Title: Studying Quantum Many-Body Systems with Artificial Neural Networks

Speakers: Stefanie Czischek

Collection/Series: Studying Quantum Many-Body Systems with Artificial Neural Networks, June 15 - June 19

Subject: Other

Date: June 18, 2026 - 1:00 PM

URL: https://pirsa.org/26060061

$$\langle \hat{H}_{TFIM} \rangle = -\mathbb{E}_{\vec{\sigma}\sim|\psi(\vec{\sigma})|^2}\left[\sum_{i=1}^{N}\sigma_i\sigma_{i+2} - h H_{loc}(\vec{\sigma})\right]$$

$$H_{loc}(\vec{\sigma}) = \sum_{i=1}^{N}\frac{\psi(\vec{\sigma}_f(i))}{\psi(\vec{\sigma})}$$

→ data generated from the trained RBM can be used to evaluate any observable

$\sigma_i^z \sigma_{i+2}^z - h H_{loc}(\sigma)\Big]$

trained RBM

observable

- we can also train the RBM to find a ground state representation of a Hamiltonian by minimizing the energy expectation value
  - → we don't need data to train on
  - → gives theoretical insights

find a

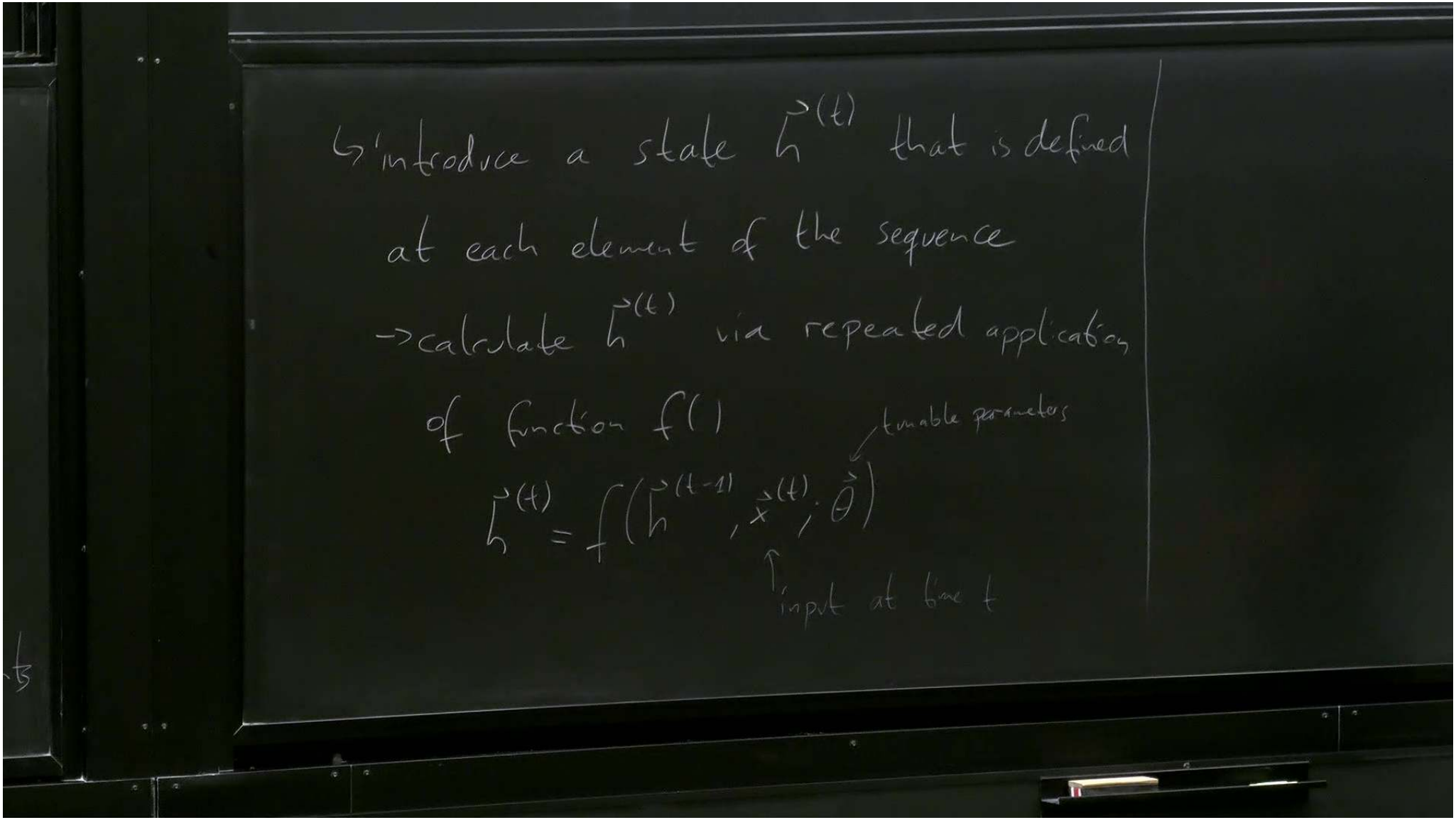
Hamiltonian

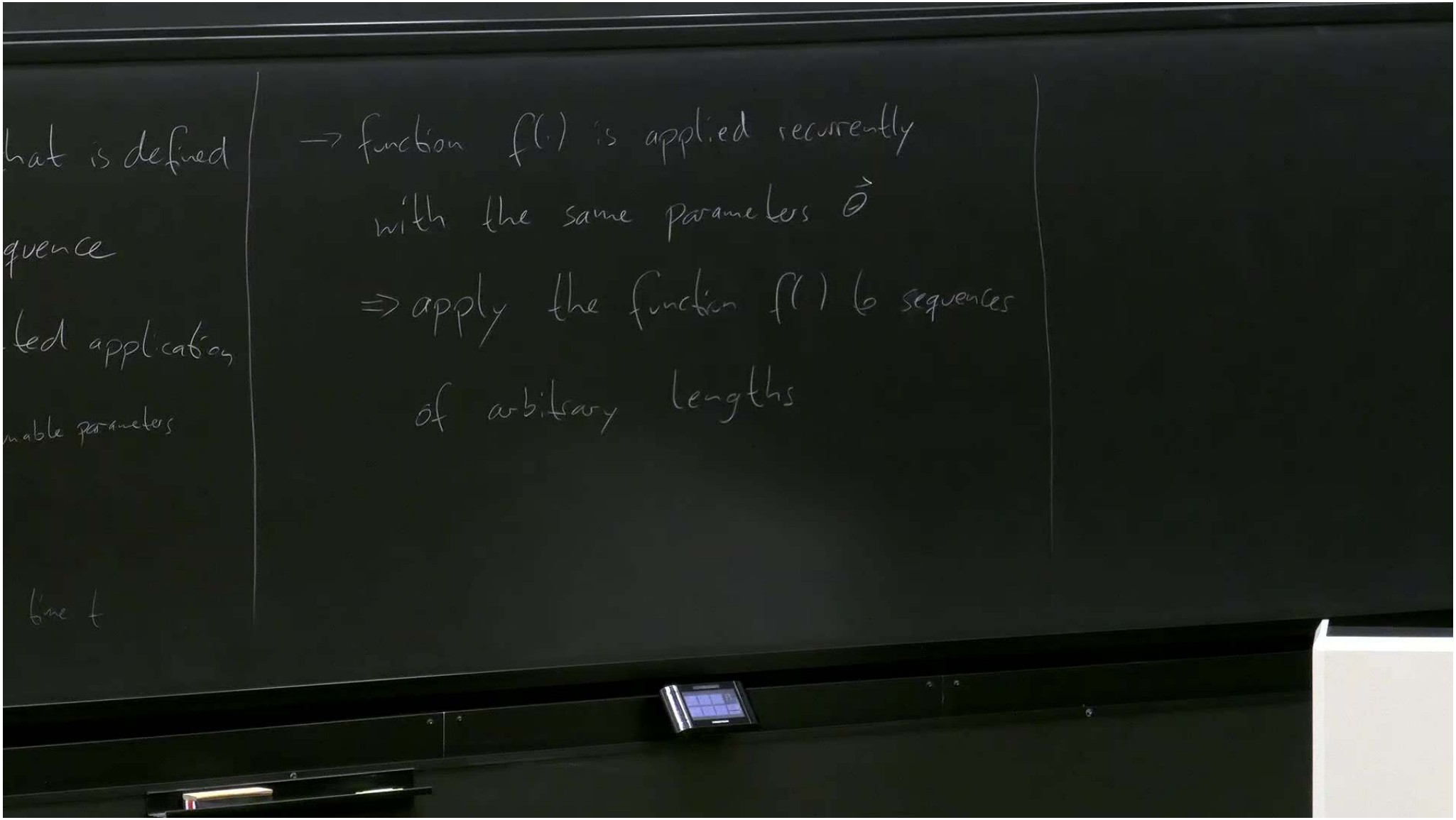
value

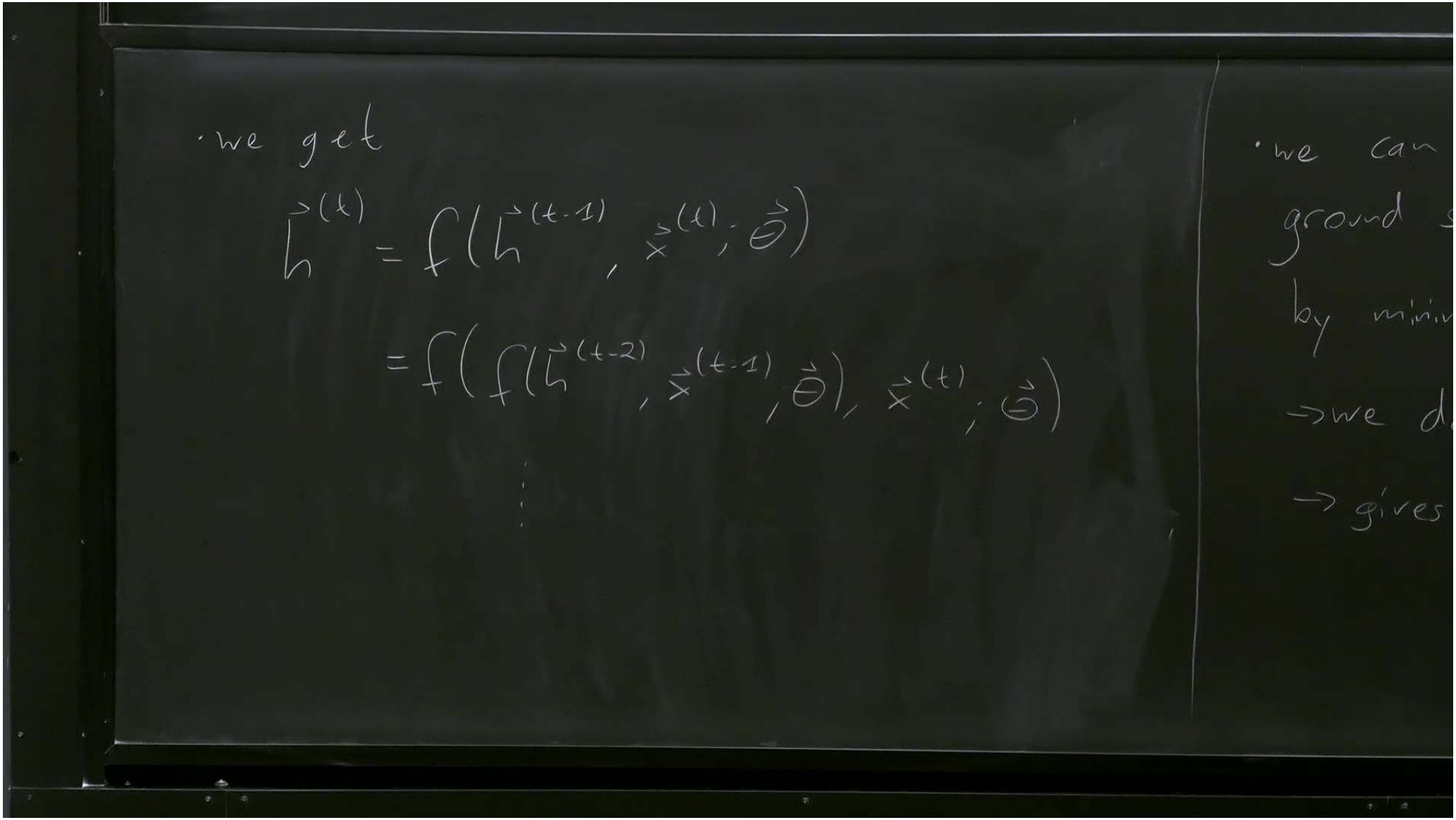
on

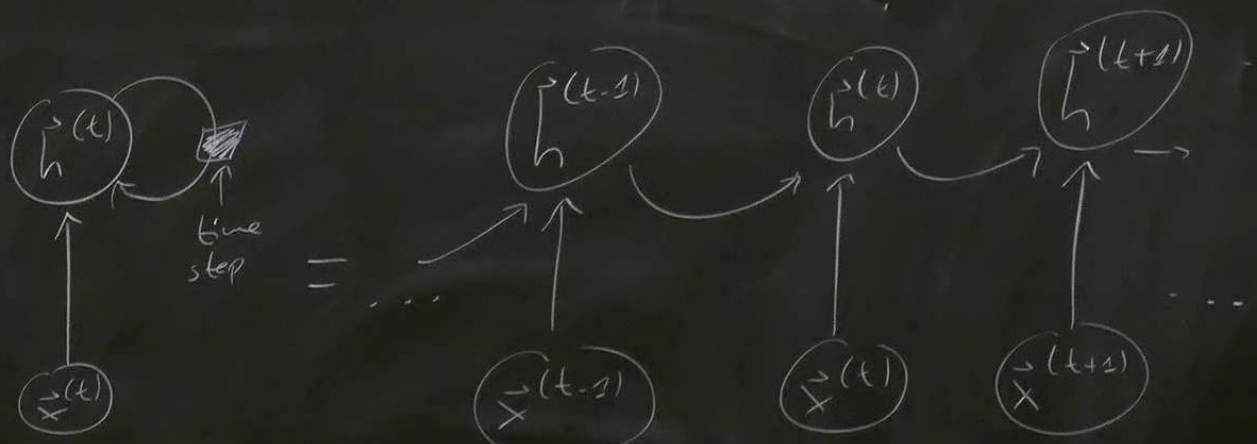
5. Recurrent Neural Networks

- focus on sequential data
  - ↳ text prediction, image generation, ...
- consider each sequence element on its own and introduce dependence on sequence elements


↳ introduce a state $\vec{h}^{(t)}$ that is defined
at each element of the sequence
→ calculate $\vec{h}^{(t)}$ via repeated application
of function $f()$
$\vec{h}^{(t)} = f(\vec{h}^{(t-1)}, \vec{x}^{(t)}; \vec{\theta})$
tunable parameters
input at time t

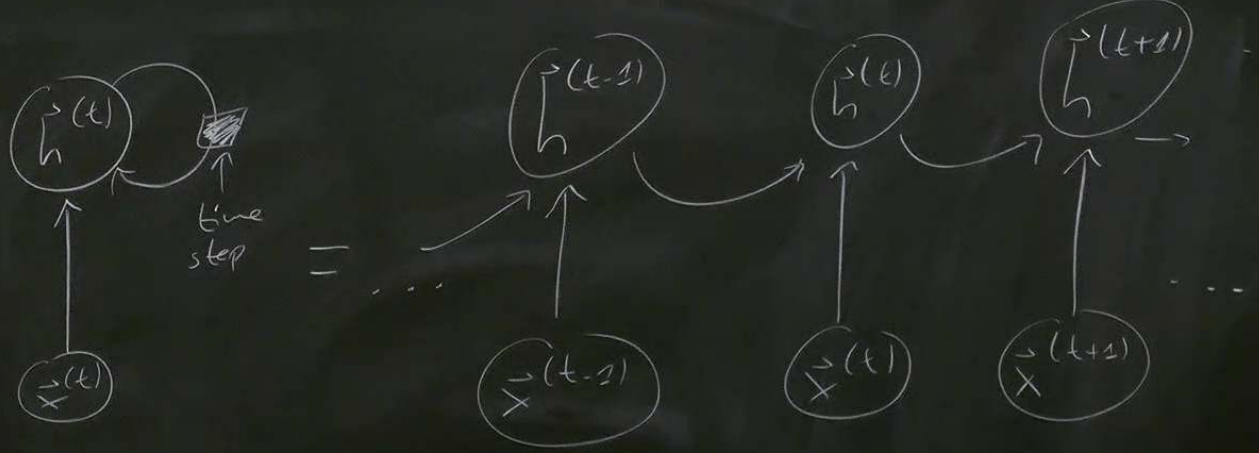

hat is defined
quence
ted applicatio
nable parameters
time t
→ function f(·) is applied recurrently
with the same parameters $\vec{\theta}$
⇒ apply the function f(·) to sequences
of arbitrary lengths


· we get
$\vec{h}^{(t)} = f(\vec{h}^{(t-1)}, \vec{x}^{(t)}; \vec{\theta})$
$= f(f(\vec{h}^{(t-2)}, \vec{x}^{(t-1)}, \vec{\theta}), \vec{x}^{(t)}; \vec{\theta})$
⋮
· we can
ground
by mini
→ we d
→ gives



⇒ two main benefits

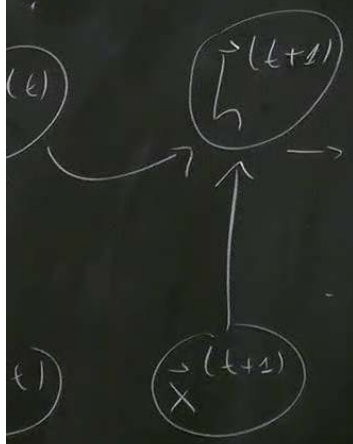
1. Regardless of the sequence length, the learned model always has the same input



2. It is possibl
and the same

⇒ two main benefits

1. Regardless of the sequence length, the learned model always has the same input size.

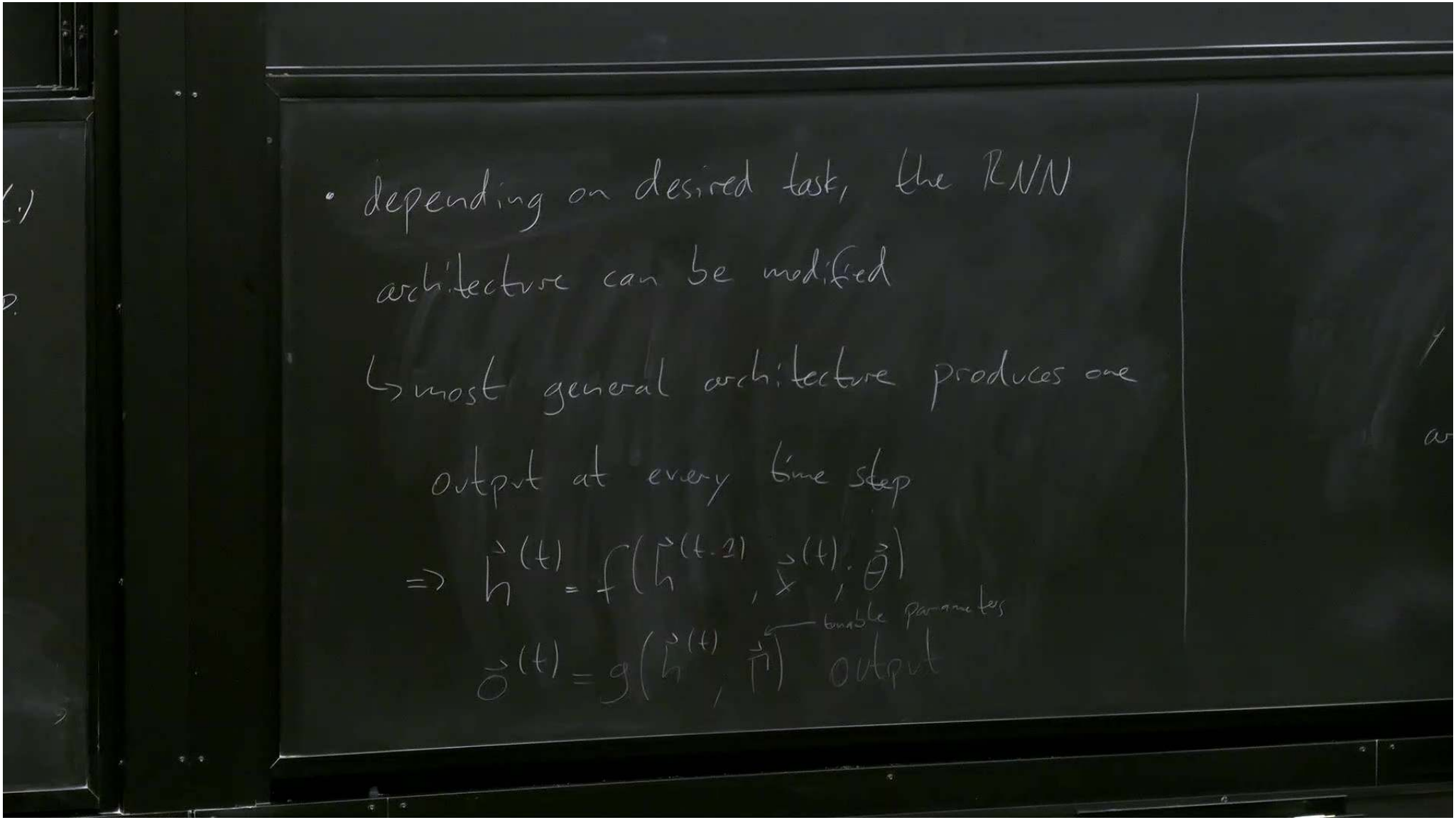


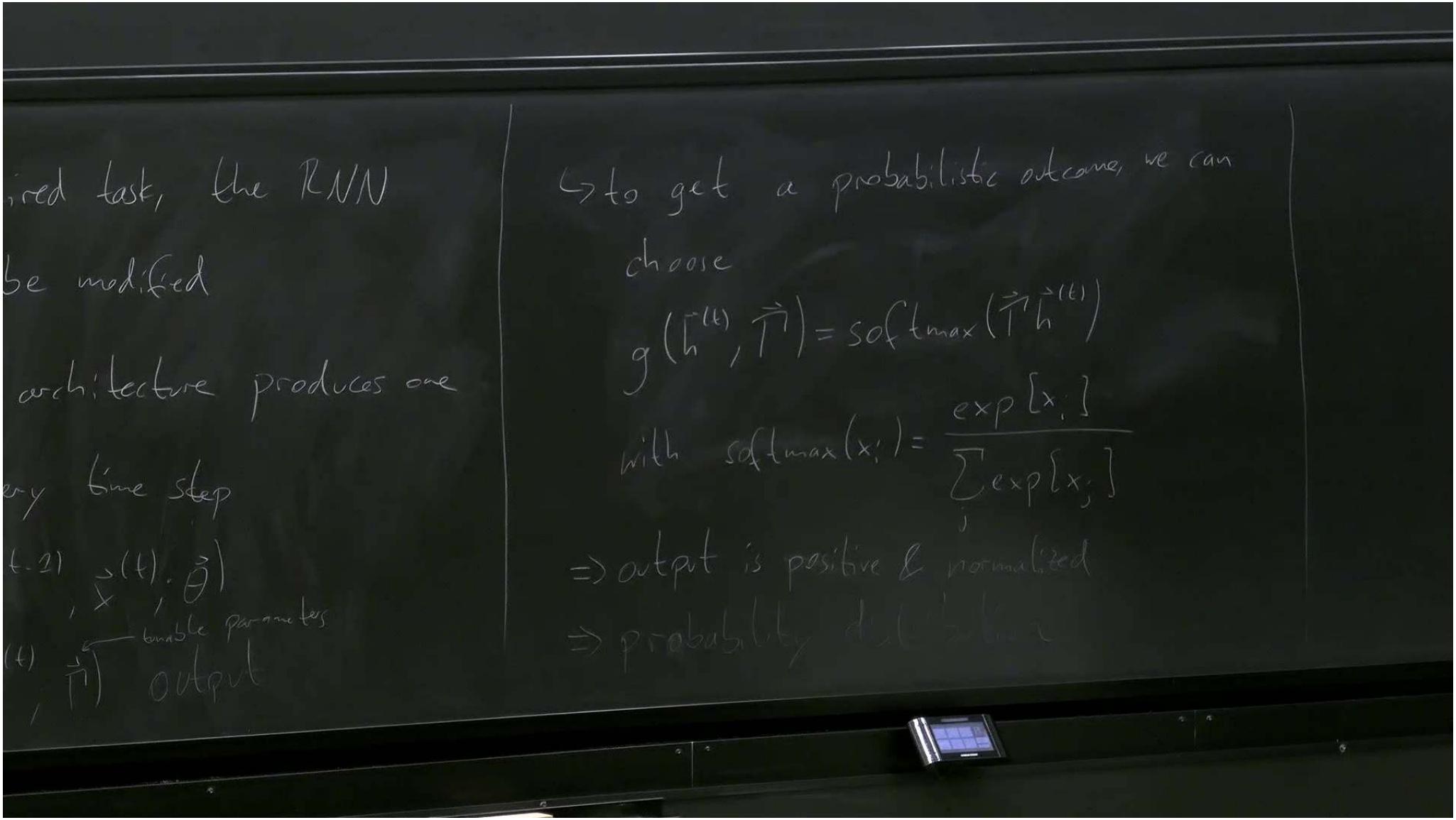
ce length, the

e same input size.

2. It is possible to use the same function $f(\cdot)$ and the same parameters $\vec{\theta}$ at every time step.

- memory about previous sequence elements is encoded in finite-dimensional $\vec{h}^{(t)}$
  $\Rightarrow$ must be lossy for arbitrary lengths


• depending on desired task, the RNN architecture can be modified
↳ most general architecture produces one output at every time step
$\Rightarrow \vec{h}^{(t)} = f(\vec{h}^{(t-1)}, \vec{x}^{(t)}, \vec{\theta})$
$\vec{o}^{(t)} = g(\vec{h}^{(t)}, \vec{\Gamma})$
tunable parameters
output


ired task, the RNN
be modified
architecture produces one
ery time step
t-1) $\vec{\theta}$)
, $\vec{x}$,
(t) trainable parameters
, $\vec{T}$) output
↳ to get a probabilistic outcome, we can
choose
$g(\vec{h}^{(t)}, \vec{T}) = softmax(\vec{T}\vec{h}^{(t)})$
with $softmax(x_i) = \frac{exp[x_i]}{\sum_j exp[x_j]}$
⇒ output is positive & normalized
⇒ probability

⇒ output probability distribution is conditioned on all previous inputs

$$\vec{o}^{(t)} = p\left[\vec{y}^{(t)} \,\middle|\, \vec{x}^{(t)}, \vec{x}^{(t-1)}, \ldots, \vec{x}^{(1)}\right]$$

→ sample a single output $\hat{\vec{y}}^{(t)}$

⇒ two

1. Reg

learn

ndi tioned

]

• autoregressive ansatz:

generated output $\vec{y}^{(t)}$ is the input in the next iteration $\Rightarrow \vec{x}^{(t+1)} = \vec{y}^{(t)}$

$$\Rightarrow \vec{o}^{(t)} = p\left[\vec{y}^{(t)} \,\middle|\, \vec{y}^{(t-1)}, \vec{y}^{(t-2)}, \dots, \vec{y}^{(0)}\right]$$

2. It is

and th

• memory

encoded

⇒ must

input in

$\vec{y}(t)$

$\cdots, \vec{y}(0)\Big]$

$\uparrow$

$\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

$\Rightarrow$ can calculate the probability for one specific sequence of length τ

$$p\left[\vec{y}(0), \vec{y}(1), \ldots, \vec{y}(\tau-1)\right] = \prod_{t=0}^{\tau-1} p\left[\vec{y}(t) \,\middle|\, \vec{y}(t-1), \ldots, \vec{y}(0)\right]$$

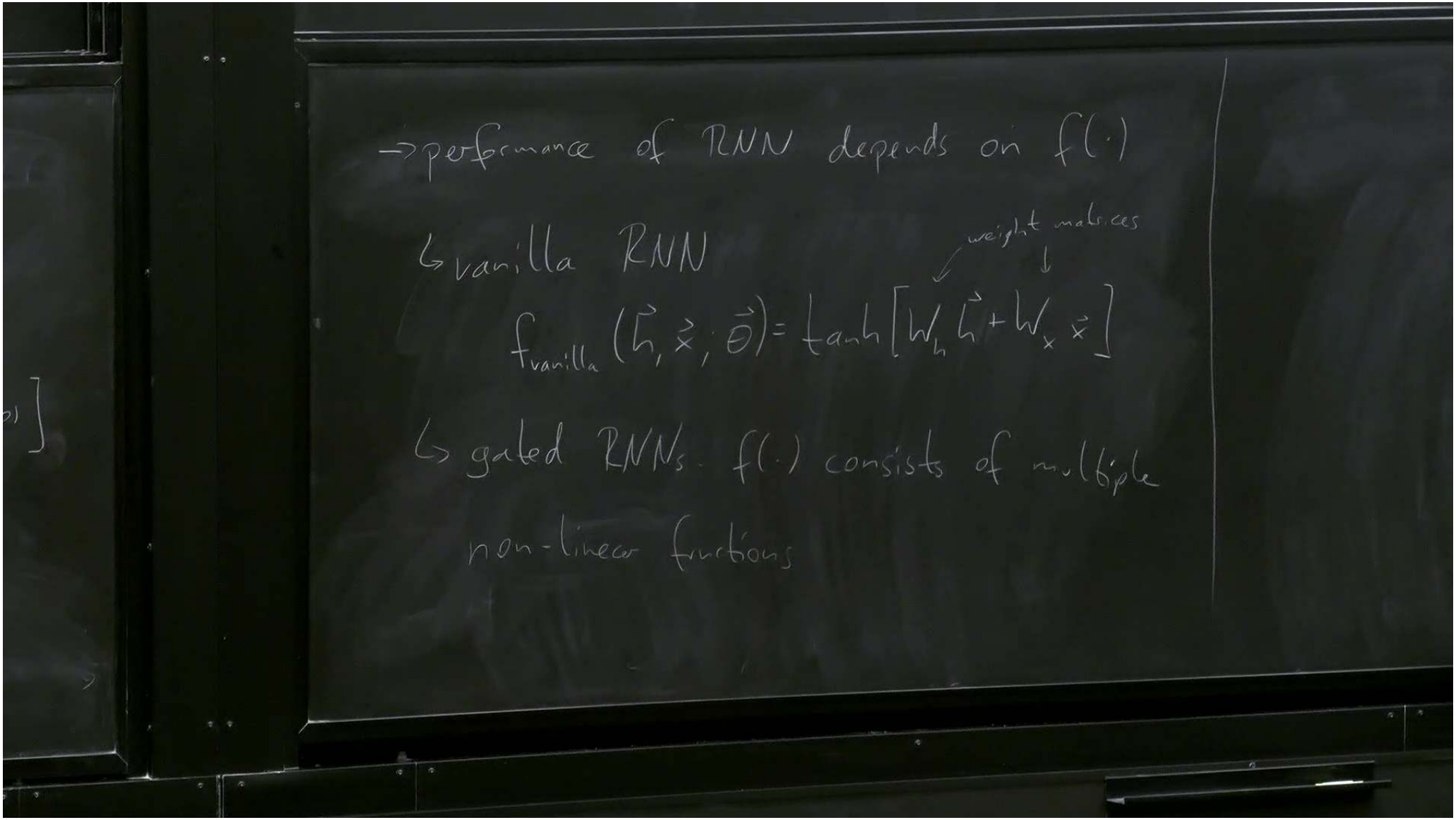
$$= \prod_{t=0}^{\tau-1} p\left[\vec{y}(t) \,\middle|\, \vec{y}(<t)\right]$$

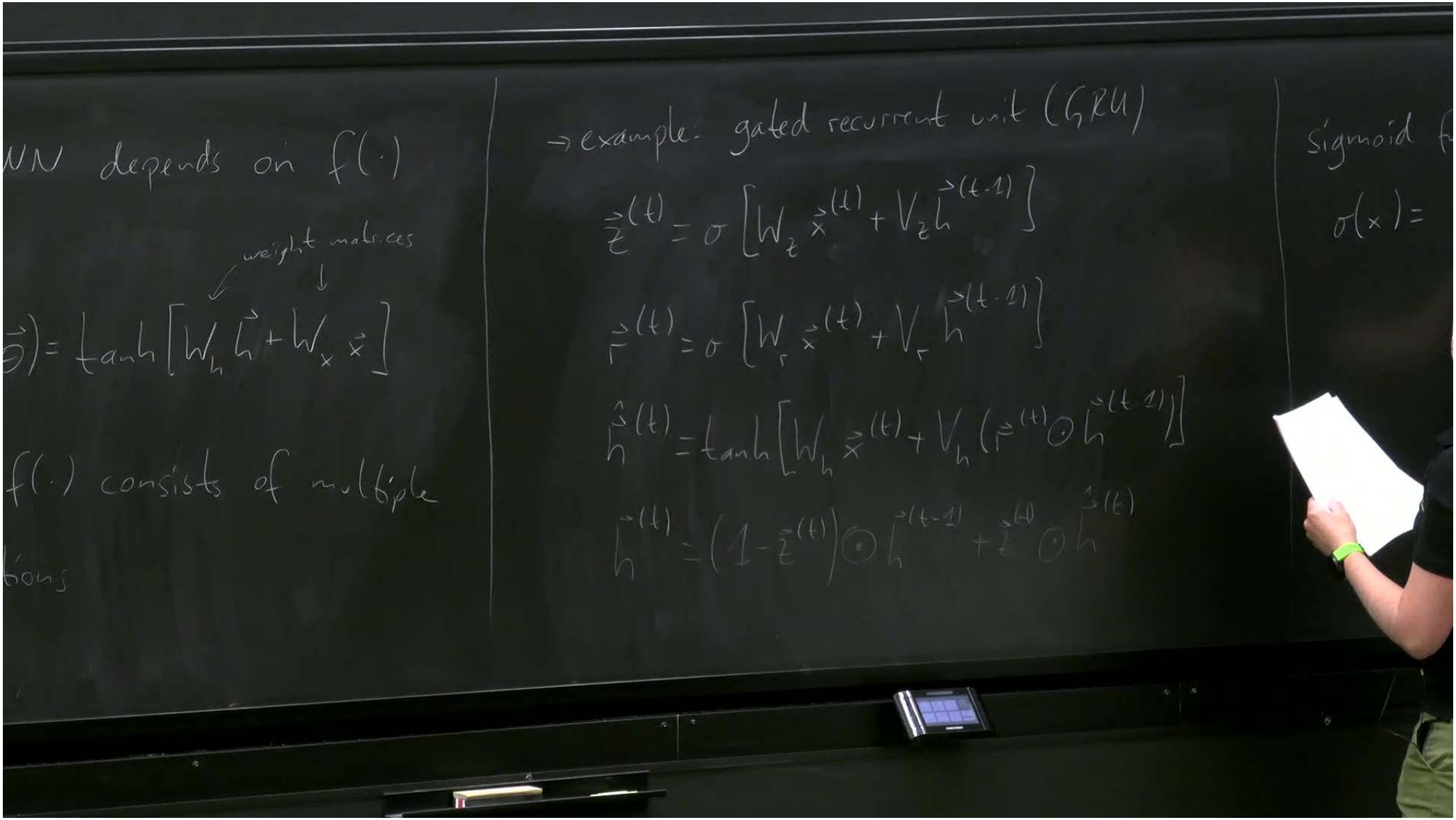
⇒ can calculate the probability for one specific sequence of length τ

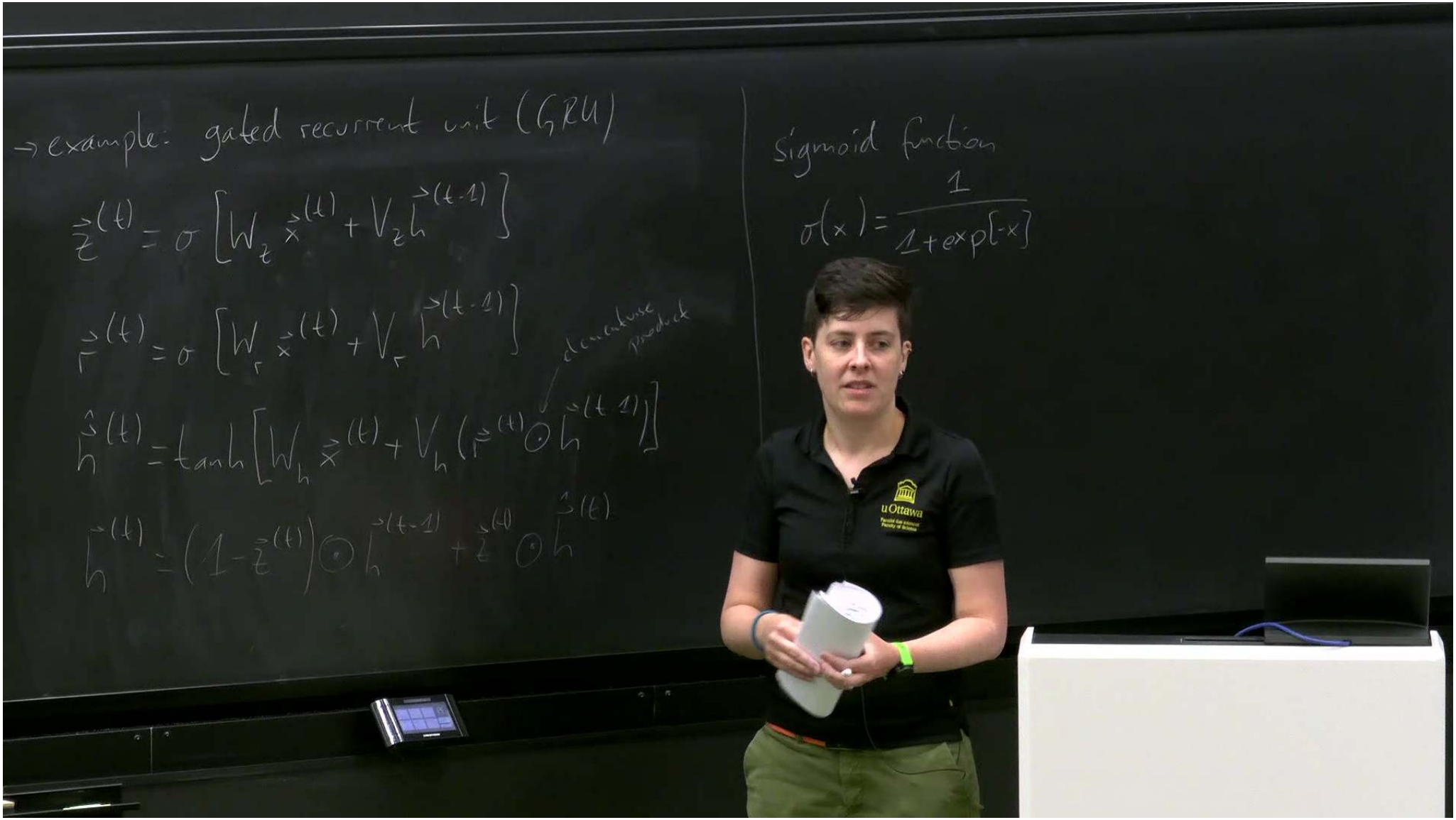
$$P\left[\hat{\vec{y}}^{(0)}, \hat{\vec{y}}^{(1)}, \ldots, \hat{\vec{y}}^{(\tau-1)}\right] = \prod_{t=0}^{\tau-1} P\left[\hat{\vec{y}}^{(t)} \,\middle|\, \hat{\vec{y}}^{(t-1)}, \ldots, \hat{\vec{y}}^{(0)}\right]$$

$$= \prod_{t=0}^{\tau-1} P\left[\hat{\vec{y}}^{(t)} \,\middle|\, \hat{\vec{y}}^{(<t)}\right]$$

⇒ distribution that is encoded in RNN


→ performance of RNN depends on f(·)
↳ vanilla RNN
weight matrices
$f_{\text{vanilla}}(\vec{h}, \vec{x}; \vec{\theta}) = \tanh\left[W_h \vec{h} + W_x \vec{x}\right]$
↳ gated RNNs. f(·) consists of multiple non-linear functions


NN depends on f(·)
weight matrices
$\vec{s}) = \tanh\left[W_h \vec{h} + W_x \vec{x}\right]$
f(·) consists of multiple
tions
→ example: gated recurrent unit (GRU)
$\vec{z}^{(t)} = \sigma\left[W_z \vec{x}^{(t)} + V_z \vec{h}^{(t-1)}\right]$
$\vec{r}^{(t)} = \sigma\left[W_r \vec{x}^{(t)} + V_r \vec{h}^{(t-1)}\right]$
$\hat{h}^{(t)} = \tanh\left[W_h \vec{x}^{(t)} + V_h (\vec{r}^{(t)} \odot \vec{h}^{(t-1)})\right]$
$\vec{h}^{(t)} = (1 - \vec{z}^{(t)}) \odot \vec{h}^{(t-1)} + \vec{z}^{(t)} \odot \hat{h}^{(t)}$
Sigmoid f
$\sigma(x) =$


→ example: gated recurrent unit (GRU)
sigmoid function
$\sigma(x) = \frac{1}{1+\exp[-x]}$
$\vec{z}^{(t)} = \sigma\left[W_z \vec{x}^{(t)} + V_z \vec{h}^{(t-1)}\right]$
$\vec{r}^{(t)} = \sigma\left[W_r \vec{x}^{(t)} + V_r \vec{h}^{(t-1)}\right]$
$\hat{h}^{(t)} = \tanh\left[W_h \vec{x}^{(t)} + V_h (\vec{r}^{(t)} \odot \vec{h}^{(t-1)})\right]$
$\vec{h}^{(t)} = (1 - \vec{z}^{(t)}) \odot \vec{h}^{(t-1)} + \vec{z}^{(t)} \odot \hat{h}^{(t)}$
elementwise product
uOttawa