Road Traffic Sign Classification under Challenging Conditions

Weixi FENG, UG; Prof. Kenneth Lam, Supervisor

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

- Introduction & Related work
- Proposed work
  - Convolutional Neural Network
  - Random Forests
- Image pre-processing
- Experiments
- Conclusions

Introduction

- Traffic Sign Classification is important for Automatic Driving System

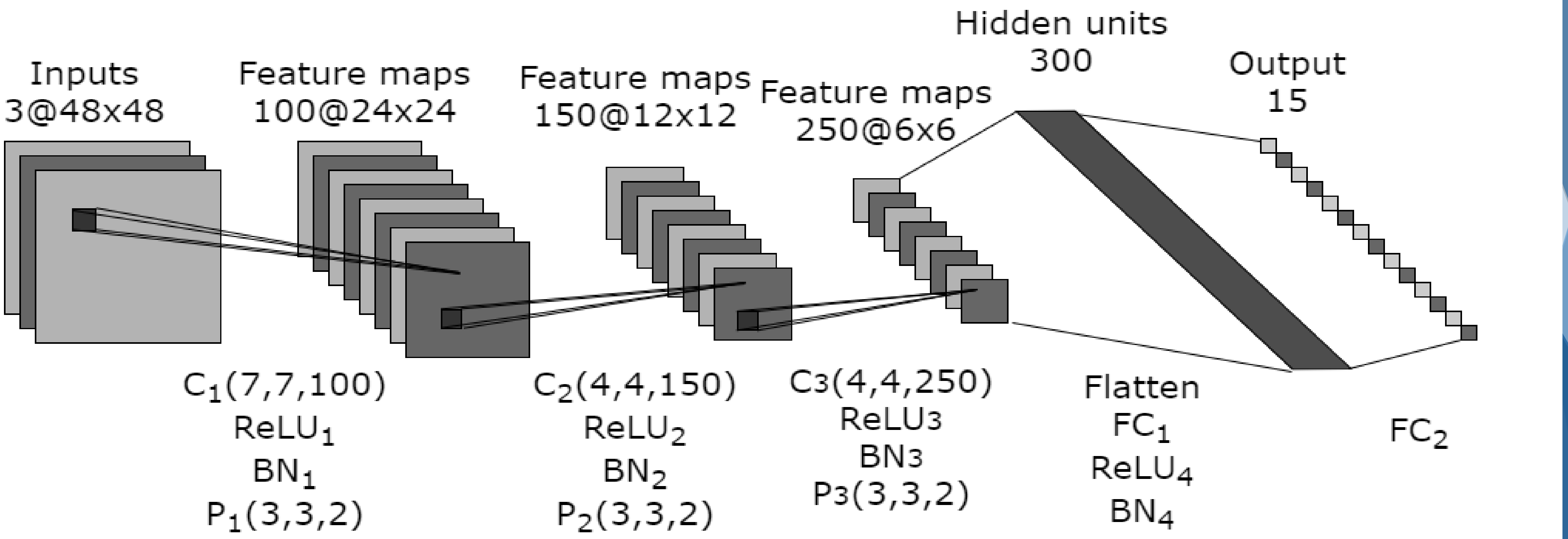
- Existing datasets: GTSRB, BelgiumTS etc.
  - do not include challenging conditions or
  - fail to emphasize the relationship between poor conditions and algorithmic performances.

Related work

- Some of the works were based on conventional classifiers such as Support Vector Machine.

- Conventional methods didn't perform as well as Deep Learning based method. Recent CNN-based methods include
  - ensemble of CNNs
  - multi-scale CNN
  - Much deeper networks

Proposed work



$C(w, w, k)$: Convolutional layer with k filters of size $w \times w$
ReLU: ReLU activation layer
BN: Batch Normalization layer
$P(w, s)$: Max-pooling layer with kernel size w and stride s

Proposed network – Batch Normalization

- Usually, we must be very careful in parameter initialization and learning rate selection.
- With Batch normalization we could
  - prevent overfitting[4].
  - prevent internal covariate shift
  - set a large learning rate at the beginning.

Input: values of x over minibatch: $\mathrm{B} = \{x_{1\ldots m}\}$

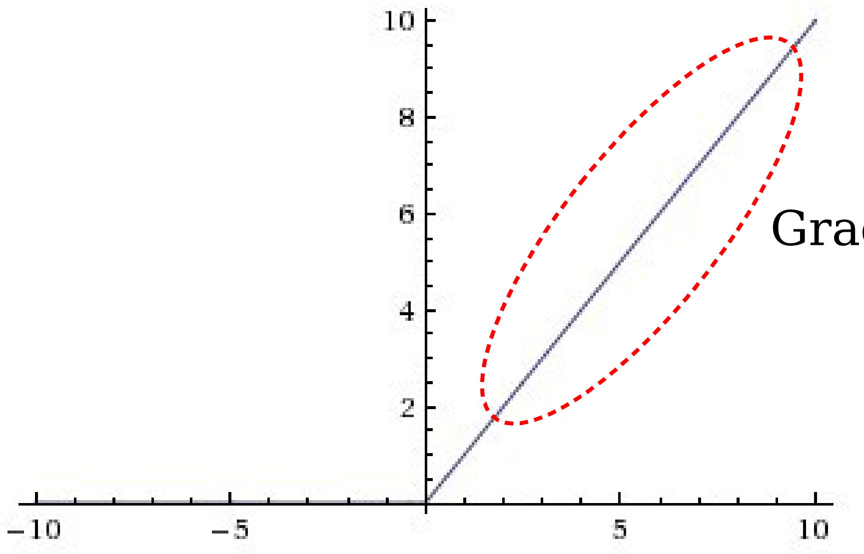
Output:

$$\widehat{x_i} = \frac{x_i - \mu_{\mathrm{B}}}{\sqrt{\sigma_{\mathrm{B}}^2 + \epsilon}} \qquad \text{normalize}$$

$$y_i = \gamma \widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i) \qquad \text{scale and shift}$$

Proposed network – Activation function

- The advantages of using ReLU, instead of tanh
  - ReLU can increase the sparsity
  - Solve the ‘vanishing gradient’ problem [5]
  - Save computation costs



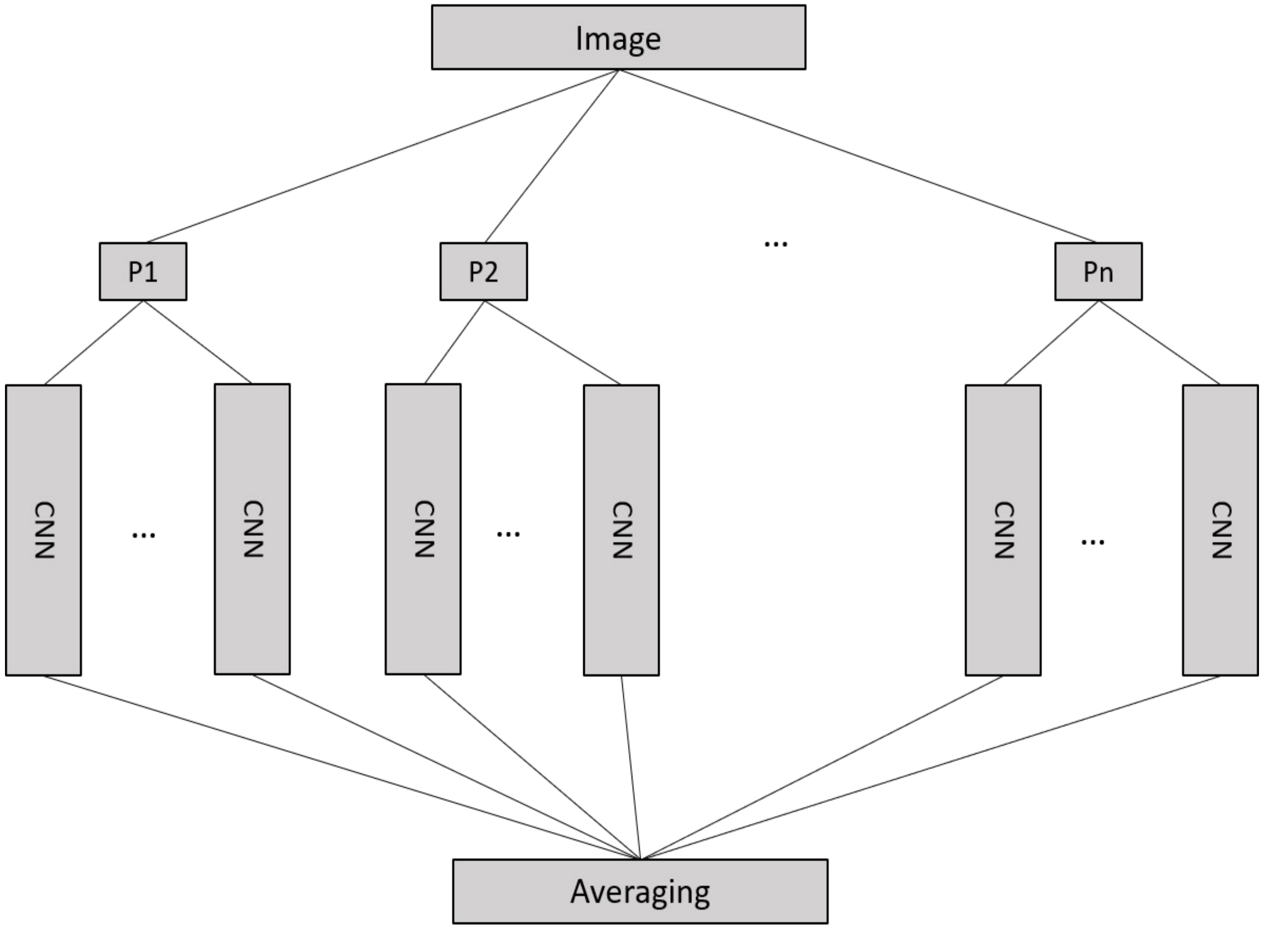
Gradient always equals to 1

Proposed work - Ensemble of CNNs

- **Theory**:
  - If the errors of P different models have zero mean and are uncorrelated, the average error might be reduced by a factor of P simply by averaging the P models[7].

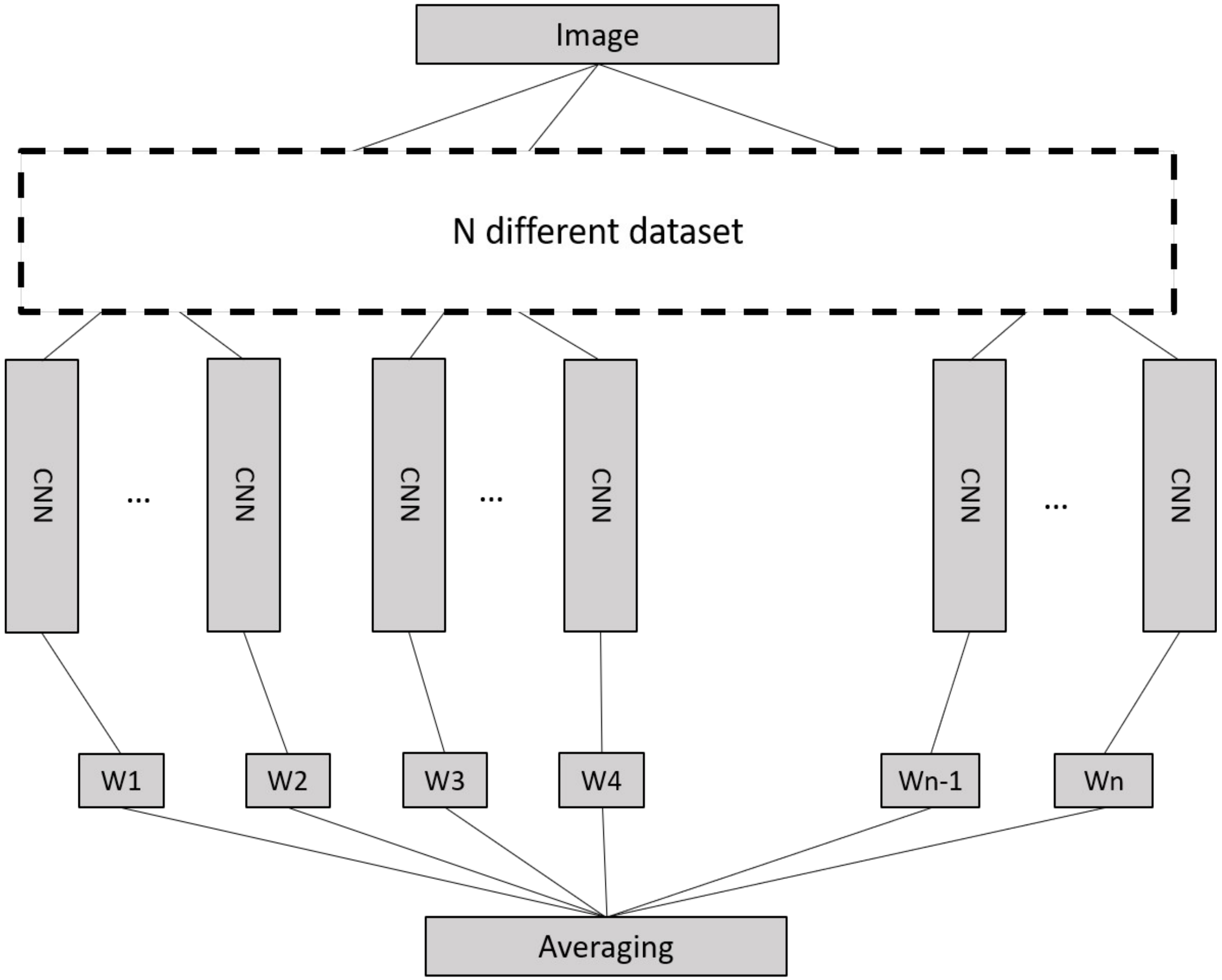
- **Problems to think:**
  - In practice, errors of models trained on similar data tend to be highly correlated.
  - Whether to optimize the combination of outputs of various models or not?
  - Additional Data?
  - Highly generalized to the unseen testing data?

Ensembles of CNNs



[2] D. Cireşan, U. Meier, J. Masci and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification", *Neural Networks*, vol. 32, pp. 333-338, 2012.

Proposed work – Ensembles of CNNs



$$E = \sum_{k=1}^{K} \left\| y_i - \sum_{i=1}^{5} w_i \vec{S}_i \right\|$$

Proposed work – Random Forests

Image Preprocesing

- RGB to grayscale
- CLAHE

Feature Extraction

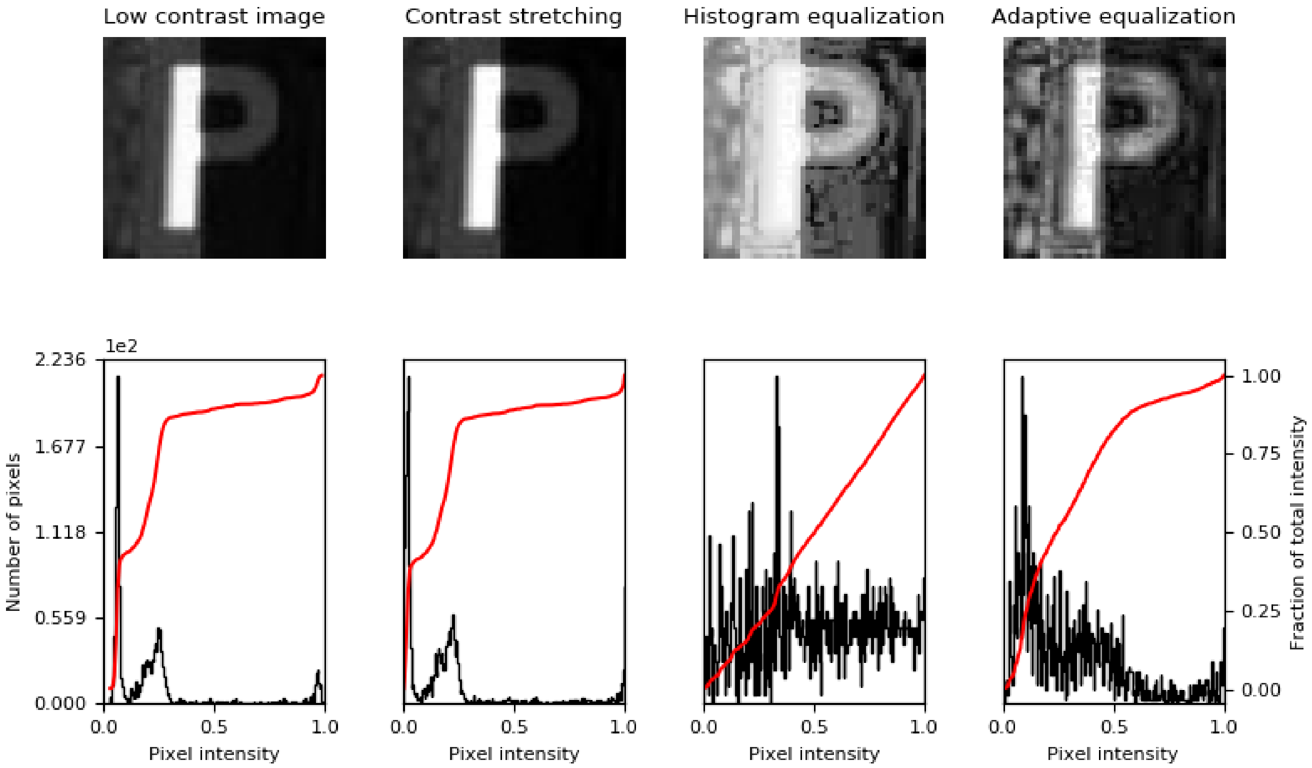
- HoGs
- Concatenated Features

Classification

- Decision Tree
- Random Forests

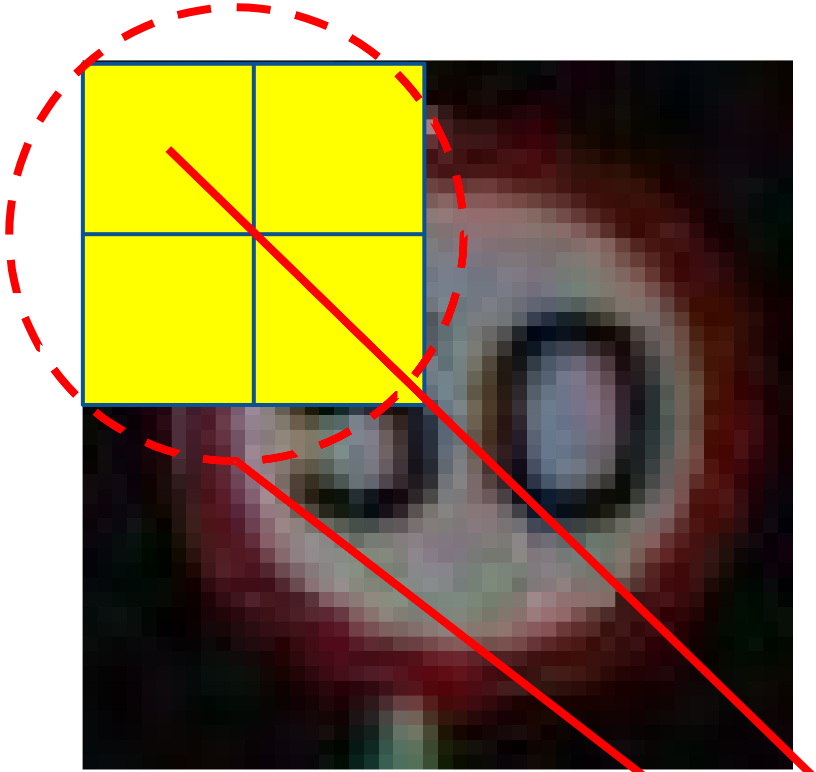
Contrast-Limited Adaptive Histogram Equalization(CLAHE)

- Global Histogram Equalization uses the histogram information of the whole image.
  - Cannot adapt the local contrast information
- CLAHE transforms a pixel value from a neighbourhood region.



Feature Extraction - HoGs

- Histogram of Oriented Gradients(HoGs) is good for describing shape and appearance. It extracts local gradient information.



1. Apply $D_x = [-1,0,1]$ and $D_y = [-1,0,1]^T$ to the entire Image. Compute $\sqrt{I_x^2 + I_y^2}$ and $\tan^{-1}\left(\frac{I_y}{I_x}\right)$ to get the gradient information at each pixel.
2. Divide image into blocks and cells. Form one histogram for each block.
3. Concatenate block histograms into one feature vector

1. Apply and to the entire Image. Compute and to get the gradient information at each pixel
2. Divide image into blocks and cells. Form one histogram for each block.
3. Concatenate block histograms into one feature vector

Feature Extraction - HoGs

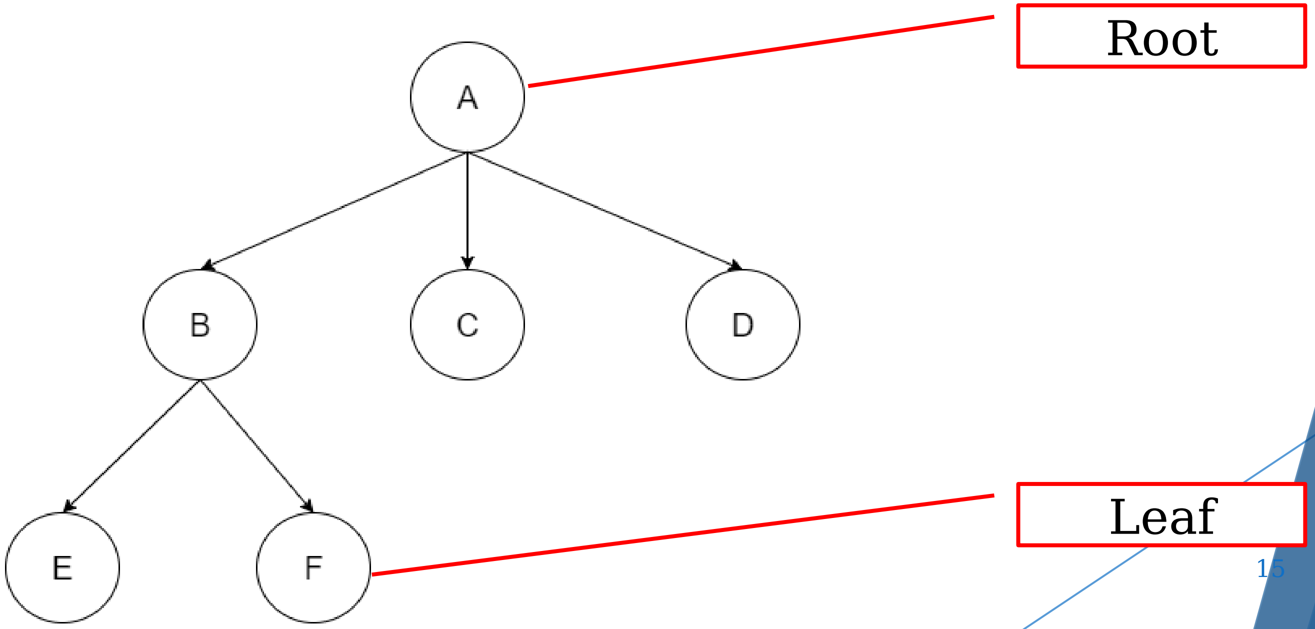
Tricks:

1. Normalize local histogram will give better results

2. Overlapping between neighbouring blocks significantly Increase the effectiveness of HoG

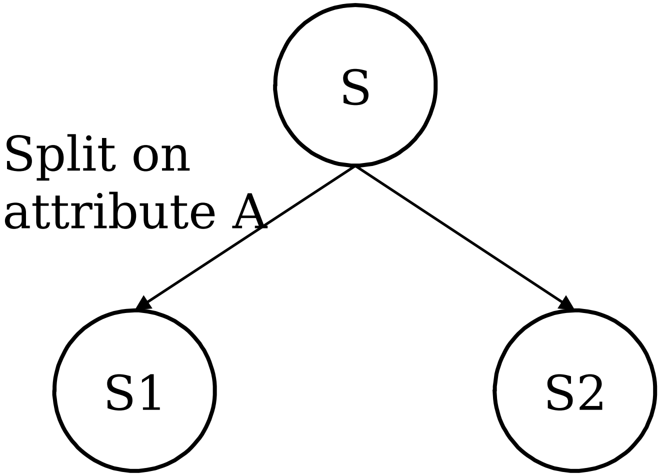
Random Forests – Decision Tree

- Tree structure is widely used for better management of data.
- It is also an effective tool for classification and detection
- How to build such a tree?
  - ID3 & C4.5 - Entropy
  - **Classification And Regression Tree(CART) - Gini**



CART algorithm

- Input dataset S, each data has M dimensions(attributes).
- **Concept**: CART algorithm splits each node into two child nodes so that the data is as "pure" as possible
  - Gini impurity
- **Steps**:
  - Each split considers attributes that has not been used.
  - Choose one attribute that gives the largest Gini gain.
  - Stop when Gini equals to zero or pre-set rules are satisfied.



$$Gini(S) = 1 - \sum_{x \in X} p^2(x)$$

$$Gain(S, A) = Gini(S) - \sum_{S_i \subseteq S} p(S_i) Gini(S_i)$$

Random Forests

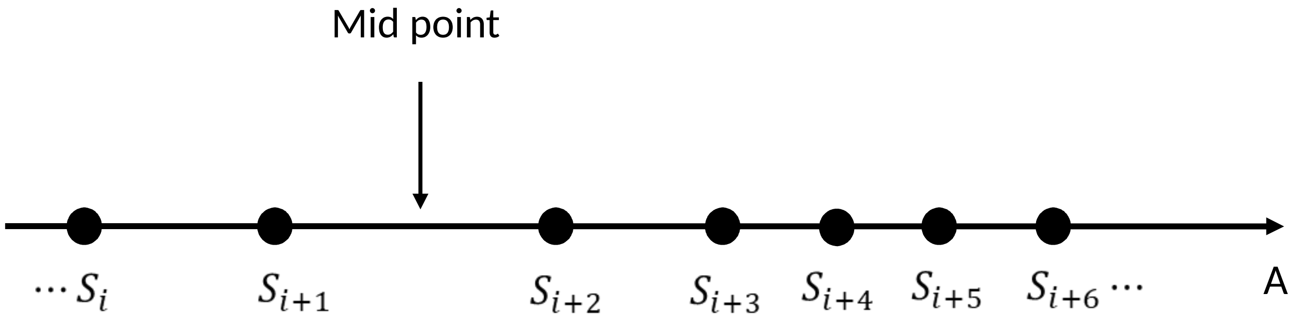
- Random Forests combines a number of decision trees into one classifier
- Applies CART-like algorithm
- Step:
  - Bootstrap aggregating
  - Train a decision tree
  - Repeat the above steps until we have enough trees

Random Forests – Bootstrap aggregating

- Assume we have dataset $S = (s_1, y_1), (s_2, y_2), \cdots, (s_N, y_N)$ where y represents a label.
- Bootstrap aggregating draw one sample from dataset S with replacement. Repeat this to get N samples. These N samples form a new training set.
- Advantages:
  - Reducing variance
  - Decrease correlation between decision trees.
  - Without increasing the bias.

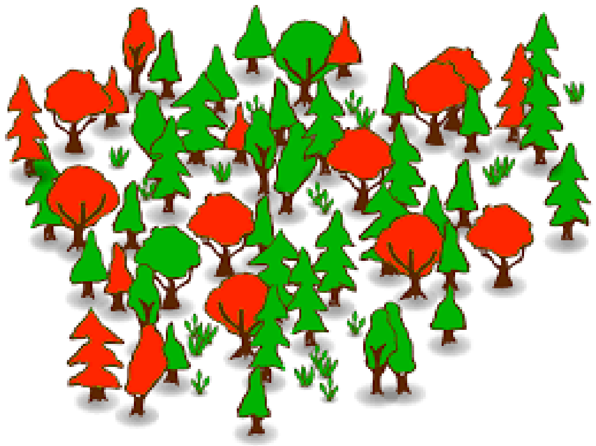
Random Forests – Training

- **Concept:**
  - CART-like algorithm
  - Random features
- **Steps**:
  - Calculate Gini at present node
  - Randomly select m attributes from M total attributes. Consider these m attributes.
  - Select the attribute that gives the largest Gini gain.
  - Split on the attribute(dimension)



Random Forests

- Random selection of training samples
- Random features
- Reason:
  - Different training samples decorrelate the trees.
  - Dominate features will increase correlation
  - Randomize the features to avoid repeatedly using dominate features
  - Further decorrelate trees



Random Forests

- Technical considerations
  - Number of trees
  - Number of features
  - Overfitting: limit depth, number of samples etc.
  - Class weights: unbalanced dataset
  - Out of bag error

Dataset

- German Traffic Sign Recognition Benchmark
  - The German Traffic Sign Recognition Benchmark(GTSRB) is an official dataset available years ago. It includes 39,209 training images in 43 classes and 12,630 testing images.

Dataset

- The VIP Cup Dataset has:
  - 49 real scenes + 49 unreal scenes
  - 14 types of traffic signs
  - 12 challenging conditions
  - 5 levels of severity

level 1 level 2 level 3 level 4
level 5

0
0

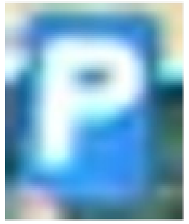
01 02 03 04 05
06

07 08 09 10 11
12

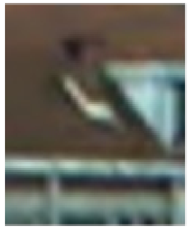
00-no challenge	07-gaussian blur
01-decolorization	08-noise
02-lens blur	09-rain
03-codec error	10-shadow
04-darkening	11-snow
05-dirty lens	12-haze
06-exposure	

Data preparation

Images larger than 50*50



Extremely small traffic signs



Negative samples

Around 35,000 images

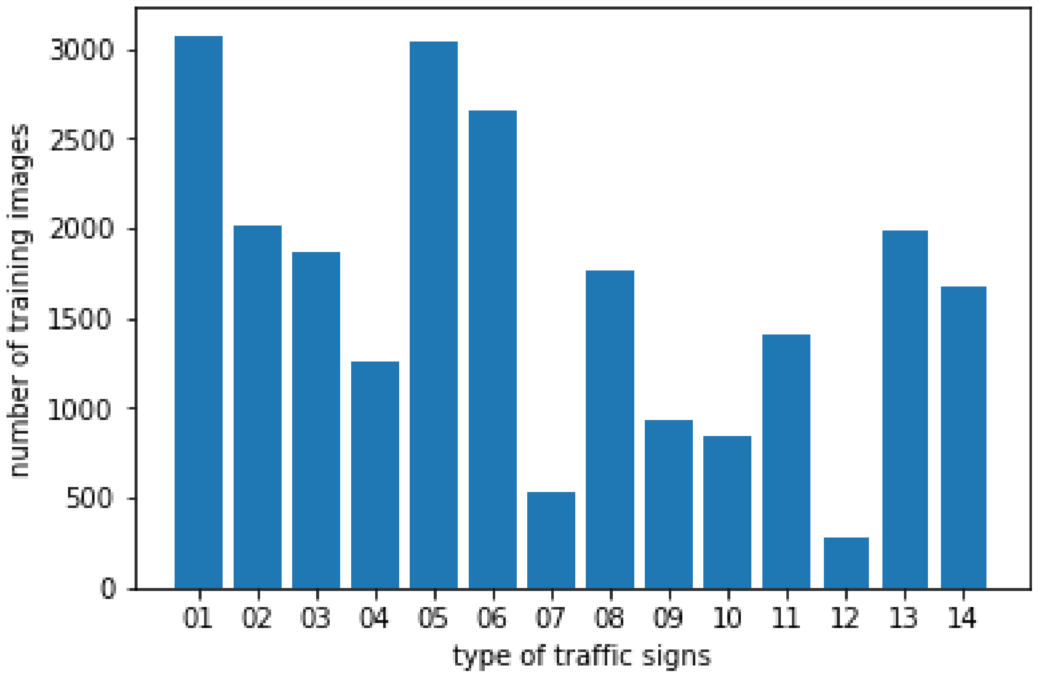
Training set(80%)

Validation set(20%)

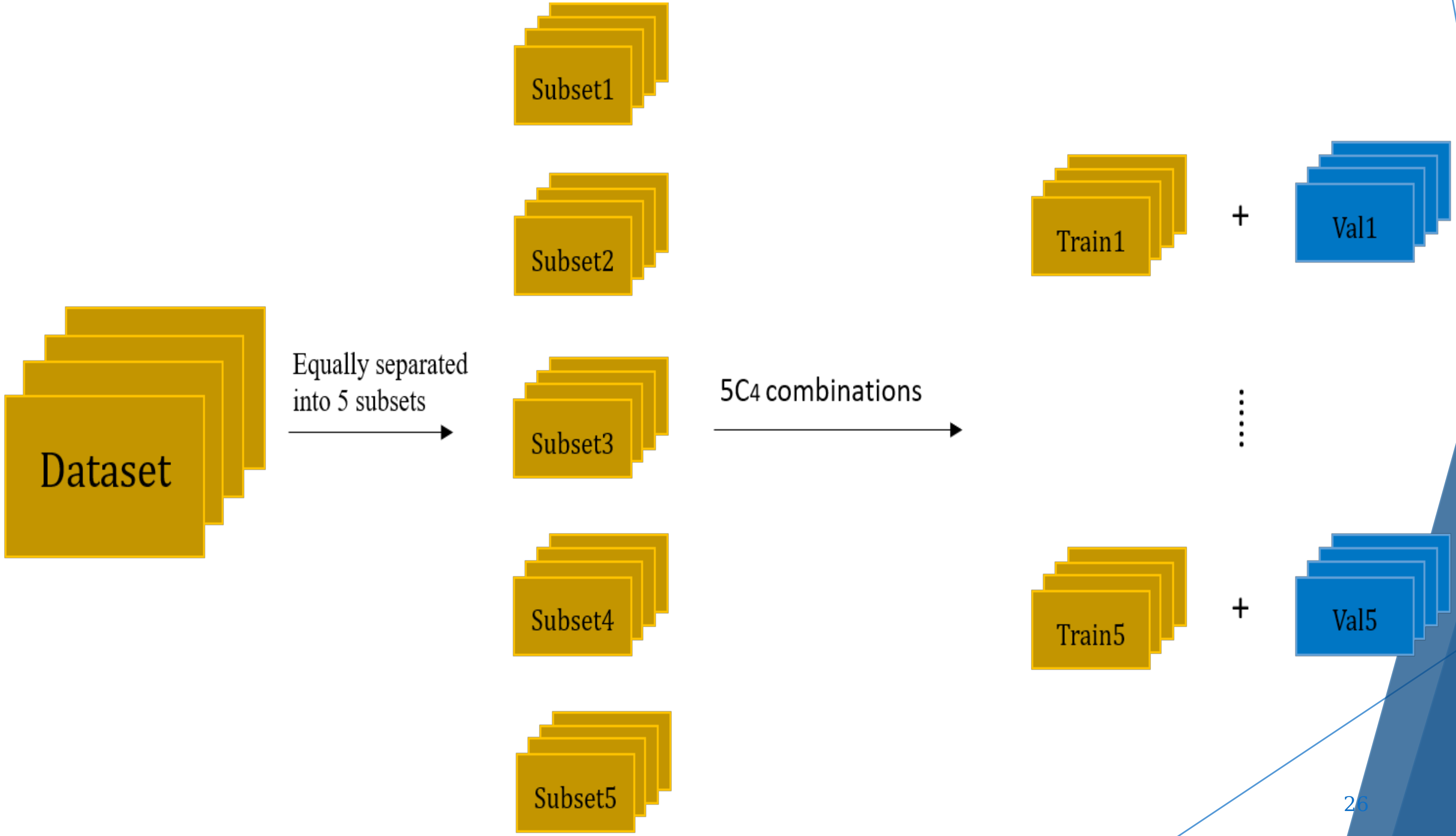
Data augmentation for ConvNet

- Compensation on type 07 and 12

- Random crop
- Random saturation
- Random contrast
- Vertical flip



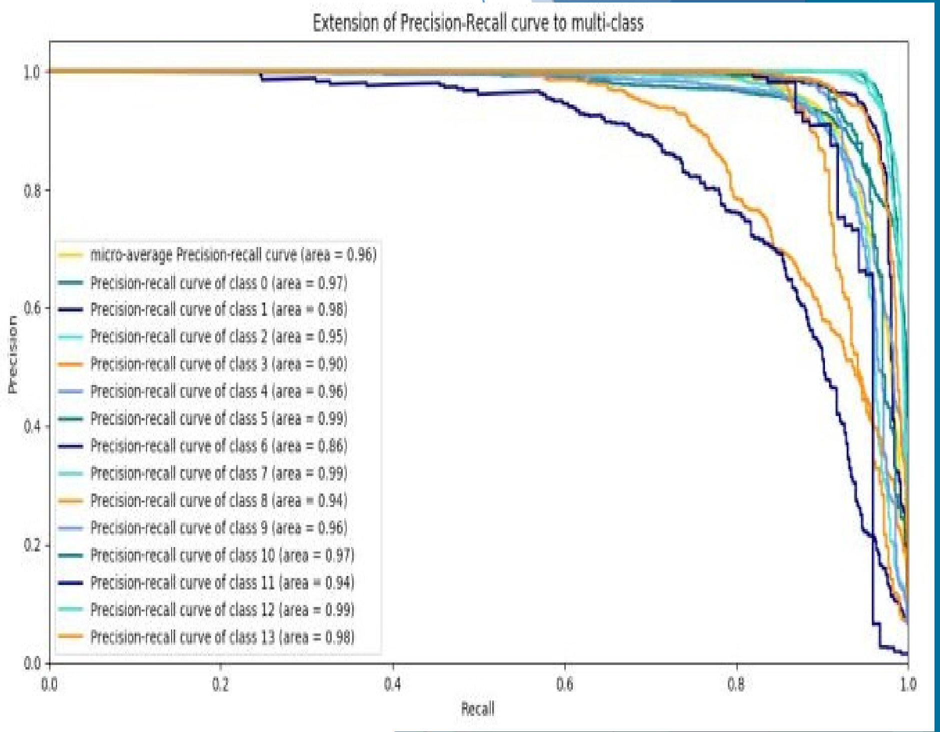
Data preparation for ConvNet



Experimental results

- Running our trained network on 12159 testing images. We gained an average accuracy 92.21%

Challenge Type	Accuracy(%)	Challenge Type	Accuracy(%)
No challenge	99.19	Gaussian blur	91.41
Decolorization	96.99	Noise	96.73
Lens blur	94.07	Rain	92.27
Codec error	59.71	Shadow	97.59
Darkening	96.48	Snow	95.62
Dirty lens	98.36	Haze	95.19
Exposure	93.04	Overall	92.21



- We found that codec error (challenge 03) is the most influential condition on our network. Around 47% of the misclassified images belong to the codec-error challenge, among which 74% are higher than level 2. Figure 2 shows some images of the codec-error challenge, and some images belonging to other challenges.

Experimental results

In order to show the effect of batch normalization and ReLU in this experiment. We tried different combinations of activation function and batch normalization. The experiment results are shown below.

Combination	ReLU only	tanh + BN	ReLU + BN
Accuracy	77.53%	83.38%	92.21%

Experimental results

Dataset	Including negative samples		
Network	Single CNN	Ensemble – Averaging	Ensemble – linear combination
Accuracy	92.21%	92.94%	93.13%

Dataset	Excluding negative samples		
Network	Single CNN	Ensemble – Averaging	Ensemble – linear combination
Accuracy	94.10%	95.17%	95.19%

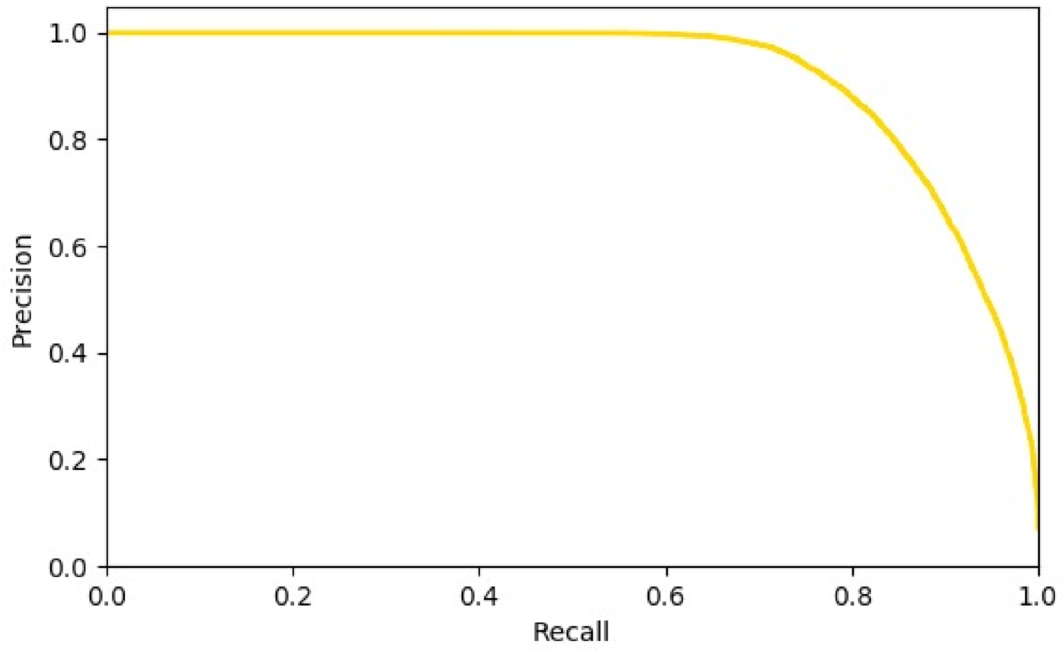
Random Forests on GTSRB

Name	Dimension	Cell Size	Block Size	Overlap	Bin	Orientation
HoG1	1568	5x5	10x10	50%	8	Unsigned
HoG2	1568	5x5	10x10	50%	8	Signed
HoG2	2916	4x4	8x8	50%	9	Unsigned

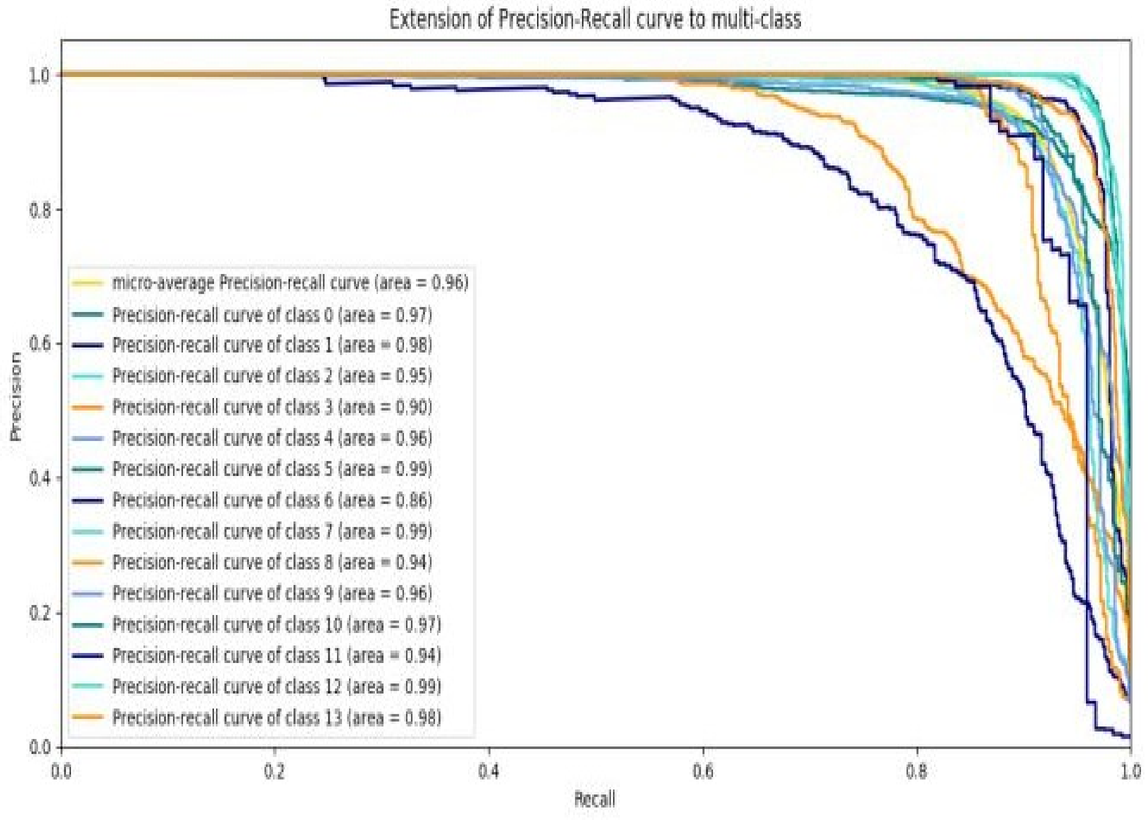
HoG	Number of Features	Number of Trees	Classification Accuracy(%)
2	Log2	50	93.91
		100	94.93
		500	95.20
	sqrt	50	95.68
		100	96.15
		500	96.77
	100	50	95.66
		100	96.28
		500	**96.92**

Random Forests on VIP Cup

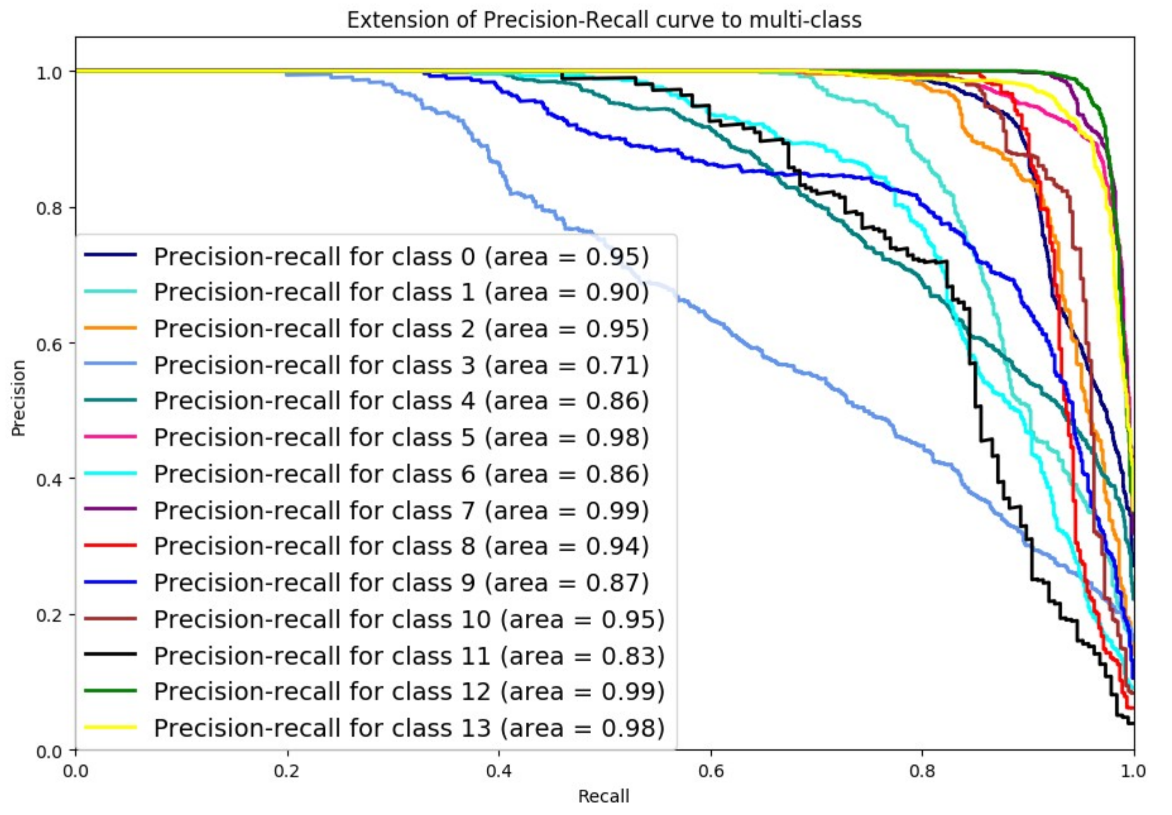
HoG block	Cells/block	nbin	No. of Trees	No. of Features	Accuracy
6x6	3x3	8	500	Sqrt	83.55
8x8	2x2	8	500	Sqrt	81.50
Concatenate HoGs			500	100	**84.43**

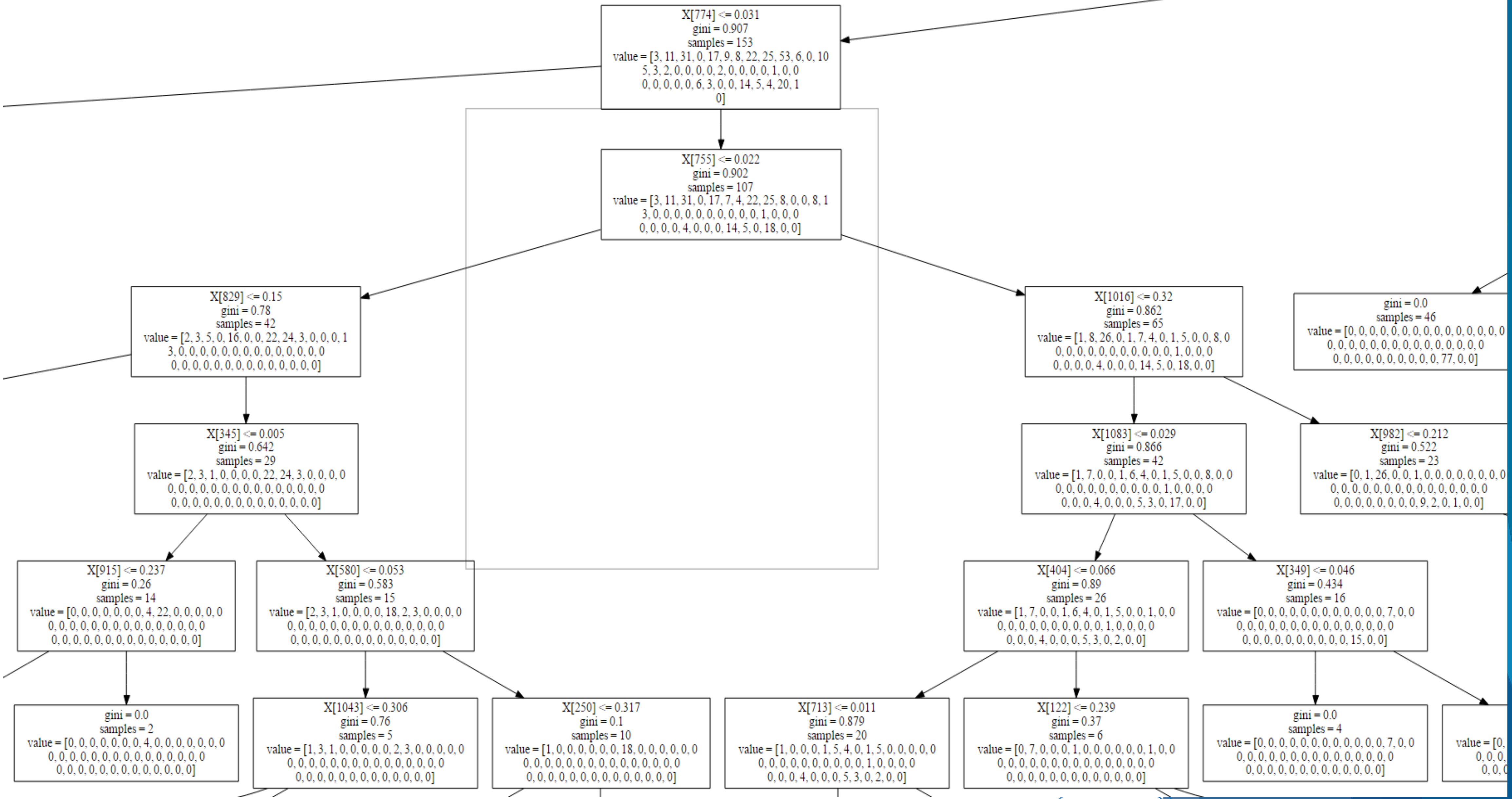


Convolutional Neural Network



Random Forests




X[774] <= 0.031
gini = 0.907
samples = 153
value = [3, 11, 31, 0, 17, 9, 8, 22, 25, 53, 6, 0, 10
5, 3, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 6, 3, 0, 0, 14, 5, 4, 20, 1
0]
X[755] <= 0.022
gini = 0.902
samples = 107
value = [3, 11, 31, 0, 17, 7, 4, 22, 25, 8, 0, 0, 8, 1
3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 4, 0, 0, 0, 14, 5, 0, 18, 0, 0]
X[829] <= 0.15
gini = 0.78
samples = 42
value = [2, 3, 5, 0, 16, 0, 0, 22, 24, 3, 0, 0, 0, 1
3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[1016] <= 0.32
gini = 0.862
samples = 65
value = [1, 8, 26, 0, 1, 7, 4, 0, 1, 5, 0, 0, 8, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0
0, 0, 0, 0, 4, 0, 0, 0, 14, 5, 0, 18, 0, 0]
gini = 0.0
samples = 46
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 77, 0, 0]
X[345] <= 0.005
gini = 0.642
samples = 29
value = [2, 3, 1, 0, 0, 0, 0, 22, 24, 3, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[1083] <= 0.029
gini = 0.866
samples = 42
value = [1, 7, 0, 0, 1, 6, 4, 0, 1, 5, 0, 0, 8, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0
0, 0, 0, 4, 0, 0, 0, 5, 3, 0, 17, 0, 0]
X[982] <= 0.212
gini = 0.522
samples = 23
value = [0, 1, 26, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 9, 2, 0, 1, 0, 0]
X[915] <= 0.237
gini = 0.26
samples = 14
value = [0, 0, 0, 0, 0, 0, 0, 4, 22, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[580] <= 0.053
gini = 0.583
samples = 15
value = [2, 3, 1, 0, 0, 0, 0, 18, 2, 3, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[404] <= 0.066
gini = 0.89
samples = 26
value = [1, 7, 0, 0, 1, 6, 4, 0, 1, 5, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0
0, 0, 0, 4, 0, 0, 0, 5, 3, 0, 2, 0, 0]
X[349] <= 0.046
gini = 0.434
samples = 16
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 15, 0, 0]
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[1043] <= 0.306
gini = 0.76
samples = 5
value = [1, 3, 1, 0, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[250] <= 0.317
gini = 0.1
samples = 10
value = [1, 0, 0, 0, 0, 0, 0, 18, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
X[713] <= 0.011
gini = 0.879
samples = 20
value = [1, 0, 0, 0, 1, 5, 4, 0, 1, 5, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0
0, 0, 0, 4, 0, 0, 0, 5, 3, 0, 2, 0, 0]
X[122] <= 0.239
gini = 0.37
samples = 6
value = [0, 7, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
gini = 0.0
samples = 4
value = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Convolutional Neural Network	Random Forests
Adv: More accurate Robust to all kinds of noises mingled together Disadv: High computational costs Requires a large amount of data Hardware requirements	Adv: Accurate Less computational costs Handle both small and large dataset Disadv: Less robust to various noises(partially because of the dataset) Less accurate than ConvNet

Conclusion

- Convolutional Neural Network performs well in classifying traffic signs. Even though images are distorted with different kinds of noise, we could still reach satisfactory results with the help of batch normalization etc.
- Creating Ensemble of ConvNets does not really increase the accuracy a lot.
  - Around 1% increase
  - Strong, highly correlated classifiers. No randomized steps.
- Random Forests performs well on GTSRB but not as well as ConvNet on VIP Cup dataset.
- A huge boost from single tree to forests
  - ~6% - > ~84%
- Can Deep learning replace Random Forests?

References

[1] J. Greenhalgh and M. Mirmehdi, “Real-time detection and recognition of road traffic signs,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1498–1506, Dec. 2012.

[2] D. Cireşan, U. Meier, J. Masci and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification", *Neural Networks*, vol. 32, pp. 333-338, 2012.

[3] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale Convolutional Networks," *The 2011 International Joint Conference on Neural Networks*, San Jose, CA, 2011, pp. 2809-2813.

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[5] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In ICML, 2013

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. ICLR, 2015.

[7] Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.