Deep End2End Voxel2Voxel Prediction

Du Tran[1,2], Lubomir Bourdev[3], Rob Fergus[1], Lorenzo Torresani[2], Manohar Paluri[1]
[1]Facebook AI Research, [2]Dartmouth College, [3]UC Berkeley
{dutran,lorenzo}@cs.dartmouth.edu {trandu,robfergus,mano}@fb.com lubomir.bourdev@gmail.com

Abstract

Over the last few years deep learning methods have emerged as one of the most prominent approaches for video analysis. However, so far their most successful applications have been in the area of video classification and detection, i.e., problems involving the prediction of a single class label or a handful of output variables per video. Furthermore, while deep networks are commonly recognized as the best models to use in these domains, there is a widespread perception that in order to yield successful results they often require time-consuming architecture search, manual tweaking of parameters and computationally intensive preprocessing or post-processing methods.

In this paper we challenge these views by presenting a deep 3D convolutional architecture trained end to end to perform voxel-level prediction, i.e., to output a variable at every voxel of the video. Most importantly, we show that the same exact architecture can be used to achieve competitive results on three widely different voxel-prediction tasks: video semantic segmentation, optical flow estimation, and video coloring. The three networks learned on these problems are trained from raw video without any form of preprocessing and their outputs do not require post-processing to achieve outstanding performance. Thus, they offer an efficient alternative to traditional and much more computationally expensive methods in these video domains.

1. Introduction

During the last decade we have witnessed a tremendous growth in the number of videos created and shared on the Internet thanks to the advances in network bandwidth and computation. In turn this has lead to a strong effort toward the creation of better tools and apps to search, browse and navigate these large and continuously expanding video collections. This poses new challenges for the computer vision community and gives new motivations to build better, faster and more generally applicable video analysis methods.

In the still-image domain deep learning has revolutionized the traditional computer vision pipeline, which typically consisted of: pre-processing, hand-construction of visual features, training of a learning model, and post-processing. Instead, the successful introduction of deep convolutional neural network [15, 11, 23, 25] has shown that much better results can be obtained through end to end learning on very large collections of image examples, where the network is trained on raw image input and it directly predicts the target output. Besides the demonstrated advantages in improved accuracy, these end to end learned models have also been shown to be often more computationally efficient than traditional hand-designed approaches because they eliminate the need for computationally expensive preprocessing and post-processing steps and because convolution can run very fast, particularly on GPUs.

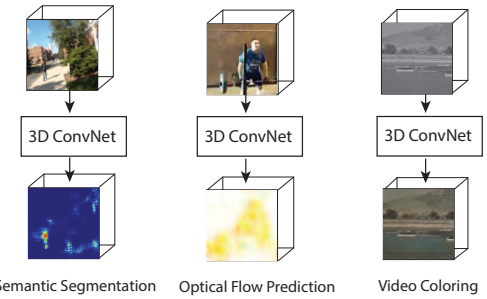

Figure 1. **Voxel to voxel prediction**: is a fine-grained video understanding task where the algorithm need to infer a variable for each input voxel. The problem has many potential applications including video semantic segmentation, optical flow prediction, depth estimation, and video coloring.

The video domain is also harnessing the benefits of this revolution but it is still lagging compared to the image setting [7, 32, 27]. In particular, most of the end to end learning approaches for video analysis have been introduced in the area of classification and detection [14, 24, 29, 28] and involve predicting a single label or few output variables per video. However, there are many computer vision problems that require labeling every single voxel of a video. Examples include optical flow computation, video semantic segmentation, depth estimation and video coloring. There have been some attempts at approaching these

pixel-labeling problems with deep learning [17, 10, 9] for images. One of the reasons is that deep networks typically involve a large set of pooling layers which significantly lower the spatial resolution of the output. In order to output pixel labels at the original resolution, several "unpooling" strategies have been proposed, including simple upsampling, and multi-scale approaches. One of the most promising solution in this genre is learning convolution filters that upsample the signal. The primary benefit of convolutional upsampling is that it only requires learning a small number of location-agnostic filters and thus it can be carried out with limited training data.

The objective of our work is to demonstrate that 3D convolutional networks (3D ConvNets) with upsampling layers enable highly effective end to end learning of voxel to voxel prediction models on various video analysis problems. Instead of building a highly specialized network for each problem, our goal is to show that the same 3D ConvNet architecture trained on three distinct application domains (optical flow prediction, semantic segmentation, video coloring) can produce competitive results on each of them. Although a thorough architecture search is likely to yield improved results, we find it useful to employ a single network model for the three distinct tasks to convey the message that deep learning methods do not necessarily require to be highly specialized for the task at hand in order to produce good results. For the same reason, we do not employ any pre-processing or post-processing of the data. Because our model is fully convolutional, it involves a small number of learning parameters which can be optimized with limited amount of supervised data. Furthermore, the elimination of computationally expensive pre-processing and post-processing methods (such as CRF optimization or variational inference) and the exclusive reliance on efficient convolution implies that our learned models run very fast and can be used in real-time video-processing applications such as those arising in big-data domains.

In summary, our work provides the following findings:

1. Fully convolutional 3D ConvNets enable end to end learning of voxel to voxel prediction models with limited training data.

2. The same exact architecture can be employed to obtain competitive results on three different voxel-labeling applications: optical flow estimation, semantic segmentation of image sequences, and video coloring.

3. In domains where supervised training data is scarce (such as in the case of optical flow), we can train our end to end learning model on the output of an existing hand-designed algorithm. We show that this results in a 3D ConvNet that achieves slightly better accuracy than the complex hand-tuned vision method but, most importantly, it is significantly more efficient.

4. While fine-tuning a pre-trained model helps in most cases, it actually hurts when the new domain requires visual features that are quite distinct from those of the pre-learned model, such as in the case of fine-tuning an action recognition network for optical flow estimation.

2. Related Work

Video analysis has been studied by the computer vision community for decades. Different approaches were proposed for action recognition including: tracking-based methods [8], bag-of-visual words [19], biologically-inspired models [13], space-time shapes [3], HMMs [12], and template-based Action-Bank [21]. Different spatio-temporal features were also introduced for video and action classification: Spatio-Temporal Interest Points [16], improved Dense Trajectories [29]. Various methods were used for action and video event detection [22, 6, 30]. Although these methods showed to work reasonably well, they are not scalable because most of them require computational intensive steps during preprocessing (e.g. tracking, background subtraction, or feature extraction) or post-processing (CRF, variational inference).

Deep learning methods have recently shown good on different computer vision problems [27, 23, 18, 11, 2]. Thanks to their large learning capacity and the ability to optimize all parameters end to end, these methods achieved good performance on classification [15] and feature learning [27, 28] provided that there is sufficient supervised training data. Among the deep learning approaches, our proposed method is most closely related to the depth estimation method described in [9], the Fully Convolutional Network (FCN) [17], and FlowNet [10]. Our method shares with these approaches the property of making pixel-level predictions. However, all these prior methods are designed for still image problems, while our method operates on videos. To the best of our knowledge, our method is the first one addressing end-to-end training of video voxel prediction.

3. Video Voxel Prediction

Problem statement. The input to our system is video with size $C \times L \times H \times W$, where C is the number of color channels, L is its temporal length (in number of frames), and H, W are the frame height and width. Then, a voxel prediction problem requires producing a target output of size $K \times L \times H \times W$, where K is an application-dependent integer denoting the number of output variables that need to be predicted *per voxel*. It is worth nothing that the size of the input video and the output prediction are the same, except only for the number of input channels C and the number of output channels K are different. Normally, $C = 3$ for the case of color video inputs and $C = 1$ for gray-scale inputs. For the three voxel-prediction applications considered

in this paper, K will have the following values: $K = 2$ for optical flow estimation (the horizontal and vertical motion displacement for each voxel), $K = 3$ for video coloring (the three color channels) and K will be equal to the number of semantic classes in the case of video semantic segmentation.

Proposed approach. We propose a novel and unified approach for video voxel prediction based on a 3D ConvNet architecture with 3D deconvolution layers. We show the generality of the model by demonstrating that a simple unified architecture can work reasonably well across different tasks without any engineering efforts in architecture search. Since our method uses 3D deconvolution layers, we will start by briefly explaining the idea of 2D deconvolution [31, 17] and then present our architecture based on 3D deconvolution for voxel prediction.

Deconvolution. The concept of deconvolution was introduced by Zeiler and Fergus [31] to visualize the internal-layer filters of a 2D ConvNet. Because the objective of this prior work was merely filter visualization, there was no learning involved in the deconvolution layers and the weights were simply set to be equal to the transpose of the corresponding pre-trained convolution layers. Instead, Long *et al.* [17] introduced the idea of deconvolution as a trainable layer in 2D ConvNets with applications to image semantic segmentation. As shown in Figure 2, a filter of a trainable deconvolution layer acts as a learnable local upsampling unit. In convolution, input signals are convolved by the kernel filter and one value is placed on the output plane. Conversely, deconvolution takes one value from the input, multiples the value by the weights in the filter, and place the result in the output channel. Thus, if the 2D filter has size $s \times s$, it generates a $s \times s$ output matrix for each pixel input. The output matrices can be stored either overlapping or not overlapping in the output channel. If not overlapping, then deconvolution with a $s \times s$ filter would upsample the input by a factor s in both dimensions. When the output matrices overlap, their contributions in the overlap are summed up. The amount of output overlap depends on the output *stride*. If the output stride is bigger than 1, then the deconvolution layer produces an outputs with size larger than the input, thus acts as an upsampler.

In our architecture, we use *3D* deconvolutional layers, instead of 2D deconvolutional layers. This means that the filters are deconvolved spatio-temporally, instead of only spatially as in 2D ConvNets.

Architecture for voxel prediction. Our architecture (which we name V2V, for voxel-to-voxel) is adapted from the C3D network described in [28], which has shown good performance for different video recognition tasks. In order to apply it to voxel-prediction problems, we simply add 3D deconvolutional layers to the C3D network. Note that C3D operates by splitting the input video into clips of 16 frames each and perform prediction separately for each clip. Thus, our V2V model also takes as input a clip of 16 frames and then outputs voxel labels for the 16 input frames. Figure 3 illustrates our V2V architecture for voxel prediction. The lower part contains layers from C3D, while the upper part has *three* 3D convolutional layers, *three* 3D deconvolutional layers, *two* concatenation layers, and *one* loss layer. All three convolutional layers (`Conv3c`,`Conv4c`, and `Conv-pre`) use filters of size $3 \times 3 \times 3$ with stride $1 \times 1 \times 1$ and padding $1 \times 1 \times 1$. `Conv3c` and `Conv4c` act as feature-map reducers, while `Conv-pre` acts as a prediction layer. `Deconv5` and `Deconv4` use filters of size $4 \times 4 \times 4$ with output stride $2 \times 2 \times 2$ and padding $1 \times 1 \times 1$. The `Deconv3` layer uses kernels of size $8 \times 4 \times 4$, an output stride of $4 \times 2 \times 2$, and padding $2 \times 1 \times 1$. Note that the number written inside the box of each layer in the Figure indicates the number of filters (e.g., 64 for `Deconv3`). The voxel-wise loss layer and `Conv-pre` layer are application-dependent and will be described separately for each of the applications considered in this paper. Since V2V shares the bottom layers with C3D, we have the option to either fine-tuning these layers starting from the C3D weights, or learning the weights from scratch. We will report results for both options in our experiments.

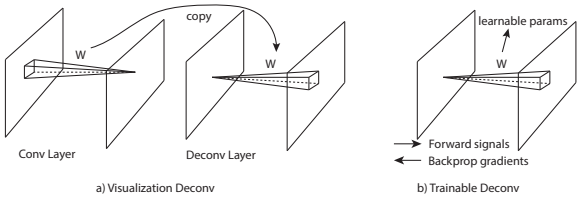

Figure 2. **Deconvolutional layers in ConvNets**. a) Visualization of the deconvolutional layer used in [31] where the filter weights are set to be equal to those of the pre-trained convolutional layer. b) Trainable deconvolutional layers [17] learn upsampling.

4. Application I: Video Semantic Segmentation

Dataset. Our experiments for video semantic segmentation are carried out on the GATECH dataset [20], which comes with a public training/test split. The training set contains 63 videos while the test set has 38 sequences. There are 8 semantic classes: sky, ground, solid (mainly buildings), porous (mainly trees), cars, humans, vertical mix, and main mix.

Training. Similarly to C3D, we down-scale the video frames to size 128×171. Because the dataset is quite small, we split each training video into all possible clips of length 16 (thus, we take overlapping clips with stride 1). For testing, we perform prediction on all non-overlapping clips of the video (stride equal to 16). We use the V2V architecture described in section 3 with $K = 8$ prediction channels, corresponding to the 8 semantic classes. We use a voxel-

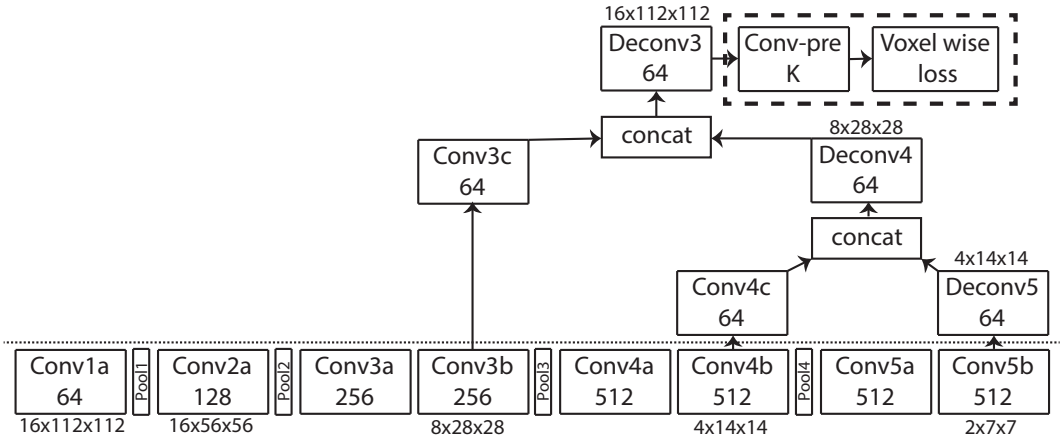

Figure 3. **V2V Architecture for Voxel Prediction**. The lower part (below dashed line) consists of layers from C3D [28]. Connected to these layers we have three 3D convolution layers: `Conv3c`,`Conv4c`,`Conv-pre` use filters of size $3 \times 3 \times 3$ with stride $1 \times 1 \times 1$. Both `Deconv5` and `Deconv4` are deconvolutional layers employing kernels of size $4 \times 4 \times 4$ with output stride of $2 \times 2 \times 2$. `Deconv3` has kernel size $8 \times 4 \times 4$ and output stride of $4 \times 2 \times 2$. The numbers inside the boxes represent the number of learning filters in that layer, while the numbers near the boxes (above or below) represent the size of output signals produced by that layer. The part inside the thick-dashed box is application-dependent.

wise softmax for the loss layer. We fine-tune the full V2V network initialized from C3D, using randomly initialized weights for the new layers. The learning rate is set initially to 10^{-4}, and it is divided by 10 every 30K iterations. The size of each mini-batch is 1. Fine-tuning is stopped at 100K iterations, approximately 9 epochs.

Baselines. We compare our V2V model with several baselines to gain better insights about our method. The first set of baselines are based on bilinear upsampling. The purpose of these baselines is to understand the benefits of our 3D deconvolution layers compared to simple upsampling. Instead of using V2V with deconvolution layers, we use only C3D up to `Conv5b`, we then add a prediction layer (analogous to `Conv-pre`). Because the prediction made at `Conv5b` has size $2 \times 7 \times 7$, we apply a bilinear upsampling to produce a prediction of the same size as the input. We call this baseline Conv5b-up. We include two other baselines, namely, Conv4b-up and Conv3b-up, corresponding to adding a prediction layer and an upsampling layer at Conv4b and Conv3b, respectively. The second set of baselines includes *V2V-remove-skip* and *V2V-remove-skip-0* where we try to understand how much the skip connections help our segmentation. *V2V-remove-skip* is an alternative architecture of V2V where we remove the skip connections from `conv3b` and `conv4b`, so that the network has only a single stream from `conv5b` followed by a few deconv layers and a prediction layer. *V2V-remove-skip-0* denotes the same architecture but trained from scratch instead of being fine-tuned from C3D. Finally, we also compare our fine-tuned V2V model with the V2V architecture trained from scratch on GATECH, which we call V2V-0. We also trained a 2D version of V2V, namely 2D-V2V. The model 2D-V2V has the same architecture as V2V except that all 3D convolutional layers, 3D pooling layers, and 3D deconvolutional layers are replaced with 2D convolutional layers, 2D pooling layers, and 2D deconvolutional layers, respectively. As we do not a have pre-trained model of 2D-V2V, we train 2D-V2V from scratch on GATECH.

Results. Figure 4 visualizes some qualitative results of semantic segmentation using V2V on GATECH. Table 1 presents the semantic segmentation accuracy on GATECH of V2V compared with all of the baselines. 2D-V2V, trained from scratch on GATECH, obtains 55.7% which is 11% below V2V-0. This result underscores the advantages of 3D convolution and 3D deconvolution over their 2D counterparts. Note also that V2V-0 is 9.3% below V2V. This predictably confirms the benefit of large-scale pre-training before fine-tuning. V2V also outperforms all bilinear upsampling baselines showing the advantages of using deconvolution over traditional upsampling. By visualizing the predictions of these methods, we can see that Conv5b-Up yields fairly accurate predictions but over-smoothed due to its big upsampling rate. On the other extreme, Conv3b-up produces finer predictions thanks to the lower upsampling rate, but its segments are noisy and fragmented because it relies on feature maps at layer 3, thus less deep and less complex than those used by Conv5b-Up. Finally, removing skip connections causes a drop in accuracy in both the case of fine-tuning (*V2V-remove-skip*) and training from scratch (*V2V-remove-skip-0*) compared to V2V and V2V-0 (1.3% and 5.9%, respectively).

5. Application II: Optical Flow Estimation

Dataset. Since there is no large-scale video dataset

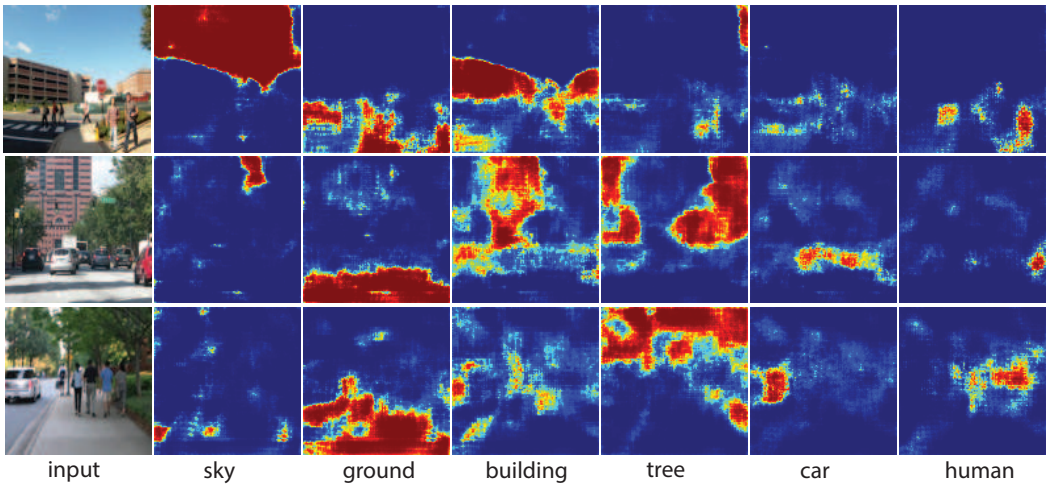

Figure 4. **Video Semantic Segmentation Results on GATECH**. The softmax prediction heat maps produced by V2V for different classes together with input frames. The last two classes are omitted due to their small populations. Best viewed in color.

Method	Train	Acc (%)
2D-V2V	from scratch	55.7
Conv3b+Up	fine-tune	69.7
Conv4b+Up	fine-tune	72.7
Conv5b+Up	fine-tune	72.1
V2V-remove-skip-0	from scratch	60.8
V2V-remove-skip	fine-tune	74.7
V2V-0	from scratch	66.7
V2V	fine-tune	**76.0**

Table 1. **Semantic segmentation on GATECH**. Comparison of several V2V variants.

available with optical flow ground truth, we fabricate our training data by applying an existing optical flow method on unlabeled video. Specifically, we use the OpenCV GPU implementation of Brox's method [4] to generate semi-truth data on both UCF101 [26] (public test split 1) and MPI-Sintel [5] (training set).

Training. We use the same V2V architecture with the number of channels at prediction layer set to $K = 2$. On both horizontal and vertical motion components, we use the Huber loss for regression as it works well with noisy data and outliers. Formally, this is given by

$$H(x) = \begin{cases} \frac{1}{2}x^2, & |x| \leq 1 \\ |x|, & \text{otherwise.} \end{cases} \quad (1)$$

To avoid numerical issues, the optical flow values are divided by a constant ($\alpha = 15$) so that most values fall in the range of $[-1, 1]$. We note that larger optical flows are still handled by the Huber loss. The V2V network takes as input clips of size $3 \times 16 \times 112 \times 112$ and produces clip outputs of size $2 \times 16 \times 112 \times 112$. The network is trained from scratch on UCF101 (using non-overlapping clips from each video) with a mini-batch size of 1. The initial learning rate is set to 10^{-8} and it is divided by 10 every 200K iterations (about 2 epochs). Training is stopped at 800K iterations. We note that, at inference time, we need to scale the predictions by $\alpha = 15$ to convert them back into the correct optical flow range.

Results. Figure 5 visualizes optical flow predicted by our V2V method and compares it with that computed by Brox's method for a few sample clips taken from the test split of UCF101. The V2V end point error (EPE) on the UCF101 test split 1 (treating Brox's optical flow as ground truth) is only 1.24. To better understand the performance of the learned V2V network, we further evaluate its performance on the training set of the MPI-Sintel dataset [5], which comes with ground truth data. This ground truth data is unbiased and allows us to assess performance independently from the accuracy of Brox's flow. Table 2 shows the EPE error obtained with two variants of our model: V2V stands for our network learned on the UCF101 Brox's flow, while finetuned-V2V denotes our model after fine-tuning V2V on Sintel ground truth data using 3-fold cross validation. The table also contains the best method on Sintel which is better than V2V by a good margin. Even though V2V is not state of the art, the results are very interesting: both V2V and finetuned-V2V perform better than their "teacher", the optical flow method that is used to generate the semi-truth training data. While the improvement is slim, it is important to highlight that V2V is *much faster* than Brox's algorithm (70x faster, see Table **??**). Thus, this experiment shows that the V2V network can be employed to learn efficient implementations of complex, hand-tuned voxel-prediction models.

To compare the runtime of V2V-Flow and Brox's method [4], we use the GPU implementation of Brox's method provided in OpenCV and extract optical flows for the whole UCF101 test split 1 by the two methods using a

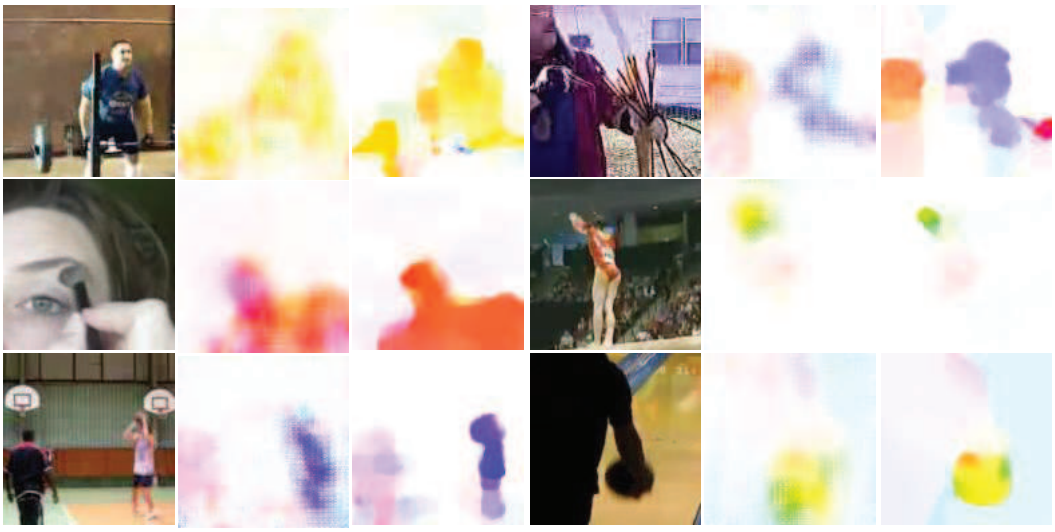

Figure 5. **Optical flow estimation on UCF101**. The output of V2V is qualitatively compared with Brox's optical flow for 6 sample clips from the UCF101 *test* split. For each example we show (from left to right): an input frame, V2V's predicted optical flow, and Brox's motion. Note that Brox's method is used to generate semi-truth data for training V2V. We see that on test videos V2V is able to predict flow of similar quality as that produced by Brox's algorithm. Best viewed in color.

Method	Brox	V2V	finetuned-V2V	FlowFields [1]
EPE	8.89	8.86	8.38	**5.81**

Table 2. **Optical flow results on Sintel**. V2V denotes our network learned from the UCF101 optical flow computed with Brox's method. The finetuned-V2V network is obtained by fine-tuning V2V on Sintel (test accuracy is measured in this case using 3-fold cross validation). Both versions of our network perform slightly better than Brox's algorithm and they allow computation of optical flow with a runtime speedup of 20 times compared to Brox's software.

NVIDIA Tesla K40. V2V-Flow is 70x faster than Brox's method. It can run at 91 fps while Brox's method operates at less than 2 fps (including I/O in both cases).

Observation. Unlike the case of video semantic segmentation application where V2V could be effectively fine-tuned from the initial C3D network, we empirically discovered that fine-tuning from C3D does not work for the case of optical flow estimation as in this case the training consistently converges to a bad local minimum. We further investigated this phenomenon by visualizing the learned filers of the first few convolutional layers for both the original C3D as well as the V2V learned from scratch on Brox's flow. The results are visible in Fig. 7. We see that the filters of the two networks look completely different. This is understandable, as C3D is trained to complete a high-level vision task, e.g. classifying sports. Thus the network learns a set of discriminative filters at the early layers. Some of these filters capture texture, some focus on discriminative motion patterns, while others respond to particular appearance or color cues. Instead, V2V is trained to perform a low-level vision task, e.g. predict motion directions. The Figure shows that the V2V filters are insensitive to color and texture as they focus exclusively on motion estimation. This explains why the pre-trained C3D model is a bad initialization to learn V2V for optical flow, but it is instead a good initialization for training V2V on semantic segmentation.

6. Application III: Video Coloring

Setup and Training. In this experiment we use UCF101 again in order to learn to color videos. We use the public training/test split 1 for the training and testing of our model. In this study we generate training data by converting the color videos to grayscale. V2V is fed with $C = 1$ input grayscale channel and it is optimized to predict the $K = 3$ ground truth original color channels. For this application we use the L2 regression loss as colors have no outliers. We use mini-batches of size 1. The learning rate is set initially to 10^{-8} and it is divided by 10 every 200K iterations. The training is stopped at 600K iterations. Similarly to the case of semantic segmentation, we compare our V2V with its 2D version baseline, 2D-V2V, both optimized on the same training set. Both models were learned from scratch.

We note that video coloring is challenging and ill-posed because there are some objects (e.g., clothes) that can be colored with any valid color. A reasonable expectation is that the coloring algorithm should learn to color correctly objects that typically occur only in one color. For example, the sky is usually blue (not always but often) and the grass is typically green. Thus, the model should learn to predict well the colors of such objects.

Results. To assess performance, we use as metric the average Euclidean distance between the predicted color and

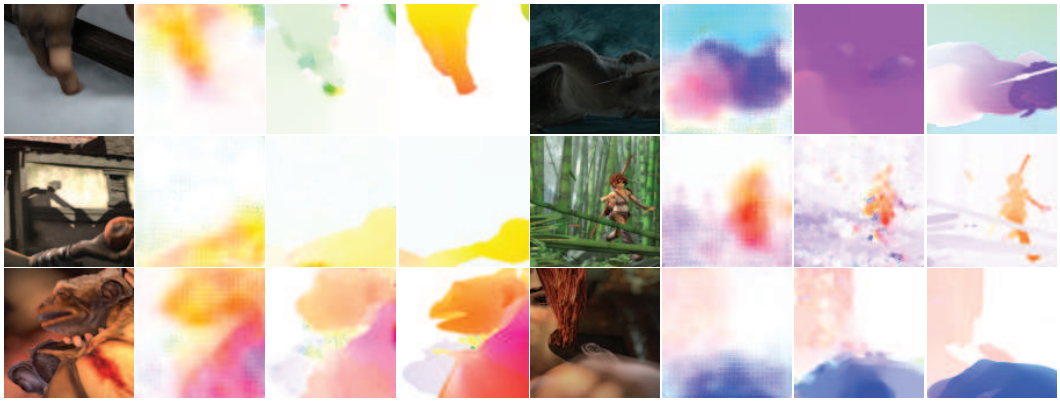

Figure 6. Visualizations of optical flow computed by the V2V network (trained on UCF101 without finetuning) for a few sample Sintel clips. For each example we show: input frame, V2V's predicted optical flow, Brox's flow, and ground truth. Best viewed in color.

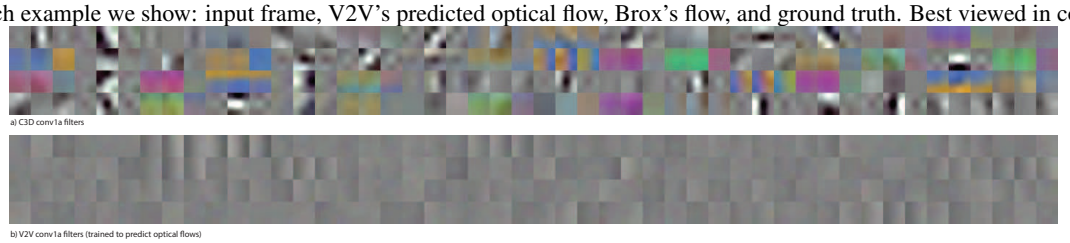

Figure 7. Visualization of `Conv1a` filters learned by C3D (top) and V2V (bottom). Note that C3D is trained to recognize actions (on Sport1M), while V2V is optimized to estimate optical flow (on UCF101). Each set shows the 64 learned filters at the `Conv1a` layer. Three consecutive square images on each row represent one filter (as kernel size is $3 \times 3 \times 3$). Each square image is upscaled to 30×30 pixels for better visualization. Best viewed in color. *GIF animation of these filters will be provided in the project website*.

the true color. Here each voxel color is represented in (r, g, b) and $r, g, b \in [0, 1]$. V2V has an average distance error (ADE) of 0.1375 whereas the 2D baseline has an ADE of 0.1495. Figure 8 presents some qualitative results of V2V on predicting voxel colors. It is interesting to see that the algorithm learns "common sense" colors such as the color of skin, sky, trees, river, sea, mountains, wood furniture, and the billiard table. For objects whose color is ambiguous, V2V applies very little coloring, leaving them almost in the original grayscale form. One can imagine extending V2V to have sparse inputs of color to make the problem well-posed for objects that can occur in various colors.

7. Conclusions

We have presented V2V, a novel architecture for voxel to voxel prediction using 3D convolutional networks. The proposed approach can be trained end to end from raw video input to predict target voxel labels without the need to preprocess or post-process the data. We have shown that the same architecture trained on three distinct application domains delivers reasonably good results on each of them. In the course of our experiments we have discovered that fine-tuning pre-trained models does not always help: for the case of optical flow estimation, learning from scratch is beneficial over fine-tuning from an action recognition model. We have also demonstrated that in absence of large-scale supervised data, V2V can be trained to reproduce the output of an existing hand-constructed voxel prediction model. Quite surprisingly, in our study the resulting learned model has accuracy superior (albeit only slightly) to its "teacher" method. We believe that bootstrapping the learning from an existing model can be an interesting avenue for future work and can be a successful strategy to learn efficient implementation of computationally expensive algorithm, such as in our case where V2V predicts optical flow with a 70x speedup over the original optical flow method that was used to generate training data. While we purposely avoided specializing the network to each task in order to emphasize the general applicability of the approach, we believe that further improvements can be obtained from more thorough architecture search.

Acknowledgment: we would like to thank colleagues at Facebook AI Research and Dartmouth Vision and Learning Group for valuable feedback and discussions.

References

[1] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *ICCV*, 2015. 6

[2] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015. 2

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, pages 1395–1402, 2005. 2

Figure 8. **Examples of video coloring with V2V on the test set of UCF101**. For each example we show (from left to right): a gray-scale input frame, the output frame colored by V2V, and the ground truth color frame. The V2V model is able to predict "common sense" colors such as the color of human skin, sky, woody furniture, river, sea, and mountain. Best viewed in color.

[4] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE TPAMI*, 33(3):500–513, 2011. 5

[5] D. Butler, J. Wulff, G. Stanley, and M. Black. A naturalistic open source movie for optical flow evaluation. In *CVPR*, 2012. 5

[6] L. Cao, Z. Liu, and T. Huang. Cross-dataset action detection. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 2

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2013. 1

[8] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proc. International Conference on Computer Vision*, pages 726–733, 2003. 2

[9] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 2

[10] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013. 1, 2

[12] N. Ikizler and D. Forsyth. Searching for complex human activities with no visual examples. *International Journal of Computer Vision*, 80(3):337–357, 2008. 2

[13] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biological inspired system for human action classification. In *Proc. International Conference on Computer Vision*, 2007. 2

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[16] I. Laptev and T. Lindeberg. Space-time interest points. In *ICCV*, 2003. 2

[17] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2, 3

[18] J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[19] J. Niebles and L. Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007. 2

[20] S. H. Raza, M. Grundmann, and I. Essa. Geometric context from video. In *CVPR*, 2013. 3

[21] S. Sadanand and J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 2

[22] H. Seo and P. Milanfar. Detection of human actions from a single example. In *Proc. International Conference on Computer Vision*, 2009. 2

[23] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1, 2

[24] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[26] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human action classes from videos in the wild. In *CRCV-TR-12-01*, 2012. 5

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2

[28] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2, 3, 4

[29] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013. 1, 2

[30] J. Yuan, Z. Liu, and Y. Wu. Discriminative video pattern search for efficient action detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2011. 2

[31] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3

[32] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 1