Introduction to hybrid modeling

Basil Kraft

February 2023

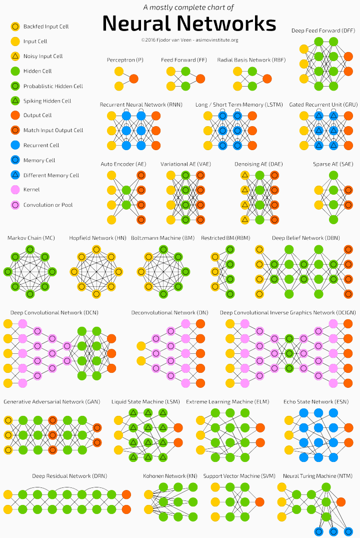
MPI for Biogeochemistry
Department Biogeochemical Integration
Jena

Schedule

- Quick recap:
  - Neural networks
  - Demo: neural networks with PyTorch
- Modeling snow with a hybrid model
  - Background & data simulation
  - Demo: hybrid modeling

Artificial neural networks (ANNs)

Artificial neural networks (ANNs)



- ANNs are highly flexible; We can adapt the model architecture to our needs / to fit the problem
- ANN's performance scales with data
- Computation based on (many) matrix operations; computation is scaleable

Artificial neural networks (ANNs)

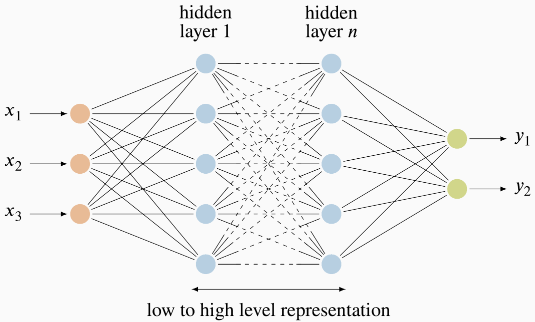


Figure 1: ANNs are hierarchical feature learners.

- ANNs consist of (learned) sequentially arranged nonlinear transformations
- Each transformation is simple, but in combination, ANNs are highly expressive

Artificial neural networks (ANNs)

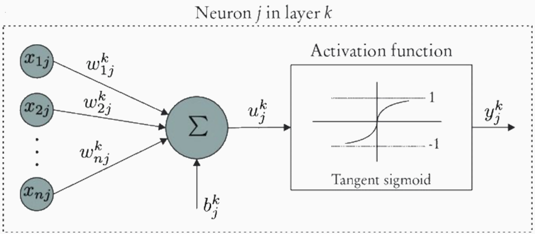


Figure 2: The perceptron: simple architecture for binary classification (or regression with target domain $-1 < y < 1$)

- The building blocks of ANNs are simple
- So-called activation functions[a] add non-linearity

[a] a non-linear, differentiable function

Artificial neural networks (ANNs)

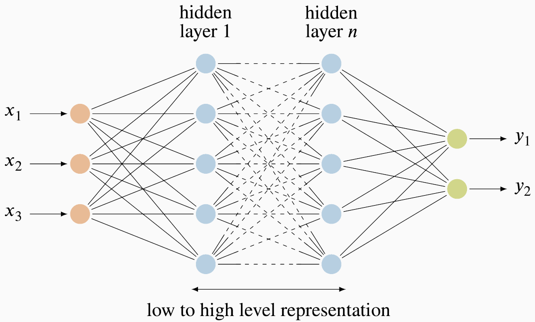


Figure 3: ANNs are hierarchical feature learners.

- The building blocks of ANNs are simple
- So-called activation functions[a] add non-linearity

[a] a non-linear, differentiable function

Artificial neural networks (ANNs)

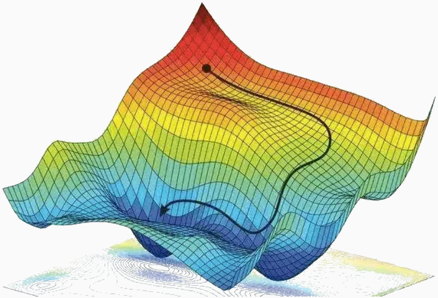


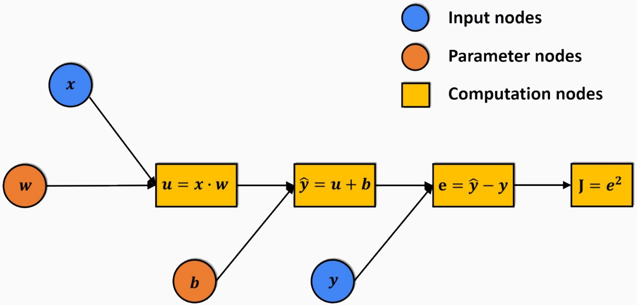
Figure 4: A 2-D loss landscape: the z-axis is the loss (which we seek to minimize), x and z are parameter space.

- ANNs are optimized with gradient descent
- The parameters are adapted iteratively by changing them in little “downhill” steps

Artificial neural networks (ANNs)

- Example: a simple linear model with one input and one output
- How do we efficiently optimize this?

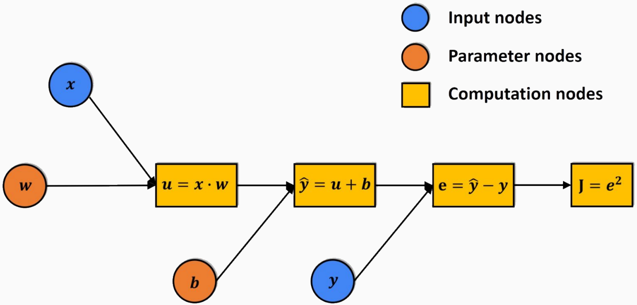
$$\frac{\partial J}{\partial w} = ?$$
$$\frac{\partial J}{\partial b} = ?$$

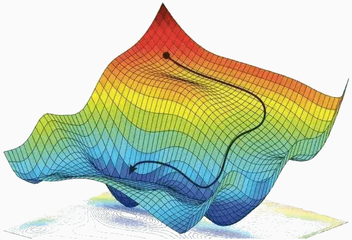


Artificial neural networks (ANNs)

- Example: a simple linear model with one input and one output
- How do we efficiently optimize this?

$$\frac{\partial J}{\partial w} = ?$$
$$\frac{\partial J}{\partial b} = ?$$

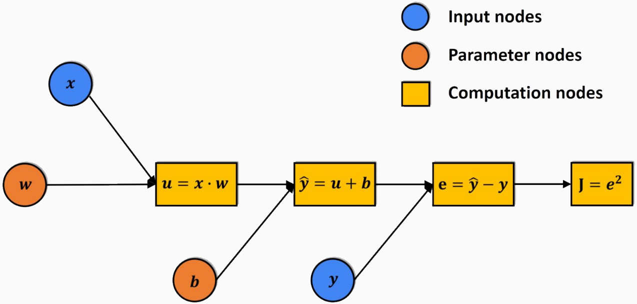


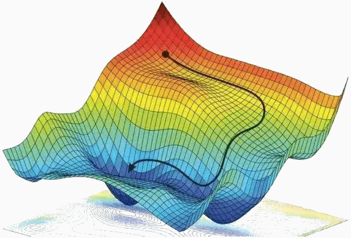


Artificial neural networks (ANNs)

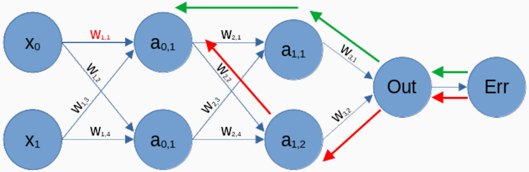
- We apply the chain rule: $\frac{\partial a}{\partial c} = \frac{\partial a}{\partial b}\frac{\partial b}{\partial c}$
- Blue terms have to be calculated only once!

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial u}\frac{\partial u}{\partial w}$$
$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial b}$$





Artificial neural networks (ANNs)

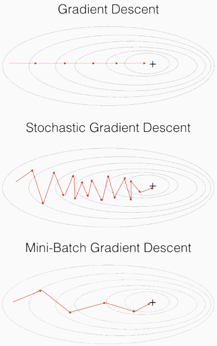


- The error is backpropagated layer-wise abd summed up at nodes
- Backpropagation is agnostic of the full model structure, we only need to know the local gradient computation.

Pseudo-code for neural net optimization

- Multiple iterations through training data (epochs)
- In each epoch, we iterate the data in minibatches without replacement

initialize(model.params)
for e in $Epochs$ **do**
    for $batch$ in $Data$ **do**
        zerograds($model.params.grads$) # reset gradients
        $x, y =$ getXY($batch$) # get batch
        $\hat{y} = model(x)$ # model forward pass
        $loss = f_{\text{loss}}(\hat{y}, y)$ # loss calculation
        $model.params.grads = backprop(loss)$ # compute gradients
        $update(model.params)$
    end for
end for



Artificial neural networks (ANNs)

- ANNs are extremely powerful (universal function approximators)

- ANNs are extremely powerful (universal function approximators)
- Using large models requires lots of training data

Artificial neural networks (ANNs)

- ANNs are extremely powerful (universal function approximators)
- Using large models requires lots of training data
- We can make models more data-efficient by using tailored architectures

Artificial neural networks (ANNs)

- ANNs are extremely powerful (universal function approximators)
- Using large models requires lots of training data
- We can make models more data-efficient by using tailored architectures
- Careful with overfitting

Artificial neural networks (ANNs)

- ANNs are extremely powerful (universal function approximators)
- Using large models requires lots of training data
- We can make models more data-efficient by using tailored architectures
- Careful with overfitting
- Luckily, there are frameworks for building and training ANNs (PyTorch, Tensorflow, etc.)

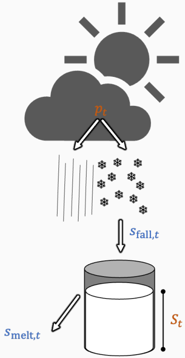
Demo: neural networks with PyTorch

Hybrid modeling

Hybrid modeling

Hybrid model of snow water equivalent

We model snow water equivalent S as a function of precipitation p, air temperature T, and solar irradiation r:
$S_t = f(p_t, T_t, r_t)$



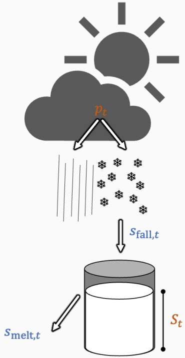
Hybrid model of snow water equivalent

We use a neural network to learn the process from data:

$S_t = f_{\mathrm{NN}}(p_t, T_t, r_t)$

This may work quite well, but

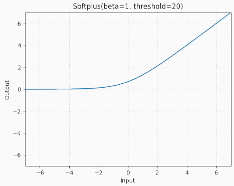
- The model can predict inplausible values (*e.g.*, $S < 0$, or snowfall but no precipitation)
- We have no control over the process, no prior knowledge used
- We cannot physically interpret what the model learned
- S is a state we can't predict from just doday's weather
- **We can do better**



Hybrid modeling

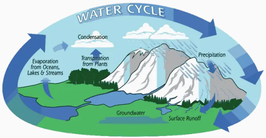
Restricting the output space such that $S \geq 0$

$$S_t = \mathrm{Sortplus}(f_{\mathrm{NN}}(p_t, T_t, r_t))$$



Account for memory effects

$$S_t = \mathrm{Sortplus}(f_{\mathrm{NN}}(p_{k \leq t}, T_{k \leq t}, r_{k \leq t}))$$

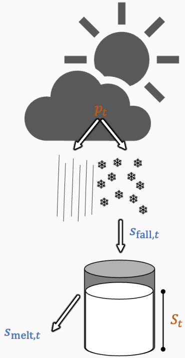


$\rightarrow$ standard practices, not hybrid modeling

Hybrid modeling

Implement explicit process knowledge (hybrid modeling)

- We assume that two main processes contribute to ΔS (we simplify a bit)
  - snowmelt s_{melt}
  - snowfall s_{fall}
- Update snow with the two quantities:
  - $S_t = S_{t-1} + \overbrace{\text{Sortplus}(f^1_{\text{NN}}(p_t, T_t, r_t))}^{s_{\text{fall},t}} - \overbrace{\text{Sortplus}(f^2_{\text{NN}}(p_t, T_t, r_t))}^{s_{\text{melt},t}}$
- Is this a good idea? Why (not)?



Hybrid modeling

- Snow fall without precipitation possible
- Snow melt with temperatures below freezing point
- Negative snow possible
- Equifinality: $S_t = S_{t-1} + s_{\mathrm{fall},t} - s_{\mathrm{melt},t}$
  $\rightarrow$ model not very useful; we need better constraints!

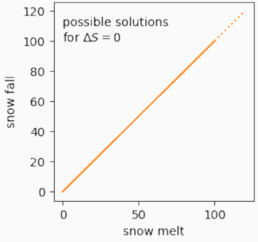
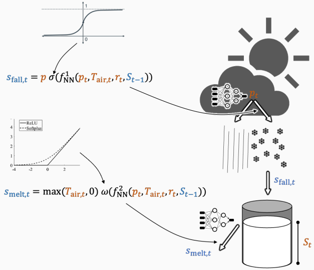


Figure 5: Equifinality: different solutions lead to the same result

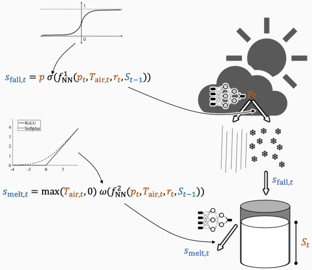
Hybrid modeling

- $s_{\mathrm{fall},t} = p_t \overbrace{\alpha_{\mathrm{fall},t}}^{f^1_{\mathrm{NN}}(p_t, T_t, r_t, S_{t-1})} , \quad 0 <= \alpha_{\mathrm{fall},t} <= 1$



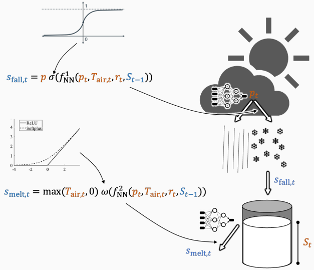
Hybrid modeling

- $s_{\text{fall},t} = p_t \overbrace{\alpha_{\text{fall},t}}^{f^1_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{fall},t} <= 1$
- $s_{\text{melt},t} = \max(0, T_t) \overbrace{\alpha_{\text{melt},t}}^{f^2_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{melt},t}$



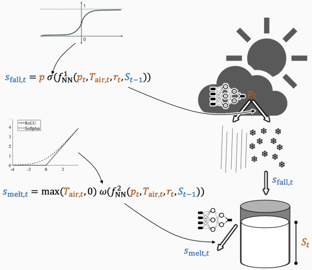
Hybrid modeling

- $s_{\text{fall},t} = p_t \overbrace{\alpha_{\text{fall},t}}^{f^1_{\text{NN}}(p_t, T_t, r_t, S_{t-1})}, \quad 0 <= \alpha_{\text{fall},t} <= 1$
- $s_{\text{melt},t} = \max(0, T_t) \overbrace{\alpha_{\text{melt},t}}^{f^2_{\text{NN}}(p_t, T_t, r_t, S_{t-1})}, \quad 0 <= \alpha_{\text{melt},t}$
- $S_t = \max(0, S_{t-1} + s_{\text{fall},t} - s_{\text{melt},t})$ ($s_{\text{melt},t}$ is actually potential snow melt here)



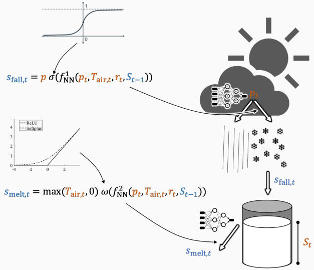
Hybrid modeling

- $s_{\text{fall},t} = p_t \overbrace{\alpha_{\text{fall},t}}^{f^1_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{fall},t} <= 1$
- $s_{\text{melt},t} = \max(0, T_t) \overbrace{\alpha_{\text{melt},t}}^{f^2_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{melt},t}$
- $S_t = \max(0, S_{t-1} + s_{\text{fall},t} - s_{\text{melt},t})$ ($s_{\text{melt},t}$ is actually potential snow melt here)
- Only snow fall with $p > 0$, only snow melt with $T > 0$

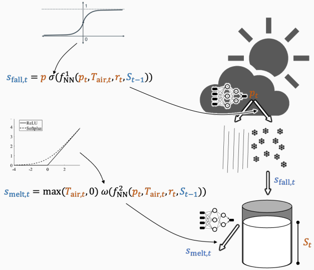


Hybrid modeling

- $s_{\text{fall},t} = p_t \overbrace{\alpha_{\text{fall},t}}^{f^1_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{fall},t} <= 1$
- $s_{\text{melt},t} = \max(0, T_t) \overbrace{\alpha_{\text{melt},t}}^{f^2_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{melt},t}$
- $S_t = \max(0, S_{t-1} + s_{\text{fall},t} - s_{\text{melt},t})$ ($s_{\text{melt},t}$ is actually potential snow melt here)
- Only snow fall with $p > 0$, only snow melt with $T > 0$
- No negative snow

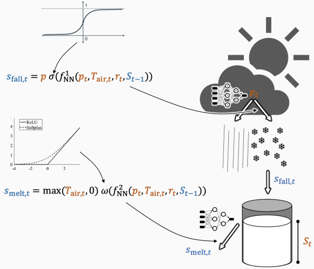


- $s_{\text{fall},t} = p_t \overbrace{\alpha_{\text{fall},t}}^{f^1_{\text{NN}}(p_t, T_t, r_t, S_{t-1})}, \quad 0 <= \alpha_{\text{fall},t} <= 1$
- $s_{\text{melt},t} = \max(0, T_t) \overbrace{\alpha_{\text{melt},t}}^{f^2_{\text{NN}}(p_t, T_t, r_t, S_{t-1})}, \quad 0 <= \alpha_{\text{melt},t}$
- $S_t = \max(0, S_{t-1} + s_{\text{fall},t} - s_{\text{melt},t})$ ($s_{\text{melt},t}$ is actually potential snow melt here)
- Only snow fall with $p > 0$, only snow melt with $T > 0$
- No negative snow
- The neural nets are informed about the current snow state as we pas S_{t-1} as input.



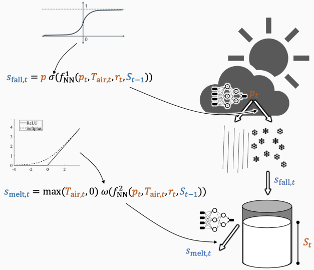
Hybrid modeling

- $s_{\text{fall},t} = p_t \overbrace{\alpha_{\text{fall},t}}^{f^1_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{fall},t} <= 1$
- $s_{\text{melt},t} = \max(0, T_t) \overbrace{\alpha_{\text{melt},t}}^{f^2_{\text{NN}}(p_t, T_t, r_t, S_{t-1})} \quad , \quad 0 <= \alpha_{\text{melt},t}$
- $S_t = \max(0, S_{t-1} + s_{\text{fall},t} - s_{\text{melt},t})$ ($s_{\text{melt},t}$ is actually potential snow melt here)
- Only snow fall with $p > 0$, only snow melt with $T > 0$
- No negative snow
- The neural nets are informed about the current snow state as we pas S_{t-1} as input.
- We gain interpretability of snow melt and snow fall, which we do not have with plain neural network appraoch



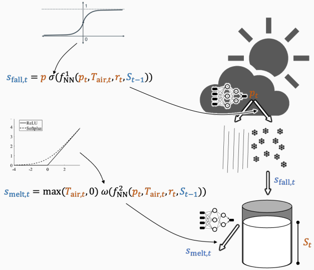
Hybrid modeling

- $s_{\text{fall},t} = p_t \overbrace{\alpha_{\text{fall},t}}^{f^1_{\text{NN}}(p_t, T_t, r_t, S_{t-1})}, \quad 0 <= \alpha_{\text{fall},t} <= 1$
- $s_{\text{melt},t} = \max(0, T_t) \overbrace{\alpha_{\text{melt},t}}^{f^2_{\text{NN}}(p_t, T_t, r_t, S_{t-1})}, \quad 0 <= \alpha_{\text{melt},t}$
- $S_t = \max(0, S_{t-1} + s_{\text{fall},t} - s_{\text{melt},t})$ ($s_{\text{melt},t}$ is actually potential snow melt here)
- Only snow fall with $p > 0$, only snow melt with $T > 0$
- No negative snow
- The neural nets are informed about the current snow state as we pas S_{t-1} as input.
- We gain interpretability of snow melt and snow fall, which we do not have with plain neural network appraoch
- What are potential pitfalls?



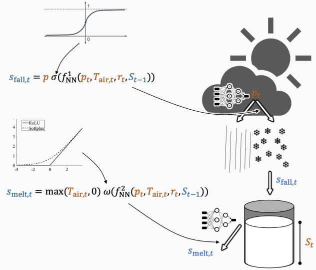
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?



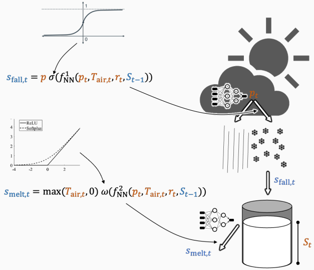
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:



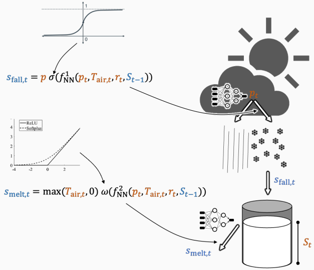
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
  - *E.g.*, snow fall includes wind transport from neighboring areas



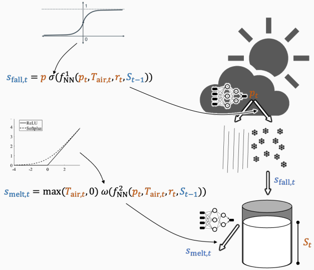
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
  - *E.g.*, snow fall includes wind transport from neighboring areas
  - *E.g.*, snow melt includes sublimation



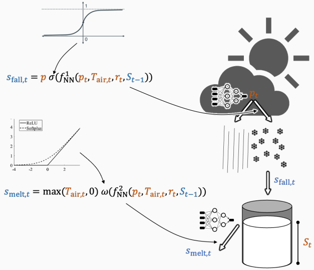
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
  - *E.g.*, snow fall includes wind transport from neighboring areas
  - *E.g.*, snow melt includes sublimation
- Biases due to wrong assumptions:



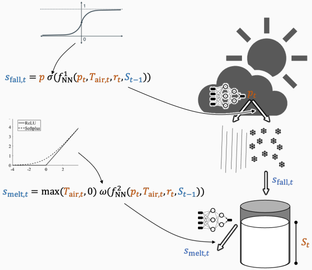
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
  - *E.g.*, snow fall includes wind transport from neighboring areas
  - *E.g.*, snow melt includes sublimation
- Biases due to wrong assumptions:
  - *E.g.*, we could have snow melt with $T < 0$ because T is air temperature.



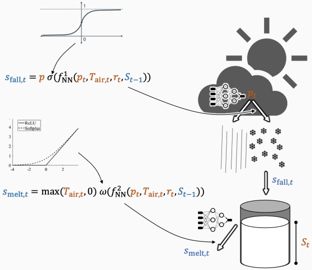
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
  - *E.g.*, snow fall includes wind transport from neighboring areas
  - *E.g.*, snow melt includes sublimation
- Biases due to wrong assumptions:
  - *E.g.*, we could have snow melt with $T < 0$ because T is air temperature.
- Biases due to data:



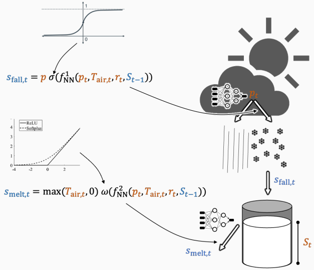
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
  - *E.g.*, snow fall includes wind transport from neighboring areas
  - *E.g.*, snow melt includes sublimation
- Biases due to wrong assumptions:
  - *E.g.*, we could have snow melt with $T < 0$ because T is air temperature.
- Biases due to data:
  - What if precipitation is underestimated? (actually the case for snowfall)



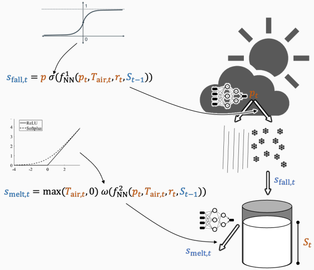
Hybrid modeling

- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
  - *E.g.*, snow fall includes wind transport from neighboring areas
  - *E.g.*, snow melt includes sublimation
- Biases due to wrong assumptions:
  - *E.g.*, we could have snow melt with $T < 0$ because T is air temperature.
- Biases due to data:
  - What if precipitation is underestimated? (actually the case for snowfall)
  - What if air temperature is overestimated?



Hybrid modeling

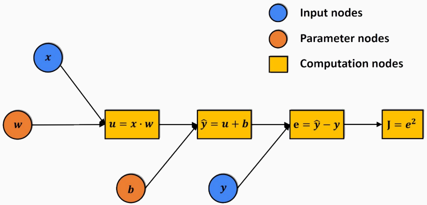
- Equifinality if $p > 0 \& T > 0$?
- Biases due to neglected processes:
    - *E.g.*, snow fall includes wind transport from neighboring areas
    - *E.g.*, snow melt includes sublimation
- Biases due to wrong assumptions:
    - *E.g.*, we could have snow melt with $T < 0$ because T is air temperature.
- Biases due to data:
    - What if precipitation is underestimated? (actually the case for snowfall)
    - What if air temperature is overestimated?
    - What if snow is biased? (many observation-based snow products saturate at higher levels)



Hybrid modeling

How do we implement hybrid models?

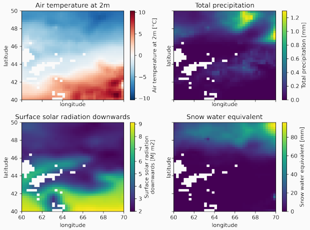
- There is no qualitative difference between neural network layers and physical equations
- The physical equations need to be differentiable and that's it, backpropagation still works!

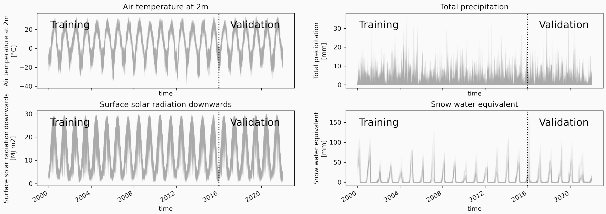


Demo: hybrid modeling of snow water equivalent with PyTorch

Artificial neural networks (ANNs)

- Daily data from ERA5 reanalysis
- Training/validation split in time (to keep things simple)





Summary

- Hybrid models can be implemented in deep learning frrameworks such as PyTorch
- Model design, data, and assuptions play a key role
- Hybrid modeling has some strengths, but also weaknesses
- Compared to machine learning, hybrid models can physically more plausible and to a certain extent interpretable