NATIONAL RESEARCH UNIVERSITY

National Research University Higher School of Economics (HSE) – N. Novgorod

IMAGE CLASSIFICATION BASED ON SEQUENTIAL ANALYSIS AND TRIGONOMETRIC SERIES

Andrey V. Savchenko
Dr. of Sci., Prof.,
Lead Researcher in HSE's international laboratory LATNA
Email: avsavchenko@hse.ru
URL: www.hse.ru/en/staff/avsavchenko

OpenTalks.AI
February 20, 2020

Outline

1. Motivation
2. PNN with trigonometric series kernel
3. Sequential analysis of high-dimensional features
4. Experiments

Motivation

Image recognition problem

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 $\{\mathbf{x}_n\}$

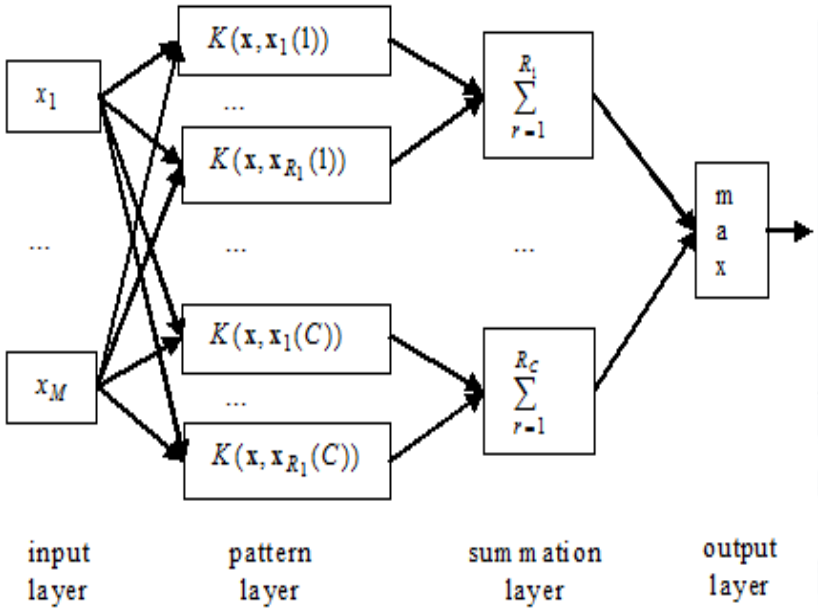
Statistical approach: empirical Bayesian classifier

$$\max_{c\in\{1,\dots,C\}} \frac{N(c)}{N}\hat{f}_c(\mathbf{x})$$

Rosenblatt-Parzen kernel with the Gaussian window

$$\hat{f}_c(\mathbf{x}) = \frac{1}{N(c)}\sum_{n=1}^{N(c)} K(\mathbf{x},\mathbf{x}_c(n))$$

$$K(\mathbf{x},\mathbf{x}_c(n)) = \frac{1}{(\sqrt{2\pi h^2})^D}\exp\left(-\frac{\rho(\mathbf{x},\mathbf{x}_c(n))}{2h^2}\right)$$



Advantages

1. High accuracy for small sample size (SSS): C≈N
2. Very high training speed

Disadvantages

1. Classification performance is low: O(*DN*)
2. Memory-based approach: space complexity is also linear

PNN with trigonometric series kernel

PCA components are L2 normalized and standardized

$$\overline{x}^{(d)} = \max\left(-x_{\max}, \min\left(x_{\max}, \lambda\frac{\tilde{x}^{(d)}}{\sigma_d}\right)\right)$$ – these features are bounded

Dirichlet kernel/trigonometric series should be used.

Canonical kernel estimate is replaced to the equivalent form, **which does not implement the brute force**

$$\hat{f}_{c;J}(\overline{x}^{(d)}) = 0.5 + \sum_{j=1}^{J}\left(a_j^{(d)}(c)\cos(\pi j\overline{x}^{(d)}) + b_j^{(d)}(c)\sin(\pi j\overline{x}^{(d)})\right)$$

$$a_{j;d}(c) = \frac{1}{N(c)}\sum_{n=1}^{N(c)}\cos(\pi j\overline{x}_{n;d}(c))$$

$$b_{j;d}(c) = \frac{1}{N(c)}\sum_{n=1}^{N(c)}\sin(\pi j\overline{x}_{n;d}(c))$$

Complexity linearly depends on the cut-off: O(*DCJ*).

Optimal cut-off for convergence:

$$J = \max\left(J_{\min}, \max_{c\in\{1,\ldots,C\}}\left\lceil\sqrt[3]{N(c)}\right\rceil\right)$$

The Dirichlet kernel is not always non-negative!

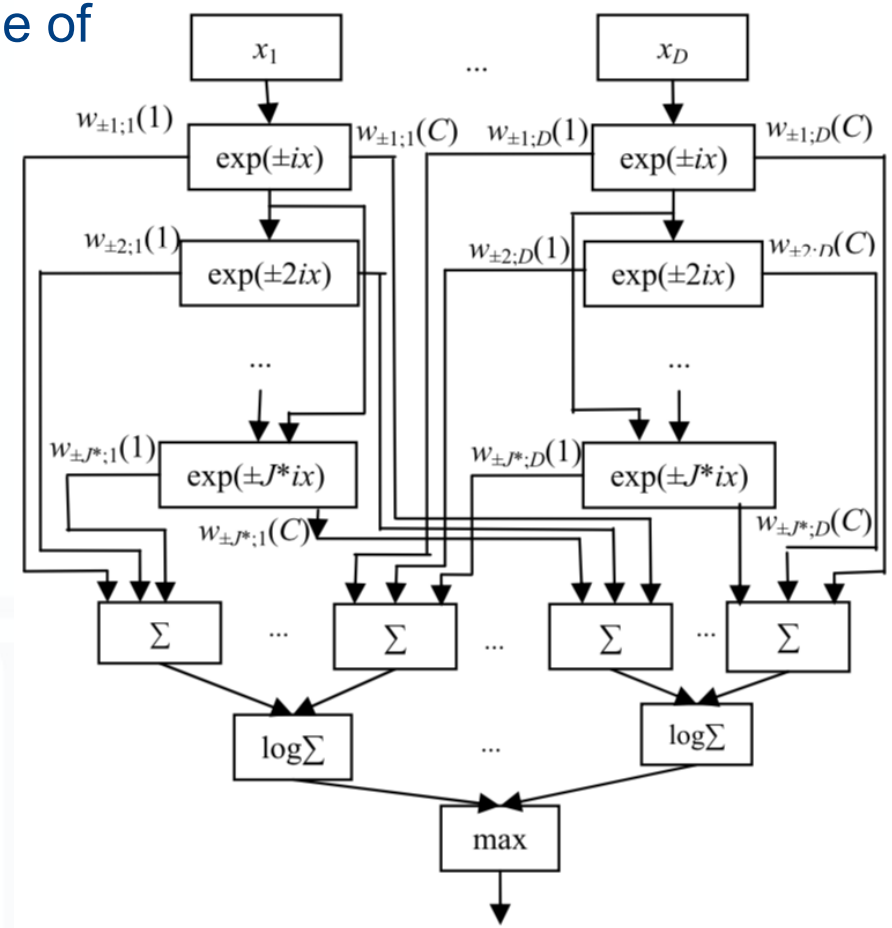
The likelihood is estimated as the average of the first *J* partial sums (Fejér kernel)

$$\check{f}_{c;J}(\overline{x}^{(d)}) = \frac{1}{J+1}\sum_{\tilde{J}=1}^{J+1}\hat{f}_{c;\tilde{J}}(\overline{x}^{(d)})$$

$$\check{f}_{c;J}(\overline{x}^{(d)}) = 0.5 + \sum_{j=1}^{J}\left(\check{a}_j^{(d)}(c)\cos(\pi j\overline{x}^{(d)}) + \check{b}_j^{(d)}(c)\sin(\pi j\overline{x}^{(d)})\right)$$

$$\check{a}_{j;d}(c) = \frac{J+1-j}{J+1}a_{j;d}(c),$$

$$\check{b}_{j;d}(c) = \frac{J+1-j}{J+1}b_{j;d}(c).$$



Advantages

- Converges to Bayesian solution
- Very high training speed
- Faster than original PNN

- Savchenko, A.V. IEEE Transactions on Neural Networks and Learning Systems, 2020
- Savchenko A.V., IEEE ICPR 2018

Sequential analysis of high-dimensional features

[Yao Y., Information Sciences, 2010]: "A **positive** rule makes a decision of **acceptance**, a **negative** rule makes a decision of **rejection**, and a **boundary** rule makes a decision of **abstaining**"

Key question: how to make a decision if the boundary region was chosen? Yao Y. Proc. of RSKT, LNCS, 2013: "Objects with a non-commitment decision may be further investigated by using fine-grained granules"

PCA (principal component analysis), scores are ordered by corresponding eigenvalues

$$\tilde{\mathbf{x}} = [\tilde{x}^{(1)}, \ldots, \tilde{x}^{(\tilde{D})}]$$

Proposed: computationally cheap representation of image at the *l*-th granularity level includes first $d^{(l)}=lm$ principal components

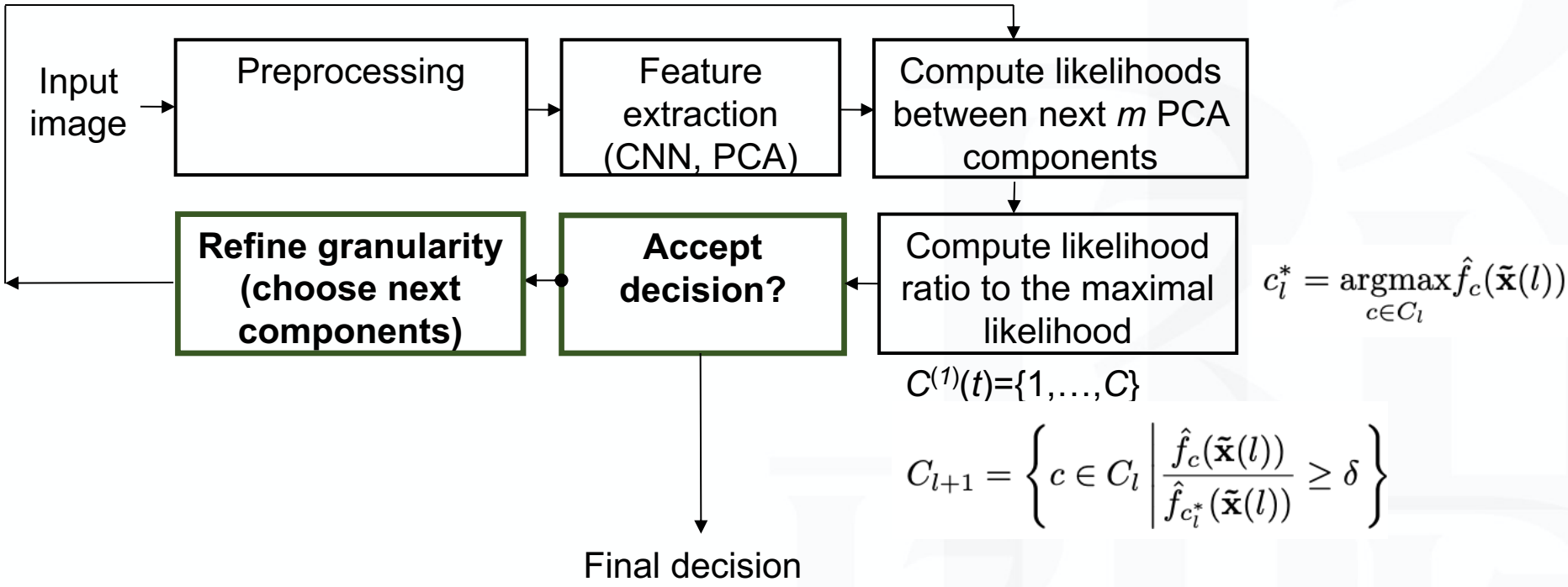
Original PNN

$$\rho\left(\tilde{\mathbf{x}}^{(l+1)}, \tilde{\mathbf{x}}_r^{(l+1)}\right) = \rho\left(\tilde{\mathbf{x}}^{(l)}, \tilde{\mathbf{x}}_r^{(l)}\right) + \sum_{d=(l-1)m+1}^{lm} \rho(\tilde{x}_d, \tilde{x}_{r;d}).$$

Our PNN

$$\hat{f}_c(\tilde{\mathbf{x}}(l)) = \hat{f}_c(\tilde{\mathbf{x}}(l-1)) \cdot \prod_{d=d_{l-1}+1}^{d_l} \check{f}_{c;J}(\overline{x}^{(d)})$$

Three-way decisions to choose robust representation of the input image



- Savchenko A.V. Information Sciences, 2019
- Savchenko A.V. Knowledge-Based Systems, 2016
- Patent RU 2706960 (22.11.2019) / Author: Savchenko A.V. Assignee: Samsung

Proposed sequential analysis for our PNN

Best runtime complexity (*m*=*D*/*L*):

$O(mN^{1/3}C^{1/3})$

Average runtime complexity:

$$O\left(N^{1/3}C^{2/3}m\sum_{l=1}^{L}\gamma^{l-1}\right)$$

Worst run-time complexity and memory space complexity:

$O(DN^{1/3}C^{1/3})$

Online classification is approximately $(N/C)^{2/3}$–times faster than instance-based learning (PNN, k-NN) if at least 5 images per class are available

Transform L_2-normed observed point $\mathbf{x}$ into principal component scores scaled by Eq. 6.

$LogF[c] = \log \frac{N(c)}{N}$, $Cand[c] = 1, c = \{1, ..., C\}$
for $l \in \{1, ..., L\}$ **do**
  for $d \in \{d_{l-1} + 1, ..., d_l\}$ **do**
    $Cos[1] = \cos(\pi \overline{x}^{(d)})$, $Sin[1] = \sin(\pi \overline{x}^{(d)})$
    for $j \in \{1, ..., J-1\}$ **do**
      $Cos[j+1] = Cos[j]Cos[1] - Sin[j]Sin[1]$
      $Sin[j+1] = Cos[j]Sin[1] + Sin[j]Cos[1]$
    end for
    for $c \in \{1, ..., C\}$ **do**
      if $Cand[c] == 1$ **then**
        $f = 0.5$
        for $j \in \{1, ..., J\}$ **do**
          $f = f + \check{a}_{j;d}(c)Cos[j] + \check{b}_{j;d}(c)Sin[j]$
        end for
        $LogF[c] = LogF[c] + \log f$
      end if
    end for
  end for
  $LogF^* = \max_{c \in \{1,...,C | Cand[c]=1\}} LogF[c]$
  for $c \in \{1, ..., C\}$ **do**
    if $LogF[c] < (LogF^* + d_l \cdot \log \delta)$ **then**
      $Cand[c] = 0$
    end if
  end for
  if $(Cand[1] + ... + Cand[C]) == 1$ **then**
    break
  end if
end for
Return the class label $c^* = \operatorname{argmax}_{c \in \{1,...,C | Cand[c]=1\}} LogF[c]$

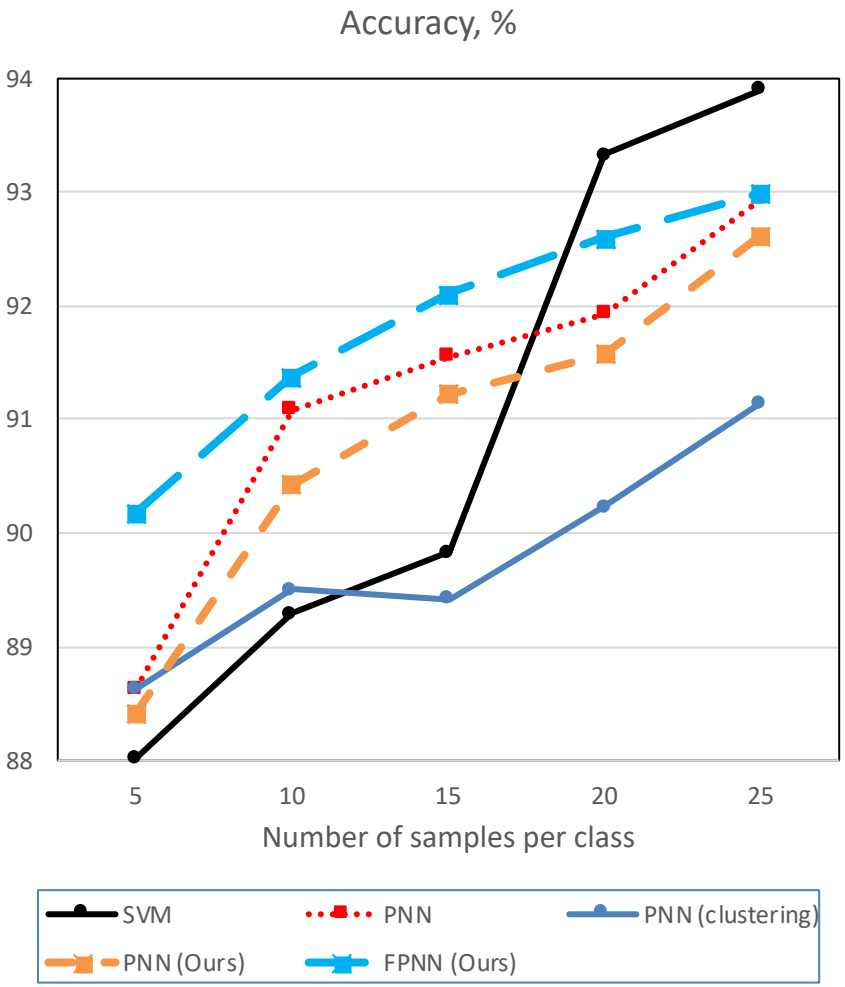
Experiments

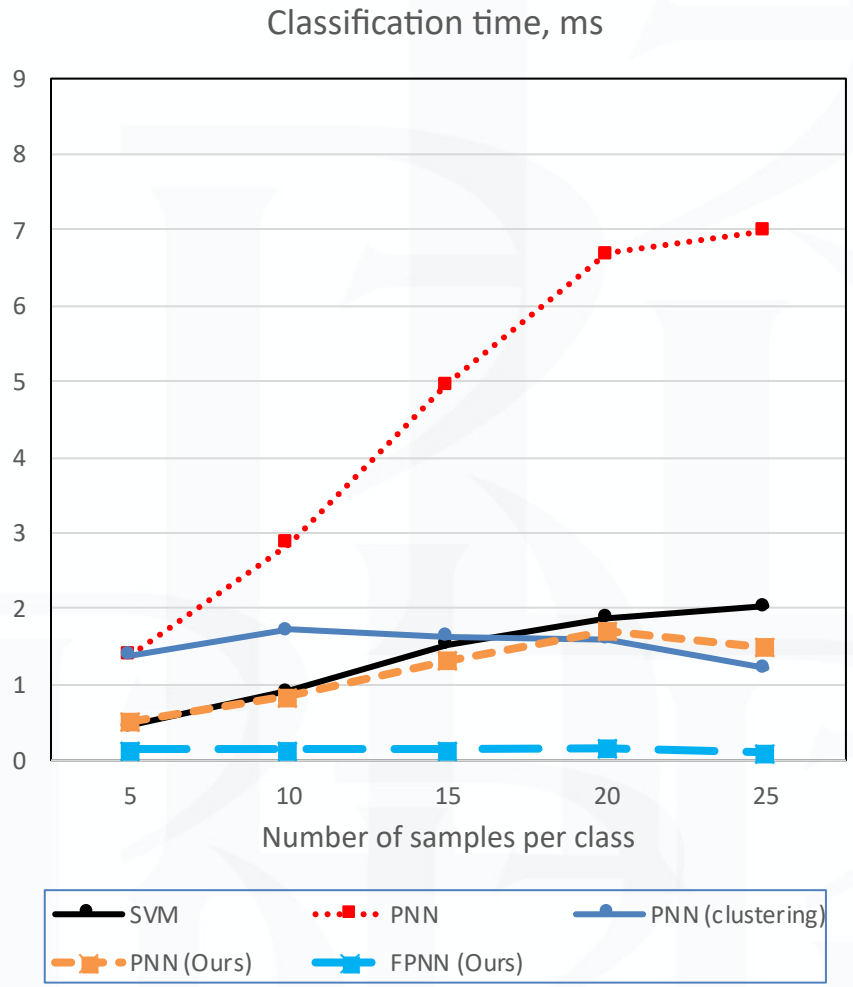
Caltech-101, 10 images per class

D	Classifier	Inception v3		InceptionResNet v2		EfficientNet-b5		EfficientNet-b7	
		Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
D	k-NN	84.36	7.157	84.01	22.89	89.51	21.709	90.43	32.846
	PNN	84.54	6.637	85.11	21.623	89.84	19.645	91.04	32.841
32	k-NN	76.64	0.182	79.03	0.655	83.54	0.185	85.93	0.559
	PNN	76.37	0.137	78.99	0.466	83.23	0.145	85.94	0.459
256	SVM	82.97	0.253	82.52	0.248	88.20	0.998	89.29	0.908
	RF	79.27	0.131	79.70	0.115	86.19	0.507	87.97	0.435
	k-NN	84.56	1.019	84.13	0.955	89.55	3.977	90.60	3.427
	PNN	84.39	0.825	83.56	0.914	89.63	3.504	91.09	2.876
	PNN (clustering)	82.68	0.395	81.82	0.436	87.84	1.372	89.50	1.729
	FPNN	86.66	0.203	84.97	0.199	90.85	0.568	91.78	0.814
	PNN (Ours)	83.92	0.162	83.54	0.142	89.05	0.633	90.43	0.838
	FPNN (Ours)	86.48	0.043	84.96	0.039	90.42	0.115	91.27	0.137

- Classification implemented using C++ language
- Classifiers from OpenCV 4
- Qt 5 framework
- Feature extraction: TensorFlow 2.0
- Features are extracted using pre-trained CNN models

Caltech-101, EfficientNet v7. Dependence of mean accuracy on the number of training instances per class





Other datasets, 10 images per class

Caltech-256

Classifier	Inception v3		InceptionResNet v2		EfficientNet-B5		EfficientNet-B7	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
SVM	70.63	3.621	74.41	4.483	80.99	1.071	82.51	3.456
RF	41.73	0.511	45.38	0.151	55.24	0.139	56.80	0.453
k-NN	72.69	8.68	75.40	9.411	82.01	2.613	83.44	11.09
PNN	70.61	7.102	74.72	9.522	81.60	2.144	83.29	9.164
PNN (clustering)	66.69	3.677	71.92	4.748	78.58	1.008	81.39	4.235
FPNN	74.78	1.815	77.32	2.207	83.75	0.492	85.25	1.759
PNN (Ours)	69.60	1.718	73.95	2.241	79.71	0.789	81.96	2.623
FPNN (Ours)	74.70	0.504	76.96	0.545	83.25	0.112	84.86	0.332

Stanford Dogs

Classifier	Inception v3		InceptionResNet v2		EfficientNet-B5		EfficientNet-B7	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
SVM	87.00	0.34	90.71	1.119	93.28	0.34	93.36	1.32
RF	85.86	0.116	89.46	0.403	92.42	0.117	92.68	0.445
k-NN	86.49	1.265	90.46	4.051	93.04	1.252	93.06	5.338
PNN	83.34	1.172	88.44	3.222	91.52	1.173	91.61	4.487
FPNN	87.21	0.23	90.32	0.746	93.45	0.23	93.17	0.846
PNN (Ours)	83.29	0.214	88.44	0.759	91.44	0.298	91.57	1.088
FPNN (Ours)	87.34	0.047	90.58	0.117	93.46	0.04	93.40	0.141

Conclusion

Proposed approach has a list of advantages

1. It saves all advantages of the PNN including the convergence to the optimal Bayesian decision
2. It significantly improves the classification running time
3. It is more accurate in most cases than k-NN, PNN and its modifications
4. C++ implementation is freely available: https://github.com/HSE-asavchenko/fast-image-recognition

and disadvantages

1. Naïve assumption about independence of PCA components
2. No distance calculation as in the Gaussian kernel in the PNN
3. Inference speed may limit the total performance

NATIONAL RESEARCH
UNIVERSITY

Thank you!