Detection of electronic components for mobile applications

Tomasz Buczek, Maciej Wielgosz, Michał Karwatowski, Marcin Pietroń, Kazimierz Wiatr

AGH UST, Faculty of Computer Science, Electronics and Telecommunication

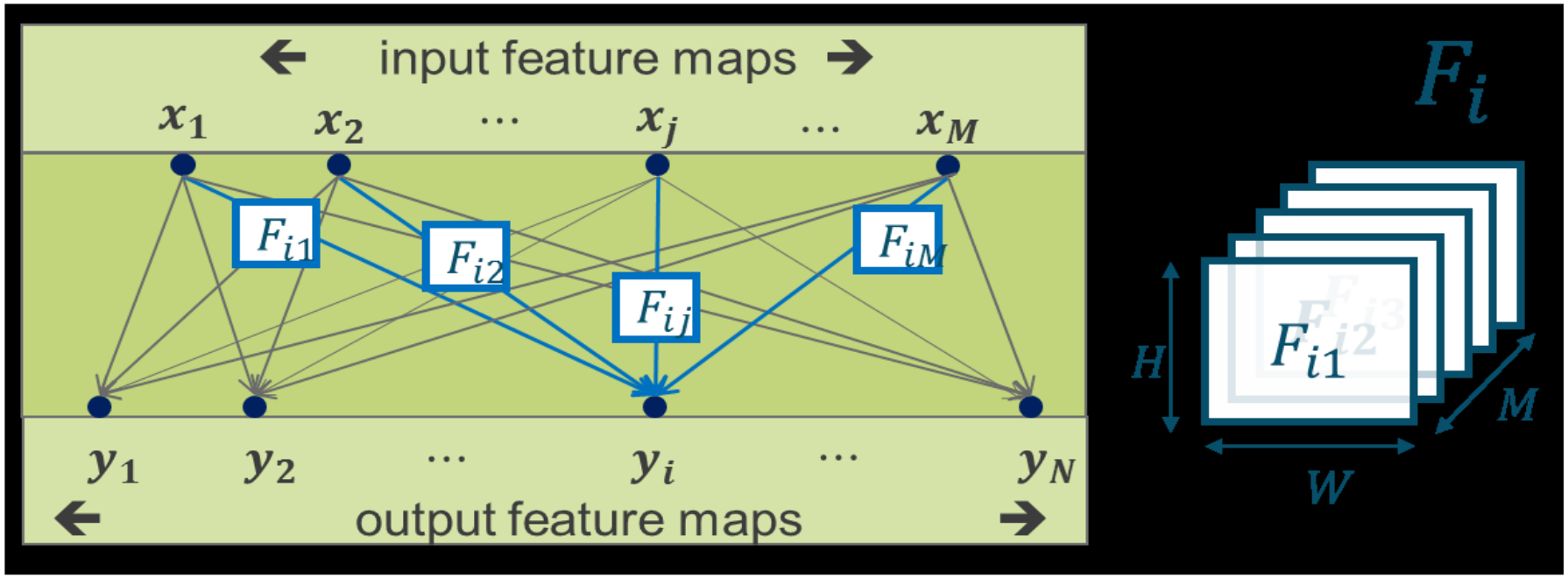
Agenda

1. CNN architecture
2. Object detection networks
3. Experiments on resistors detection
4. Compression of detection networks

Object detection

- Production lines in factories
- Self driving car
- Drones
- Humanoid robots
- etc.

CNN architectures



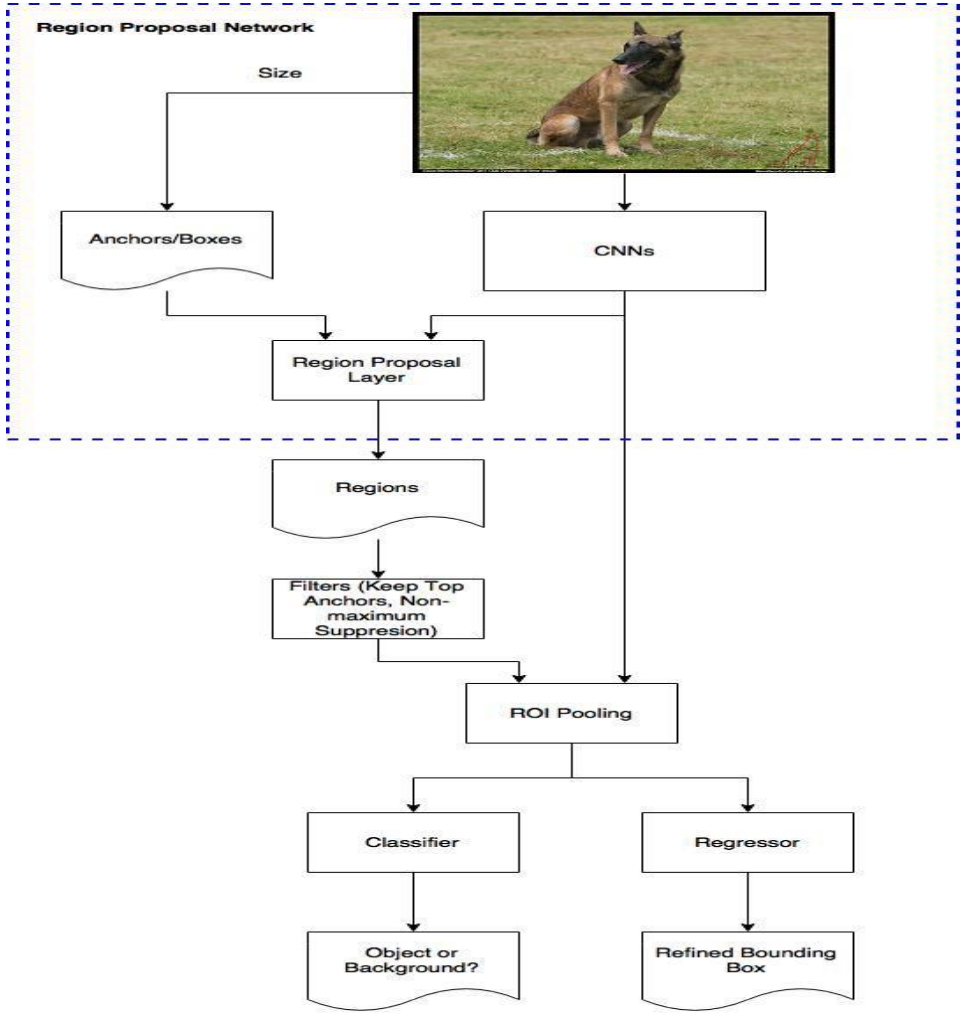
CNN architectures

- Alexnet, Zfnet
- VGG-16, VGG-19
- Resnet
- Inception
- Inception resnet
- Nasnet

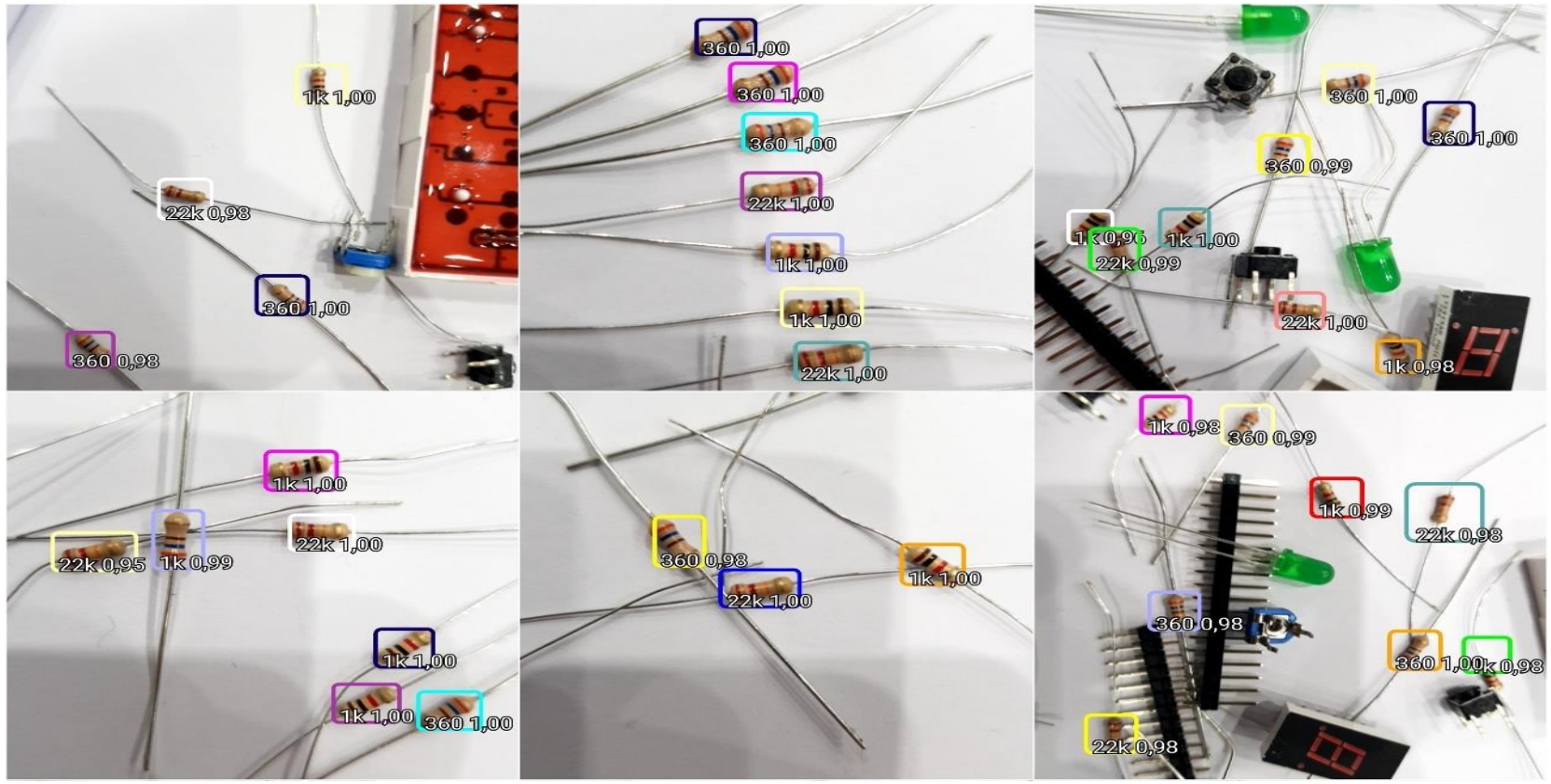
Object detection networks

- r-cnn
- fast and faster r-cnn
- yolo
- ssd
- mask r-cnn

Object detection architectures



Experiments



Experiments

network model	N classes	N training steps	N epochs	mAP	mAP@ .50IoU	total loss	avg. inference time pre-quantization	avg. inference time post-quantization	porteable to Android
faster_rcnn_inception_v2	1	3000	48	**0.554**	**0.972**	**0.200**	-	-	no
ssd_mobilenet_v1_coco	1	3000	48	0.243	0.687	4.749	175ms	130ms	yes
ssd_mobilenet_v1_coco	3	3000	110	0.048	0.226	9.704	-	121ms	yes
ssd_mobilenet_v1_coco	3	12000	440	0.507	0.970	3.143	-	**120ms**	yes

Compression of object detection networks

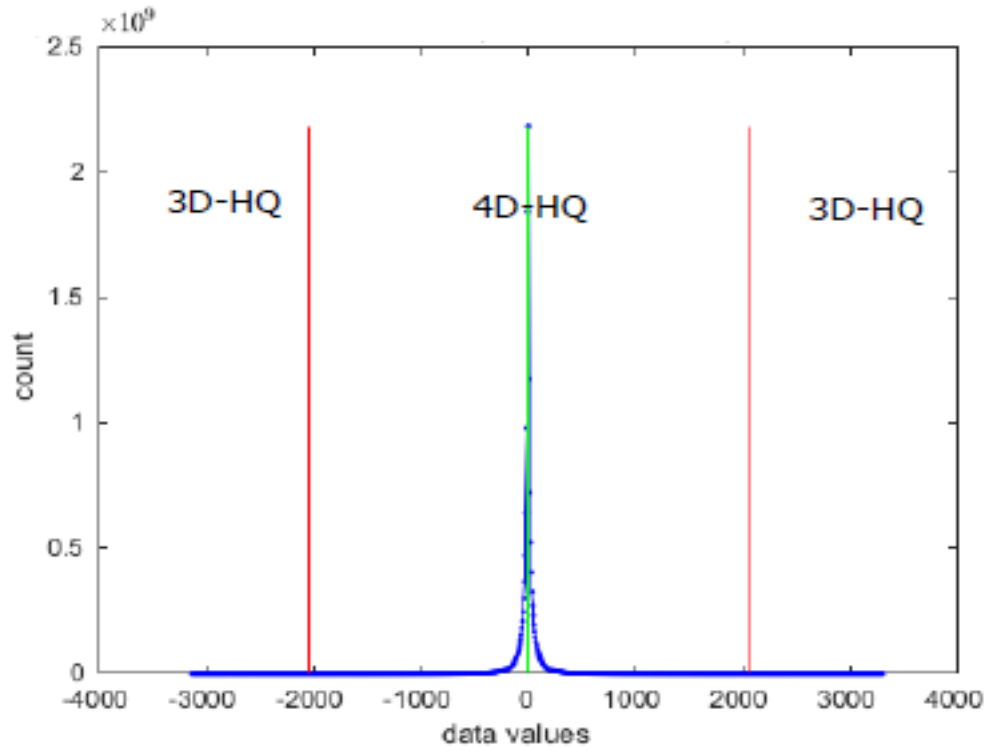
Format $< \text{coeff}, \text{data} >$	Relative Complexity [M MACs]
$< 8b, 8b >$	$224 = HPWMN$
$< 4b, 8b >$	$112 = \frac{1}{2} < 8b, 8b >$
$< 4b, 4b >$	$56 = \frac{1}{4} < 8b, 8b >$
$< \text{flp}, \text{flp} >$	$\geq 2016 = 9 < 8b, 8b >$
$N = 256, P = 27^2, M = 48, H = W = 5$	

Compression of object detection networks

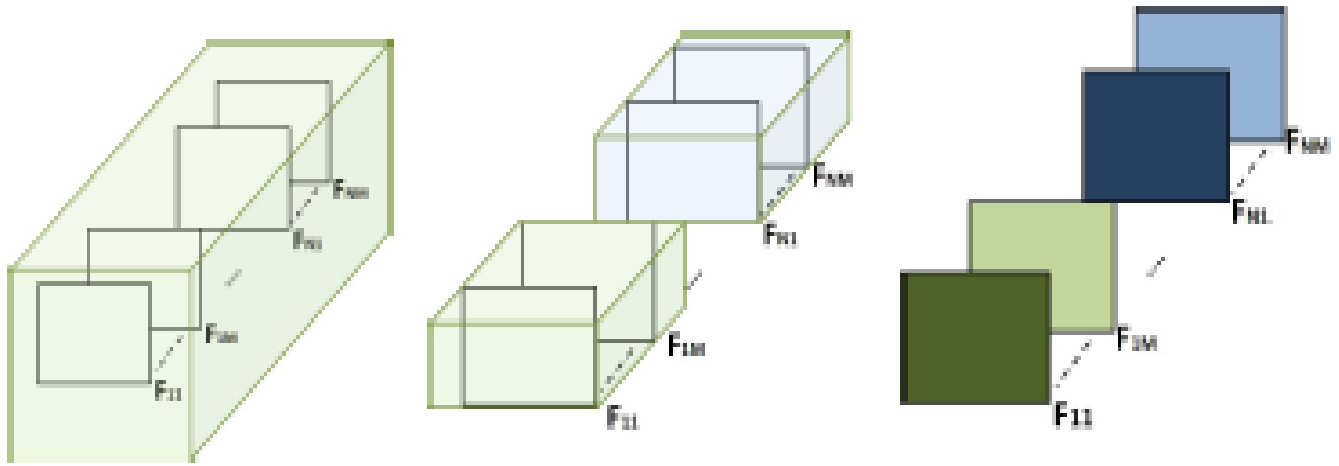
- Quantization
- Pruning
- K-means clustering
- Dimensionality reduction of filters
- All of these methods with and without retraining

Quantization - activations sparsity

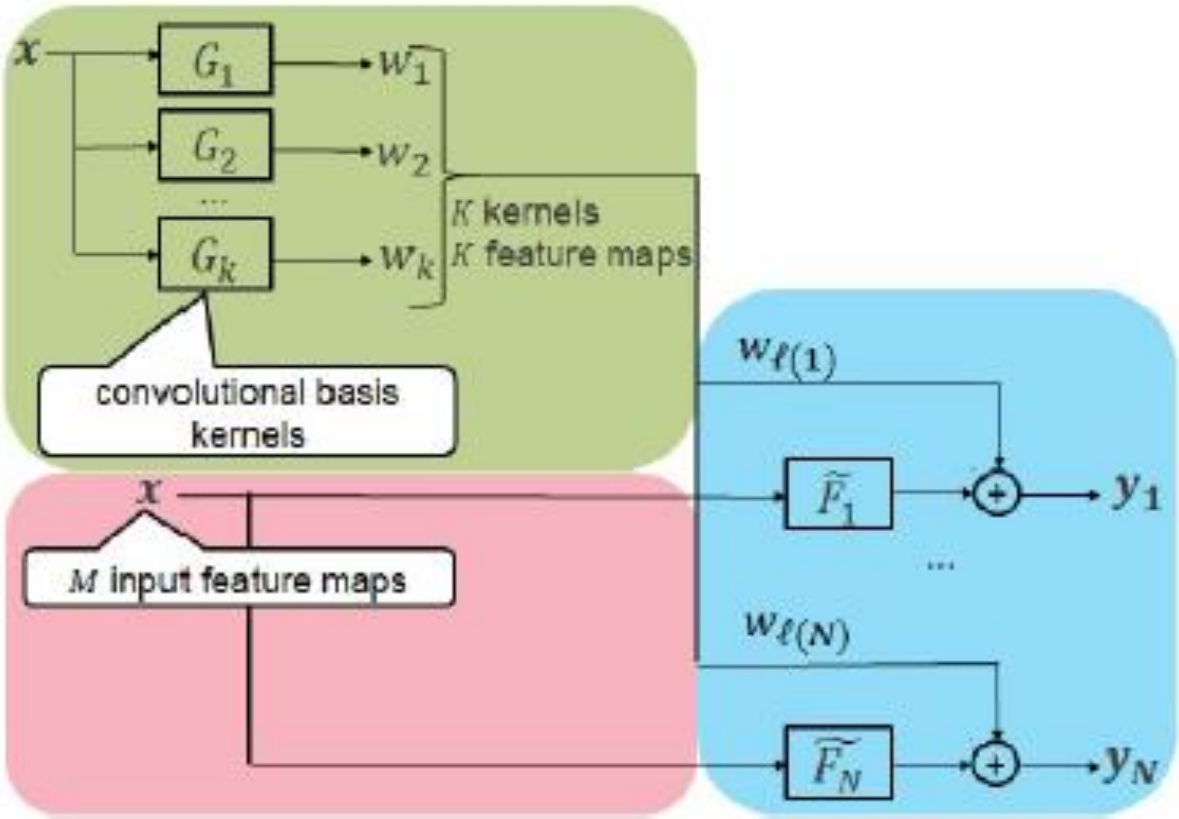
- histogram analysis



Quantization - 3D, 4D



Quantization - K-means clustering



Pruning

Algorithm 1 Pruning algorithm based on random hill climbing

```
1: Input: number_of_buckets_to_which_hist_is_divided
2: Input: drop_in_accuracy_threshold
3: compute_hist_coeff_of_all_layers
4: layer = random_or_prioritized_choose_layer_for_pruning()
5: for number_of_iterations do
6:     if (top_1-baseline) < drop_in_accuracy_thresh then
7:         prune_next_bucket_from_hist()
8:     else
9:         reverse_pruning_bucket_from_hist()
10:    end if
11:    fitness = compute_new_fitness()
12:    top1 = compute_accuracy()
13:    if fitness < best_fitness then
14:        next_solution = current_solution
15:    else
16:        next_solution = best_solution
17:    end if
18:    layer_sensitivity_update
19: end for
```

Pruning

- high sparsity – sparse matrix cudnn operations with conv layers
- in case of fully connected sparse matrix multiplication
- fully connected up to 80-90% sparsity
- Yolo_v2 - set of constant bounding boxes (**y**ou **o**nly **l**ook **o**nce)
- Less the 1% in mAP on COCO and VOC Pascal with 8 bit precision
- about 25-30% of average sparsity, weighted close to 50%

Reinforcement learning for compression

- Q - learning
- Actor critic (policy gradient with Q learning)
- use for choosing layers for pruning, steps and other parameters
- choosing layers for compression
- combination - compression, pruning, quantization

GAN (Adversarial networks) for small object detection