Conquering the CNN Over-Parameterization Dilemma: A Volterra Filtering Approach for Action Recognition

Siddharth Roheda, Hamid Krim
North Carolina State University at Raleigh, NC, USA
{sroheda, ahk}@ncsu.edu

Abstract

The importance of inference in Machine Learning (ML) has led to an explosive number of different proposals in ML, and particularly in Deep Learning. In an attempt to reduce the complexity of Convolutional Neural Networks, we propose a Volterra filter-inspired Network architecture. This architecture introduces controlled non-linearities in the form of interactions between the delayed input samples of data. We propose a cascaded implementation of Volterra Filtering so as to significantly reduce the number of parameters required to carry out the same classification task as that of a conventional Neural Network. We demonstrate an efficient parallel implementation of this Volterra Neural Network (VNN), along with its remarkable performance while retaining a relatively simpler and potentially more tractable structure. Furthermore, we show a rather sophisticated adaptation of this network to non-linearly fuse the RGB (spatial) information and the Optical Flow (temporal) information of a video sequence for action recognition. The proposed approach is evaluated on UCF-101 and HMDB-51 datasets for action recognition, and is shown to outperform state of the art CNN approaches.

1 Introduction

Human action recognition is an important research topic in Computer Vision, and can be used towards surveillance, video retrieval, and man-machine interaction to name a few. The survey on Action Recognition approaches (Kong and Fu 2018) provides a good progress overview. Video classification usually involves three stages (Wang et al. 2009; Liu, Luo, and Shah 2009; Niebles, Chen, and Fei-Fei 2010; Sivic and Zisserman 2003; Karpathy et al. 2014), namely, visual feature extraction (local features like Histograms of Oriented Gradients (HoG) (Dalal and Triggs 2005), or global features like Hue, Saturation, etc.), feature fusion/concatenation, and lastly classification. In (Yi, Krim, and Norris 2011), an intrinsic stochastic modeling of human activity on a shape manifold is proposed and an accurate analysis of the non-linear feature space of activity models is provided. The emergence of Convolutional Neural Networks (CNNs) (LeCun et al. 1998), along with the availability of large training datasets and computational resources have come a long way to obtaining the various steps by a single neural network. This approach has led to remarkable progress in action recognition in video sequences, as well as in other vision applications like object detection (Sermanet et al. 2013), scene labeling (Farabet et al. 2012), image generation (Goodfellow et al. 2014), image translation (Isola et al. 2017), information distillation (Roheda et al. 2018b; Hoffman, Gupta, and Darrell 2016), etc. In the Action Recognition domain, datasets like the UCF-101 (Soomro, Zamir, and Shah 2012), Kinetics (Kay et al. 2017), HMDB-51 (Kuehne et al. 2011), and Sports-1M (Karpathy et al. 2014) have served as benchmarks for evaluating various solution performances. In action recognition applications the proposed CNN solutions generally align along two themes: *1. One Stream CNN* (only use either spatial or temporal information), *2. Two Stream CNN* (integrate both spatial and temporal information). Many implementations (Carreira and Zisserman 2017; Diba, Sharma, and Van Gool 2017; Feichtenhofer, Pinz, and Zisserman 2016; Simonyan and Zisserman 2014) have shown that integrating both streams leads to a significant boost in recognition performance. In Deep Temporal Linear Encoding (Diba, Sharma, and Van Gool 2017), the authors propose to use 2D CNNs (pre-trained on ImageNet (Deng et al. 2009)) to extract features from RGB frames (spatial information) and the associated optical flow (temporal information). The video is first divided into smaller segments for feature extraction via 2D CNNs. The extracted features are subsequently combined into a single feature map via a bilinear model. This approach, when using both streams, is shown to achieve a *95.6 %* accuracy on the UCF-101 dataset, while only achieving *86.3 %* when only relying on the RGB stream. Carreira et al. (Carreira and Zisserman 2017) adopt the GoogLeNet architecture which was developed for image classification in ImageNet (Deng et al. 2009), and use 3D convolutions (instead of 2D ones) to classify videos. This implementation is referred to as the Inflated 3D CNN (I3D), and is shown to achieve a performance of *88.8 %* on UCF-101 when trained from scratch, while achieving a *98.0 %* accuracy when a larger dataset (Kinetics) was used for pre-training the entire network (except for the classification layer). While these

approaches achieve near perfect classification, the models are extremely heavy to train, and have a tremendous number of parameters (25M in I3D, 22.6M in Deep Temporal Linear Encoding). This in addition, makes the analysis, including the necessary degree of non-linearity, difficult to understand, and the tractability elusive. In this paper we explore the idea of introducing controlled non-linearities through interactions between delayed samples of a time series. We will build on the formulations of the widely known Volterra Series (Volterra 2005) to accomplish this task. While prior attempts to introduce non-linearity based on the Volterra Filter have been proposed (Kumar et al. 2011; Zoumpourlis et al. 2017) , most have limited the development up to a quadratic form on account of the explosive number of parameters required to learn higher order complexity structure. While quadratic non-linearity is sufficient for some applications (eg. system identification), it is highly inadequate to capture all the non-linear information present in videos.

Contributions: In this paper, we propose a Volterra Filter (Volterra 2005) based architecture where the non-linearities are introduced via the system response functions and hence by controlled interactions between delayed frames of the video. The overall model is updated on the basis of a cross-entropy loss of the labels resulting from a linear classifier of the generated features. An efficiently cascaded implementation of a Volterra Filter is used in order to explore higher order terms while avoiding over-parameterization. The Volterra filter principle is also exploited to combine the RGB and the Optical Flow streams for action recognition, hence yielding a performance driven non-linear fusion of the two streams. We further show that the number of parameters required to realize such a model is significantly lower in comparison to a conventional CNN, hence leading to faster training and significant reduction of the required resources to learn, store, or implement such a model.

2 Background and Related Work

2.1 Volterra Series

The physical notion of a system is that of a black box with an input/output relationship y_t/x_t. If a non-linear system is time invariant, the relation between the output and input can be expressed in the following form (Volterra 2005; Schetzen 1980),

$$y_t = \sum_{\tau_1=0}^{L-1} \boldsymbol{W}^{\boldsymbol{1}}_{\tau_1} x_{t-\tau_1} + \sum_{\tau_1,\tau_2=0}^{L-1} \boldsymbol{W}^{\boldsymbol{2}}_{\tau_1,\tau_2} x_{t-\tau_1} x_{t-\tau_2} + ... + \sum_{\tau_1,\tau_2,...,\tau_K=0}^{L-1} \boldsymbol{W}^{\boldsymbol{K}}_{\tau_1,\tau_2,...,\tau_K} x_{t-\tau_1} x_{t-\tau_2} ... x_{t-\tau_K}, \tag{1}$$

where L is the number of terms in the filter memory (also referred to as the filter length), $\boldsymbol{W}^{\boldsymbol{k}}$ are the weights for the k^{th} order term, and $\boldsymbol{W}^{\boldsymbol{k}}_{\tau_1,\tau_2,...,\tau_k} = 0$, for any $\tau_j < 0$, $j = 1, 2, ..., k$, $\forall k = 1, 2, ..., K$ due to causality. This functional form is due to the mathematician Vito Volterra (Volterra 2005), and is widely referred to as the Volterra Series. The calculation of the kernels is a computationally complex problem, and a K^{th} order filter of length L, entails solving L^K equations. The corresponding adaptive weights are a result of a target energy functional whose minimization iteratively adapts the filter taps as shown in Figure 1.

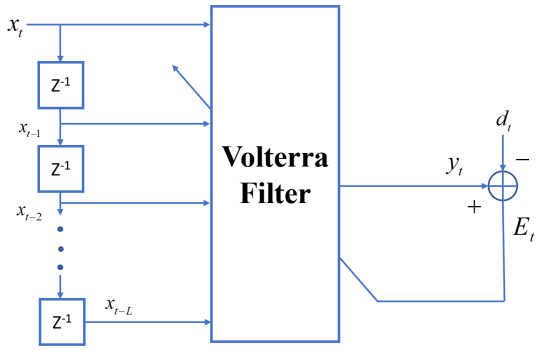

Figure 1: Adaptive Volterra Filter

It is worth observing from Equation 1 that the linear term is actually similar to a convolutional layer in CNNs. Non-linearities in CNNs are introduced only via activation functions, and not in the convolutional layer, while in Equation 1 we will introduce non-linearities via higher order convolutions.

2.2 Nested Volterra Filter

In (Osowski and Quang 1994), a nested reformulation of Equation 1 was used in order to construct a feedforward implementation of the Volterra Filter,

$$y_t = \sum_{\tau_1=0}^{L-1} x_{t-\tau_1} \left[\boldsymbol{W}^{\boldsymbol{1}}_{\tau_1} + \sum_{\tau_2=0}^{L-1} x_{t-\tau_2} \left[\boldsymbol{W}^{\boldsymbol{2}}_{\tau_1,\tau_2} + \sum_{\tau_3=0}^{L-1} x_{t-\tau_3} [\boldsymbol{W}^{\boldsymbol{3}}_{\tau_1,\tau_2,\tau_3} + ...] \right] \right]. \tag{2}$$

Each factor contained in the brackets can be interpreted as the output of a linear Finite Impulse Response (FIR) filter, thus allowing a layered representation of the Filter. A nested filter implementation with $L = 2$ and $K = 2$ is shown in Figure 2. The length of the filter is increased by adding modules in parallel, while the order is increased by additional layers. Much like any multi-layer network, the weights of the synthesized filter are updated layer after layer according to a backpropogation scheme. The nested structure of the Volterra Filter, yields much faster learning in comparison to that based on Equation 1. It, however, does not reduce the number of parameters to be learned, leading to potential over-parameterization when learning higher order filter approximations. Such a structure was used for a system identification problem in (Osowski and Quang 1994). The mean square error between the desired signal (d_t) and the output of the filter (y_t) was used as the cost functional to be minimized,

$$E_t = \frac{1}{2}(d_t - y_t)^2, \tag{3}$$

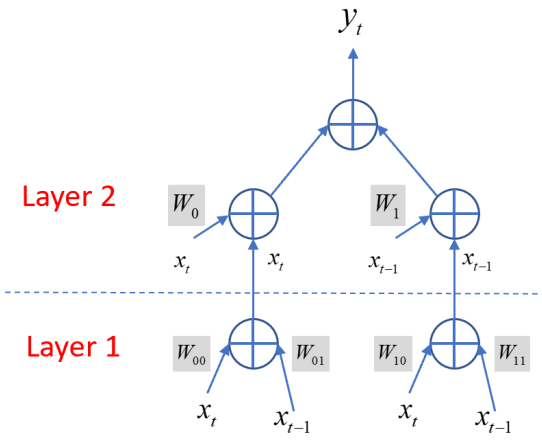

Figure 2: Nested Volterra Filter

and the weights for the k^{th} layer are updated per the following equations,

$$\boldsymbol{W}^{\boldsymbol{k}}_{\tau_1,\tau_2,...,\tau_k}(t+1) = \boldsymbol{W}^{\boldsymbol{k}}_{\tau_1,\tau_2,...,\tau_k}(t) - \eta \frac{\partial E_t}{\partial \boldsymbol{W}^{\boldsymbol{k}}_{\tau_1,\tau_2,...,\tau_k}}, \tag{4}$$

$$\frac{\partial E_t}{\partial \boldsymbol{W}^{\boldsymbol{k}}_{\tau_1,\tau_2,...,\tau_k}} = x_{t-\tau_k} x_{t-\tau_{k-1}} ... x_{t-\tau_1}(y_t - d_t). \tag{5}$$

2.3 Bilinear Convolution Neural Networks

There has been work on introducing 2^{nd} order non-linearities in the network by using a bi-linear operation on the features extracted by convolutional networks. Bilinear-CNNs (B-CNNs) were introduced in (Lin, RoyChowdhury, and Maji 2015) and were used for image classification. In B-CNNs, a bilinear mapping is applied to the final features extracted by linear convolutions leading to 2^{nd} order features which are not localized. As a result a feature extracted from the lower right corner of a frame in the B-CNN case, may interact with a feature from the upper right corner, and these two are not necessarily related (hence introducing erroneous additional characteristics), and this is in contrast to our proposed approach which highly controls such effects. Compact Bilinear Pooling was introduced in (Gao et al. 2016) where a bilinear approach to reduce feature dimensions was introduced. This was again performed after all the features had been extracted via linear convolutions and was limited to quadratic non-linearities. In our work we will explore non-linearities of much higher orders and also account for continuity of information between video frames for a given time period with the immediately preceding period. This effectively achieves a Recurrent Network-like property which accounts for a temporal relationship.

3 Problem Statement

Let a set of actions $\mathcal{A} = \{a_1, ..., a_I\}$, be of interest following a observed sequence of frames $\boldsymbol{X}_{T\times H\times W}$, where T is the total number of frames, and H and W are the height and width of a frame. At time t, the features $\boldsymbol{F}_t = g(\boldsymbol{X}_{[t-L+1:t]})$, will be used for classification of the sequence of frames $\boldsymbol{X}_{[t-L+1:t]}$ and mapped into one of the actions in $\mathcal{A}$, where L is the number of frames in the memory of the system/process. A linear classifier with weights $\boldsymbol{W}^{cl} = \{\boldsymbol{w}_i^{cl}\}_{i=1,...,I}$, and biases $\boldsymbol{b}^{cl} = \{b_i^{cl}\}_{i=1,...,I}$ will then be central to determining the classification scores for each action, followed by a soft-max function ($\rho(.)$) to convert the scores into a probability measure. The probability that the sequence of frames are associated to the i^{th} action class is hence the result of,

$$P_t(a_i) = \rho(\boldsymbol{w}_i^{cl^T}.\boldsymbol{F}_t + b_i^{cl}) = \frac{exp(\boldsymbol{w}_i^{cl^T}.\boldsymbol{F}_t + b_i^{cl})}{\sum_{m=1}^{I} exp(\boldsymbol{w}_m^{cl^T}.\boldsymbol{F}_t + b_m^{cl})}. \tag{6}$$

4 Proposed Solution

4.1 Volterra Filter based Classification

In our approach we propose a Volterra Filter structure to approximate a function $g(.)$. Given that video data is of interest here, a spatio-temporal Volterra Filter must be applied. As a result, this 3D version of the Volterra Filter discussed in Section 2 is used to extract the features,

$$\boldsymbol{F}_{\begin{bmatrix} t \\ s_1 \\ s_2 \end{bmatrix}} = g\left(\boldsymbol{X}_{\begin{bmatrix} t-L+1:t \\ s_1-p_1:s_1+p_1 \\ s_2-p_2:s_2+p_2 \end{bmatrix}}\right) = \sum_{\tau_1,\sigma_{11},\sigma_{21}} \boldsymbol{W}^{\boldsymbol{1}}_{\begin{bmatrix} \tau_1 \\ \sigma_{11} \\ \sigma_{21} \end{bmatrix}} x_{\begin{bmatrix} t-\tau_1 \\ s_1-\sigma_{11} \\ s_2-\sigma_{21} \end{bmatrix}}$$
$$+ \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21} \\ \tau_2,\sigma_{12},\sigma_{22}}} \boldsymbol{W}^{\boldsymbol{2}}_{\begin{bmatrix} \tau_1 \\ \sigma_{11} \\ \sigma_{21} \end{bmatrix}\begin{bmatrix} \tau_2 \\ \sigma_{12} \\ \sigma_{22} \end{bmatrix}} x_{\begin{bmatrix} t-\tau_1 \\ s_1-\sigma_{11} \\ s_2-\sigma_{21} \end{bmatrix}} x_{\begin{bmatrix} t-\tau_2 \\ s_1-\sigma_{12} \\ s_2-\sigma_{22} \end{bmatrix}} + ... \tag{7}$$

where, $\tau_j \in [0, L-1]$, $\sigma_{1j} \in [-p_1, p_1]$, and $\sigma_{2j} \in [-p_2, p_2]$. Following this formulation, and as discussed in Section 3, the linear classifier is used to determine the probability of each action in $\mathcal{A}$. Updating the filter parameters is pursued by minimizing some measure of discrepancy relative to the ground truth and the probability determined by the model. Our adopted measure herein is the cross-entropy loss computed as,

$$E = \sum_{t,I} -d_{t_i} log P_t(a_i), \tag{8}$$

where, $t \in \{1, L+1, 2L+1, ..., T\}$, $i \in \{1, 2, ..., I\}$, and d_{t_i} is the ground truth label for $\boldsymbol{X}_{[t-L+1:t]}$ belonging to the i^{th} action class. In addition to minimizing the error, we also include a weight decay in order to ensure generalizability of the model by penalizing large weights. So, the overall cost functional which serves as a target metric is written as,

$$\min_g \sum_{t,I} -d_{t_i} \log \rho(\boldsymbol{w}_i^{cl^T}.g(\boldsymbol{X}_{[t-L+1:t]}) + b_i^{cl}) + \frac{\lambda}{2}\left[\sum_{k=1}^{K} \left\|\boldsymbol{W}^{\boldsymbol{k}}\right\|_2^2 + \left\|\boldsymbol{W}^{cl}\right\|_2^2\right], \tag{9}$$

where ρ is the soft-max function, and K is the order of the filter.

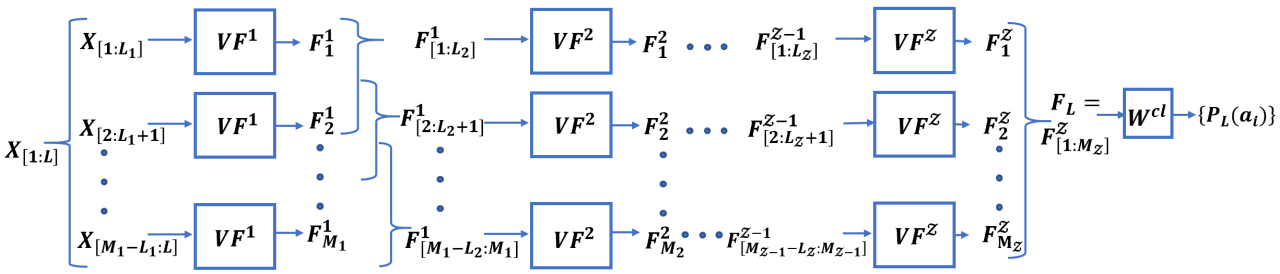

Figure 3: Block diagram for an Overlapping Volterra Neural Network

4.2 Non-Linearity Enhancement: Cascaded Volterra Filters

A major challenge in learning the afore-mentioned architecture arises when higher order non-linearities are sought. The number of required parameters for a K^{th} order filter is,

$$\sum_{k=1}^{K}(L.[2p_1+1].[2p_2+1])^k. \tag{10}$$

This complexity increases exponentially when the order is increased, thus making a higher order (> 3) Volterra Filter implementation impractical. To alleviate this limitation, we use a cascade of 2^{nd} order Volterra Filters, wherein, the second order filter is repeatedly applied until the desired order is attained. A K^{th} order filter is realized by applying the 2^{nd} order filter $\mathcal{Z}$ times, where, $K = 2^{2(\mathcal{Z}-1)}$. If the length of the first filter in the cascade is L_1, the input video $\boldsymbol{X}_{[t-L+1:t]}$ can be viewed as a concatenation of a set of shorter videos,

$$\boldsymbol{X}_{[t_L:t]} = \Big[\boldsymbol{X}_{[t_L:t_L+L_1]}\ \boldsymbol{X}_{[t_L+L_1:t_L+2L_1]} \\ ...\boldsymbol{X}_{[t_L+(M_1-1)L_1:t_L+M_1L_1]}\Big], \tag{11}$$

where $M_1 = \frac{L}{L_1}$, and $t_L = t-L+1$. Now, a 2^{nd} order filter $g_1(.)$ applied on each of the sub-videos leads to the features,

$$\boldsymbol{F}^1_{t_{[1:M_1]}} = \Big[g_1(\boldsymbol{X}_{[t_L:t_L+L_1]}) \quad g_1(\boldsymbol{X}_{[t_L+L_1:t_L+2L_1]}) \\ ...g_1(\boldsymbol{X}_{[t_L+(M_1-1)L_1:t_L+M_1L_1]})\Big]. \tag{12}$$

A second filter $g_2(.)$ of length L_2 is then applied to the output of the first filter,

$$\boldsymbol{F}^2_{t_{[1:M_2]}} = \Big[g_2(\boldsymbol{F}^1_{t_{[1:L_2]}}) \quad g_2(\boldsymbol{F}^1_{t_{[L_2+1:2L_2]}}) \\ ...g_2(\boldsymbol{F}^1_{t_{[(M_2-1)L_2+1:(M_2L_2)]}})\Big], \tag{13}$$

where, $M_2 = \frac{M_1}{L_2}$. Note that the features in the second layer are generated by taking quadratic interactions between those generated by the first layer, hence, leading to 4^{th} order terms.

Finally, for a cascade of $\mathcal{Z}$ filters, the final set of features is obtained as,

$$\boldsymbol{F}^{\mathcal{Z}}_{t_{[1:M_{\mathcal{Z}}]}} = \Big[g_{\mathcal{Z}}(\boldsymbol{F}^{\mathcal{Z}-1}_{t_{[1:L_{\mathcal{Z}}]}}) \quad g_{\mathcal{Z}}(\boldsymbol{F}^{\mathcal{Z}-1}_{t_{[L_{\mathcal{Z}}+1:2L_{\mathcal{Z}}]}}) \\ ...g_{\mathcal{Z}}(\boldsymbol{F}^{\mathcal{Z}-1}_{t_{[(M_{\mathcal{Z}}-1)L_{\mathcal{Z}}+1:(M_{\mathcal{Z}}L_{\mathcal{Z}})]}})\Big], \tag{14}$$

where, $M_{\mathcal{Z}} = \frac{M_{\mathcal{Z}-1}}{L_{\mathcal{Z}}}$.

Note that these filters can also be implemented in an overlapping fashion leading to the following features for the z^{th} layer, $z \in \{1, ..., \mathcal{Z}\}$,

$$\boldsymbol{F}^{z}_{t_{[1:M_z]}} = \Big[g_z(\boldsymbol{F}^{z-1}_{t_{[1:L_z]}}) \quad g_z(\boldsymbol{F}^{z-1}_{t_{[2:L_z+1]}}) \\ ...g_z(\boldsymbol{F}^{z-1}_{t_{[(M_{z-1})-L_z+1:M_{z-1}]}})\Big], \tag{15}$$

where $M_z = M_{z-1} - L_z + 1$. The implementation of an Overlapping Volterra Neural Network (O-VNN) to find the corresponding feature maps for an input video is shown in Figure 3.

Proposition 1. *The complexity of a K^{th} order cascaded Volterra filter will consist of,*

$$\sum_{z=1}^{\mathcal{Z}}\Big[(L_z.[2p_{1_z}+1].[2p_{2_z}+1]) \\ +(L_z.[2p_{1_z}+1].[2p_{2_z}+1])^2\Big] \tag{16}$$

parameters.

Proof. For a 2^{nd} order filter ($K = 2$), the number of parameters required is $\Big[(L.[2p_1+1].[2p_2+1]) + (L.[2p_1+1].[2p_2+1])^2\Big]$ (from equation 10). When such a filter is repeatedly applied $\mathcal{Z}$ times, it will lead to $\sum_{z=1}^{\mathcal{Z}}\Big[(L_z.[2p_{1_z}+1].[2p_{2_z}+1])+(L_z.[2p_{1_z}+1].[2p_{2_z}+1])^2\Big]$ parameters with order $K = 2^{2(\mathcal{Z}-1)}$. □

Furthermore, if a multi-channel input/output is considered, the number of parameters is,

$$\sum_{z=1}^{\mathcal{Z}}(n_{ch}^{z-1}.n_{ch}^{z})\Big[(L_z.[2p_{1_z}+1].[2p_{2_z}+1]) + (L_z.[2p_{1_z}+1].[2p_{2_z}+1])^2\Big], \quad (17)$$

where n_{ch}^{z} is the number of channels in the output of the z^{th} layer.

4.3 System Stability

The discussed system can be shown to be stable when the input is bounded, i.e. the system is Bounded Input Bounded Output (BIBO) stable.

Proposition 2. *An O-VNN with $\mathcal{Z}$ layers is BIBO stable if* $\forall z \in \{1, ..., \mathcal{Z}\}$,

$$\sum_{\tau_1,\sigma_{11},\sigma_{21}} \left| \boldsymbol{W}^{\boldsymbol{z1}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}} \right| + \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21}\\ \tau_2,\sigma_{12},\sigma_{22}}} \left| \boldsymbol{W}^{\boldsymbol{z2}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}} \right| < \infty. \quad (18)$$

Proof. Consider the z^{th} layer in the Cascaded implementation of the Volterra Filter,

$$\boldsymbol{F}^{z}_{[1:M_z]} = \Big[g_z(\boldsymbol{F}^{z-1}_{t_{[1:L_z]}}) \quad g_z(\boldsymbol{F}^{z-1}_{t_{[2:L_z+1]}}) \; ... g_z(\boldsymbol{F}^{z-1}_{t_{[(M_{z-1})-L_z+1:(M_{z-1})]}}) \Big], \quad (19)$$

where, $M_z = \frac{M_{z-1}}{Lz}$. Then for $m_z \in \{1, ..., M_z\}$,

$$\left| \boldsymbol{F}^{z}_{\begin{bmatrix} m_z\\ s_1\\ s_2\end{bmatrix}} \right| = \left| g_z\left(\boldsymbol{F}^{z-1}_{\begin{bmatrix} m_{z-1}-L_z+1:m_z\\ s_1-p_1:s_1+p1\\ s_2-p_2:s_2:p_2\end{bmatrix}} \right) \right| \quad (20)$$

$$= \Bigg| \sum_{\tau_1,\sigma_{11},\sigma_{21}} \boldsymbol{W}^{\boldsymbol{z1}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}} f^{z-1}_{\begin{bmatrix}(L_z+m_z-1)-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} + \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21}\\ \tau_2,\sigma_{12},\sigma_{22}}} \boldsymbol{W}^{\boldsymbol{z2}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}} f^{z-1}_{\begin{bmatrix}(L_z+m_z-1)-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} f^{z-1}_{\begin{bmatrix}(L_z+m_z-1)-\tau_2\\ s_1-\sigma_{12}\\ s_2-\sigma_{22}\end{bmatrix}} \Bigg| \quad (21)$$

$$\leq \sum_{\tau_1,\sigma_{11},\sigma_{21}} \left| \boldsymbol{W}^{\boldsymbol{z1}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}} \right| \left| f^{z-1}_{\begin{bmatrix}(L_z+m_z-1)-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} \right| + \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21}\\ \tau_2,\sigma_{12},\sigma_{22}}} \left| \boldsymbol{W}^{\boldsymbol{z2}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}} \right| \left| f^{z-1}_{\begin{bmatrix}(L_z+m_z-1)-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} \right| \left| f^{z-1}_{\begin{bmatrix}(L_z+m_z-1)-\tau_2\\ s_1-\sigma_{12}\\ s_2-\sigma_{22}\end{bmatrix}} \right| \quad (22)$$

$$\leq A \sum_{\tau_1,\sigma_{11},\sigma_{21}} \left| \boldsymbol{W}^{\boldsymbol{z1}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}} \right| + A^2 \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21}\\ \tau_2,\sigma_{12},\sigma_{22}}} \left| \boldsymbol{W}^{\boldsymbol{z2}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}} \right|. \quad (23)$$

Note that Equation 23 states that a bounded input yields $\sum_{\tau_1,\sigma_{11},\sigma_{21}} \left| f^{z-1}_{\begin{bmatrix}(L_z+m_z-1)-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} \right| \leq A$, for some $A < \infty$. Hence, the sufficient condition for the system to be BIBO stable is,

$$\sum_{\tau_1,\sigma_{11},\sigma_{21}} \left| \boldsymbol{W}^{\boldsymbol{z1}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}} \right| + \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21}\\ \tau_2,\sigma_{12},\sigma_{22}}} \left| \boldsymbol{W}^{\boldsymbol{z2}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}} \right| < \infty. \quad (24)$$

If the input data (i.e. video frames) is bounded, so is the output of each layer provided that Equation 24 is satisfied $\forall z \in \{1, ..., Z\}$, making the entire system BIBO stable. □

4.4 Synthesis and Implementation of Volterra Kernels

As noted earlier, the linear kernel (1^{st} order) of the Volterra filter is similar to the convolutional layer in the conventional CNNs. As a result, it can be easily implemented using the 3D convolution function in Tensorflow (Abadi et al. 2016). The second order kernel may be approximated as a product of two 3-dimensional matrices (i.e. a seperable operator),

$$\boldsymbol{W}^{\boldsymbol{2}}_{L\times P_1\times P_2\times L\times P_1\times P_2} = \sum_{q=1}^{Q} \boldsymbol{W}^{\boldsymbol{2}}_{a_{q_{L\times P_1\times P_2\times 1}}} \boldsymbol{W}^{\boldsymbol{2}}_{b_{q_{1\times L\times P_1\times P_2}}}, \quad (25)$$

where, $P_1 = 2p_1 + 1$, and $P_2 = 2p_2 + 1$. Taking account of Equation 7 yields,

$$g\left(\boldsymbol{X}_{\begin{bmatrix} t-L+1:t\\ s_1-p_1:s_1+p_1\\ s_2-p_2:s_2+p_2\end{bmatrix}} \right) = \sum_{\tau_1,\sigma_{11},\sigma_{21}} \boldsymbol{W}^{\boldsymbol{1}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}} x_{\begin{bmatrix} t-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} + \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21}\\ \tau_2,\sigma_{12},\sigma_{22}}} \sum_{q=1}^{Q} \boldsymbol{W}^{\boldsymbol{2}}_{a_{q_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}}}} \boldsymbol{W}^{\boldsymbol{2}}_{b_{q_{\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}}}} x_{\begin{bmatrix} t-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} x_{\begin{bmatrix} t-\tau_2\\ s_1-\sigma_{12}\\ s_2-\sigma_{22}\end{bmatrix}} \quad (26)$$

$$= \sum_{\tau_1,\sigma_{11},\sigma_{21}} \boldsymbol{W}^{\boldsymbol{1}}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}} x_{\begin{bmatrix} t-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} + \sum_{q=1}^{Q} \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21}\\ \tau_2,\sigma_{12},\sigma_{22}}} \left(\boldsymbol{W}^{\boldsymbol{2}}_{a_{q_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}}}} x_{\begin{bmatrix} t-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} \right) \left(\boldsymbol{W}^{\boldsymbol{2}}_{b_{q_{\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}}}} x_{\begin{bmatrix} t-\tau_2\\ s_1-\sigma_{12}\\ s_2-\sigma_{22}\end{bmatrix}} \right). \quad (27)$$

A larger Q will provide a better approximation of the 2^{nd} order kernel. The advantage of this class of approximation is two-fold. Firstly, the number of parameters can be further reduced, if for the z^{th} layer, $(L_z.[2p_{1_z}+1].[2p_{2_z}+1])^2 > 2Q(L_z.[2p_{1_z}+1].[2p_{2_z}+1])$. The trade-off between performance and available computational resources must be accounted for when opting for such an approximation. Additionally, this makes it easier to implement the higher order kernels in Tensorflow (Abadi et al. 2016) by using the built in convolutional operator.

The approximate quadratic layers in the Cascaded Volterra Filter (see Figure 3) yield the following number of parameters,

$$\sum_{z=1}^{Z}\Big[(L_z.[2p_{1_z}+1].[2p_{2_z}+1]) + 2Q(L_z.[2p_{1_z}+1].[2p_{2_z}+1])\Big]. \quad (28)$$

We will evaluate and compare both approaches when implementing the second order kernel (i.e. approximation and exact method) in Section 5.

4.5 Two-Stream Volterra Networks

Most previous studies in action recognition in videos have noted the importance of using both the spatial and the temporal information for an improved recognition accuracy. As a result, we also propose the use of Volterra filtering in combining the two information streams, exploring a potential non-linear relationship between them. In Section 5 we will see that this actually boosts the performance, thereby indicating some non-linear relation between the two information streams. Independent Cascaded Volterra Filters are first used in order to extract features from each modality,

$$\boldsymbol{F}^{\mathcal{Z}^{RGB}}_{[1:M_{\mathcal{Z}}]} = g^{RGB}_{\mathcal{Z}}(...g^{RGB}_{2}(g^{RGB}_{1}(\boldsymbol{X}^{RGB}_{[t-L+1:t]}))) \quad (29)$$

$$\boldsymbol{F}^{\mathcal{Z}^{OF}}_{[1:M_{\mathcal{Z}}]} = g^{OF}_{\mathcal{Z}}(...g^{OF}_{2}(g^{OF}_{1}(\boldsymbol{X}^{OF}_{[t-L+1:t]}))). \quad (30)$$

Upon gleaning features from the two streams, an additional Volterra Filter is solely used for combining the generated feature maps from both modalities,

$$F^{(RGB+OF)}_{t} = \sum_{\tau_1,\sigma_{11},\sigma_{21},u_1} \boldsymbol{W^1}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\\ u_1\end{bmatrix}} f^{\mathcal{Z}^{u_1}}_{\begin{bmatrix}M_{\mathcal{Z}}-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} + \sum_{\substack{\tau_1,\sigma_{11},\sigma_{21},u_1\\ \tau_2,\sigma_{12},\sigma_{22},u_2}} \boldsymbol{W^2}_{\begin{bmatrix}\tau_1\\ \sigma_{11}\\ \sigma_{21}\end{bmatrix}\begin{bmatrix}\tau_2\\ \sigma_{12}\\ \sigma_{22}\end{bmatrix}} f^{\mathcal{Z}^{u_1}}_{\begin{bmatrix}M_{\mathcal{Z}}-\tau_1\\ s_1-\sigma_{11}\\ s_2-\sigma_{21}\end{bmatrix}} f^{\mathcal{Z}^{u_2}}_{\begin{bmatrix}M_{\mathcal{Z}}-\tau_2\\ s_1-\sigma_{12}\\ s_2-\sigma_{22}\end{bmatrix}}, \quad (31)$$

where $\tau_j \in [0, L_{\mathcal{Z}+1}]$, $\sigma_{1j} \in [-p_1, p_1]$, $\sigma_{2j} \in [-p_2, p_2]$, and $u_j \in [RGB, OF]$. Figure 6-(c) shows the block diagram for fusing the two information streams.

5 Experiments and Results

We proceed to evaluate the performance of this approach on two action recognition datasets, namely, UCF-101 (Soomro, Zamir, and Shah 2012) and HMDB-51 (Kuehne et al. 2011), and the comparison of the results with recent state of the art implementations is given in Tables 1 and 2. Table 1 shows the comparison with techniques that only exploit the RGB stream, while Table 2 shows the comparison when both information streams are used. Note that our comparable performance to the state of the art is achieved with a significantly lower number of parameters (see Table 4). Furthermore, a significant boost in performance is achieved by allowing non-linear interaction between the two information streams. The Optical Flow is computed using the TV-L1 algorithm (Zach, Pock, and Bischof 2007). Note that we train the network from scratch on both datasets, and do not use a larger dataset for pre-training, in contrast to some of the previous implementations. The implementations that take advantage of a different dataset for pre-training are indicated by a *'Y'* in the pre-training column, while those that do not, are indicated by *'N'*. When training from scratch the proposed solution is able to achieve best performance for both scenarios: one stream networks (RGB frames only) and two-stream networks (RGB frames & Optical Flow). To fuse the two information streams (spatial and temporal), we evaluate the following techniques:

1. *Decision Level Fusion (Figure 6-(a)):* The decision probabilities $P^{RGB}_t(a_i)$ and $P^{OF}_t(a_i)$ are independently computed and are combined to determine the fused probability $P^{f}_t(a_i)$ using *(a) Weighted Averaging:* $P^{f}_t(a_i) = \beta^{RGB}P^{RGB}_t(a_i) + \beta^{OF}P^{OF}_t(a_i)$, where $\beta^{RGB} + \beta^{OF} = 1$, which control the importance/contribution of the RGB and Optical Flow streams towards making a final decision, or *(b) Event Driven Fusion (Roheda et al. 2018a; 2019):* $P^{f}_t(a_i) = \gamma P^{\text{MAX MI}}_t(a^{RGB}_i, a^{OF}_i) + (1-\gamma)P^{\text{MIN MI}}_t(a^{RGB}_i, a^{OF}_i)$, where γ is a pseudo measure of correlation between the two information streams, $P^{\text{MAX MI}}_t(.)$ is the joint distribution with maximal mutual information, and $P^{\text{MIN MI}}_t(.)$ is the joint distribution with minimal mutual information.
2. *Feature Level Fusion:* Features are extracted from each stream independently, and are subsequently merged before making a decision. For this level of fusion we consider a simple *Feature Concatenation* (see Figure 6-(b)), and *Two-Stream Volterra Filtering* (see Section 4.5, Figure 6-(c)).

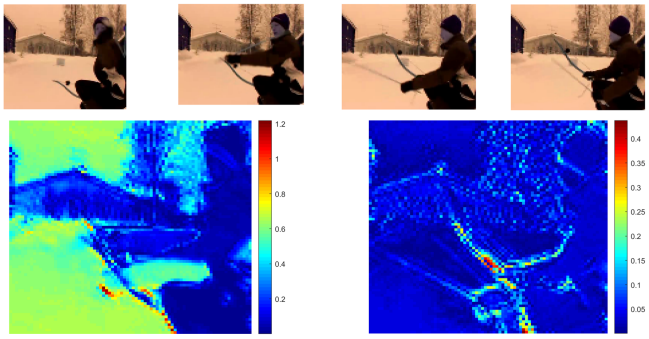

Figure 4: (a): Input Video, (b): Features extracted by only RGB stream, (c): Features extracted by Two-Stream Volterra Filtering

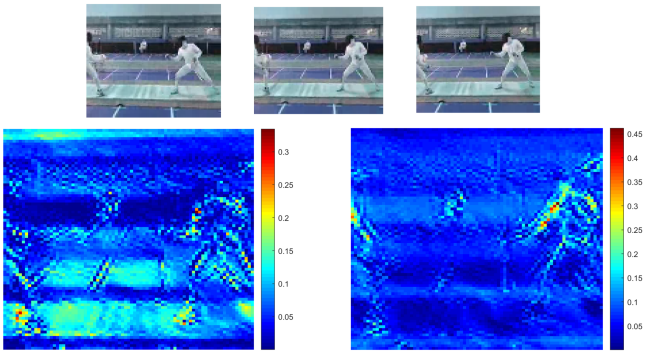

Figure 5: (a): Input Video, (b): Features extracted by only RGB stream, (c): Features extracted by Two-Stream Volterra Filtering

The techniques are summarized in Figure 6. In our implementation we use an O-VNN with 8 layers on both the RGB stream and the optical stream. Each layer uses $L_z = 2$ and

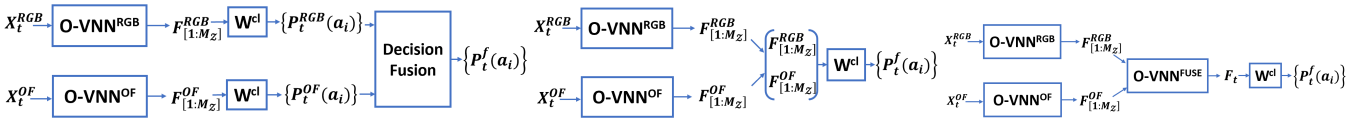

Figure 6: (a): Decision Level Fusion, (b): Feature Concatenation, (c): Two-Stream Volterra Filtering

Method	**Pre-Training**	**Avg Accuracy UCF-101**	**Avg Accuracy HMDB-51**
Slow Fusion (Karpathy et al. 2014)	*Y* (Sports-1M)	64.1 %	-
Deep Temporal Linear Encoding Networks (Diba, Sharma, and Van Gool 2017)	*Y* (Sports-1M)	86.3 %	60.3 %
Inflated 3D CNN (Carreira and Zisserman 2017)	*Y* (ImageNet + Kinetics)	**95.1 %**	**74.3 %**
Soomro et al, 2012	*N*	43.9 %	-
Single Frame CNN (Karpathy et al. 2014; Krizhevsky, Sutskever, and Hinton 2012)	*N*	36.9 %	-
Slow Fusion (Karpathy et al.; Baccouche et al.; Ji et al.)	*N*	41.3 %	-
3D-ConvNet (Carreira and Zisserman 2017; Tran et al. 2015)	*N*	51.6 %	24.3 %
Volterra Filter	*N*	38.19 %	18.76 %
O-VNN (exact)	*N*	**58.73 %**	**29.33 %**
O-VNN (Q=7)	*N*	53.77 %	25.76 %

Table 1: Performance Evaluation for one stream networks (RGB only): The proposed algorithm achieves best performance when trained from scratch

Method	**Pre-Training**	**Avg Accuracy UCF-101**	**Avg Accuracy HMDB-51**
Two-Stream CNNs (Simonyan and Zisserman 2014)	*Y* (ILSVRC-2012)	88.0 %	72.7 %
Deep Temporal Linear Encoding Networks (Diba, Sharma, and Van Gool 2017)	*Y* (BN-Inception + ImageNet)	95.6 %	71.1 %
Two Stream Inflated 3D CNN (Carreira and Zisserman 2017)	*Y* (ImageNet + Kinetics)	98.0 %	80.9 %
Two-Stream O-VNN (Q=15)	*Y* (Kinetics)	**98.49 %**	**82.63 %**
Two Stream Inflated 3D CNN (Carreira and Zisserman 2017)	*N*	88.8 %	62.2 %
Weighted Averaging: O-VNN (exact)	*N*	85.79 %	59.13 %
Weighted Averaging: O-VNN (Q=7)	*N*	81.53 %	55.67 %
Event Driven Fusion: O-VNN (exact)	*N*	85.21 %	60.36 %
Event Driven Fusion: O-VNN (Q=7)	*N*	80.37 %	57.89 %
Feature Concatenation: O-VNN (exact)	*N*	82.31 %	55.88 %
Feature Concatenation: O-VNN (Q=7)	*N*	78.79 %	51.08 %
Two-Stream O-VNN (exact)	*N*	**90.28 %**	**65.61 %**
Two-Stream O-VNN (Q=7)	*N*	86.16 %	62.45 %

Table 2: Performance Evaluation for two stream networks (RGB & Optical Flow): The proposed algorithm achieves best performance on both datasets.

$p_{1_z}, p_{2_z} \in \{0, 1, 2\}$. The outputs of the two filters are fed into the fusion layer which combines the two streams. The fusion layer uses $L_{\text{Fuse}} = 2$ and $p_{1_{\text{Fuse}}}, p_{2_{\text{Fuse}}} \in \{0, 1, 2\}$. It is clear from Table 2 that performing fusion using Volterra Filters significantly boosts the performance of the system. This shows that there does exist a non-linear relationship between the two modalities. This can also be confirmed from the fact that we can see significant values in the weights for the fusion layer (see Table 3). Figures 4 and 5 show one of the

	$u = RGB$	$u = OF$	$u = Fusion$
$\frac{1}{2}\|\boldsymbol{W}^u\|_2^2$	352.15	241.2	341.3

Table 3: Norm of $\boldsymbol{W}^u$, where $u \in [RGB, OF, Fusion]$.

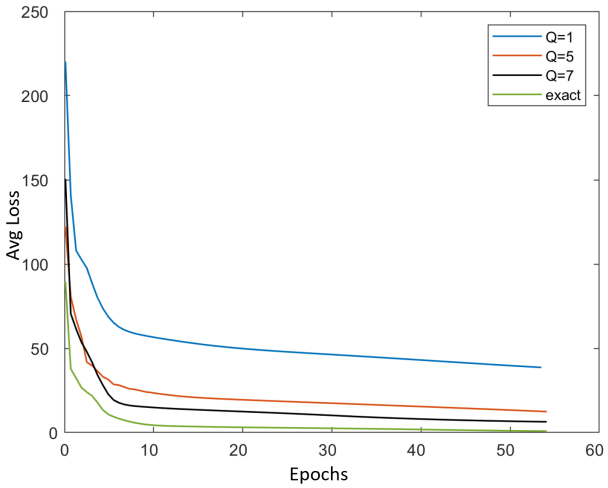

Figure 7: Epochs vs Loss for various number of multipliers for a Cascaded Volterra Filter

feature maps for an archery video and a fencing video. From Figures 4, 5-(b),(c) it can be seen that when only the RGB stream is used, a lot of the background area has high values, while on the other hand, when both streams are jointly used, the system is able to concentrate on more relevant features. In 4-(c), the system is seen to concentrate on the bow and arrow which are central to recognizing the action, while in 5-(c) the system is seen to concentrate on the pose of the human which is central to identifying a fencing action. Figure 7 shows the Epochs vs Loss graph for a Cascaded Volterra Filter when a different number of multipliers (Q) are used to approximate the 2^{nd} order kernel. The green plot shows the loss when the exact kernel is learned, and it can be seen that the performance comes closer to the exact kernel as Q is increased.

6 Conclusion

We proposed a novel network architecture for recognition of actions in videos, where the non-linearities were introduced by the Volterra Series Formulation. We propose a Cascaded Volterra Filter which leads to a significant reduction

Method	Number of Parameters	Processing Speed (secs/video)
Deep Temporal Linear Encoding	22.6M	-
Inflated 3D CNN	25M	3.52
O-VNN (exact)	**4.6M**	0.73
O-VNN (Q=7)	3.7M	0.61
Two-Stream O-VNN (exact)	**10.1M**	**1.88**
Two-Stream O-VNN (Q=7)	8.2M	1.54
Two-Stream O-VNN (Q=1)	2.5M	0.34

Table 4: Comparison of number of parameters required and processing speed with the state of the art. A video with 60 frames is evaluated

in parameters while exploring the same complexity of non-linearities in the data. Such a Cascaded Volterra Filter was also shown to be a BIBO stable system. In addition, we also proposed the use of the Volterra Filter to fuse the spatial and temporal streams, hence leading to a non-linear fusion of the two streams. The network architecture inspired by Volterra Filters achieves performance accuracies which are comparable to the state of the art approaches while using a fraction of the number of parameters used by them. This makes such an architecture very attractive from a training point of view, with an added advantage of reduced storage space to save the model.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 265–283.

Baccouche, M.; Mamalet, F.; Wolf, C.; Garcia, C.; and Baskurt, A. 2011. Sequential deep learning for human action recognition. In *International workshop on human behavior understanding*, 29–39. Springer.

Carreira, J., and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6299–6308.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Diba, A.; Sharma, V.; and Van Gool, L. 2017. Deep temporal linear encoding networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2329–2338.

Farabet, C.; Couprie, C.; Najman, L.; and LeCun, Y. 2012. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1915–1929.

Feichtenhofer, C.; Pinz, A.; and Zisserman, A. 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1933–1941.

Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T. 2016. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 317–326.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Hoffman, J.; Gupta, S.; and Darrell, T. 2016. Learning with side information through modality hallucination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 826–834.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134.

Ji, S.; Xu, W.; Yang, M.; and Yu, K. 2012. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35(1):221–231.

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1725–1732.

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Kong, Y., and Fu, Y. 2018. Human action recognition and prediction: A survey. *arXiv preprint arXiv:1806.11230*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T. 2011. Hmdb: a large video database for human motion recognition. In *2011 International Conference on Computer Vision*, 2556–2563. IEEE.

Kumar, R.; Banerjee, A.; Vemuri, B. C.; and Pfister, H. 2011. Trainable convolution filters and their application to face recognition. *IEEE transactions on pattern analysis and machine intelligence* 34(7):1423–1436.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear cnns for fine-grained visual recognition. *arXiv preprint arXiv:1504.07889*.

Liu, J.; Luo, J.; and Shah, M. 2009. Recognizing realistic actions from videos in the wild. Citeseer.

Niebles, J. C.; Chen, C.-W.; and Fei-Fei, L. 2010. Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision*, 392–405. Springer.

Osowski, S., and Quang, T. V. 1994. Multilayer neural network structure as volterra filter. In *Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS'94*, volume 6, 253–256. IEEE.

Roheda, S.; Krim, H.; Luo, Z.-Q.; and Wu, T. 2018a. Decision level fusion: An event driven approach. In *2018 26th European Signal Processing Conference (EUSIPCO)*, 2598–2602. IEEE.

Roheda, S.; Riggan, B. S.; Krim, H.; and Dai, L. 2018b. Cross-modality distillation: A case for conditional generative adversarial networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2926–2930. IEEE.

Roheda, S.; Krim, H.; Luo, Z.-Q.; and Wu, T. 2019. Event driven fusion. *arXiv preprint arXiv:1904.11520*.

Schetzen, M. 1980. The volterra and wiener theories of nonlinear systems.

Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; and LeCun, Y. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

Simonyan, K., and Zisserman, A. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, 568–576.

Sivic, J., and Zisserman, A. 2003. Video google: A text retrieval approach to object matching in videos. In *null*, 1470. IEEE.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 4489–4497.

Volterra, V. 2005. *Theory of functionals and of integral and integro-differential equations*. Courier Corporation.

Wang, H.; Ullah, M. M.; Klaser, A.; Laptev, I.; and Schmid, C. 2009. Evaluation of local spatio-temporal features for action recognition.

Yi, S.; Krim, H.; and Norris, L. K. 2011. Human activity modeling as brownian motion on shape manifold. In *International Conference on Scale Space and Variational Methods in Computer Vision*, 628–639. Springer.

Zach, C.; Pock, T.; and Bischof, H. 2007. A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, 214–223. Springer.

Zoumpourlis, G.; Doumanoglou, A.; Vretos, N.; and Daras, P. 2017. Non-linear convolution filters for cnn-based learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 4761–4769.