# Convolutional Nearest Neighbors: Reinterpreting Convolution Through K-Nearest Neighbor Selection

Bowdoin

Mingi Kang, Jeova Farias Ph.D.

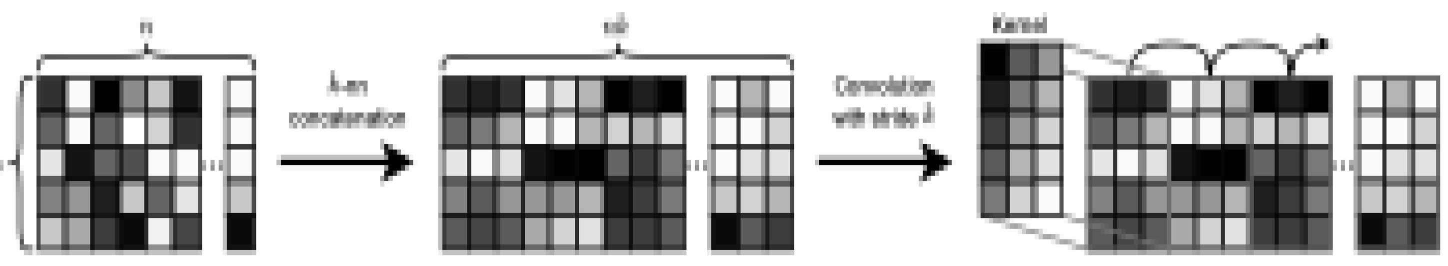
Bowdoin College, ME

## INTRODUCTION

- **Convolutional Nearest Neighbor** (**ConvNN**) reinterprets convolution as k-nearest neighbor aggregation with flexible neighbor selection criteria.
- Standard convolution implicitly performs k-NN with fixed spatial distance (e.g., 3x3 kernel = k = 9 spatially-adjacent neighbors including self).
- ConvNN generalizes this by allowing neighbor selection based on:
  - Spatial distance (reduces to standard convolution)
  - Feature similarity (cosine/Euclidean)
  - Hybrid spatial-feature metrics
- Core Algorithm of ConvNN:
  1. Compute pairwise similarities between all spatial positions
  2. Select k-nearest neighbors per position via hard top-k
  3. Aggregate neighbors with learnable weights (1D convolution)

## BASE ALGORITHM

### ConvNN Visualization




### 1. Similarity Computation

$$S = XX^\top \in \mathbb{R}^{n\times n} \text{ where } S_{ij} = x_i^\top x_j$$

### 2. K-Nearest Neighbor Selection

$$I_k = k\text{-}argmax(XX^\top) \in \mathbb{R}^{n\times n}$$

$$\text{Neighbors} = X[I_k[i,:],:] \in \mathbb{R}^{k\times n}$$

**Algorithm 1** Convolutional Nearest Neighbors 1D

**Input**: $\mathbf{X} \in \mathbb{R}^{B\times C\times N}$ (batch × channels × tokens)
**Parameters**: $k$ (number of neighbors)
**Output**: $\mathbf{Y} \in \mathbb{R}^{B\times C'\times N}$

**// For each batch element**
Let $X = \mathbf{X}[b,:,:]^\top \in \mathbb{R}^{N\times C}$ with columns $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$

**// Step 1: Compute similarity matrix**
Assume each $\mathbf{x}_i$ is $\ell_2$-normalized: $\|\mathbf{x}_i\|_2 = 1$
Compute similarity: $S = XX^\top \in \mathbb{R}^{N\times N}$ where $S_{ij} = \mathbf{x}_i^\top \mathbf{x}_j$

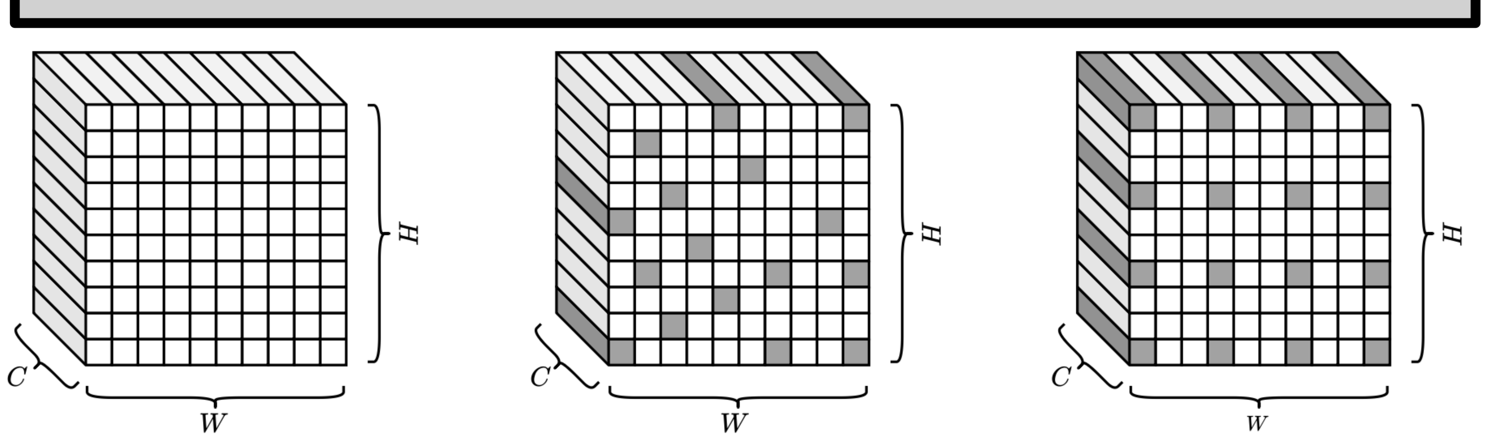
**// Step 2: Find k-nearest neighbors**
$I_k = argmax_k(S) \in \{0,1\}^{N\times N}$

**// Step 3: Gather features**
**for** $i \in [1, N]$ **do**
  $\mathcal{N}_k(\mathbf{x}_i) = X[I_k[i,:],:] \in \mathbb{R}^{k\times C}$
  $\mathbf{V}_{prime}[:,:,i\cdot k:(i+1)\cdot k] = \mathcal{N}_k(\mathbf{x}_i)^\top$
**end for**

**// Step 4: Convolve**
$\mathbf{Y} = \text{Conv1D}(\mathbf{V}_{prime}, \text{kernel\_size} = k, \text{stride} = k)$

**return Y**

## SIMILARITY COMPUTATION SPEED-UPS




(a) Input Tensor (b) After Random Sparsification $n = 16$ (c) After Spatial Sparsification $n = 4$

- To reduce $O(N^2)$ complexity of all to all similarity computation, we introduce two sampling methods: Random and Spatial Sparsification.
- Trade-off between computational efficiency and neighbor selection quality is controlled by sampling parameter $n$ (number of pixel sampled).

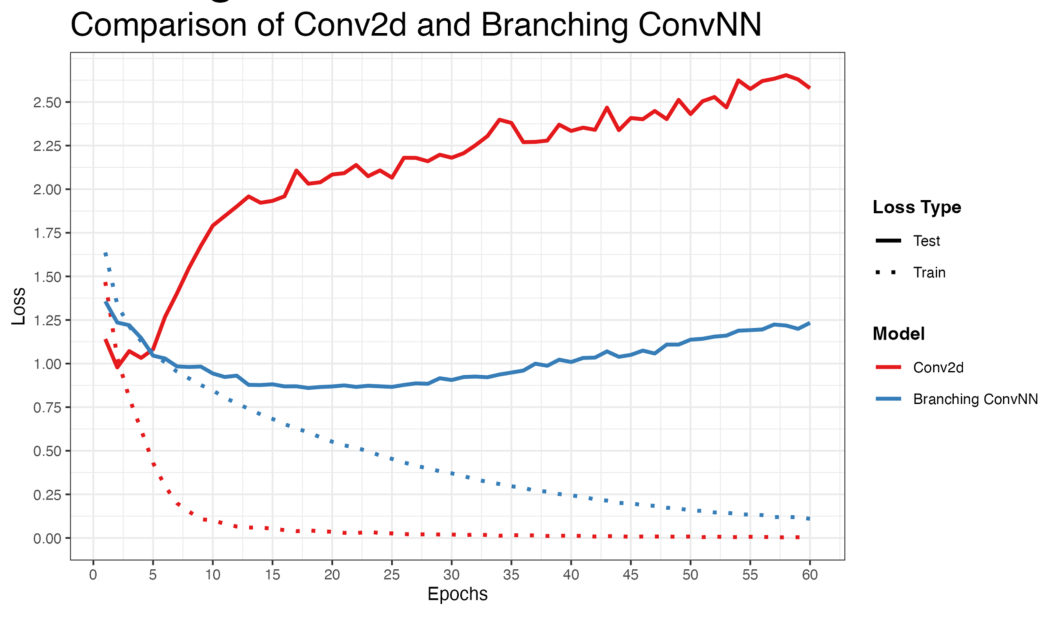
## ARCHITECTURE AND TRAINING

- **Architecture**: VGG-11 with Conv2d layers replaced by ConvNN and branching layers
- **Dataset**: CIFAR-10 image classification
- **Training**: 60 epochs with AdamW (lr=1e-5, wd=1e-6), StepLR scheduler (gamma=0.95, step=2)
- **Variants tested**:
  - Location-only (spatial distance)
  - Feature-only (cosine similarity)
  - Hybrid (weighted combination)
  - Branching with ratio (e.g., 50% Conv2d + 50% ConvNN)

## RESULTS

### Training and Test Loss

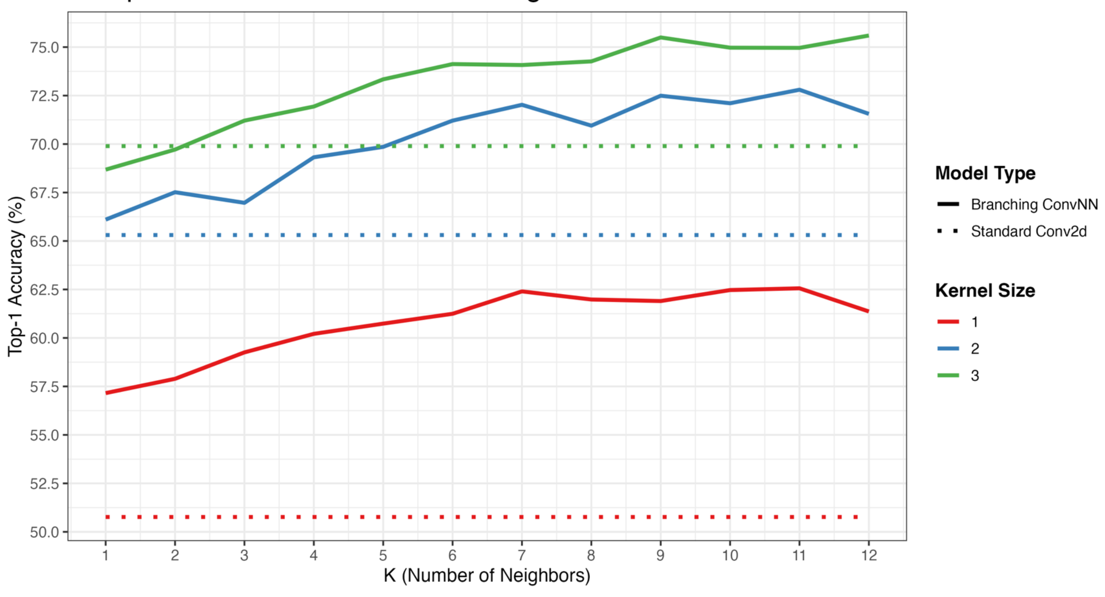
Comparison of Conv2d and Branching ConvNN




Branching ConvNN = Branching with branching ratio 0.500, kernel_size = 3, K = 9, Feature Similarity and Aggregation.

### Model Accuracy by Kernel Size and Type

Comparison of Conv2d and Branching ConvNN




Branching ConvNN = Branching with branching ratio 0.250, Location + Feature Similarity and Aggregation.

Table 1: CIFAR10 ConvNN Branching Ratio (Color Similarity and Color Aggregation)

| Models | Branching Ratio ($\lambda$) | Params | Top-1 Acc. | Test Loss | GFlops |
|---|---|---|---|---|---|
| Conv2d | 0.000 | $130.015M$ | 69.78% | 2.57 | 0.293 |
| Branching | 0.125 | $130.015M$ | **73.49%** | 1.81 | 0.325 |
| Branching | 0.250 | $130.015M$ | **74.32%** | 1.56 | 0.325 |
| Branching | 0.500 | $130.015M$ | **73.61%** | 1.23 | 0.325 |
| Branching | 0.750 | $130.015M$ | 68.63% | 1.23 | 0.325 |
| Branching | 0.875 | $130.015M$ | 65.66% | 1.33 | 0.325 |
| ConvNN | 1.000 | $130.015M$ | 50.250% | 1.84 | 0.325 |

VGG 11 Architecture with kernel_size = 3 (Conv2d), K = 9 (ConvNN)
Branching Models: $\lambda \times \text{ConvNN} + (1-\lambda) \times \text{Conv2d}$

Table 2: CIFAR10 ConvNN Branching Ratio (Location + Color Similarity and Color Aggregation)

| Models | Branching Ratio ($\lambda$) | Params | Top-1 Acc. | Test Loss | GFlops |
|---|---|---|---|---|---|
| Conv2d | 0.000 | $130.015M$ | 69.78% | 2.57 | 0.293 |
| Branching | 0.125 | $130.015M$ | **72.92%** | 1.92 | 0.331 |
| Branching | 0.250 | $130.015M$ | **74.20%** | 1.52 | 0.331 |
| Branching | 0.500 | $130.015M$ | **73.16%** | 1.24 | 0.331 |
| Branching | 0.750 | $130.015M$ | **69.98%** | 1.22 | 0.331 |
| Branching | 0.875 | $130.015M$ | 64.77% | 1.33 | 0.331 |
| ConvNN | 1.000 | $130.015M$ | 52.70% | 1.80 | 0.331 |

VGG 11 Architecture with kernel_size = 3 (Conv2d), K = 9 (ConvNN)
Branching Models: $\lambda \times \text{ConvNN} + (1-\lambda) \times \text{Conv2d}$

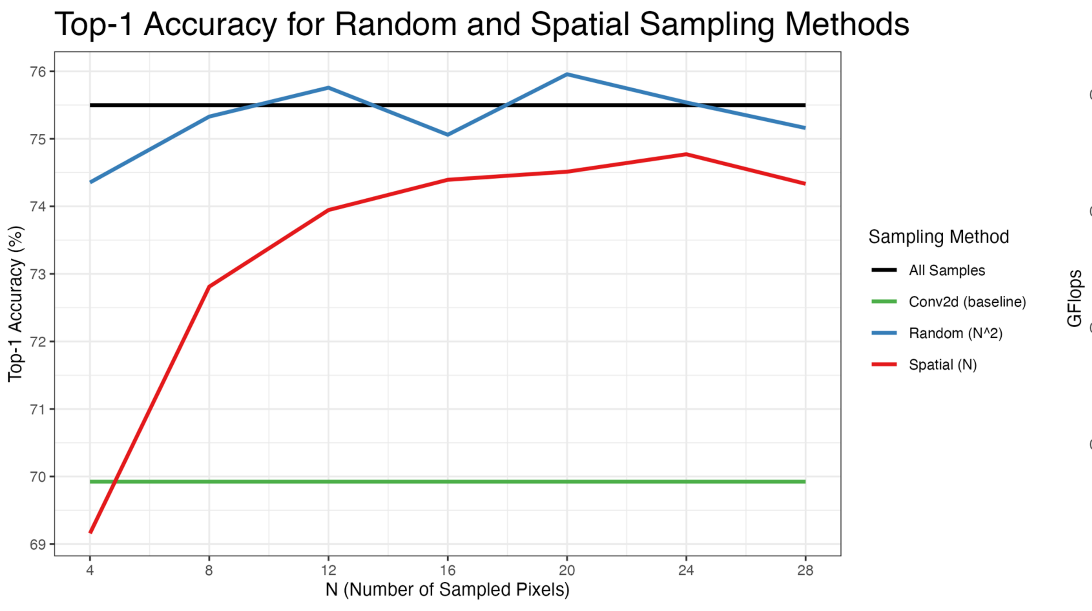
Table 3: CIFAR10 ConvNN Branching Ratio (Location + Color Similarity and Location + Color Aggregation)

| Models | Branching Ratio ($\lambda$) | Params | Top-1 Acc. | Test Loss | GFlops |
|---|---|---|---|---|---|
| Conv2d | 0.000 | $130.015M$ | 69.78% | 2.57 | 0.293 |
| Branching | 0.125 | $130.021M$ | **73.75%** | 1.85 | 0.331 |
| Branching | 0.250 | $130.028M$ | **75.22%** | 1.46 | 0.331 |
| Branching | 0.500 | $130.040M$ | **74.52%** | 1.17 | 0.331 |
| Branching | 0.750 | $130.052M$ | 69.49% | 1.15 | 0.331 |
| Branching | 0.875 | $130.059M$ | 66.14% | 1.25 | 0.325 |
| ConvNN | 1.000 | $130.065M$ | 60.09% | 1.44 | 0.325 |

VGG 11 Architecture with kernel_size = 3 (Conv2d), K = 9 (ConvNN)
Branching Models: $\lambda \times \text{ConvNN} + (1-\lambda) \times \text{Conv2d}$

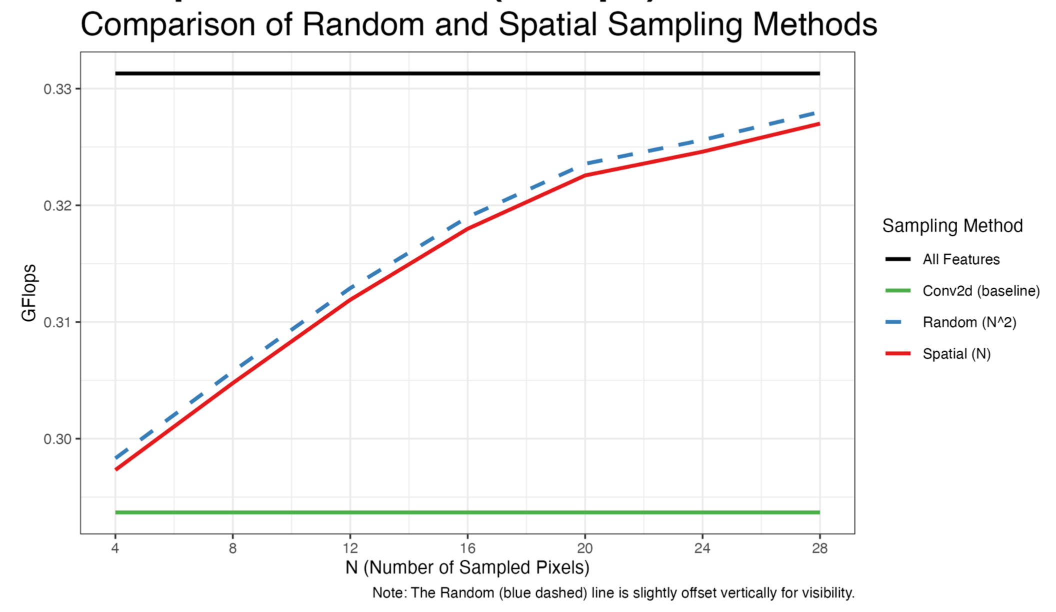
### Model Performance vs. N

Top-1 Accuracy for Random and Spatial Sampling Methods




### Computational Cost (GFlops) vs. N

Comparison of Random and Spatial Sampling Methods




Branching ConvNN = Branching with branching ratio 0.250, Location + Feature Similarity and Aggregation. Spatial Sampling = $N = N \times N$ sub grid 3, Random Sampling = $N^2$ pixels.

## CONVOLUTION AND ATTENTION

### 1. Convolution

$$S = D = 2(1 - X^\top X) \in \mathbb{R}^{n\times n} \text{ where } D_{ij} = \| x_i - x_j \|_2^2 = 2(1 - x_i^\top x_j)$$

$$I_k = k\text{-}argmax(2(1 - X^\top X)) \in \mathbb{R}^{n\times n}$$

$$\text{Neighbors} = X[I_k[i,:],:] \in \mathbb{R}^{k\times n}$$

### 2. Convolutional Nearest Neighbor

$$S = XX^\top \in \mathbb{R}^{n\times n} \text{ where } S_{ij} = x_i^\top x_j$$

$$I_k = k\text{-}argmax(XX^\top) \in \mathbb{R}^{n\times n}$$

$$\text{Neighbors} = X[I_k[i,:],:] \in \mathbb{R}^{k\times n}$$

### 3. Attention

$$QK^\top \in \mathbb{R}^{n\times n} \text{ where } Q = w_Q X,\ K = w_k X$$

$$A(Q,K) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) \in \mathbb{R}^{n\times n}$$

$$\text{Attention}(Q,K,V) = A(Q,K)V \text{ where } V = w_v X$$

## DISCUSSION

- **Hybrid similarity** (spatial + feature) outperforms pure spatial or pure feature selection
- **Branching architecture** achieves best performance by combining ConvNN's global context with Conv2d's spatial locality.
- ConvNN unifies convolution and attention as neighbor aggregation differ:
  - Spatial-only → standard convolution
  - All positions with soft weights with linear projection → self-attention
  - ConvNN occupies the middle ground with hard, content-aware selection
- **Feature work**: Extend to Vision Transformers, explore learnable similarity metrics, investigate soft vs. hard selection.

## REFERENCES

- A. Buades, B. Coll and J. . -M. Morel, "A non-local algorithm for image denoising," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, pp. 60-65 vol. 2, doi: 10.1109/CVPR.2005.38.
- Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).Plötz, Tobias, and Stefan Roth. "Neural nearest neighbors networks." *Advances in Neural information processing systems* 31 (2018).
- Wang, Xiaolong, et al. "Non-local neural networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
- Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).