CudaTree

Training Random Forests on the GPU

Alex Rubinsteyn (NYU, Mount Sinai)

@ GTC on March 25th, 2014

What's a Random Forest?

A bagged ensemble of randomized decision trees. Learning by uncorrelated memorization.

```
trees = []
for i in 1 .. T:
  Xi, Yi = random_sample(X, Y)
  t = RandomizedDecisionTree()
  t.fit(Xi,Yi)
  trees.append(t)
```

Few free parameters, popular for "data science"

Decision Tree Training

Random Forest trees are trained like normal decision trees (CART), but use random subset of features at each split.

```
bestScore = ∞;  bestThresh = None
for i in RandomSubset(nFeatures):
 Xi, Yi = sort(X[:, i], Y)
 for nLess, thresh in enumerate(Xi):
   score = Impurity(nLess, Yi)
   if score < bestScore:
     bestScore = score
     bestThresh = thresh
```

Random Forests: Easy to Parallelize?

- Each tree gets trained independently!
- Simple parallelization strategy: "*each processor trains a tree*"
- Works great for multi-core implementations (wiseRF, scikit-learn, &c)
- Terrible for GPU

Random Forest GPU Training Strategies

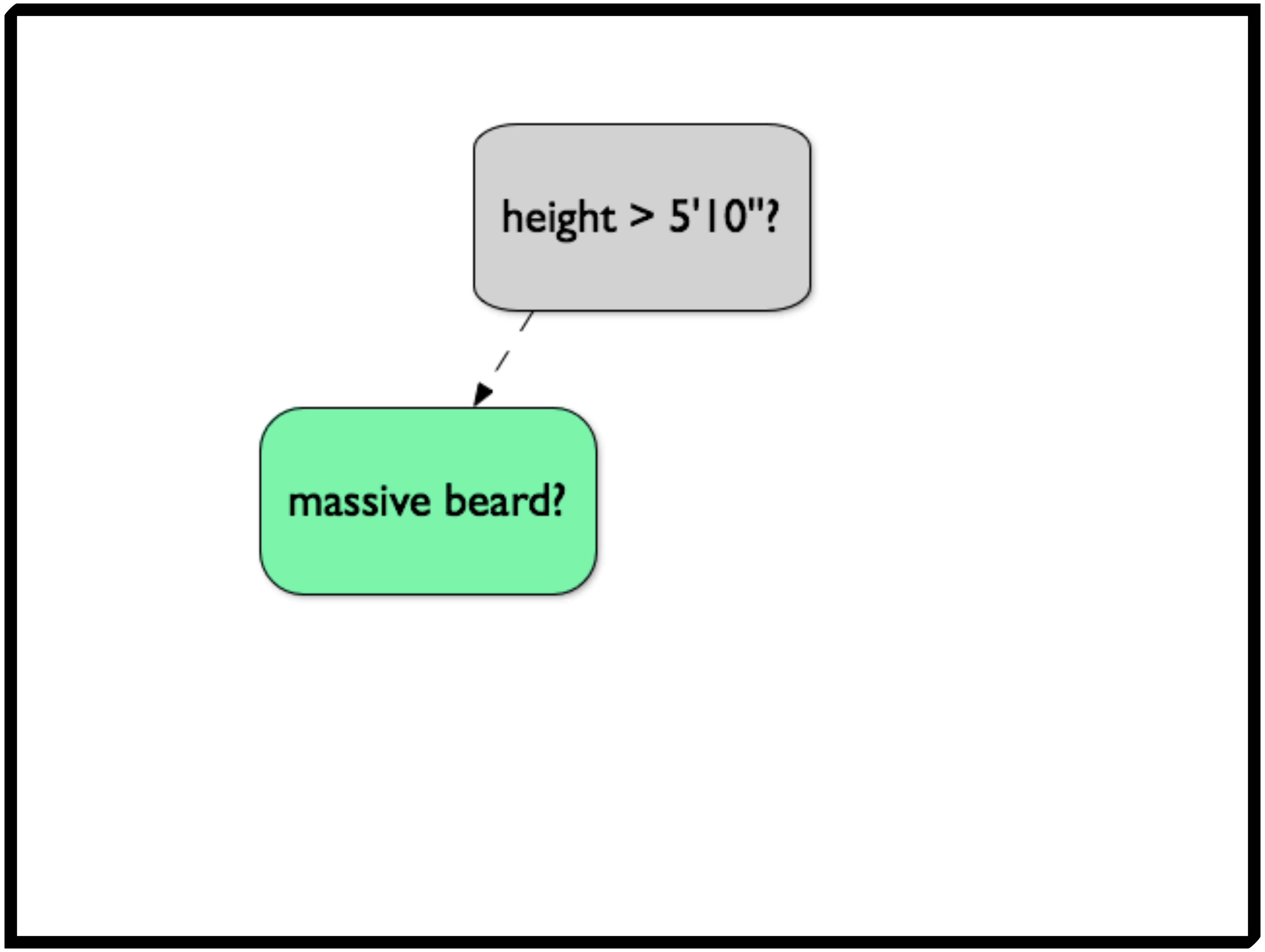
- Tree per Thread: naive/slow
- Depth-First: data parallel threshold selection for one tree node
- Breadth-First: learn whole level of a tree simultaneously (threshold per block)
- Hybrid: Use Depth-First training until too few samples, switch to Breadth-First

Depth-First Training

Thread block = subset of samples

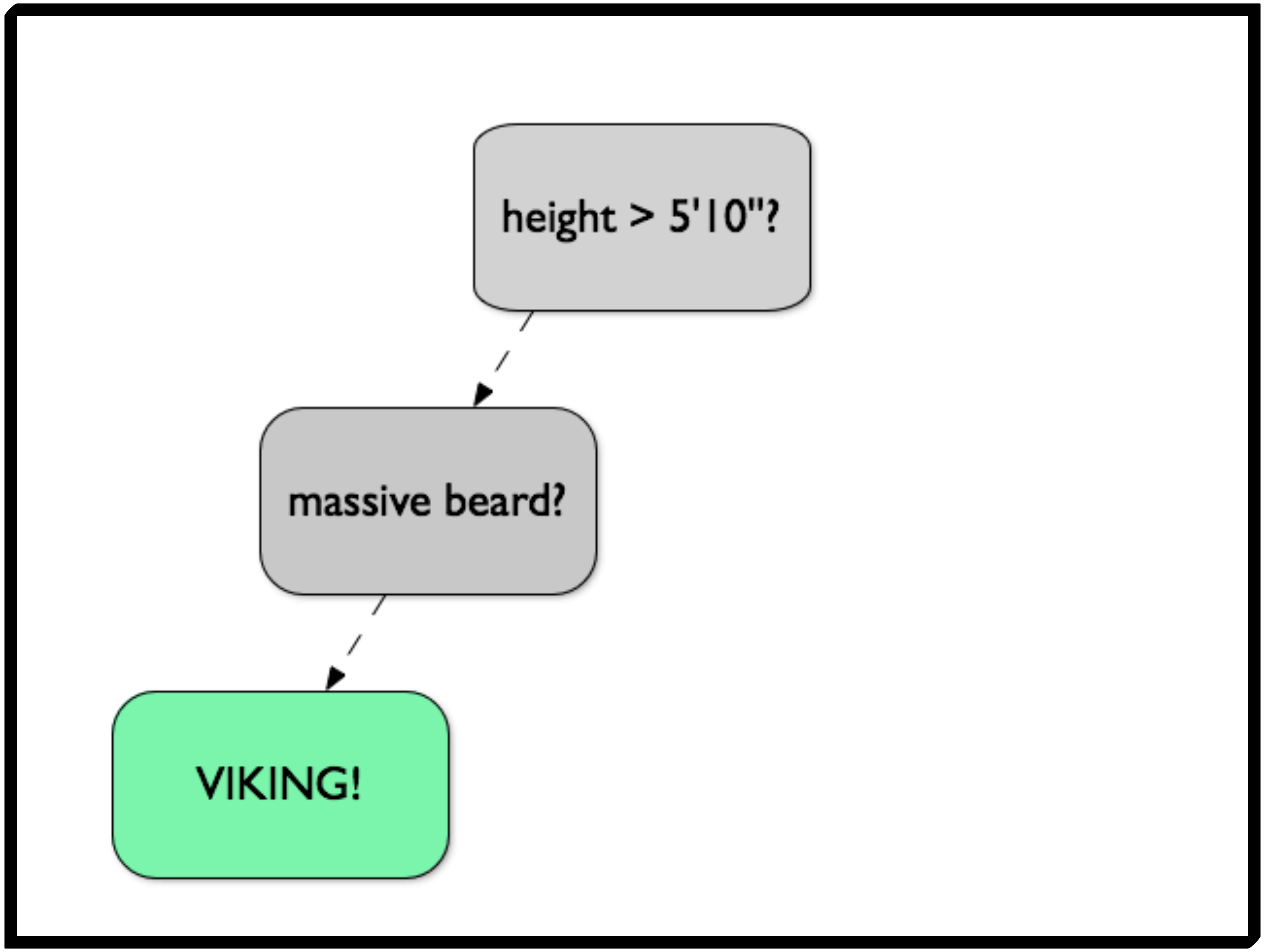
Depth-First Training

Thread block = subset of samples



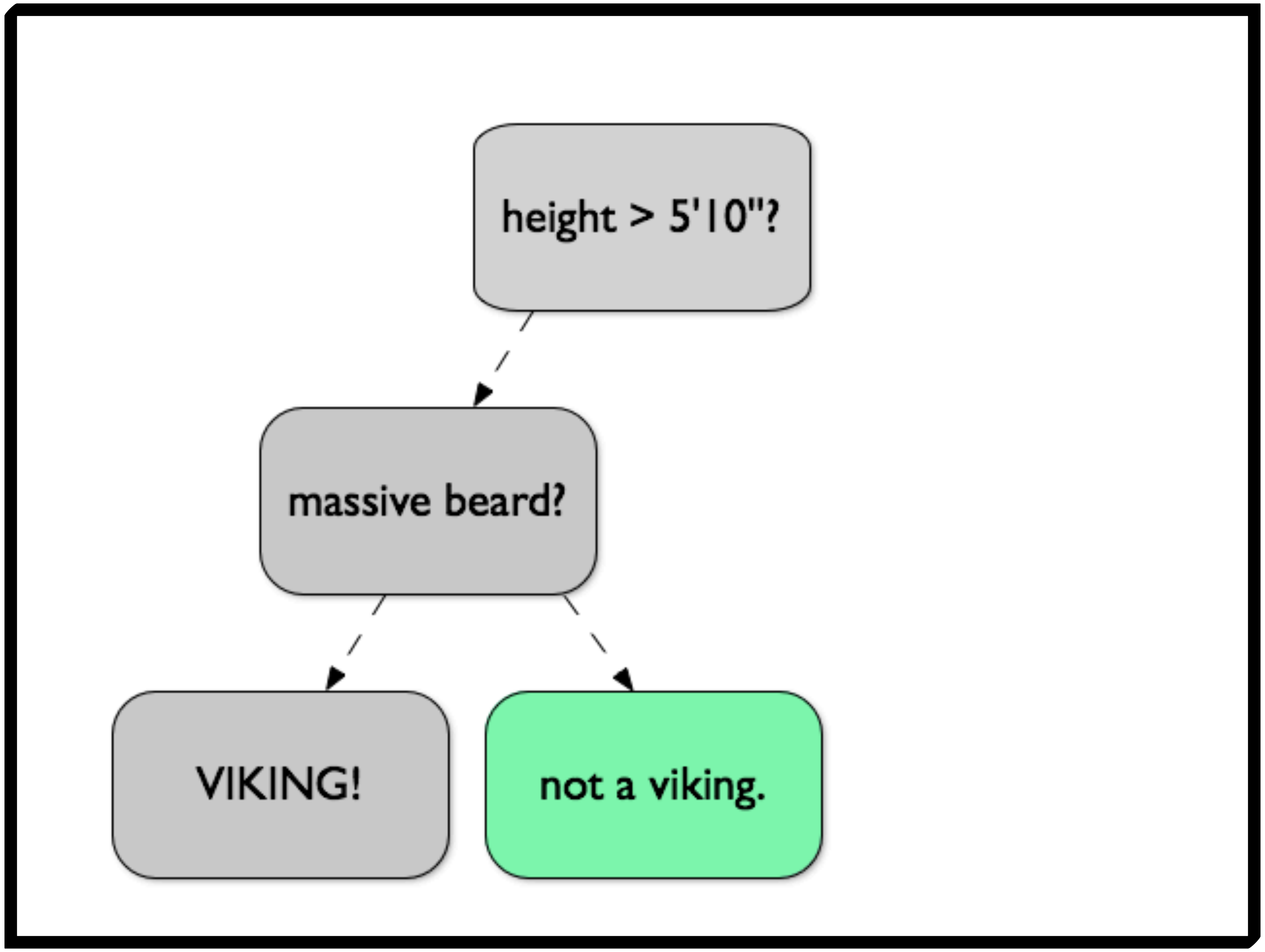
Depth-First Training

Thread block = subset of samples



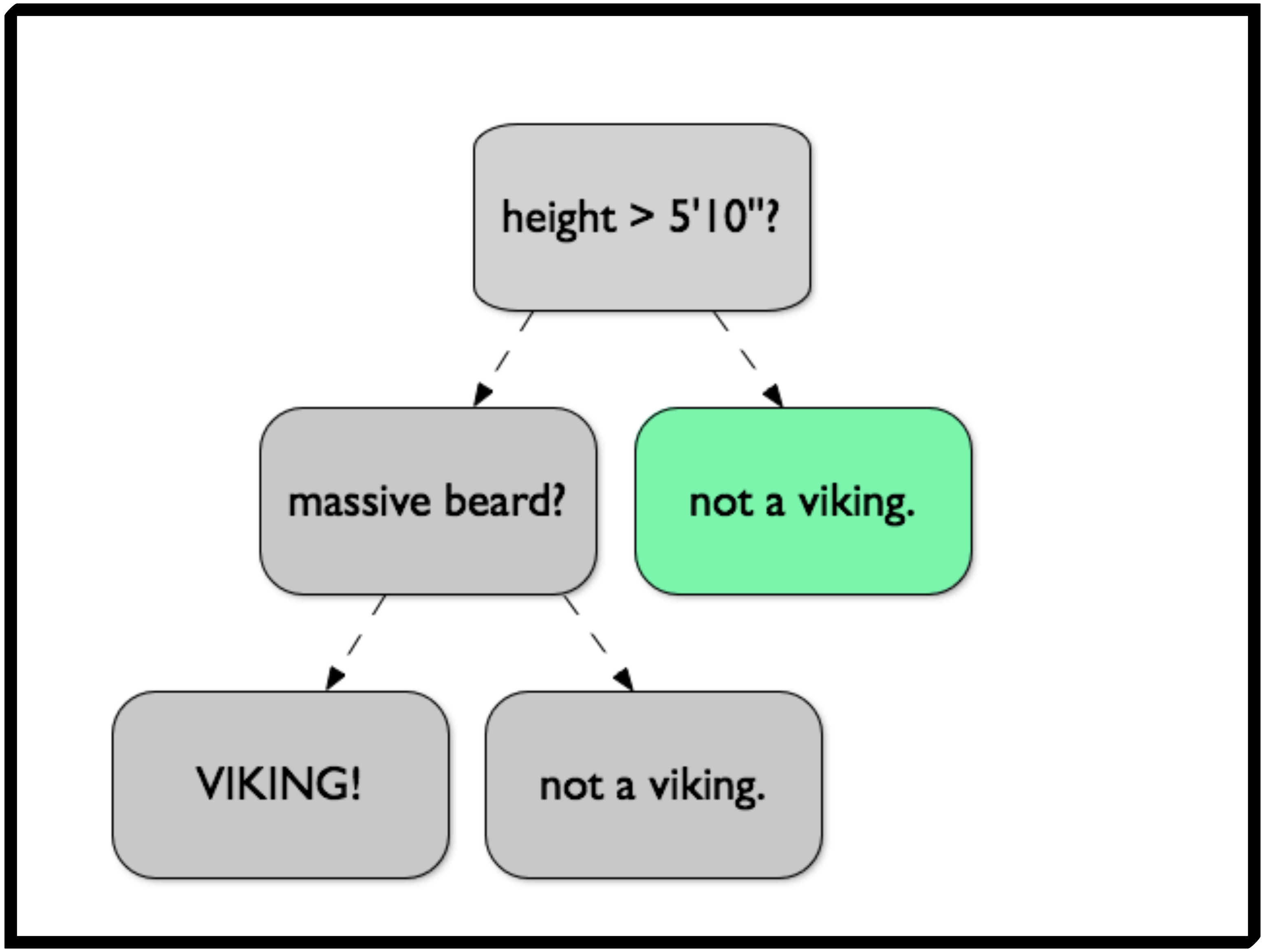
Depth-First Training

Thread block = subset of samples



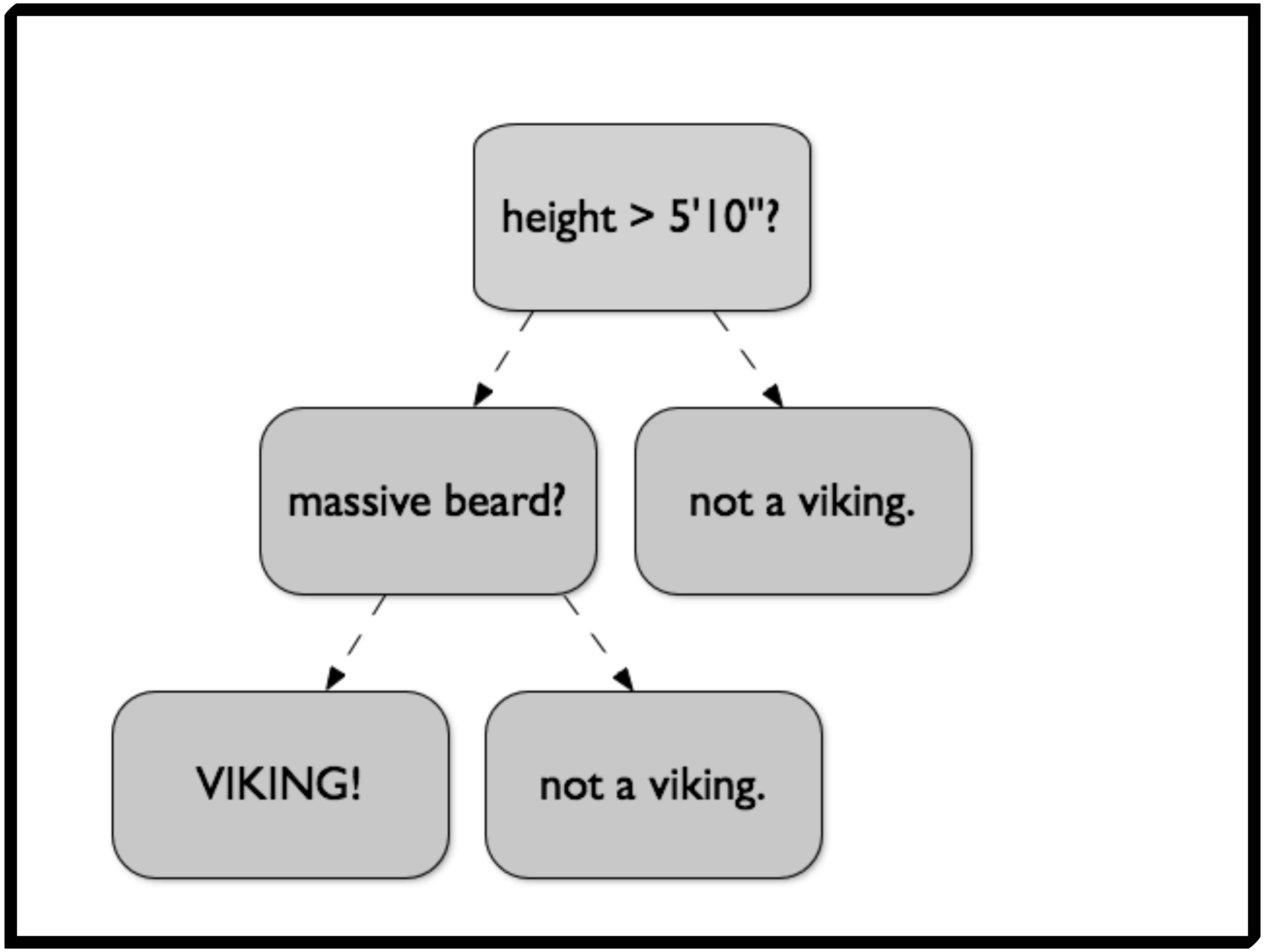
Depth-First Training

Thread block = subset of samples



Depth-First Training

Thread block = subset of samples



Depth-First Training: Algorithm Sketch

(1) **Parallel Prefix Scan:** compute label count histograms

(2) **Map:** Evaluate impurity score for all feature thresholds

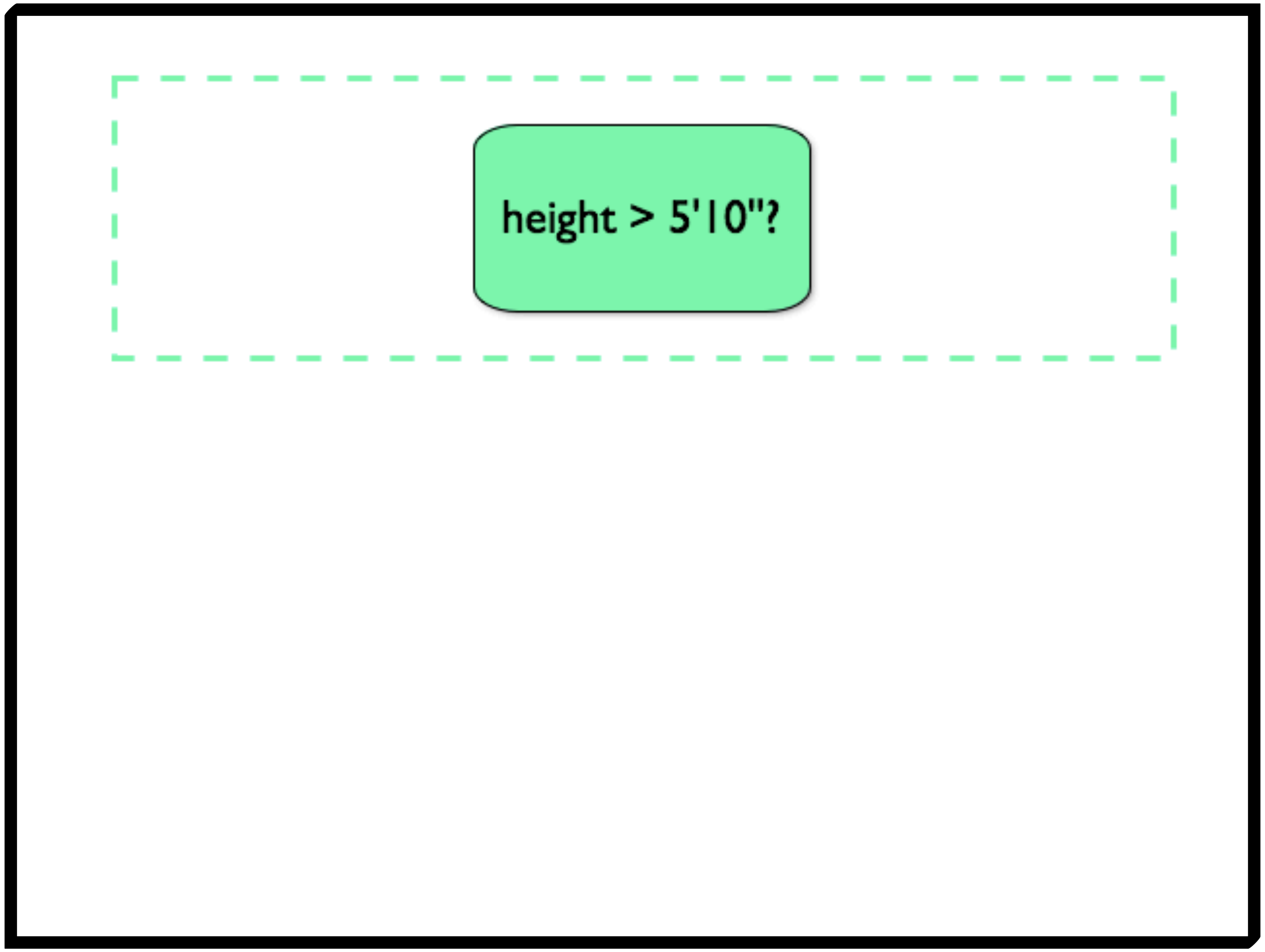
(3) **Reduce:** Which feature/threshold pair has the lowest impurity score?

(4) **Map:** Marking whether each sample goes left or right at the split

(5) **Shuffle:** keep samples/labels on both sides of the split contiguous.

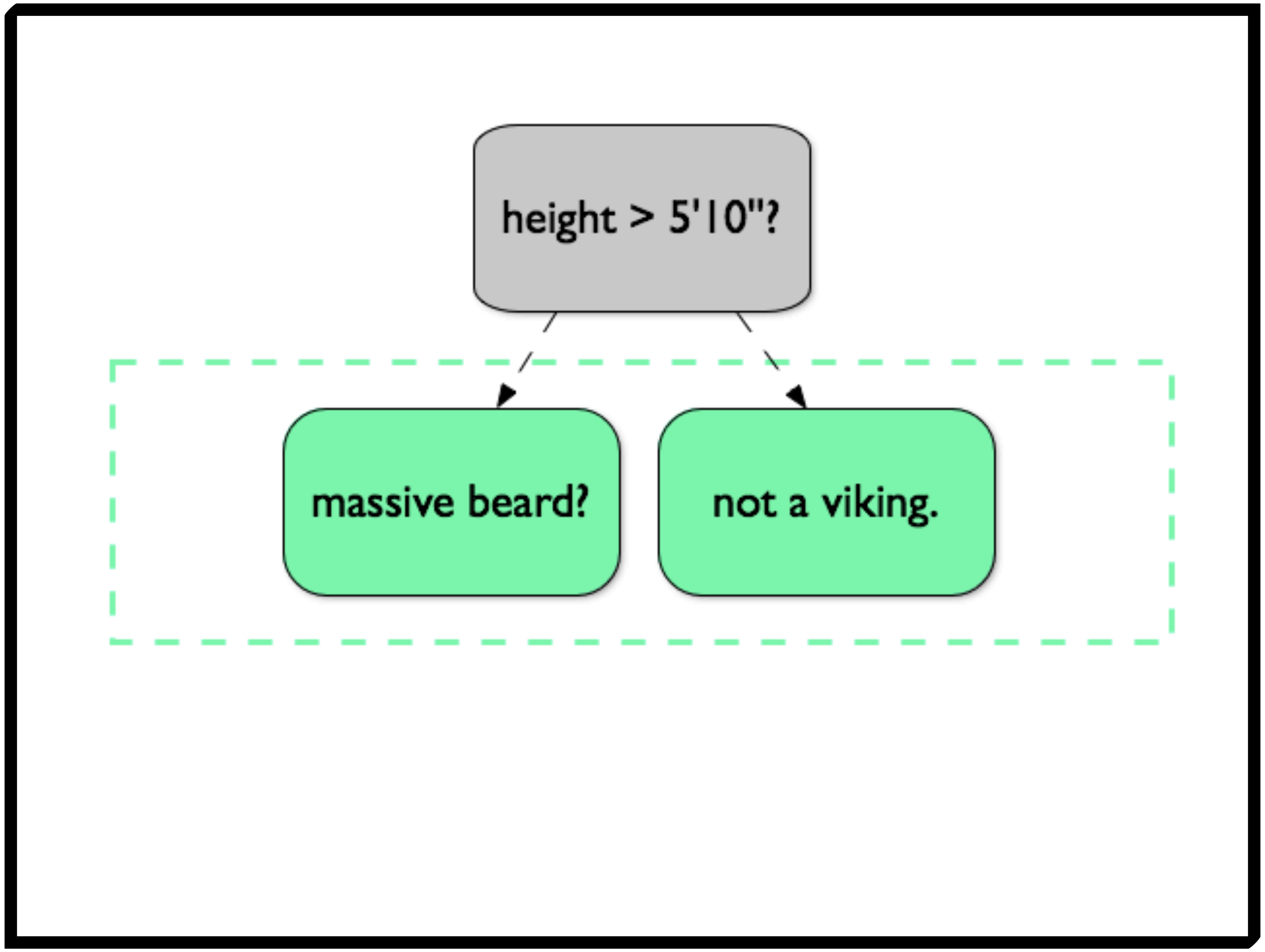
Breadth-First Training

Thread block = tree node



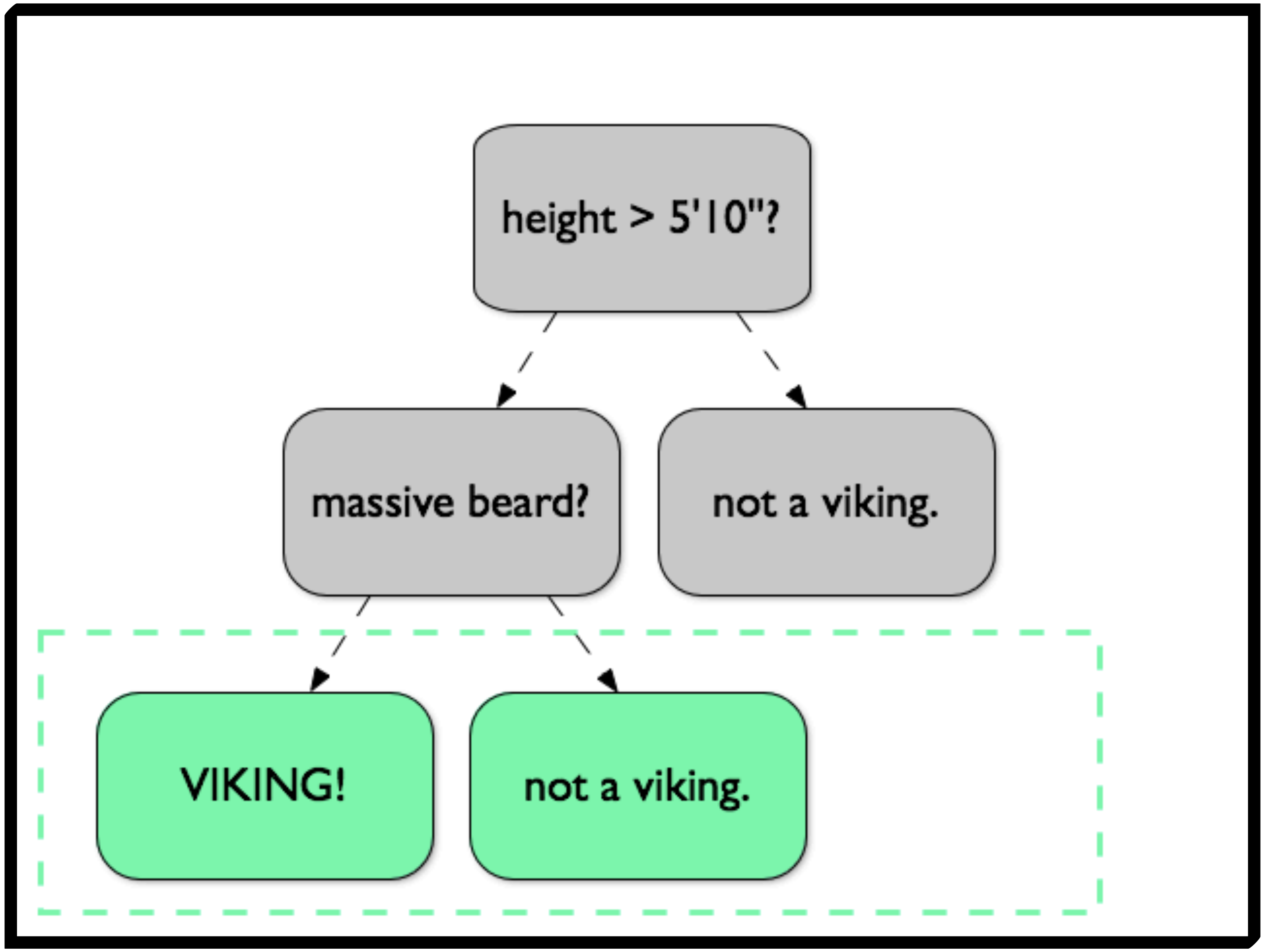
Breadth-First Training

Thread block = tree node



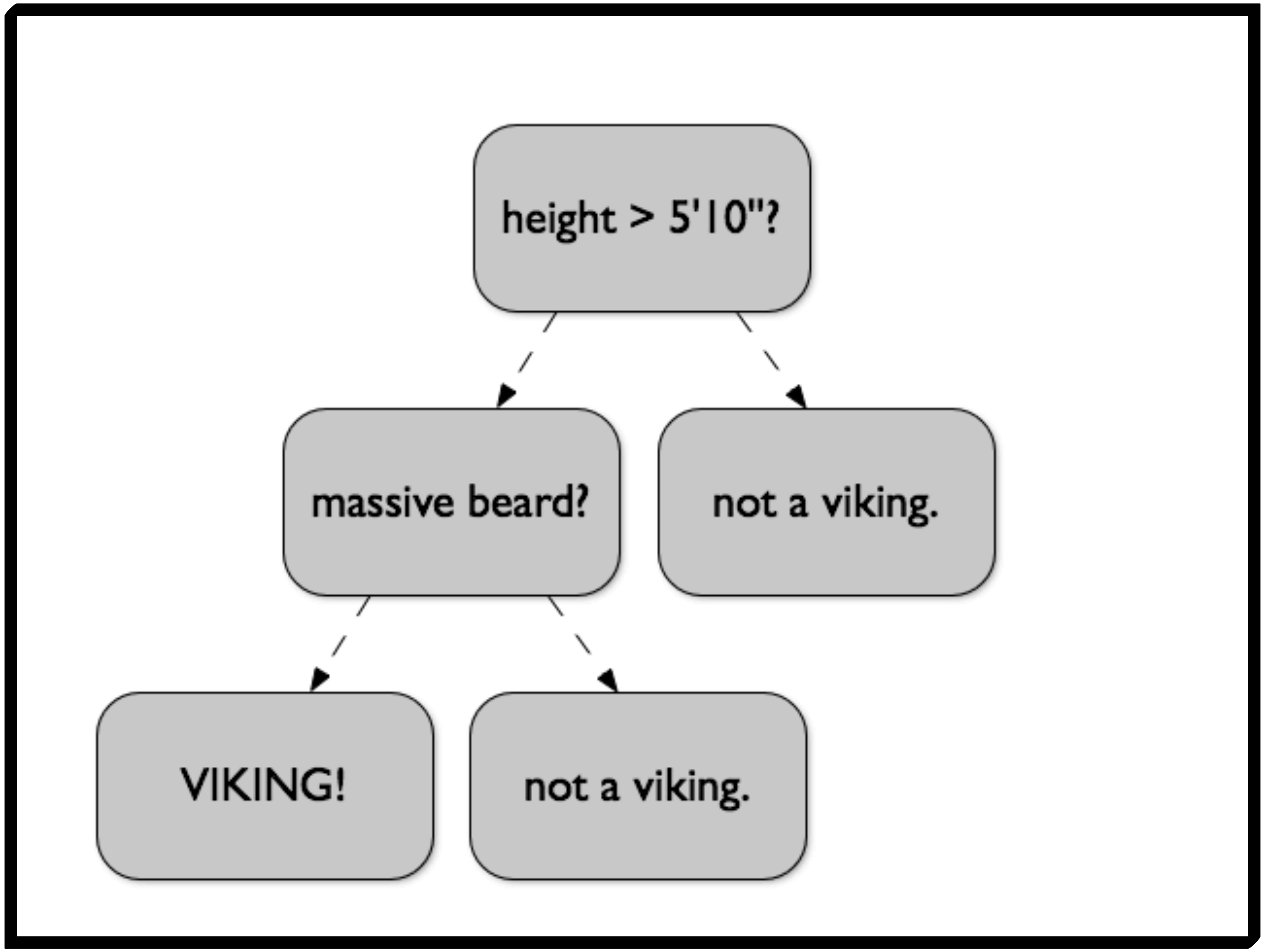
Breadth-First Training

Thread block = tree node



Breadth-First Training

Thread block = tree node



Breadth-First Training: Algorithm Sketch

Uses the same sequence of data parallel operations (Scan label counts, Map impurity evaluation, &c) as Depth-First training but ***within each thread block***

Fast at the bottom of the tree (lots of small tree nodes), insufficient parallelism at the top.

Hybrid Training

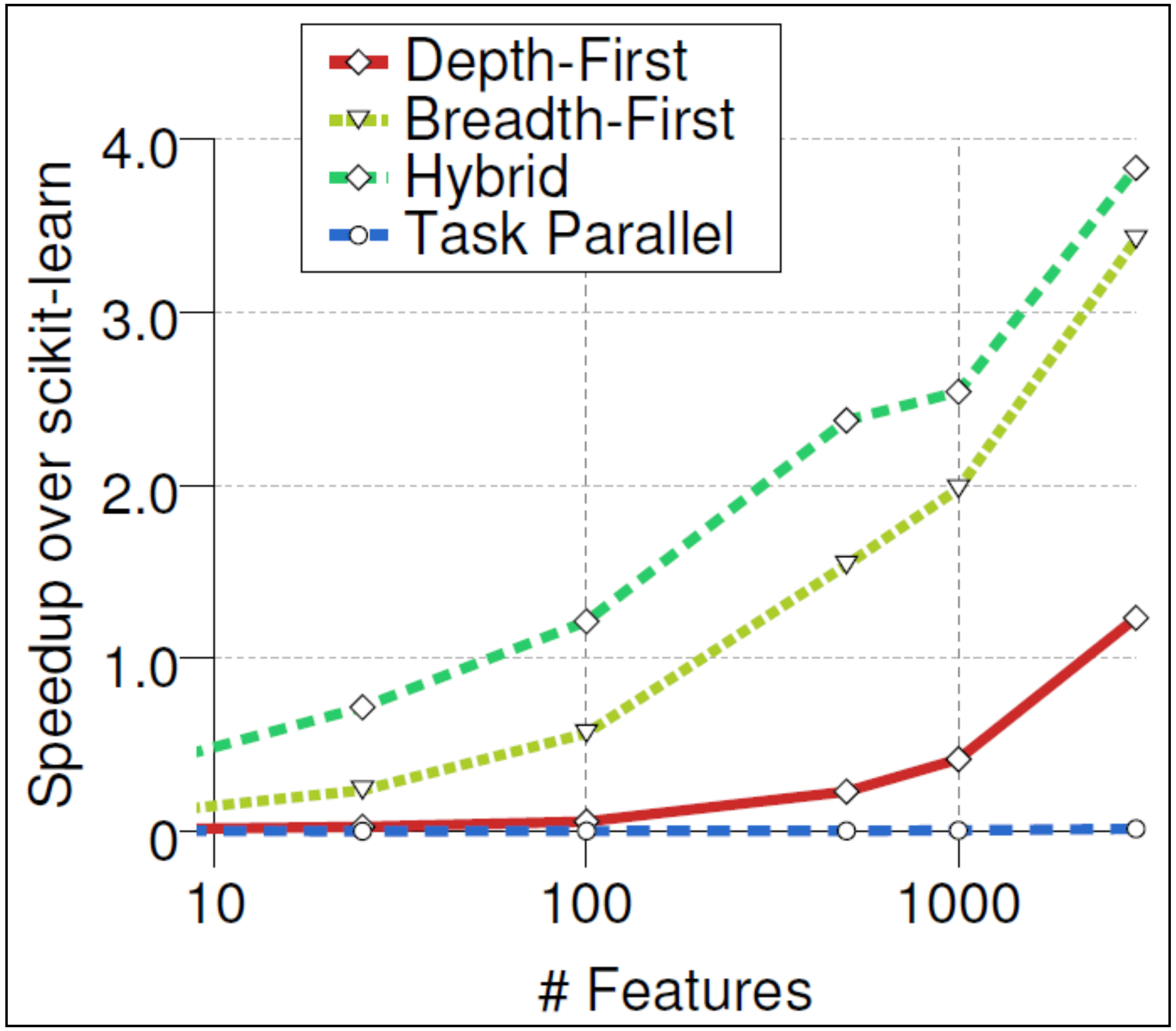
Add a tree node to the Breadth-First queue when it contains fewer samples than:

$$\boxed{3702 + 1.58c + 0.0577n + 21.84f}$$

- ★ ***c*** = number of classes
- ★ ***n*** = total number of samples
- ★ ***f*** = number of features considered at split

(coefficients from machine-specific regression, needs to be generalized)

GPU Algorithms vs. Number of Features



- Randomly generated synthetic data
- Performance relative to sklearn 0.14

Benchmark Data

Dataset	Samples	Features	Classes
ImageNet	10k	4k	10
CIFAR100	50k	3k	100
covertype	581k	57	7
poker	1M	11	10
PAMAP2	2.87M	52	10
intrustion	5M	41	24

Benchmark Results

Dataset	wiseRF	sklearn 0.15	CudaTree (Titan)	CudaTree + wiseRF
ImageNet	23s	**13s**	*27s*	25s
CIFAR100	160s	180s	*197s*	**94s**
covertype	*107s*	73s	67s	**52s**
poker	*117s*	98s	59s	**58s**
PAMAP2	*1,066s*	**241s**	934s	757s
intrustion	667s	*1,682s*	199s	**153s**

6- core Xeon E5-2630, 24GB, GTX Titan, n_trees = 100, features_per_split = sqrt(n)

Thanks!

- Installing: `pip install cudatree`
- Source : https://github.com/EasonLiao/CudaTree
- Credit : **Yisheng Liao** did most of the hard work, **Russell Power** & **Jinyang Li** were the sanity check brigade.