TRICKS AND PLUG-INS FOR GRADIENT BOOSTING IN IMAGE CLASSIFICATION

IEEE Big Data 2025 Conference

Author: Biyi Fang, Truong Vo, Jean Utke, Diego Klabjan

Engineering Science and Applied Mathematics
Industrial Engineering and Management Sciences

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Motivation

Key challenges:

- CNNs need large compute and long training cycles
- Architecture search and tuning are expensive
- BoostCNN reduces tuning but is slow and memory heavy

Goal:

- Create a boosting framework that is faster, lighter, and more accurate

Backgrounds

Why combine boosting and CNNs:

- Boosting reduces bias and strengthens weak learners
- CNNs extract hierarchical features

Limitation:

- Standard BoostCNN trains full models each round leading to high cost

Subgrid BoostCNN Algorithm

Core contributions:

- Dynamic subgrid selection that focuses on informative regions
- Reuse of convolutional feature extractors
- Training small weak learners efficiently

Impact:

- Higher accuracy, lower compute, robust to randomness

Importance Index

Definition:

- For each pixel (j,k), compute the gradient magnitude of loss w.r.t. input
- Average across samples and channels

Interpretation:

- High score means model is sensitive to that pixel

Use:

- Remove less important rows and columns to form subgrid

Subgrid Selection Process

Steps:

1. Compute pixel importance
2. Rank importance for rows and columns
3. Remove about 10 percent least important rows and columns
4. Keep about 81 percent of pixels

Benefits:

- Avoids random noise
- Preserves critical structure

Architecture Reuse Strategy

Reuse components:

- Copy feature extractor from previous learner
- Attach new classifier head that matches subgrid size

Advantages:

- Avoids retraining entire CNN
- Works for variable input sizes
- Faster convergence across boosting rounds

Algorithm Summary

Subgrid BoostCNN Algorithm:

1. Train initial full CNN
2. Compute boosting weights
3. Compute importance index
4. Select subgrid and create modified inputs
5. Build weak learner with reused backbone
6. Train on squared loss vs boosting weights
7. Add learner to ensemble with shrinkage factor

Experimental Setup

Datasets:

- CIFAR10, SVHN, ImageNetSub (100 classes)

Architectures:

- ResNet18, ResNet50, ResNet101

Training:

- 10 weak learners
- Each trained for 15 epochs using ADAM

Baselines:

- Single CNN
- BoostCNN
- e-CNN ensemble
- Subgrid e-CNN

Results: Accuracy Improvement

Findings:

- Subgrid BoostCNN consistently achieves the best accuracy
- Up to 12.10 percent improvement over base CNN
- Outperforms BoostCNN by up to 4.19 percent
- Much faster for equal compute budget

Results: Stability

Variance study:

- Subgrid BoostCNN shows small variance across seeds
- Subgrid e-CNN shows much higher fluctuation

Reason:

- Boosting aligns weak learners toward residuals
- Subgrid selection reduces randomness

Comparison with Deeper CNNs

Observation:

- Subgrid BoostCNN with ResNet50 beat single ResNet101 significantly

Insight:

- A strong ensemble is more effective than one deeper model
- Good for large scale problems where deeper models are expensive

Conclusion

Summary:

- Subgrid BoostCNN provides faster and more accurate boosting
- Importance driven pixel selection improves efficiency
- Robust across datasets and architectures

Future directions:

- Adaptive subgrid rates
- Extension to detection and segmentation
- Integration with synthetic data and curriculum design

THANK YOU FOR YOUR ATTENTION

truongvo2025@u.northwestern.edu

Northwestern | McCORMICK SCHOOL OF ENGINEERING