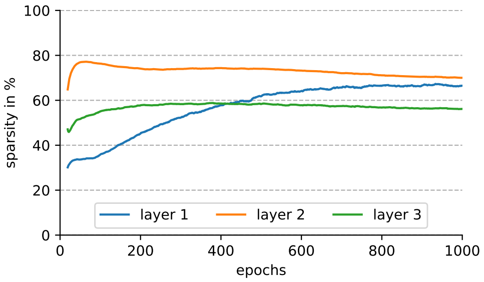


(a)

(b)

Figure S3: **(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.





(a) **(b)**

Figure S4: EGRU with 2000 hidden units on the Penn Treebank language modeling task with pseudo-derivative $\epsilon = 0.6$. **(a)** layer-wise forward sparsity **(b)** layer-wise backward sparsity

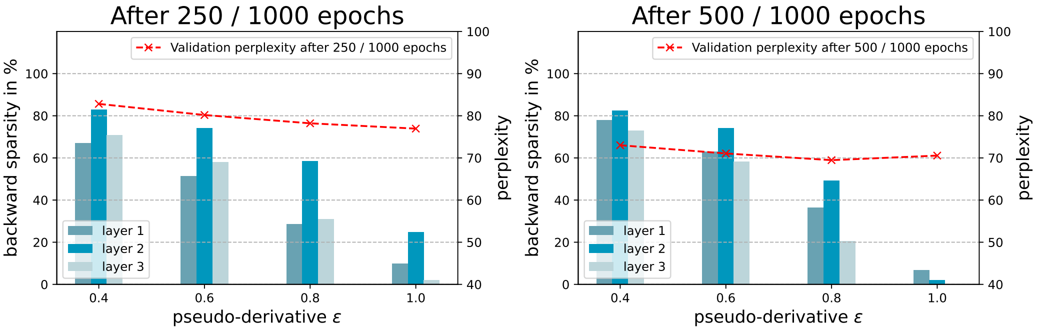


Figure S5: Backward sparsity for EGRU with 2000 hidden units on the Penn Treebank language modeling task with varying pseudo-derivative support ϵ.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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`.

[5] 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.

[6] 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`.

[7] 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`.

[8] Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[9] 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.

[10] 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.

[11] 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.

[12] 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-`.

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

[14] 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`.

[15] 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.

[16] 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.

[17] 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.