LDS-Inspired Residual Networks

Anastasios Dimou, Dimitrios Ataloglou, Kosmas Dimitropoulos, *Member, IEEE,*
Federico Alvarez, *Member, IEEE,* and Petros Daras, *Senior Member, IEEE*

***Abstract*—Residual Networks (ResNets) have introduced a milestone for the Deep Learning community due to their outstanding performance in diverse applications. They enable efficient training of increasingly deep networks, reducing the training difficulty and error. The main intuition behind them is that instead of mapping the input information, they are mapping a residual part of it. Since the original work, a lot of extensions have been proposed to improve information mapping. In this paper, a novel extension of the residual block is proposed inspired by Linear Dynamical Systems, called LDS-ResNet. Specifically, a new module is presented that improves mapping of residual information by transforming it in a hidden state and then mapping it back to the desired feature space using convolutional layers. The proposed module is utilized to construct multi-branch Residual blocks for Convolutional Neural Networks (CNNs). An exploration of possible architectural choices is presented and evaluated. Experimental results show that LDS-ResNet outperforms the original ResNet in image classification and object detection tasks on public datasets such as CIFAR-10/100, ImageNet, VOC and MOT2017. Moreover, its performance boost is complementary to other extensions of the original network such as pre-activation and bottleneck, as well as stochastic training and Squeeze-Excitation.**

***Index Terms*—ResNet, linear dynamical systems, convolutional neural networks, image classification, object detection.**

I. Introduction

WHILE increasingly deeper networks are proposed in the literature, their performance gets saturated and even degrades rapidly after a certain depth [1]. Despite the intuition that a deeper model should perform better than its shallower counterpart, adding more layers may lead to a higher training error. The difficulty to train such a deeper network stems from the increasing complexity and the diminishing returns. Deep Residual Networks (ResNets) [1] introduced a new approach for Image Recognition with Deep Residual Learning. ResNets led to winning entries in the 2015 ImageNet [2] and MS COCO [3] competitions, in important tracks such as image classification, object detection, and semantic segmentation. The robustness of ResNets has since been proven in diverse visual recognition tasks and non-visual tasks including speech recognition systems [4].

D. Ataloglou, K. Dimitropoulos and P.Daras are with the Information Technologies Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece (e-mail: {ataloglou,dimitrop,daras}@iti.gr).

F. Alvarez is with the Universidad Politécnica de Madrid, Madrid, Spain (e-mail: fag@gatv.ssr.upm.es).

A. Dimou is both with the Universidad Politécnica de Madrid, Madrid, Spain and the Information Technologies Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece (e-mail: dimou@iti.gr).

Manuscript received June 6, 2018; revised August 24, 2018; accepted 3 September, 2018.

ResNets consist of many stacked "Residual Blocks". Each block can be expressed in a general form:

$$y_l = h(x_l) + F(x_l, W_l), \tag{1}$$

$$x_{l+1} = f(y_l), \tag{2}$$

where x_l and x_{l+1} are input and output, respectively, of the l_{th} block, and F is a residual function. The central idea of ResNets is to learn the additive residual function F with respect to $h(x_l)$. The key choice in [1] is to use an identity mapping as h and a ReLU function [5] as f. The former is realized by attaching an identity skip connection, $h(x_l) = x_l$.

Instead of trying to fit each stack of layers directly to a desired underlying mapping, ResNet layers are trained to fit a residual mapping. The original mapping $F_{orig}(x_l)$ is recast into $F(x_l) + x_l$. It is argued that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. Hypothesizing that an identity mapping is the optimal one, it would be easier to push the residual to zero than to fit an identity mapping by a stack of non-linear layers [1]. ResNets that are over 100-layer deep have shown state-of-the-art accuracy for several challenging recognition tasks.

In this paper, Residual Networks are re-visited focusing on the residual block as it proves to be the key for their success, while the depth of the network seems to have a supplementary effect. ResNets proved that residual mapping is easier to model than the original one [6]. Building on this insight, it is argued that methodologies for better modelling the residual information are expected to further improve the performance of the network.

An effective strategy to improve the modelling of the residual information is presented by Inception-like multi-branch architectures [7]. Inception models are based on a split-transform-merge strategy, where the input is tunnelled into specialized branches that can model a different aspect of the residual information and are then merged again. These lower-dimensional embeddings create a strict subspace of the solution space of a single large layer operating on a high-dimensional embedding. However, Inception modules are approaching the representational power of a high-dimensional embedding with a considerably lower computational complexity, allowing the exploration of deeper networks.

The proposed method shares its principles between the two most important families of neural networks, namely ResNets and Inception networks. From ResNets, the research community has gained the insight that it is easier to try to model a residual of the initial information content rather than its whole. On the other hand, Inception networks have proved that modelling different aspects of the information content, using dedicated branches with different spatial support, can lead to improved performance. Starting from Inception v4 [8], skip connections were integrated to Inception-like models, confirming the compatibility of the two approaches.

A new approach to break down the information content is proposed here, aiming to improve data modelling by a residual block. It is argued that exploiting data statistics, a new representation of the residual data can be constructed to improve residual information modelling. This representation can offer a more generic description of the underlying texture that is easier to model. Towards this end, in this paper we consider a sequence of convolutional layers as a Markov process, in which the statistical representation of the process in the future, i.e., in the next layer, is completely determined by the present state, i.e. the output of the previous layer. Inspired by Linear Dynamical Systems (LDS) and their ability to estimate the output of such systems through two stochastic processes, we initially design a novel module that simulates an LDS process and then propose various LDS block architectures aiming to better estimate the output of the residual block. A 3-path block is proposed, featuring: (1) a skip connection, (2) standard convolutional layers and (3) LDS modules, aiming at better modelling the residual information and ease the training process.

In order to understand the role of LDS-like modelling in a ResNet block, an exploration of the architecture design phase is presented. It is shown that such ResNet blocks achieve significantly better accuracy compared to the original ResNet block in the image classification problem. Experiments suggest that LDS-ResNet performs better than ResNets with a similar number of parameters. Moreover, it is shown that the performance boost is additive to other extensions of the original network such as pre-activation and bottleneck architectures, as well as stochastic training and Squeeze-Excitation. The new residual unit design presents improvements in the image classification and object detection tasks in public datasets such as CIFAR-10/100, ImageNet, VOC and MOT2017.

The contributions of this work can be summarized in the following aspects:

- A new module inspired by Linear Dynamical Systems (LDS) is introduced and the addition of a separate processing branch, consisting of stacked LDS modules, is proposed to better model the residual information.
- An extensive exploration of different architectures including the proposed module is performed. Different network topologies are developed and tested to compare performance with the original ResNet for multiple network depths. Moreover, the complementarity of the proposed methodology with other ResNet extensions is assessed.
- The proposed network design has been benchmarked on image classification and object detection tasks in popular publicly available datasets with favorable results.

II. Related work

Since ResNet appearance, the concept of residual learning has been extensively utilized and extended in various ways. In [6], authors proposed *Wide Residual Networks (WRNs)* to address the training difficulties and saturated performance of increasingly deeper models. WRNs were constructed by simultaneously decreasing the depth and increasing the width of residual networks, using a higher number of convolutional kernels and feature maps. These modifications resulted in much swallower networks that could generally achieve better performance than their deeper counterparts, while being easier to train.

In another attempt to alleviate training related inefficiencies in deep neural networks, such as vanishing gradients, diminishing feature reuse and long training times, the authors in [9] followed a different approach. Starting with very deep neural networks, they proposed to stochastically drop a percentage of residual units in each training iteration and replace them with identity connections. This method, which extended the functionality of Dropout [10] to residual blocks, was able to reduce training time and continue improving the test error in residual networks with more than 1000 layers.

In ResNets, the input of each residual unit is forwarded to the next one through the identity shortcut and an addition operation with that unit's output. Thus, identity connections exist only between successive units and input information is merged and altered before it can be utilized by subsequent units. In *DenseNets* [11], the feature-maps of all preceding units are concatenated and used as inputs, while a unit's output is used as input by all subsequent units. Concatenating feature maps learned by multiple different layers increases feature reuse throughout the network, which leads to improved training efficiency and lower classification error.

In [12], a deep dual-stream architecture was proposed, named *ResNet in ResNet (RiR)*, composed of generalized residual units that combine intersecting residual and transient paths. The generalized residual units retained the benefits of residual learning, while improving expressivity and the ease to remove unnecessary information.

In order to boost the representational power of the network, [13] focused on interdependencies between the different channels of convolutional outputs and proposed the Squeeze-and-Excitation (SE) block, which adaptively re-calibrates channel-wise feature responses. Through feature recalibration, the network uses global information and learns to selectively emphasize informative features and suppress less useful ones. Combined with residual connections, SE-ResNet units have set a new record in the ImageNet classification challenge.

The current literature includes papers that improve performance by easing the learning procedure [6], [9], adding complexity to the network architecture [11], [12] or modifying each filters contribution [13]. In the proposed approach, it is argued that by transforming the residual data to a hidden state and then mapping it back to desired feature space with convolutional layers, mimicking Linear Dynamical Systems, significantly improves the network performance.

III. Proposed Method

A. Background

In the literature, Linear Dynamical Systems (LDS) have been widely used for the analysis of multi-dimensional data, such as time series and dynamic textures [14], [15], [16],

[17]. An LDS is associated with a first order Auto-Regressive Moving Average (ARMA) process with white zero mean independent and identically distributed (IID) Gaussian input and for this reason LDS are also known as linear Gaussian state-space models. The stochastic modelling of both signal's dynamics and appearance is encoded by two stochastic processes, in which dynamics are represented as a time-evolving hidden state process $x_t \in R^n$ and the observed data $y_t \in R^d$ as a linear function of the state vector:

$$x_t = Ax_{t-1} + Bv_t \tag{3}$$

$$y_t = Cx_t + w_t \tag{4}$$

where $A \in R^{n\times n}$ is the transition matrix of the hidden state, while $C \in R^{d\times n}$ is the mapping matrix of the hidden state to the output of the system. The quantities w_t and Bv_t are the measurement and process noise, respectively, with $w_t \sim N(0, R)$ and $Bv_t \sim N(0, Q)$.

In general, LDS attempt to associate the output of the system with a linear function of a state variable, while in each time instant the state variable depends linearly on the state of the previous time instant. However, many researchers have attempted to introduce a non-linear observation function f in (4), i.e., $y_t = f(x_t)$, maintaining in (3) the linear state transition [18], while others have used non-linear functions in both stochastic processes [19]. Besides the modelling of time-evolving data, LDS have been effectively applied to problems involving the analysis of spatial information, such as still images, where the evolution of data is performed in the spatial domain, i.e., in consecutive pixels, instead of discrete time instances [20].

B. LDS module

Motivated by the aforementioned, in this paper, we aim to create a novel LDS-inspired module that simulates the operation of an LDS system in order to improve the residual mapping of each block in a ResNet. Towards this end, instead of using a single identity mapping, we introduce a new branch with stacked LDS modules in order to calculate an estimated output for each residual block.

More specifically, when used within a CNN, the input to the proposed LDS module corresponds to a feature volume, which is a 3-D tensor of size $h \times w \times d_{in}$, where h, w are the spatial dimensions and d_{in} is the number of feature maps. The proposed LDS module operates over 3-D patches $X_{t-1} \in R^{n\times n\times d_{in}}$ along the spatial dimensions of the input tensor (we used $n = 3$ in our experiments). More specifically, after applying the first-mode unfolding to X_{t-1} and obtaining a 2-D matrix $x_{t-1} \in R^{n\times n\cdot d_{in}}$, the corresponding output $y_t \in R^{d_{out}}$ is calculated as follows:

$$x_t = Ax_{t-1} \tag{5}$$

$$y_t = f(W, x_t) \tag{6}$$

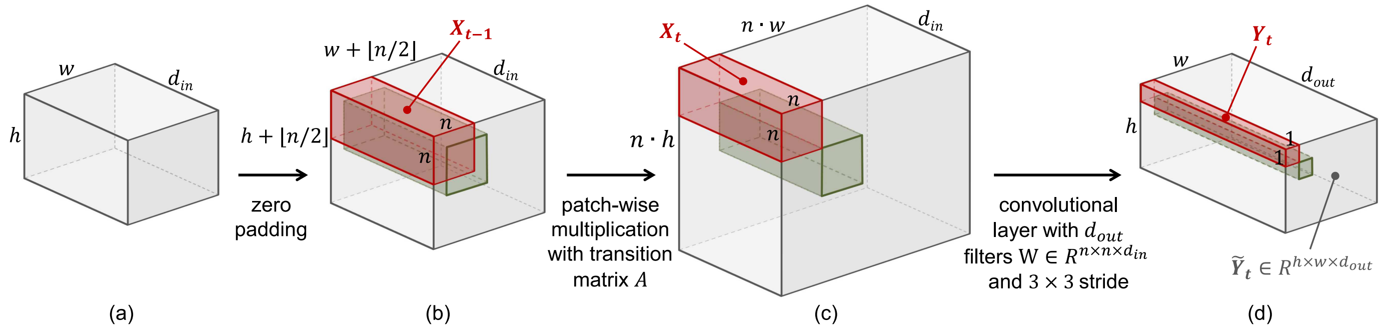
where $A \in R^{n\times n}$ is a square transition matrix, $x_t \in R^{n\times n\cdot d_{in}}$ is a transformation of the unfolded input patch and $f()$ is a mapping function, which uses x_t and a set of trainable parameters W to calculate the output y_t. In (5) and (6), t refers to the current processing step within a CNN structure, while $t-1$ to the previous one. All patch-wise outputs y_t are concatenated to a single 3-D tensor at the output of the LDS module.

The operation of the proposed LDS module is graphically presented in Fig. 1. Calculating the LDS module's output $\widetilde{Y}_t$ involves three distinct steps. Firstly, in order to preserve the spatial dimensions between the input and the output tensors, the input tensor is zero padded along the spatial dimensions. When n is an odd scalar, the resulting tensor has a size of $(h + \lfloor n/2 \rfloor) \times (w + \lfloor n/2 \rfloor) \times d_{in}$. Then, we apply (5) individually to each 3-D patch X_{t-1} by unfolding it to a 2-D matrix and multiplying by A. Multiplication products are folded back to 3-D subvolumes and stored successively without overlapping, resulting in an intermediate volume of $(n \cdot h) \times (n \cdot w) \times d_{in}$. Finally, this intermediate volume is convolved with a set of d_{out} filters $W \in R^{n\times n\times d_{in}}$, using a standard convolutional layer. Convolution is performed with a stride of n in each spatial dimension, in order to align the filters W with the regions corresponding to each of the 3-D patches X_{t-1} used in the previous step. In this way, the output of the LDS module $\widetilde{Y}_t$ is a $h \times w \times d_{out}$ tensor. The stride of the convolutional operation within the LDS module can be adjusted to multiples of n, which can prove useful in classification tasks, where the spatial dimensions of the input are usually reduced multiple times throughout the network. For example, a stride of $k \cdot n$ will reduce both the width and the height of the input tensor by k.

LDS modules can be stacked to form an LDS stack (Fig. 2). In this case, the output of every LDS module is used as input by the subsequent module. A residual block can be modified by adding an LDS stack in a separate processing branch. Fig. 3 shows the resulting LDS block in an abstract form, where the input X is processed by two parallel branches. Similarly to the original ResNet, the first branch consists of successive convolutional layers, while the second branch is either an LDS module or an LDS stack. The outputs Y and $\widetilde{Y}$ of the two branches are then combined by a function $G()$, before the addition of the input X, which is forwarded through an identity shortcut. The function $G()$ depends on the examined block type and is described in section III-C.

1) Transition matrix calculation: We investigate two different approaches regarding the calculation of the transition matrix A. In the first one, its values are randomly initialized and subsequently optimized during CNN training through backpropagation. In this case, the values of A are trainable parameters, along with the filters of the convolutional layer. Aiming to preserve the spatial invariance property of CNNs, we always use the same matrix A when multiplying with values belonging to the same feature map of the LDS module's input tensor. On the other hand, we explored both the usage of a single matrix A for the whole input and d_{in} independent matrices, each dedicated to a specific channel of the input tensor.

In the second approach, inspired by [17], A is recalculated every time directly for each input patch X_{t-1} and is not optimized through backpropagation. Firstly, X_{t-1} is unfolded to produce a 2-D matrix. Starting from the first feature map,



w
d_{in}
h
$h + \lfloor n/2 \rfloor$
zero padding
$w + \lfloor n/2 \rfloor$
X_{t-1}
d_{in}
n
n
patch-wise multiplication with transition matrix A
$n \cdot h$
X_t
$n \cdot w$
d_{in}
n
n
convolutional layer with d_{out} filters W $\in R^{n \times n \times d_{in}}$ and 3 × 3 stride
w
Y_t
d_{out}
h
1
1
$\tilde{Y}_t \in R^{h \times w \times d_{out}}$
(a)
(b)
(c)
(d)

Fig. 1. Graphical illustration of the proposed LDS module's operation. The 3-D volumes depicted in gray are: a) LDS module's input, b) zero-padded input, c) intermediate volume storing the multiplication products of each 3-D patch X_{t-1} with the transition matrix A and d) LDS module's output. The input to the LDS module is a 3-D tensor. In (b), the green patch is shifted by one position down and to the right compared to the red patch. In (c), their multiplication products with the transition matrix A are stored without overlapping. In (d), the outputs corresponding to each X_{t-1} are vectors with d_{out} elements.

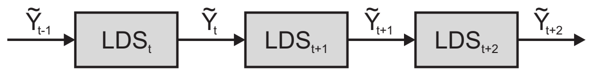


Fig. 2. An LDS stack, composed of multiple LDS modules. The output of each LDS module can be directly used as input by the subsequent one.

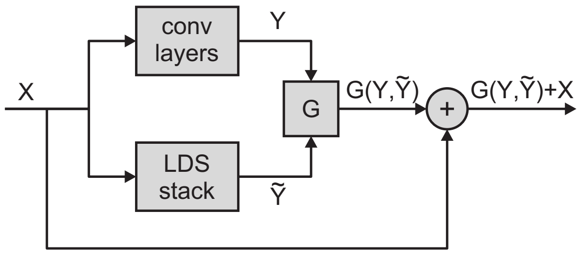


Fig. 3. Abstact form of an LDS block, combining convolutional layers, LDS modules and an identity shortcut.

each $n \times n$ plane is placed to the right of the previous one, producing a 2D matrix $x_u \in R^{n \times n \cdot d_{in}}$ Then, A is calculated as:

$$A = X_2 X_1^T (X_1 X_1^T)^{-1} \tag{7}$$

where

$$X_1 = [x_u(1), x_u(2), ..., x_u(3d_1 - 1)] \tag{8}$$

$$X_2 = [x_u(2), x_u(3), ..., x_u(3d_1)] \tag{9}$$

and $x_u(i)$ is the i-th column of x_u.

2) LDS module implementation: The LDS module can be easily constructed utilizing a deep learning framework. It consists of three layers: a padding layer, a custom layer and a 2-D convolutional layer. The padding and the convolutional layers are included in most deep learning frameworks. For the latter, we used the CuDNN library [21] to speed up the training and inference processes. Calculation of the transition matrix A (when needed) and patch level multiplication (5) are executed by the custom layer. As this can become a computationally heavy operation for larger input sizes, the custom layer was implemented in CUDA to enable high level of parallelization. The code will be available upon the acceptance of this paper.

C. LDS blocks

In this section, the exploitation of the LDS module within residual blocks is explored in different setups. After describing the baseline residual block, different LDS block architectures are implemented and evaluated. An overview of tested block alternatives can be seen in Fig. 4.

1) Baseline Residual block: The baseline residual block, depicted in Fig. 4a, consists of two parallel branches. The first branch includes two 2-D convolutional layers with 3×3 filters, whereas the second one is a simple identity shortcut. Both convolutional layers use zero padding of 1 pixel and are followed by a batch normalization layer [22]. The outputs of the two branches are added and the second ReLU activation [5] is placed after the addition. When the number of feature maps of the input is changed by the convolutional layers, either A) the shortcut's output is padded with zeros to match the first branch's output, or B) the identity shortcut is replaced by a projection shortcut, which consists of a 1×1 convolutional layer with the desired number of filters, followed by a batch normalization layer. When a convolutional layer reduces the spatial dimensions of the input by performing strided convolution, a spatial average pooling layer with the same stride is added in the identity shortcut branch or the same stride is employed by the 1×1 convolutional layer, in case of a projection shortcut.

2) Baseline LDS block: The baseline LDS block is presented in Fig. 4b. The first branch of the baseline residual block is replaced by a two pathway structure. The first path is similar to that of the baseline residual block, including two convolutional layers, followed by batch normalization and ReLU activation layers. In the second path, the convolutional layers are substituted with LDS modules. Each LDS module has the internal structure described in section III-B. The convolutional layers inside the LDS modules always use the same number of filters as the corresponding ones of the first path, but their stride is multiplied by 3. The outputs of the two pathways are concatenated along the third dimension, resulting in a volume with the same spatial dimensions, but double the amount of feature maps. An additional 1×1 convolutional layer is placed after the concatenation, which restores the number of feature maps to that of the input by blending the information received from the convolutional and the LDS

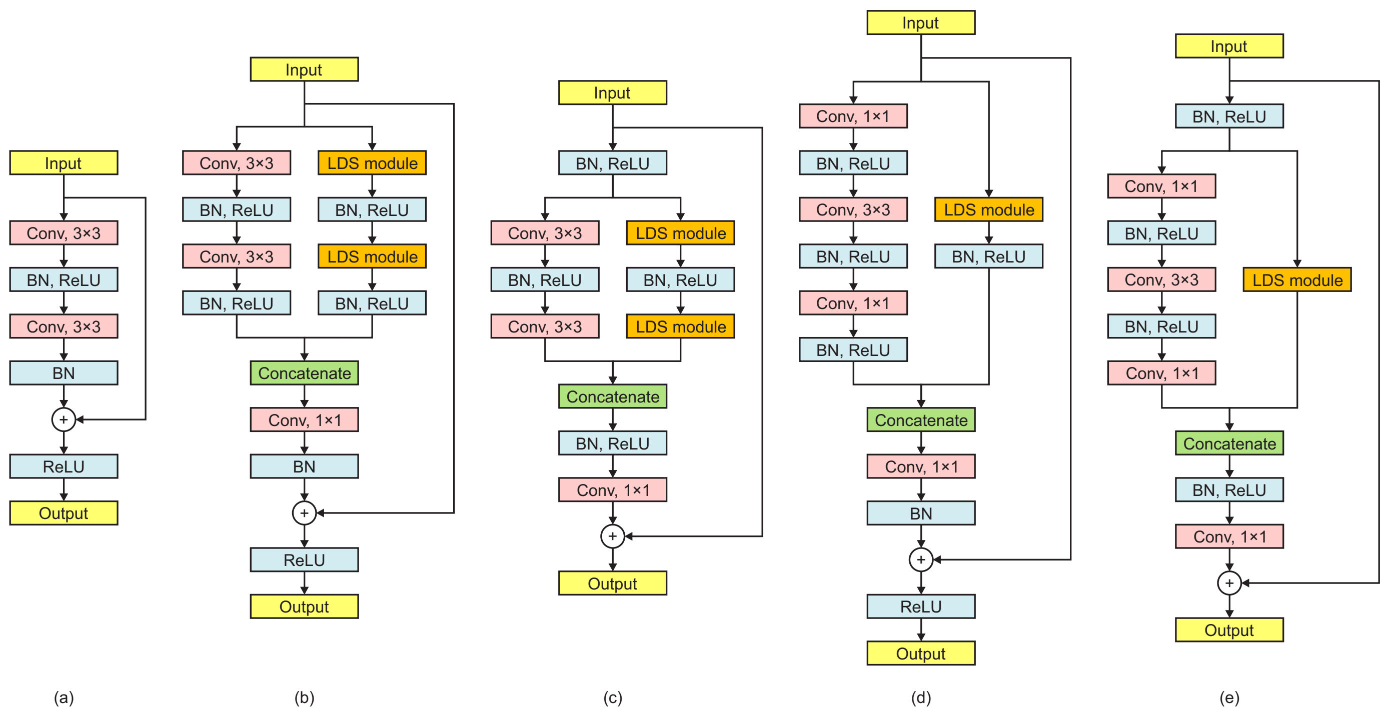


Fig. 4. Block architectures: a) baseline residual, b) baseline LDS, c) LDS with full pre-activation, d) bottleneck LDS, e) bottleneck LDS with full pre-activation

paths. The identity shortcut is added after the concatenation and dimensionality reduction, following the same rules as in the residual case.

3) Pre-activated LDS block: In [23], authors suggested that moving the batch normalization and activation layers of a residual block before the convolutional layers can have a positive impact on the performance of a CNN. This pre-activation scheme has been employed in the LDS block to explore the complementarity of the two methods. Fig. 4c shows how pre-activation was applied to the LDS block. In this case, the ReLU layer that followed the addition is moved before the 1×1 convolutional layer. This can prove beneficial in deeper networks, where such blocks are usually stacked, as information can be freely propagated through the shortcuts, without being altered by intermediate layers. Pre-activation is also applied in the case of projection shortcuts.

4) Bottleneck LDS block: Employing 3×3 filters on all convolutional layers can be computationally expensive for larger input sizes and deeper networks. The bottleneck architecture, originally proposed in [1], modifies the baseline residual block by stacking three convolutional layers instead of two. The first and third convolutional layers reduce and restore the number of features maps of the input volume, using 1×1 filters. The 3×3 convolution is performed on a smaller volume, which is usually equal to $1/4$ of the input. When the number of filters for the 3×3 convolutional layers is fixed, the two alternatives have roughly the same computational complexity, but the network consisting of bottleneck blocks will be deeper. Fig. 4d shows the structure of the bottleneck LDS block. As there is only one 3×3 convolutional layer in the convolutional path, one corresponding LDS module is placed in the LDS path. Since the LDS module operates on patches along the spatial dimensions, placing more LDS modules to match the 1×1 convolutional layers would be unorthodox, as these would have to operate on 1×1 patches (both the patch and the transition matrix A would be a scalar value).

5) Bottleneck pre-activated LDS block: Our final block type combines the LDS module with both the pre-activation and the bottleneck schemes. The resulting bottleneck pre-activated LDS block is depicted in Fig. 4e. Similar to the simple bottleneck case (Fig. 4d), a single LDS module is used, while batch normalization and ReLU layers are moved before convolutional layers, as in Fig. 4c.

IV. Experimental Results

In this section, the above described architectures are validated in different datasets and applications. Initially, the proposed networks are compared in the task of classification in 3 different datasets, namely CIFAR-10, CIFAR-100 and ImageNet. Experiments are performed to assess the performance of each residual block in multiple network depths. The effect of the parameter number to the network performance is also examined. Subsequently, the proposed network is applied for object detection purposes on 2 different datasets, namely VOC and MOT Challenge 2017.

We utilized Torch7 [24] and TensorFlow 1.4.0 [25] for all classification and detection experiments respectively. We also used the the cuDNN library [21] to further reduce processing times. Experiments were performed on a computer equipped with an NVIDIA GTX Titan X graphics card, CUDA v8.0.44 and CuDNN v5.1.5.

TABLE I
CNN ARCHITECTURE FOR CIFAR EXPERIMENTS

layers	# filters	stride	output size
input			$32 \times 32 \times 1$
convolution, 3×3	16		$33 \times 32 \times 16$
BN, Relu			
LDS block $\times$ k	16		$32 \times 32 \times 16$
LDS block $\times$ k	32	2	$16 \times 16 \times 32$
LDS block $\times$ k	64	2	$8 \times 8 \times 64$
average pooling, 8×8			$1 \times 1 \times 64$
fully connected			10 or 100

A. CIFAR classification

The CIFAR-10 benchmark dataset [26] consists of 60,000 32×32 RGB images, classified into 10 classes. The dataset is split in 50,000 training and 10,000 test samples. Both training and test sets contain equal number of instances from each class. The CIFAR-100 dataset comprises the same images as CIFAR-10, except that they are divided into 100 finer classes.

We utilized the CIFAR datasets to study the effect of different block architectures and LDS module variants. We considered three different network depths for experiments using the baseline LDS block: 20, 44 and 110 layers. Additional experiments were performed with the pre-activated and bottleneck LDS blocks for the deeper network structures.

For all CIFAR experiments, we selected the overall CNN architecture that was used by [1], which is also presented in Table I. A 2-D convolutional layer with 16 3×3 filters is placed first, followed by batch normalization and ReLU layers. The main network consists of 3 block groups with a total of $3k$ blocks. The number of filters used for convolutional operations is 16, 32 and 64, respectively. The second and the third block groups reduce the spatial dimensions by applying a stride of 2 in the first convolutional layers of their first blocks. Setting the value of k to 3, 7 and 18 leads to 20, 44 and 110-layer CNNs, respectively. For bottleneck architectures, $k = 18$ leads to a 164-layer deep CNN, as each bottleneck block has three convolutional layers instead of two. Finally, a 8×8 average pooling layer reduces the output size to a vector of size 64 and a fully connected layer produces the final class scores, which are 10 or 100 for CIFAR-10 and CIFAR 100 respectively.

For non-bottleneck blocks, when the output of a block has different dimensions compared to its input, the identity shortcut is replaced by a pooling and/or padding operation (option A). For bottleneck blocks, a projection shortcut (option B) is used instead. When pre-activated blocks are used, additional batch normalization and ReLU layers are placed after the third block group. In this case, in the very first block of the CNN, the batch normalization and ReLU layers placed before the two paths in Fig. 4c and 4e are omitted.

To enable fair comparison, we used the same training parameters as in [1] for all CIFAR experiments. Thus, training lasted for a total of 164 epochs, which is approximately 64k iterations with a batch size of 128 images. Initial learning rate was set to 0.1 and divided by a factor of 10 twice, at the beginning of epochs 82 and 122. CNNs were trained using Softmax loss and Stochastic Gradient Descent (SGD), with a momentum of 0.9 and a weight decay of 0.0001 was used for all trainable parameters. All images were normalized by subtracting the mean and dividing by the standard deviation computed over the entire training set. For data augmentation, training images were zero padded by 4 pixels along each direction and a random 32×32 patch was selected at each iteration. Additionally, we used random horizontal flipping with a 0.5 probability.

TABLE II
CIFAR-10 RESULTS (BASELINE AND PRE-ACTIVATED BLOCKS)

depth	residual	LDS-1a	LDS-1b	LDS-2
20	8.75 [1]	6.82	**6.75**	7.41
44	7.17 [1]	**6.24**	6.45	7.03
110	6.61 [1]	**5.74**	7.11	6.10
110-pre	6.37 [23]	5.48	**5.36**	5.73

TABLE III
CIFAR-100 RESULTS (BASELINE AND PRE-ACTIVATED BLOCKS)

depth	residual	LDS-1a	LDS-1b	LDS-2
20	31.92	29.46	**29.30**	29.99
44	29.49	27.88	**27.36**	28.13
110	28.62	26.18	**26.15**	26.66
110-pre	26.92	25.47	**24.79**	25.70

The following notations are used in Tables II to VI: *residual* refers to simple residual blocks, *LDS-1a* to LDS blocks where a single trainable transition matrix A is used for all feature maps, *LDS-1b* to LDS blocks with different trainable A matrices for each feature map and *LDS-2* to LDS blocks were A is calculated using (7). CNNs with pre-activated blocks are denoted with a *-pre* suffix after their depth.

1) Baseline LDS results: Table II presents the error rate in the CIFAR-10 test set for non-bottleneck ResNet and LDS-Resnet architectures. We observe that in most cases, CNN based on LDS blocks exhibit a significantly lower error rate compared to their residual counterparts. Table III shows the experimental results in the CIFAR-100 test set. In these experiments, all LDS blocks performed significantly better than the residual ones. Error rates with residual blocks were obtained from our experiments using the same experimental setup, as there were no relevant results for non-bottleneck architectures in [1] and [23].

The introduction of the additional LDS path increases the width and doubles the number of trainable parameters in LDS-ResNets. To investigate the possible impact on performance, we performed two sets of ablation experiments, following different approaches to construct alternative non-LDS models with the same depth and number of paramaters as LDS-ResNets. In the first one, the original ResNets were widened by using more filters in each convolutional layer. In particular, to double the overall number of trainable parameters, the number of filters in each convolutional layer was multiplied by $\sqrt{2}$. In the second approach, the LDS modules in Fig. 4b were substituted with standard convolutional layers, resulting in 3-branch inception-like residual blocks, composed of two identical convolutional branches (which were concatenated in the same way) and the shortcut connection.

A performance comparison is presented in Table IV. It can be seen that in deeper networks (44 and 110 layers),

TABLE IV
COMPARISON BETWEEN LDS-RESNET AND ALTERNATIVE RESNET ARCHITECTURES WITH SIMILAR NUMBER OF PARAMETERS (CIFAR-10)

	ResNet [1]		widened ResNet		3-branch ResNet		LDS-ResNet	
depth	params	error	params	error	params	error	params	error
20	0.27 M	8.75	0.55 M	6.94	0.57 M	6.85	0.57 M	**6.82**
44	0.66 M	7.17	1.36 M	7.58	0.39 M	6.63	1.39 M	**6.24**
110	1.73 M	6.61	3.56 M	6.75	3.65 M	6.35	3.65 M	**5.74**

TABLE V
CIFAR-10 RESULTS (BOTTLENECK BLOCKS)

depth	residual [23]	LDS-1a	LDS-1b	LDS-2
164	5.93	4.57	**4.27**	4.40
164-pre	5.46	4.49	**4.39**	4.61

which are the ones that also achieve the lowest error rates, there are no performance benefits when widening the original ResNets. On the other hand, LDS-ResNets outperforms both alternative architectures by an increasing margin as the depth of the network increases. Thus, the performance benefits can be mostly attributed to the proposed LDS module, rather than the increase of the network's width and parameters.

2) Pre-activated LDS results: Results with pre-activated blocks are presented in Tables II and III for the 110-layer CNNs. In both datasets, the pre-activated block exhibits better performance than the baseline blocks, both for residual and LDS blocks. Overall, using non-bottleneck blocks, the best performing CNN in CIFAR-10 has an error rate of 5.36 (20% lower compared to its residual equivalent) and is 110-layer deep, composed of pre-activated LDS blocks with different transition matrices for each feature map (LDS-1b). Similarly, in CIFAR-100, the best performing CNN reduces the error rate from 26.92% to 24.79% using the same type of LDS blocks.

3) Bottleneck LDS results: Tables V and VI depict the error rates using bottleneck blocks, in CIFAR-10 and CIFAR-100 test sets, respectively. It can be seen that all LDS block variants outperform residual blocks in both datasets, despite the fact that, due to GPU memory limitations, CNNs with LDS blocks were trained with a lower batch size of 64 images. Furthermore, the error rate is significantly lower compared to non-bottleneck blocks for all block variants. On the other hand, compared to non-bottleneck blocks, pre-activation has a more limited impact on performance, especially when bottleneck LDS blocks are used. The best performing CNNs achieved an error rate or 4.27% in CIFAR-10, using LDS-1b bottleneck blocks and 20.97% in CIFAR-100, using pre-activated LDS-2 bottleneck blocks.

4) Combination and comparison with other ResNet extensions: A key advantage of the proposed LDS module is that, besides improving ResNet performance, it can be used complementary to other ResNet extensions and related methodologies.

TABLE VI
CIFAR-100 RESULTS (BOTTLENECK BLOCKS)

depth	residual [23]	LDS-1a	LDS-1b	LDS-2
164	25.16	22.38	21.84	**21.24**
164-pre	24.33	21.96	21.53	**20.97**

TABLE VII
TEST ERROR COMPARISON IN CIFAR-10/100

method	configuration	params	CIFAR-10	CIFAR-100
ResNet [1]	110-layer	1.7 M	6.61	28.62
Stochastic Depth [9]	110-layer, *drop rate* = 0.5 (linear decay)	1.7 M	5.25	24.98
Squueze-Excitation [13]	110-layer, *compression rate* = 1 (our implementation)	1.9 M	6.10	26.08
DenseNet [11]	100-layer, *growth rate* = 12	7.0 M	4.10	20.20
Wide ResNet [6]	40-layer, 4× wide	8.9 M	4.53	21.18
ResNet In ResNet [12]	18-layer + wide RiR	10.3 M	5.01	22.90
LDS	110-layer, baseline LDS blocks	3.7 M	5.74	26.15
LDS + stochastic	110-layer, *drop rate* = 0.5 (linear decay)	3.7 M	4.07	21.44
LDS + SE	110-layer, *compression rate* = 1	3.9 M	5.56	25.13
LDS + stochastic + SE	110-layer, *drop rate* = 0.5 (linear decay), *compression rate* = 1	3.9 M	3.94	20.77
LDS + stochastic + SE (wider)	same as the above but starting with 21 channels	6.7 M	3.90	20.03
LDS + stochastic + SE (wider)	same as the above but starting with 22 channels	7.3 M	**3.87**	**19.56**

Results reported in subsections IV-A2 and IV-A3 highlight the complementarity of the LDS module with pre-activated and bottleneck CNN architectures. In this subsection, we build upon baseline LDS-ResNets and assess them in combination with the Stochastic Depth [9] and Squeeze-Excitation (SE) [13] methods. We also compare the performance of these composite models with other ResNet extensions, selecting models with similar number of parameters when possible.

Table VII presents the experimental results. For non-LDS models, results are reported as they appear in the respective publications, except for the SE method, where they correspond to our implementation of the method (optimized for the *compression rate* hyperparameter), as no results for the CIFAR datasets were available in the paper. Whenever stochastic depth was applied, we followed the training schedule used in [9].

As can be seen, the addition of stochastic depth and SE alone significantly improves over both the respective non-LDS methods and the baseline LDS results. Nonetheless, stochastic depth has a greater impact than SE, achieving a test error of 4.07 in CIFAR-10, which already compares favorably to the rest of non-LDS ResNet extentions. Combining both stochastic depth and SE with LDS further reduces the error in both datasets. While not yet achieving best result in CIFAR-100, it is important to note that the "LDS + stochastic + SE" model has significantly less trainable parameters (3.9 M) than DenseNet (7.0 M). Finally, we trained two wider models with approximately the same number of parameters as DenseNet, which reduced the CIFAR-100 test error to 19.56, thus surpassing in performance all entries of Table VI.

5) Training time: Training duration depends on the depth of the network and the usage of bottleneck blocks. For CNNs with baseline LDS blocks, training lasted on average for 2.4, 5.6 and 14.2 hours, for the 20, 44 and 110-layer models respectively. The 164-layer models with bottleneck LDS blocks, required 33.1 hours to complete the training process. As pre-activated blocks are composed of the same elements as regular ones and only alter the sequence of operations within each block, their introduction has only a negligible effect on training time.

B. ImageNet classification

While experimentation with the CIFAR dataset provided the means to explore different architectures and to evaluate their performance, its very small sized images (32 × 32 pixels) prohibited us from utilizing the trained networks in real-world applications, such as the detection scenarios presented in section IV-C, where higher resolution inputs are common. To that end, we utilized the ImageNet dataset [2], both to evaluate the classification capabilities of our method on higher resolution images and to use the trained models as a basis for our detection experiments.

The ImageNet 2012 classification dataset consists of 1.28 million training and 50k validation images of variable size, classified into 1000 classes. Each class contains 732-1300 training and exactly 50 validation images. The average image resolution is 469 × 387 pixels.

Since training with such a large dataset can be time consuming, we extracted a subset of the ImageNet dataset by randomly selecting 500 classes and half of the training images for each selected class. All validation images from the 500 selected classes were used for testing. Thus, the selected subset has a total of 320k training and 25k test images. We performed experiments on the same subset using both residual and LDS blocks, allowing for fair comparison between them.

We chose the overall structure of ResNet-50 with bottleneck residual blocks as the baseline architecture for ImageNet experimentation. Then, all residual blocks were replaced by bottleneck LDS blocks (Fig. 4d) with different transition matrices A for every feature map (variant LDS-1b), which were the ones that achieved the best results in CIFAR-10. The resulting architecture is presented in Table VIII. The main network consists of four block groups, with different number of blocks and feature maps. Strides are always applied to the last block of the group. In these blocks, a max pooling layer with 1 × 1 filters and stride 2 is added replacing the identity shortcut. When the number of feature maps changes between blocks, projection shortcuts are used.

Preprocessing of training images involved random horizontal flipping with 0.5 probability, scaling with the smaller edge to equal a random size taken from a uniform distribution in the $[256, 512]$ range and finally, cropping a random 224×224 patch. Test images were scaled with their smaller edge to be exactly 256 pixels and a single 224 × 224 patch was later cropped from the center of the scaled image. Mean RGB value was subtracted from both training and test images.

Aiming to reduce the required training time, rather than maximizing the classification performance of the network, we limited the training procedure to 50 epochs, with the initial learning rate of 0.1 divided by 10 twice at the beginning of epochs 21 and 36. Due to the larger input size and the increased number of intermediate feature maps, a lower batch size of 32 was used to fit the LDS models in GPU memory. As in CIFAR experiments, all CNNs were trained using Softmax loss and SGD with 0.9 momentum and 0.0001 weight decay. Training on ImageNet, which was the most computationally expensive task of our experiments, lasted for almost 17 days, using a dual GPU setup with an additional NVIDIA Tesla K40c.

Table IX shows the top-1 and top-5 classification error in the selected test set. For the top-1 error, an image is considered to be classified correctly only when the true label matches the top scoring output class, whereas for the top-5 error, the true label should be among the five classes with the higher scores. LDS blocks reduce both top-1 and top-5 errors by 15% and 24% respectively. Fig. 5 presents the convergence of the networks during training. It can be seen that the CNN containing LDS blocks converges faster and achieves a lower training loss.

TABLE VIII
CNN architecture for ImageNet experiments

layers	# filters	stride	output size
input			224 × 224 × 1
convolution, 7 × 7	64	2	112 × 112 × 64
BN, Relu			
max pooling, 3 × 3		2	56 × 56 × 64
bottleneck LDS block ×3	256	2	28 × 28 × 256
bottleneck LDS block ×4	512	2	14 × 14 × 512
bottleneck LDS block ×6	1024	2	7 × 7 × 1024
bottleneck LDS block ×3	2048		7 × 7 × 2048
average pooling, 7 × 7			1 × 1 × 2048
fully connected			500

TABLE IX
ImageNet results

	residual	LDS-1b
top-1 error	32.29	**27.38**
top-5 error	11.97	**9.06**

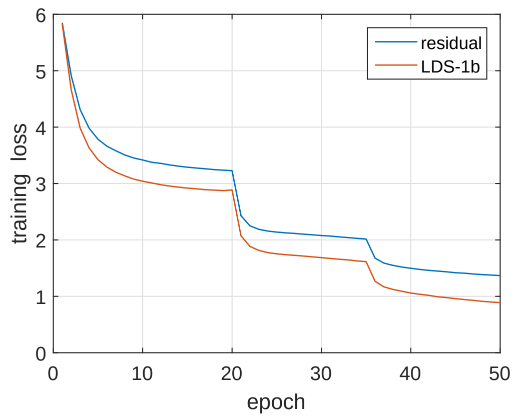


Fig. 5. Convergence comparison in ImageNet classification using residual and LDS-1b bottleneck blocks.

C. Object detection

LDS-ResNets can improve the performance of ResNet-based methods methods in any domain. In this subsection, we evaluate LDS-ResNets in object detection. We have to note that experiments reported here aim to assess the effectiveness of the proposed method, which corresponds to a generic methodology, in an additional domain, rather than designing an architecture tailored to this specific task and aiming to achieve state-of-the-art results in a specific dataset.

Faster-RCNN [27] is a widely used architecture for object detection, using VGG or ResNet as a base network for region proposal generation and labelling the detected objects. First, the base network receives the input image and outputs a set of feature maps. Then, based on this intermediate representation, a class agnostic Region Proposal Network (RPN) suggests possible object locations. Finally, the region proposals with the higher objectness scores are forwarded to the classifier, which assigns a class label to each one of them and refines the bounding box. Its modular architecture makes it ideal for testing new neural network models.

Faster-RCNN was selected to evaluate the performance improvement when the base network is replaced with LDS-ResNet. The pre-trained LDS-ResNets with the ImageNet dataset (as explained in section IV-B) were utilized as the base network of Faster-RCNN, including all layers of Table VIII up to the third block group. The Region Proposal Network (RPN) was placed between the third and the fourth block groups, leaving the final three blocks along with the average pooling and fully connected layers to be part of the classifier.

We utilized the PASCAL VOC [28] and the MOT2017Det [29] datasets. The PASCAL VOC detection dataset comprises variable sized RGB images with multiple ground truth bounding boxes of objects over 20 classes. As is common practice, we used the VOC2007 and the VOC2012 training and validation sets for training purposes, with a total of 16551 images containing 40058 objects. For evaluation, we used the VOC2007 test set with 4952 images and 12032 objects. The MOT2017Det training set comprises 7 videos with a total of 5316 frames, captured by static or moving cameras in various environmental conditions. Each frame contains on average 21.1 pedestrians, leading to over 112k ground truth bounding boxes in total. We split this set equally into two subsets, taking the first half of each video for training and the second half for testing.

Training was performed in an end-to-end manner by combining classification and box regression losses from the RPN and the classifier modules. Non Maximal Suppression (NMS) with an Intersection over Union (IoU) of 0.7 was utilized to reduce the number of highly overlapping region proposals at the RPN output. Training was performed using a single image as input to Faster-RCNN and a sampled subset of 128 region proposals as input to the classifier module, containing equal amount of objects and background cases. The 7×7 convolutional layer at the beginning of the network and the first block group were kept fixed during training. Also, all batch normalization layers were set to inference mode. Our Faster-RCNN implementation uses TensorFlow and is based on [30].

TABLE X
FASTER-RCNN DETECTION RESULTS (MAP) USING RESNET AND LDS-RESNET BASE NETWORKS

dataset	residual	LDS-1b
PASCAL VOC	67.87	**73.34**
MOT2017Det	68.31	**69.51**

All images were resized with their shortest side to equal 600 pixels. Training datasets were augmented with horizontally flipped images. Using a single image as input, training lasted for 110k and 70k iterations for the PASCAL VOC and the MOT2017Det datasets respectively. Initial learning rate of 0.001 was reduced by a factor of 10 after 80k or 50k iterations. Trainable parameters were optimized using SGD with 0.9 momentum and a weigh decay of 0.0001. Training with the PASCAL VOC dataset required 74 hours, while the MOT2017Det dataset required 47 hours.

During testing, the top scoring 300 region proposals (after NMS) were forwarded to the classifier. Using this configuration, the proposed architecture operated at a rate of 0.8 images per second in both datasets. Following the official PASCAL VOC evaluation rules, an IoU higher than 0.5 between the detection and the corresponding ground truth bounding boxes was required for a detection to be counted as a True Positive. Additionally, each ground truth box could be associated at most once with a detection box, counting multiple detection of the same object as False Positives and any remaining non-associated ground truth boxes as False Negatives. Evaluation metrics in the MOT2017Det dataset were obtained according to challenge rules using the provided evaluation scripts.

Detection performance was quantified by Average Precision (AP). For each examined class, AP was obtained from the Precision-Recall curve by averaging the computed precision at 0.1 recall intervals. Table X presents the mean Average Precision (mAP) in both datasets, using Faster-RCNN with residual and LDS-1b blocks. Fig. 6 shows the AP for each individual class in the PASCAL VOC dataset. It can be seen that LDS blocks outperform residual ones in every single class, leading to an overall performance boost of 5.51% in terms of mAP. Fig. 7 compares the Precision-Recall curves for the pedestrian class in the MOT2017Det dataset, where LDS blocks also lead to superior performance. Detection examples in the PASCAL VOC and MOT2017Det datasets are presented in Fig. 8 and 9 respectively. The results presented in this subsection are indicative of the expected performance benefit in other methods that employ Faster-RCNN, if the proposed LDS-ResNet is utilized as the base network.

V. CONCLUSION

An extension of Residual Networks has been proposed inspired by Linear Dynamical Systems modelling. The residual information content is transformed in a hidden state mimicking an LDS system. This transformation is learnt based on the statistics of the training content and, therefore, it leads to a more generic representation of the information content that we claim and prove that it can be easier learnt. Therefore,

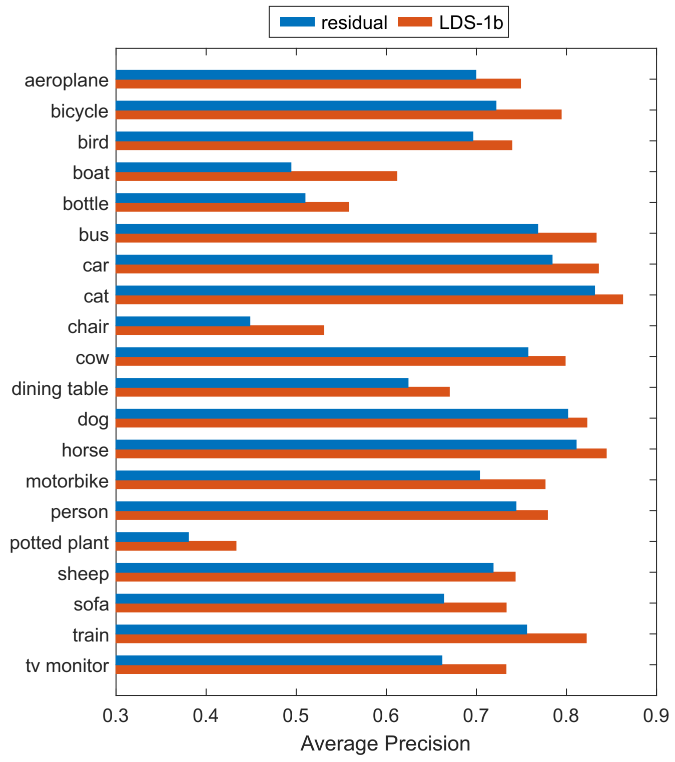


Fig. 6. Average Precission per class in the VOC2007 test set

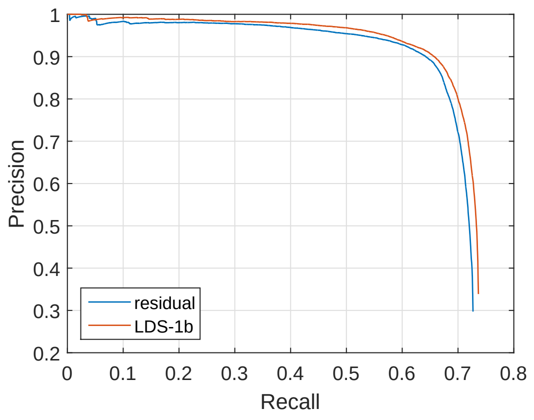


Fig. 7. Precision-Recall curves in the MOT2017Det test set.

the proposed LDS module essentially breaks down the learning procedure in a content-based hierarchical way. The skip connection transfers the whole image, leaving the residual modelling to the other two branches. The LDS branch is learning a generalized version of the residual information and the CNN branch is learning the residual of the residual information, which is even more sparse than the original residual information and, thus, easier to learn.

An exploration of the network architecture space has been presented to assess the performance of the LDS module in multiple depths and in conjunction with other ResNet-based methodologies and extensions. The proposed LDS blocks can be utilized in any architecture that employs residual blocks, simply by replacing them. Experiments on image classification and object detection using the CIFAR-10/100, ImageNet, PASCAL VOC and MOT2017Det datasets have demonstrated the outstanding performance and robustness of the proposed approach.

Acknowledgment

This work was supported by the European Project: SURVANT http://survant-project.eu/ Grant no. 720417 within the H2020 FTIPilot-2015.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[4] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, “The microsoft 2016 conversational speech recognition system,” in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5255–5259.

[5] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[6] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, “Going deeper with convolutions.” Cvpr, 2015.

[8] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning.” in *AAAI*, vol. 4, 2017, p. 12.

[9] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep networks with stochastic depth,” in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[11] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, no. 2, 2017, p. 3.

[12] S. Targ, D. Almeida, and K. Lyman, “Resnet in resnet: generalizing residual architectures,” *arXiv preprint arXiv:1603.08029*, 2016.

[13] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” *arXiv preprint arXiv:1709.01507*, 2017.

[14] A. Ravichandran, R. Chaudhry, and R. Vidal, “Categorizing dynamic textures using a bag of dynamical systems,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 342–353, 2013.

[15] G. Luo, S. Yang, G. Tian, C. Yuan, W. Hu, and S. J. Maybank, “Learning human actions by combining global dynamics and local appearance,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 12, pp. 2466–2482, 2014.

[16] K. Dimitropoulos, P. Barmpoutis, and N. Grammalidis, “Higher order linear dynamical systems for smoke detection in video surveillance applications,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 5, pp. 1143–1154, 2017.

[17] K. Dimitropoulos, P. Barmpoutis, A. Kitsikidis, and N. Grammalidis, “Classification of multidimensional time-evolving data using histograms of grassmannian points,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 4, pp. 892–905, April 2018.

[18] A. B. Chan and N. Vasconcelos, “Classifying video with kernel dynamic textures,” in *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*. IEEE, 2007, pp. 1–6.

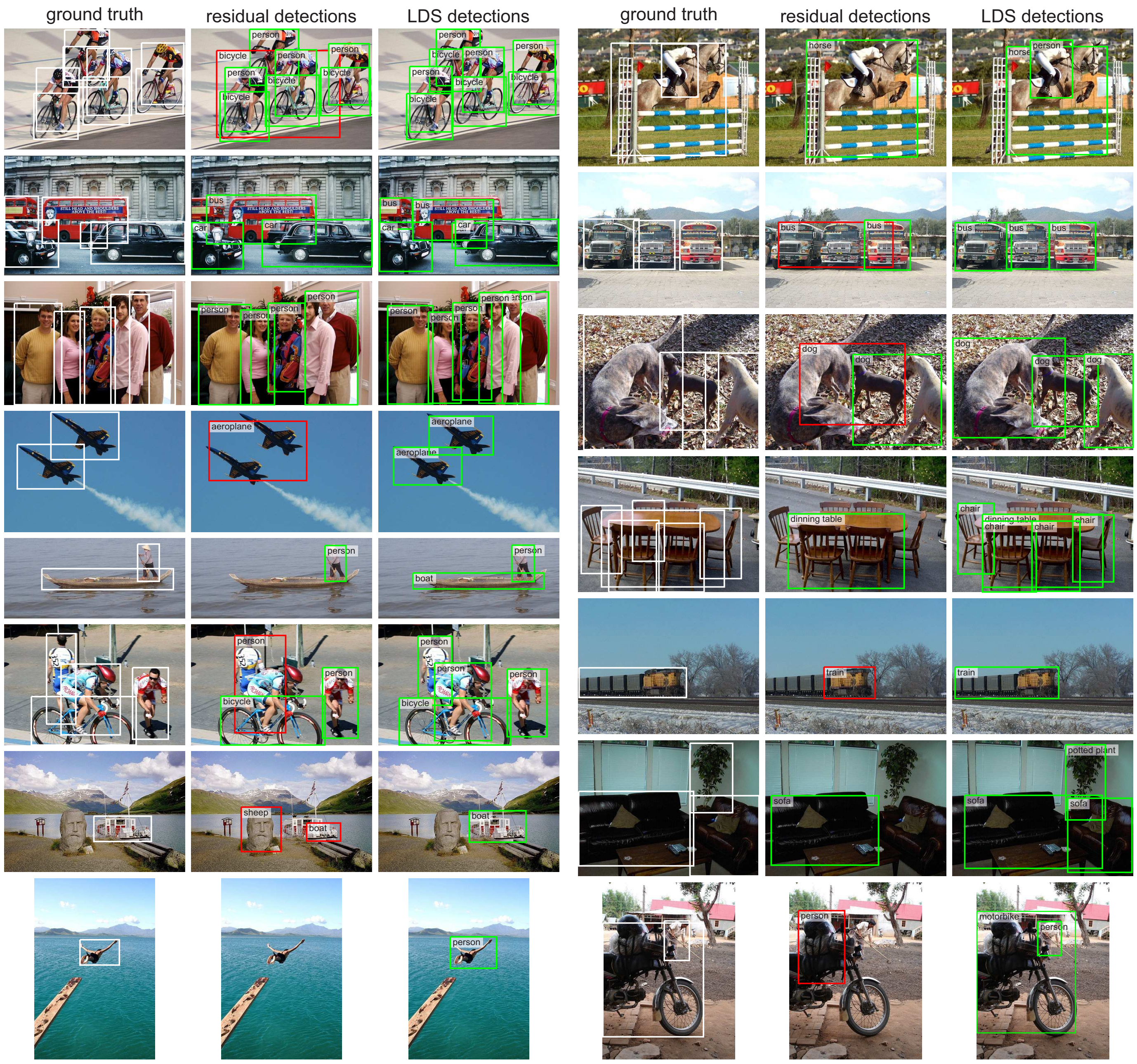


Fig. 8. Detection examples in the PASCAL VOC dataset. Ground truth boxes are depicted in the left images, detections using Faster-RCNN with residual blocks in the middle and detections using LDS blocks in the right ones. Detections counted as True Positives are presented with green bounding boxes, whereas False positives (either due to wrong class label or low IoU value with the ground truth) are presented with red. To obtain these detections, score thresholds have been optimized independently for each ground truth class and block type by maximizing the sum of precision and recall.

[19] G. Zhou, N. Dong, and Y. Wang, "Non-linear dynamic texture analysis and synthesis using constrained gaussian process latent variable model," in *Circuits, Communications and Systems, 2009. PACCS'09. Pacific-Asia Conference on*. IEEE, 2009, pp. 27–30.

[20] K. Dimitropoulos, P. Barmpoutis, C. Zioga, A. Kamas, K. Patsiaoura, and N. Grammalidis, "Grading of invasive breast carcinoma through grassmannian vlad encoding," *PloS one*, vol. 12, no. 9, p. e0185110, 2017.

[21] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.

[24] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS workshop*, no. EPFL-CONF-192376, 2011.

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[26] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[28] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[29] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A

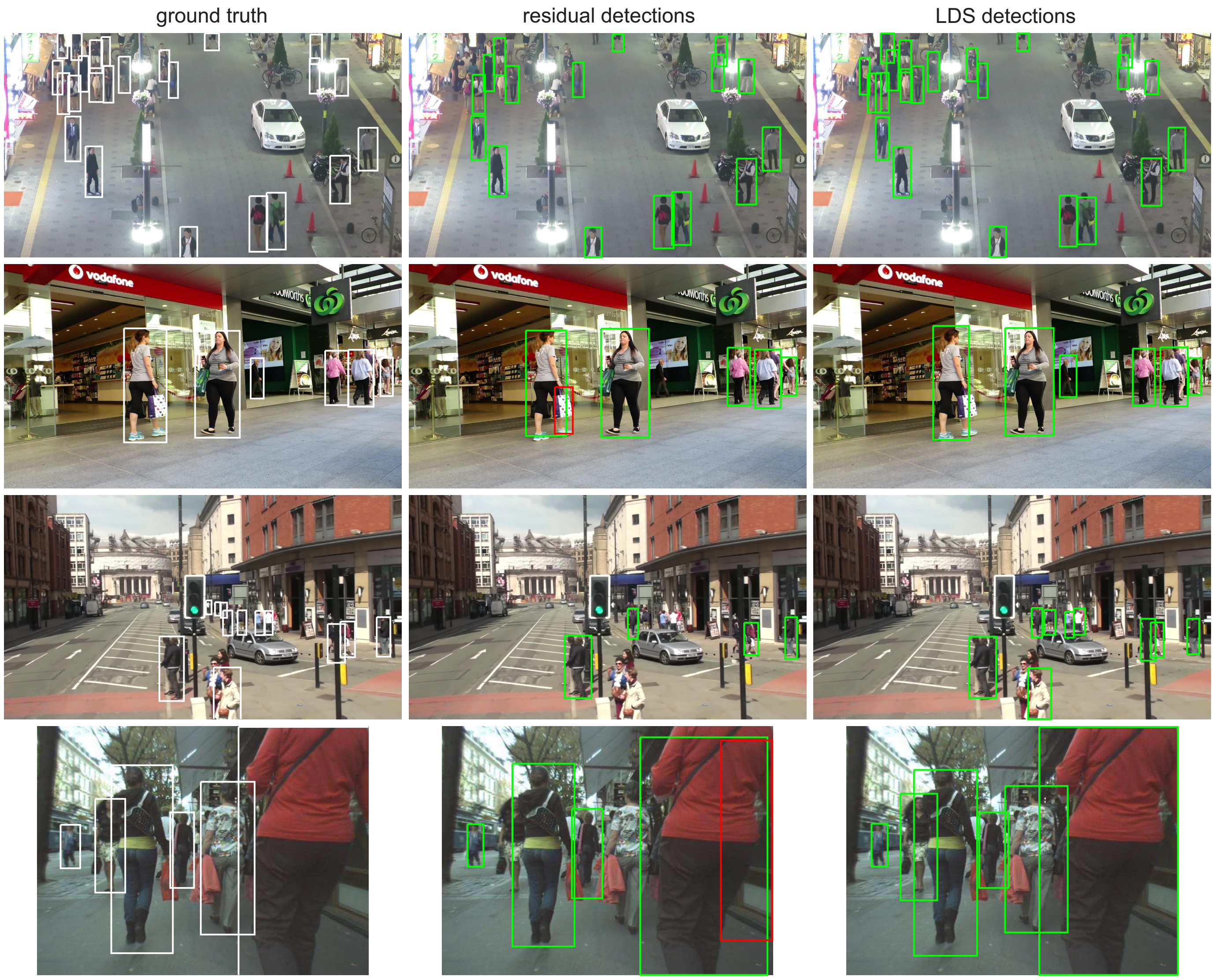
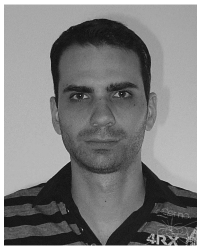


Fig. 9. Detection examples in the MOT2017Det dataset. Ground truth boxes are depicted in the left images, detections using Faster-RCNN with residual blocks in the middle and detections using LDS blocks in the right ones. Detections counted as True Positives are presented with green bounding boxes, whereas False positives (either due to wrong class label or low IoU value with the ground truth) are presented with red. To obtain these detections, score thresholds have been optimized independently for each block type by maximizing the sum of precision and recall.

benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[30] X. Chen and A. Gupta, "An implementation of faster rcnn with study for region sampling," *arXiv preprint arXiv:1702.02138*, 2017.

Anastasios Dimou graduated from Aristotle University of Thessaloniki as an Electrical and Computer Engineer (Dipl.) in 2004 and was awarded a Professional Doctorate in Engineering (PDEng) in Information and Communication Technology at the Technical University of Eindhoven, the Netherlands in 2007. Since 2015, he is a PhD candidate at Universidad Politecnica de Madrid. From February 2008 he is working as a Research Associate in the Information Technologies Institute (ITI) of the Centre for Research and Technology Hellas (CERTH). His research interests are covering image and video processing and analysis, smart security applications, semantic web technology, video coding and broadcasting quality. He has participated in numerous research projects and he co-authored 25+ publications in refereed journals, book chapters and papers in international conferences.



Dimitrios Ataloglou is an Electrical and Computer Engineer with a MSc in Medical Informatics. He received his diploma in 2014 from the Faculty of Engineering of Aristotle University of Thessaloniki and his MSc in 2017 form the School of Medicine of the same university. He is a member of the Technical Chamber of Greece. He has previously worked as a Computer Programmer/Analyst. Since May 2017, he has been working as a Research Associate at the Information Technologies Institute (ITI) of the Center for Research and Technology Hellas (CERTH). His research interests include deep learning methods, image processing and biomedical applications.

Kosmas Dimitropoulos is a research fellow at Visual Computing Lab of Information Technologies Institute - Centre for Research and Technology Hellas (ITI-CERTH) and an academic faculty member of AIMove (Artificial Intelligence and Movement in Industries and Creation) Post-Master Programme at MINES ParisTech University. He holds a diploma in Electrical and Computer Engineering and a Ph.D. degree in Applied Informatics. His main research interests lie in the fields of multi-dimensional data modelling and analysis, human computer interaction, virtual reality and serious games. His involvement with these research areas has led to the co-authoring of more than 100 publications in refereed journals and international conference proceedings. He has participated in several European and national research projects and has served as a regular reviewer for a number of international journals and conferences. He is a member of IEEE and the Technical Chamber of Greece.

Federico Alvarez (M'07) received the Telecom Engineer degree (Hons.) in 2003 and Ph.D. degree (cum laude) in 2009, both by "Universidad Politécnica de Madrid" (UPM) where he is currently working as assistant professor. From 2003 he is member of the research group "Visual Telecommunications Applications group" (GATV) of UPM. Dr. Álvarez current interests include multimedia processing and analytics, networked media and multi-sensor fusion. He had participated in national and international standardization groups and he co-authored 20 papers in refereed journals, 10+ books and book chapters and 40+ papers in international conferences.

Petros Daras is a Research Director at the Information Technologies Institute of the Centre for Research and Technology Hellas. He received the Diploma in Electrical and Computer Engineering, the MSc degree in Medical Informatics and the Ph.D. degree in Electrical and Computer Engineering all from the Aristotle University of Thessaloniki, Greece in 1999, 2002 and 2005, respectively. He is the head researcher of the Visual Computing Lab coordinating the research efforts of more than 50 scientists. His research interests include 3D media processing and compression, multimedia indexing, classification and retrieval, annotation propagation and relevance feedback, bioinformatics and medical image processing. He has co-authored more than 200 papers in refereed journals and international conferences, and has been involved in more than 50 national and international research projects.