CS11-711 Advanced NLP

Combining Multiple Models

Graham Neubig

Carnegie Mellon University

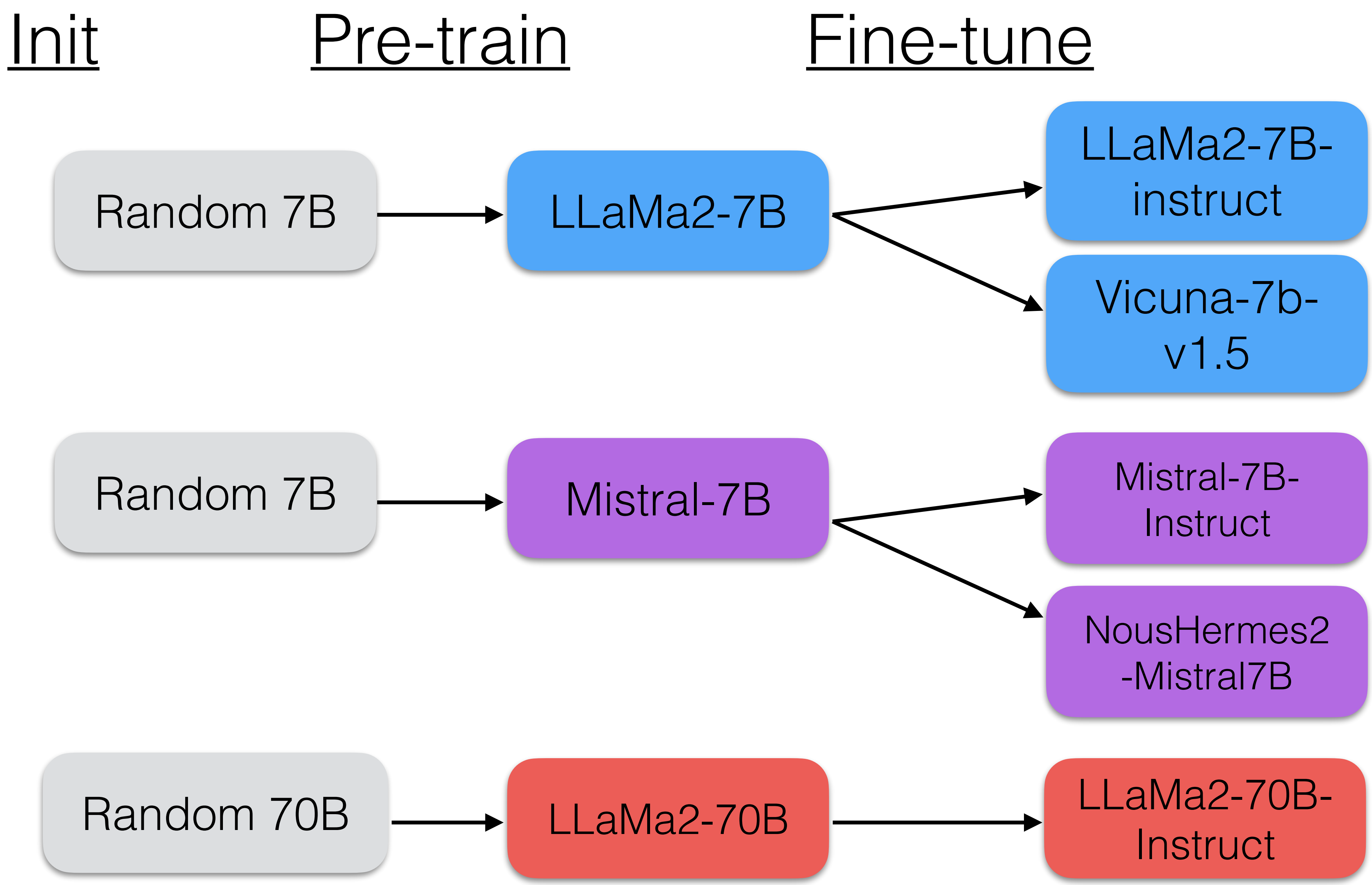
Language Technologies Institute

Site

https://phontron.com/class/anlp2024/

Many Models Exist!

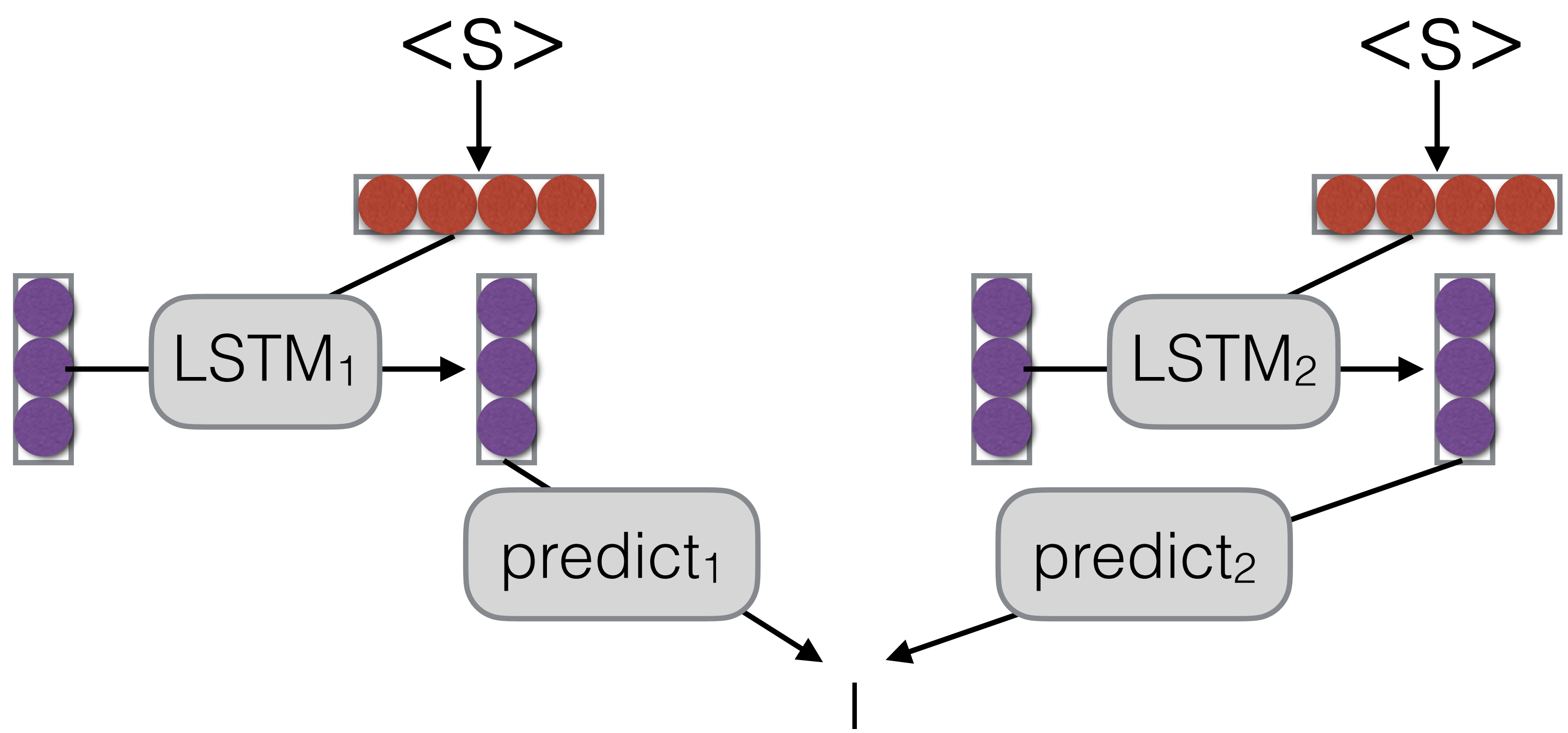
- Different architectures or training result in different P(Y|X)



Model Ensembling

Ensembling

- Combine predictions from multiple models



- Why?
  - Multiple models make somewhat uncorrelated errors
  - Models tend to be more uncertain when they are about to make errors
  - Smooths over idiosyncrasies of the model

Linear Interpolation

- Take a weighted average of the M model probabilities

$$P(y_j \mid X, y_1, \ldots, y_{j-1}) = \sum_{m=1}^{M} \underbrace{P_m(y_j \mid X, y_1, \ldots, y_{j-1})}_{\text{Probability according to model } m} \underbrace{P(m \mid X, y_1, \ldots, y_{j-1})}_{\text{Probability of model } m}$$

- Second term often set to a constant, independent of context

Log-linear Interpolation

- Weighted combination of log probabilities, normalize

$$P(y_j \mid X, y_1, \ldots, y_{j-1}) =$$

$$\underbrace{\mathrm{softmax}}_{\text{Normalize}} \left(\sum_{m=1}^{M} \underbrace{\lambda_m(X, y_1, \ldots, y_{j-1})}_{\text{Interpolation coefficient for model } m} \underbrace{\log P_m(y_j \mid X, y_1, \ldots, y_{j-1})}_{\text{Log probability of model } m} \right)$$

- Interpolation coefficient often set to a constant

Linear or Log Linear?

- Think of it in logic!
- **Linear:** “Logical OR”
    - the interpolated model likes any choice that a model gives a high probability
    - use models with models that capture different traits
    - necessary when any model can assign zero probability
- **Log Linear:** “Logical AND”
    - interpolated model only likes choices where all models agree
    - use when you want to restrict possible answers

Stacking

- What if we have two very different models where prediction of outputs is done in very different ways?
- e.g. a phrase-based translation model and a neural MT model (Niehues et al. 2017)
- Stacking uses the **output of one system in calculating features for another** system

Efficient Methods for Using Multiple Models

Problem with Ensembling: Cost

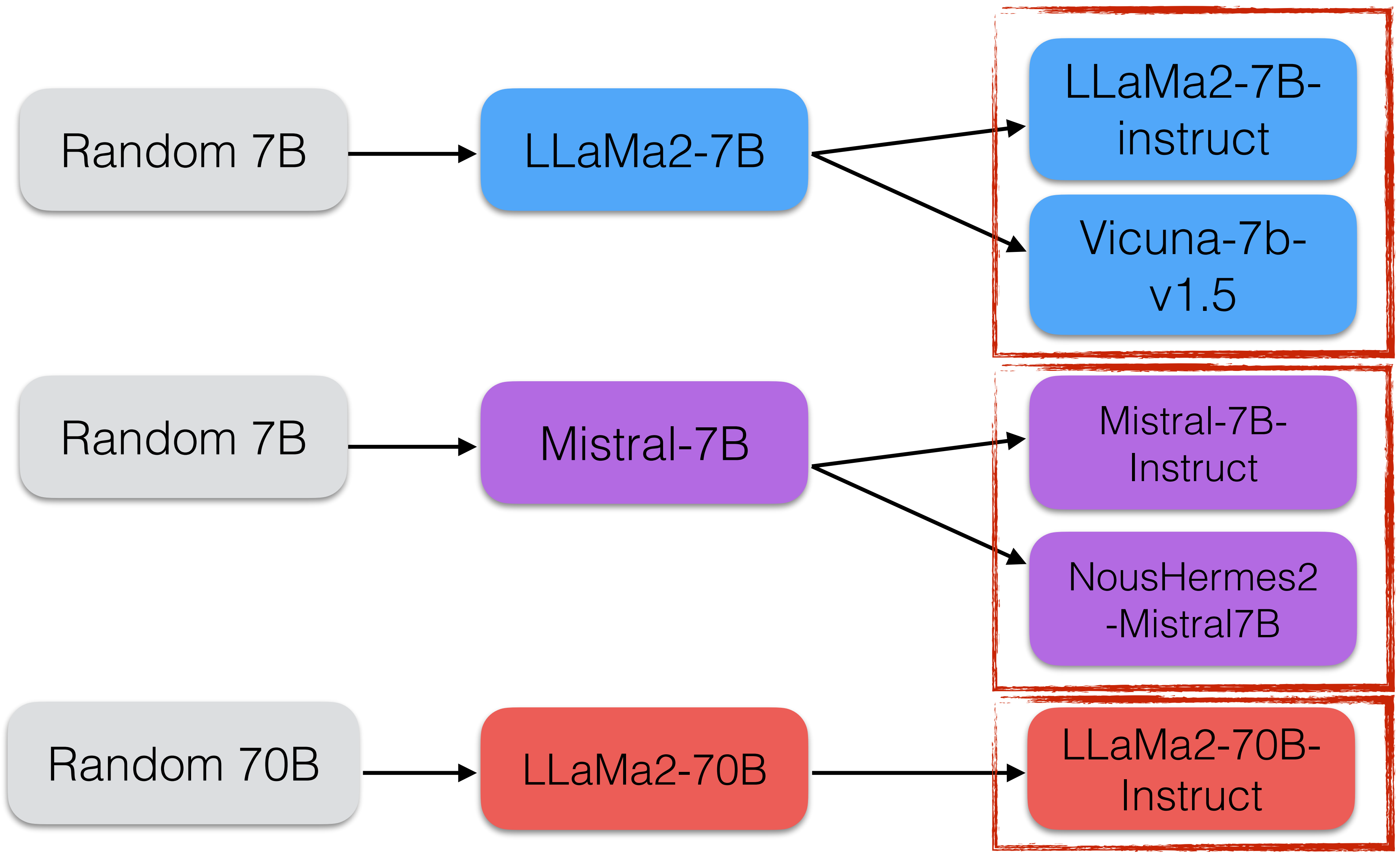
- Simple ensembling is expensive: it requires running two models in parallel
- Is there any way we can more easily combine together two models

Parameter Averaging
(e.g. Utans 1996)

- Parameter averaging is a cheap way to get some good effects of ensembling
- Basically, average the parameters of multiple models
  - **Checkpoint averaging:** write out models several times near the end of training, and take the average of parameters
  - **Fine-tuned model merging:** fine tune in several different ways, then average

Can only Average Related Models

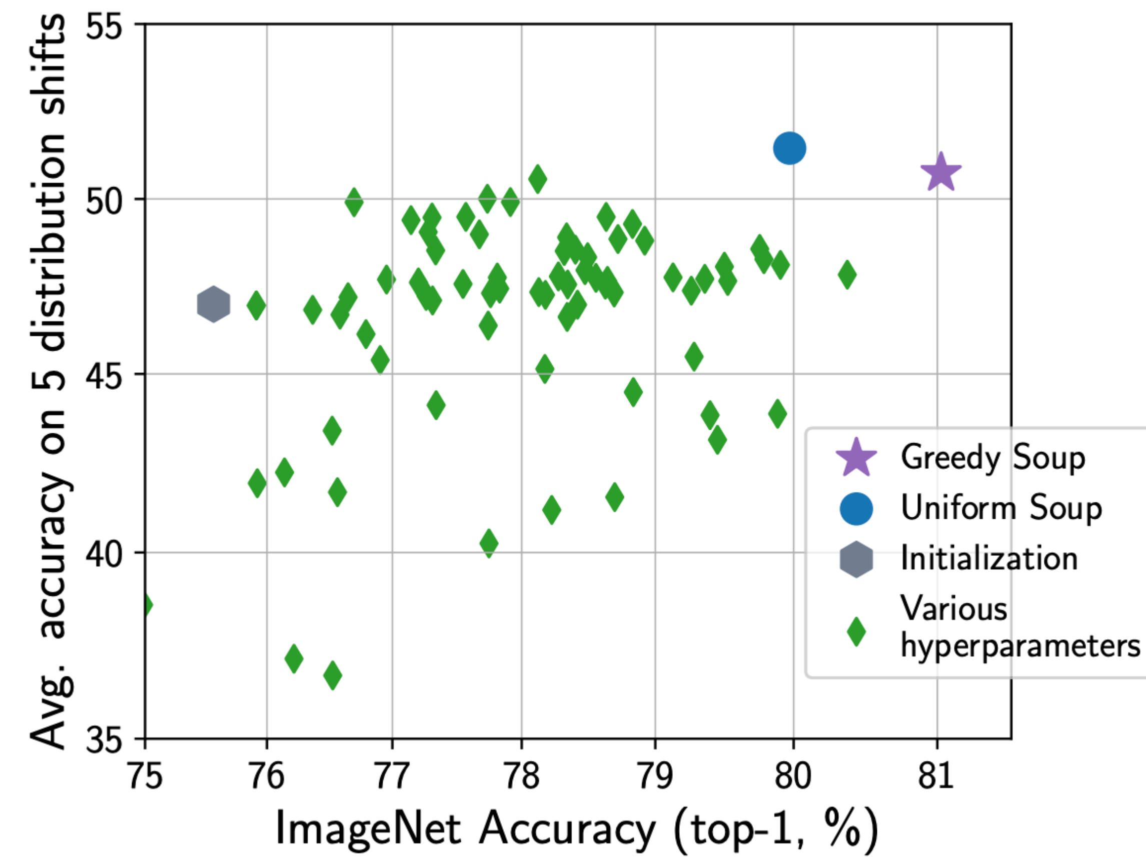
- Models must originate from the same pre-trained checkpoint



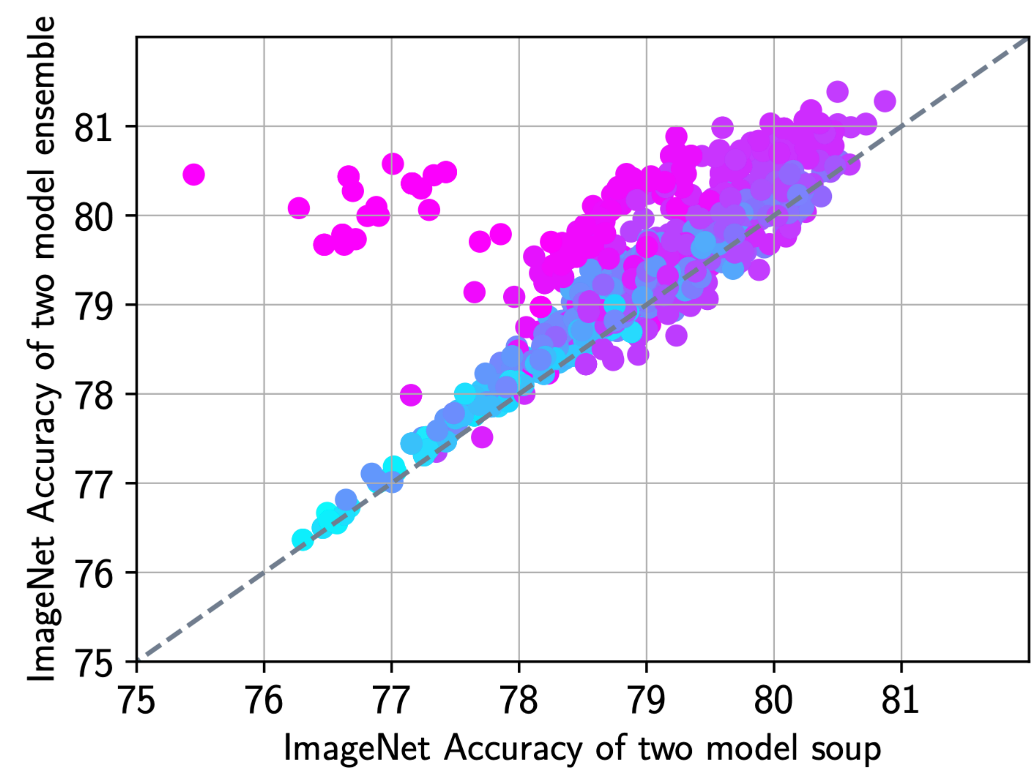
- **Quiz:** why is this?

Model Soups (Wortsman et al. 2022)

- Examines two strategies:
  - Uniform averaging
  - Greedy averaging (add one, and keep if it improves)

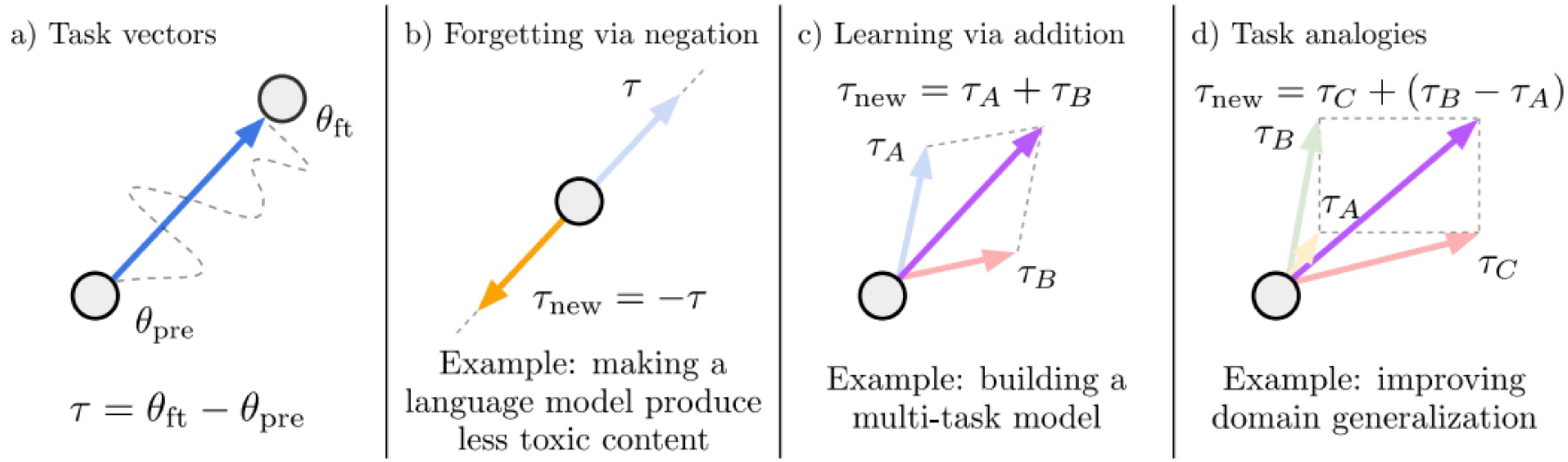


- Demonstrates that averaging is correlated with resembling

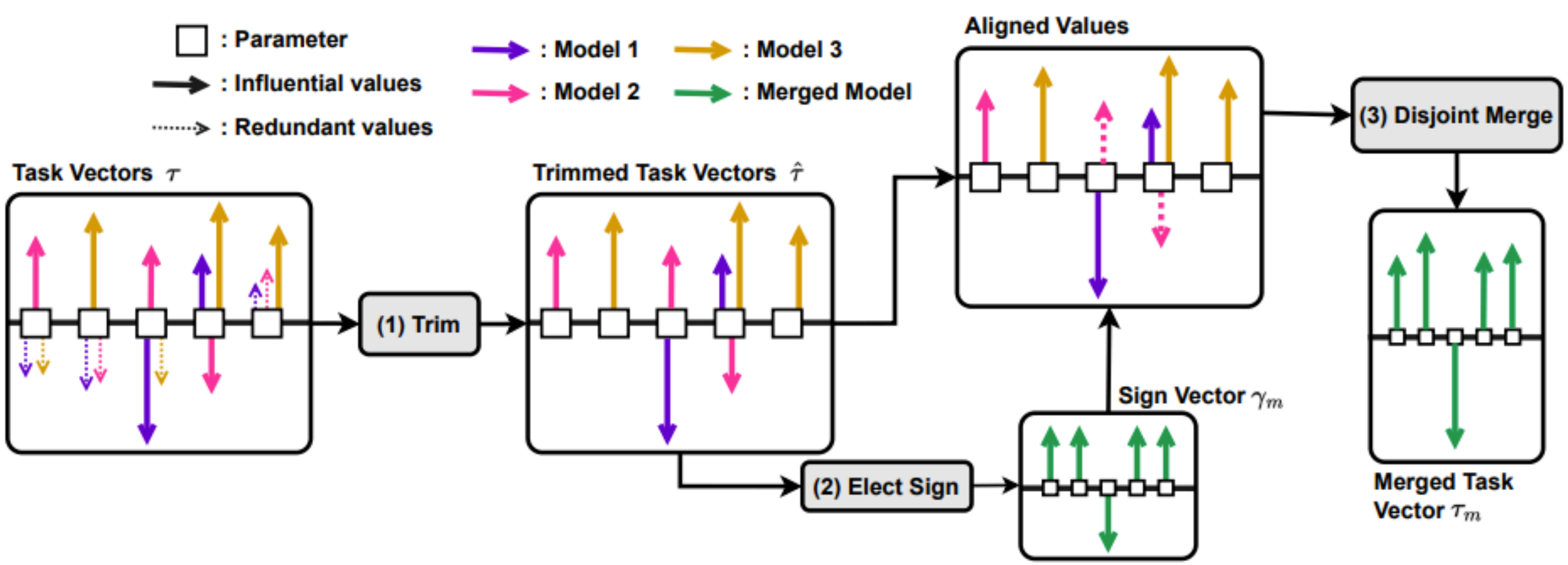


Task Vectors

- Quantify changes from a base models through “task vectors” (Ilharco et al. 2022)



- TIES: resolves conflicts through max and sign (Yadav et al. 2023)



Software: mergekit

- https://github.com/arcee-ai/mergekit
- Implements a number of different methods for model merging

Method	`merge_method` value	Multi-Model	Uses base model
Linear (Model Soups)	`linear`	✅	❌
SLERP	`slerp`	❌	✅
Task Arithmetic	`task_arithmetic`	✅	✅
TIES	`ties`	✅	✅
DARE TIES	`dare_ties`	✅	✅
DARE Task Arithmetic	`dare_linear`	✅	✅
Passthrough	`passthrough`	❌	❌

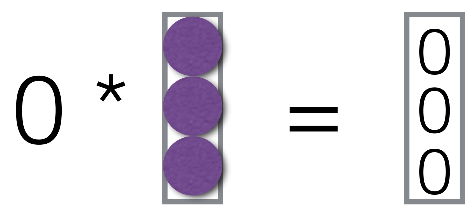
Ensemble Distillation (e.g. Kim et al. 2016)

- **Problem:** parameter averaging only works for models within the same run
- Knowledge distillation trains a model to **copy the ensemble**
  - Specifically, it tries to match the distribution over predicted words
  - Why? We want the model to make the same mistakes as an ensemble
- Shown to increase accuracy notably

Sparse Mixture of Experts Models

Sparse Computation

- What happens when a scalar-tensor multiplication is zero?



  - Result is guaranteed to be zero! No computation needed
- This can happen in many parts of a model:
  - Single rows in a matrix multiply
  - Larger tensors
  - Whole models in an ensemble

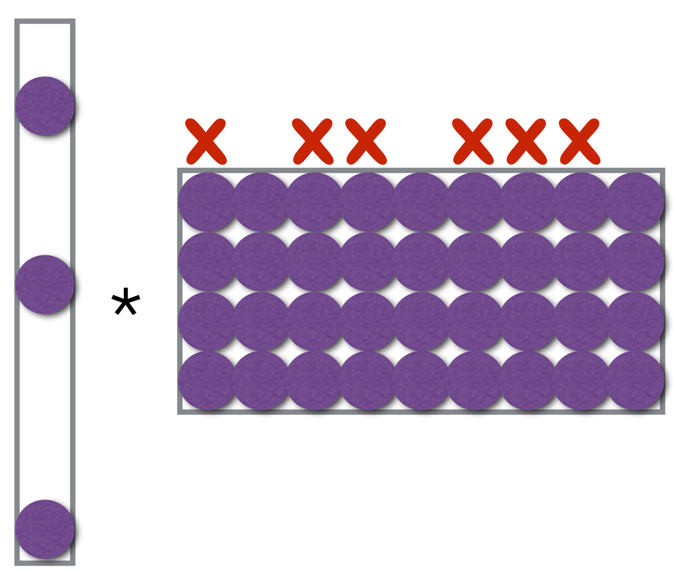
→ optimized by GPU

→ sparse MoE models

→ just don’t use that model

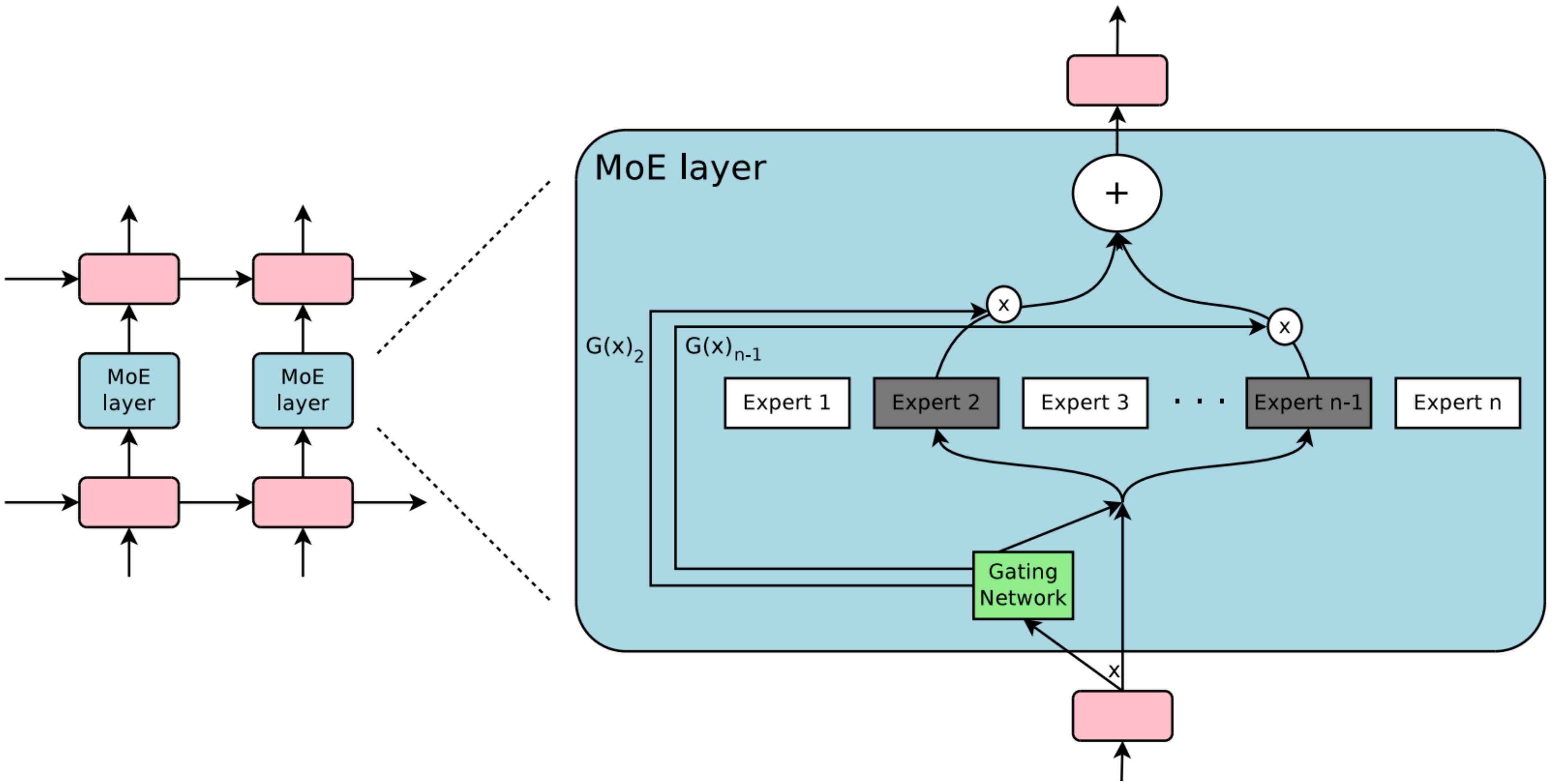
GPU-level Sparsity

- NVIDIA GPUs support various types of sparsity through the cuSPARSE library and tensor cores
- Examples, vector-matrix multiply with sparse vector (e.g. one that comes from ReLU activation)



Sparsely Gated Mixture of Experts Layer (Shazeer+ 2017)

- Select a subset of FFNs to actually execute



$$g(x) = \mathrm{softmax}(\mathrm{keep\_top\_k}(f_{\mathrm{gating}}(x), k))$$

$$\mathrm{keep\_top\_k}(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

Questions?