Targeted Kernel Networks: Faster Convolutions with Attentive Regularization

Kashyap Chitta(✉)

The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA
kchitta@andrew.cmu.edu

Abstract. We propose Attentive Regularization (AR), a method to constrain the activation maps of kernels in Convolutional Neural Networks (CNNs) to specific regions of interest (ROIs). Each kernel learns a location of specialization along with its weights through standard backpropagation. A differentiable attention mechanism requiring no additional supervision is used to optimize the ROIs. Traditional CNNs of different types and structures can be modified with this idea into equivalent Targeted Kernel Networks (TKNs), while keeping the network size nearly identical. By restricting kernel ROIs, we reduce the number of sliding convolutional operations performed throughout the network in its forward pass, speeding up both training and inference. We evaluate our proposed architecture on both synthetic and natural tasks across multiple domains. TKNs obtain significant improvements over baselines, requiring less computation (around an order of magnitude) while achieving superior performance.

Keywords: Soft attention · Region of interest · Network acceleration

1 Introduction

Convolutional Neural Networks (CNNs) have been largely responsible for the significant progress achieved on visual recognition tasks in recent years [23,27,33]. By sharing weights to be used by convolutional kernels across the entire spatial area of their input activations, CNNs use translated replicas of learned feature detectors, allowing them to translate knowledge about good weight values acquired at one position in an image to all other positions. This leads to translational equivariance– a translated input to a convolutional layer will end up producing an identically translated activation after passing through it.

Though it works well in nearly all situations, it is possible for this 'knowledge translation' to be a double-edged sword. By sharing weights across the whole input, we bias the network to prioritize learning representations that would be useful over the entire image area. Due to this, it may have to compromise on learning some weights that are critical to the network's final objective, simply

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11132, pp. 379–397, 2019.
https://doi.org/10.1007/978-3-030-11018-5_34

because these weights were useful only in a small area of the whole image. The possibility of this happening increases if the inputs possess a uniform spatial layout.

Assuming that the network inputs are captured or preprocessed in a way that provides some spatial structure, certain objects are more likely to be in particular locations than others. For example, if the inputs are all upright faces cropped with a face detector, it is far more likely to find an eye in one of the top quadrants of the input than in the bottom ones. In images of outdoor scenes, it is more likely to see blue skies at the top than the bottom. More often than not, there is some such spatial structure associated with the inputs to any visual recognition task. This means that based on what a kernel is supposed to look for, independently learning weights at different spatial locations can potentially generate better representations.

A locally connected layer takes this idea to the extreme– its forward pass involves convolutions with no weight sharing at all, with a different kernel for every spatial location in the input. By perfectly aligning facial images and then learning representations using locally connected layers, human-level accuracy was first achieved in face recognition [64]. Unfortunately, the feasibility of this approach is limited due to the heavy dependence on perfect alignment of inputs and drastic increase in parameter count, leading to a requirement of far more training data (since there is no longer any 'knowledge translation').

Only sharing weights over selected Regions of Interest (ROIs) is another possibility that has been explored, implemented by training separate CNNs on different ROIs and merging their representations at a point deeper in the network [38,39,60,73]. The kernels are now specialized to their input ROIs, and the parameter count increase is controlled by architectural choices. Finding the right ROIs to use, however, is a tedious step usually requiring domain-specific knowledge to be done effectively. Any manual selection, even by the best domain experts, would almost certainly not lead to the most optimal choice of ROIs for the given task and network topology.

An alternative approach would be to learn the most optimal ROI for each kernel directly from the data, by end-to-end training. Trying to do this as a tuple of the ROI center and spatial size results in models that are not differentiable and require complex learning procedures [3]. We propose Attentive Regularization (AR), a method to achieve this using a differentiable attention mechanism [21], allowing our models to be trained end-to-end with simple backpropagation. The key idea behind AR is to associate each rectangular ROI with the parameters of a smooth differentiable attention function. The attention function helps generate gradients of the loss with respect to the location and size of the ROI. Figure 1 illustrates the effect of AR, comparing it with a standard convolution and a fixed ROI based approach. For the purpose of illustration, we use a 'layer' operating on an RGB image with only four kernels, each looking for a semantically meaningful part.

An attractive consequence of having ROIs for each kernel is computational efficiency– computing convolutions over small ROIs for every kernel in a layer

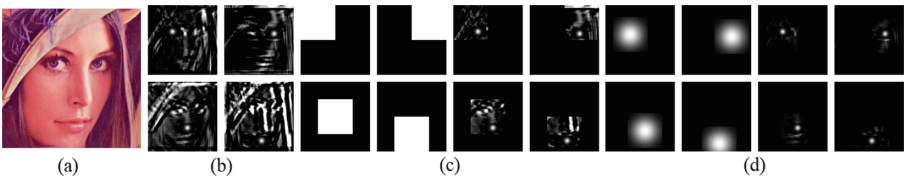

Fig. 1. (a) Input (b) Activations after a standard convolution with four kernels (actually correlation filters). These kernels are optimized to be activated by the left eye, right eye, nose and mouth respectively. However, they give large, unpredictable responses across the image. (c) Manual ROI selection and activations after convolutions on these selected ROIs. (d) The proposed approach, attention functions learned from data, and activations after AR. We observe that through spatial specialization, even crude features can become powerful, as they become independent of other spatial locations.

greatly reduces redundant operations in the network, speeding up both training and evaluation.

Our contribution is three-fold: First, we propose and describe AR and its incorporation into existing CNN architectures, resulting in Targeted Kernel Networks (TKNs). Second, we evaluate TKNs on digit recognition benchmarks with coarse alignment in the form of digit centering, as well as synthetic settings with more alignment, significantly outperforming CNN baselines. Finally, we demonstrate their application for network acceleration on more complicated structured data, like faces and road traffic signs.

2 Related Work

Regularizing CNNs. Deep CNNs have a vast potential to overfit data when they have to be trained from scratch. Conventional machine learning approaches to handle this like weight decay, data augmentation, and model ensembles alleviate the problem only to an extent. Dropout [57] was one of the most successful methods for regularizing layers with very large parameter counts in CNNs [33,55].

Most recent models have substituted this with some constraint on the activations [31], the most popular of which is batch normalization [28]. This uses other images in the mini-batch along with learned scaling parameters to constrain the activations using computed statistics. We force the network to find good weights without giving the kernels free access to all spatial locations in the image during training, with a similar approach of applying constraints through learned parameters.

Spatially Specialized CNNs. Several approaches look into architectures that operate on ROIs, specifically in object detection [19,20,48]. However, these methods typically propose ROI based object candidates for each input image, and not for the network kernels. Additional bounding box supervision is also necessary

to learn these proposals. Unlike these methods, ROIs at a kernel level have been used in facial action unit detection [14], but the regions are hand-crafted [38,39,73].

Attention. One of the most promising trends in research is the emergence of attention based models. Early work in this area [10,34,51] was inspired by the process of sequential recognition used by the biological vision system in humans. Recent adaptations have leveraged the representational power of deep neural networks with visual attention for a variety of tasks, some of which were image classification [4,17,59,66], image generation [21], image captioning [5,25,42,68], visual question answering [44,53,67,69], action recognition [18] and one-shot learning [54]. More closely related approaches to AR are attempts at multi-layer [52] and multi-channel [5] attention. Our main advantage over existing soft attention methods is that we systematically remove computational processing throughout the network while maintaining the fully differentiable property. Other approaches require hard attention with reinforcement learning for network acceleration.

Efficient CNNs. Cheng et al. [8] summarize model compression and acceleration approaches into four categories– parameter pruning and sharing [7,22,35,46,56], low-rank factorization [11,30,63], transferred or compact convolutional filters [12,62,65], and knowledge distillation [6,26,49,70]. One of the primary goals of early attention models was increasing efficiency [3]. This has resurfaced recently in the form of various architectures for spatially restricting computation.

Spatial Computation Restriction in CNNs. Dynamic Capacity Networks [2] define attention maps to apply sub-networks to only specific input patches for fine representations, which they later combine with the representations of a coarse network. Similarly, SBNet [47] uses a low resolution sub-network to obtain a computation mask for the main deep network. A more recent idea uses a learnable application of channel-wise sparsity to completely eliminate certain kernels dynamically [13]. All these techniques restrict computation to the uncertain regions of the current image, whereas in our work, we restrict computation to certain (learnable) regions for all images. The two ideas are orthogonal and computational gains could be observed by combining them.

PerforatedCNNs [16] study strategies for skipping calculation of convolutions tied to certain spatial locations in a convolutional layer. These strategies are loosely based on using grid-like lattices, where computations at the intermediate points are approximated with interpolation. Our work removes computation in a similar fashion, but no interpolation is required since we do not have any intermediate values to recover.

3 Attentive Regularization

In its simplest form, AR can be considered an additional layer operating on the activation of a convolutional layer using an attention window. We begin by explaining the one-dimensional implementation in this form before moving on to the generalized version and higher dimensional inputs.

3.1 AR in One Dimension

Consider the activation tensor $\mathbf{A} \in \mathbb{R}^{D \times L}$ resulting from a one dimensional convolution of a sequence of length L with D different kernels. Let $\mathbf{a}^C \in \mathbb{R}^L$ denote the row of this tensor corresponding to the C^{th} kernel in the layer. The objective of AR is to constrain each one of these activation vectors using a differentiable attention function f_{att}. The window for attention is constructed as this function drops off numerically from 1 to 0. By sampling $f_{att}(x)$ at L equally spaced points, we obtain an equivalent attention vector representing our function, $\mathbf{f}_{att} \in \mathbb{R}^L$. Element-wise multiplication can now be used to weigh the original activation vector using its corresponding attention vector:

$$\mathbf{a}^C_{att} = \mathbf{a}^C \odot \mathbf{f}^C_{att} \tag{1}$$

where $\mathbf{a}^C_{att}$ is the attentively regularized activation along the channel C, and $\odot$ denotes the element-wise product.

The key to optimizing the area of specialization of the kernels is now a problem of learning the right parameters to define the function f_{att}.

3.2 Differentiable Functions for Attention

The most obvious choice of f_{att} to create a smooth attention window is the Gaussian function:

$$f_{att}(x; \mu, \sigma) = e^{-(x-\mu)^2/2\sigma^2} \tag{2}$$

This is completely parametrized by two variables, its mean μ and variance σ^2. Every time an update is applied to the convolutional layer weights during backpropagation, we can also update these two parameters in the AR layer. By varying μ, the attention can translate to the optimal location in the sequence, and varying σ^2 allows the layer to learn the optimal scale, i.e., the amount of focus to pay at the chosen location.

We also experimented with Cauchy functions, which have distinctively heavier tails than the corresponding Gaussians as shown in Fig. 2. We premised that this property would improve the gradient flow and help speed up the training of our layers, following [54]. The Cauchy function with mean μ and scale parameter σ is given by:

$$f_{att}(x; \mu, \sigma) = \frac{1}{\left[1 + \left(\frac{x-\mu}{\sigma}\right)^2\right]} \tag{3}$$

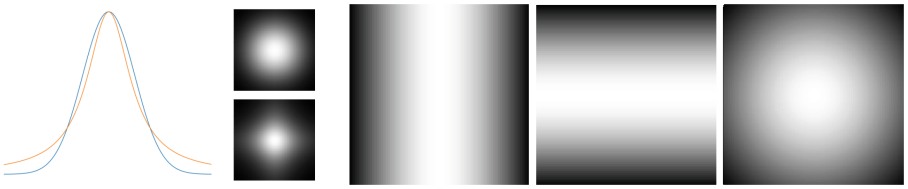

Fig. 2. Left: Gaussian (blue) and Cauchy (orange) attention functions and the equivalent bivariate functions (Gaussian on top). The Cauchy function has more weight in the tail of the distribution. Right: One slice of $\mathbf{F}_x$, $\mathbf{F}_y$ and $\mathbf{F}_{att}$ associated with a single 2D kernel at initialization, using the Gaussian function as f_{att}. Due to linear separability, AR can be trained extremely efficiently. (Color figure online)

3.3 AR in Two Dimensions

The same logic used for one dimension can be generalized to images by considering two-dimensional attention maps associated with each kernel, instead of the attention vectors used for sequences. The input tensor $\mathbf{A} \in \mathbb{R}^{C \times H \times W}$ has slices $\mathbf{A}^C \in \mathbb{R}^{H \times W}$. We build the attention map by sampling $\mathbf{F}^C_{att}$ from a bivariate $\mathcal{F}_{att}(x, y)$ along both dimensions. While using the Gaussian function, this now takes the form:

$$\mathcal{F}_{att}(x, y; \mu_x, \mu_y, \sigma_x, \sigma_y, \rho) = e^{-\alpha(x,y)} \tag{4}$$

where

$$\alpha(x, y) = (f(x))^2 - 2\rho f(x) f(y) + (f(y))^2 \tag{5}$$

$$f(x) = \frac{x - \mu_x}{\sigma_x} \tag{6}$$

$$f(y) = \frac{y - \mu_y}{\sigma_y} \tag{7}$$

The attentively regularized activation $\mathbf{A}^C_{att}$ is now obtained by the same procedure of element-wise multiplication as in Eq. (1).

In our experiments, we found that the correlation parameter ρ introduces an unnecessary degree of freedom to the attention map, as all scales and translations can be achieved by learning only μ_x, μ_y, σ_x and σ_y. Setting $\rho = 0$ allows for more efficiency through a linearly separable implementation. Let the i^{th} row of $\mathbf{A}^C$ be denoted by $\mathbf{a}^{(C,i,:)}$. We initially compute an intermediate activation $\mathbf{A}^C_{int}$ by performing the following operation on all i rows of $\mathbf{A}^C$:

$$\mathbf{a}^{(C,i,:)}_{int} = \mathbf{a}^{(C,i,:)} \odot \mathbf{f}^C_x \tag{8}$$

and then follow up with an operation on each of the j columns of $\mathbf{A}^C_{int}$ to get our final activation $\mathbf{A}^C_{att}$:

$$\mathbf{a}^{(C,:,j)}_{att} = \mathbf{a}^{(C,:,j)}_{int} \odot \mathbf{f}^C_y . \tag{9}$$

Here $\mathbf{f}_x \in \mathbb{R}^H$ and $\mathbf{f}_y \in \mathbb{R}^W$ are simply two separate one-dimensional attention vectors sampled from:

$$f_x(x; \mu_x, \sigma_x) = e^{-(x-\mu_x)^2/2\sigma_x^2} \tag{10}$$

$$f_y(y; \mu_y, \sigma_y) = e^{-(x-\mu_y)^2/2\sigma_y^2} \tag{11}$$

when using the Gaussian function.

3.4 Tensor-Based Implementation

While working with batch-sized tensors, it is more efficient to pre-compute the entire tensor $\mathbf{F}_{att} \in \mathbb{R}^{C\times H\times W}$ directly from the parameter vector of means $\mathbf{m} \in \mathbb{R}^C$ and the vector of scale parameters $\mathbf{s} \in \mathbb{R}^C$ after using tile operations to broadcast them to the required dimensions. The combined tensor of all C attention vectors $\mathbf{f}_{att} \in \mathbb{R}^{C\times H}$ (or $\mathbb{R}^{C\times W}$) can be computed as:

$$\mathbf{f}_{att}(\mathbf{x}; \mathbf{m}, \mathbf{s}) = e^{-(\mathbf{x}-\mathbf{m})^2/2\mathbf{s}^2} \tag{12}$$

Where $\mathbf{x}$ is a range vector (0 to H or 0 to W) scaled to lie in $[0, 1]$. $\mathbf{m}$ is initialized to a vector with each entry 0.5 so the window is initially centered. $\mathbf{s}$ is initialized to a vector of ones, such that the window tapers off from a value of 1 at the center to $f(\sigma = 1)$ at the image boundaries. For the two-dimensional case, $\mathbf{f}_x \in \mathbb{R}^{C\times H}$ and $\mathbf{f}_y \in \mathbb{R}^{C\times W}$ are computed as in Eq. (12), broadcasted into three dimensions ($\mathbb{R}^{C\times H\times W}$), and $\mathbf{F}_{att}$ is computed as

$$\mathbf{F}_{att} = \mathbf{F}_x \odot \mathbf{F}_y \tag{13}$$

This is illustrated in Fig. 2. Every forward pass, an AR layer computes the element-wise product of its input and this attention function. After the backward pass, the function shifts slightly based on the updates to the vectors $\mathbf{m}$ and $\mathbf{s}$. The forward pass layer function is defined as:

$$\mathbf{A}_{att} = \mathbf{A} \odot \mathbf{F}_{att}. \tag{14}$$

In this work, we limit ourselves to AR in two dimensions. Its extension to higher dimensions is trivial, using linearly separable one-dimensional attention windows along each input dimension.

3.5 Efficient Convolutions with Targeting

$\mathbf{F}_{att}$ multiplicatively scales $\mathbf{A}$ in the forward pass. Over training, as the values in $\mathbf{m}$ and $\mathbf{s}$ change, a portion of the activation far enough away from the mean on the attention window gets scaled down to very small values. This effect is magnified when AR is used repeatedly, leading to a large number of near-zero activations through the network.

We exploit the fact that these activations are all located far from the mean, by performing the convolution operation for each kernel in only a rectangular ROI

around the mean. This is mathematically equivalent to using an approximation to $\mathbf{F}_{att}$ for AR, with values below a certain threshold clipped down to zero. We determine this ROI, given by its top-left and bottom-right coordinates:

$$\begin{aligned}\mathbf{roi}^C = &[(\mathbf{m}_x - \frac{\mathbf{s}_x}{\sqrt{2}}) \times W; (\mathbf{m}_y - \frac{\mathbf{s}_y}{\sqrt{2}}) \times H; \\ &(\mathbf{m}_x + \frac{\mathbf{s}_x}{\sqrt{2}}) \times W; (\mathbf{m}_y + \frac{\mathbf{s}_y}{\sqrt{2}}) \times H].\end{aligned} \tag{15}$$

This sliced ROI is used by a *target* layer that efficiently performs the composite operation of both convolution and AR.

$$\mathbf{A}^C_{tar}[\mathbf{roi}^C] = \mathbf{A}^C[\mathbf{roi}^C] * \mathbf{K}^C. \tag{16}$$

$$\mathbf{A}_{att} = \mathbf{A}_{tar} \odot \mathbf{F}_{att} \tag{17}$$

where $\mathbf{K}^C$ is the C^{th} kernel in the target layer, $\mathbf{A}$ is the input activation, $\mathbf{A}_{tar}$ is the intermediate result after the sliced convolution and $\mathbf{A}_{att}$ is the layer output. * denotes the single channel 2D convolution operation.

During training, the values of $\mathbf{m}$ and $\mathbf{s}$ are clipped such that the size of the ROI never collapses to a value smaller than the kernel width. In addition, the overall ROI values are clipped so as to not exceed the boundaries of the input activation. At initialization, the ROI for all kernels is the entire input activation.

In all our experiments, convolutions are done after the required amount of padding at the input boundaries so as to maintain constant spatial dimensions. We do not use an additive bias term in any convolutional layer. Our models were implemented with TensorFlow [1] and Keras [9].

4 Experiments

We empirically demonstrate the effectiveness of TKNs on four tasks: digit recognition on the MNIST [36] and SVHN [45] datasets, traffic sign recognition on the German Traffic Sign Recognition Benchmark [58], and facial analysis on the UNBC-McMaster Pain Archive [43]. We also generate the tlMNIST dataset as a sanity check for TKNs, which aids us in understanding the visualizations of the learned attention mechanism.

4.1 Datasets

MNIST. The MNIST dataset contains 28×28 grayscale images of handwritten numerical digits (0–9). The dataset is divided into 60,000 images for training and 10,000 for testing. The number of images per digit is not uniformly distributed. We perform no data augmentation or preprocessing except division of pixel values by 255 to place them in the range $[0, 1]$.

tlMNIST. The tlMNIST dataset, short for top-left MNIST, is a set of 56×56 grayscale images generated directly from MNIST. The 60,000 training images are created by placing each digit from the training partition of MNIST into the top-left 28×28 quadrant of the images, and selecting 3 other digits from the same partition randomly to place in the other 3 image quadrants. The 10,000 image test set is similarly generated using only the test partition of MNIST. We use identical settings for both MNIST and tlMNIST experiments. The idea behind this task is to introduce a known synthetic 'alignment' to the data, so that it can be used as a sanity check for TKNs (kernels should focus on the top-left). Figure 3 shows some image samples from this dataset.

Fig. 3. tlMNIST data. The label assigned to each sample is that of the number in the top-left quadrant. The other three numbers serve as distractors to vanilla CNNs.

SVHN. The SVHN dataset contains 32×32 RGB digit images, cropped from pictures of house numbers. There are 73,257 images in the training set, 26,032 images in the test set, and 531,131 images for additional training. The digit of interest is centered in the cropped images, but nearby digits and other distractors are kept in the image. We train on only the 73,257 images in the training set, and report performances on the test set. Following [71], we do no preprocessing except pixel intensity scaling.

GTSRB. GTSRB contains RGB images of road traffic signs taken in Germany, with bounding boxes provided for 43 different classes of signs. The main challenges of this dataset are low resolution and contrast. We follow the standard split for evaluation, involving 39,209 training images and a test set of 12,630 images. We preprocess each cropped bounding box by resizing it to 32×32, followed by pixel intensity scaling.

Pain. The Pain Archive is a major publicly available test bed for research in facial analysis of induced pain expression. It consists of 200 video sequences of 25 subjects with 48,398 frames in total, each annotated with 66 facial landmarks and pain intensity levels (on a scale of 0–16). We split off around 30% of the data (sequences of 7 of the subjects) for validation and use the remaining 70% for training. This is a challenging task, which is also well suited to TKNs as we can preprocess the frames to create scale and viewpoint invariance. This is done by using the 66 landmark annotations to warp the faces to a frontal upright reference position before cropping and scaling to 48×48. We perform data augmentation by adding a small Gaussian noise to the landmarks before warping, and also randomly flipping the faces horizontally after warping.

4.2 Training

Our networks on the digit recognition tasks are trained using stochastic gradient descent (SGD). On MNIST and tlMNIST we train using batch size 128 for 20 epochs. The initial learning rate is set to 0.1, and is divided by 10 at the epochs 10 and 15. On SVHN, we train our models for 40 epochs with a batch size of 64. The learning rate is set to 0.1 initially, and is lowered by a factor of 10 after 20 epochs. Following [27], we use a weight decay of 10^{-4} and a Nesterov momentum [61] of 0.9 without dampening.

Table 1. CNN baselines. Convolutional layers replaced in TKNs are marked in **bold**. θ is the compression factor used to reduce the number of channels using a 1×1 convolution at the transition blocks. The activation depths are reduced from C to $(1-\theta)C$ at these layers. LocNet is a small localization network used to perform a learnable affine transform on the input. The final regression or classification layer is an FC layer with dimensions based on the task (10 for MNIST/SVHN, 43 for GTSRB, and 1 for Pain). The softmax cross entropy loss is used for classification, and a Euclidean loss is used for regression.

Layers	Output	CNN6	DN10	DN40	STN
Convolution (1)	$n \times n$	**5 × 5 conv**	3 × 3 conv		LocNet
			[3 × 3 conv] × 2	$\begin{bmatrix} 1\times1 \text{ conv} \\ \mathbf{3\times3\ conv} \end{bmatrix} \times 6$	**7 × 7 conv**
Transition (1)	$\frac{n}{2} \times \frac{n}{2}$	2 × 2 max pool	1 × 1 conv	1 × 1 conv, $\theta = 0.5$	2 × 2 max pool
			2 × 2 avg pool		
Convolution (2)		**5 × 5 conv**	**[3 × 3 conv]** × 2	$\begin{bmatrix} 1\times1 \text{ conv} \\ \mathbf{3\times3\ conv} \end{bmatrix} \times 6$	**5 × 5 conv**
Transition (2)	$\frac{n}{4} \times \frac{n}{4}$	2 × 2 max pool	1 × 1 conv	1 × 1 conv, $\theta = 0.5$	2 × 2 max pool
			2 × 2 avg pool		
Convolution (3)		**5 × 5 conv**	**[3 × 3 conv]** × 2	$\begin{bmatrix} 1\times1 \text{ conv} \\ \mathbf{3\times3\ conv} \end{bmatrix} \times 6$	**3 × 3 conv**
Flatten	1 × 1	328D FC 192D FC	$\frac{n}{4} \times \frac{n}{4}$ global avg pool		2 × 2 max pool 96D FC

On GTSRB and Pain, we use the Adam optimizer [32] with a learning rate of 0.001, and train for a total of 100 epochs with a batch size of 64. For GTSRB, we use a higher weight decay of 0.05. We adopt the weight initialization introduced by He et al. [24]. We checkpoint the models after every epoch of training and report the error rates of the best single model. Test errors were only evaluated once for each task and model setting.

4.3 Network Architectures

To show that the benefits of AR are model-agnostic, we use four different CNN baselines across experiments. They are summarized in Table 1. The first is a vanilla 6-layer CNN network with 3 convolutional layers (with 256, 256 and 128 kernels respectively) and 3 fully connected (FC) layers. The last layer is

Table 2. Error rates (%) on the MNIST dataset. Our best results in **bold**. AR improves both performance and efficiency.

Network	Params	FLOPs	Error
Network in network [41]	-	-	0.47
Deeply-supervised nets [37]	-	-	0.39
Competitive multi-scale convolution [40]	4.48M	632M	0.33
CapsNet [50]	8.21M	202M	0.25
CNN6	4.59M	368M	0.42
TKN6 (Gaussian, $L_2 = 10^{-4}, \beta = 2$)	4.59M	52.9M	0.48
TKN6 (Cauchy, $L_2 = 10^{-4}, \beta = 4$)	4.59M	28.6M	0.43
DN10	44.7K	11.3M	0.48
TDN10 (Gaussian, $L_2 = 10^{-4}, \beta = 2$)	45.0K	6.93M	0.42
TDN10 (Cauchy, $L_2 = 10^{-4}, \beta = 4$)	45.0K	**6.26M**	**0.38**

regularized with a dropout [57] of 0.5, and the ReLU non-linearity is used for all intermediate layers.

The second is a DenseNet [27] with 3 densely connected blocks of 2 layers each. We use a growth rate of 12 and do not perform compression at the transition layers between blocks. We denote this model as DN10. Note that all convolutions in DenseNets are actually performed as the composite function, Batch Normalization [28] – ReLU – convolution.

We use a single baseline for our SVHN and Pain Archive experiments, a DenseNet-BC architecture with 3 blocks of 12 layers each. There are 21 connections in each block. We use a growth rate of 36, dropout with probability 0.2 after each convolution, and a compression factor of 0.5 at the 2 transition layers. We denote this model as DN40.

The final baseline is a Spatial Transformer Network (STN) [29] for GTSRB. This network learns how to warp the inputs with an affine transformation such that they are ideally aligned for the task. This meshes well with TKNs which are designed for aligned data. The main network we use is a 5-layer CNN with 3 convolutional layers (with 128, 128 and 256 kernels respectively) and 2 FC layers. We use batch normalization between all intermediate layers, and a dropout of 0.6 for the final FC layer. The localization network that computes warp parameters is a smaller version of the same network, with 3 convolutional layers of the same kernel size (with 16, 32 and 64 kernels respectively) and 3 FC layers (128, 64 and 6 units).

Given a CNN baseline, converting it to an equivalent TKN involves replacing convolutional layers with target layers. For the CNN6 and STN baselines, we simply replace all the convolutional layers in the main network, giving TKN6 and TSTN. For the DenseNet baselines, we replace the 3×3 convolutional layers within the dense blocks, assuming that the bulk of the representation learning

Table 3. Error rates (%) on the tlMNIST dataset. Our best results in **bold**. With AR, performance on tlMNIST becomes equivalent to the standard MNIST task.

Network	Params	FLOPs	Error
CNN6	10.76M	1470M	0.83
TKN6 (Gaussian, $L_2 = 10^{-4}, \beta = 2$)	10.76M	145M	0.48
TKN6 (Cauchy, $L_2 = 10^{-4}, \beta = 2$)	10.76M	125M	0.48
TKN6 (Cauchy, $L_2 = 10^{-4}, \beta = 4$)	10.76M	68.3M	0.53
DN10	44.7K	45.2M	0.50
TDN10 (Gaussian, $L_2 = 10^{-4}, \beta = 2$)	45.0K	29.7M	0.41
TDN10 (Cauchy, $L_2 = 10^{-4}, \beta = 2$)	45.0K	**25.6M**	**0.38**

Table 4. Error rates (%) on the SVHN dataset. Our best results in **bold**. We obtain state-of-the-art results on this reduced SVHN training set.

Network	Params	FLOPs	Error
CapsNet [50]	1.00M	41.3M	4.25
DN40	0.83M	357M	3.17
TDN40 (Cauchy, $L_2 = 10^{-4}, \beta = 2$)	0.83M	**205M**	**3.11**

happens in these layers. We keep the initial, transitional and bottleneck 1×1 convolutions as they are. We call these Targeted DenseNets (TDN10 and TDN40).

Further, there are three design choices within a target layer which we vary– the choice of attention function (Gaussian or Cauchy); an L_2 weight penalty on scale parameters $\mathbf{s}_x$ and $\mathbf{s}_y$ to encourage more 'targeted' or 'focused' representations; and a multiplicative factor β we build up the L_2 penalty by as we go deeper into the network, based on the intuition that deeper layers benefit less from weight sharing than shallow ones. This build up factor is applied by scaling the L_2 penalty by a factor of β for all layers in the Convolution (2) block, and β^2 for all layers in the Convolution (3) block of the network.

4.4 Results

We compare our results on MNIST to other approaches that use single models with no data augmentation in Table 2. Our best model does better than all previous CNN based methods on MNIST except a competitive multi-scale convolutional approach [40]. We are also outperformed by CapsNets [50], a new kind of neural network and not a drop in modification like AR. Both these network types have far more parameters and computational expenses than ours.

The TKNs corresponding to the CNN6 baseline (TKN6) match its performance, coupled with a huge boost in efficiency (**13× less floating point operations** in the forward pass). Introducing target layers benefits both the efficiency and performance when used with the DN10 models (**21% reduced error rate**).

Table 5. Error rates (%) on the GTSRB dataset. We achieve comparable performance with nearly a 3× reduction in #FLOPS.

Network	Params	FLOPs	Error
STN	1.18M	145M	1.52
TSTN (Cauchy, $L_2 = 0.001, \beta = 2$)	1.18M	55.7M	1.53

Table 6. Regression errors (on a scale of 0–16) on the UNBC-McMaster Pain Archive. Here, we achive a 2× reduction in #FLOPS without loss in performance.

Network	Params	FLOPs	MSE	MAE
DN40	0.83M	802M	1.67	0.51
TDN40 (Cauchy, $L_2 = 10^{-4}, \beta = 4$)	0.83M	391M	1.67	0.50

The results on tlMNIST are shown in Table 3. Since the input data is highly 'aligned', we see significant improvement in results for both baselines. Another interesting observation is that the performance of the best networks on tlMNIST matches the MNIST results, showing that the effect of additional distractors has been completely negated by AR.

The results on SVHN are shown in Table 4. Since models with the Gaussian attention function were far more difficult to tune in experiments on MNIST, we fix the Cauchy attention function for the remaining experiments. We obtain the best reported results (to our knowledge) on the reduced SVHN dataset where the extra training images are not used.

The classification errors on the GTSRB test set are shown in Table 5. We also report the mean squared error (MSE) and mean average error (MAE) for regression on the validation partition of the Pain Archive in Table 6. On both tasks, we see distinctive benefits in terms of efficiency without loss in performance, showing the applicability of AR to network acceleration on practical tasks. Because we adopted hyper-parameter settings optimized for the CNN baselines in our study, we believe that further gains in accuracy of TKNs may be obtained by more detailed tuning of hyper-parameters and learning rate schedules.

5 Discussion

Figure 4 shows the attention maps $\mathbf{F}_{att}$ learned by the TDN10 models corresponding to each of the six target layers. Our experimental results combined with these visualizations give us some insight into the role of attention in CNN architectures.

Implicit Attention in CNNs. A surprising observation is the near-identical error rate of the DN10 baseline on both MNIST (0.48%) and tlMNIST (0.50%). The network has no explicit way to pay more attention to any part of the input

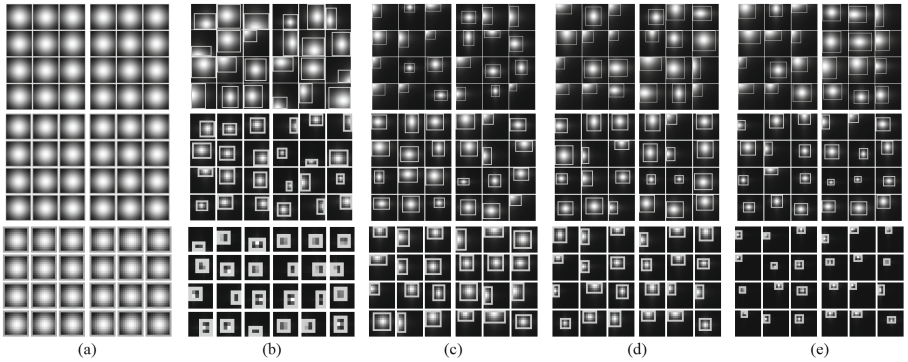

Fig. 4. Attention maps using the *Cauchy* function. (a) Initialization. (b) After training on MNIST, $L_2 = 10^{-4}, \beta = 4$. We notice large portions of the attention maps are vacant, particularly in the deeper layers. (c) After training on tlMNIST, $L_2 = 2 \times 10^{-4}, \beta = 1$. (d) tlMNIST, $L_2 = 10^{-4}, \beta = 2$. Though (c) and (d) have similar computational costs, (d) obtains slightly better performance. (e) tlMNIST, $L_2 = 10^{-4}, \beta = 4$. Slightly better gains in efficiency can be obtained by scaling β instead of L_2.

images, since it has no max pooling or FC layers. This means that for the tlMNIST task, the convolutional architecture itself learns to 'attend' to only the top-left portion of the image. This is possible because of the large convolutional receptive fields of the deeper layers. Each unit in the final convolutional layer has an effective receptive field larger than the entire input image. For tlMNIST, these units can learn locations associated with a given handwritten digit by simply looking for not just the digit, but a formation of the digit and a pattern associated with its location, such as the empty space to its bottom right and some portion of the three random digits around it. This is still an inconvenient task, which is why TKNs significantly improve the baseline (24% reduction in error rate). The size of the receptive fields explains why the attention maps of the deeper layers in Fig. 4(c), (d) and (e) are not all on the extreme top-left portion of the image.

Fully Convolutional TKNs. Each TKN kernel location is parametrized by $\mathbf{m}_x$, $\mathbf{m}_y$, $\mathbf{s}_x$ and $\mathbf{s}_y$, which are all relative values with respect to the absolute height and width of the image. Spatial structure in terms of layout is the crucial ingredient in the performance of TKNs, and if this remains similar, they can be applied to images of varying sizes and aspect ratios by using the same relative learned parameters scaled as per the new input resolution. To apply a TKN in a fully convolutional manner over a large image (for example, as a face detector), we first convert the relative parameters to absolute parameters by choosing a scaling for the attention layers in our fully convolutional TKN. This means a TKN learned at any resolution can be specialized to any other resolution by adjusting the chosen parameter scaling.

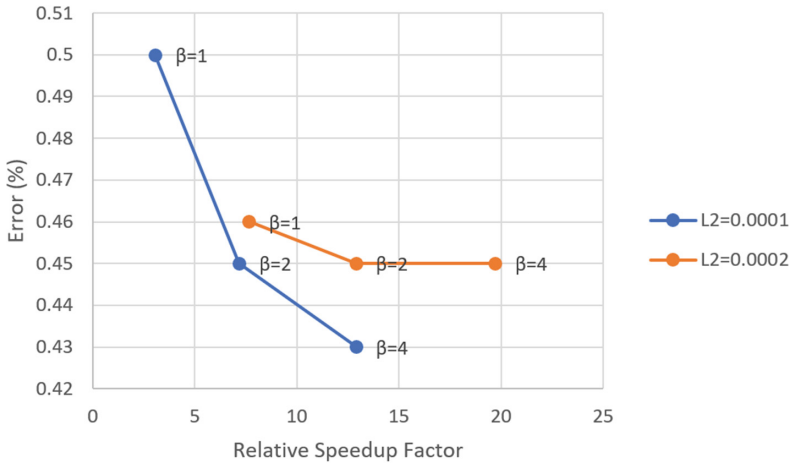

Fig. 5. Effect of L_2 and β on performance and efficiency. Relative speedup factor is the ratio of #FLOPs between the network and the CNN6 baseline. Better speed-performance tradeoffs are achieved through larger values of β.

Network Interpretability. In traditional CNNs, we have seen how a deeper convolutional kernel may represent a mixture of patterns using its implicit attention and large receptive field. For example, when dealing with facial images, a kernel may learn to be activated by a certain combination of the eyes and mouth. Such complex knowledge representations greatly decrease the interpretability of the network [72]. By introducing attention explicitly, kernels in TKNs can be encouraged to look at tiny areas in the inputs by increasing L_2. This makes them much more likely to be associated with single objects or parts, increasing the network interpretability. This is of great value when we need humans to trust a network's predictions.

Network Acceleration. Figure 5 shows the performance of TKN6 on MNIST as the L_2 penalty is varied. We see that a trade-off between speed and accuracy can be tuned by adjusting this penalty term while training. We also see that building up the L_2 penalty gradually over depth using β improves performance in comparison to having a fixed penalty throughout the network. This validates our assumption that deeper, more abstract features require less weight sharing.

6 Conclusion

We proposed a new regularization method for CNNs called Attentive Regularization. It constrains the activation maps throughout the network to lie within specific ROIs associated with each kernel. This is done through a simple yet powerful modification of the convolutional layers, retaining end-to-end trainability with backpropagation. In our experiments, TKNs give a consistent improvement

in efficiency over baselines in synthetic and natural settings, and competitive results to the state-of-the-art on benchmark datasets. Our experiments validate the idea that simplifying soft attention mechanisms to specific parametric distributions has potential for significant network acceleration.

In this study, we optimize for the attention parameters **m** and **s** for each kernel directly during training. In future work, we aim to study the effect of generating these parameters adaptively per image. Another extension to the proposed variant of TKNs would be to model the attention with a more complex function (like a mixture of Gaussians), or to use multiple kernels with different attention maps for the same output channel, making them deformable [15]; to handle complex images where a single ROI per kernel may be insufficient.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous distributed systems. CoRR abs/1603.04467 (2016)
2. Almahairi, A., Ballas, N., Cooijmans, T., Zheng, Y., Larochelle, H., Courville, A.C.: Dynamic capacity networks. CoRR abs/1511.07838 (2015)
3. Ba, J., Mnih, V., Kavukcuoglu, K.: Multiple object recognition with visual attention. CoRR abs/1412.7755 (2014)
4. Cao, C., et al.: Look and think twice: capturing top-down visual attention with feedback convolutional neural networks. In: ICCV (2015)
5. Chen, L., et al.: SCA-CNN: spatial and channel-wise attention in convolutional networks for image captioning. In: CVPR (2017)
6. Chen, T., Goodfellow, I.J., Shlens, J.: Net2Net: accelerating learning via knowledge transfer. CoRR abs/1511.05641 (2015)
7. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. CoRR abs/1504.04788 (2015)
8. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. ArXiv e-prints (2017)
9. Chollet, F.: Keras (2015). https://github.com/fchollet/keras
10. Denil, M., Bazzani, L., Larochelle, H., de Freitas, N.: Learning where to attend with deep architectures for image tracking. Neural Comput. **24**, 2151–2184 (2012)
11. Denton, E., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. CoRR abs/1404.0736 (2014)
12. Dieleman, S., Fauw, J.D., Kavukcuoglu, K.: Exploiting cyclic symmetry in convolutional neural networks. CoRR abs/1602.02660 (2016)
13. Dong, X., Huang, J., Yang, Y., Yan, S.: More is less: a more complicated network with less inference complexity. CoRR abs/1703.08651 (2017). http://arxiv.org/abs/1703.08651
14. Ekman, P., Friesen, W., Hager, J.: Facs manual. In: A Human Face (2002). https://www.scirp.org/(S(i43dyn45teexjx455qlt3d2q))/reference/ReferencesPapers.aspx?ReferenceID=1850657
15. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**, 1627–1645 (2010)

16. Figurnov, M., Vetrov, D.P., Kohli, P.: PerforatedCNNs: acceleration through elimination of redundant convolutions. CoRR abs/1504.08362 (2015). http://arxiv.org/abs/1504.08362
17. Fu, J., Zheng, H., Mei, T.: Look closer to see better: recurrent attention convolutional neural network for fine-grained image recognition. In: CVPR (2017)
18. Girdhar, R., Ramanan, D.: Attentional pooling for action recognition. In: NIPS (2017)
19. Girshick, R.B.: Fast R-CNN. CoRR abs/1504.08083 (2015)
20. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR abs/1311.2524 (2013)
21. Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., Wierstra, D.: DRAW: a recurrent neural network for image generation. In: ICML (2015)
22. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural network with pruning, trained quantization and huffman coding. CoRR abs/1510.00149 (2015)
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
24. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. CoRR abs/1502.01852 (2015)
25. Hendricks, L.A., Venugopalan, S., Rohrbach, M., Mooney, R.J., Saenko, K., Darrell, T.: Deep compositional captioning: describing novel object categories without paired training data. CoRR abs/1511.05284 (2015)
26. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. ArXiv e-prints (2015)
27. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR, July 2017
28. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167 (2015)
29. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. CoRR abs/1506.02025 (2015)
30. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. CoRR abs/1405.3866 (2014)
31. Kawaguchi, K., Kaelbling, L.P., Bengio, Y.: Generalization in deep learning. ArXiv e-prints (2017)
32. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014)
33. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
34. Larochelle, H., Hinton, G.E.: Learning to combine foveal glimpses with a third-order Boltzmann machine. In: NIPS (2010)
35. Lebedev, V., Lempitsky, V.S.: Fast ConvNets using group-wise brain damage. CoRR abs/1506.02515 (2015)
36. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2324 (1998)
37. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. ArXiv e-prints (2014)
38. Li, W., Abtahi, F., Zhu, Z.: Action unit detection with region adaptation, multi-labeling learning and optimal temporal fusing. CoRR abs/1704.03067 (2017)
39. Li, W., Abtahi, F., Zhu, Z., Yin, L.: EAC-NET: a region-based deep enhancing and cropping approach for facial action unit detection. CoRR abs/1702.02925 (2017)

40. Liao, Z., Carneiro, G.: Competitive multi-scale convolution. CoRR abs/1511.05635 (2015)
41. Lin, M., Chen, Q., Yan, S.: Network in network. CoRR abs/1312.4400 (2013)
42. Lu, J., Xiong, C., Parikh, D., Socher, R.: Knowing when to look: adaptive attention via a visual sentinel for image captioning. In: CVPR (2017)
43. Lucey, P., Cohn, J.F., Prkachin, K.M., Solomon, P.E., Matthews, I.: Painful data: the UNBC-McMaster shoulder pain expression archive database. In: Face and Gesture 2011 (2011)
44. Nam, H., Ha, J.W., Kim, J.: Dual attention networks for multimodal reasoning and matching. In: CVPR (2017)
45. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
46. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: imagenet classification using binary convolutional neural networks. CoRR abs/1603.05279 (2016)
47. Ren, M., Pokrovsky, A., Yang, B., Urtasun, R.: SBNet: sparse blocks network for fast inference. CoRR abs/1801.02108 (2018). http://arxiv.org/abs/1801.02108
48. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR abs/1506.01497 (2015)
49. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: FitNets: hints for thin deep nets. CoRR abs/1412.6550 (2014)
50. Sabour, S., Frosst, N., Hinton, G.E.: Dynamic routing between capsules. ArXiv e-prints (2017)
51. Schmidhuber, J., Huber, R.: Learning to generate artificial fovea trajectories for target detection. Int. J. Neural Syst. **2**, 125–134 (1991)
52. Seo, P.H., Lin, Z., Cohen, S., Shen, X., Han, B.: Hierarchical attention networks. CoRR abs/1606.02393 (2016)
53. Shih, K.J., Singh, S., Hoiem, D.: Where to look: focus regions for visual question answering. CoRR abs/1511.07394 (2015)
54. Shyam, P., Gupta, S., Dukkipati, A.: Attentive recurrent comparators. In: ICML (2017)
55. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
56. Srinivas, S., Babu, R.V.: Data-free parameter pruning for deep neural networks. CoRR abs/1507.06149 (2015)
57. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
58. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: The German traffic sign recognition Benchmark: a multi-class classification competition. In: IEEE International Joint Conference on Neural Networks (2011)
59. Stollenga, M.F., Masci, J., Gomez, F.J., Schmidhuber, J.: Deep networks with internal selective attention through feedback connections. CoRR abs/1407.3068 (2014)
60. Sun, Y., Wang, X., Tang, X.: Deep learning face representation from predicting 10,000 classes. In: CVPR (2014)
61. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: ICML (2013)
62. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR abs/1602.07261 (2016)
63. Tai, C., Xiao, T., Wang, X., E, W.: Convolutional neural networks with low-rank regularization. CoRR abs/1511.06067 (2015)

64. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: closing the gap to human-level performance in face verification. In: CVPR (2014)
65. Wu, B., Iandola, F.N., Jin, P.H., Keutzer, K.: SqueezeDet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. CoRR abs/1612.01051 (2016)
66. Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., Zhang, Z.: The application of two-level attention models in deep convolutional neural network for fine-grained image classification. CoRR abs/1411.6447 (2014)
67. Xiong, C., Merity, S., Socher, R.: Dynamic memory networks for visual and textual question answering. CoRR abs/1603.01417 (2016)
68. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. CoRR abs/1502.03044 (2015)
69. Yang, Z., He, X., Gao, J., Deng, L., Smola, A.: Stacked attention networks for image question answering. In: CVPR (2016)
70. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. CoRR abs/1612.03928 (2016)
71. Zagoruyko, S., Komodakis, N.: Wide residual networks. CoRR abs/1605.07146 (2016)
72. Zhang, Q., Wu, Y.N., Zhu, S.: Interpretable convolutional neural networks. CoRR abs/1710.00935 (2017)
73. Zhao, K., Chu, W.S., Zhang, H.: Deep region and multi-label learning for facial action unit detection. In: CVPR (2016)