In Proceedings of the IEEE International Conference on Computer Vision (ICCV) 2017.

A Spatiotemporal Oriented Energy Network for Dynamic Texture Recognition

Isma Hadji
York University, Toronto
hadjisma@cse.yorku.ca

Richard P. Wildes
York University, Toronto
wildes@cse.yorku.ca

Abstract

This paper presents a novel hierarchical spatiotemporal orientation representation for spacetime image analysis. It is designed to combine the benefits of the multilayer architecture of ConvNets and a more controlled approach to spacetime analysis. A distinguishing aspect of the approach is that unlike most contemporary convolutional networks no learning is involved; rather, all design decisions are specified analytically with theoretical motivations. This approach makes it possible to understand what information is being extracted at each stage and layer of processing as well as to minimize heuristic choices in design. Another key aspect of the network is its recurrent nature, whereby the output of each layer of processing feeds back to the input. To keep the network size manageable across layers, a novel cross-channel feature pooling is proposed. The multilayer architecture that results systematically reveals hierarchical image structure in terms of multiscale, multiorientation properties of visual spacetime. To illustrate its utility, the network has been applied to the task of dynamic texture recognition. Empirical evaluation on multiple standard datasets shows that it sets a new state-of-the-art.

1. Introduction

Hierarchical representations play an important role in computer vision [30]. The challenge of extracting useful information (*e.g*. objects, materials and environmental layout) from images has led to incremental recovery of progressively more abstracted representations. Convolutional networks (ConvNets) provide an interesting contemporary example of this paradigm yielding state-of-the-art results on a range of classification and regression tasks, *e.g*. [18, 36, 43, 8]. While such learning-based approaches show remarkable performance, they typically rely on massive amounts of training data and the exact nature of their representations often remains unclear. Deeper theoretical understanding should lessen dependence on data-driven design, which is especially important when training data is limited. In complement, the present work explores a more controlled approach to network realization. A small vocabulary of theory motivated, analytically defined filtering operations are repeatedly cascaded to yield hierarchical representations of input imagery. Although the same operations are applied repeatedly, different information is extracted at each layer as the input changes due to the previous layer's operations. Since the primitive operations are specified analytically, they do not require training data and the resulting representations are readily interpretable. Further, the network yields state-of-the-art results on the important and challenging task of dynamic texture recognition.

Previous work pursuing controlled approaches to hierarchical network realization variously relied on biological inspiration and analytic principles. Most biologically-based approaches mimic the multilayered architecture of the visual cortex with cascades of simple and complex cells [29, 24, 16]. Typically, these approaches still use learning and leave open questions regarding the theoretical basis of their designs. More theoretically driven approaches typically focus on network architecture optimization, *e.g*. the number of filters/layer or the number of learned weights [19, 9]. Other work made use of predetermined filters at all layers, as learned via PCA [4] or over a basis of 2D derivatives [14]. Two commonalities appear across these efforts. First, they address only a single aspect of their architecture, while assuming all others are fixed. Second, they rely on learning in optimization. More closely related to the present work is ScatNet [3]. This network is rigorously defined to increase invariance to signal deformations via hierarchical convolution with filters of different frequency tunings. In its restricted application to 2D spatial images, ScatNet bases its design on optimization with respect to 2D invariances. In contrast, the present approach considers 3D spacetime images, which leads to a spatiotemporal orientation analysis for extracting varying dynamic signal properties across levels, including invariance maximization via a multiscale network. Moreover, while the present approach analytically specifies the number of filters/layer, ScatNet's choice of wavelets limits it from analytically specifying the number of filters/layer, which instead are chosen empirically.

Dynamic texture recognition also has received much attention. Here, features used can be largely categorized as learned vs. hand-crafted. Those in the former category rely on either autoregressive [35] or Linear Dynamical Systems (LDS) [32]. Recent trends typically rely on LDS with dictionary learning [23, 12, 27]. The main downside of such approaches comes from their inability to represent patterns beyond their training data. In contrast, hand-crafted approaches to dynamic texture analysis typically eschew modeling the underlying dynamical system in favor of more directly encoding the spacetime signal with an attempt to balance discriminability and generalizability. Such approaches can be subcategorized according to treating each frame as a static texture [11, 40], 3D Local Binary Patterns [42, 28], reliance on optical flow [25, 21] and building more directly on spatiotemporal filtering [5, 39, 17, 38, 7]. This last approach is most akin to the present work, as it also has at its core the application of spatiotemporal filters to data streams. In contrast, the proposed approach exploits repeated application of such filters and combination of their outputs at each layer to extract progressively more abstract information.

Contributions. In light of previous research, the contributions of this paper are as follows. **1)** A novel processing network, based on a repeated spatiotemporal oriented energy analysis, is presented where the layers interact via a recurrent connection so the output feeds back to the input. The resulting multilayer architecture systematically reveals the hierarchical structure in its input. Exposed properties progress from simple and local to abstract and global across layers, but are always interpretable in terms of the network's explicit design. **2)** At every layer, extracted feature maps are combined via cross-channel pooling to yield coherent groups based on the employed filters. This innovation constrains the representation dimensionality while maintaining interpretability and high discriminating power. **3)** Every stage of processing in the network is designed based on theoretical considerations. Ties to biological modeling are also established. This design approach removes the need for a learning phase, which is not always feasible, *e.g*., when confronted with modest datasets. **4)** The resulting network is used as a novel approach to representation and recognition of dynamic textures (DTs). In empirical evaluation on standard datasets, the system achieves superior performance to virtually all current alternative DT recognition approaches. Code is available at https://github.com/hadjisma/soe-net.

2. Technical approach

The proposed network architecture is designed to capture spatiotemporal image structure across multiple layers of processing as shown in Fig. 1. The input to the system is a three-dimensional, $\mathbf{x} = (x, y, t)^\top$, spacetime volume, $V(\mathbf{x})$, and the output is a volume of feature maps, $\boldsymbol{F}(\mathbf{x})$, that capture the spatiotemporal structure. Each processing layer, $\mathcal{L}_k$, is comprised of a sequence of four stages: convolution, rectification, normalization and pooling. A key novelty of the approach is the repeated filtering, whereby

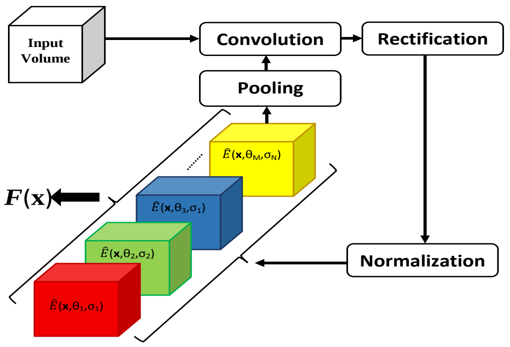
Input Volume
Convolution
Rectification
Pooling
$\hat{E}(\mathbf{x}, \theta_M, \sigma_N)$
$\hat{E}(\mathbf{x}, \theta_3, \sigma_1)$
$\hat{E}(\mathbf{x}, \theta_2, \sigma_2)$
$\hat{E}(\mathbf{x}, \theta_1, \sigma_1)$
$\boldsymbol{F}(\mathbf{x})$
Normalization

Figure 1. Overview of the **SOE-Net** Architecture. The same set of operations are repeatedly applied via a recurrent connection; however, different information, $\boldsymbol{F}$, is extracted at each pass as the input changed due to the operations of the previous pass.

the final processing stage of each layer (pooling) feeds back to the initial processing stage (convolution) to yield a subsequent layer of processing, $\mathcal{L}_{k+1}$. The entire process is repeated K times, after which the energy remaining in the signal about to be fed back through has essentially vanished. The final output of the network, $\boldsymbol{F}(\mathbf{x})$, is the set of feature maps extracted at the K^{th} layer. Each layer in the network corresponds to Spatiotemporal Oriented Energy filtering; therefore, the network is dubbed **SOE-Net**.

2.1. Repeated filtering

Key to the success of multilayer architectures (*e.g*. ConvNets) is their ability to extract progressively more abstract attributes of their input at each successive layer and thereby yield a powerful data representation. As a simple example, a degree of shift-invariance emerges via linear filtering, followed by nonlinear rectification and pooling: The exact position of the extracted features becomes immaterial across the pooling support. A similar interpretation becomes more subtle, however, as the entire filter-rectify-pool block of operations is repeated, especially when using learned filters.

Consider the simple example of Fig. 2. The left side depicts a sinusoidal pattern alternately moving rightward and staying static across time. On the right, the same pattern is moving rightward behind a similarly textured static picket fence. An initial stage of directional (motion) filtering cannot extract the difference in the overall dynamic behavior of the two patterns. However, further directional filtering that operates on the output of the initial layer can detect the overall patterns and thereby allows to make the distinction between the two different dynamic textures. More generally, this example shows how powerful features can emerge across multiple layers of processing: The exposed properties progress from simple and local to abstract and global.

Motivated by these observations, the proposed SOE-Net is designed to extract progressively more abstract representations of the input signal at each layer, while maintaining interpretability. To achieve these ends, repeated filtering is employed via a recurrent connection, whereby the output of

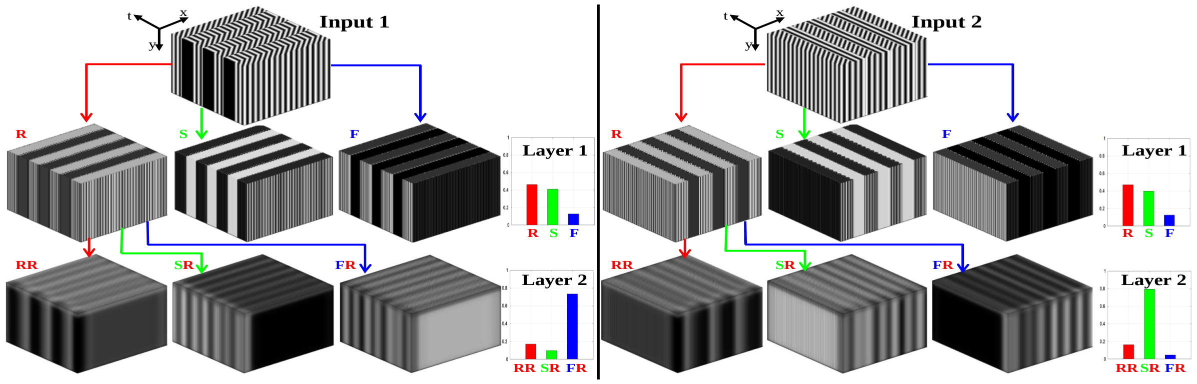

Figure 2. Emergence of Abstract Features via Repeated Filtering. (Left) Input synthetic sinusoidal pattern alternately moves right (orientation along x-t diagonal) and stays static (orientation parallel to time axis). (Right) Same pattern moving right behind a static picket fence with same spatial pattern (*i.e.* background motion viewed between the spaces in a stationary picket fence). **SOE-Net** is used with filters tuned to **R**ightward motion, **S**tatic (no motion) and **F**lickering (pure temporal change). **Layer 1** captures the local rightwardly moving and static portions; but not the more abstract alternating temporal move-stop pattern of the left input, nor the fact that the right input maintains the same spatially interleaved moving and static stripes. Indeed, measurements aggregated over the volumes (shown as histograms on the right) are the same at this layer. The difference is revealed at **L2** after applying the same filters on the **R** response of **L1**: The move-stop behavior of the left texture becomes explicit and yields a large **F** response. In contrast, the constant rightward motion and picket fence of the right texture yield a large **S** response. In practice, more directional filters are applied at each layer; see Sec. 2.2.

layer $\mathcal{L}_k$, denoted as $\boldsymbol{L}_k$, feeds back to the initial convolutional stage (*i.e.* convolution with the same filter set) to yield a subsequent layer of processing, $\mathcal{L}_{k+1}$. Since the processing at each layer is defined precisely (see following subsections), it will be interpretable. Also, since it is applied repeatedly, abstraction emerges. This process is depicted in Fig. 3 with an unfolded recurrence, and symbolized as

$$\boldsymbol{L}_{k+1} = \mathcal{L}_{k+1}\left(\boldsymbol{L}_k\right), \tag{1}$$

with $\boldsymbol{L}_1 = \mathcal{L}_1\left(V(\mathbf{x})\right)$. Interestingly, biological models have advocated that similar processing (termed $Filter \rightarrow Rectify \rightarrow Filter$) takes place in visual cortex to deal with higher-order image structures [2].

2.2. Convolution

Convolution serves to make local measurements revealing salient properties of its input. Given a temporal sequence of images, local spatiotemporal orientation is of fundamental descriptive power, as it captures the 1^{st}-order correlation structure of the data irrespective of the underlying visual phenomena [1, 37, 5]. Also, such measures provide a uniform way to capture spatial appearance and dynamic properties of the underlying structure. Spatial patterns (*e.g.* static texture) are captured as the filters are applied within the image plane. Dynamic attributes of the pattern are obtained by filtering at orientations extending into time.

Based on the above theoretical motivations, in this work convolution is designed to extract local measurements of multiscale, spatiotemporal orientation. Still, having committed to convolution that captures spatiotemporal orientation, a variety of options exist for specifying an exact set of filters, *e.g.* Gabor, lognormal, wavelet or Gaussian derivatives. Here, Gaussian derivative filters are selected for two main reasons, *c.f.* [10]. First, for any given order of Gaus-

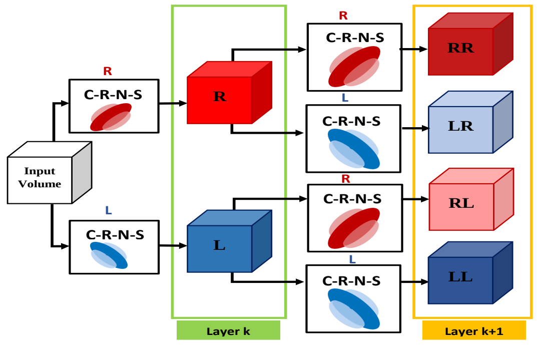

Figure 3. Unfolding the **SOE-Net** Recurrent Connection. Local spatiotemporal features at various orientations are extracted with an initial processing layer, $\mathcal{L}_k$. C-R-N-S indicate Convolution, Rectification, Normalization and Spatiotemporal pooling, as detailed in Secs. 2.2, 2.3, 2.4 and 2.5, resp., while R and L indicate rightward vs. leftward filtered data, resp., with symbol strings (*e.g.* LR) indicating multiple filterings. A network with only 2 filters (*i.e.* 2 orientations) is shown for illustration. Each of the feature maps at layer $\mathcal{L}_k$ is treated as a new separate signal and fed back to layer $\mathcal{L}_{k+1}$ to be convolved with the same set of filters but at a different effective resolution due to spatiotemporal pooling.

sian derivative, a set of basis filters that allow synthesis of the response at an arbitrary orientation can be specified. This property makes it possible to set the exact number of filters used at each layer on a theoretical basis, rather than experiments or learning. Second, these filters are separable, which provides efficient implementation. In particular, 3D Gaussian 3^{rd}- order derivative filters are used, $G_{3D}^{(3)}(\theta_i, \sigma_j)$, with θ_i and σ_j denoting the 3D filter orientation and scale, resp. Given an input spacetime volume, $V(\mathbf{x})$, a set of output volumes, $C(\mathbf{x}; \theta_i, \sigma_j)$, are produced as

$$C(\mathbf{x}; \theta_i, \sigma_j) = G_{3D}^{(3)}(\theta_i, \sigma_j) * V(\mathbf{x}), \tag{2}$$

with $*$ denoting convolution. Since 3^{rd}- order Gaussian filter are used, $M = 10$ filters are required to span the space of orientations [10]. The directions, θ, are chosen to uniformly sample 3D orientations as the normals to the faces of an icosahedron, with antipodal directions identified. The number of scales, σ, is determined by the size of the spacetime volume to be analyzed; details are provided in Sec. 2.7.

2.3. Rectification

The output of the convolutional stage, $C(\mathbf{x}; \theta_i, \sigma_j)$, is comprised of positive and negative responses. Both indicate a contrast change along the direction of the oriented filters and hence structure along that direction. It is therefore interesting to keep the distinction between the two responses. Also, in anticipation of the subsequent pooling stage of processing, it is critical to perform some type of rectification. Otherwise, the pooling across positive and negative responses will attenuate the signal. By squaring the individual responses, it is possible to consider the results in terms of spectral energy, *e.g.* [37]. Based on these considerations, the present approach makes use of a nonlinear operation that splits the signal into two paths: The first path carries the positive responses, while the second carries the negative responses, both of which are pointwise squared to relate to spectral energy measurements to yield

$$\begin{aligned} E^+(\mathbf{x}; \theta_i, \sigma_j) &= (\max[C(\mathbf{x}; \theta_i, \sigma_j), 0])^2 \\ E^-(\mathbf{x}; \theta_i, \sigma_j) &= (\min[C(\mathbf{x}; \theta_i, \sigma_j), 0])^2 \end{aligned} \quad . \tag{3}$$

Interestingly, biological findings suggest a model for cortical simple cells that includes a nonlinearity in the form of two half wave rectifications that treat the positive and negative outputs in two different paths [13]. Similarly, recent ConvNet analysis also revealed that learned filters tend to form pairs with opposite phases [34].

Beyond physical interpretation (signed energy) and relation to biology, use of the squaring function at this stage is advantageous as it allows for keeping track of the frequency content of the processed signal (*i.e.* doubling the maximum frequency present). This information plays an important role in specifying the pooling stage parameters, detailed below. Given that the following processing stages treat the two paths in the exact same way, in the remainder of the paper the $+$ and $-$ superscripts will be suppressed when referring to energy measurements, $E(\mathbf{x}; \theta_i, \sigma_j)$.

2.4. Normalization

Owing to the bandpass nature of the filtering used in the convolutional stage, (2), the responses are invariant to overall additive brightness offsets. Still, the responses remain sensitive to multiplicative contrast variations, *i.e.* the filter response increases with local contrast independent of local orientation structure. Moreover, the responses after rectification, (3), are unbounded from above, which makes practical signal representation challenging. Divisive normalization is a way to correct for these difficulties. Significantly, it also serves to lessen statistical dependencies present in natural images via signal whitening [22]. Therefore, the next stage operates via pointwise division of the rectified measurements by the sum over all orientations to yield

$$\hat{E}(\mathbf{x}; \theta_i, \sigma_j) = \frac{E(\mathbf{x}; \theta_i, \sigma_j)}{\sum_{m=1}^{M} E(\mathbf{x}; \theta_m, \sigma_j) + \epsilon}. \tag{4}$$

Here, ϵ serves as a noise floor and to reduce numerical instabilities. It is specified as the standard deviation of the energy measurements across all orientations. Once again, it is interesting to note that biological modeling of visual processing have employed a similar divisive normalization, including use of a signal adaptive saturation constant [13]. Finally, note that features captured at this layer, $\hat{E}(\mathbf{x}; \theta_i, \sigma_j)$, are measures of Spatiotemporal Oriented Energy [37, 5]; therefore, the overall network is named **SOE-Net**.

2.5. Pooling

Two pooling mechanisms are employed to achieve the desired level of abstraction. First, spatiotemporal pooling is performed following a frequency decreasing path. Thus, the same set of filters in the convolution block operate on lower spatiotemporal frequencies at each subsequent layer thereby revealing new information from the originally input signal. Second, the feature maps extracted for each layer are linearly combined, through cross-channel pooling, to capture more complex structures at each layer while preventing the number of feature maps from exponential increase.

Spatiotemporal pooling. Spatiotemporal pooling serves to aggregate normalized responses, (4), over spacetime. Aggregation provides for a degree of shift invariance in its output, as the exact position of a response in the pooling region is abstracted. In SOE-Net, spacetime pooling is implemented via low-pass filtering with a 3D Gaussian, $G_{3D}(\gamma)$, followed by downsampling, $\downarrow_\tau (\cdot)$, where γ is the standard deviation and τ is the sampling period,

$$\boldsymbol{S}_k(\mathbf{x}; \theta_i, \sigma_j) = \downarrow_\tau \left(G_{3D}(\gamma) * \hat{E}(\mathbf{x}; \theta_i, \sigma_j) \right). \tag{5}$$

Here, a key question is how to specify the pooling parameters, γ and τ? Typical ConvNets rely on heuristic choices, as their learning based approach does not yield enough theoretical insight for a formal answer. In contrast, since the signal properties of the present architecture have been specified precisely, these parameters can be specified analytically. First, applying a 3^{rd}- order Gaussian derivative filter, $G_{3D}^{(3)}$, with zero mean and standard deviation, σ, in the convolution stage greatly attenuates frequencies $\omega_c > \frac{3\sqrt{3}}{\sigma}$ (*i.e.* a factor of ≈ 3 beyond the central frequency). However, given the frequency doubling effect from the squaring in the rectification stage, consideration must instead be given to $\eta = 2\omega_c$ as the effective cut-off frequency. Correspondingly, it is appropriate to select the low-pass filter of the pooling stage with a cut-off frequency $\omega_l = \alpha\eta$, with $0 < \alpha < 1$ to ensure operations on lower spatiotemporal

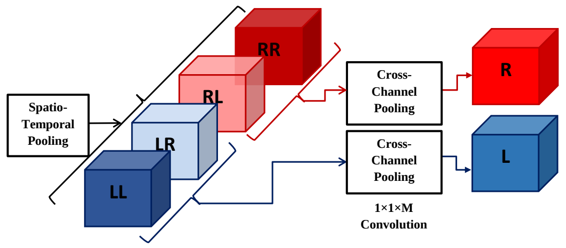

Figure 4. Overview of the Proposed Cross-Channel Pooling.

frequencies at each layer. This implies taking $\gamma = \frac{3}{\alpha\eta}$ (*i.e.* approximating the cut off frequency to be ≈ 3 standard deviations away from the central frequency).

To avoid aliasing in downsampling, the sampling theorem is used such that $\omega_s > 2\alpha\eta$. Correspondingly, the sampling period is $\tau = \beta\frac{2\pi}{\omega_s}, 0 < \beta < 1$. In implementation, a conservative $\beta = 0.5$ is employed. Notably, use of low-pass filtering with decreasing frequencies guarantees an energy decay from layer $\mathcal{L}_k$ to layer $\mathcal{L}_{k+1}$.

Cross-channel pooling. Cross-channel pooling serves to aggregate pooled responses, (5), across feature maps. Figure 3 unfolds the recurrence to illustrate how once all features maps from layer $\mathcal{L}_k$ have gone separately through the recurrence, the number of feature maps, $\boldsymbol{S}_k$, for each scale, σ_j, used in the convolution block is M^k. This situation is unsatisfactory for two reasons. First, it fails to capture the emergence of common attributes across feature maps that result from applying the same filter orientation, θ_i, to the inputs from $\mathcal{L}_{k-1}$. Second, there is potential for explosion of the representation's size as many layers are cascaded. Both of these concerns can be dealt with by pooling across all feature maps from $\mathcal{L}_k$ that result from filtering with a common orientation, as illustrated in Fig. 4.

To formalize pooling across feature maps, recall that beyond the very first layer, each $\boldsymbol{S}_k$ derives from input that was itself parameterized by an orientation, θ_m^{k-1}, from filtering at the previous layer, $\mathcal{L}_{k-1}$. This dependence is now captured explicitly by extending its parameterization to $\boldsymbol{S}_k(\mathbf{x};\theta_i,\sigma_j,\theta_m^{k-1})$. This extension allows the desired cross-channel pooling to produce the final output of $\mathcal{L}_k$ as

$$\boldsymbol{L}_k(\mathbf{x};\theta_i,\sigma_j) = \frac{1}{M}\sum_{m=1}^{M}\boldsymbol{S}_k(\mathbf{x};\theta_i,\sigma_j,\theta_m^{k-1}). \tag{6}$$

In implementation this operation is realized as a $1\times 1\times M$ convolution with an averaging kernel. Note that the summation holds vacuously at the very first layer of processing.

Theoretically, (6) can be seen as an oriented energy generalization of the construction for derivatives of multivalued images *c.f.* [6, 33, 31], but is novel in the current context of oriented energy processing. In particular, given a multivalued function $\boldsymbol{f}(\mathbf{x}) = [f_1(\mathbf{x}),...,f_M(\mathbf{x})]^T$ with $\mathbf{x} = (x_1,...,x_N)^T$ and defining $\mathsf{G} = \sum_{m=1}^{M}\nabla f_m(\mathbf{x})\nabla^T f_m(\mathbf{x})$, a measure of change along $\hat{\mathbf{v}}$ is given as $\hat{\mathbf{v}}^T\mathsf{G}\hat{\mathbf{v}}$. Note that letting $\hat{\mathbf{v}} = \hat{\mathbf{e}}_i$, with $\hat{\mathbf{e}}_i$ the unit vector along x_i, yields $\hat{\mathbf{e}}_i^T\mathsf{G}\hat{\mathbf{e}}_i = \sum_{m=1}^{M}(\frac{\partial f_m}{\partial x_i})^2$. Comparatively, in (6) the multivalued function is just the set of feature maps obtained from a previous layer, $\mathcal{L}_{k-1}$, that has been differentially filtered and squared in the convolution and rectification blocks, resp. Thus, $\boldsymbol{L}_k(\mathbf{x};\theta_i,\sigma_j)$ captures the energy along direction θ_i taken across the multivalued $\boldsymbol{S}_k(\mathbf{x};\theta_i,\sigma_j,\theta_m^{k-1})$.

Conceptually, the cross-channel pooling, (6), produces a set of M feature maps (one for each filter orientation, θ) that capture the amount of structure present in the previous layer at that orientation, irrespective of the source (*i.e.* irrespective of a particular feature map at the previous layer). This approach thereby yields immediate insight into the nature of the representation, in contrast to alternative approaches that rely on random cross-channel combinations, *e.g.* [20]. Reflecting back on the motivating examples, Fig. 2, the proposed processing was used to generate the feature maps, yielding exactly the desired abstractions.

2.6. Dynamic texture recognition

The SOE-Net representation, $\boldsymbol{F}(\mathbf{x};\theta_i,\sigma_j)$, extracted after the normalization block of the K^{th} layer, provides a rich hierarchical description of visual spacetime in terms of multiscale spatiotemporal orientation. As such, it has potential to serve as the representational substrate for a wide variety of spacetime image understanding tasks (*e.g.* segmentation, tracking and recognition). As an illustrative example, dynamic texture (DT) recognition is considered. This task has been considered for two main reasons. First, DTs are pervasive phenomena in the visual world. Second, the ability to recognize these patterns provides the basis for a number of further applications, including video indexing, scene recognition and change detection in surveillance and monitoring.

For the specific application of dynamic texture recognition, each spacetime volume, $V(\mathbf{x})$, to be recognized is taken to contain a single pattern. Therefore, the pointwise extracted feature maps, $\boldsymbol{F}(\mathbf{x};\theta_i,\sigma_j)$, can be aggregated over a region, Ω, that covers the entire spacetime texture sample to be classified to yield a global feature vector,

$$\mathcal{F}(\theta_i,\sigma_j) = \sum_{\mathbf{x}\in\Omega}\boldsymbol{F}(\mathbf{x};\theta_i,\sigma_j). \tag{7}$$

The resulting global feature vector is l_2 normalized to yield the overall descriptor $\hat{\mathcal{F}}(\theta_i,\sigma_j)$. To compare the feature distributions of an input query to database entries, the Bhattacharyya coefficient is used, as it provides a principled way to compare two normalized distributions kept as histograms.

While previous work made use of spatiotemporal oriented energies for dynamic textures [5], it differs from the current work in five significant ways. First and foremost, previous work did not use repeated filtering, (1), for hierarchical abstraction of image structure. Second, it marginalized appearance information so that it did not capture purely spatial structure and therefore was less able than the current approach to capitalize on both appearance and dynamics.

Third, it did not separate the opposite phase responses in rectification, (3). Fourth, it did not employ multiple scales, σ, (2). Fifth, no cross-channel pooling was involved in previous work, while it has a key role in the current work, (6).

2.7. Implementation details

Convolution with the Gaussian 3^{rd} derivatives, (2), is realized with separable 13-tap filters, with $M = 10$ orientations/scale and $\sigma = 1$. Multiscale filtering is realized by applying the same filters across levels of a Gaussian pyramid, with factor of 2 subsampling between levels. The number of scales, $|\sigma|$, is chosen automatically to avoid undue border effects depending on the size of the input spacetime volume: The coarsest scale is the last such that the filter can fit entirely within the corresponding pyramid level. Unless otherwise noted, this constraint yields $|\sigma| = 2$ for all datasets, except Dyntex++ where $|\sigma| = 1$ due to the very small size of its videos $(50 \times 50 \times 50)$. Similarly, the number of iterations over the network, (1), *i.e.* the number of layers, K, is stopped when the spatiotemporal support of the signals to be fed back through the recurrence is less than or equal to the filter size; see Sec. 3 for specifics by dataset. Due to the cross-channel pooling, (6), the output of the convolution block starting from layer 2 is always $M^2 \times |\sigma|$ feature maps. Also, because the proposed rectification strategy, (3), splits the results of each convolutional block into 2 paths, the effective number of feature maps after the rectification block is doubled at each layer. Therefore, the dimension, D_K, of feature vectors extracted after the normalization block of the K^{th} layer is $D_K = M^2 \times |\sigma| \times 2^K$.

3. Empirical evaluation

SOE-Net is evaluated according to standard protocols on the two most recent dynamic texture datasets, YUVL [5] and Dyntex [26]. YUVL is the larger with 610 sequences grouped into 5 classes (YUVL1) or by further subdividing 3 of the initial classes into 6 subclasses (YUVL2), but only using 509 sequences from the full set; see [5]. Here, a third organization is introduced (YUVL3), where 2 classes neglected in YUVL2 are reinstated to use the entire set divided into 8 classes. Dyntex is used in 5 main variations: Dyntex-35 has 35 classes with 10 sequences/class [41]; Dyntex++ has 36 classes with 100 sequences/class [11]; Alpha, Beta and Gamma have 60, 162 and 275 sequences, divided into 3, 10 and 10 classes, resp [7]. The same protocol is used for all datasets. Feature vectors from the last layer of SOE-Net are input to a Nearest-Neighbor (NN) classifier using the leave-one-out procedure [32, 5, 27]. Although NN is not state-of-the-art as a classifier, it is appropriate here where the goal is to highlight the discriminative power of the features without obscuring performance via use of more sophisticated classifiers. Still, for completeness and fair comparison to previous work, results also are reported using Nearest Class Center (NCC) and SVM classifiers in Sec. 3.3.

3.1. Component-wise validation

SOE-Net's theory based component specifications are now validated empirically. Primarily, classification accuracy on YUVL is used for this goal, as it is the largest available and is well organized according to pattern dynamics.

Convolution. The theoretical basis for use of multiscale, spatiotemporally oriented filters, in general, and Gaussian derivative filters, in particular, was given in Sec. 2.2. Still, a choice remains regarding the order of the derivative used. Table 1 compares 2^{nd}-, 3^{rd}- and 4^{th}-order Gaussian derivatives, in terms of classification accuracy, while using a single layer and scale of SOE-Net. The results show $G3$ as the best performer. This choice provides the right balance between tuning specificity and the numerical stability of relatively low-order filtering. In light of these observations, all subsequent results rely on 3^{rd}-order Gaussian derivatives.

	YUVL1			YUVL2			YUVL3		
	G2	G3	G4	G2	G3	G4	G2	G3	G4
SOE-Net (3D)	83.3	**91.1**	89.8	85.1	**90.2**	90.1	74.1	**84.6**	82.9

Table 1. Comparison of 2^{nd}, 3^{rd}, 4^{th} order Gaussian derivatives.

Also, one could question the benefit of 3D filtering that captures temporal information in recognizing dynamic textures over 2D filtering that captures only spatial appearance. Thus, a 2D version of SOE-Net is used as a baseline comparison. Further, a 2D state-of-the-art hand crafted network, originally proposed for 2D texture analysis, ScatNet [3] is compared. In both cases, 2D frames are supplied to the 2D networks. Table 2 shows the decided advantage of 3D over 2D filtering for dynamic texture recognition.

	YUVL1	YUVL2	YUVL3
ScatNet (2D) [3]	68.7	69.7	64.8
SOE-NET (2D)	64.0	70.1	60.8
SOE-Net (3D)	**95.6**	**91.7**	**91.0**

Table 2. Benefits of 3D vs. 2D filtering.

Multiple layers and scales. Table 3 documents the benefits of multiscale filtering, (2), at each level of the proposed network as well as those of multiple layers, (1). The results show that addition of multiple layers and scales consistently improve classification accuracy, with an increase ranging from $\sim 2\%$ to $\sim 6\%$ in going from a single layer and scale to multiple layers and scales. Significantly, the combined multiscale, multilayer results also outperform the best results previously presented on this dataset of 94% and 90% for YUVL1 and YUVL2, respectively [5]. (There are no previously reported results on YUVL3.) These results highlight the pivotal role of the recurrent connection in the proposed network that allows it to decompose the input signal to reveal novel information across scales and layers.

Notably, the network instantiation applied to YUVL is limited to 2 layers and scales due to the small spatiotemporal extent of some sequences in the dataset; see Sec. 2.7 for how dataset extent determines layers and scales. Based on these results, all SOE-Net results presented in the remainder of this paper are based on feature vectors formed through

	YUVL1			YUVL2			YUVL3		
	Scale 1	Scale 2	[Scale 1, Scale 2]	Scale 1	Scale 2	[Scale 1, Scale 2]	Scale 1	Scale 2	[Scale 1, Scale 2]
Layer 1	91.1	87.5	92.9	90.2	85.3	90.9	84.6	80.8	86.2
Layer 2	94.3	91.9	**95.6**	89.0	88.2	**91.7**	86.7	86.9	**91.0**
[5]	-	-	94.0	-	-	90.0	-	-	-

Table 3. Classification accuracy on the YUVL dataset using SOE-Net with multiple layers and scales.

the concatenation of the last layer's output at all scales.

Two path rectification. Next, the advantage of the proposed two path rectification, (3), is evaluated. Comparison is made to two alternative rectification approaches: 1) full wave rectification, where the positive and negative signals are combined as its input signal is simply pointwise squared; 2) ReLU rectification, where only the positive part of its input signal is retained. Table 4 shows the benefit of the proposed rectification approach. It is seen that not only is there value of not neglecting one signal component (*i.e.* full-wave outperforms ReLU), but moreover additional benefit comes from keeping the positive and negative components separate, which suggest their complementarity.

	YUVL1	YUVL2	YUVL3
Full Wave Rectification	94.2	90.3	89.3
Positive Path (ReLU)	94.2	89.4	88.4
Two Paths	**95.6**	**91.7**	**91.0**

Table 4. Benefits of the proposed two path rectification approach.

Normalization. Normalization, (4), serves to increase invariance to contrast changes. To document this advantage, an instantiation of SOE-Net without the normalization block is compared. The results in Table 5 clearly demonstrate the usefulness of this step.

	YUVL1	YUVL2	YUVL3
No Normalization	90.6	87.2	82.8
With Normalization	**95.6**	**91.7**	**91.0**

Table 5. Benefits of the normalization step.

Spatiotemporal pooling. The theoretical basis for use of a Gaussian filter in spatiotemporal pooling was given in Sec. 2.5. Here, this choice is validated through comparison to two alternative pooling approaches [15]: 1) the Gaussian filter is replaced with a simple boxcar filter; 2) the more widely used max pooling is considered. Table 6 shows the benefit of the proposed spatiotemporal pooling approach that outperforms other pooling strategies by at least 4%.

	YUVL1	YUVL2	YUVL3
boxcar filter	91.7	87.3	87.0
max pooling	91.6	87.4	85.7
Gaussian filter	**95.6**	**91.7**	**91.0**

Table 6. Benefits of the used spatiotemporal pooling approach.

Cross-channel pooling. SOE-Net's novel cross-channel (CC) pooling plays a pivotal role in keeping the dimensionality manageable. Table 7 compares the size of the feature vectors with and without CC-pooling for $|\sigma| = 1$. Note that dimensionality reduction does not occur until layer 3, as final feature vector output occurs prior to where CC-pooling would apply and is vacuous at level 1; see Sec. 2.5. To examine the impact of such striking dimensionality reduction on classification, SOE-Net is compared with and without CC-pooling. Here, a dataset with spatiotemporal size big enough to support 3 layers is needed, as it is the point where dimensionality reduction becomes apparent. Also, comparing beyond layer 3 is not computationally feasible due to dimensionality explosion without CC-pooling, even though it is reasonable with CC-pooling, as done in Secs. 3.2 and 3.3. Since the spatial dimensions of YUVL only support up to layer 2, two variations of Dyntex (Beta and Dyntex_35) are used here. Significantly, on both datasets accuracy of SOE-Net with vs. without CC-pooling is comparable, *i.e.* 97.7% vs. 96.8% on Dyntex_35 and 95.7% vs. 95.1% on Beta, although the feature vectors extracted using CC-pooling are an order of magnitude smaller. These results show that CC-pooling keeps network size manageable while maintaining high discriminating power.

	L1	L2	L3	L4	L5
SOE-Net (a)	20	400	8000	160000	3200000
SOE-Net (b)	**20**	**400**	**800**	**1600**	**3200**

Table 7. Feature dimensions (a) without vs. (b) with CC-pooling.

3.2. Comparison to a learned 3D ConvNet

It is interesting to compare the performance of SOE-Net to a learning based 3D ConvNet. For this comparison C3D [36] is used for 3 main reasons. 1) Currently, C3D is the only ConvNet trained end-to-end using 3D filters only without any pre-processing of the input volumes (*e.g.* to extract optical flow). This architecture makes it the most similar trained network to SOE-Net that also relies on 3D convolutions on the raw input volumes. 2) This pre-trained network has shown state-of-the-art performance, without any fine tuning, on a variety of tasks, including action recognition, object recognition and dynamic scene classification, which is very similar to dynamic texture recognition. Indeed, C3D is advocated as a general feature extractor for video analysis tasks without fine tuning (see Sec. 3.3 in [36]). 3) It is not feasible to fine tune a network with any of the available DT datasets due to their very limited size. So, alternative ConvNets that require such data for fine tuning prior to testing cannot be compared in a meaningful fashion.

C3D features are extracted (FC-6 activations as specified in [36]) for all versions of the YUVL and Dyntex, except Dyntex++, which is discarded as the small size of its videos ($50 \times 50 \times 50$) precludes application of C3D. For experiments with the considered Dyntex versions, the relatively large size of the sequences makes it possible to push SOE-Net to 3 layers on Dyntex_35 and to 5 layers on Alpha, Beta, Gamma. The results in Table 8 show that SOE-Net outperformed C3D on the majority of cases (5 of 7). The larger performance gaps between SOE-Net and C3D on the YUVL dataset are of particular note, especially given that SOE-Net uses only 2 layers on YUVL.

	YUVL1	YUVL2	YUVL3	Alpha	Beta	Gamma	Dyntex_35
C3D [36]	88.0	89.8	85.5	100	96.3	95.0	96.3
SOE-Net	**95.6**	**91.7**	**91.0**	98.3	**96.9**	93.6	**97.7**

Table 8. Comparison of SOE-Net features versus C3D features.

Method		Alpha		Beta		Gamma		Dyntex_35		Dyntex++
		SVM	NCC	SVM	NCC	SVM	NCC	NN	NCC	SVM
Learning-based	[11]	-	-	-	-	-	-	-	-	63.7
	[23]	-	-		-	-	-	98.6	-	-
	[12]	-	-		-	-	-	-	-	92.8
	[27]	87.8	86.6	76.7	69.0	74.8	64.2	**99.0**	**97.8**	**94.7**
Hand-crafted	[39]	84.9	83.6	76.5	65.2	74.5	60.8	-	97.6	89.9
	[38]	82.8	-	75.4	-	73.5	-	-	96.7	89.2
	[17]	-	-	-	-	-	-	-	96.5	88.8
	[42]	83.3	-	73.4	-	72.0	-	-	97.1	89.8
	[7]	-	85.0	-	67.0	-	63.0	-	-	-
	SOE-NET	**96.7**	**96.7**	**95.7**	**86.4**	**92.2**	**80.3**	97.7	93.1	94.4

Table 9. Comparison to state-of-the-art methods on Dynamic Texture recognition.

Significantly, design of YUVL was motivated by interest in building a true *dynamic* texture dataset. So, it groups patterns based on their dynamics, rather than their appearance. Thus, relative performance on YUVL suggests that SOE-Net is more able to capitalize on dynamic information than C3D. Further, SOE-Net either outperforms or is on par with C3D on datasets that group dynamic textures with more emphasis on visual appearance, *i.e*., Dyntex, suggesting that SOE-Net is able to capitalize on appearance information as well. Overall, these results show that SOE-Net can capture rich, discriminative information, be it dynamic or static appearance, without reliance on extensive and costly training.

3.3. Dynamic texture recognition state-of-the-art

Comparison now is made to state-of-the-art approaches designed for dynamic texture recognition using the currently most widely used DT datasets (*i.e*. various breakdowns of Dyntex, as YUVL has received notably less attention). In particular, SOE-Net is compared to 4 learning based [11, 23, 12, 27] and 5 hand-crafted [39, 38, 42, 17, 7] DT descriptors. Following previous research using Dyntex [39, 38, 42, 17, 7, 27], evaluation is performed variously using SVM, Nearest Class Center (NCC) and Nearest Neighbor (NN) classifiers. For SVM, the same protocol used in previous research is followed, whereby 50% of samples per category are used for training and the rest for testing.

Table 9 shows that SOE-Net outperforms all other approaches on the Alpha, Beta and Gamma datasets, with sizable performance gaps between $\approx 9 - 19\%$ using SVM . Using NCC, SOE-Net outperforms all other methods by at least 10%. Overall, these results speak decisively for the high discriminative power of SOE-Net's descriptors.

Similarly, on Dyntex++ SOE-Net extracts stronger features than all hand-crafted methods, achieving an accuracy that is 4.5% higher than previous state-of-the-art approach [39]. Also, under SVM, SOE-Net is on par with state-of-the-art learning based method, with only marginal difference ($\approx 0.3\%$). Moreover, using a NN classifier SOE-Net outperforms all other methods, with an accuracy of 95.6% (results not shown in table, as not reported by others). These results again confirm the discriminative power of the proposed representation. Notably, due to the extremely small size of the Dyntex++ videos, SOE-Net was restricted to only a single scale and 2 layers on this particular dataset.

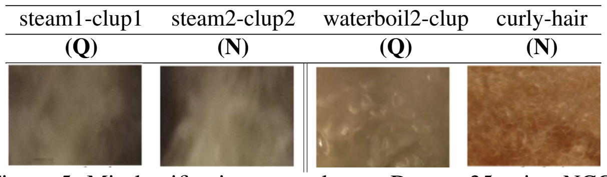

Figure 5. Misclassification examples on Dyntex_35 using NCC. **(Q)** is the query class. **(N)** is the nearest class in the database.

Finally, on Dyntex_35 SOE-Net performs slightly worse than state-of-the-art using the NN classifier (a difference of ≈1.3%); however, the difference is greater using the NCC classifier. Interestingly, closer examination of these results reveals that confusions typically occur between slightly different views of the same physical process, which naturally yield visually very similar dynamic textures. Other confusions arise from different physical processes, which happen to yield similar visual appearances. Figure 5 shows examples. Overall, the fact that these are the types of confusions made by the network suggests an ability to generalize across viewpoint, which arguably is more important than ability to make fine grained distinctions between viewpoints during texture classification as well as other recognition tasks.

4. Conclusion

This paper presented a novel hierarchical spatiotemporal network based on three key ideas. First, a multilayer repeated filtering architecture is employed. Second, design decisions have been theoretically motivated without relying on learning or other empirically driven decisions. Third, in addition to adding insight into convolution, rectification, normalization and spatiotemporal pooling, a novel cross-channel pooling has been introduced that keeps the representation compact while maintaining representational clarity. The repeated filtering architecture and theory driven design makes the representation understandable in terms of multiorientation, multiscale properties. Further, by eschewing learning, the approach does not rely on training data. Finally, the benefits of SOE-Net were shown on dynamic texture recognition, where it extends the state-of-the-art.

References

[1] E. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, 1991. 2.2

[2] C. L. Baker and I. Mareschal. Processing of second-order stimuli in the visual cortex. *Progress in Brain Research*, 134:171–91, 2001. 2.1

[3] J. Bruna and S. Mallat. Invariant scattering convolution networks. *PAMI*, 35:1872–1886, 2013. 1, 3.1

[4] T. H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. PCANet: A simple deep learning baseline for image classification? *TIP*, 24:5017–5032, 2015. 1

[5] K. Derpanis and R. Wildes. Spacetime texture representation and recognition based on spatiotemporal orientation analysis. *PAMI*, 34:1193–1205, 2012. 1, 2.2, 2.4, 2.6, 3, 3.1, 3.1

[6] S. DiZenzo. A note on the gradient of a multi-image. *CVGIP*, 33:116–125, 1986. 2.5

[7] S. Dubois, R. Peteri, and M. Michel. Characterization and recognition of dynamic textures based on the 2D+T curvelet. *Sig. Im. & Vid. Proc.*, 9:819–830, 2013. 1, 3, 3.2, 3.3

[8] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common muli-scale convolutional architecture. In *ICCV*, 2015. 1

[9] J. Feng and T. Darrell. Learning the structure of deep convolutional networks. In *ICCV*, 2015. 1

[10] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *PAMI*, 13:891–906, 1991. 2.2, 2.2

[11] B. Ghanem and A. Narendra. Max margin distance learning for dynamic texture. In *ECCV*, 2010. 1, 3, 3.2, 3.3

[12] M. Harandi, C. Sanderson, C. Shen, and B. Lovell. Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution. In *ICCV*, 2013. 1, 3.2, 3.3

[13] D. J. Heeger. Nonlinear model of neural responses in cat visual cortex. In M. Landy and J. Movshon, editors, *Computational Models of Visual Processing*, chapter 9, pages 119–134. MIT Press, Cambridge, 1991. 2.3, 2.4

[14] J. H. Jacobsen, J. V. Gemert, Z. Lou, and A. W. Smeulders. Structured receptive fields in CNNs. In *CVPR*, 2016. 1

[15] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009. 3.1

[16] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007. 1

[17] H. Ji, X. Yang, H. Ling, and Y. Xu. Wavelet domain multifractal analysis for static and dynamic texture classification. *TIP*, 22:286–299, 2013. 1, 3.2, 3.3

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86:2278–2324, 1998. 1

[20] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014. 2.5

[21] Z. Lu, W. Xie, J. Pei, and J. Huang. Dynamic texture recognition by spatio-temporal multiresolution histograms. In *WACV*, 2005. 1

[22] S. Lyu and E. P. Simoncelli. Nonlinear image representation using divisive normalization. In *CVPR*, pages 1–8, 2008. 2.4

[23] A. Mumtaz, E. Coviello, G. Lanckriet, and A. B. Chan. Clustering dynamic textures with the hierarchical EM algorithm for modeling video. *PAMI*, 35:1606–1621, 2013. 1, 3.2, 3.3

[24] J. Mutch and D. G. Lowe. Multiclass object recognition with sparse, localized features. In *CVPR*, 2006. 1

[25] R. Peteri and D. Chetverikov. Dynamic texture recognition using normal flow and texture regularity. In *IbPRIA*, 2005. 1

[26] R. Peteri, F. Sandor, and M. Huiskes. DynTex: A comprehensive database of dynamic textures. *PRL*, 31:1627–1632, 2010. 3

[27] Y. Quan, Y. Huang, and H. Ji. Dynamic texture recognition via orthogonal tensor dictionary learning. In *ICCV*, 2015. 1, 3, 3.2, 3.3

[28] J. Ren, X. Jiang, and J. Yuan. Dynamic texture recognition using enhanced lbp features. In *ICASSP*, 2013. 1

[29] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nat. Neuro.*, 2:1019–1025, 1999. 1

[30] A. Rodriguez-Sanchez, M. Fallah, and A. Leonardis. Hierarchical object representation in the visual cortex and computer vision. *Frontiers in Comp. Neuro.*, 9, 2015. 1

[31] Y. Rubner and C. Tomasi. Coallescing texture descriptors. In *Proceedings of the ARPA IUW*, 1996. 2.5

[32] P. Saisan, G. Doretto, Y. Wu, and S. Soatto. Dynamic texture recognition. In *CVPR*, 2001. 1, 3

[33] G. Sapiro and D. Ringach. Anisotropic diffusion of multivalued images with applications to color filtering. *TIP*, 5:1582–1586, 1996. 2.5

[34] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2016. 2.3

[35] M. Szummer and R. W. Picard. Temporal texture modeling. In *ICIP*, 1996. 1

[36] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 1, 3.2, 3.2

[37] R. P. Wildes and J. R. Bergen. Qualitative spatiotemporal analysis using an oriented energy representation. In *ECCV*, 2000. 2.2, 2.3, 2.4

[38] Y. Xu, S. Huang, H. Ji, and C. Fermuller. Scale-space texture description on sift-like textons. *CVIU*, 116:999 – 1013, 2012. 1, 3.2, 3.3

[39] Y. Xu, Y. Quan, H. Ling, and H. Ji. Dynamic texture classification from fractal analysis. In *ICCV*, 2011. 1, 3.2, 3.3

[40] F. Yang, G. Xia, G. Liu, L. Zhang, and X. Huang. Dynamic texture recognition by aggregating spatial and temporal features via SVMs. *Neurocomp.*, 173:1310 – 1321, 2016. 1

[41] G. Zhao and M. Pietikainen. Dynamic texture recognition using volume local binary patterns. In *ECCV*, 2006. 3

[42] G. Zhao and M. Pietikainen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *PAMI*, 29:915–928, 2007. 1, 3.2, 3.3

[43] C. Zzegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *NIPS*, 2013. 1