5th International Workshop on Deep Learning in Computational Physics

June 29, 2021

Online

A Convolutional Hierarchical Neural Network Classifier

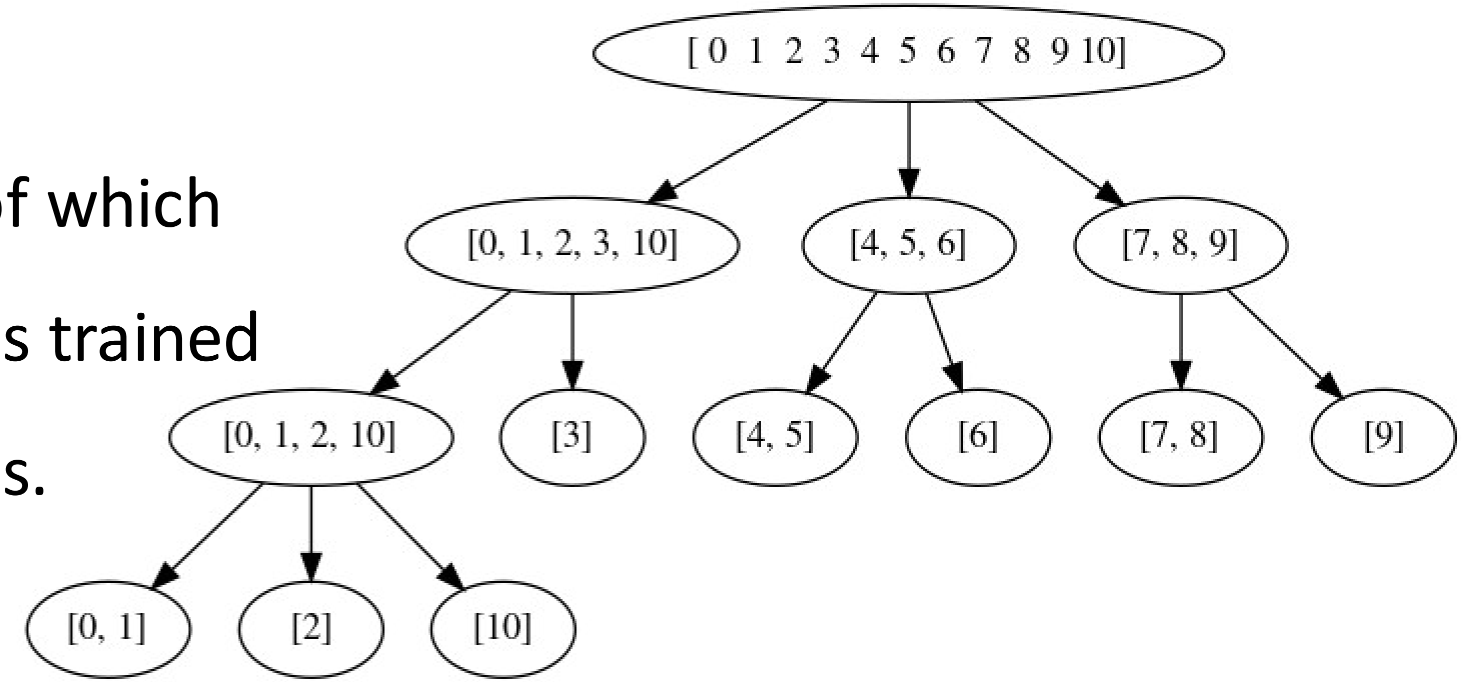
I.M.Gadzhiev, S.A.Dolenko*

D.V.Skobeltsyn Institute of Nuclear Physics,
M.V.Lomonosov Moscow State University

Hierarchical Neural Network Classifier (HNNC)

Solves multi-class classification problems.

HNNC is a tree, in each node of which there is a neural network that is trained on a certain subset of all classes.



Class groups are formed adaptively.

HNNC Training Algorithm

- The neural network in each node is a “weak” multi-layer perceptron (MLP) with a small number of neurons (2-4) in the single hidden layer, solving a classification problem with one-hot encoding at the output
- After a pre-defined number of training epochs, the statistics of the network output on the training set is analyzed
- If some classes are consistently mixed up (for the majority of “voting” input patterns of a class), such classes are merged by modification of the desired output
- The cycle “training – class merging” is repeated while classes continue to merge (and the number of classes is greater than 2), and until the network error for the modified class set on the validation dataset stops to decrease for a pre-defined timeout
- For each class group that has been formed by class merging, a new node is trained

Convolutional HNNC (CHNNC)

- Designed to solve multi-class image classification problems
- Each node contains a convolutional neural network (CNN) instead of an MLP
- The hierarchy of the classes obtained with “weak” CNNs can be re-used with “strong” CNNs to improve the recognition rate (too low with “weak” CNNs)

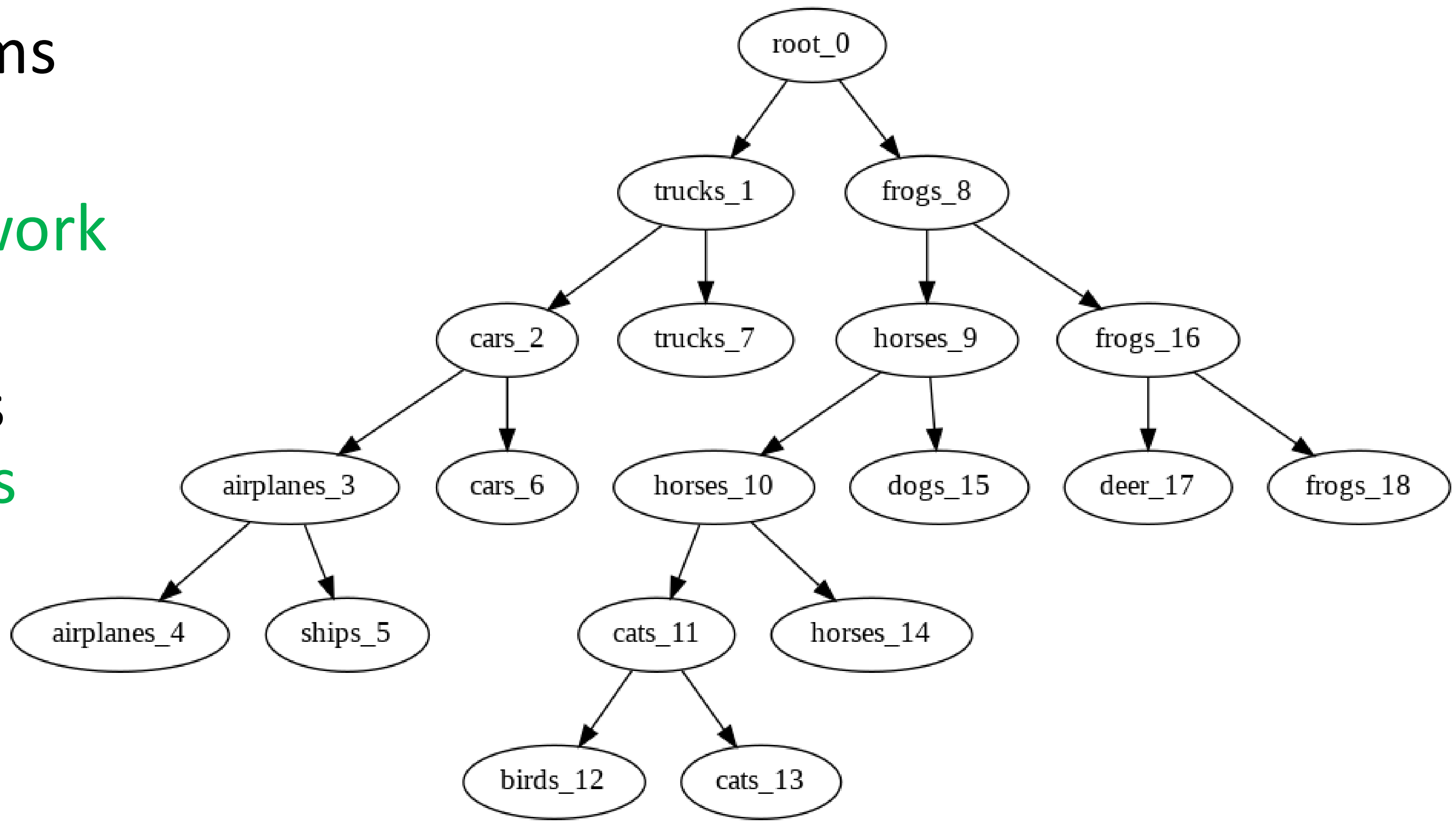
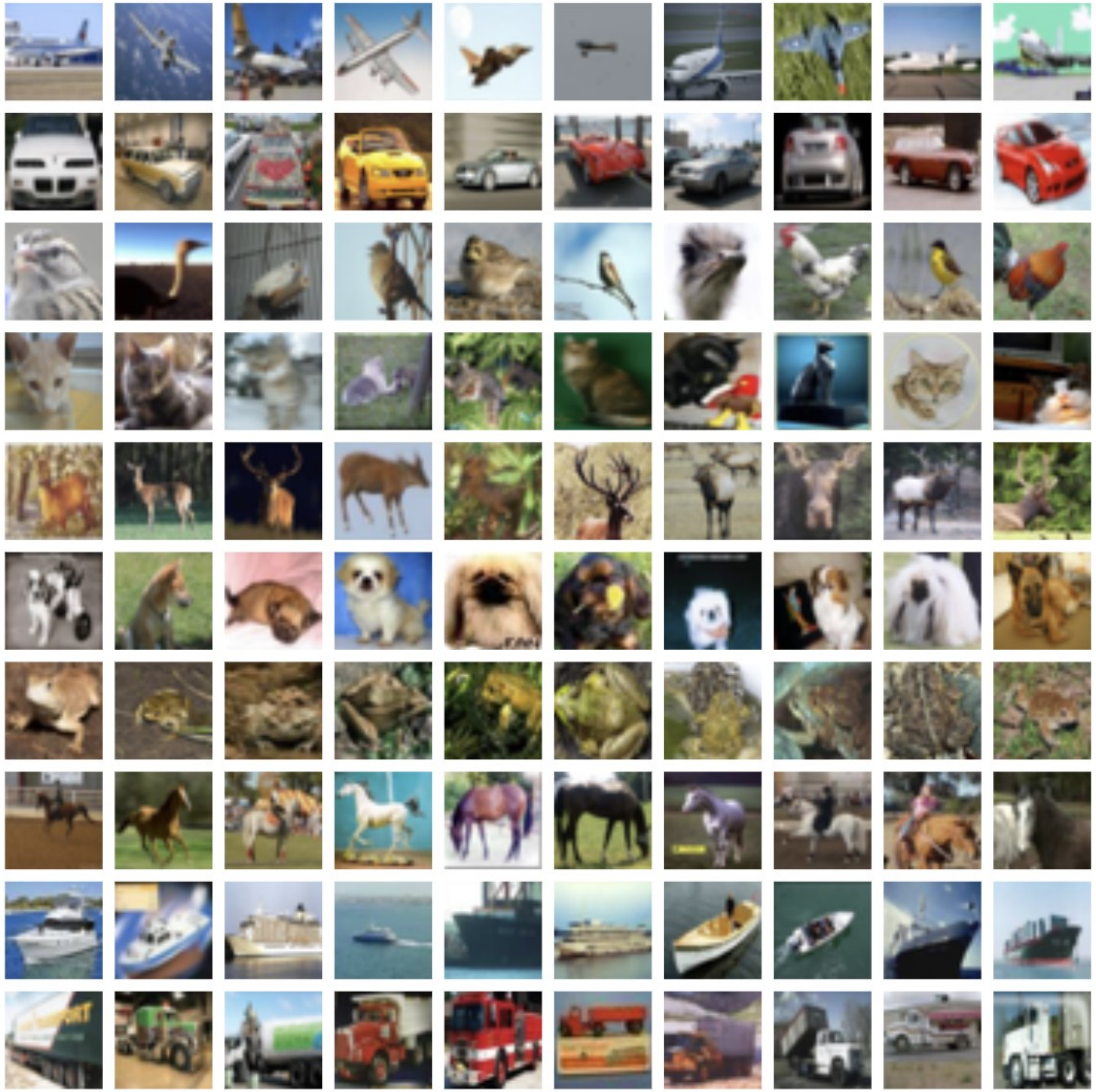


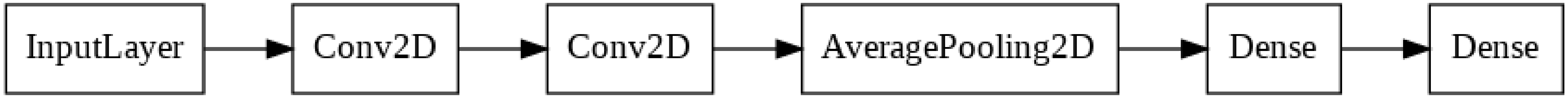
Image Classification Benchmark Problems

The figure displays patterns from benchmark datasets **CIFAR-10** and **CIFAR-100**, consisting of images of 10 and 100 classes, respectively.



Architectures of CNN Models in a Node

The weak CNN used at the first stage of the algorithm.
The single convolutional layer contains 2-3 neurons (filters).
Each dense layer contains 2-3 neurons in the single hidden layer.

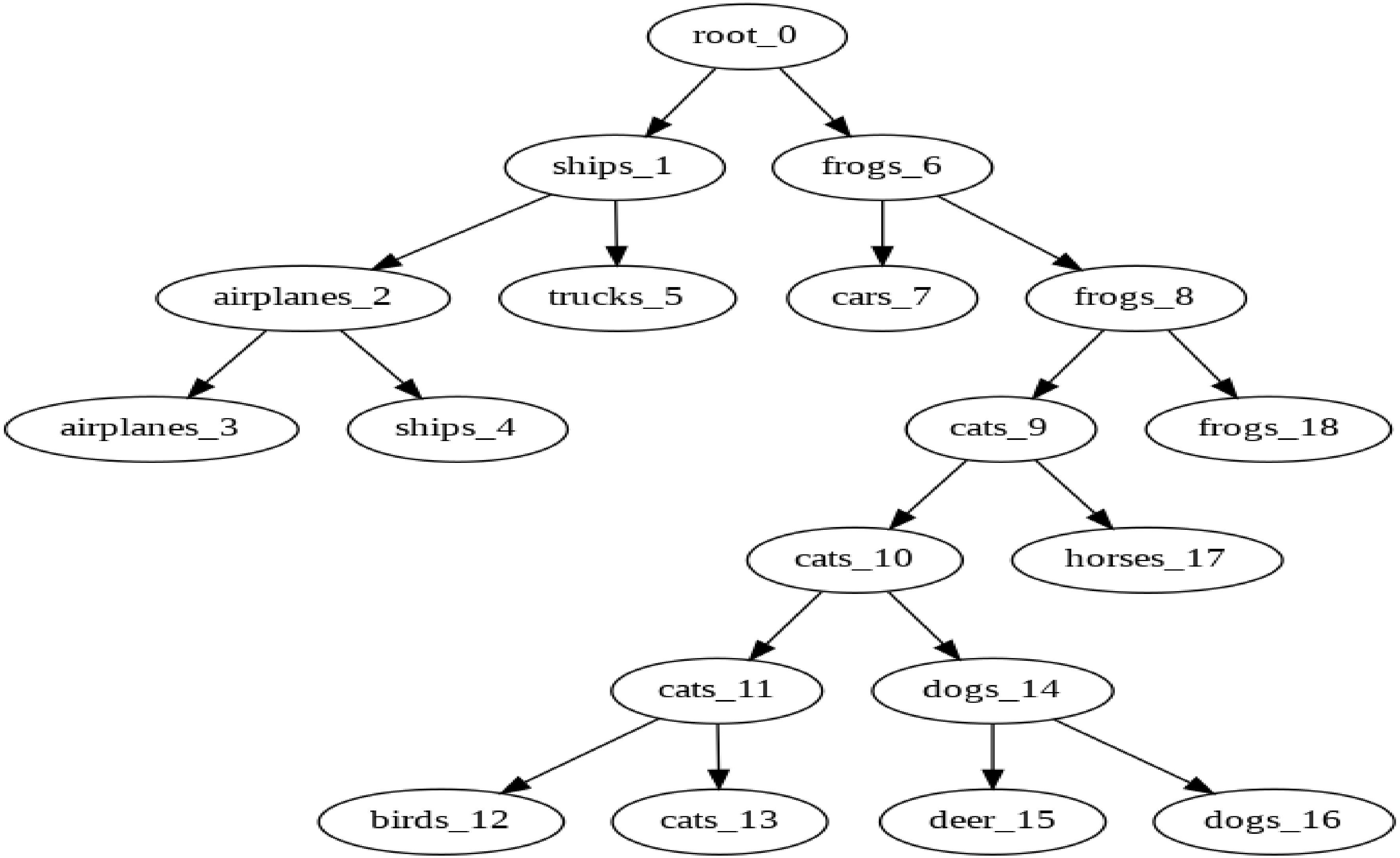


The strong CNN used at the second stage of the algorithm.
Each convolutional layer contains 32 neurons (filters).
Each dense layer contains 10 neurons in the single hidden layer.

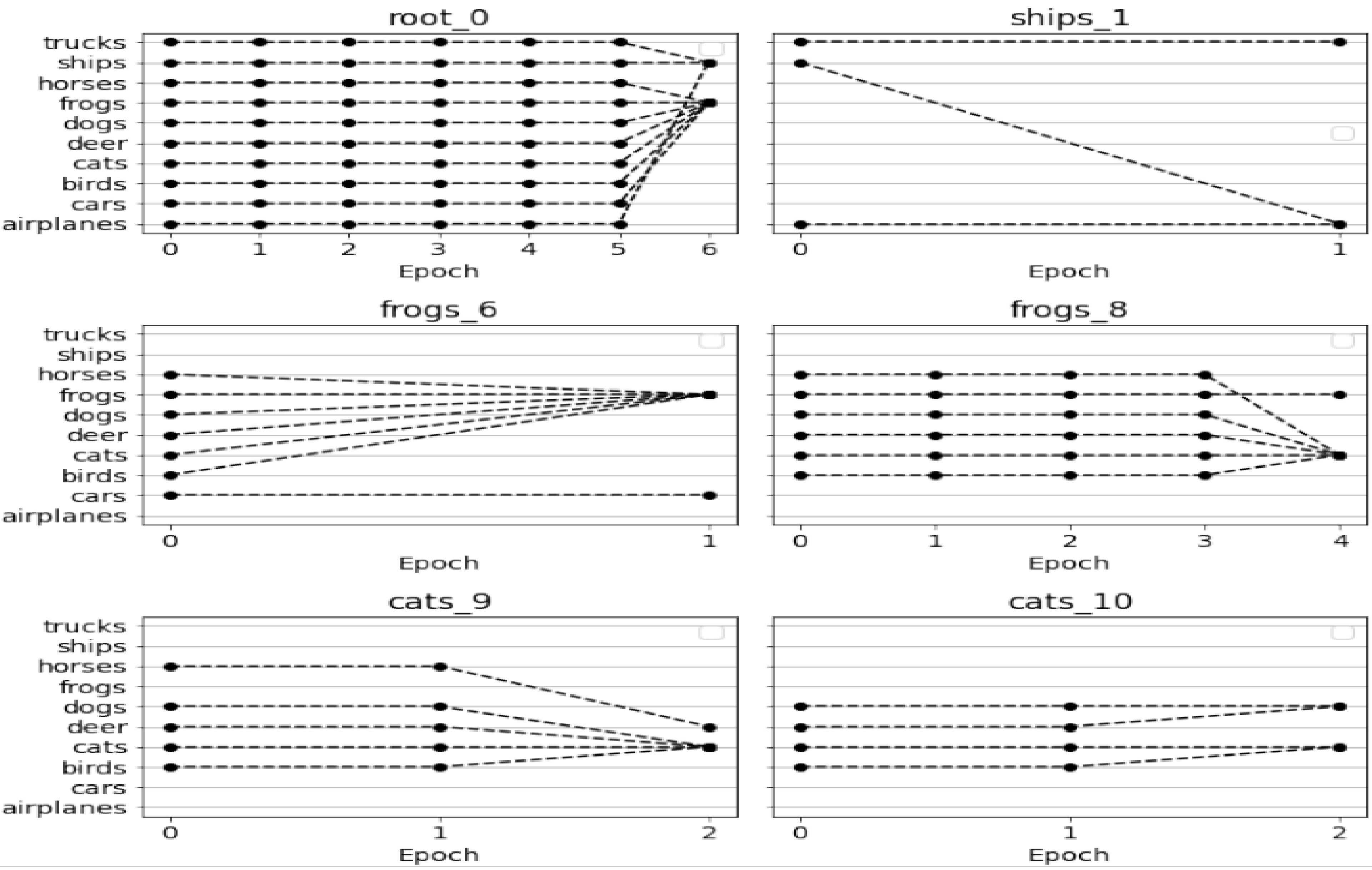
First Stage - Construction of the Hierarchy

- The weak CNN is trained on the initial set of data.
- After a pre-defined number of training epochs,
  the classes, which get mixed up, are merged.
- The cycle “training – class merging” is repeated
  while classes continue to merge
  (and the number of classes is greater than 2),
  and until the network error for the modified class set
  on the validation dataset
  stops to decrease for a pre-defined timeout.
- For each class group that has been formed by class merging,
  the described algorithm is applied recursively

An Example of Class Hierarchy for CIFAR-10



An Example of Class Merging Diagram (CIFAR-10)



Second Stage - Training Strong Models

- A CNN with stronger parameter values is placed in each node of the resulting hierarchy.
- Training is repeated without class merging.
- The weak models are effective for tree construction, but their use result in low recognition rate.
- Use of strong models allows one to reduce the effects of error multiplication when moving down the tree. (Multiplication of errors of weak models result in an extremely weak classifier).

Construction of the Hierarchy: Pattern Voting

- To determine which classes the network mixes up most often, voting of patterns of each class is performed on the training set.
- If most of the patterns from *Class i* are assigned to *Class j*, then these two classes are merged into one.
- **Problem** – influence of random weight initialization on the voting.
- Result – *random merging* of classes and low reproducibility of the design of the classification tree.

Pattern Voting: Activation Threshold and Voting Patterns Fraction Threshold

- The thresholds are introduced to solve the problem of random merging
- **Activation Threshold**:
  The votes of only those patterns,
  for which the maximum activation of the output neuron
  is greater than the **Activation Threshold**,
  are taken into account
- **Voting Patterns Share Threshold:**
  Class i is merged with *Class j* only if
  the percentage of *Class i* patterns that voted for *Class j*
  is greater than the **Voting Patterns Share Threshold**

Selection of the Activation Threshold

Two methods of selection of the **Activation Threshold** are tested:

1. Setting fixed threshold values for all nodes.

   The optimal value of the threshold is determined by enumeration.

2. Setting the threshold equal to the upper boundary of the output activation localization area with random initialization.

   We define the localization area as

   $$\theta = \mu + k\sigma$$

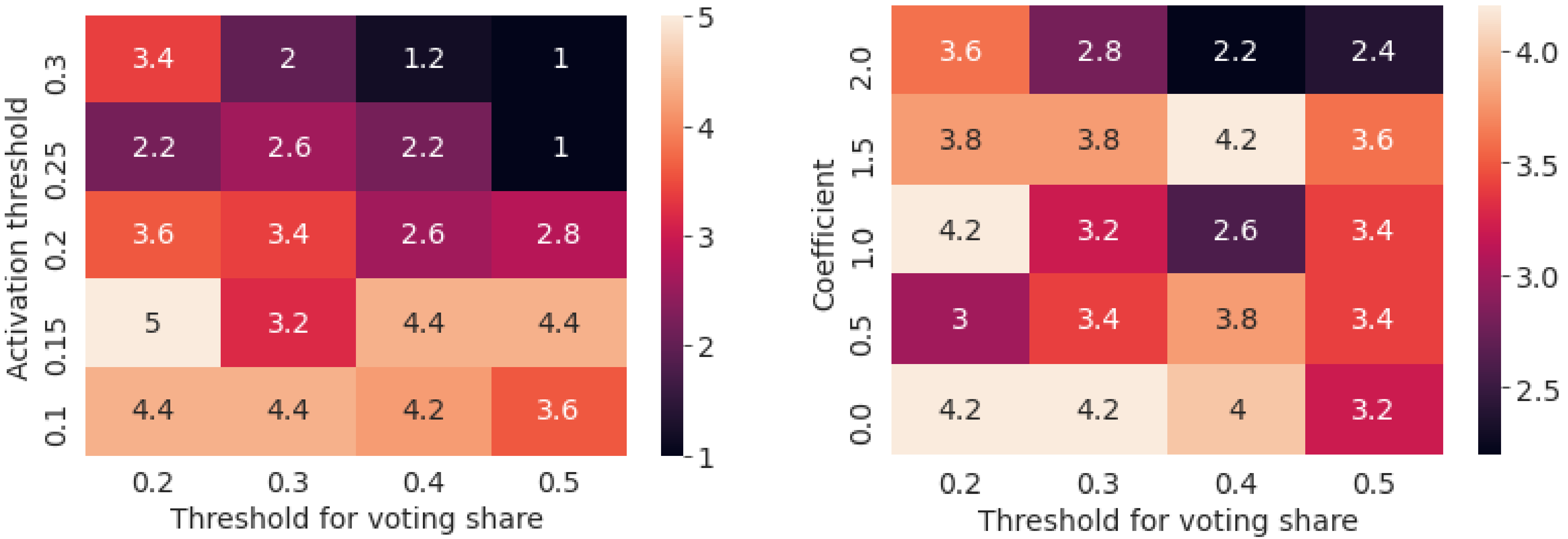
   where μ and σ are the mean and standard deviation of the output activations on the patterns from the training set in this node with random initialization of the weights.

   k is the coefficient selected by enumeration.

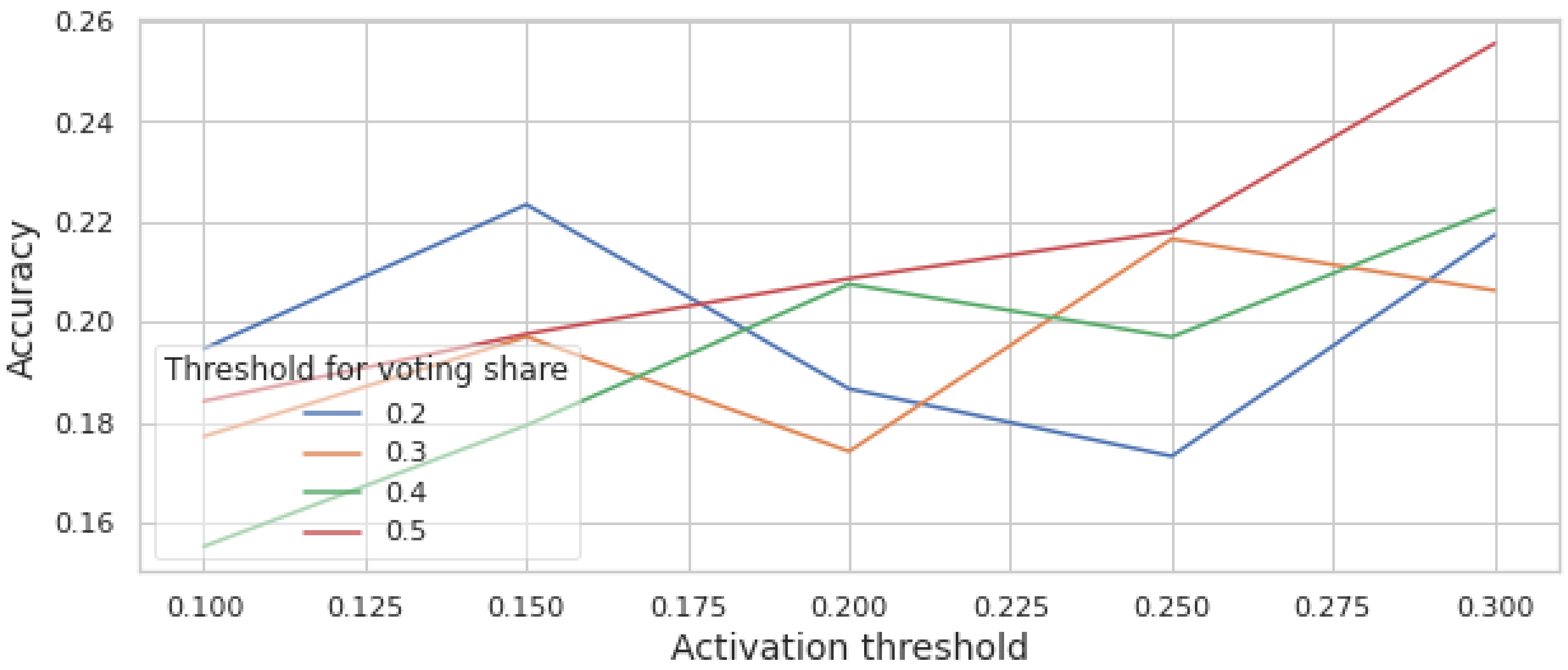
Algorithm Testing for Various Threshold Values

- Benchmark Dataset – CIFAR-10.
- Weak CNN: 3 filters 2×2 in 1 convolutional layer,
  3+10 neurons in the fully connected layers.
- Strong CNN: 32 filters 4×4 in each of the 2 convolutional layers,
  10+10 neurons in the fully connected layers.
- Adam, learning rate 0.001 (Tensorflow library).
- 5 runs for each pair of threshold values, results averaged.
- For the first and second algorithm stages,
  the training set was split into two parts in 1:4 proportion.
- **Voting Patterns Share Threshold** values from 0.2 to 0.5 step 0.1.
- Two methods for **Activation Threshold** values:
  1. Fixed values from 0.1 to 0.4 step 0.05
  2. Values of k from 0 to 2 step 0.5

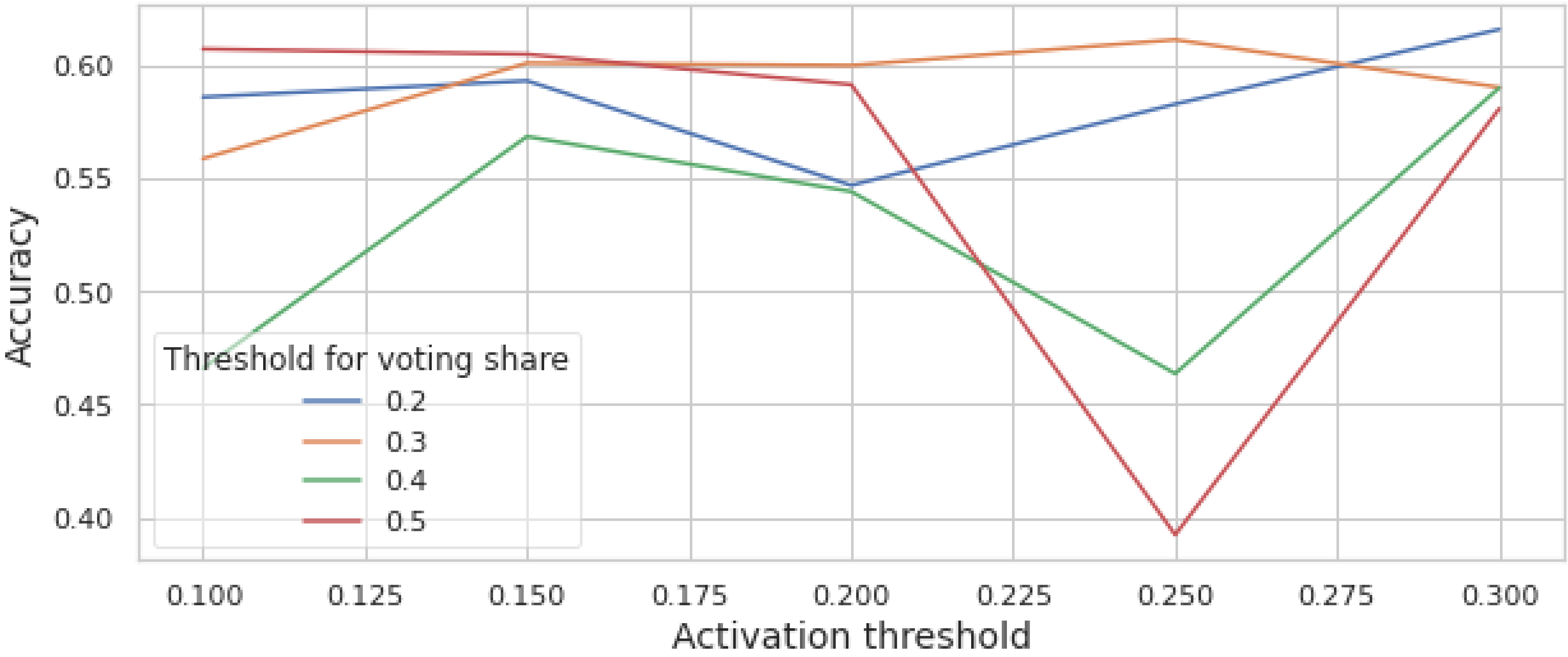
Threshold Values Control the Height of the Tree



Results: Fixed Activation Threshold, Stage 1 Only

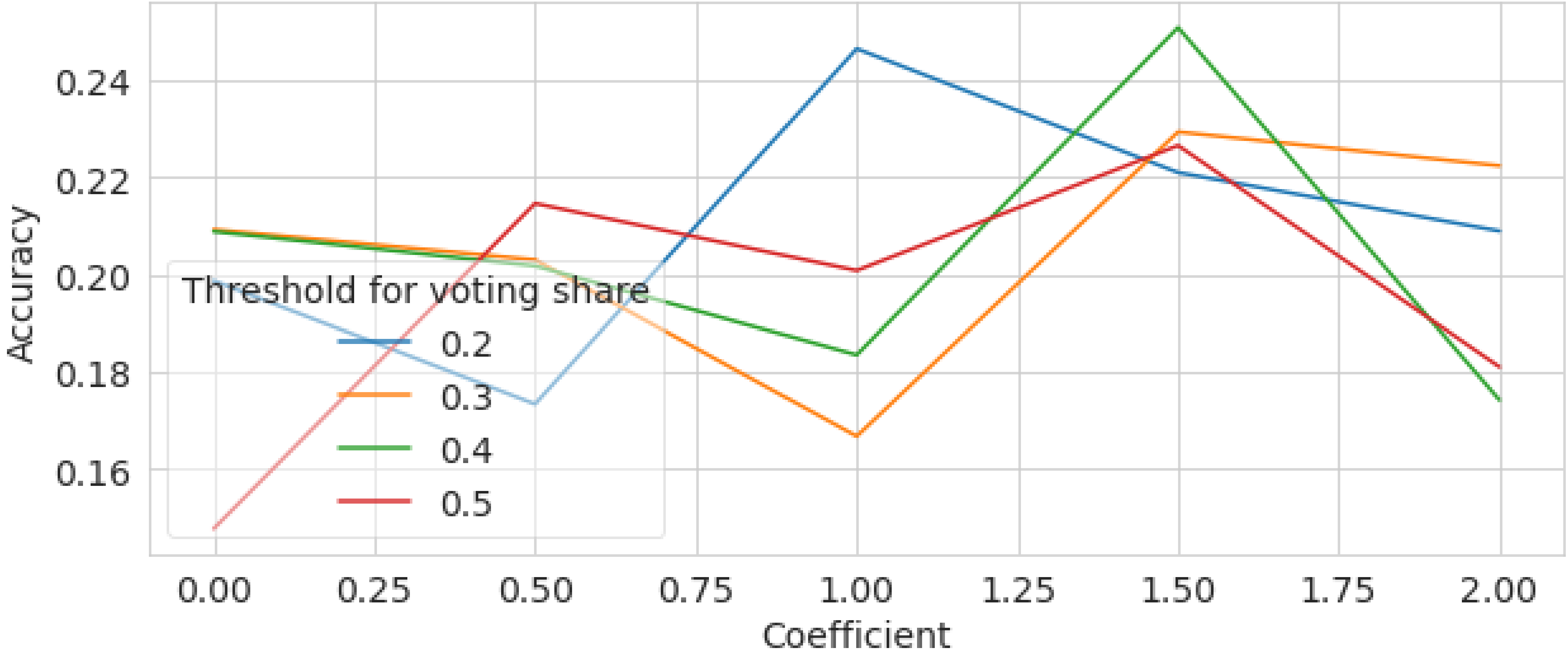


Results: Fixed Activation Threshold, Both Stages

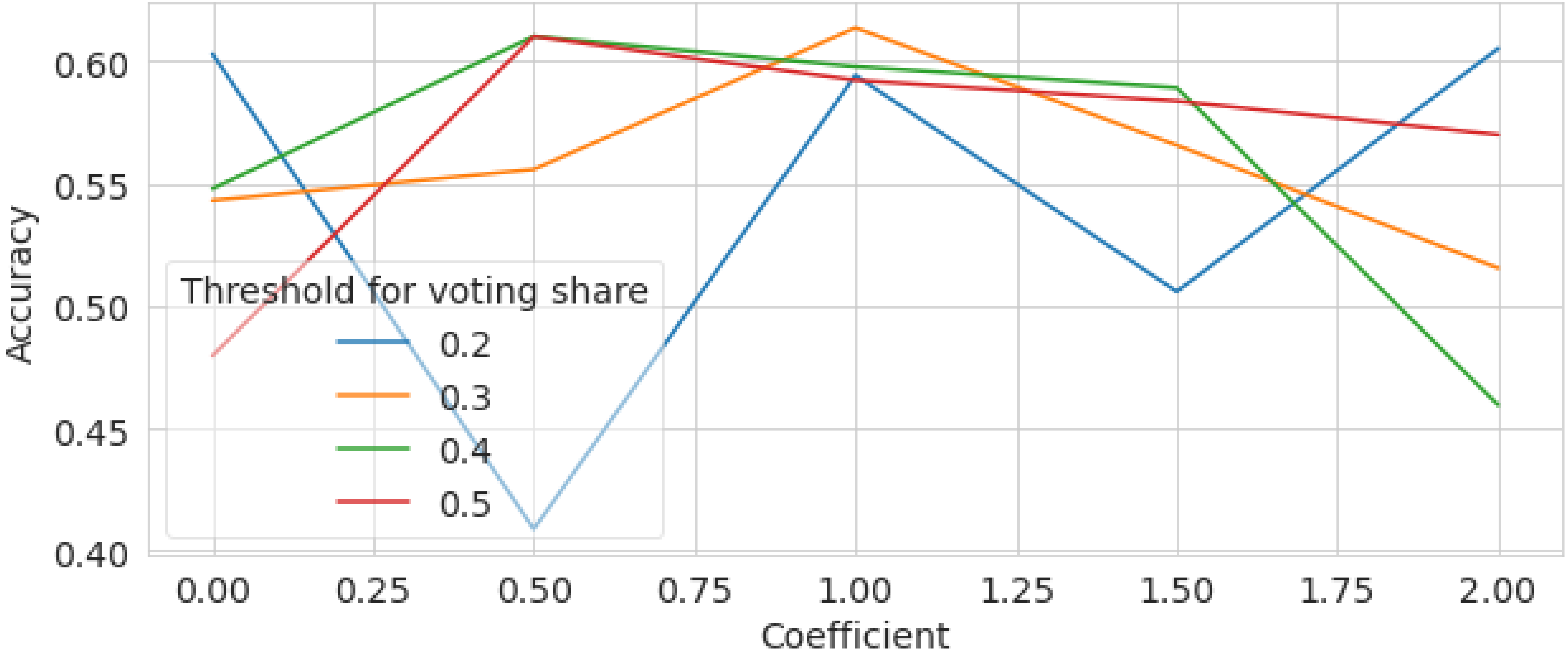


The second stage of the algorithm provides cardinal improvement of the results

Results: Adaptive Activation Threshold, Stage 1 Only



Results: Adaptive Activation Threshold, Both Stages



There is no pronounced dependence of the results on the threshold values

Conclusion

- We have introduced a novel version of the Hierarchical Neural Network Classifier – a **Convolutional Hierarchical Neural Network Classifier**, and an algorithm for its construction.
- The necessity of the second stage of the algorithm has been demonstrated.
- The obtained class hierarchy can be re-used in other algorithms.
- Introduction of two types of threshold demonstrated no pronounced dependence on the threshold values. However, the best results for CIFAR-10 problem were obtained for the two-stage algorithm with adaptive **Activation Threshold** with $k = 1$ and **Voting Patterns Share Threshold** equal to 0.3.
- The algorithm requires further investigation of the optimal methods for setting of the threshold values at the first stage and of the optimal parameters at the second stage
- Testing on other benchmark problems is also required

Thank you for your attention!