HSE University – N. Novgorod

SEQUENTIAL ANALYSIS IN IMAGE RECOGNITION:

HOW TO IMPROVE SPEED OF INFERENCE AND CLASSIFICATION

Andrey V. Savchenko

Dr. of Sci., Prof.,

[1] laboratory LATNA at HSE

[2] Samsung-PDMI Joint AI Center

[3] MADE (Mail.ru)

Email: avsavchenko@hse.ru

URL: www.hse.ru/en/staff/avsavchenko

OpenTalks.AI
February 4, 2021

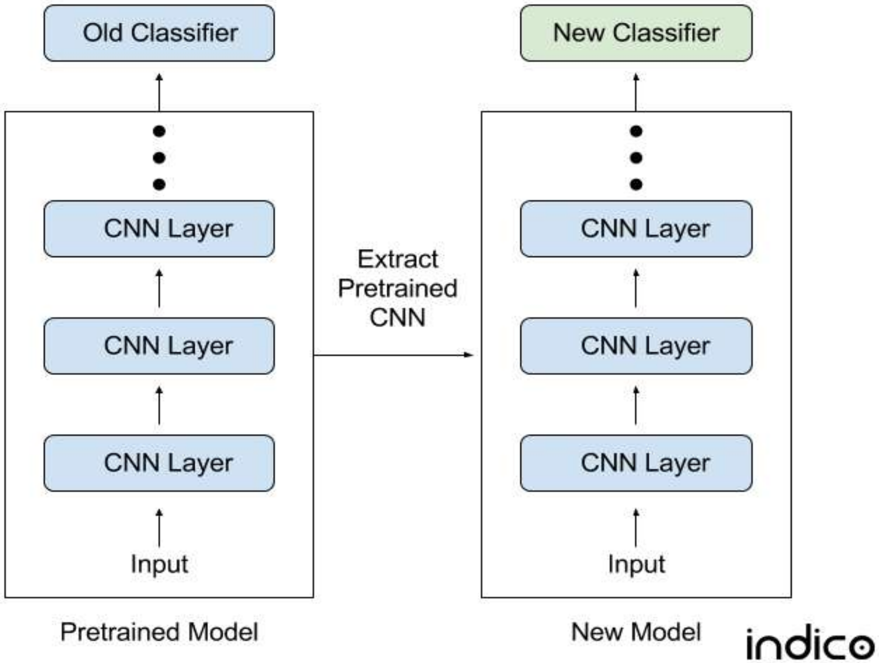
1. Motivation
2. Adaptive convolutional neural networks (CNNs)
3. Experimental results
4. Conclusion

Motivation

It is required to assign an observed image X to one of C classes. Training set contains N reference images (examples) $\{X_n\}$, $n \in \{1, \dots N\}$, with known class label $c_n \in \{1, \dots C\}$

1. Fine-tune convolutional neural network (CNN) pre-trained on ImageNet, Places, etc.

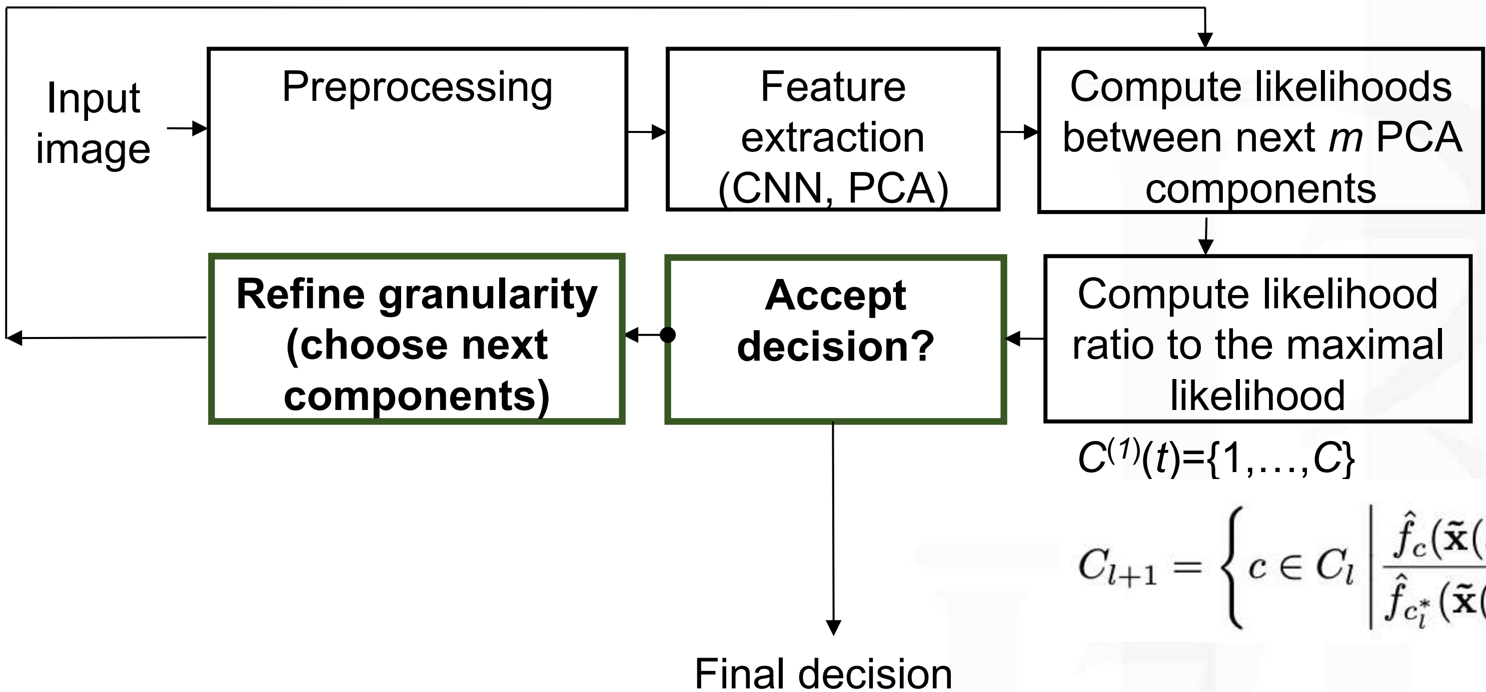
2. Classify *embeddings (features)* from one of the last CNN's layers: D-dimensional feature vector $\mathbf{x}=[x_1, \dots, x_D]$. Training set is associated with embeddings $\{\mathbf{x}_n\}$



The more the better

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

- **Inference in deep CNNs is slow**
- Do we need to perform the whole inference for every input image including the simplest one?



$$c_l^* = \underset{c \in C_l}{\operatorname{argmax}} \hat{f}_c(\tilde{\mathbf{x}}(l))$$

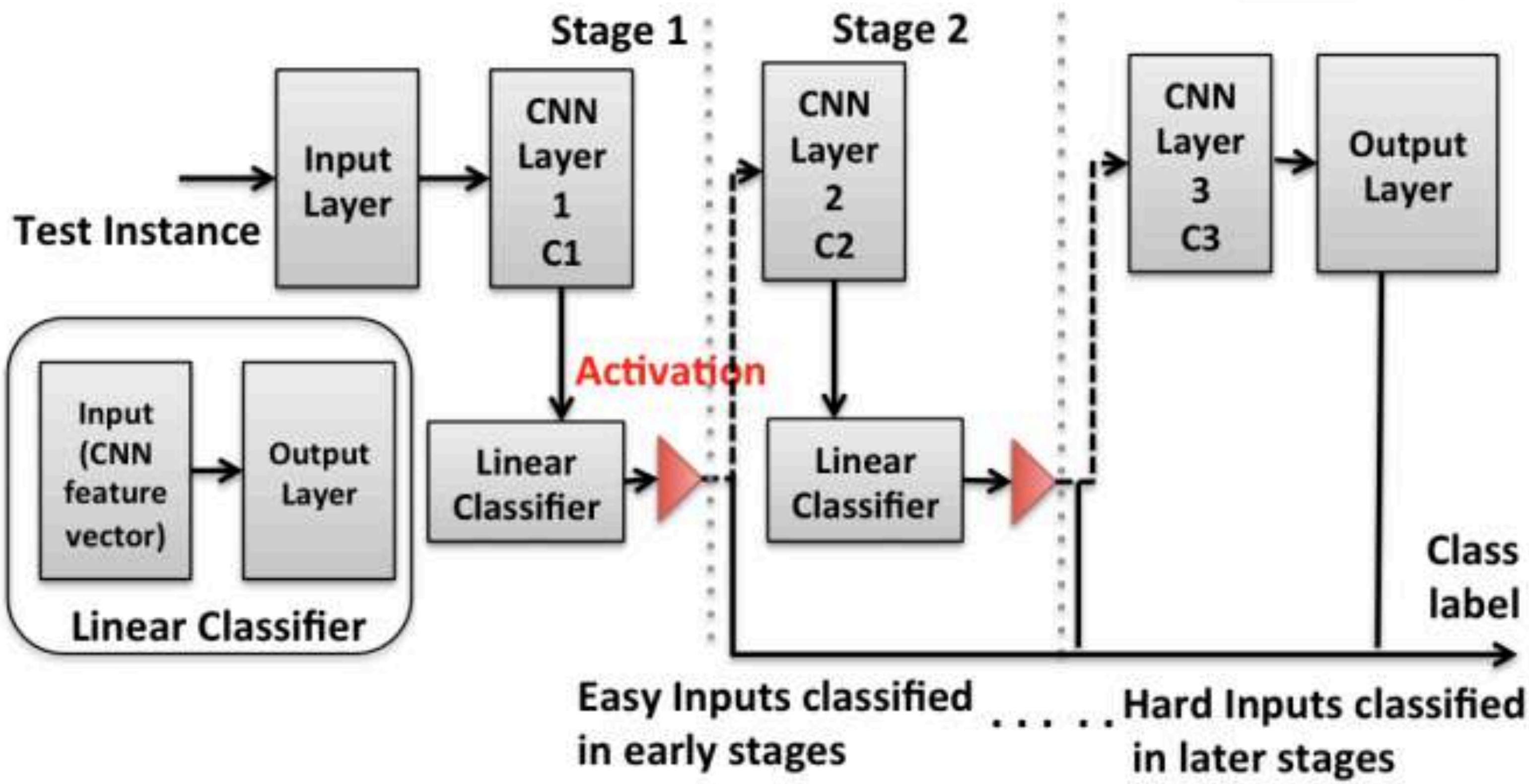
$C^{(1)}(t)=\{1,\ldots,C\}$

$$C_{l+1} = \left\{ c \in C_l \,\middle|\, \frac{\hat{f}_c(\tilde{\mathbf{x}}(l))}{\hat{f}_{c_l^*}(\tilde{\mathbf{x}}(l))} \geq \delta \right\}$$

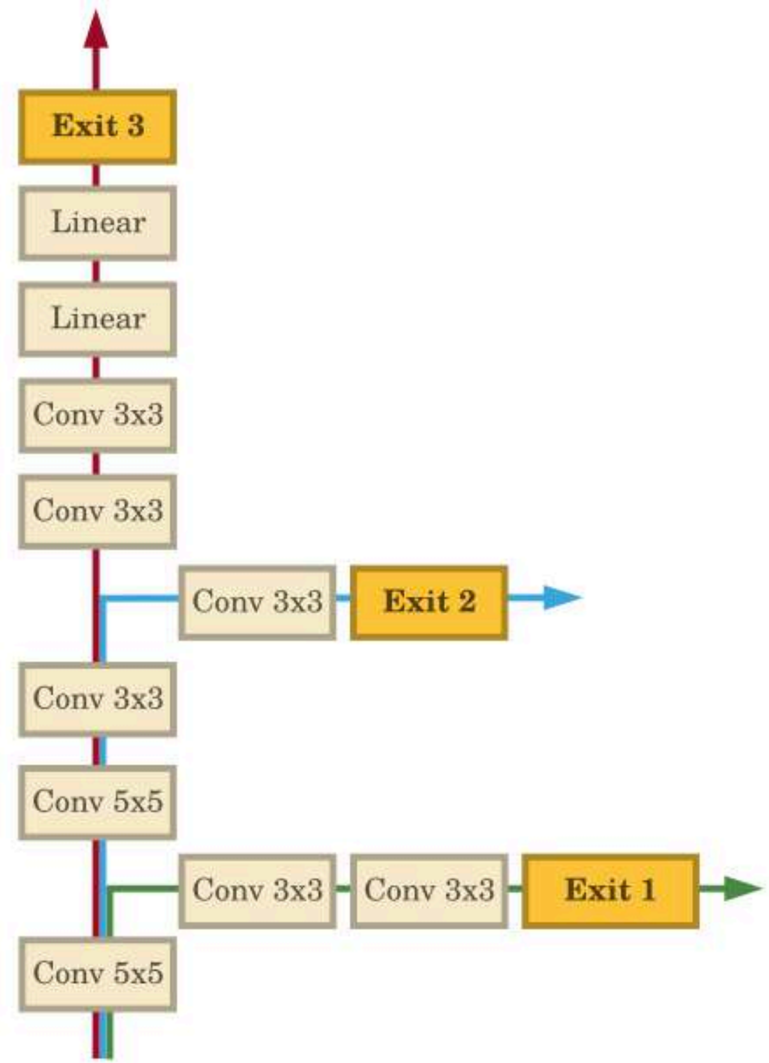
- Savchenko A.V. Information Sciences, 2019
- Savchenko A.V. Knowledge-Based Systems, 2016

Adaptive CNNs

Conditional deep learning (CDL) network

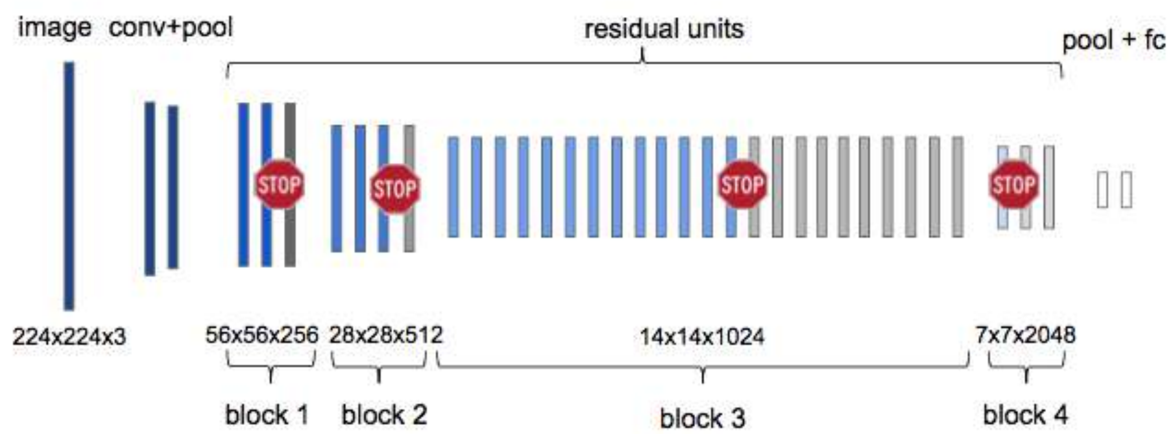


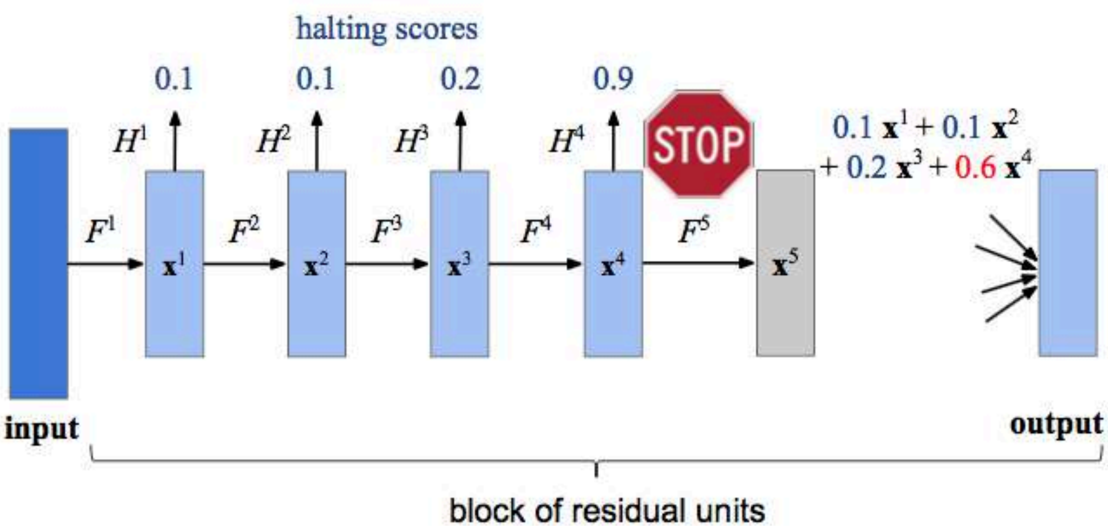
https://arxiv.org/pdf/1509.08971.pdf



https://arxiv.org/pdf/1709.01686.pdf

Adaptive computational time for ResNets





$$\mathbf{x}^0 = \mathbf{input},$$
$$\mathbf{x}^l = F^l(\mathbf{x}^{l-1}) = \mathbf{x}^{l-1} + f^l(\mathbf{x}^{l-1})$$
$$\mathbf{output} = \mathbf{x}^L.$$
$$h^l = H^l(\mathbf{x}^l) = \sigma(W^l \operatorname{pool}(\mathbf{x}^l) + b^l),$$

https://arxiv.org/abs/1612.02297

Other dynamic/adaptive CNNs

- K. Berestizshevsky and G. Even, “Dynamically sacrificing accuracy for reduced computation: Cascaded inference based on softmax confidence,” International Conference on Artificial Neural Networks (ICANN), 2019.
- A. Veit and S. Belongie, “Convolutional networks with adaptive inference graphs,” ECCV, 2018.
- R. Teja Mullapudi et al, “HydraNets: Specialized dynamic architectures for efficient inference,” CVPR, 2018.
- Leroux, Sam et al, “IamNN: Iterative and adaptive mobile neural network for efficient image classification”, ICLR2018 workshop
- Rao, Yongming et al, “Runtime Network Routing for Efficient Image Classification”, PAMI 2018
- T.Bolukbasi et al,“Adaptive neural networks for efficient inference,” ICML, 2017.
- Li, Zhichao et al, ”Dynamic computational time for visual attention”, International Conference on Computer Vision (ICCV) 2017,

…

- M > 1 intermediate layers (exit branches) are arbitrarily chosen to split the whole computational graph into *M* sequentially connected parts
- GAP and L2-norm layers are added in each exit branch.

- Input image is represented as a hierarchy of feature vectors $\mathbf{x}_1, ..., \mathbf{x}_M$.
- Sequential decision-making: *M* classifiers are trained, each classifier predicts *C*-dimensional vector of confidence scores $s_m^{(c)}(\mathbf{x}_m) \geq 0$

$$c_m^*(\mathbf{x}_m) = \underset{c \in \{1,...,C\}}{\operatorname{argmax}} s_m^{(c)}(\mathbf{x}_m). \quad (2)$$

$$\max_{c \in \{1,...,C\}} s_m^{(c)}(\mathbf{x}_m) > s_m. \quad (3)$$

Require: observed image
Ensure: class label
1: Initialize $\mathbf{z}_0$ by the RGB matrix of an input image X
2: **for** each intermediate layer $m \in \{1, ..., M\}$ **do**
3: Compute the output of the m-th layer $\mathbf{z}_m = f_{exit_m}(\mathbf{z}_{m-1}; \boldsymbol{\theta})$ (1)
4: Transform activations $\mathbf{z}_m$ into feature vector $\mathbf{x}_m$
5: Predict the confidence scores $\mathbf{s}_m(\mathbf{x}_m)$ using the m-th classifier
6: **if** $m = M$ OR condition (3) holds **then**
7: **return** class label $c_m^*(\mathbf{x}_m)$ (2)
8: **end if**
9: **end for**

Training

- How to choose thresholds?
- Train classifier on part of training set, predict confidence scores on the remaining training examples and fix the false acceptance rate (FAR) α_m

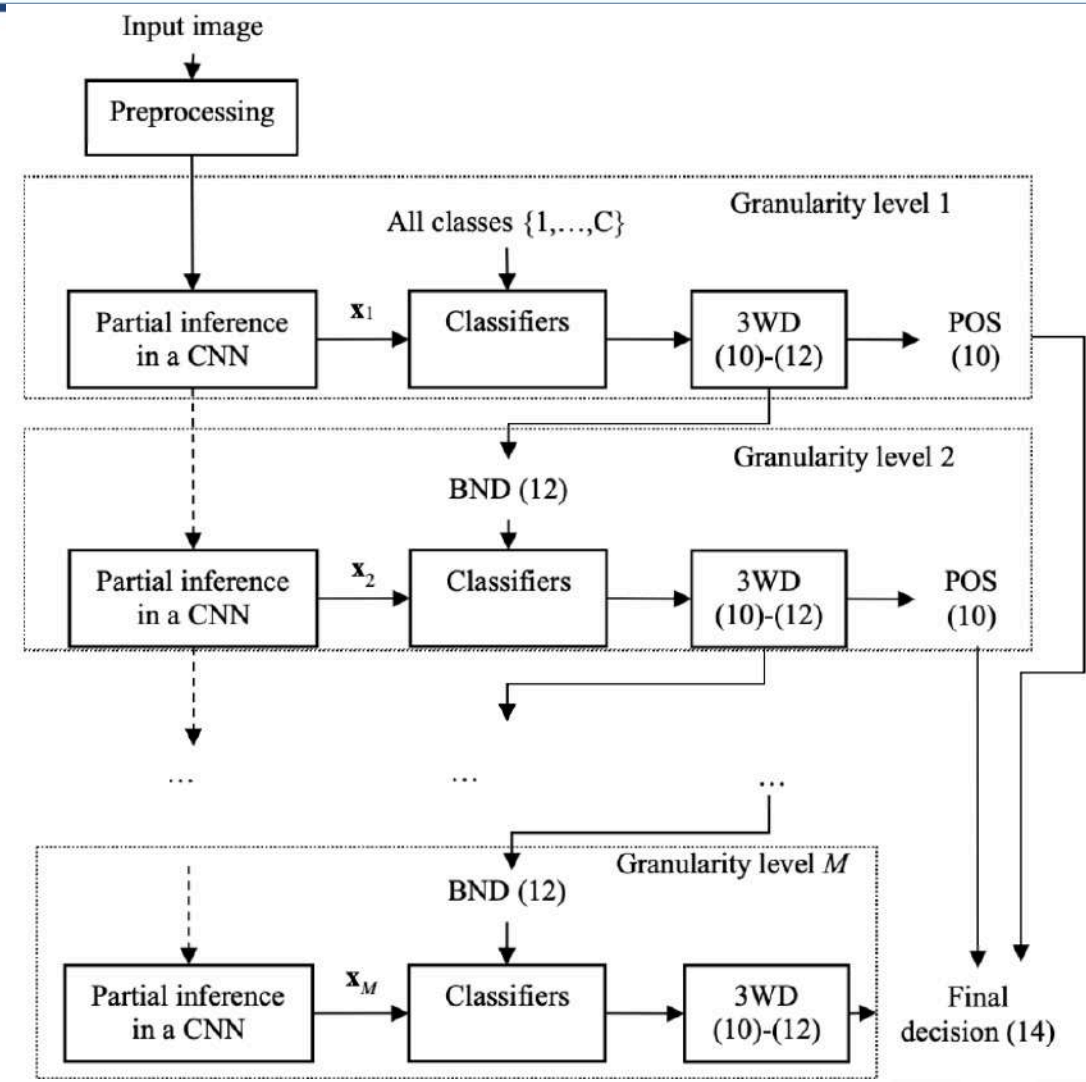
$$\sum_{n \notin \{n_1,\dots,n_K\}} H\left[\max_{c \neq c(n)} s_m^{(c)}(\mathbf{x}_{n;m}) - s_m\right] = \lfloor \alpha_m (N-K) \rfloor, \quad (4)$$

- How to choose FAR at the *m*-th level based on a given confidence level α of the whole decision-making procedure?
- Multiple testing problem with the Benjamini-Hochberg correction

$$\alpha_m = \alpha \cdot m/M$$

(Optional) Add fully-connected LR classifiers to all M branches and fine-tune the CNN on the given training set
for each intermediate layer $m \in \{1, ..., M\}$ **do**
  for each training instance $n \in \{1, ..., N\}$ **do**
    Feed the n-th image into a CNN and compute the outputs $\mathbf{x}_{n;m}$ at the m-th layer
  end for
  Split N instances in a stratified fashion using train/test split ratio δ to get indices $\{n_1, ..., n_K\}, K = \lceil N \cdot \delta \rceil$
  Train the m-th classifier, e.g., the linear one-versus-all SVM in order to maximize the squared hinge loss, using feature vectors $\{(\mathbf{x}_{n_k;m}, c(n))\}, k \in \{1, ..., K\}$
  Initialize a list of maximal intra-class scores $S = []$
  for each validation instance $n \notin \{n_1, ..., n_K\}$ **do**
    Append the maximal inter-class confidence score $\max_{c \neq c(n)} s_m^{(c)}(\mathbf{x}_{n;m})$ to the list S
  end for
  Assign the $\lfloor \alpha \cdot m(N-K)/M \rfloor$-th largest element from S to the threshold s_m (4) using the Benjamini-Hochberg correction
  Retrain the m-th classifier using all feature vectors $\{(\mathbf{x}_{n;m}, c(n))\}, n \in \{1, ..., N\}$
end for
return M classifiers and their thresholds $\{s_m\}$

Our approach



- Savchenko, IJCNN 2020
- Savchenko, Information Sciences 2021 (in print)
- Савченко, Записки научных семинаров ПОМИ, 2021

Experiments

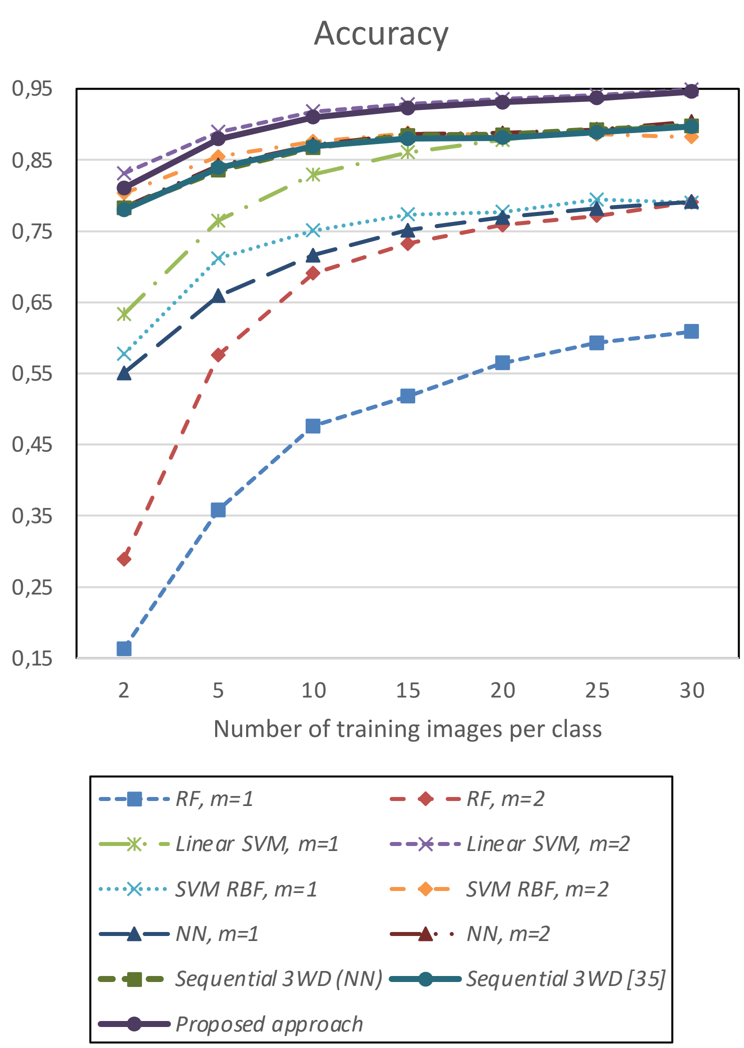
Datasets

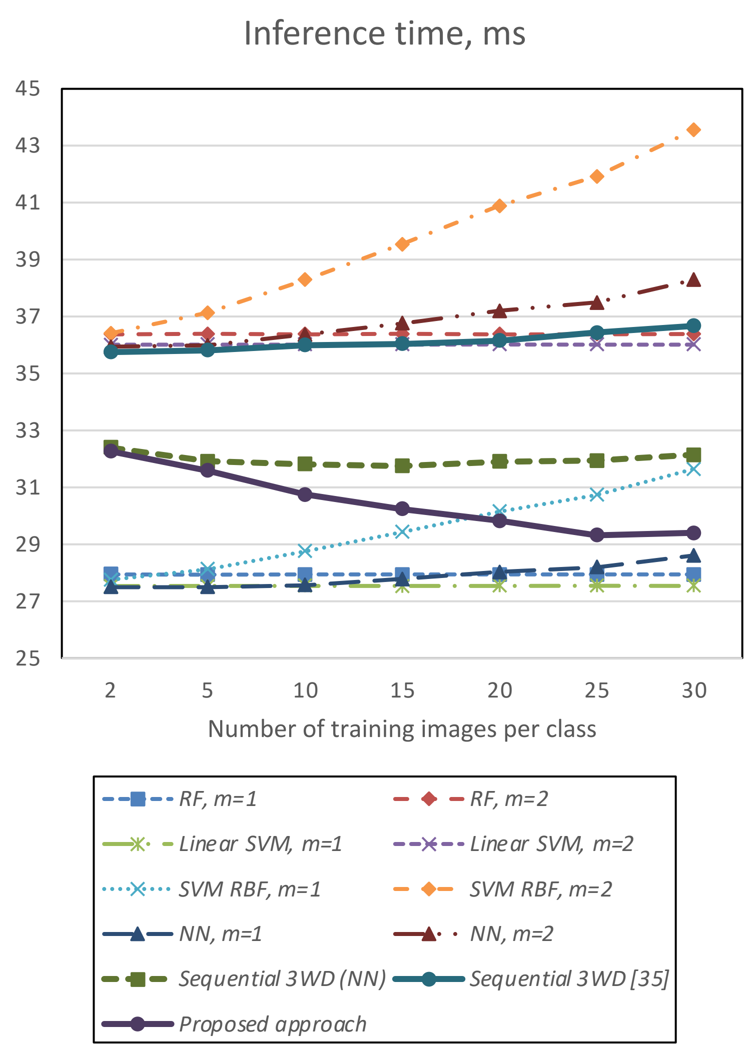
1. Caltech-101 Object Category dataset, which contains 8677 images of C = 102 classes including distractor background category.

2. Caltech-256 dataset with 29780 images of C = 257 classes including the clutter category.

3. Stanford dogs dataset that contains 20580 images and C = 120 classes.

Caltech-101 dataset, InceptionResNet v2





Performance analysis: Caltech-256 (60 instances per class), InceptionResNet v2

Classifier	Layers	block17_17_ac+block8_5_ac+penultimate				mixed_5b+block8_5_ac+penultimate	
		pre-trained		fine-tuned (all heads)		fine-tuned (all heads)	
		accuracy, %	time, ms	accuracy, %	time, ms	accuracy, %	time, ms
LR	first	0.91	26.29	1.85	26.34	7.82	4.61
LR	second	67.59	34.80	80.20	35.05	56.23	34.97
LR	last	62.88	42.94	77.81	40.78	78.86	40.51
RF	first	65.00	30.13	66.71	30.24	22.70	8.65
RF	second	76.40	38.51	78.43	39.00	79.32	45.14
RF	last	61.05	43.47	79.22	44.86	80.36	50.96
Linear SVM	first	81.10	26.40	80.75	26.52	41.25	5.01
Linear SVM	second	87.23	34.91	86.35	35.34	86.77	35.93
Linear SVM	penultimate	83.97	43.09	82.87	40.92	83.30	41.31
Cascaded inference [19]	all	-	-	77.73	37.75	71.06	34.47
CDL [11]	all	-	-	77.79	37.96	72.67	35.14
Ours, fixed threshold	all	84.07	29.36	86.14	33.57	86.14	32.40
Ours, adaptive threshold	all	86.95	30.09	86.10	30.21	86.20	30.31

Experimental results for EfficientNet-b7

Classifier	Layers	Caltech-101 block6b_add+block6f_add+penultimate accuracy, %	Caltech-101 block6b_add+block6f_add+penultimate time, ms	Caltech-256 block6f_add+block7b_add+penultimate accuracy, %	Caltech-256 block6f_add+block7b_add+penultimate time, ms	Stanford Dogs block7b_add+penultimate accuracy, %	Stanford Dogs block7b_add+penultimate time, ms
RF	first	60.10	176.47	46.52	185.48	86.75	205.87
RF	last	91.40	215.18	82.90	215.41	92.80	215.06
Linear SVM	first	82.12	172.72	71.18	181.76	86.82	198.03
Linear SVM	last	95.24	211.63	92.16	211.66	93.41	207.40
Ours, fixed threshold	all	92.07	178.71	83.42	191.46	90.53	200.27
Ours, adaptive threshold	all	94.49	181.91	91.13	197.82	92.99	202.18

Example results of image recognition, Caltech-101

						
Prediction at "block_6b_add" ($s_1 = 1.83$)	Confidence Class	1.14 scorpion	0.81 cannon	0.97 beaver	0.34 wrench	1.64 elephant
Prediction at "block_6f_add" ($s_2 = 1.36$)	Confidence Class	1.64 crab	1.66 anchor	2.03 dolphin	18.38 mayfly	1.52 kangaroo

The proposed approach is less limited than existing adaptive CNNs:

1. The inference speed of any pre-trained and fine-tuned CNN is increased even in few-shot learning with domain adaptation
2. In contrast to many existing methods that were developed for ResNets, proposed algorithm can be applied with any architecture
3. 1.06-1.7-times speed up over existing techniques on both CPU and GPU without significant accuracy degradation

and disadvantages

1. It is impossible to exit from one of the parallel convolutional layers
2. This study clearly highlights the main restriction of the practical application of all adaptive neural networks for few-shot learning: accuracy for the features extracted from early layers is 7-26% lower when compared to accuracy for features from deep layers.

NATIONAL RESEARCH
UNIVERSITY

Thank you!

This work is supported by HSE and RSF grant 20-71-10010