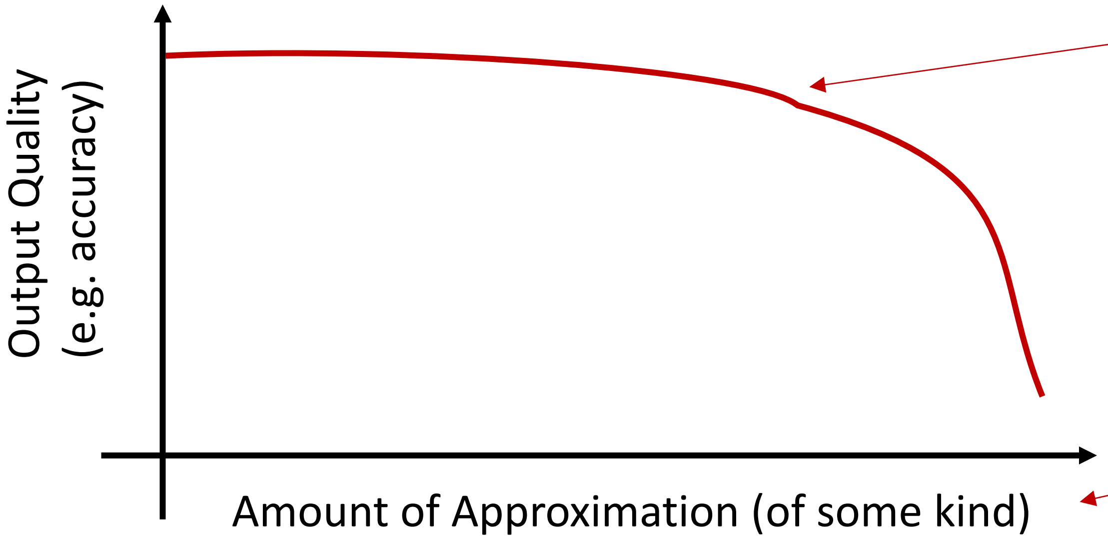
DL Optimizations for IoT Devices

Tolerance to Approximations in Deep Learning

Deep Learning Models Tolerate Approximations

- Most machine learning models, and deep learning ones in particular, are known to be quite *resilient* to various kinds of approximations.
- Changing slightly the input data or the internal computations, often doesn't change the final result (e.g. predicted class label).



A lot of approximation, little quality degradation

- reducing the precision of computations (quantization)
- skipping computations (pruning)
- etc.

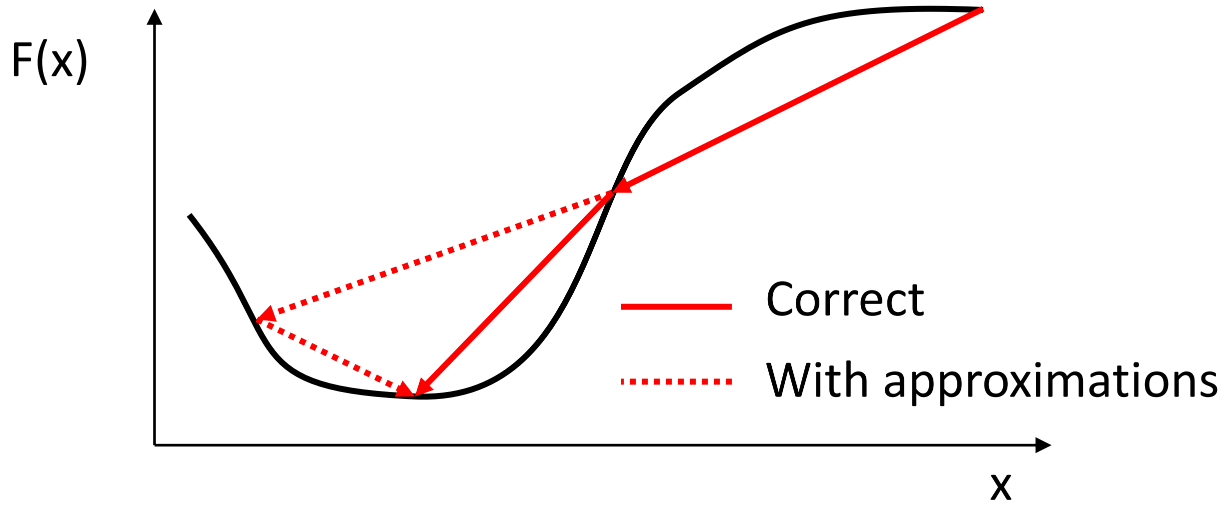
Deep Learning Models Tolerate Approximations

- We can exploit this tolerance to approximations to make our models faster, smaller and more efficient
- Intuitively, we can approximate complex computations with simpler ones, large data with more compact representations, etc.
- This is an instance of a paradigm called **Approximate Computing,** which does not only apply to ML, but also to other domains, such as multimedia processing, optimization, etc.

Deep Learning Models Tolerate Approximations

- Yes, but why are DL models tolerant to approximations?

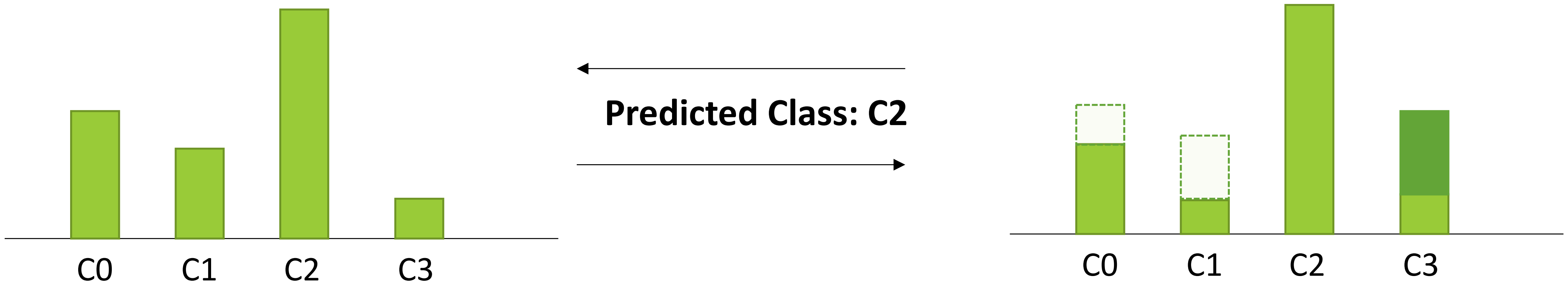
- **Reason 1:** Gradient descent-based training algorithms converge even in presence of small "deviations" from the correct gradient direction of a mini-batch.



Deep Learning Models Tolerate Approximations

- Yes, but why are DL models tolerant to approximations?

- **Reason 2:** often we don't care about the exact output value. For instance, the output of a NN for multi-class classification are probability scores for each class. But finally, for most applications, we only care about the *argmax* of the output array, not about the exact probability values.



Deep Learning Models Tolerate Approximations

- Yes, but why are DL models tolerant to approximations?

- **Reason 3:** (linked with #2) DL models are highly **redundant.** They approximate a function using many more parameters and computations than those that are actually needed to reach that accuracy.

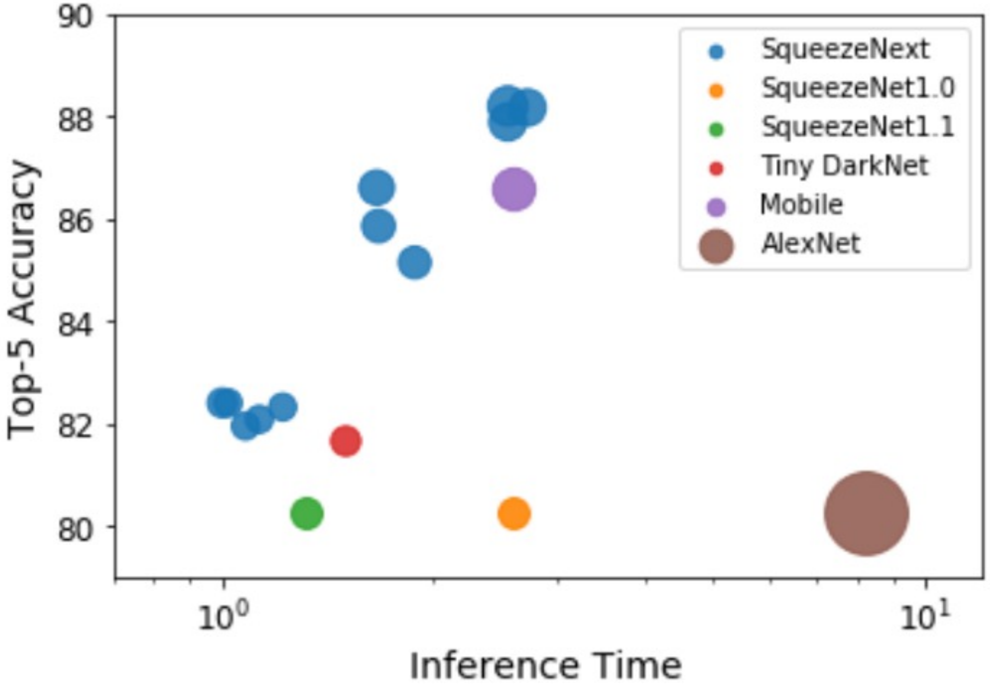
- Why (intuitively)?
  - If we knew the target function, we would be able remove this redundancy, but we don't.
  - So, we normally "over-design" our model to make sure that it has a sufficient *capacity* to create a mapping that is close enough to the original function.

Deep Learning Models Tolerate Approximations

- Because of "Reason 3", in many cases, the **best way** to improve the efficiency of a DL model is to **simplify the network architecture!**

- Reduce # of layers, change type of layer (e.g. depth-wise/group-wise convolution), <u>downscale input</u>, etc.

- This often yields more efficient networks <u>without accuracy loss</u>!!
  - Example: SqueezeNet vs AlexNet (50x less parameters, same accuracy)



Deep Learning Models Tolerate Approximations

- However, architectural improvements still require human creativity
  - NN architecture design has replaced feature engineering in classic ML
  - Hard, time-consuming, manual process.

- Fortunately, we also have systematic ways to simplify Deep Learning models by inserting approximations. In particular, we'll focus on:
  1. **Data Quantization**
  2. **Network Pruning**
  3. **Neural Architecture Search**
  4. **Adaptive/dynamic models**

Deep Learning Models Tolerate Approximations

- Importantly, these techniques can yield performance/energy benefits only thanks to a **synergy between hardware and ML models.** In other words:
  - Model simplifications must be designed in such a way that they actually result in a more efficient execution on the target hardware…
  - …or vice versa, the hardware must be designed to exploit a given model simplification!

Deep Learning Models Tolerate Approximations

- Importantly, these techniques can yield performance/energy benefits only thanks to a **synergy between hardware and ML models.** In other words:

  - Model simplifications must be designed in such a way that they actually result in a more efficient execution on the target hardware…

  - …or vice versa, the hardware must be designed to exploit a given model simplification!

- Since this is not a HW design course, we'll focus mostly on model optimizations that can yield benefits on general purpose HW (MCUs, CPUs, etc.)
  - Particularly useful for edge devices, where the HW platform is typically general purpose