



# NWP Models with ML Components: Challenges and Perspectives

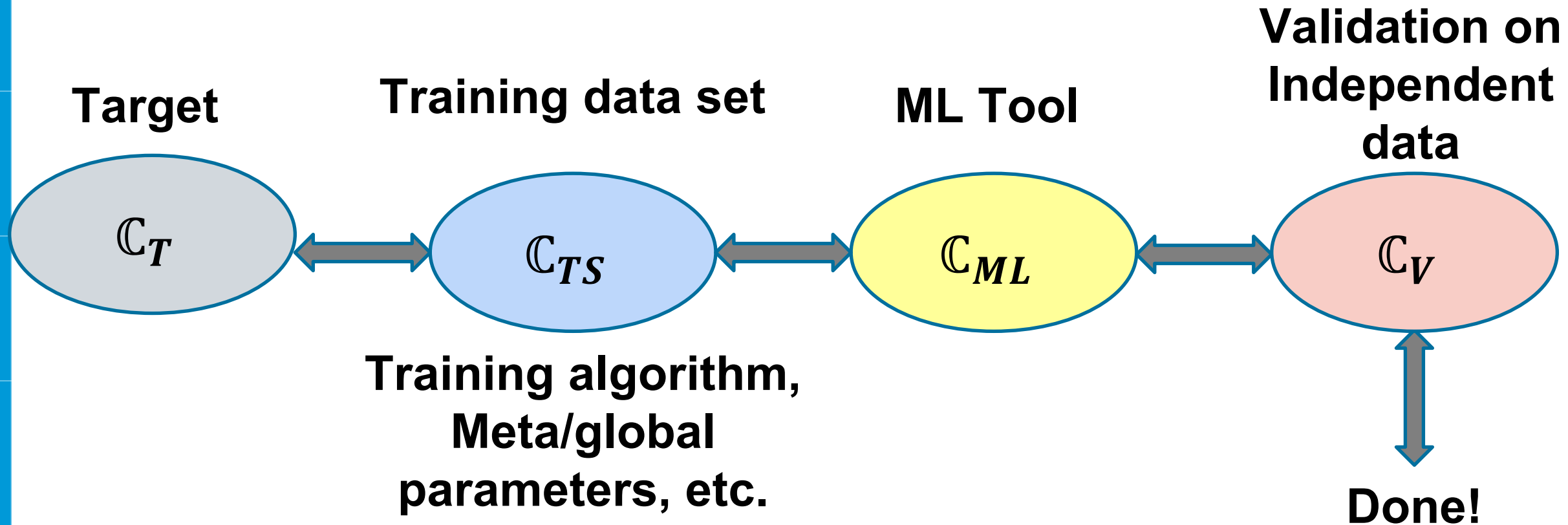
Vladimir Krasnopolsky  
NOAA/NWS/NCEP/EMC



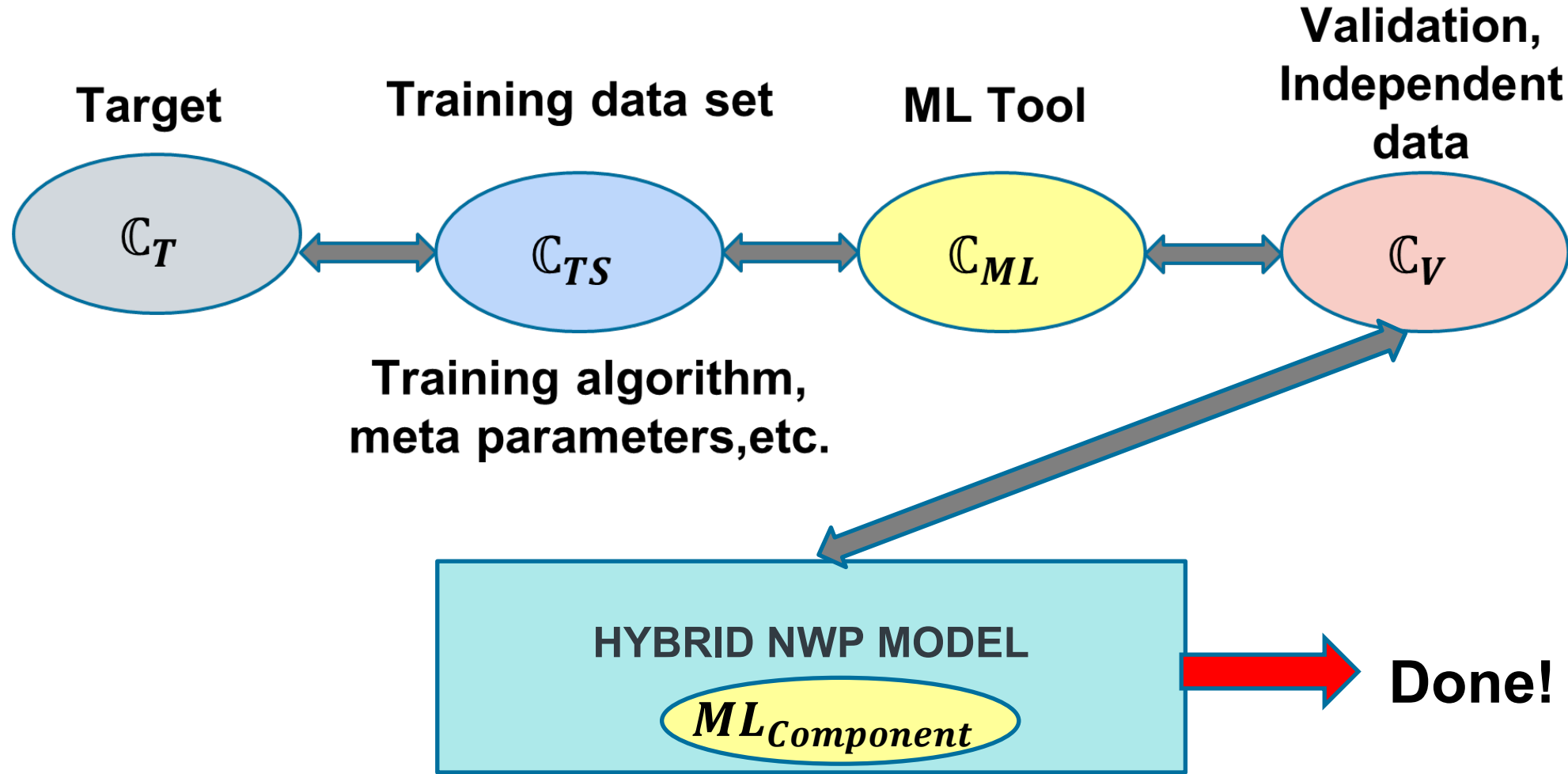
# Outline

- **ML components in Numerical Weather and Climate Models – Hybrid Models**
- **Challenges faced by ML applications in Hybrid NWP Models:**
  - Stability
  - Speedup
  - Improvements
- **Conclusions**

# Traditional ML problem: development of a ML application



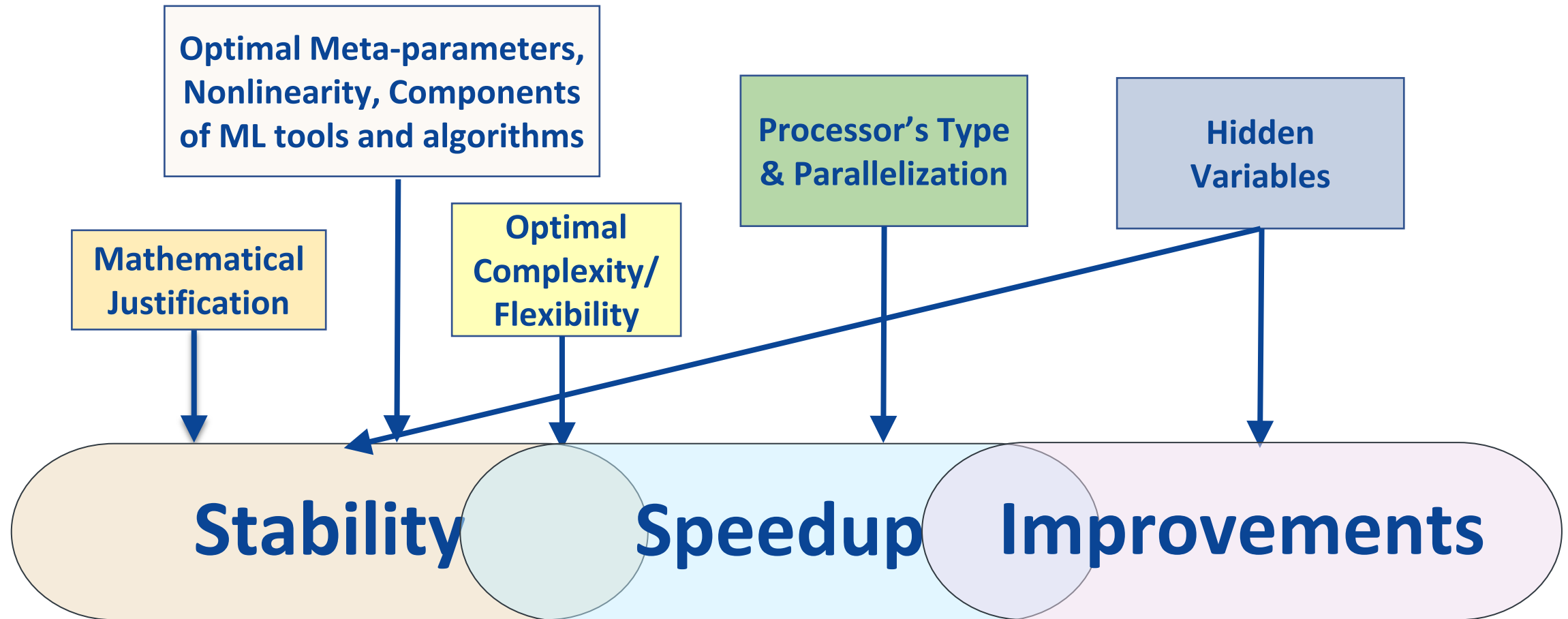
# ML application for NWP model



# Major Requirements to ML Components in Hybrid Numerical Models

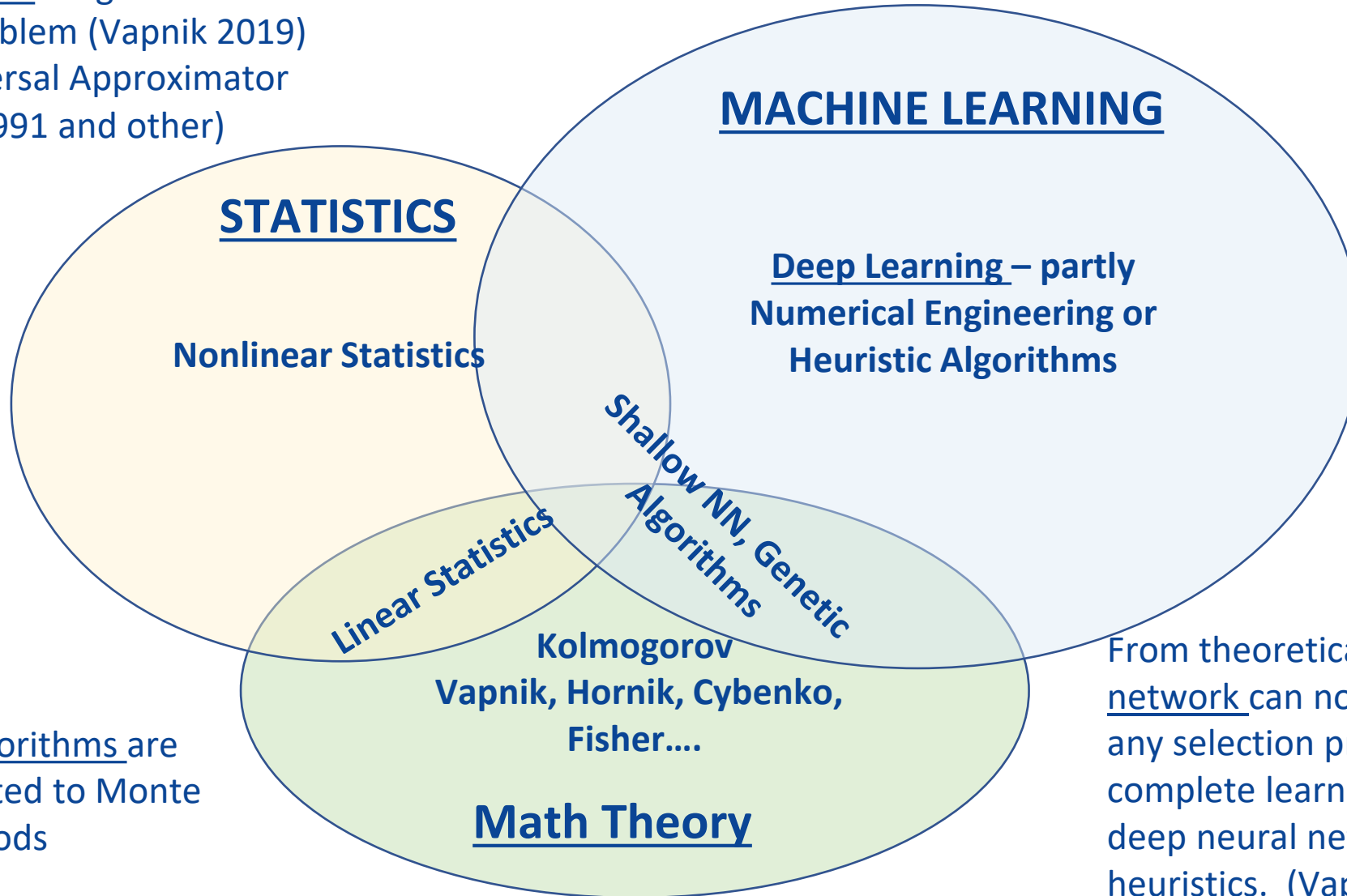
- **Stability**
  - Ability to integrate a hybrid model for as long as the original one with at least the same results
- **Speedup**
  - Hybrid model should be (significantly) faster than the original one
- **Improvements**
  - Hybrid model should better account for subgrid effects and produce better forecast

# Challenges for ML in Hybrid NWP Model



# Mathematical justification

Shallow NN is a generic solution of ML problem (Vapnik 2019) and Universal Approximator (Hornik 1991 and other)



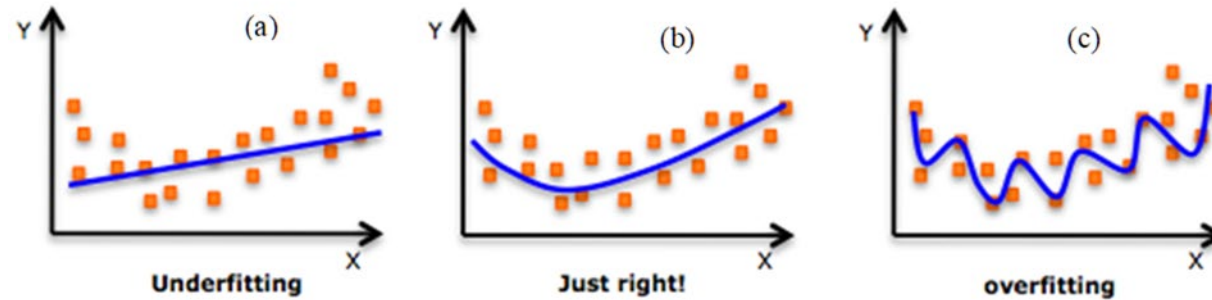
While deep learning is successful in a number of applications, it is not yet well understood theoretically. (Poggio et al. 2021)

Genetic Algorithms are closely related to Monte Carlo Methods

From theoretical point of view, deep network can not guarantee solution of any selection problem that constitute complete learning problem. Constructing deep neural network one uses different heuristics. (Vapnik 2019)

# Example: DNN (second descend) <https://arxiv.org/pdf/2205.15549.pdf>

- Large DNNs can be trained to fit training data exactly (achieving zero training error) and still achieve good generalization for test data.
- This contradicts conventional statistical wisdom that overfitting leads to poor generalization.



- Classical theories developed in ML and statistics cannot explain generalization performance of DL networks in the case of second descend.
- Is this a general property of DNN?
- Or probably it is an artifact of:
  - used data sets
  - various heuristics used for DNN architecture
  - used initialization procedures



# Optimal complexity/flexibility

## • Sallow vs deep NNs

- SNN complexity and architecture are uniquely described by a simple metric:

$\mathbb{C}_{SNN} = k \cdot (n + m + 1) + m$ , where  $n$  – number of inputs,  $m$  – number of outputs,  $k$  – number of hidden neurons in one hidden layer

- Similar metric for DNN complexity does not describe complexity AND architecture uniquely:

$\mathbb{C}_{DNN} = \sum_{i=0}^K k_{i+1}(k_i + 1)$ , where  $k_i$  is the number of neurons in the layer  $i$  ( $i = 0$  and  $i = K$  correspond to the input and output layers, respectively).

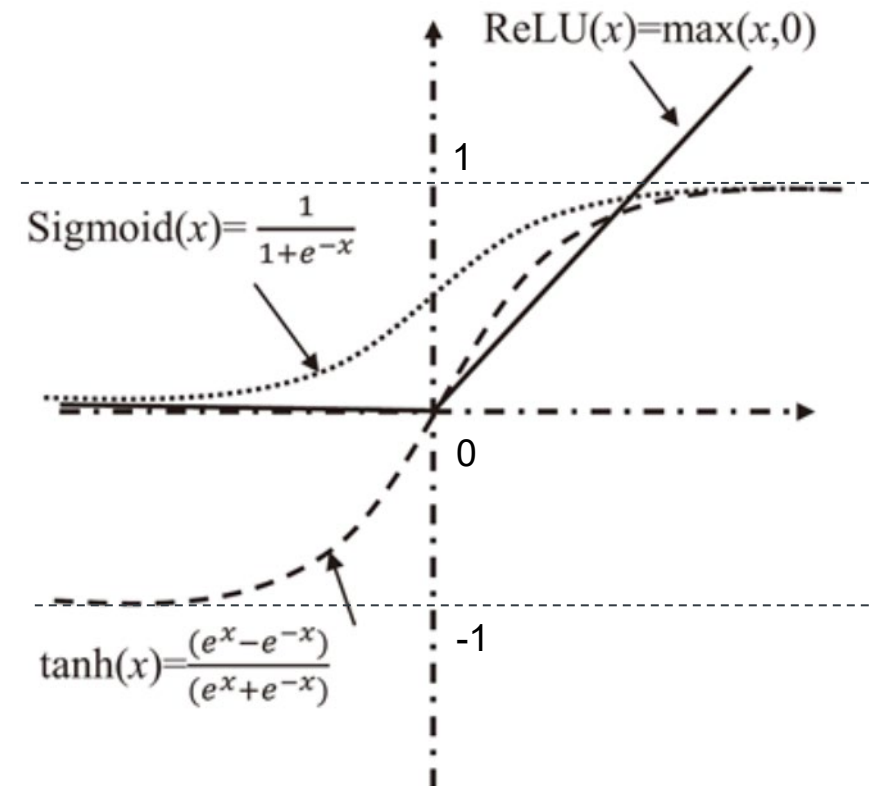
- Both  $\mathbb{C}_{SNN}$  and  $\mathbb{C}_{DNN}$  show the number of parameters/weights of the corresponding NN
- Higher complexity of DNN is a plus, it makes it a versatile tool; however, it makes it more difficult to control:
  - Optimizing SNN architecture – single-parameter problem
  - Optimizing DNN architecture – multi-parameter problem
- Complexity/flexibility of NN is closely related to generalization ability of the NN and should correspond to complexity of the training set. => **May challenge stability of hybrid NWP models!**

# Nonlinearity

- SNN:  $Y = B^1 + A^1 \cdot \phi(B^0 + A^0 \cdot X)$
- Nonlinearity of SNN increases “linearly” with adding new hidden neurons
- DNN:  $Y = X^{n+1} = B^n + A^n \cdot \phi \left( B^{n-1} + A^{n-1} \cdot \phi \left( B^{n-2} + A^{n-2} \cdot \phi \left( B^{n-3} + \dots \phi(B^0 + A^0 \cdot X) \right) \right) \right)$
- Nonlinearity of SNN increases “non-linearly” with adding new hidden neurons in new hidden layers
- Higher nonlinearity of DNN is a plus, it makes it a versatile tool; however, it makes it more difficult to control
- Nonlinearity of NN is closely related to generalization ability of the NN (nonlinear extrapolation is an ill-posed problem) and should correspond to nonlinearity of the training set and the target mapping => **May challenge stability of hybrid NWP models!**

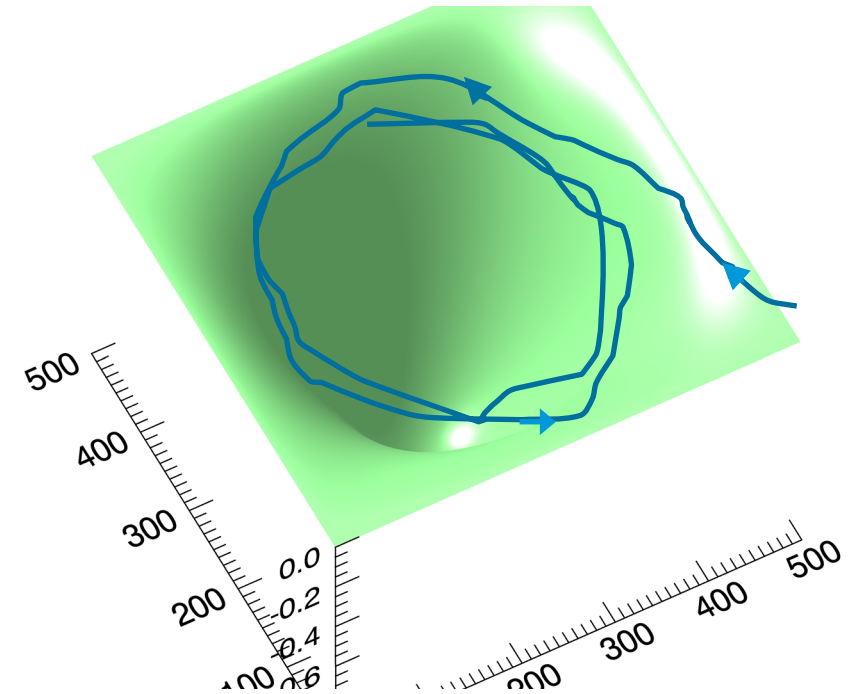
# Choice of activation function

- Most of proven math theorems assumes activation function (AF) to be monotonically increasing, continuous, continuously differentiable, and limited, e. g. like  $\tanh(x)$
- In applications many different AF are used
- If an AF proved to be successful in a ML application (e.g., classification), it does not mean that it is good for using in hybrid NWP models. Tendencies are often used in hybrid NWP models => **May challenge stability of hybrid NWP models!**



# Learning Rates and early stopping

- Ott et al. (2020) demonstrated that there is a **negative correlation between the offline MSE and online stability** when using tendencies as outputs in hybrid NWP models.
- Since tendencies are often used in the real-world simulations, an ML-based parameterization that can support long-term integration should have a high accuracy regarding training and validation.
  1. Adjustable learning rate should be used in iterative training algorithms
  2. Early stopping should be used carefully.
  3. Because simulated data are used for training (practically no noise) early stopping may not be necessary.



# Challenges with Implementation in NWP models: Evolution of PUs

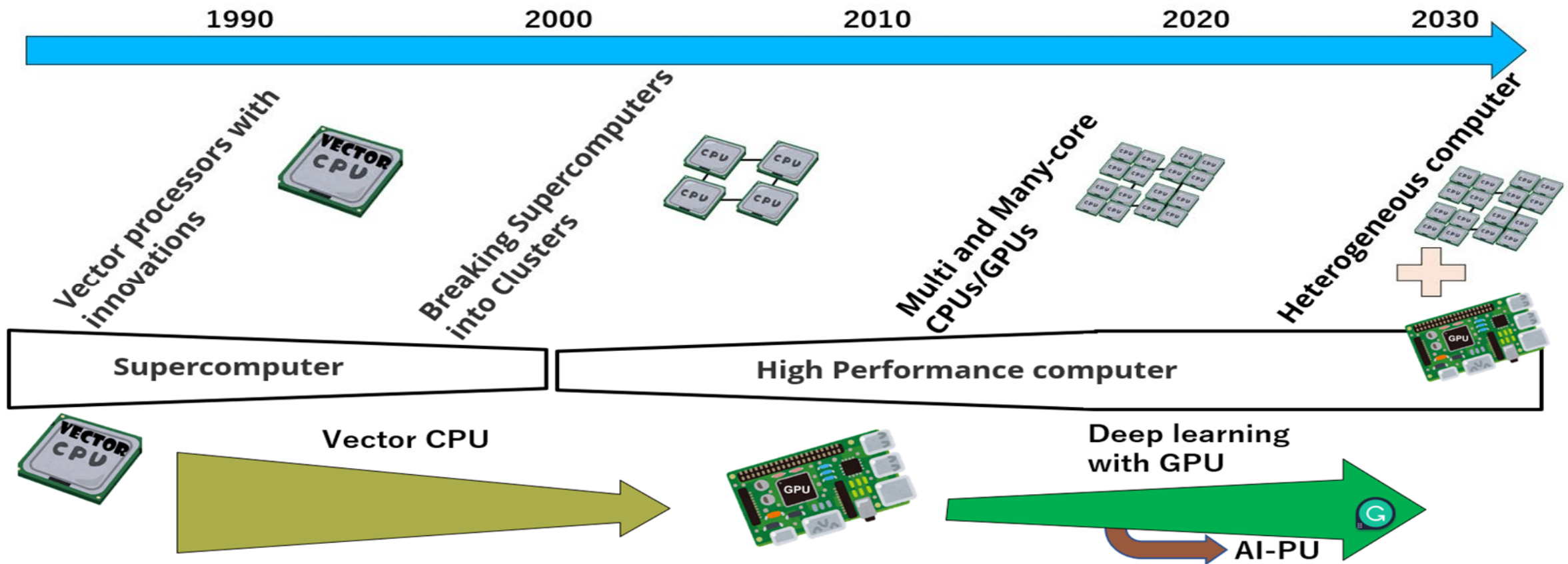


Figure from: Tosiya Nakaegawa, 2022

- AI-PUs processing units for deep learning or artificial intelligence (AI accelerators) have been developed.
- An AI-PU has similar features to a GPU but has special matrix processing chips capable of a high volume of low-precision computations (single or even half precision).

# Implementation in NWP models

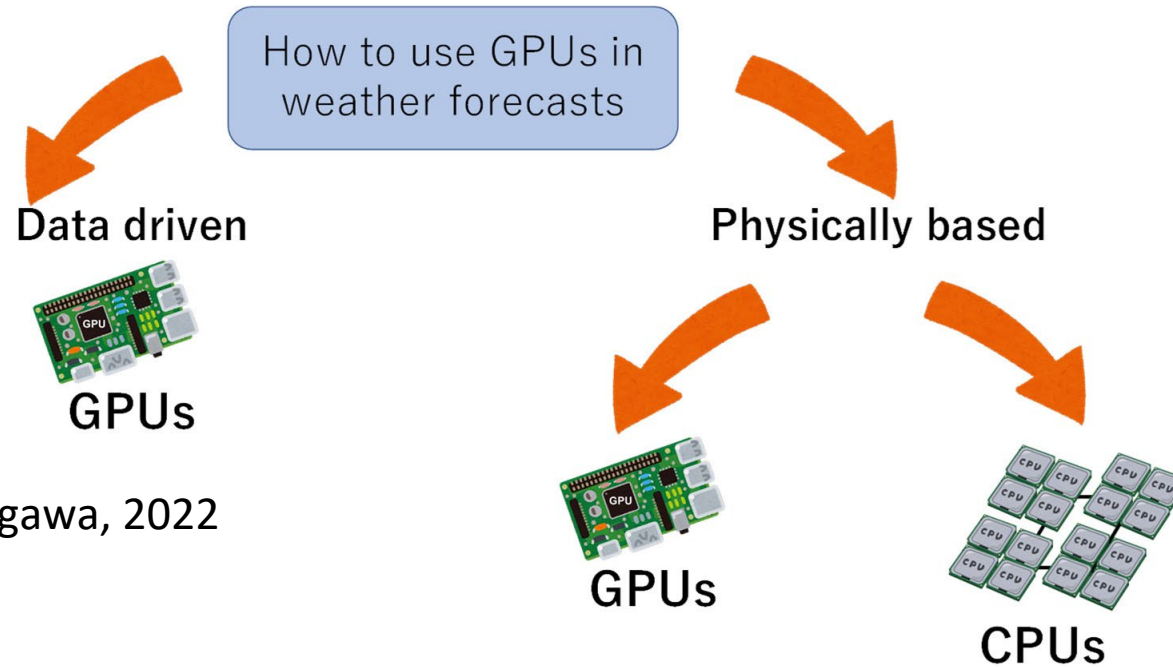
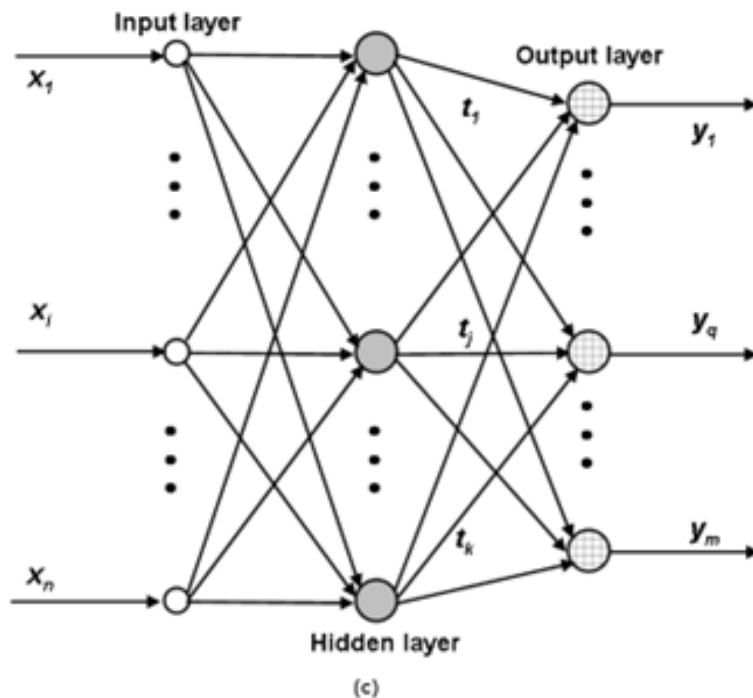


Figure from: Tosiya Nakaegawa, 2022

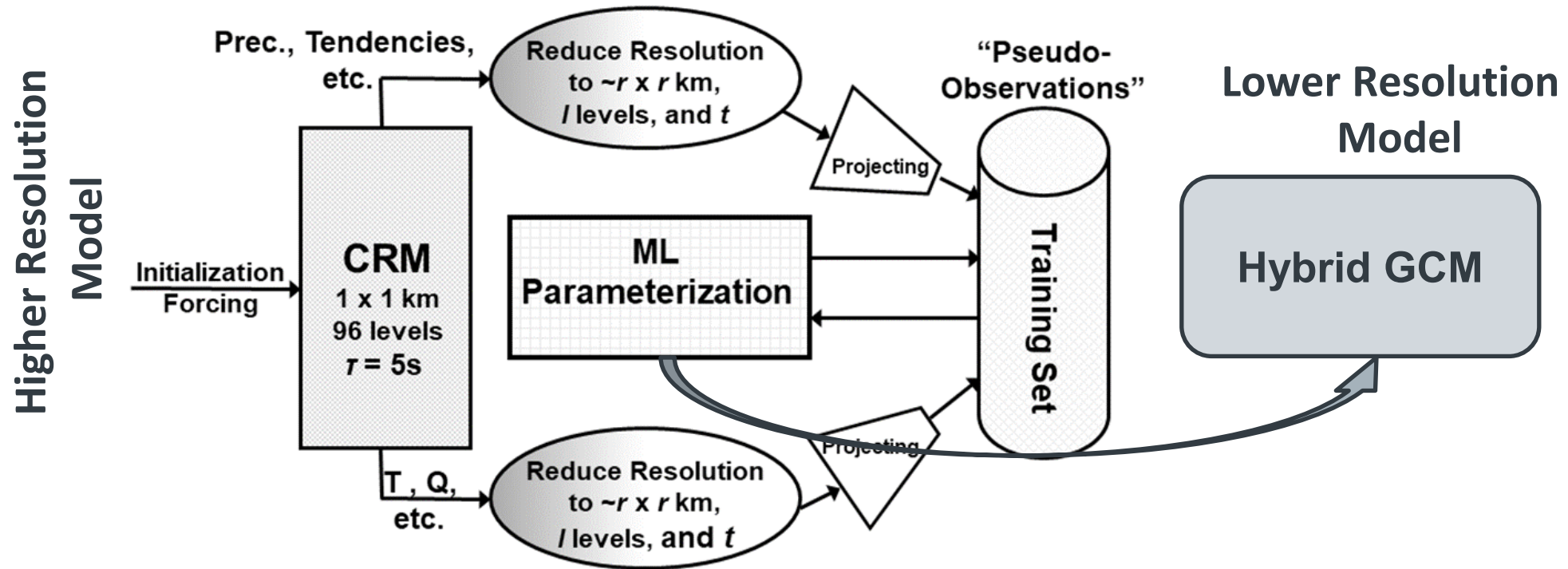
- Many deep learning models have open-source frameworks (several of them are freely available, e.g., TensorFlow and Keras) and use GPUs.
- The original codes of the NWP model are written for CPU-based computers (at NCEP, ECMWF, UKMO, DWD, etc.) and need to be rewritten or converted to run on GPUs.

# Unused Reserve for Speedup

- Neurons in hidden layers are independent and could be calculated simultaneously (in parallel)
- However, the NWP model has been already parallelized and CPUs are not vector CPUs



# Hidden variables: Developing new ML parameterizations



- Projecting – omitting/averaging out some of Higher-Resolution Model (CRM) variables that do not have counterparts in Lower-Resolution Model (GCM) – they become hidden variables for GCM
- Incorrect reducing resolution and projecting may challenge stability and expected improvements of forecasts of hybrid models.



# Conclusions

- ML made significant progress in developing applications for NWP and Climate models
- Hybrid models demonstrate promising performance
- There are several major requirements for hybrid models to become operationally useful:
  - High stability
  - Sufficient speedup
  - Significant improvements in forecast upon original model
- To fulfill these requirements, more attention should be paid to ML roots (math justification and theoretical understanding of ML tools, more attention to components of ML tools, options of training algorithms, and HPC architecture and components)



# Questions?










# Backup slides



# Some Suggestions

- 
- 
- 
- 
- 
- Proper use of machine learning methods requires some level of interdisciplinary cooperation between scientists
    - Close collaboration of ML expert, modelers, high level programmers, software engineers, etc.
  - Better knowledge of tools and methods. Fortunately, many institutions are now organizing workshops and seminars that are freely available online to help to tackle this problem.
  - it is important first to properly understand the processes and relationships between meteorological and environment variables in analyzed problems, and to correctly implement an adequate machine learning method and **not to use it as a black box**
  - RF or SNN require less knowledge in the field of machine learning and are more suitable for beginners while DL or CNN needs more experience to be used properly.

# DNN computational limits <https://www.csail.mit.edu/news/computational-limits-deep-learning>

- deep learning is reaching its computational limits
  - data intensive
  - computationally intensive
- offered solutions:
  - Increasing computing power: Hardware accelerators (GPU/TPU)
  - deep learning being forced towards less computationally-intensive methods of improvement,
    - Reducing computational complexity: Network Compression and Acceleration.
    - Finding high-performing small deep learning architectures: Neural Architecture Search and Meta Learning.



# ML components in Numerical Weather and Climate



- **Data preparation**

- Fast and accurate retrieval algorithms for geophysical parameters
- Quality control algorithms
- Downscaling



- **Data assimilation**

- Fast forward models for direct assimilation of radiances
- Fast observation operators to assimilate bio, chemistry, etc. observations
- Fast adjoints for 4-D Var
- ML data assimilation system



- **Model physics**

- Fast emulation of existing slow parameterizations (radiation, microphysics, etc.)
- Fast model physics
- New improved ML parameterizations



- **Post-processing model outputs**

- Bias corrections
- Nonlinear MOS
- Nonlinear ensemble averaging





# ML challenges in Weather and Climate Fields



- ML challenges imposed by properties of meteorological data and the complexity of the weather forecasting problem (Schultz et al. 2021)

