Neural Networks

Transfer Learning and Catastrophic Interference

CSCI 4850/5850

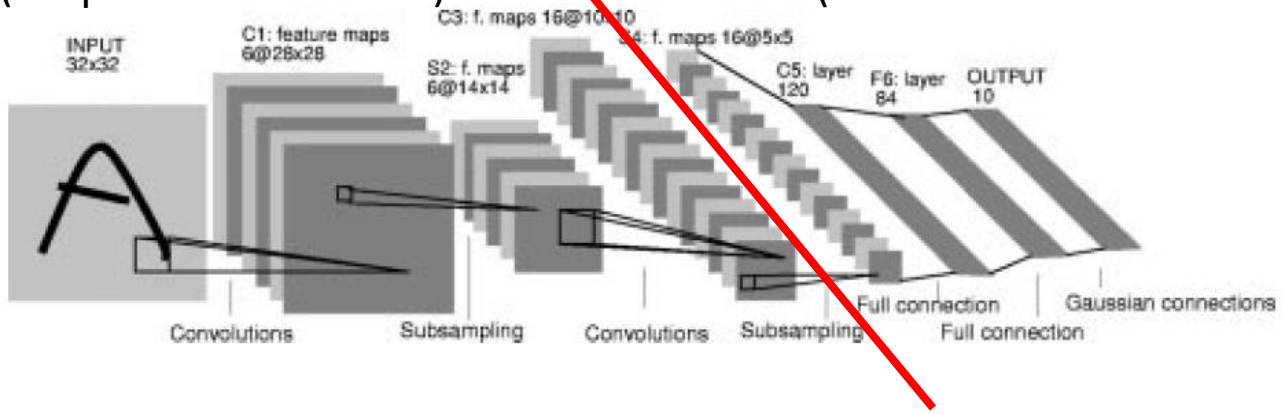
Transfer Learning

- For many tasks, useful features might potentially transfer to another task
  - Recognizing digits -> Recognizing letters
  - Recognizing trees -> Recognizing bushes
  - Recognizing cats -> Recognizing animals
  - Recognizing animals -> Recognizing cats (probably easier)
- Might reduce training time (or maybe even generalization performance) if these features didn't need to be relearned from scratch
- Removal of the early layers and transferring the *pre-trained* model into another network **may be advantageous**
- The **torchvision.models** package contains several pre-trained models…

Task-general features
(keep for other tasks…)

Slice

Task-specific features and mappings
(not so useful in other tasks)



Challenges when Learning from Data

Large samples are needed

plenty of cats and non-cats

plenty of different digits/letters

plenty of different sequences

plenty of events

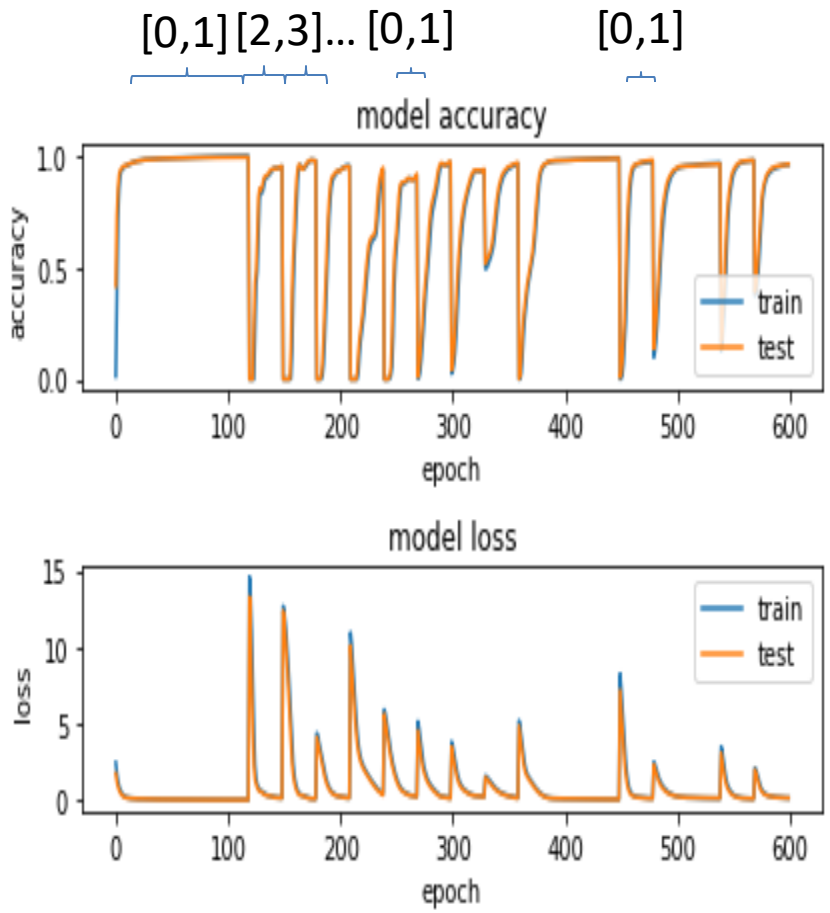
Even with large samples, training order matters

Temporal/Reinforcement learning: the *replay memory is* needed to ensure that the network didn't stay focused on **short-term spatial and temporal correlations** in the data

These can exist in even small data sets…

MNIST Example…

- Let's imagine we want to work on learning to classify digits a couple at a time
  - One might think this would be easier, no?
  - Breaking the learning phase into [0,1] then [2,3] then [4,5], etc.
- We can train on the first set…
  - Reach 98% criterion
  - Move to the next set, OK
  - Returning to the [0,1] set we saw earlier…
  - Have to relearn those digits!
- No matter how long we continue this pattern, we will never learn the digits set in this way



No regularization can fix this generalization issue!
Although, it can *sometimes* help…

Catatrophic Interference/Forgetting

Remember that weights are shared in a neural network

Each batch update results in updating the weights in the direction of the negative error gradient

This batch update is specific to the batch

Larger batch sizes can help this

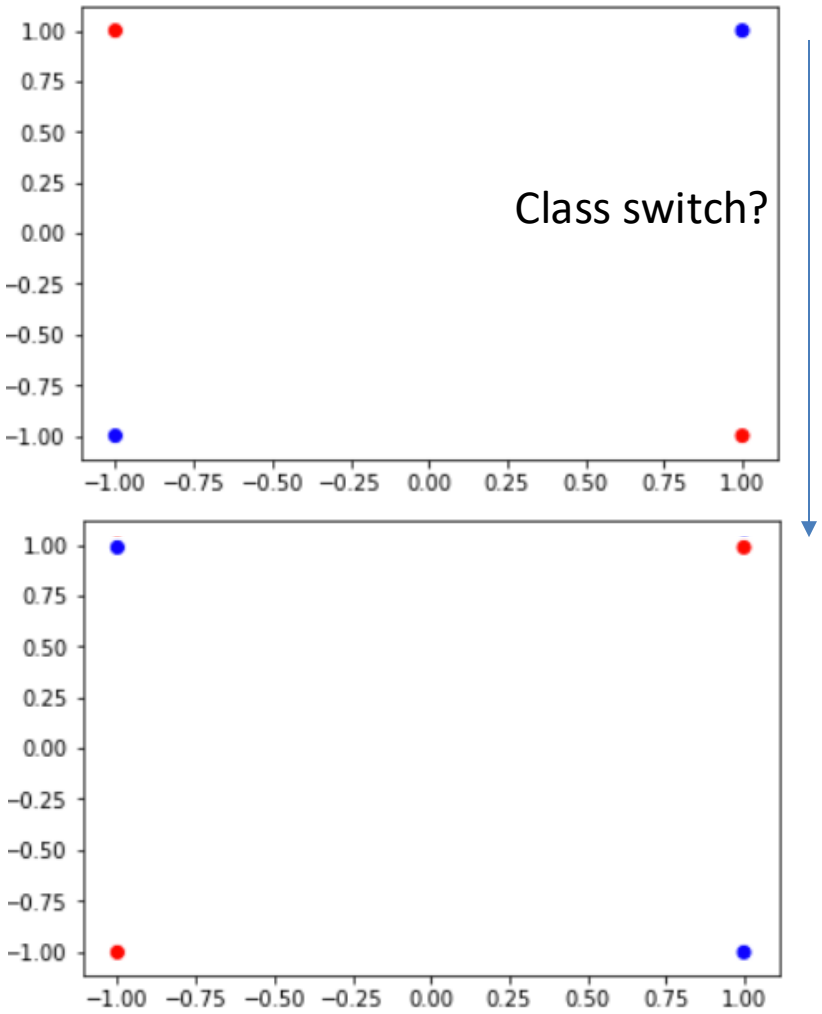
Batch size = entire data set = true gradient

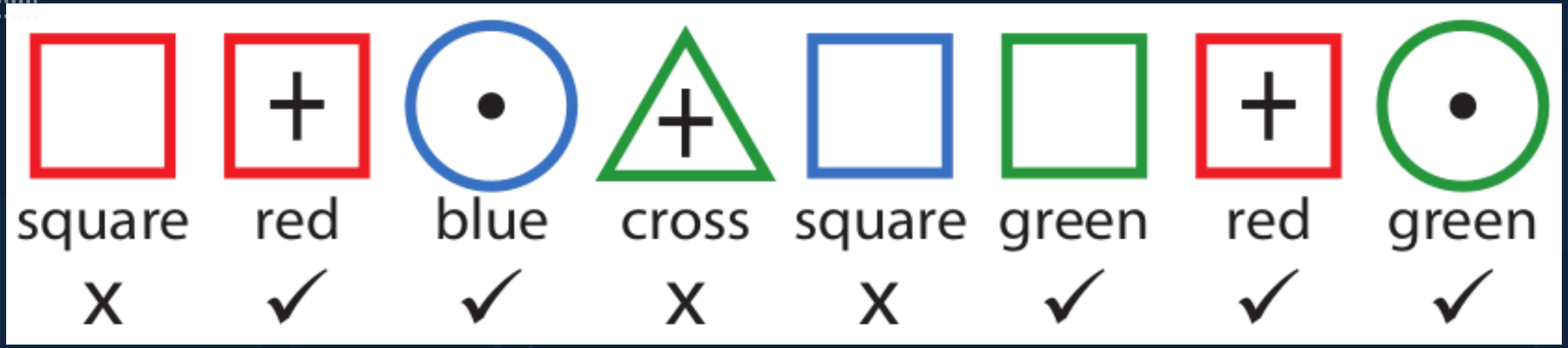
However, even small batch sizes are reasonable so long as you take an *independent sample* of your data points…

Are there problems where this still can't work?

Really Hard: XOR vs. XNOR!

- Some data may need to be reclassified according to a ***different function***
- Maybe we want the network to detect cats (cat/nocat) but then detect dogs (dog/nodog)
- Are there other examples?





Wisconsin Card Sort Task (WCST)

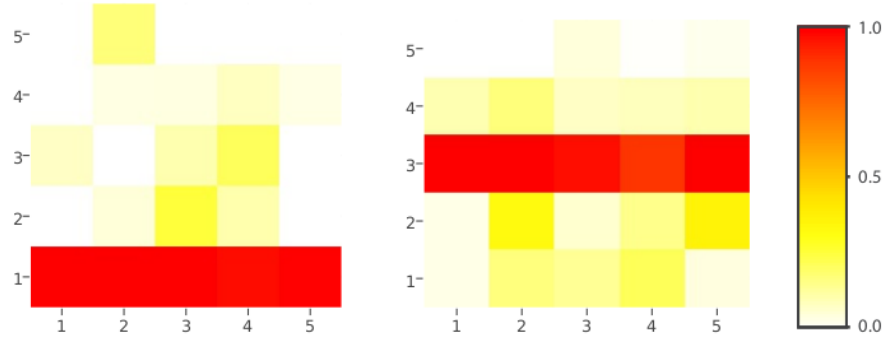
- Common test used in the cognitive sciences for understanding how people reorient with changing task demands…
  - Once task is learned, the rules may switch *unexpectedly*
  - Changing functions? How can we get around this?

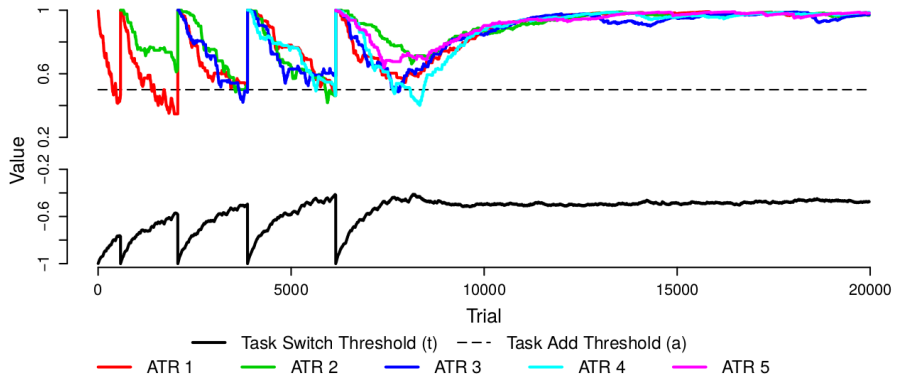
Context

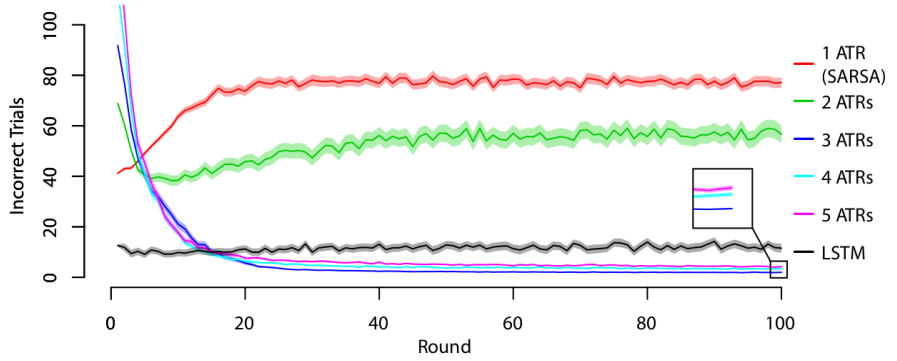
- For some problems, when we can interleave examples, we add a context unit…
  - XOR: [-1, -1, 1]; XNOR: [-1, -1, -1]
  - Third entry indicates *task information*
  - If interleaved training is used, this will work!
  - However, you are essentially just encoding the task a feature (provded as input)…
- However, the WCST is *unpredictable* (nondeterministic)

N-Task Learning (Jovanovich and Phillips, 2017)

- Network stores an *internal representation of the context* which is **convolved** with the current inputs
- Network *learns* **how many tasks** and when to ***optimally* switch** between them
- Outperforms best competitor to-date LSTM

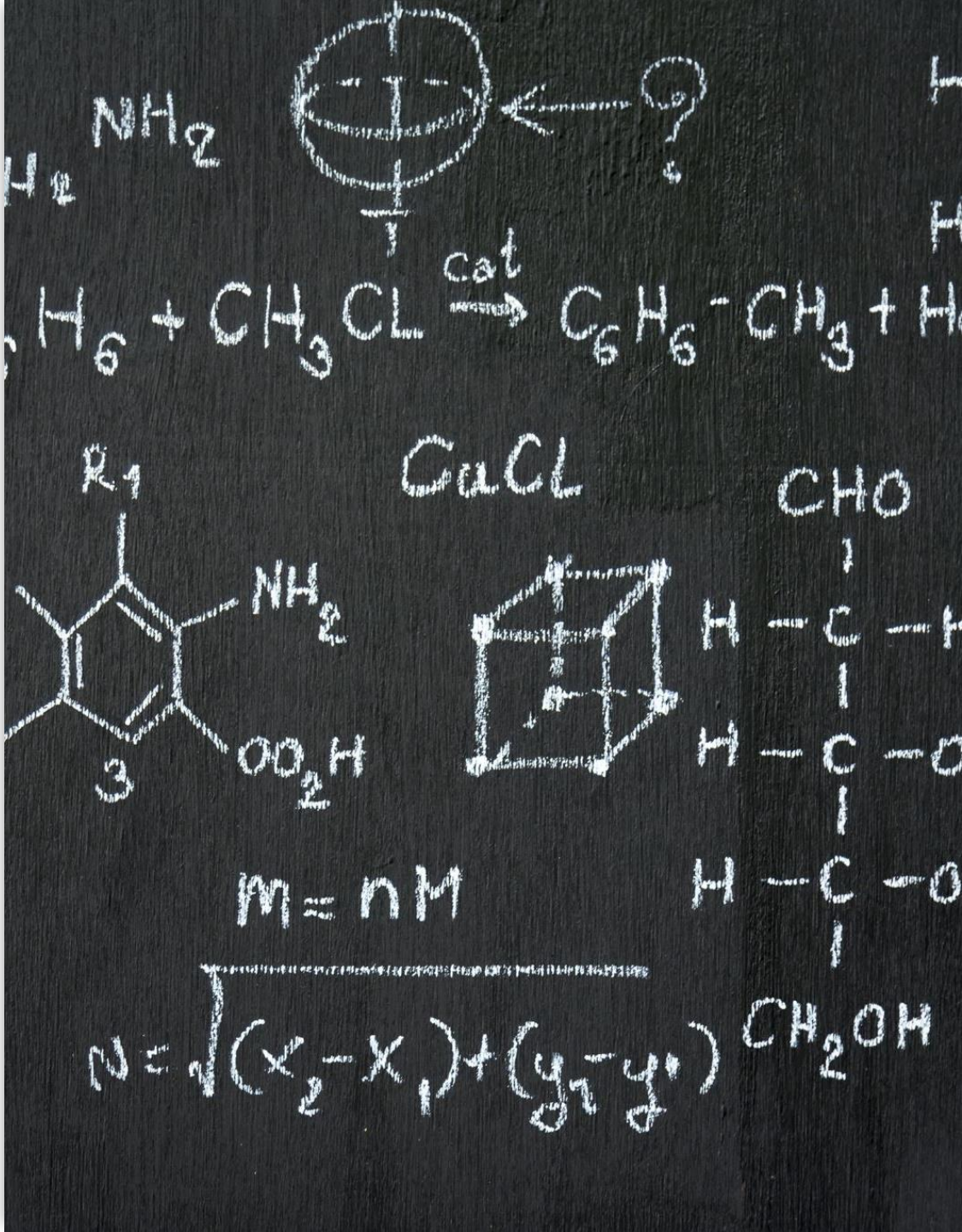






How does it do it?

- The convolution operation makes the input feature vectors (approximately) orthogonal to one another for each task
  - Orthogonal patterns generally lead to orthogonal gradients
  - Weight updates tend to be non-competitive…
  - Even hebbian or single-layer delta-rule learning can solve the problem in this space
- Each context must be stored independently
  - Next iteration will involve *learning* these convolution representations as well



Alternative Strategies

- Weight update slowing (Kirkpatrick et al., 2017)
  - Important weights for tasks are identified and learning rate selectively lowered…
  - Need to test head-to-head with n-Task soon… ☺
- Complementary learning systems approach (McClelland et al., 1995)
  - General framework requires two networks
    - Short-term
    - Long-term
  - Long term trained by generative process with short term examples folded in (replay-like)