SAN : Scale-Space Attention Networks

Yash Garg
School of CIDSE
Arizona State University
Tempe, Arizona, USA
ygarg@asu.edu

K. Selçuk Candan
School of CIDSE
Arizona State University
Tempe, Arizona, USA
candan@asu.edu

Maria Luisa Sapino
Department of Informatics
University of Turin
Turin, Italy
mlsapino@di.unito.it

***Abstract*—Deep neural networks (DNNs), especially convolutional neural networks (CNNs), have been effective in various data-driven applications. Yet, DNNs suffer from several major challenges; in particular, in many applications where the input data is relatively sparse, DNNs face the problems of overfitting to the input data and poor generalizability. This brings up several critical questions: "Are all inputs equally important?" "Can we selectively focus on parts of the input data in a way that reduces overfitting to irrelevant observations?" Recently, attention networks showed some success in helping the overall process focus onto parts of the data that carry higher importance in the current context. Yet, we note that the current attention network design approaches are not sufficiently informed about the key data characteristics in identifying salient regions in the data. We propose an innovative robust feature learning framework, *scale-invariant attention networks* (*SAN*), that identifies salient regions in the input data for the CNN to focus on. Unlike the existing attention networks, *SAN* concentrates attention on parts of the data where there is major change across space and scale. We argue, and experimentally show, that the salient regions identified by *SAN* lead to better network performance compared to state-of-the-art (attentioned and non-attentioned) approaches, including architectures such as LeNet, VGG, ResNet, and LSTM, with common benchmark datasets, MNIST, FMNIST, CIFAR10/20/100, GTSRB, ImageNet, Mocap, Aviage, and GTSDB for tasks such as image/time series classification, time series forecasting and object detection in images.**

***Index Terms*—attention module, attention networks, convolutional neural networks**

I. Introduction

Deep neural networks (DNNs), including convolutional neural networks (CNNs) have seen successful applications in many data engineering domains, such as text processing [1], [2], data alignment [3], recommender systems [4], time series search and processing [5]–[7], and media search and analysis [8]–[12]. More recently, CNNs' successful application in a variety of data-intensive domains has led to a shift away from feature-driven algorithms into the design of CNN architectures crafted for specific datasets and applications.

DNNs owe their success to large *depth* and *width*: this introduces a large number of trainable parameters (from tens of thousands [13] to hundreds of millions [14]) and enables learning of a rich and discriminating representation of the data [13]–[17]. However, as [18] points out, "*increase in the depth of the network can lead to model saturation or even degradation in accuracy*". More specifically, in many applications where the input data is relatively sparse, DNNs face the problems of overfitting to the input data and poor generalizability. This brings up several critical questions: "Are all inputs equally important?" "Can we selectively focus on parts of the input data in a way that reduces overfitting to irrelevant observations?" Works such as [19]–[21] has shown that salient information can help improve the model performance, both deep learning and machine learning models.

Partially funded by: NSF #1610282 (DataStorm), #1633381 (Complex Systems), #1629888 (GEARS), #1827757 (PFI-RP), and #1909555 (pCAR)

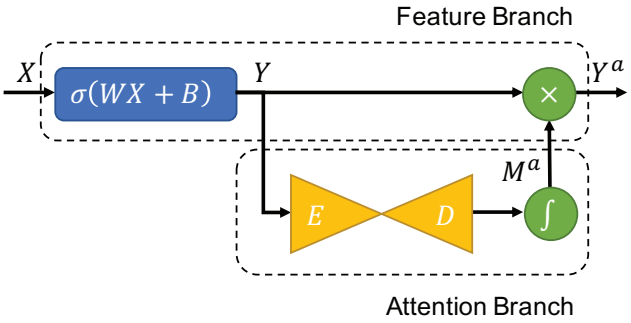


Fig. 1: Overview of conventional attention module

A. Attention Networks

The need for working with a limited number of trainable parameters to learn high performing network architectures necessitates techniques to help focus on the most relevant parts of the data. One way to achieve this is through fusion of multi-modal data characteristics, such as channel (i.e. latent) and spatial relationships in images, where information transferred from different modalities help strengthen and weaken their individual impacts [17]. Another common approach is to learn multi-scale features [16] to capture a rich representation of data. More recently, attention networks gained popularity as a more effective way to tackle this challenge [22]. Commonly, networks with attention modules have two, *feature* and *attention*, branches (see Figure 1). The feature branch is analogous to the conventional networks where the neural structure extracts features from data, whereas, in the attention branch, the network aims to quantify the importance of the input features to focus on. The attention mechanism, on the other hand, enables the network to focus on a specific, contextually-relevant, subset of the features [23].

2375-026X/20/$31.00 ©2020 IEEE
DOI 10.1109/ICDE48307.2020.00079

Attention mechanisms have been developed for different types of data. For instance, the original attention network (proposed by Bahdanau in [22]) was designed to recover attention to help identify a subset of input features important to a given state in a recurrent neural network (RNN) used for analyzing sequence data. In image analysis, on the other hand, the attention branch may aim to translate the spatial and channel level contextual relationship into an attention mask [17]. Since the introduction of the attention mechanism, there has been significant amount of work done that has enabled state-of-the-art networks, enriched with attention mechanisms, to outperform their predecessors [17], [23]–[26]. While these and other works, some of which discussed in Section II, have provided strong evidence regarding the promise of the attention mechanisms in reducing overfit and improving accuracy, we note that the current approaches suffer from a shared shortcoming: *While the existing mechanisms leverage multi-modal information, they fail to consider information that a cross-scale examination of the latent features may reveal.*

B. Contributions: Scale-Space Attention Networks (SAN)

In a recent work, [21] has shown that scale-space based approaches can be used to inform the design of CNNs – in particular, even though localized features, like SIFT, may not lead to very accurate classifiers themselves, the information these features capture at different scales might nevertheless be used to inform the design of the hyper-parameters (e.g. number of layers, number of kernels per layer) of CNNs. In this paper, we build on a similar observation and argue that a *scale-space* driven technique can also be used to design better attention mechanisms that can help *focus* attention of the deep neural network to parts of the data that are most critical.

Fundamentally, a CNN architecture extracts increasingly complex (multi-scale) features through layers of interleaved convolutional and pooling layers, coupled with non-linearity enablers, such as ReLU and tanh functions. In this paper, we show that we can adapt the CNN architecture to implement a robust scale-space based attention mechanism that focuses processing onto contextually salient aspects of the data. In particular, we propose a novel scale-space attention network, *SAN*, which brings together the following key ideas:

- **Identifying salient changes in scale-space**: Traditional attention mechanism consider layers in isolation when generating attention. In this paper, we argue that comparing and contrasting latent features from two adjacent layers, to locate salient changes in scale-space, is a more effective attention strategy.
- **Attention to extrema**: Given that a neighborhood in the input is likely to have changes in varying amplitudes, we argue that the attention should be given to extrema, where the changes in scale-space are local maxima.
- **Attention region extraction through smoothed extrema**: We translate these extrema into attention masks by applying a convolutional operation around the extrema – this helps avoid noisy artifacts in the latent representation which could adversely affect performance.

As we experimentally validate in Section IV, the proposed *SAN framework* has the following advantages: (a) *SAN* detects and describes salient changes in the latent features to identify detailed and diverse attention masks that help boost network performance while retaining finer details of the patterns. (b) *SAN* consistently performs better than the competitors in both bottleneck and full attention scenarios (see details in Section IV-C). (c) *SAN* framework is able to learn a high-performing network architecture with minimal computational overhead.

C. Organization of Paper

The following sections are organized as follow: Section II discusses current state-of-the-art approaches and their shortcomings, Section III presents the proposed framework, in Section IV we evaluate *SAN*, and in Section V we conclude.

II. Related Work

Successful application of DNNs in diverse domains [6]–[12], [27], [28] has motivated the community to devise novel network architectures that outperform the prior art.

A. Design of DNNs

A common approach to design DNNs is to hand-craft specialized network architecture for specific domain and data. As early as 1998, Lecun [29], proposed a *five* layer convolutional network to detect hand-written digits. The increasing prevalence of more complex datasets, such as ImageNet [30], led to more complex design of the hand-crafted networks [14]–[16], [31], [32]. These span from 10s to 100s of layers with hundreds of millions of trainable parameters. While these networks have different architectures, they often leverage common design optimizations that have been shown to improve the network performance. For instance, batch-normalization [33] is used to address the problem of covariate shift in the network by normalizing individual batch output of the layers to facilitate early convergence of the network; ReLU [34] is used to handle the problem of vanishing gradients by eliminating the negative gradient in the feed forward phase of the network. To help with the design of new architectures or for improving existing ones, recently several hyper-parameter search strategies have been proposed: these include, grid-search [35] and random search [36]. Both strategies perform principled hyper-parameter search and have shown to determine an optimal hyper-parameter configuration, however, they heavily rely on domain expert input to hand-craft hyper-parameter search space. In a recent work, [21] has shown that scale-space based approaches can be used to inform the design of the hyper-parameters (e.g. number of layers, number of kernels per layer) of CNNs.

B. Attention Networks

Despite these advances, traditional DNNs are still faced with the problem of performance degradation with the increase in the depth [18]: these networks tend to saturate after a certain depth and networks suffer from limited generalization of input data into a fixed-length encoding [22]. Attention

mechanisms [17], [22], [24], [25] aim to address this issue. In [22], attention is used for improving thesequence translation task, from *English* to *German*, using recurrent neural nets (RNNs). This work highlighted that not every input feature (word) in a sequence is equally important, rather focusing on a different subset of input features may be more appropriate at different stages of the translation process. Building on this observation, attention has been applied to different applications (image captioning [1], recommender systems [4], multi-task learning [5], question generation [2]) and different network architectures, including CNNs [37] and LSTMs [38]. [17] was one of the early efforts in attentioned image understanding; the authors proposed a residual attention mechanism which emulates residual learning by introducing the attention module as a residual connection comprising of an autoencoder module. Convolutional block attention module [24] and bottleneck attention module [25] proposed to leverage spatial and channel relationships in an image dataset to learn attention masks that summarize the importance of channels in the images and locate where the most information resides spatially. In this paper, we argue that these works suffer from global (rather than local) summarization and from the fact that they do not leverage cross-layer information for discovering attention. To address this shortcoming, we propose an informed attention network that leverages salient changes in latent features and transform them into rich (diverse and detailed) attention masks.

III. *SAN*: Scale-Space Attention Networks

As discussed in Section I, the success of deep neural networks can be credited to the increase in their *depth*s and *width*s thanks to modern hardware. This increase has enabled these networks to learn sufficiently complex patterns contained in the dataset. Convolution Neural Networks (CNNs), which seek multi-scale features, have proven especially successful in image and time series understanding. However, not every part of the data is of equal importance for extracting features and avoiding overfitting, especially in the presence of sparse data, necessitates the network to learn the importance (*attention*) of different parts of the data. In this section, we present a novel *scale-space attention network* (*SAN*) framework that identifies salient changes in latent features across scales and translates them into robust (detailed and diverse) attention (Figure 3).

A. Convolutional Neural Networks and Attention Modules

A convolutional neural network (CNN [37]) is a type of neural network that works by leveraging the local spatial arrangements by establishing connections among small spatial regions across adjacent layers (Figure 2).

1) Convolutional Neural Architectures: A CNN consists of several complementary components organized into layers:

- Each *convolution* layer links local-spatial data (i.e., pixels at the lowest layer) through a set of channels (or *kernel*s) that represent the local spatial features.
- Since each convolution layer operates on the output of the previous convolution layer, higher layers correspond to increasingly complex features obtained by combining lower-complexity features.
- Since relevant features of interest can be of different sizes, *pooling/subsampling* layers are introduced among convolution layers: these pooling layers carry out down-sampling of the output of a convolution layer, thereby (given a fixed kernel size) effectively doubling the size of the feature extracted by the corresponding filter.

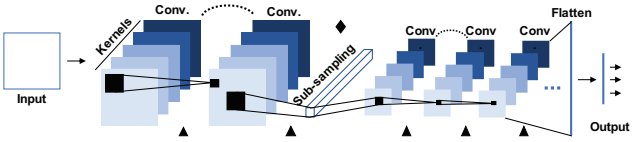


Fig. 2: Outline of a convolutional neural network [21]: a sequential arrangement of layers with localized spatial connections interleaved with pooling operations that scale the features extracted from the image. In this paper, we consider two positions for integrating the attention branches: in bottleneck (b) attention, we only attach attention modules right before subsampling (marked with ♦ in the figure); in full (f) attention, the attention modules are applied at each and every trainable layer (marked with ▲ in the figure)

Intuitively, a CNN searches for increasingly complex local features that can be used for understanding (and interpreting) the content of a dataset. Such latent features (*deep representations*) are fundamental to the success of the deep neural networks, as each layer in the network sequentially feeds on the latent features (output) of the previous layers to learn rich and abstract features. More formally, a neural network ($\mathcal{N}$) is a sequential arrangement of layers ($\mathcal{L}$), mainly convolutional and dense layers, to map the input $\boldsymbol{X}$ to output $\boldsymbol{Y}$ as follows:

$$\boldsymbol{Y} = \mathcal{N}(\boldsymbol{X}) = \mathcal{L}_L(\mathcal{L}_{L-1}(\ldots\mathcal{L}_2(\mathcal{L}_1(\boldsymbol{X})))); \quad (1)$$

here, $\boldsymbol{X} \in \mathbb{R}^{N\times D}$ and $\boldsymbol{Y} \in \mathbb{R}^{N\times O}$ where N is the number of samples, D is the dimensionality of the sample, O is the number of class labels, and L is the number of layers. Any given layer $\mathcal{L}_l$ can be generalized (*perceptron*) as,

$$\mathcal{L}_l(\boldsymbol{X}_l) = \sigma_l(\boldsymbol{W}_l\boldsymbol{X}_l + \boldsymbol{B}_l), \quad (2)$$

where $\boldsymbol{X}_l$ (the output of layer $l-1$, s.t. $\boldsymbol{X}_l = \boldsymbol{Y}_{l-1}$) is the input to the layer l (for $l = 1$, $\boldsymbol{X}_1 = \boldsymbol{X}$) and σ_l, $\boldsymbol{W}_l$, and $\boldsymbol{B}_l$ are the layer's activation function, weight, and bias respectively. Note that, if the l^{th} layer has m_l neurons and the $(l-1)^{th}$ layer has n_l neurons, then $\boldsymbol{Y}_l \in \mathbb{R}^{m_l\times 1}$, $\boldsymbol{X}_l \in \mathbb{R}^{n_l\times 1}$, $\boldsymbol{W}_l \in \mathbb{R}^{m_l\times n_l}$ and $\boldsymbol{B} \in \mathbb{R}^{m_l\times 1}$.

2) Attention Masks: As discussed in Sections I-A and II, several researchers noticed that significant amount of waste in learning and inference effort can be avoided if the attention is directed towards parts of a data that are likely to contain interesting patterns. This is achieved by attaching so called *attention modules* to this neural architecture, where the output of the attention module is used to weight the features learned in the CNN [17], [23], [24]. Such attention mechanisms have

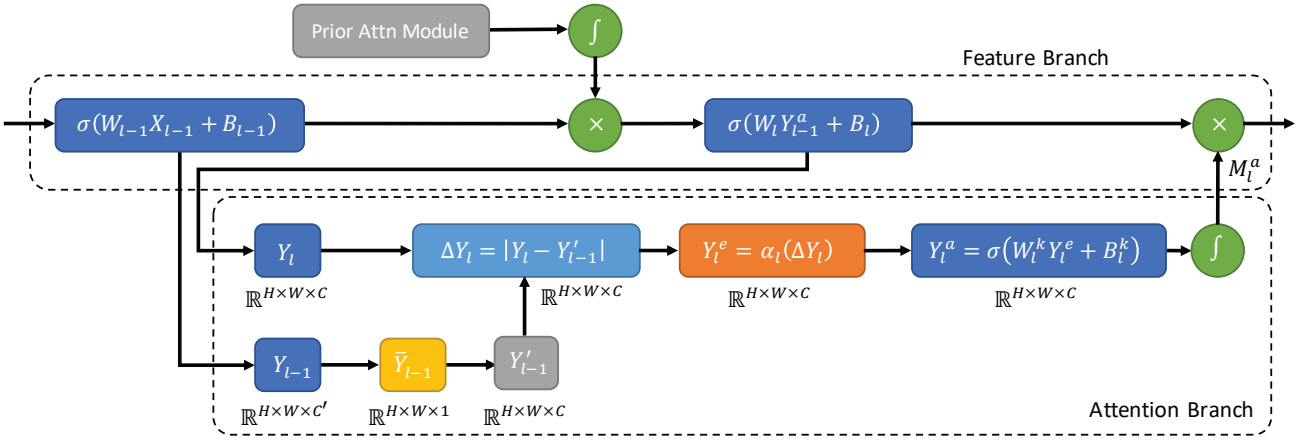


Fig. 3: Abstract overview of the proposed attention module

shown to help improve the network performance by facilitating the network with the ability to learn to *highlight* important and *suppress* unimportant features.

In CNNs, attention is achieved through the introduction of attention masks. As visualized in Figure 1, the layer contains an additional component called *attention mask* ($\boldsymbol{M}_l^a$):

$$\mathcal{L}_l^a = \mathcal{L}_l \odot \boldsymbol{M}_l^a\langle \boldsymbol{Y}_l\rangle. \tag{3}$$

Here, $\boldsymbol{M}_l^a$ highlights the important local regions in the image, and/or suppresses the unimportant regions. However, conventional attention mechanisms fail to consider information that a cross-scale examination of the latent features may reveal, and, in this paper, we argue that salient changes in the scale-space can be identified through a cross-scale examination of the latent features and the extrema in these changes can be leveraged for more effective attention masks.

B. Feature Search in Scale-Space

In the literature, there are several localized feature extraction algorithms for images: these include SURF [39], HOG [40], and SIFT [41]. In particular, SIFT has been the *de facto* representation strategy for content-based image retrieval as these features have shown robustness against rotation, scaling, and various distortions. Intuitively, each feature corresponds to a region in a given image that is *different* from its neighborhood, also in different image scales. These stable patterns are extracted through a multi-step approach, including (a)scale-space construction, (b) candidate key-point identification, (c) pruning of poorly localized, non-robust features, and (d) descriptor extraction.

The scale-space used for feature search is constructed through an iterative smoothing process, which uses Gaussian convolutions to create different versions of the input data, each with different amount of detail. Robust localized features are then located where the differences between neighboring regions (possibly in different scales) are large – in other words, these keypoints are located at the *local extrema* of the scale space defined by the difference-of-Gaussian (DoG) of the input image. More specifically, an l-layer state space of an input image, $\mathbb{I}$, is defined as a set of data matrices $\{\boldsymbol{I}_0, \ldots, \boldsymbol{I}_L\}$, where $\boldsymbol{I}_l = \boldsymbol{I}\{\sigma_0 \times k^l\}$, is a smoothed version of the input image for some smoothing parameter σ_0 and a scaling parameter $k > 1$. Given this, a DoG, $\mathbb{G}$, is created by considering a sequence of difference matrices $\{\boldsymbol{D}_0, \ldots, \boldsymbol{D}_{L-1}\}$, where $\boldsymbol{D}_l = |\boldsymbol{I}_{l+1} - \boldsymbol{I}_l|$ and feature candidates are sought at the local maxima and minima of the resulting DoG: each $\boldsymbol{D}_l[x, y]$ (where x and y are the rows and columns, respectively) is compared against its 26 ($= 3^3-1$) neighbors (spatial neighbors in the scale l and neighboring scales $l-1$ and $l+1$) and the triplet $\langle l, x, y\rangle$ is selected as a candidate only if it is close to being an extremum among these neighbors[1].

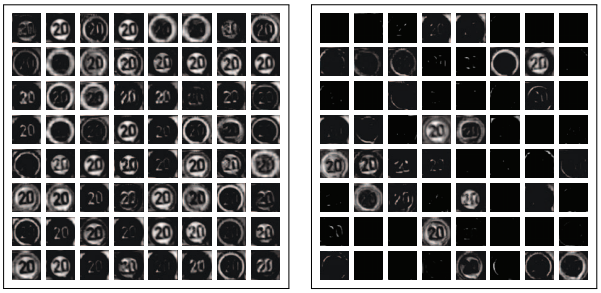
C. Scale-Space Attention Networks (SAN)

Despite their success in object recognition and image search, SIFT features described above have recently been overshadowed by CNNs in many image recognition tasks [10]–[12], [27], [28]. Yet, as discussed earlier, this advantage of CNNs are subject to several constraints: most importantly, due to the large number of parameters that need to be learned from data, CNNs require a lot of data objects for training. Features likes SIFT, on the other hand, are (a) relatively cheaper to obtain and (b) since they encode the key domain knowledge "a robust feature is one that is maximally different from its immediate neighborhood both in space and scale" algorithmically, they do not require training data. In this section, we construct a novel attention module for CNNs based on a similar observation: "*the CNN should pay special attention to latent features that are maximally different from their immediate neighborhood both in space and scale*". In particular, unlike conventional attention mechanisms (Equation 3), we propose to leverage outputs from two adjacent layers when constructing the attention module:

$$\mathcal{L}_l^a = \mathcal{L}_l \odot \boldsymbol{M}_l^a\langle \boldsymbol{Y}_l, \boldsymbol{Y}_{l-1}\rangle. \tag{4}$$

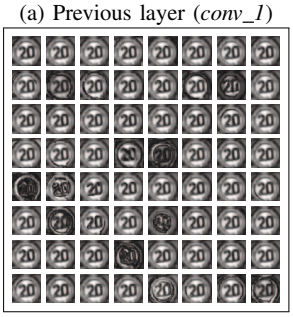
As discussed in the rest of this section, here $\boldsymbol{M}_l^a \in [0, 1]$, is a *soft*-attention mask obtained by identifying and augmenting the salient local regions within the latent features based on

[1]The number of neighboring triplets may be less than 26 if the triplet is at the boundary of the image or scale space.

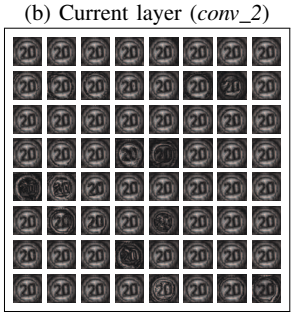


(a) Previous layer (*conv_1*)

(b) Current layer (*conv_2*)



(c) DoC construction



(d) Extrema extraction

(e) Extrema smoothing

(f) Learned *SAN* mask

Fig. 4: Sample outputs of the various components of *SAN*– these samples are taken at the *first* bottleneck position in VGG-16, implementing attention on the outputs from *conv_1* ($\boldsymbol{Y}_{l-1}$) and *conv_2* ($\boldsymbol{Y}_l$), with 64 channels (kernels) represented here using an 8×8 grid. (4a) shows the output from *conv_1*; (4b) shows the output observed at *conv_2*; (4c) shows the DoC extracted from these two layers; (4d) highlights the detected extrema; and (4e) shows the output of the extrema smoothing step; finally, (4f) shows the 64 *detailed and diverse* attention masks learned by the proposed *SAN* module

informative local changes. More specifically, we propose to compare latent features (outputs) from two adjacent layers, $l-1$ and l to help identify the robust salient region, as opposed to relying only an individual layer output, $\boldsymbol{L}_l$, as in conventional attention networks. We detail the process, visualized in Figure 5, next:

1) Difference-of-Convolutions (DoC) Construction: Figures 4a and 4b show sample kernels learned in two consecutive layers. In this paper, we argue (and experimentally show in Section IV) that we can learn diverse attention masks by considering these two adjacent layers together. In order to extract salient regions in layer l, we first construct a difference-

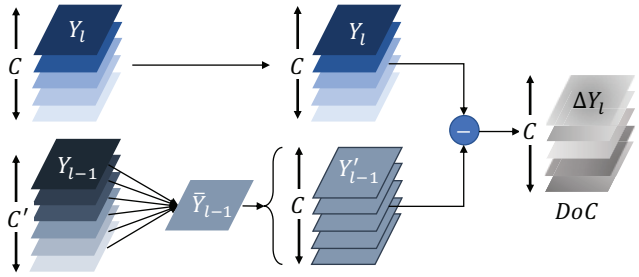


Fig. 5: Overview of the difference-of-convolutions (DoC) construction module in *SAN*: the module takes latent features ($\boldsymbol{Y}$) from two consecutive layers ($l-1$ and l) and transforms the latent features $\boldsymbol{Y}_{l-1}$ into the same basis space as $\boldsymbol{Y}_l$ by taking average along the channel axis ($\overline{\boldsymbol{Y}}_{l-1}$), followed by the expansion of the channel dimension through replication to obtain $\boldsymbol{Y}^{'}_{l-1}$; finally, we take the absolute difference ($\triangle \boldsymbol{Y}_l = |\boldsymbol{Y}_l - \boldsymbol{Y}^{'}_{l-1}|$) to obtain the DoC

of-convolutions (DoC) representation, which helps facilitate localization of scale-space changes that are prominent in an image. Note that unlike SIFT [41], where the DoG is constructed by performing a pixel-by-pixel subtraction of two Gaussian smoothed images, in *SAN*, we seek the difference among the latent features $\boldsymbol{Y}_{l-1}$ and $\boldsymbol{Y}_l$, where the convolution kernels themselves are learned from the data. Therefore, the DoC is computed from the outputs of the two consecutive convolution layers, $l-1$ and l, as follows:

$$\triangle \boldsymbol{Y}_l = |\boldsymbol{Y}_l - \boldsymbol{Y}_{l-1}| \tag{5}$$

As we experimentally show in Section IV, taking the absolute difference (as opposed to simple difference) has significant positive impact on the attention performance – this is because attention needs to be given to, not only maxima, but extrema of the difference of Gaussians. In addition, taking a non-absolute difference might cause multiple counter-intuitive effects in the network: first, the introduction of negative gradients may lead vanishing gradient problem as positive and negative gradients might cancel each other; secondly, the negative difference to *sigmoid* function will push the attention towards "0", as $sigmoid(x) \in [0, 0.5],\ \forall\ x \leq 0$.

Note, however, that there is a significant problem with the Equation 5: the latent features from the two layers do not have one-to-one correspondence, therefore the difference operation is not well defined: $\boldsymbol{Y}_l \in \mathbb{R}^{H \times W \times C}$ and $\boldsymbol{Y}_{l-1} \in \mathbb{R}^{H \times W \times C'}$, where H and W is height and width of the input image and C and C' are the number of channels/kernels in the layers, l and $l-1$, respectively. Therefore, the set of channels (kernels) $\mathbb{C}$ and $\mathbb{C}'$ where $C = |\mathbb{C}|$ and $C' = |\mathbb{C}'|$ potentially represent two different sets of basis vectors. Therefore, to implement DoC over these different sets of basis vectors, we propose to take average along the channel dimension, s.t.

$$\overline{\boldsymbol{Y}}_{l-1}[h, w, 1] = \frac{1}{C} \sum_{c=1}^{C} \boldsymbol{Y}_{l-1}[h, w, c] \quad \forall h = 1 \ldots H, w = 1 \ldots W \tag{6}$$

where $\overline{\boldsymbol{Y}}_{l-1} \in \mathbb{R}^{H \times W \times 1}$ represents the channel average of $\boldsymbol{Y}_{l-1}$. We, then, expand the channel dimension of $\overline{\boldsymbol{Y}}_{l-1}$ as

$$\boldsymbol{Y}_{l-1}^{'} = stack(\overline{\boldsymbol{Y}}_{l-1}, C'), \tag{7}$$

where C' is the number of channels of $\boldsymbol{Y}_l$ and the "$stack$" operation allows for stacking C' many replicas of $\overline{\boldsymbol{Y}}_l$ along the channel dimension to obtain $\boldsymbol{Y}_{l-1}^{'} \in \mathbb{R}^{H \times W \times C'}$. Consequently, the representative $\boldsymbol{Y}_l^{'}$ is now comparable to $\boldsymbol{Y}_l$ and the proposed attention mechanism, *SAN*, can be applied on two adjacent convolutional layers with different number of channels without a padding operation to align the dimensions.

Given the above, we define the salient change across the latent features (updating the Equation 5) as follows:

$$\triangle \boldsymbol{Y}_l = |\boldsymbol{Y}_l - \boldsymbol{Y}_{l-1}^{'}|. \tag{8}$$

Here, $\triangle \boldsymbol{Y}_l$ represents the change in latent features defined in terms of the absolute difference between the two latent features. We present sample results in Figure 4c: as we see in the figure, the DoCs discovered using two consecutive CNN layers retain large degrees of detail.

2) Extrema Detection: The *SAN* attention mechanism leverages the computed values of $\triangle \boldsymbol{Y}_l$ to learn the attention mask $\boldsymbol{M}_l^a$; but, we cannot use DoC directly as an attention mechanism: One reason for this is that the DoC itself can be subject to noise. This problem can be resolved by using the extrema of DoC rather than the DoC itself. However, simply detecting an extremum by exploring the neighborhood and marking it as "1" if it is a local extremum and "0" if not, might lead to a salt-and-pepper noise in attention, severely limiting the network's learning ability. Since our goal is to focus on the changes that are robust and prominent, we instead propose a weighted extrema detection mechanism, as follows:

$$\begin{gathered} \forall h = 1 \dots H, w = 1 \dots W, c = 1 \dots C \\ \boldsymbol{Y}_l^e[h, w, c] = \alpha_{h,w,c} \times \triangle \boldsymbol{Y}_l[h, w, c], \end{gathered} \tag{9}$$

where

$$\begin{gathered} \forall h' \in \{h-1, h, h+1\},\ w' \in \{w-1, w, w+1\} \\ \alpha_{h,w,c} = \frac{\#of \triangle \boldsymbol{Y}_l[h, w, c] \geq \triangle \boldsymbol{Y}_l[h', w', c]}{9}. \end{gathered} \tag{10}$$

Here, $\alpha \in (0, 1]$ is the weighing parameter representing the portion of the DoC neighborhood (3×3 region around the coordinates $\langle h, w \rangle$) for channel c in layer l smaller than the DoC value $\triangle \boldsymbol{Y}_l[h, w, c]$. As we see in the degree of contrast present in the sample results in Figure 4d, this step helps highlight the salient points in the DoC while suppressing non-informative regions – the use of localized weighing *suppresses* noisy perturbations and retains more salient latent changes.

3) Extrema Smoothing: While the soft extrema detection mechanism on $\triangle \boldsymbol{Y}_l$ proposed above allows for *highlighting* salient changes and *suppressing* the noise through localized weighing, this operation can still leak certain amount of noise and artifacts in the weighed output, $\boldsymbol{Y}_l^e[h, w, c]$. Therefore, we further propose to leverage trainable convolutional layers, with kernels the same size as the kernels (k) of the feature extraction branch of the network, to smooth away such artifacts:

$$\boldsymbol{Y}_l^a = \sigma\left(\boldsymbol{Y}_l^e * \boldsymbol{W}_l^k + b_l^k\right) \tag{11}$$

The application of this additional convolutional layer acts as a blurring operation that smooths the unintended extrema artifacts, thus enabling the learning of a more robust attention mask. Sample results are presented in Figure 4e – note that, the smoothing operation, not only eliminates artifacts, but also boosts diversity relative to the pre-smoothed version of the extrema. The validity of this observation and the positive impact of this additional smoothing step are validated in the experimental evaluation section (Section IV, Table IX).

4) Attention Mask Recovery: In the final step of *SAN*, we pass the convolved output, $\boldsymbol{Y}_l^a$, through the *sigmoid* function to learn the final attention mask, $\boldsymbol{M}_l^a$, highlighting the salient attention regions:

$$\boldsymbol{M}_l^a = sigmoid(\boldsymbol{Y}_l^a). \tag{12}$$

Intuitively, the sigmoid function takes a real-valued vector of attentions and maps them to values in the range $[0, 1]$ such that entries in the vector that are away from 0 are saturated to 0 or 1 depending on whether they are negative or positive, respectively, and entries ~ 0 take a non-boundary value between 0 and 1 following a sigmoid shape. Consequently, $\boldsymbol{M}_l^a$ serves as a soft attention mask s.t. $\boldsymbol{M}_l^a[h, w, c] \in [0, 1]$ for layer l.

Figure 4f illustrates the rich (detailed and diverse) attention masks learned by the proposed scale-space attention network, *SAN*, as it intelligently uses the outputs of two adjacent convolutional layers to discover salient local regions to focus the attention.

IV. Experiments

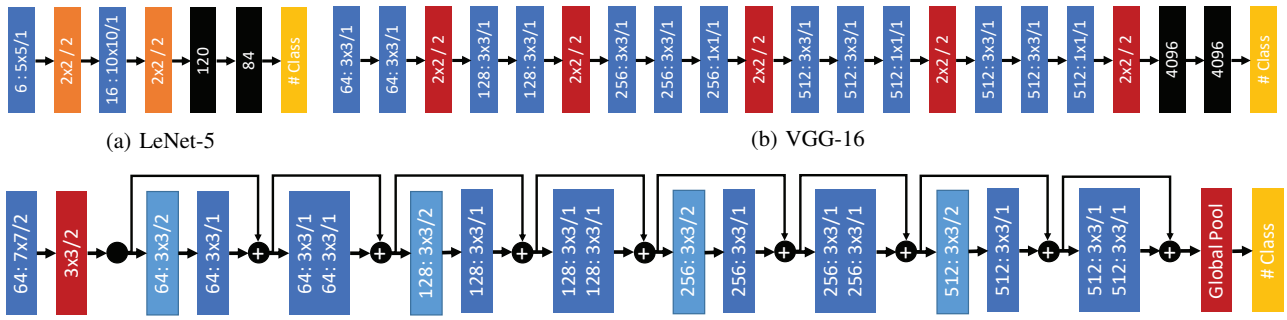
In this section, we experimentally evaluate of the proposed *SAN* framework and compare it against the baseline, non-attentioned networks (LeNet-5 [29], VGG [14] and ResNet [15] - see Section IV-B) as well as the major competitors (CBAM [24], BAM [25], and RAN [23] (see Section IV-D for more details) in bottleneck and full positions (IV-C).

We implemented *SAN* in Python environment (3.5.2) using Keras Deep Learning Library (2.2.4-tf) [42] with TensorFlow Backend (1.14.0) [43]. All experiments were performed on an Intel Xeon E5-2670 2.3 GHz Quad-Core Processor with 32GB RAM equipped with Nvidia Tesla P100 GPU with 16 GiB GDDR5 RAM with CUDA-10.0 and cuDNN v7.6.4[2].

A. Datasets

- For the simpler LeNet network, we consider data sets recorded in controlled environments:
  - **MNIST** contains $60k$ and $10k$ training and testing handwritten digit images of 28×28 resolution [13].

[2]Results presented in this paper were obtained using NSF testbed: "*Chameleon: A Large-Scale Re-configurable Experimental Environment for Cloud Research*"



(a) LeNet-5 (b) VGG-16

(c) RESNet-18

Fig. 6: Overview of the network architectures for LeNet-5 [29], VGG-16 [14], and RESNet-18 [15]. Colors represent, Blue: Convolution (stride=1), Light-Blue: Convolution (stride=2), Orange: avg-pooling (avg), Red: max-pooling (max), Black: fully connected (fc), and Yellow: output layer (fc-softmax)

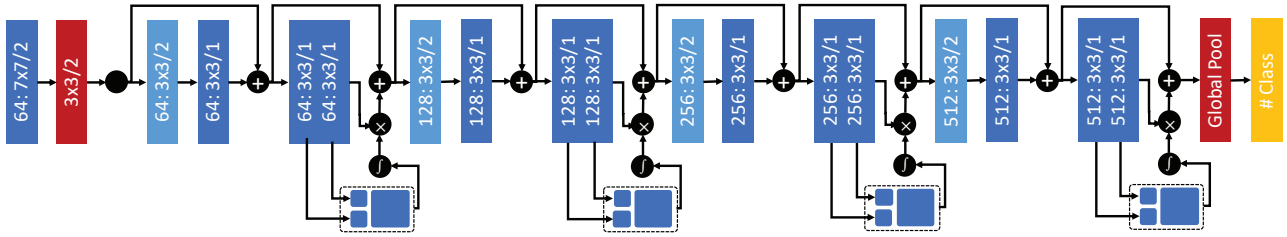


Fig. 7: RESNet-18 with bottleneck attentions (attention applied before pooling layers)

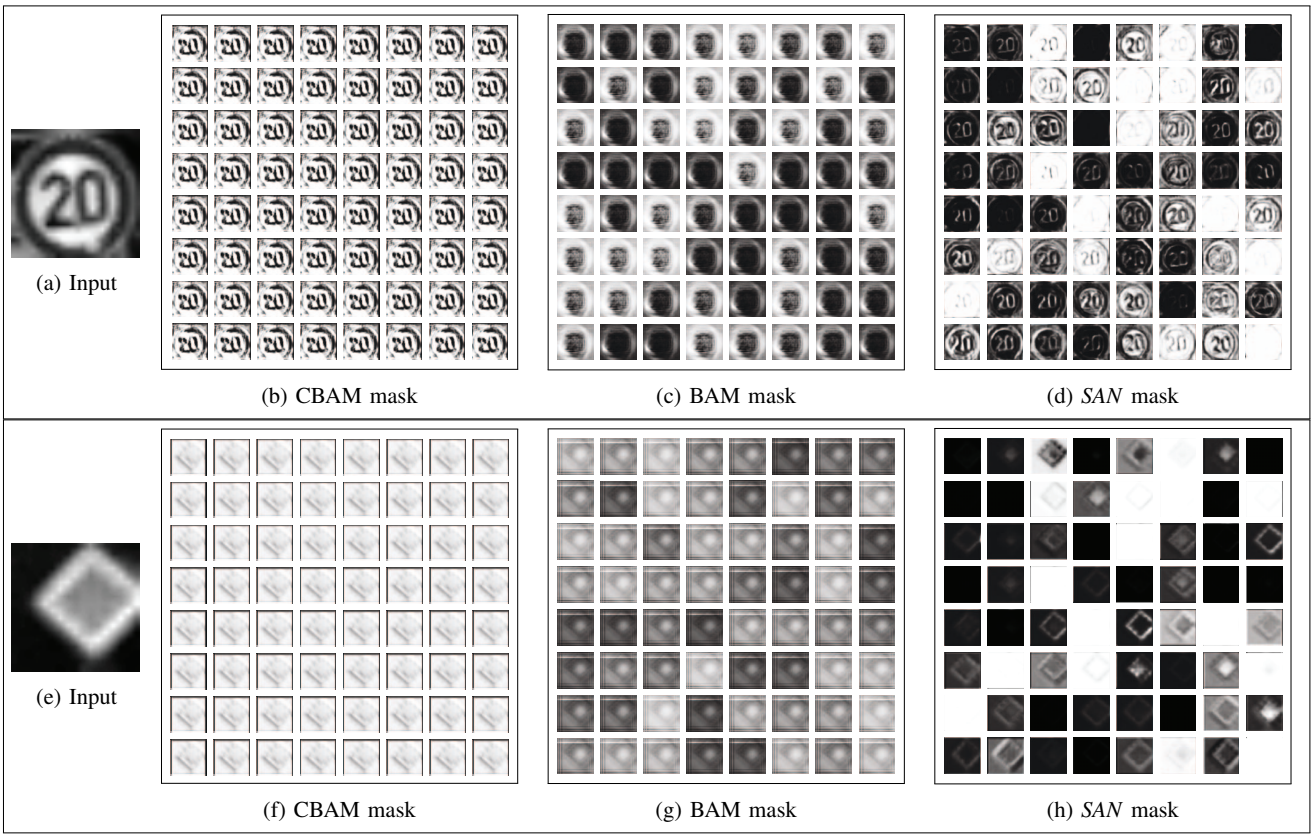
  - **FMNIST** contains $60k$ and $10k$ training and testing images of 28×28 resolution with 10 classes [44].
- For the more complex VGG/ResNet Network, we consider data sets recorded in real-world settings:
  - **CIFAR10/20/100** contains $50k$ training and $10k$ testing images, respectively, with 32×32 resolution and the dataset contains 10, 20, and 100 labels [45].
  - **GTSRB** dataset contains 39,209 and 12,630 training and testing images for 43 unique traffic sign [46].
  - **GTSDB** is an object detection dataset with bounding boxes representing the positions of signs [47].
  - **ImageNet** contains $\sim$1.23 million images for 1K real-world entities, with $\sim$1K images per entity [30].
- For recurrent networks, we consider multi-variate time series data sets:
  - **Mocap**: contains sensor (62) recording for 184 subjects for 8 gestures [48].
  - **FFC**: constains flight statistics for fuel usage, including temperature and wing position for 500 flights.

B. Baseline (Non-Attentioned) Architectures

1) LeNet-5: Designed for recognizing handwritten digits [29], LeNet-5 is a relatively simple network with 5 trainable (2 convolution and 3 dense) and 2 non-trainable layers using average pooling (Figure 6a). LeNet demonstrated that localized image features (handcrafts) can be substituted by deep features through the use of back-propagation of the classification error. The two convolution layers contain 6 and 16 kernels and dense layers have 120 and 84 kernels. Hidden layers are *tanh* and the final layer is *softmax*. LeNet's simplicity has made it the benchmark architecture for datasets recorded in constrained environments, such as MNIST, and FMNIST, in many works [33], [49], [50].

2) VGG: With the increase in complexity of data [30], a deeper and more complex architecture was required. An answer to this requirement was the VGG network [14] (Figure 6b), a 16 and 19 layer networks with 13 and 16 convolution layers respectively and 3 dense layers, with interleaved 5 max-pooling layers. VGG demonstrated that small kernel sizes (e.g. 3×3) can achieve better accuracies than using large kernels (e.g. 5×5 or 11×11). Furthermore, VGG leverages ReLU as the hidden activation to overcome the problem of vanishing gradient, as opposed to *tanh*. Furthermore, the network uses a convolutional layer with kernel size 1×1 as 7^{th}, 10^{th} and 13^{th} layers in the network to introduce additional non-linearity and uses rectification operation. In addition to the kernel size, VGG proposed to slide the convolution kernel by 1 unit in each spatial direction and pooling kernel by 2 units along each spatial dimension. Given the ability of VGG network to learn the complex pattern in the real-world dataset, we use the network on datasets, such as CIFAR10/20/100, GTSRB, and ImageNet that contains complex, real-world objects.

3) ResNet: As seen in Section I, much of the success of neural networks lies in their *depth* and *width*, however, as [18] shows, the network saturates, and may even degrade, after a certain depth is reached. ResNet [15] demonstrated that the problem of accuracy saturation/degradation might be alleviated



(a) Input (b) CBAM mask (c) BAM mask (d) *SAN* mask

(e) Input (f) CBAM mask (g) BAM mask (h) *SAN* mask

Fig. 8: Attention masks learned by CBAM, BAM and *SAN* module for GTSRB dataset when placed at the *first* bottleneck position in VGG-16: *SAN* masks are more diverse and retain finer details from the input images

by the use of *residual connections* (Figure 6c). We consider ResNet architecture for two depths: 18 and 50. For instance, ResNet-18 consists of 17 convolutional layers with varying (64, 128, 256, and 512) number of convolutional kernels, a 2D maxpooling layer, a global average layer, and a fully connected layer. First convolutional layer uses a kernel size of 7×7 and the remainder use 3×3 as kernel size. Each convolutional layer is followed by batch-normalization [33] and ReLU [34].

4) LSTM-4: We used a 4 layers LSTM architecture comprised of 64 LSTM units each, with average pooling following every two recurrent layer, and final dense layer with *softmax* (classification) and *linear* (forecasting) as output activation. We compare SAN to recurrent attention network (RAN) [23].

C. Positioning the Attention Modules

As noted in Equation 1, and seen in Figure 6, a neural network is a sequence of layers interleaved by sub-sampling (pooling) layers. This means that there are multiple locations in the network where the latent features are being generated and transfered forward. As we have discussed in Section III-A, in this paper, we consider two alternative attention strategies:

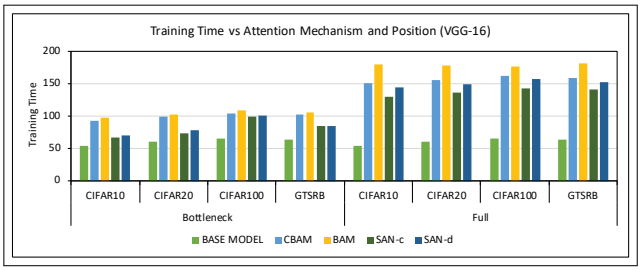
- *Bottleneck placement strategy*: attention modules are applied before the data is down-size at pooling layers.
- *Full placement strategy*: in this case, attention modules are placed for every trainable layer in the network.

Figure 7 shows the version of the ReSNet-18 networks extended with attention modules, under bottleneck strategy.

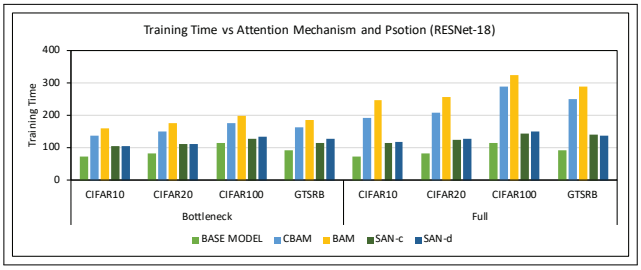
D. Competitors

In this section, we consider the following competitors:

- *Non-Attentioned Baselines*: As the basic baseline, we consider the networks without any attention module. In particular, we explore *four* types of baseline architectures: LeNet-5, VGG, RESNet, LSTM-4 (see Section IV-B for more details).
- *Convolutional Block Attention Network (CBAM)*: CBAM is an attention module [24] that is designed to leverage contextual relationships among channel and spatial latent features to learn the attention mask. This is achieved by sequentially considering channel attention followed by spatial attention. Intuitively, the channel attention helps learn "*meaningfulness*" of the image, followed by spatial attention to learn "*where*" this meaningful information lies in the image. [24] suggested that the CBAM modules are placed after convolutional layers.
- *Bottleneck Attention Module (BAM)*: In contrast to the CBAM's sequential approach towards learning channel



(a) VGG-16 model architecture



(b) RESNet-18 model architecture

Fig. 9: Model training time (*in seconds*)

and spatial attention, BAM [25] computes channel and spatial attention simultaneously, similar to inception networks [16]. BAM creates *three* branches in the network, 1) feature branch, 2) channel branch, and 3) spatial branch. In the *feature* branch, the latent features are propagated forward, similar to the conventional networks, whereas channel and spatial branch learn the respective attention masks. Note that unlike CBAM (which relied on conventional convolution layers), BAM used dilated convolution layers. BAM recommended that the attention modules be placed before the bottleneck.

– *Recurrent Attention Network (RAN)* [23]aims at learning the subset of input feature at t while relying on the model output at time *t-1*.

As described above, CBAM and BAM use different (full vs. bottleneck) attention module placement strategies. In this paper, we consider both strategies when comparing *SAN* against the competitors. Figure 8 displays bottleneck attention for three sample images under CBAM, BAM, and the proposed *SAN* attention mechanisms.

Note also that CBAM and BAM rely on different (conventional and dilated) types of convolution layers. We, therefore, trained two different versions of the proposed attention modules: *SAN*-c with conventional convolution layers and *SAN*-d with dilated convolutional kernels.

E. Experimental Results

1) Classification Accuracy: To evaluate the effectiveness of *SAN* framework, and demonstrate its robustness to the network architecture, in this section we measure classification accuracies on *three* network architectures (LeNet-5, VGG-16, and RESNet-18) and on *seven* benchmark datasets (MNIST, FMNIST, CIFAR10/20/100, GTSRB and ImageNet)[3]. Top-1 ans Top-5 Classification accuracy results are presented in Tables I, IV, II, and III. We define top-k accuracy as the ratio of the experiments in which the true class label is observed among top-k candidate class labels.

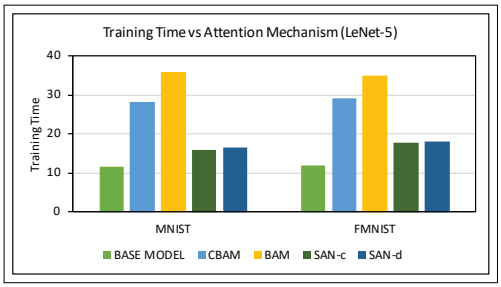


Fig. 10: Model training time (*seconds*) for LeNet-5

Figure I shows top-1 and top-5 classification results for the complex ImageNet dataset for VGG-16 and RESNet-18 network architectures. As we see in this figure, *SAN*-c and *SAN*-d consistently outperform the baselines and the attention competitors, CBAM and BAM. We see in the figure that the results are relatively comparable for bottleneck and full strategies and also that the version of *SAN* which uses dilated convolutional kernels provides the overall highest accuracy gains under all scenarios. On the average, the accuracy gains provided by *SAN*-d is $4.91\times$ the accuracy gains provided by CBAM and $1.68\times$ the accuracy gains provided by BAM over the baseline. BAM's inception-style approach of having independent parallel branch allows for better summarization of contextual information into attention mask than CBAM, however, the approach of taking the global average and maximum to summarize entire spatial information into single value limits the performance gains. *SAN* leverages the salient changes in latent features to identify points of attention to outperform both of these competitors.

In Table IV, we evaluate the classification performance of different architectures on MNIST and FMNIST datasets with LeNet-5 architecture. Note that the LeNet-5 architecture is a special case, where the same architecture with *SAN* attention module represents both bottleneck and full attention model, as LeNet has only 2 convolution layers and each layer is followed by a down-sampling layer thus making it both bottleneck and full attention architecture simultaneously. Therefore, Table IV does not present full and bottleneck results separately. As we see in the figure, thanks to the simplicity of the data, the baseline architecture has 98.37% (for MNIST) and 89.43% (for FMNIST) classification accuracy without any attention. Even in this scenario where there is very limited room for improvement, *SAN*-d improved the accuracy to 98.7% (for MNIST) and 89.94% (for FMNIST). In contrast, CBAM resulted in a drop in accuracy to 97.88% (for MNIST) and

[3]We train two types of *SAN* models, first with conventional convolutions (CBAM) and *SAN* d with dilated convolutional kernels (BAM).

TABLE I: Model classification accuracies (top-1 and top-5) for ImageNet data for VGG-16/RESNet-18 model architecture

	VGG-16				RESNet-18			
	Bottleneck		Full		Bottleneck		Full	
Datasets	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Base Model	71.90	90.6	71.90	90.60	70.40	89.45	70.40	89.45
CBAM	72.40	90.97	72.43	91.25	70.95	89.63	70.73	89.91
BAM	72.89	92.46	73.06	92.96	71.12	89.99	71.35	90.45
***SAN*-c**	73.01	93.24	73.87	93.58	71.64	91.45	71.88	91.53
***SAN*-d**	**73.57**	**93.97**	**74.26**	**94.07**	**72.01**	**92.87**	**72.38**	**92.93**

TABLE II: Model classification accuracies (top-1 and top-5) for VGG-16 model architecture

	Bottleneck								Full							
Datasets	**CIFAR10**		**CIFAR20**		**CIFAR100**		**GTSRB**		**CIFAR10**		**CIFAR20**		**CIFAR100**		**GTSRB**	
Accuracy	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Base Model	72.70	93.12	45.17	70.01	31.07	53.01	96.21	99.84	72.70	93.12	45.17	70.01	31.07	53.01	96.21	99.84
CBAM	76.57	95.23	46.14	72.56	32.17	54.02	96.38	99.96	74.65	94.26	45.74	72.16	32.51	54.67	97.45	100.00
BAM	76.15	94.85	48.95	76.89	32.96	55.65	96.85	100.00	76.42	95.63	46.79	73.68	35.96	61.45	97.73	100.00
***SAN*-c**	78.42	97.86	50.23	78.99	34.58	58.99	97.96	100.00	79.89	97.99	51.14	82.99	37.59	63.48	98.31	100.00
***SAN*-d**	**79.01**	**99.50**	**52.84**	**82.03**	**36.14**	**61.23**	**98.25**	**100.00**	**81.24**	**99.98**	**54.88**	**86.48**	**39.03**	**68.45**	**98.95**	**100.00**

TABLE III: Model classification accuracies (top-1 and top-5) for RESNet-18 model architecture

	Bottleneck								Full							
Datasets	**CIFAR10**		**CIFAR20**		**CIFAR100**		**GTSRB**		**CIFAR10**		**CIFAR20**		**CIFAR100**		**GTSRB**	
Accuracy	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Base Model	68.55	90.51	49.32	73.00	37.83	62.95	97.85	99.98	68.55	90.51	49.32	73.00	37.83	62.95	97.85	99.98
CBAM	73.76	96.40	53.39	80.34	40.95	65.65	98.11	99.99	74.40	97.42	51.73	75.89	40.74	67.97	98.52	100.00
BAM	73.42	93.42	53.53	81.13	40.72	65.39	98.42	100.00	72.67	96.71	54.77	82.64	40.49	67.01	98.95	100.00
***SAN*-c**	74.82	97.86	55.23	85.82	41.74	67.10	99.24	100.00	75.03	98.99	52.74	78.95	44.14	72.43	99.53	100.00
***SAN*-d**	**78.89**	**99.97**	**56.53**	**87.26**	**41.49**	**66.95**	**99.76**	**100.00**	**79.67**	**100.00**	**54.94**	**83.98**	**47.06**	**74.42**	**99.85**	**100.00**

TABLE IV: Model classification accuracies (top-1 and top-5) for LeNet-5 model architecture

Datasets	**MNIST**		**FMNIST**	
Accuracy	**Top-1**	**Top-5**	**Top-1**	**Top-5**
Base Model	98.37	99.98	89.43	99.87
CBAM	97.88	99.98	89.27	99.85
BAM	98.52	99.99	89.64	99.90
***SAN*-c**	98.56	99.99	89.75	99.92
***SAN*-d**	**98.70**	**100.00**	**89.94**	**99.96**

89.27% (for FMNIST). Attention using BAM, on the other hand, provides some gains (98.52% for MNIST) and 89.64% for FMNIST), but lower than the gains provided by *SAN*-d.

In Tables II and III, we evaluate the application of different attention modules on VGG-16 and RESNet-18 architectures and evaluate their performance on relatively complex CIFAR10/20/100 and GTSRB datasets, under bottleneck and full attention placement strategies. These two figures reconfirm that, overall, *SAN*-d is the best attention strategy, providing significant up to 9.71% accuracy gains over the baseline.

Figure 8 provides sample attention masks to explain the key reasons behind the accuracy gains of *SAN*. As we see in this figure, *SAN* is able to learn rich (diverse and detailed) and robust attention masks. In stark contrast, CBAM learns only a single attention mask shared across all channels in the convolutional layer, severely limiting its ability to learn rich attention. While BAM does learn an explicit attention mask for each individual channel, it is able to retain coarser details - adding only limited richness to the network. In short, *SAN*'s ability to account for the differences between two consecutive convolutional layers enables rich and robust attention masks leading to significant boost in network performance.

Table V demonstrates that SAN is able to outperform CBAM and BAM for deeper networks as well. SAN's ability to learn salient changes across layers proves beneficial, also when the level of abstraction increases in deeper networks.

2) Model Training Time: In Figures 9 and 10, we compare the computational cost (training time) of *SAN* framework against the competitors. The figures show that the proposed *SAN* mechanism introduces much smaller training overhead than the competitors, CBAM and BAM. While *SAN*-d, with dilated convolutional kernels, requires more training time than *SAN*-c, the difference is slight, and higher accuracy gains of *SAN*-d makes that difference worthwhile. One important observation comparing Figures 9a and 9b, is that on the same data, *SAN* provides significantly higher training execution gains over CBAM and BAM on (residual connection-based) RESNet-18 than on VGG-16. In fact, while the training cost for *SAN* strategies are similar for both networks, CBAM and BAM's training costs doubles when residual connections are introduced. This indicates that the scale-invariant robust attention generated through DoC extrema lead to a much more effective use of the residual connections. Overall, *SAN* provides models with higher classification accuracies compared to CBAM and BAM, at a significantly lower training cost.

3) Time Series Classification and Forecasting: In Table VI and VII, we see that *SAN* outperforms the base model as well as RAN for both classification and forecasting of multi-variate time series. For the classification task (Table VI), we observe that, while all three models are able to reach 100% model accuracy, SAN leads to 43% drop in model loss compares to

TABLE V: Classification accuracy for deeper models - CIFAR10 (VGG-19 and ResNet-50) - Bottleneck

Network	**VGG-19**		**ResNet-50**	
Accuracy	Top-1	Top-5	Top-1	Top-5
Base	69.48	86.95	67.79	90.86
CBAM	70.24	89.85	65.82	89.74
BAM	71.41	94.01	68.76	94.51
***SAN*-c**	77.14	96.21	71.36	96.89
***SAN*-d**	**77.36**	**96.88**	**72.73**	**97.25**

TABLE VI: Model classification accuracy and loss for Mocap dataset for LSTM model[4]

Metric	**Accuracy**		**Loss** (MAE)	
Position	**Bottleneck**	**Full**	**Bottleneck**	**Full**
Base Model	100.00	100.00	0.1118	0.1118
RAN	100.00	100.00	0.1055	0.0751
SAN	100.00	100.00	**0.0641**	**0.0084**

TABLE VII: Model forecasting accuracy for flight fuel consumption dataset for LSTM model[4]

Metric	**Accuracy** (cos. sim.)		**Loss** (MAE)	
Position	**Bottleneck**	**Full**	**Bottleneck**	**Full**
Base Model	0.9672	0.9672	40.88	40.88
RAN	0.9526	0.9745	37.04	33.40
SAN	**0.9745**	**0.9773**	**33.40**	**31.48**

TABLE VIII: Model object detection accuracy for GTSDB dataset for VGG-16 architecture

Position	**Bottleneck**	**Full**
Base Model	86.79	
CBAM	88.25	90.67
BAM	89.87	91.71
***SAN*-c**	92.45	93.01
***SAN*-d**	**93.88**	**95.63**

Base model and 39% for RAN attention module for bottleneck positions, and for full attention position, SAN leads to 92% drop in loss against Base model and 89% against RAN, this demonstrated that the importance of input features learned (attention mask) by SAN is more informed and robust than the mask learned by RAN. For the forecasting task, SAN leads to maximum accuracy and minimum forecasting error[4].

4) Object Detection: In Table VIII, we observe that for both bottleneck and full positions, SAN can better learn to detect objects of interest in an image. This highlights the importance of leveraging the salient changes across layers(while using only a single trainable layer in the attention module) as opposed to CBAM and BAM which rely on more than *one* trainable layers in each module.

5) Ablation Studies: We finally present, in Table IX, ablation studies that validate the three key hypotheses underlying the *SAN* attention framework:

- *Cross-layer channel alignment*: As discussed in Section III-C1, the latent features from layers $l-1^{th}$ and l^{th} layer are not represented on the same basis; therefore, we propose an efficient transformation that maps these two layers onto a common basis, without padding. In Table IX, *w CCA* refers to case where channel alignment is used as described, whereas *w/o CCA* refers to the case where channels are not aligned.
- *Absolute difference for extrema DoC construction*: As discussed in Section III-C1, we seek attention at the extrema of the DoC – not only maxima – in order to prevent the "*sigmoid*" operation on the latent features to wipe-out heavily negative values, we define DoC using absolute difference. In Table IX, *w Abs* refers to case where absolute differences are used to construct DoC, whereas *w/o Abs* refers to the case where simple (non-absolute) difference is used.
- *Attention to the extrema of DoC*: As mentioned above, to seek points of attention, we look at the extrema of DoC. In Table IX, *w Extrema* refers to case where an extrema search step is applied on the DoC, whereas *w/o Extrema* refers to the case where the DoC is used directly without seeking its extrema.
- *Extrema smoothing*: As discussed in section III-C3, while extrema help identify the salient points in the latent features, smoothing of these extrema can help eliminate noise and improve robustness. In Table IX, *w Smoothing* refers to case where a final smoothing step is applied, whereas *w/o Smoothing* refers to the case where the smoothing step is omitted.

As we see in the table, the highest accuracies are obtained when all four steps are combined. It is especially interesting to see that, alone, extrema detection does not improve accuracy – results with extrema detection, but without smoothing ($\#4$) are not better than results without extrema detection ($\#3$); however, when combined with the final smoothing step ($\#5$) to eliminate artifacts, extrema detection is very effective in boosting the overall accuracy.

V. Conclusion

In this work, we presented a robust and model-independent attention module that aims to guide the attention of the network architecture to salient localized regions in the image to boost the network accuracy, with minimal training overhead. To achieve this goal, we propose an innovative robust feature learning framework with novel scale-invariant attention networks (*SAN*) that identify salient regions in the input data using extrema of the difference of Gaussians. Unlike the existing attention networks, *SAN* primarily concentrates attention on parts of the data where there is major change across space and scale. We experimentally evaluated and showed that the proposed attention module, *SAN*, can be successfully applied to various state-of-the-art architectures, such as LeNet-5, VGG-16, and RESNet-18, as an add-on to significantly boost the effectiveness, including for benchmark datasets . Experiments further showed that, *SAN* leads to minimal training overhead in comparison to the attention modules, such as CBAM, BAM, and RAN.

[4]In Section IV-E3, we present both model accuracy and loss. As in classification results, all models reach 100% accuracy, therefore, we leverage model loss as a measure to compare the models and for forecasting, loss is used to measure the divergence of the forecasting results from ground truth.

TABLE IX: Model classification accuracy vs model architecture and dataset summarizing the performance of different blocks involved in devising *SAN* module ("–": incompatible configuration when two layers have different channel counts)

	Architectures	**LeNet-5**		**VGG-16**				**RESNet-18**			
	Datasets	**MNIST**	**FMNIST**	**CIFAR10**	**CIFAR20**	**CIFAR100**	**GTSRB**	**CIFAR10**	**CIFAR20**	**CIFAR100**	**GTSRB**
0	**Base Model**	98.37	89.43	72.7	45.17	31.07	96.21	68.55	49.32	37.83	96.21
1	**no-Abs**, **no-CCA**, **no-Extrema**, **no-Smoothing**	–	–	68.83	39.48	22.07	93.25	65.82	46.36	41.53	93.51
2	**Abs**,**no-CCA**, **no-Extrema**, **no-Smoothing**	–	–	70.45	42.17	30.99	93.96	68.83	48.42	42.3	94.01
3	**Abs**,**CCA**, **no-Extrema**, **no-Smoothing**	98.34	89.55	73.95	40.19	32.34	95.98	71.53	49.17	42.86	96.23
4	**Abs**, **CCA**, **Extrema**, **no-Smoothing**	98.17	89.28	72.79	42.3	29.21	94.96	70.59	48.62	42.17	96.34
5	**Abs**, **CCA**, **Extrema**, **Smoothing**	**98.56**	**89.75**	**78.42**	**50.23**	**34.58**	**97.96**	**74.82**	**55.23**	**44.14**	**98.31**

REFERENCES

[1] T. Sellam, K. Lin, I. Huang, M. Yang, C. Vondrick, and E. Wu, "Deepbase: Deep inspection of neural networks," in *SIGMOD*, 2019.

[2] X. Lu, "Learning to generate questions with adaptive copying neural networks," in *SIGMOD*, 2019.

[3] J. Dai, M. Zhang, G. Chen, J. Fan, K. Y. Ngiam, and B. C. Ooi, "Fine-grained concept linking using neural networks in healthcare," in *SIGMOD*, 2018.

[4] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li, "Air: Attentional intention-aware recommender systems," in *ICDE*. IEEE, 2019.

[5] C.-H. Yeh, Y.-C. Fan, and W.-C. Peng, "Interpretable multi-task learning for product quality prediction with attention mechanism," in *ICDE*. IEEE, 2019.

[6] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *ICWAIM*. Springer, 2014.

[7] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in *IJCAI*, 2015, pp. 3995–4001.

[8] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *TNN*, 1997.

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE SPM*, 2012.

[10] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *CVPR*, 2015.

[11] X. Z. M. M. R. F. Pierre Sermanet, David Eigen and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014.

[12] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *ICPR*. IEEE, 2012.

[13] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *IEEE*, 1998.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[17] F. Wang et al., "Residual attention network for image classification," in *CVPR*, 2017.

[18] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *CVPR*, 2015.

[19] S. Liu and et. al., "Notes2: Networks-of-traces for epidemic spread simulations," in *Workshops AAAI*, 2015.

[20] Y. Garg and S. R. Poccia, "On the effectiveness of distance measures for similarity search in multi-variate sensory data," in *ICMR*, 2017.

[21] Y. Garg and K. S. Candan, "Racknet: Robust allocation of convolutional kernels in neural networks for image classification," in *ICMR*, 2019.

[22] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR*, 2015.

[23] V. Mnih, N. Heess, A. Graves *et al.*, "Recurrent models of visual attention," in *NIPS*, 2014.

[24] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *ECCV*, 2018.

[25] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *BMVC*, 2018.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[28] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.

[29] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *SCGCV*, 1999.

[30] J. Deng et. al., "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[31] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.

[32] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[35] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *ICML*, 2007.

[36] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, 2012.

[37] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *NIPS*, 1990, pp. 396–404.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[39] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *CVIU*, 2008.

[40] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *ITPAMI*, 2010.

[41] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[42] F. Chollet *et al.*, "Keras," 2015.

[43] M. Abadi, P. Barham, and et. al., "Tensorflow: A system for large-scale machine learning," in *USENIX OSDI*, 2016.

[44] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[45] A. Krizhevsky, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[46] "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012.

[47] S. Houben et. al., "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *IJCNN*, 2013.

[48] R. Gross and J. Shi, "The cmu motion of body (mobo) database," 2001.

[49] R. Collobert, J. Weston, and et. al., "Natural language processing (almost) from scratch," *JMLR*, 2011.

[50] X. Glorot and et. al., "Deep sparse rectifier neural networks," in *AISTATS*, 2011.