Predicting Future Optical Flow from Static Video Frames

Pol Rosello
Stanford University
prosello@stanford.edu

Abstract

Static frames from videos can provide information about the inherent motion in the scene. In particular, given an image of someone performing an action, humans can reasonably guess how pixels in the scene will move in subsequent frames. In this work, we use an unsupervised learning procedure to predict optical flow from static video frames by training an AlexNet convolutional neural network architecture on the HMDB-51 video dataset from scratch. We also train a fully-convolutional architecture which achieves similar performance with over an order-of-magnitude reduction in the number of network parameters. We extend our procedure to generate video predictions in raw pixel space through iterative flow prediction followed by spline interpolation, and suggest methods to improve the quality of our generative video model.

1. Introduction

Given a single frame from a video, human beings are good at predicting where objects in the frame will move to next. For example, in Figure 1, we can say with reasonable certainty that the torso of the woman is about to move up and to the left. Predicting actions and movement in videos has been a source of active research. A number of studies have attempted to predict the overall action or event that is about to take place [16, 17, 21] while others attempt to predict future frame representations using LSTMs [12]. The ability to anticipate the immediate future could provide a significant advantage in fields such as robotics, where an agent that interacts with the external world could plan around what it believes to be the most likely future sequence of events. One way to anticipate future actions is to attempt to predict future *optical flow*. Optical flow is a vector field which represents the apparent motion of each pixel between two frames. A method that reliably predicts optical flow in the immediate future could be helpful in predicting the paths of pedestrians in a busy intersection, in restoring video with missing frames, or in compressing video by deliberately dropping frames which can be reliably predicted.

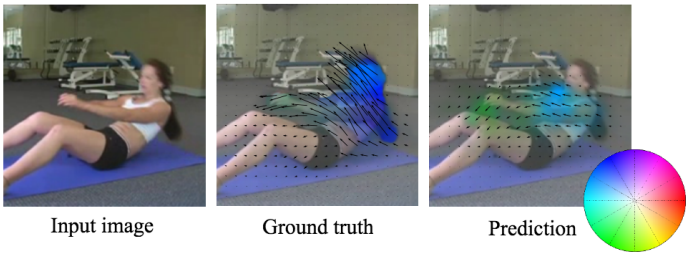

Figure 1: Extracted optical flow from a HMDB-51 frame, along with our model's prediction. We visualize the flow by superimposing the HSV color corresponding to the angle of the flow at each pixel.

Notably, the last two applications require the transformation of optical flow predictions back to pixel space.

In [11], random forests were used to predict dense optical flow from a single static image in the small KTH dataset [13]. Very recently, [18] used an AlexNet convolutional neural network in the style of [6] to predict optical flow from a static image, significantly outperforming previous results. In this work, we implement and train a similar architecture from scratch and compare its performance to a fully-convolutional neural network with no pooling or fully-connected layers. The input to each model is a static 200x200 RGB frame from a video and the output is a coarse 20x20 vector field representing the predicted optical flow that the frame will experience in the immediate future. Figure 1 shows an example of a real prediction by our model. Overall, both architectures generate successful predictions in scenes with unambiguous motions and compare well quantitatively with the state-of-the-art. We also find that the fully-convolutional neural network achieves similar performance as [18]'s approach with over 15 times fewer parameters. We extend our flow prediction model to warp the original image through spline interpolation. By iteratively feeding the warped image back into the network to generate a new optical flow prediction, we are able to generate predictions from single frames in the form of raw video. Generated videos are convincing for a few frames before the

predictions experience distortion. In the final section of this paper, we propose two extensions to our model that may improve our synthetic video predictions.

2. Dataset and Features

We train our models on the HMDB-51 dataset [7], which is originally intended for action recognition and is made up of natural footage of humans performing 51 different activities such as walking or boxing. Computing the optical flow between two frames is an active area of research. To obtain the ground truth optical flow between two frames, we use the publicly-available DeepFlow implementation in [20]. Training is therefore unsupervised since ground truth labels are automatically computed.

We extracted over 48,000 frames and their corresponding optical flows from more than 2,300 HMDB-51 videos. Only the videos which were labeled as being of medium or good quality and containing no camera movement were processed. This is because compression artifacts in poor quality video result in unreliable optical flow estimation. Similarly, shaky camera footage would add too much movement between frames to be able to extract a meaningful optical flow, and the computational complexity of stabilizing every video before extracting the flow was deemed too great for this project. Each frame in the dataset was resized so that its height was 200 pixels, and a center 200x200 crop was taken. The optical flow for each frame was then computed using DeepFlow, averaging the flow between the frame and the subsequent three frames to mitigate the effects of noise. The optical flow was extracted every 5 frames in the video, effectively downsampling the original dataset by a factor of 5. The resulting 48,000 frames and their corresponding optical flows were shuffled and split into training, validation, and test sets. The mean frame was subtracted from the dataset to center the data and the pixel values were scaled to between -1 and 1.

3. Methods

Given an input 200x200 RGB frame, our model must produce a two-dimensional vector field representing the predicted optical flow that the frame will experience in the immediate future. We predict a coarse 20x20 optical flow corresponding to the flow at pixels spaced 10 units apart in each dimension in the ground truth.

Following the lead of [8, 18, 19], we formulate the problem as a classification problem rather than a regression problem. Specifically, we first resize the computed flows to 20x20 flows and then cluster all flow vectors in the training set into a codebook of 40 representative clusters using the minibatch K-Means implementation from [10]. We use the centroids of the 40 clusters to generate 20x20 class labels for each frame in the training and validation sets, with each entry representing the cluster which most closely approximates the motion of that pixel. The results of resizing and classifying vectors are shown in Figure 2. The resulting flows are not significantly different from the ground truth flows, although there is some loss of precision.

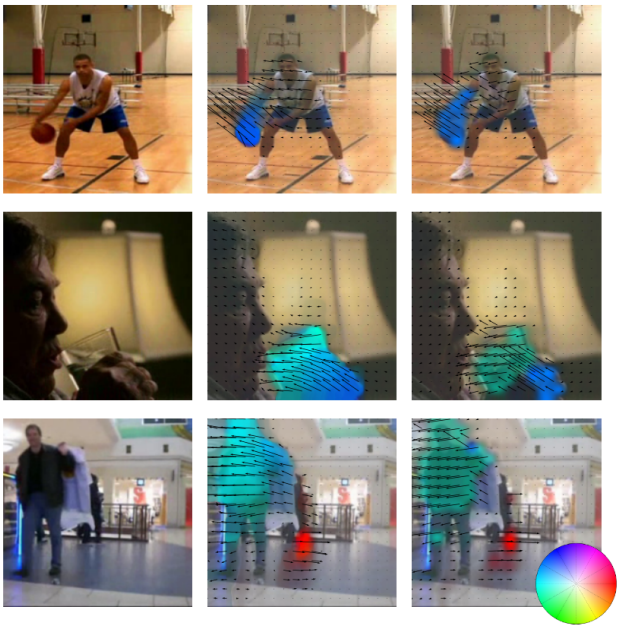

Figure 2: Effects of clustering optical flow vectors. The first column shows the original image, the second column shows the original optical flow and the third column shows the coarse, clustered optical flow.

We use the spatial softmax function from [18, 19] to quantify the loss of the network for a given set of parameters. Let I represent the input image and Y be the ground truth optical flow labels represented as quantized clusters. Then our spatial loss function $L(I, Y)$ is:

$$L(I,Y) = -\sum_{i=1}^{20\times 20}\sum_{c=1}^{40}(\mathbb{1}(y_i = c)\log F_{i,c}(I))$$

where $F_{i,c}(I)$ is the probability that the ith pixel will follow the optical flow of cluster c and $\mathbb{1}(x)$ is the indicator function.

At test-time, we can still predict flow vectors in continuous space by using the cluster scores of each pixel as weights in a weighted sum of the corresponding cluster centroids. More explicitly, if we denote the 40 cluster centroids as $\{\vec{m}_1, ..., \vec{m}_{40}\}$, we predict pixel i's optical flow $\vec{f}_i$ as:

$$\vec{f}_i = \sum_{c=1}^{40} F_{i,c}\vec{m}_c$$

Layer	Filters	Width	Stride	Pad	Dropout	Output
Input	-	-	-	-	-	200x200x3
ConvBN	96	11	4	2	-	49x49x96
MaxPool	-	3	2	0	-	24x24x96
ConvBN	256	5	1	2	-	24x24x256
MaxPool	-	3	2	0	-	11x11x256
ConvBN	384	3	1	1	-	11x11x384
ConvBN	384	3	1	1	-	11x11x384
ConvBN	256	3	1	1	-	11x11x256
MaxPool	-	3	2	0	-	5x5x256
FC	-	-	-	-	0.5	(5x5x256)x4096
FC	-	-	-	-	0.5	4096x4096
FC	-	-	-	-	-	4096x16000

Table 1: AlexNet-like CNN architecture.

Layer	Filters	Width	Stride	Pad	Dropout	Output
Input	-	-	-	-	-	200x200x3
ConvBN	96	7	4	2	-	50x50x96
ConvBN	128	3	2	1	-	25x25x128
ConvBN	256	4	2	0	-	22x22x256
ConvBND	256	3	1	1	0.3	22x22x256
ConvBND	512	3	1	0	0.3	20x20x512
ConvBND	512	3	1	1	0.3	20x20x512
ConvBN	512	3	1	1	-	20x20x512
ConvBN	40	1	1	0	-	20x20x40

Table 2: All-convolutional architecture.

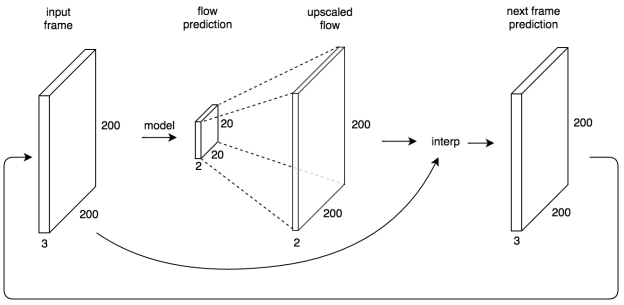

Figure 3: Generative video procedure.

We use two different network architectures to approach this problem: an AlexNet-like network from [18] and an all-convolutional network in the spirit of [15] with no pooling or fully-connected layers. The networks are summarized in Tables 1 and 2. ConvBN represents a convolutional layer followed by a spatial batch normalization layer and a ReLU nonlinearity. ConvBND represents the same with a dropout layer after spatial batch normalization. FC represents a fully-connected layer, and MaxPool represents a spatial max pooling layer.

The AlexNet network's 16,000-wide output is split into 400 groups of 40 (each group corresponding to the vector cluster scores for a single pixel) and the softmax loss for

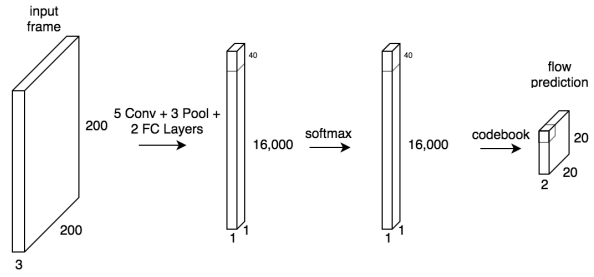

Figure 4: Flow prediction using the AlexNet architecture.

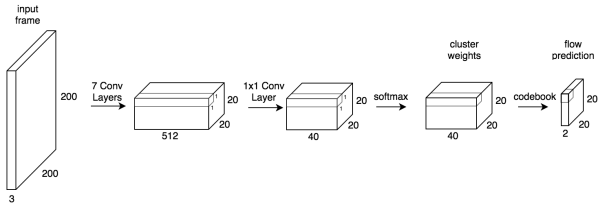

Figure 5: Flow prediction using the all-convolutional architecture.

each group is evaluated against its corresponding target label. The losses are then summed across all 400 groups as described above. This is illustrated in Figure 4.

In the all-convolutional network, pooling operations are achieved through convolutional layers. Instead of aggressive pooling to successively reduce the spatial dimensions of the activations to 1x1, the convolutional layers stop shrinking the spatial dimensions when the desired 20x20 output is reached. We can think of the next to last activation volume as containing a 512-deep feature vector for each output pixel prediction. The final 40 1x1 convolutions entail learning a 512-deep filter for each flow cluster that can transform the feature vectors of pixels into a class score for that filter's cluster. This is illustrated in Figure 5. The removal of the fully-connected layers drastically reduces the number of parameters in the network from about 112 million to 7.16 million, while both networks use similar amounts of memory.

We considered horizontal flips as a data augmentation technique but this would have entailed reclassifying every label in the training set – it is not sufficient to also horizontally flip the target label since this does not change the direction of the target flow vectors.

Finally, we use the test-time optical flow predictions of our models to generate video predictions in raw pixel space. We use the input image to generate an initial optical flow prediction. This flow prediction is then scaled up to 200x200, and the scaled flow is used to warp the original frame to generate a new frame. The warping is achieved

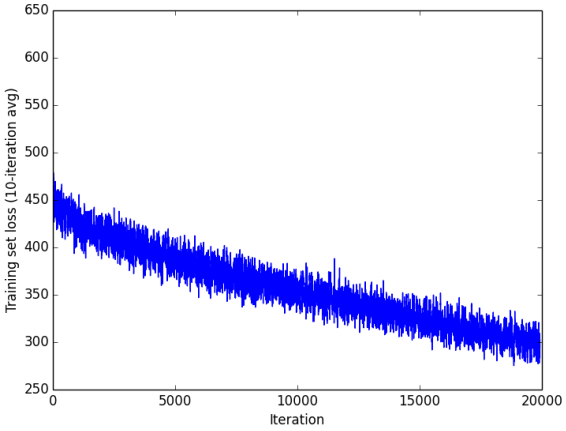

Figure 6: Training set loss of all-convolutional network over time. The AlexNet network loss decayed very similarly so we omit the plot.

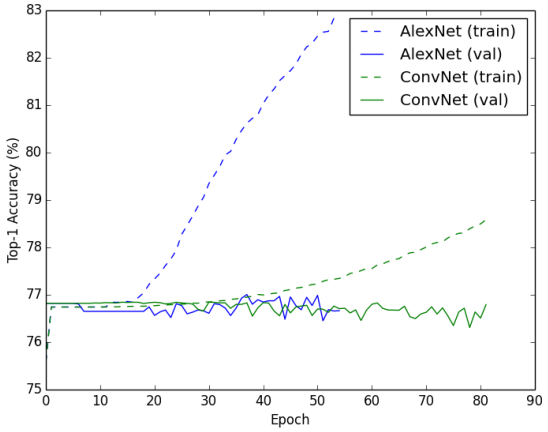

Figure 7: Average per-pixel top-1 accuracy of both models over time.

through interpolation. Denote the coordinates of pixel i as $\vec{x}_i$. We use the RGB value of pixel i as the value of the "pixel" at position $\vec{x}_i + \vec{f}_i$ in the new frame, and use bilinear spline interpolation to fill in the new frame's pixels at the regularly spaced pixel positions. The resulting frame is then used as a new input to the model, and the procedure is repeated. This model is summarized in Figure 3. Smoother video is obtained by generating multiple frames per optical flow prediction at intermediate steps. If we generate m frames for every optical flow prediction, the kth frame of this prediction is generated by warping as above at positions $\vec{x}_i + \frac{k}{m}\vec{f}_i$ for all i.

4. Experiments and Results

We implemented both networks using Torch7 [2], and trained them on AWS on a NVIDIA GRID K520 GPU with 4 GB of memory. We used OpenCV [1] for frame extraction and SciPy [4] for preprocessing. We used a minibatch size of 125 for both networks. The AlexNet network was trained using a learning rate of 0.0001. The all-convolutional network was trained using a learning rate of 0.001, which was reduced by a factor of 10 after 3 epochs of training. No L2 regularization was used on the AlexNet network, but a regularization of 1×10^{-5} was used on the all-convolutional network. We used the Adam update rule from [5] to optimize the loss function with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and initialized layer weights using the Xavier method from [3]. The AlexNet network was trained for 55 epochs and the all-convolutional network was trained for 81 epochs.

During training, we tracked the loss on the training set and used this to optimize hyperparameters. We also tracked

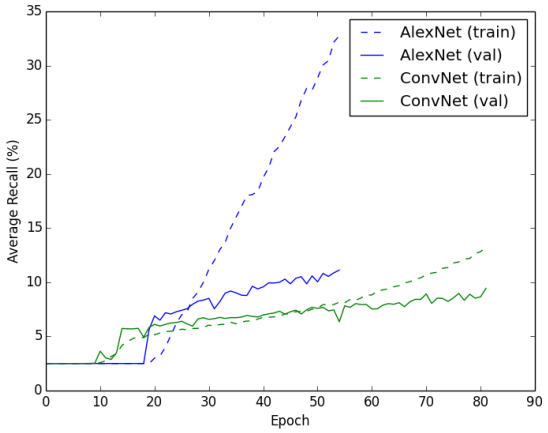

Figure 8: Average per-cluster recall of both models over time.

the top-1 accuracy over all pixels and the average recall across all clusters on the training and validation sets. We increased the regularization penalty and dropout frequency and restarted training from scratch whenever signs of heavy overfitting were exhibited. The results of training each network are summarized in Figures 6, 7 and 8. Note that the training set accuracy was computed during the learning process after each iteration (unlike the validation set accuracy which was computed at the end of each epoch) so during the first few epochs training set accuracy trails validation set accuracy.

Perhaps the biggest challenge to training networks on this dataset was the skewed nature of the labels. About

Model	Recall	Precision	Top-1	Top-5	Top-10
AlexNet	11.12%	15.88%	76.67%	91.9%	96.5%
All-Conv Net	9.44%	12.73%	75.89%	90.7%	96.3%

Table 3: Validation set statistics of final models. The precision and recall values refer to the average precision and recall over all cluster classes, respectively. Top-k refers to the percentage of pixels where the ground truth cluster class was among the k cluster classes with the highest scores.

75% of all flow vectors belong to the cluster corresponding to no optical flow motion. Accordingly, hyperparameter choices were hard to make because training and validation set accuracy tended to stay constant for the first few epochs, predicting no motion for every single pixel. Tracking the average recall across clusters was therefore instrumental to evaluating the performance of networks over time. Indeed, as seen in Figure 7 the validation set top-1 accuracy tended to stay flat or even slightly fall. However, the average recall increased, indicating that the network was learning to differentiate between types of motion. Finding an aggressive learning rate that broke through the initial phase of predicting the most frequent cluster (without leading to an exploding loss) was instrumental to training successful models.

The all-convolutional model did a much better job avoiding overfitting on the training set, as evidenced by how closely the validation set recall tracks the training set recall. This is likely due to the L2 regularization penalty we imposed, as well as the implicit regularization caused by the smaller number of parameters it uses. Although we ran out of budget, we suspect that if we had continued to train the all-convolutional network we would have continued to improve the recall rate beyond that of the AlexNet network, since the training set accuracy on the latter was already very high by the 55th epoch. The final accuracy statistics on the validation set for both networks are shown in Table 3. Although the Top-1 accuracy is not much better than guessing no motion for every pixel, the Top-5 accuracy is much higher. This is encouraging because at test-time we take into account the scores of all cluster classes, not just the top one. We also show the confusion matrix for the networks on the validation set in Figure 9. The matrix shows that it is common for the models to predict no motion as the highest-scoring class for pixels that do experience motion in the ground truth. Again, since our model takes into account all class scores, this is not a deal-breaker. We compute other quantitative evaluation metrics on the test set later on in this section.

We experimented with VGG-like architectures [14] but were unable to have the network learn beyond predicting no motion for all pixels. We suspect this is due to a narrower range of appropriate hyperparameter choices in deeper networks. Moreover, we were forced to use a smaller batch size of 25 due to the larger memory requirement of this network, so the increased noise in the loss signal was also likely to blame. We also experimented with weighting the loss function for each class with a factor inversely proportional to the frequency of the class, but found that while this provided considerably better recall rates in the initial stages of learning, the top-1 accuracy on the validation set was only slightly better than random.

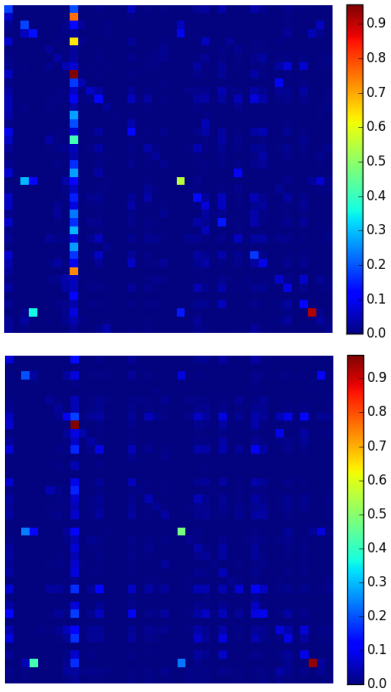

Figure 9: Confusion matrices for the AlexNet model (top) and the all-convolutional model (bottom) on the validation set. Rows are normalized to the number of pixels of that cluster type. The value of the square at row i, column j is equal to the fraction of pixels that move according to flow cluster i that were predicted by the model to move according to flow cluster j. The brighter diagonal represents pixels that were classified correctly. The vertical band at the ninth column corresponds to the no-movement cluster, so it represents pixels that were predicted to stay still but moved in the ground truth.

Qualitative results of the networks on the test set are shown in Figure 10. There was no significant qualitative advantage to one of the two networks. Strongest results were seen in frames with clear actions, such as punching, walking, or performing pullups or pushups. The networks tended to perform poorly on scenes with fine-grained or ambiguous motions such as smiling or talking, as seen in Figure 10(e). Many frames which were not correctly predicted

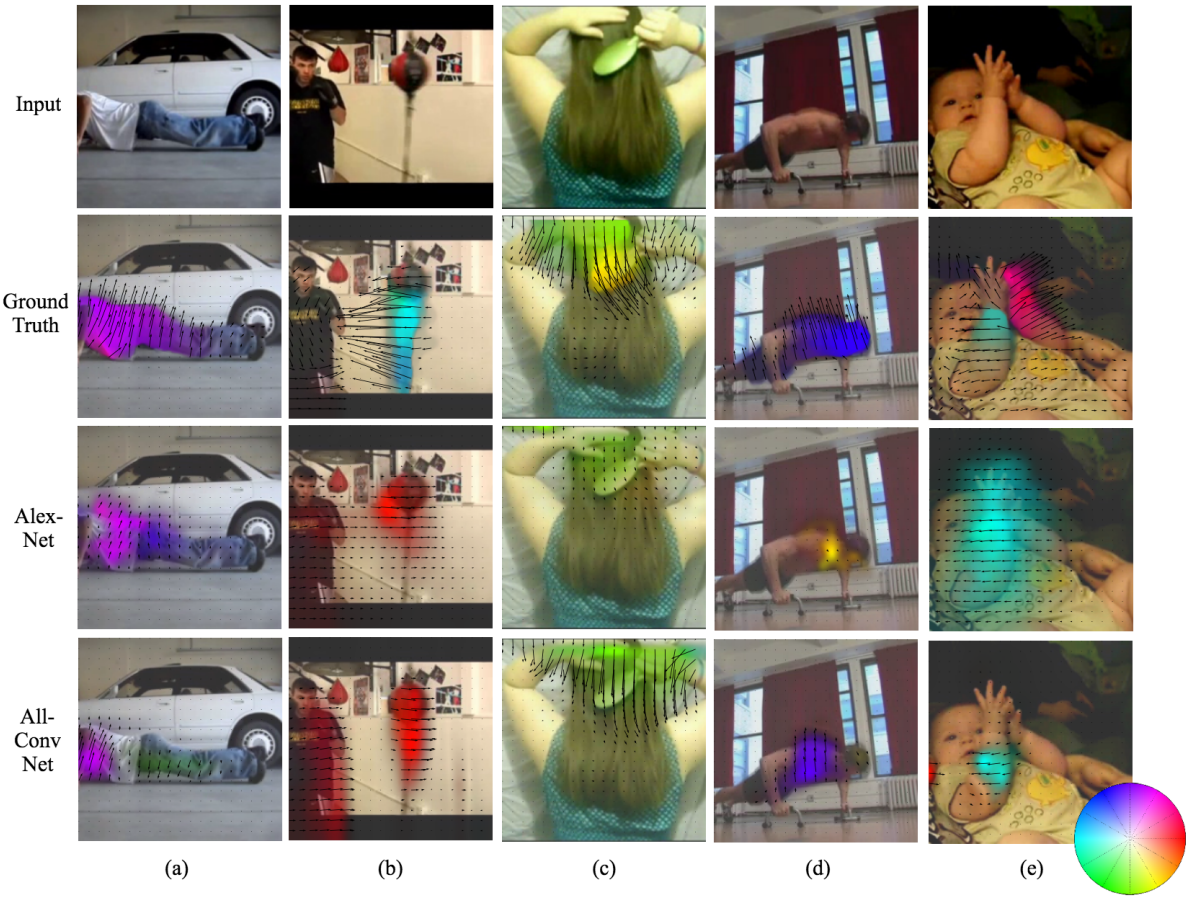

Figure 10: Predicted optical flows for five different HMDB-51 frames.

by the networks were similarly unpredictable by humans. Some mispredictions were consistent with the motion ambiguity in the scene. For example, in Figure 10(b), both networks predicted rightward motion of the boxer and the bag but the ground truth was opposite. However, based on the input frame, it is hard to say in which of the two directions the motion will proceed.

The top-1 pixel accuracy for each pixel is plotted in Figure 11 for the all-convolutional model. As expected, the accuracy is higher around the edges of the image where motion is less likely to take place. The center of the image is the most unpredictable region since most actions are centered in the frame. We omit this plot for the AlexNet model since it is nearly identical.

We evaluate the performance of the models against the ground truth optical flow using a number of performance metrics. Given vectors $\{u_1, u_2, ..., u_N\}$ in the predicted optical flow and ground truth vectors $\{v_1, v_2, .., v_N\}$, the average Euclidean distance between vectors in the ground truth and the predicted optical flow $\frac{1}{N}\sum_{i=1}^{N}||u_i - v_i||$ gives a general estimate of how far off the predictions are. The average cosine similarity between vectors $\frac{1}{N}\sum_{i=1}^{N}\frac{u_i^T v_i}{\|u_i\|\|v_i\|}$ gives an estimate of how well the model predicts the direction of movement. Finally, the average orientation similar-

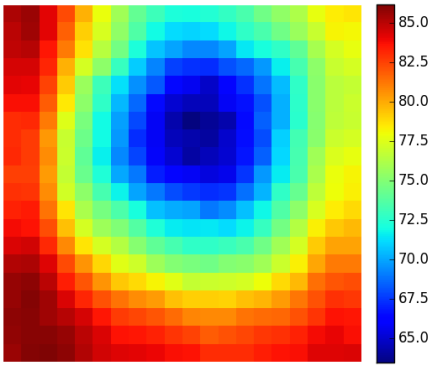

Figure 11: Average top-1 cluster accuracy per pixel for the AlexNet model.

Model	L2	Dir	Ori	Top-5	Top-10	#params
[11]	1.23	0.000	0.461	-	-	-
[18]	1.21	0.016	0.636	90.2	95.9	112M
All-Conv Net	1.12	0.036	0.664	90.7	96.3	7.16M
AlexNet	1.10	0.096	0.676	91.9	96.5	112M

Table 4: Evaluation results. L2 refers to the average Euclidean distance, Dir refers to the average cosine similarity, Ori refers to the average orientation similarity, and Top-k refers to the percentage of predictions where the ground truth cluster was among the k clusters with the highest scores. The lower the L2 distance, the better. The higher the other metrics, the better. The results of [11] on HMDB-51 are quoted from [18]. The results of [18] on HMDB-51 are taken from their model trained on both HMDB-51 and UCF101.

ity between vectors $\frac{1}{N}\sum_{i=1}^{N}\frac{|u_i^T v_i|}{\|u_i\|\|v_i\|}$ gives an estimate of how parallel the predicted vectors are to the ground truth. This is especially important in cases where the exact direction of movement is ambiguous, but one of two directions can be given. For example, in Figure 1, the person in the frame may be on the way down instead of up, in which case the direction of each vector will be flipped. Likewise, in Figure 10(b) it is hard to say from the original frame whether the person will move right or left.

The results of the quantitative evaluation of both models are summarized in Table 4. Note that while we outperform both previous works, we likely gain from overfitting on the 51 action types encountered in HMDB-51, while [18] is probably a more general model that could outperform our approaches on outside datasets. Nevertheless, our models perform competitively with the state of the art, and the all-convolutional network performs well with a fraction of the parameters. We think we could have significantly improved the metrics of both networks with more training time since we were forced to cut training short before the loss plateaued due to a limited computational budget.

Generative video samples are available in this report's submission as well as at `https://youtu.be/CPUnVs3X48c`. We used 5 successive flow predictions per video and generated 5 frames per prediction. The generated videos are convincing and capture the underlying motion of the scene, although distortion becomes too great in the last few frames. We believe the coarse nature of the flow predictions is a large source of error. In the next section, we propose an alternative architecture that we believe could significantly outperform our current ones with regard to generative video.

5. Conclusion and Future Work

In this work, we explored two different convolutional neural network architectures to predict optical flow given a static video frame. We analyzed the predictions of our models qualitatively and compared their quantitative performance to past approaches to the problem. We used iterative predictions alongside spline interpolation to generate short video predictions in pixel space. Overall, the networks predict optical flow remarkably well in frames with clear actions, and make sensible mistakes.

An extension of our models that we would have liked to explore given more time would be to generate pixel-wise predictions rather than coarse ones. Because we formulated our problem as a classification problem rather than a regression problem, many of the techniques used in pixel-wise segmentation models such as [9]'s – namely fractionally-strided convolution to upscale the activation volumes – can apply. The process to generate new frames would be the same as in Figure 3, except the upscaling step would be learned rather than simple interpolation. In this way, we would be able to generate dense 200x200 flow predictions, greatly reducing artifacts in warping that arise due to the sparse nature of our current predictions.

A second extension would entail learning sequences of flows using LSTMs. Using a warped image as the only input to make a second flow prediction does not explicitly take into account the temporal structure of past flow patterns. For example, if the optical flow in a frame of someone walking points to the right, it is very likely that the optical flow at the next frame will also point to the right. [18] briefly experiments with learning sequences of flows and achieves some success by clustering optical flow frames into 1000 clusters and using a series of fully connected layers to predict the optical flow frame at each timestep, with each layer having access to the states of the past layers. We suspect the combination of these two extensions could generate much higher quality synthesized video.

References

[1] G. Bradski. The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[4] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2016-03-13].

[5] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[7] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

[8] L. Ladicky, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014.

[9] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] S. L. Pintea, J. C. Gemert, and A. W. Smeulders. Déjà vu: Motion prediction in static images. In *ECCV*, 2014.

[12] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

[13] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[16] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*, 2015.

[17] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating the future by watching unlabeled video. *arXiv preprint arXiv:1504.08023*, 2015.

[18] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *ICCV*, 2015.

[19] X. Wang, D. F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. *arXiv preprint arXiv:1411.4958*, 2014.

[20] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013.

[21] J. Yuen and A. Torralba. A data-driven approach for event prediction. In *ECCV*, 2010.