Activity recognition and Intensity estimation using Event Cameras

A THESIS

submitted by

ARPAN PAUL

in partial fulfilment of the requirements
for the award of the degree of

of

Dual Degree in Electical Engineering (B.Tech and M.Tech)

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

May 2019

THESIS CERTIFICATE

This is to certify that the thesis titled **Activity recognition and Intensity estimation using Event Cameras**, submitted by **Arpan Paul**, to the Indian Institute of Technology, Madras, for the award of the degree of **Dual Degree**, is a bonafide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Kaushik Mitra
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 10th May 2019

ACKNOWLEDGEMENTS

I am very grateful to my guide, Dr. Kaushik Mitra, Computational Photography Laboratory, IIT Madras, for providing me with all the guidance to complete the project. His guidance and expertise were really valuable and played a key role in completion of this project. I want to thank him especially for granting me the freedom and opportunity to work on the topic of my interest, to fail and learn, and work at my pace. Without his encouragement and support, this project would not have been completed.

I would like to thank my PhD mentor Prasan Shedligeri for helping and contributing towards the progress of the project. His technical expertise, patience and an easy to work with attitude were extremely crucial. He helped me lift my spirits when the project stopped progressing.

I would also like to thank my lab mate Ashish Upadhyay (CH14B070) and Chinni Chaitanya for being their with me during the project. Ashish helped me in discussing new approaches in the deep learning and machine learning projects. I was working on an extension to the paper published by Chinni. He helped me in ramping up with the topics and was there with me throughout.

I would also like to thank the Computational Photography Laboratory, IIT Madras for allowing me to pursue my project in a disturbance free environment.

Regards,
Arpan Paul
EE14B113

ABSTRACT

KEYWORDS: Event Cameras ; DVS; Deep Learning; Intensity Estimation; Activity Recognition

Event cameras are an emerging class of cameras that only report pixel-wise intensity changes (called "events") over a large dynamic range with extreme high temporal resolution. They offer other several advantages such as low latency and low power consumption in contrast to conventional cameras. In this thesis, we try to solve two vision problems - Human Activity Recognition (HAR) and Video Intensity estimation using event cameras.

We have proposed a deep learning solution to the HAR problem as an extension to the previous work using SVM classifier (Baby *et al.*, 2018). We have used outdoor activity data (UCF11 DVS video dataset) for our experiment. Event streams windowed into videos are trained by a Deep Network (CNN + LSTM) classifier.

In the intensity estimation problem, we have tried intensity reconstruction of an intermediate video frame between two consecutive video frames of normal cameras (30fps) using events between them. The framework comprises of an event interpolation system followed by denoising autoencoder. We were able to reconstruct videos at 60fps.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Frame based conventional cameras operate at a fixed frame rate irrespective of the intensity values of the frames, whether they are changed or not. This often leads to issues like data redundant frames, high memory usage etc. Event cameras on the other hand, respond asynchronously to changes in pixel illumination and offer high temporal resolution over large dynamic range.

There are a myriad of applications that need high frame rate as well as low power consumption at the same time such as surveillance, remote sensing, and security where the cameras may be battery operated (or solar power driven) and therefore have stringent limitations on power consumption.

These two individual features (high frame rate of event cameras and rich pixel information of frame based cameras) provide complementary information. Fusing image frames with the output of event cameras offers the opportunity to create an image state infused with high-temporal-resolution. We aim to use this fusion to solve typical vision problems like Activity/Gesture recognition and Intensity estimation in our paper.

DVS is intrinsically suitable for gesture/activity recognition since it does not record any static information about the scene. Thus, we can avoid the overhead of preprocessing algorithms such as background subtraction and contour extraction used in conventional image processing. This part of the paper is an extension to the previous paper by my lab. The previous algorithm involved Bag of Words and Motion maps framework which attempted to solve this 11 class classification problem by SVM classifier. This paper attempted to obtain better classification results using deep learning architecture. We have trained the UCF11 DVS video dataset on CNN features followed by LSTM units.

In the second part of the paper, We present a intensity estimation formulation of an intermediate frame between two consecutive video frames using event frames. We also use a denoising autoencoder to smoothen the image reconstruction.

CHAPTER 2

Human Activity Recognition

2.1 Previous Work

Our solution is an extension to (Baby *et al.*, 2018). explore the feasibility of using DVS for Human Activity Recognition (HAR).

They propose to use feature maps known as Motion Maps. These are basically different 2-D projection slices, x - y,x - t and y - t, obtained by averaging each left out dimension respectively. DVS event streams are converted into a video by accumulating events over a time window and Motion Maps are created out of them. They fuse Motion Maps with Motion Boundary Histogram (MBH) (Lo and Tsoi, 2014) to extract features. SURF features and MBH features using dense trajectory are also extracted. Bag of features encoding from both these descriptors are combined and given to linear SVM classifier (one-vs-all).

This classifier gives good performance on the benchmark DVS dataset as well as on a real DVS gesture dataset collected by our lab.

However, the current solution has a lot of complex feature engineering. Some of the feature extracts like BoVW may not perform fairly on complex data. Morever, current framework may not be robust to unknown datasets. There are a of parameters (especially in SVM) that need to fine tuning based on underlying data. Deep Learning architecture would provide a black box solution to the problem and are much easier to train.

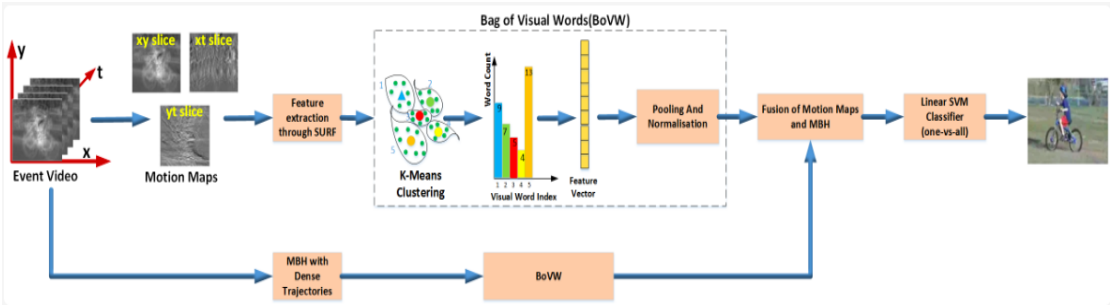


Figure 2.1: HAR

2.2 Proposed Solution

DVS video dataset is generated by acquiring event streams into a frame over a time window. Being asynchronous, every video in the dataset has a different number of event frames. So, we have sampled every input video to a fixed number of frames (5) to obtain uniformity.

We have proposed a convolutional neural network framework inspired by (Ng *et al.*, 2015) which processes these event frames and outputs a feature vector. The idea behind CNN is to learn the spatial event features across the sampled frames in the video.

This 2 layered 2D CNN takes frames of size 96x120 as input (Bicubic interpolation to reduce the frame size and subsequently, the number of parameters). These frame are then processed by square convolutional layers of size 5 each followed by max-pooling and batch normalization. Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift). BatchNorm also helps each layer of a network to learn by itself a little bit more independently of other layers.

Finally each sampled frame produces a 5760 feature vector. Now, since the spatial features are learned, these vectors are passed into a Recurrent network to learn the temporal dependencies.

Standard recurrent networks have trouble learning over long sequences due to the problem of vanishing and exploding gradients . In contrast, the Long Short Term Memory (LSTM) uses memory cells to store, modify, and access internal state, allowing it to better discover long-range temporal relationships.

We use a deep LSTM architecture in which the output from one LSTM layer is input for the next layer. We experimented with various numbers of layers and memory

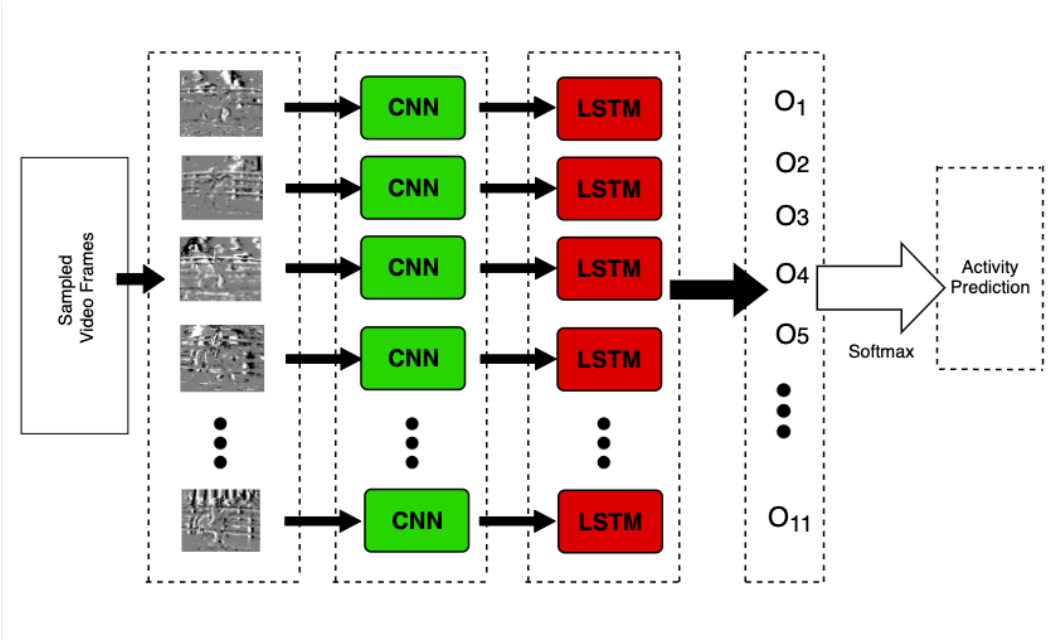


Figure 2.2: Proposed Architecture

cells, and chose to use two stacked LSTM layers, with 1200 memory cells. Following the LSTM layers, a Softmax classifier makes a prediction at every frame. In order to combine LSTM frame level predictions into a single video-level prediction, we returned the prediction at the last time step T.

2.2.1 Training

We have trained our framework on UCF11 DVS dataset. It comprises of 11 outdoor activities (Basketball, Biking, Walking with Dog, Tennis Swing etc). The learning rate has been initially set to 5×10^{-4} scheduled with exponential decay at the rate $\gamma = 0.5$ at every alternate epoch.

Instead of classical stochastic gradient we have used adam optimization (Diederik P. Kingma, 2015) to update our parameters. They offer several advantages like invariant to diagonal rescale of the gradients and are computationally efficient.

2.3 Results

We have trained the model on NVidia GeForce 10GB GPU and the total training time is approximately 10 hours. We observed that there is nice convergence in the classification loss.

However, classification accuracy between classes vary quite much. The average classification accuracy obtained is around **75-80%**

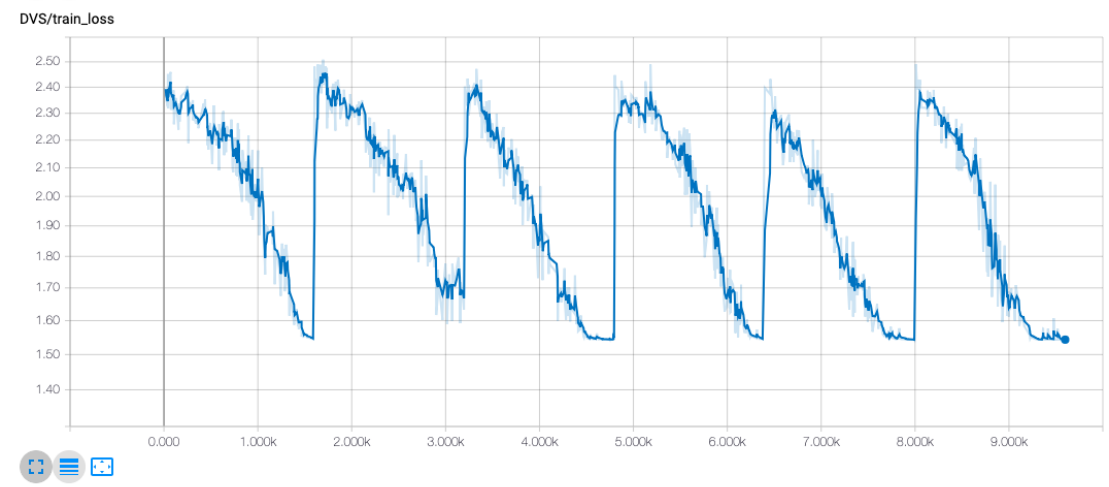


Figure 2.3: Training Epochs

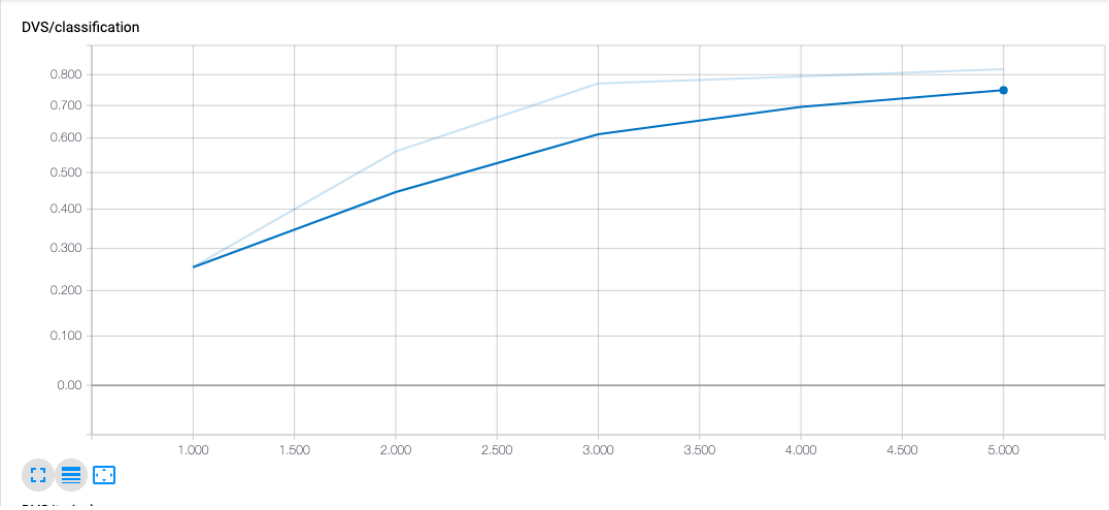


Figure 2.4: Classification Accuracy

2.4 Inferences

We have achieved satisfactory accuracy when compared to the previous work in DVS activity recognition. However, there are many areas which are unexplored and can be improved upon.

Firstly, the manual sampling is one area which could be done better. Random sampling/Video slicing may be impractical and unrealistic. We could have tried something called segmented sampling as shown in this work (Wang *et al.*, 2017).

The frames are densely recorded in the videos, the content changes relatively. Segment based sampling provides a new temporal structure modeling. This strategy is essentially a kind of sparse and global sampling. Concerning the property of spareness, only a small number of sparsely sampled snippets would be used to model the temporal structures in a human action. Therefore, no matter how long the action videos will last for, our sampled snippets would always roughly cover the visual content of whole video, enabling us to model the long-range temporal structure throughout the entire video.

In our architecture, ConvNet captures only the spatial information and outputs a feature vector. We fail to capture the motion feature dynamics. As suggested in (Ma *et al.*, 2017), they exploit the temporal information using two stream Inception style ConvNets using raw optical flow images as well. They explore the correlations between spatial and temporal streams by using two different proposed fusion frameworks. They focus on two models that can be used to process temporal data: Temporal Segment LSTMs (TS-LSTM) which leverage recurrent networks and convolution over temporally-constructed feature matrices (Temporal-ConvNet)

CHAPTER 3

Video Intensity Estimation

3.1 Previous Work

Conventional videos have a frame rate of 30fps which captures whole intensity image frames and are hence temporally sparse. However, in applications of high speed video they become impractical. Event cameras provide asynchronous events with high dynamic range and temporal resolution in the order of μs.

In this work (Scheerlinck *et al.*, 2018), the authors have suggested an asynchronous and efficient complementary filter which continuously fuses image frames and events into a single high-temporal-resolution image state. They have implemented a simple iterative algorithm to estimate image intensity frame $\hat{L_o}$ which is a solution to complementary filter ODE. The proposed complementary flter architecture combines a high-pass version of $L^E(p,t)$ (Events) with a low-pass version of $L^F(p,t)$ to reconstruct an (approximate) all-pass version of $L(p,t)$.

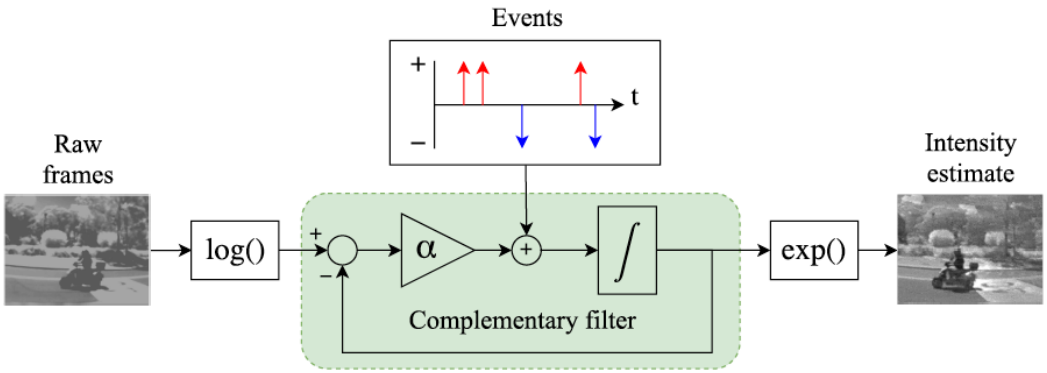


Figure 3.1: Complementary Filter

In the next work (Reinbacher *et al.*, 2016), they propose a variational model that accurately models the behaviour of event cameras, enabling reconstruction of intensity images with arbitrary frame rate in real-time. They formulate the intensity image

reconstruction problem as the solution of the optimisation problem

$$u^n = \underset{u \in C^1(\Omega, R+)}{\operatorname{argmin}} [E(u) = D(u, f^n) + R(u)]$$

where $D(u, f^n)$ is a data term that models the camera noise and $R(u)$ is a regularisation term that enforces some smoothness in the solution.

Figure 3.2: Sample results from the method. The image **a** shows the raw events and **b** is the result of our reconstruction. The time since the last event has happened for each pixel is depicted as a surface in **c** with the positive and negative events shown in green and red respectively

We have implemented the complementary filter in Python on our dataset. The second work is also a similar algorithm.

After implementing complementary filter we observed two shortcomings. Since the value of contrast threshold wasn't defined properly (even the authors had taken a random constant). Value of the threshold needs to be adaptive to the corresponding event frame. As a result, it produced a lot of noisy frames. Once the reconstruction is done, we also observe a discontinuity when moving from one image frame to other. We observe a sudden jump in frame transition.

3.2 Proposed solution

Videos have a much lesser frame rate when compare to high stream events. W In our approach, we have tried to estimate an intermediate intensity frame between two consecutive frames in a conventional video.

Our framework consists of two parts intensity reconstruction followed by Denoising

autoencoder.

We know that an event is fired when the log-intensity of a certain pixel varies beyond a threshold.The event firing process can be mathematically expressed as,

$$e_t = \begin{cases} 1 & \theta > \epsilon_p \\ -1 & \theta < -\epsilon_p \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $\theta = \log(I_t + b) - \log(I_t + b)$. Note that $e_t = 0$ indicates that the event pixel does not fire.

We use this formulation to reconstruct raw image frames. At every successive frame, the pixel intensity v is compared to a reset value r from the previous frame to decide how many events happened in that frame. The initial reset value r_0 for a pixel is the intensity of that pixel in the original image (v_0), and is updated as described in Algorithm 1.Repeating this process for every frame gives us pixel estimate at every raw image frame. Our approach is to iterate from both the sides of the consecutive frames (iterate backwards from right frame and iterate forward for left frame) to reach midway from both the sides.

Once we get two raw images in the middle, they are passed through Noise2Noise autoencoder.

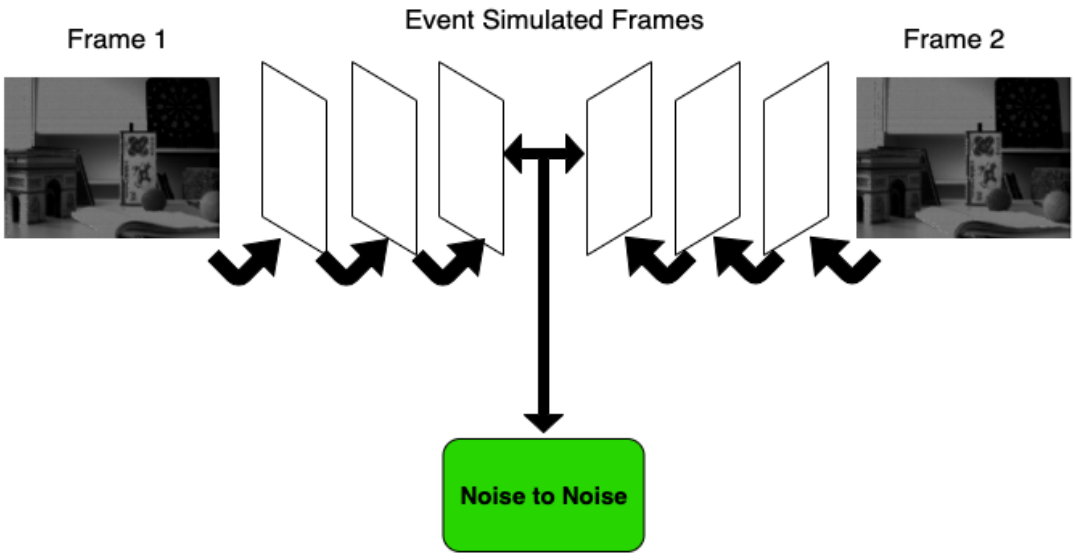


Figure 3.3: Frame intensity reconstruction

Noise2Noise decoder(Lehtinen *et al.*, 2018) helps us to learn to restore images by only looking at corrupted examples. The form of the typical training task for a set of input-target pairs (x_i, y_i), where the network function $f_\theta(x)$ is parameterized by θ:

$$\underset{\theta}{\operatorname{argmin}} \mathbb{E}_{\{x,y\}}\{L(f_\theta, y)\}$$

The property of L_2 minimization is that on expectation, the estimate remains unchanged if we replace the targets with random numbers whose expectations match the targets.

This implies that, in principle, we can corrupt the training targets of a neural network with zero-mean noise without changing what the network learns. So the formulation becomes empirical risk minimization task,

$$\underset{\theta}{\operatorname{argmin}} \sum_i \{L(f_\theta(\hat{x_i}), \hat{y_i})\}$$

The authors have solved this objective function using modifies Unet.

3.2.1 Modified U Network

Table 3.1 shows the structure of the U-network (Ronneberger *et al.*, 2015). For all basic noise removal experiments with RGB images, the number of input and output channels were n = m = 3.

The network weights were initialized following (He *et al.*, 2015). No batch normalization, dropout or other regularization techniques were used. Training was done using ADAM (Diederik P. Kingma, 2015) with parameter values $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$.

Learning rate was kept at a constant value during training except for a brief ramp-down period at where it was smoothly brought to zero. Learning rate of 0.001 was used for the experiments. Minibatch size of 4 was used in the experiments.

Let us compute the expected error in L_2 norm minimization task when corrupted targets $\{\hat{y}_i\}_{i=1}^N$ are used in place of the clean targets $\{y_i\}_{i=1}^N$, with N a finite number. Let y_i be arbitrary random variables, such that $\mathbb{E}_{\hat{y}_i} = y_i$. As usual, the point of least

NAME	N_{out}	FUNCTION
input	n	
enc_conv0	48	Convolution 3×3
enc_conv1	48	Convolution 3×3
pool1	48	Maxpool 2×2
enc_conv2	48	Convolution 3×3
pool2	48	Maxpool 2×2
enc_conv3	48	Convolution 3×3
pool3	48	Maxpool 2×2
enc_conv4	48	Convolution 3×3
pool4	48	Maxpool 2×2
enc_conv5	48	Convolution 3×3
pool5	48	Maxpool 2×2
enc_conv6	48	Convolution 3×3
upsample5	48	Upsample 2×2
concat5	96	Concatenate output of pool4
dec_conv5a	96	Convolution 3×3
dec_conv5b	96	Convolution 3×3
upsample4	96	Upsample 2×2
concat4	144	Concatenate output of pool3
dec_conv4a	96	Convolution 3×3
dec_conv4b	96	Convolution 3×3
upsample3	96	Upsample 2×2
concat3	144	Concatenate output of pool2
dec_conv3a	96	Convolution 3×3
dec_conv3b	96	Convolution 3×3
upsample2	96	Upsample 2×2
concat2	144	Concatenate output of pool1
dec_conv2a	96	Convolution 3×3
dec_conv2b	96	Convolution 3×3
upsample1	96	Upsample 2×2
concat1	$96+n$	Concatenate input
dec_conv1a	64	Convolution 3×3
dec_conv1b	32	Convolution 3×3
dev_conv1c	m	Convolution 3×3, linear act.

Table 3.1: Network architecture used in our experiments. N_{out} denotes the number of output feature maps for each layer. Number of network input channels n and output channels m depend on the experiment. All convolutions use padding mode "same", and except for the last layer are followed by leaky ReLU activation function with $\alpha = 0.1$. Other layers have linear activation. Upsampling is nearest-neighbor.

deviation is found at the respective mean. The expected squared difference between these means across realizations of the noise is then:

$$\begin{aligned}
&\mathbb{E}_{\hat{y}}[\frac{1}{N}\sum_i y_i - \frac{1}{N}\sum_i \hat{y}_i]^2 \\
=&\frac{1}{N^2}\left[\mathbb{E}_{\hat{y}}(\sum_i y_i)^2 - 2\mathbb{E}_{\hat{y}}(\sum_i y_i)(\sum_i \hat{y}_i) + \mathbb{E}_{\hat{y}}(\sum_i \hat{y}_i)^2\right] \\
=&\frac{1}{N^2}\mathrm{Var}(\sum_i \hat{y}_i) \\
=&\frac{1}{N}\left[\frac{1}{N}\sum_i\sum_j \mathrm{Cov}(\hat{y}_i, \hat{y}_j)\right]
\end{aligned} \tag{3.2}$$

In the intermediate steps, we have used $\mathbb{E}_{\hat{y}}\sum_i \hat{y}_i = \sum_i y_i$ and basic properties of (co)variance. If the corruptions are mutually uncorrelated, the last row simplifies to

$$\frac{1}{N}\left[\frac{1}{N}\sum_i \mathrm{Var}(y_i)\right] \tag{3.3}$$

In either case, the variance of the estimate is the average (co)variance of the corruptions, divided by the number of samples N. Therefore, the error approaches zero as the number of samples grows. The estimate is unbiased in the sense that it is correct on expectation, even with a finite amount of data.

The above derivation assumes scalar target variables. When $\hat{y}_i$ are images, N is to be taken as the total number of scalars in the images, i.e., #images $\times$ #pixels/image $\times$ #color channels.

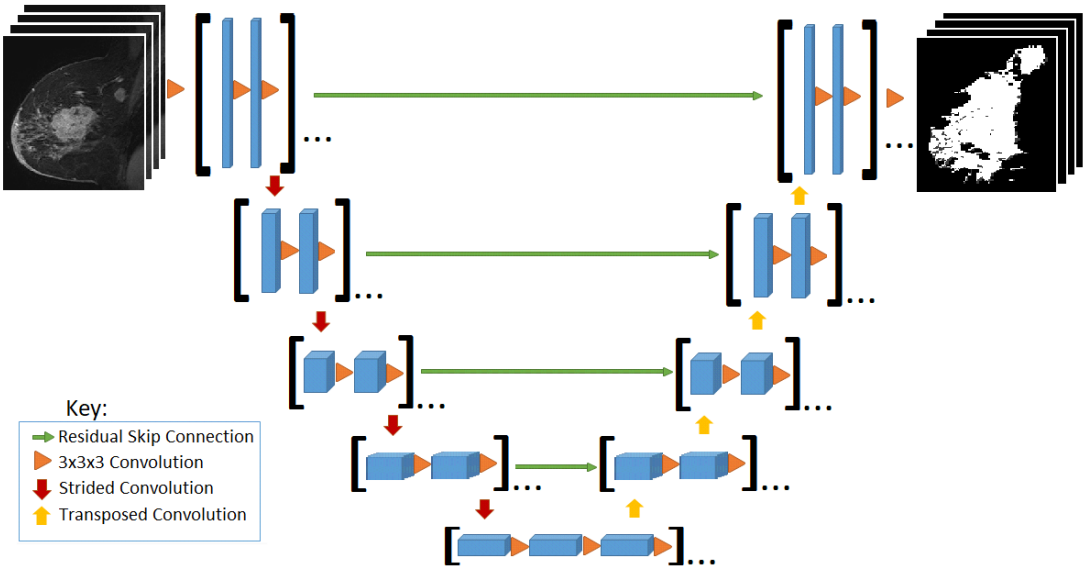


Figure 3.4: UNet

3.3 Results

We have trained our model from DAVIS events and video dataset as shown in the work (Mueggler *et al.*, 2016). The total training time comes to be around 4 hours.

In denoised results, we have stated PSNR with respect to the final frame. Same in the case of final reconstruction results.

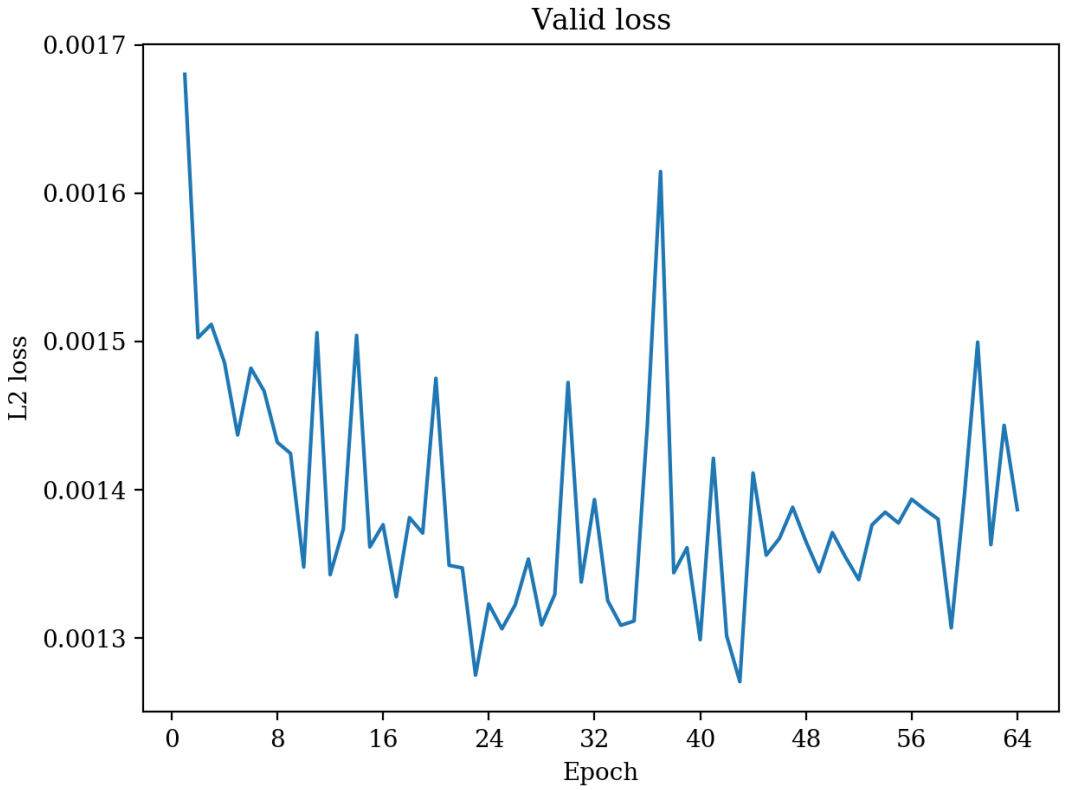
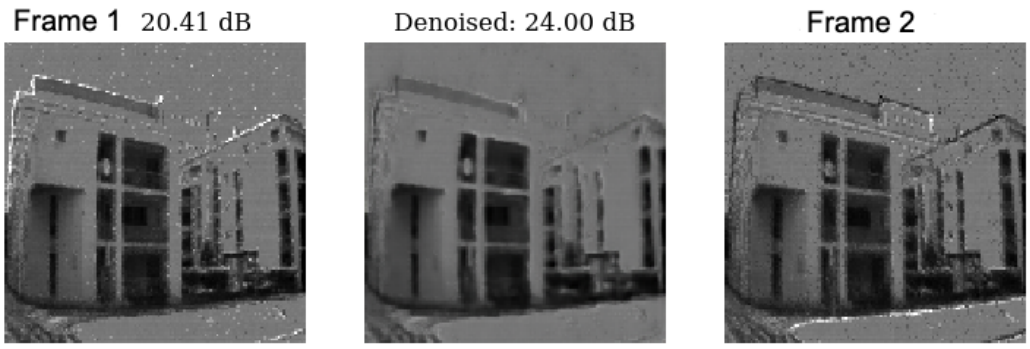
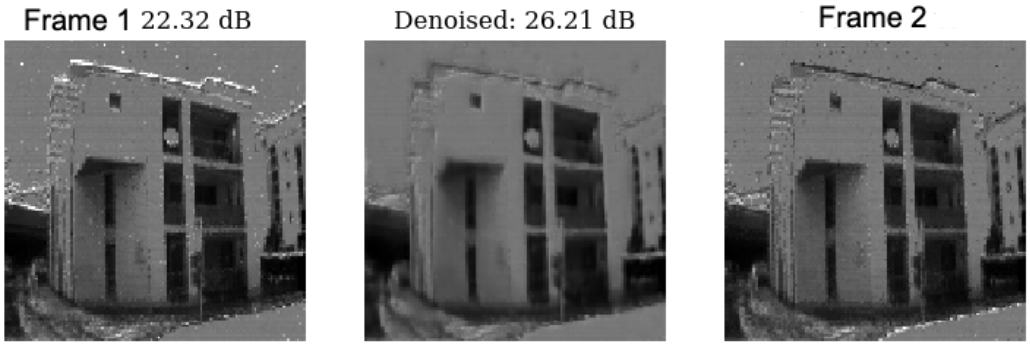


Figure 3.5: Validation loss of UNet

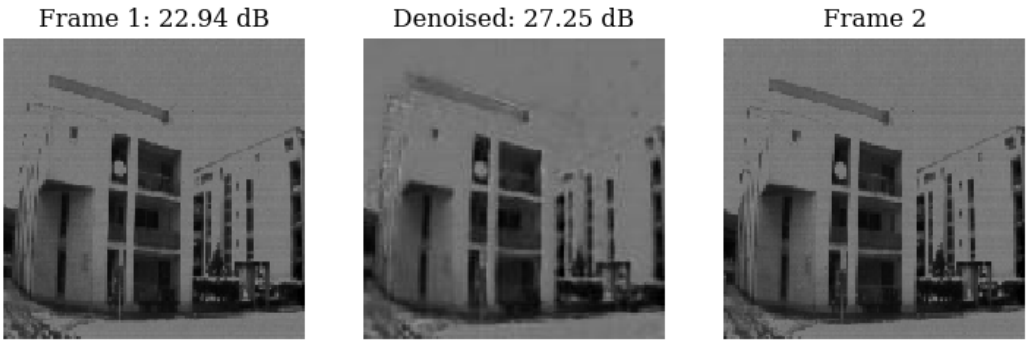


(a)



(b)

Figure 3.6: UNet denoising



(a)

(b)

Figure 3.7: Final Reconstruction

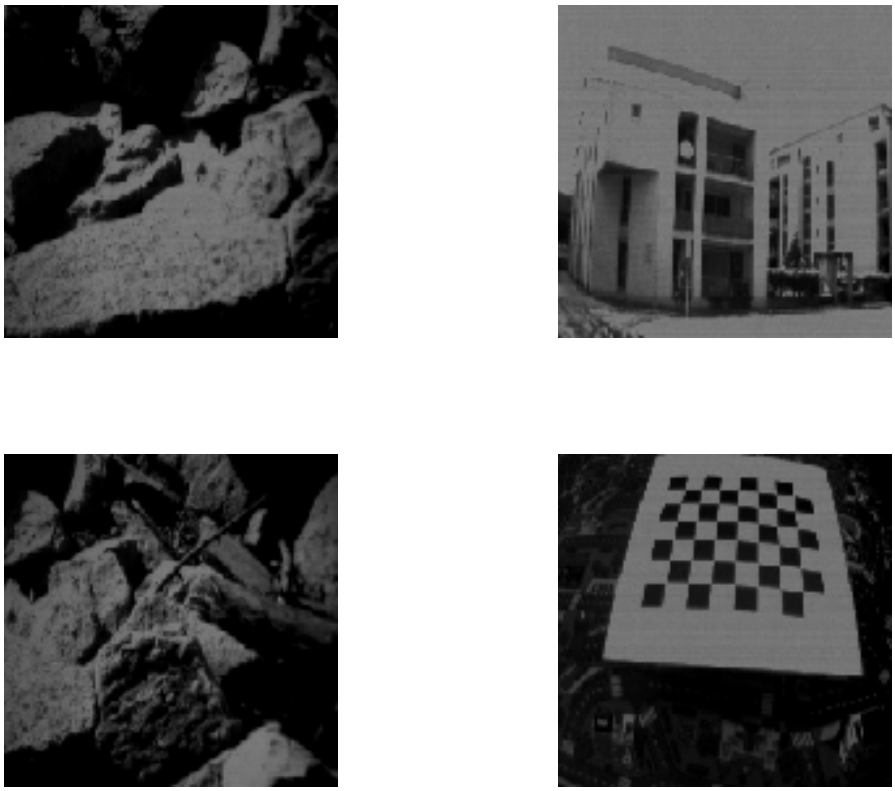


Figure 3.8: Intensity Gifs at 60fps

3.4 Inference

Even though we have achieved a good PSNR on the reconstruction there were quite a few areas that we could have improved upon.

One work we are really inspired with is (Wang *et al.*, 2019). They show us to differentiable model to approximate the intensity reconstruction process, which enables stochastic gradient descent optimization using automatic differentiation

The event pixel responds to intensity changes in log scale with very low time latency ($\sim 1\mu s$). That is, as long as the log-intensity increases above a threshold ϵ_p or decreases below a threshold ϵ_n, the pixel will report a binary signal (positive or negative). The event pixels preserve differential information, but the absolute gradient of the intensity is lost. Thus, the data inference problem is ill-posed and results in many solutions

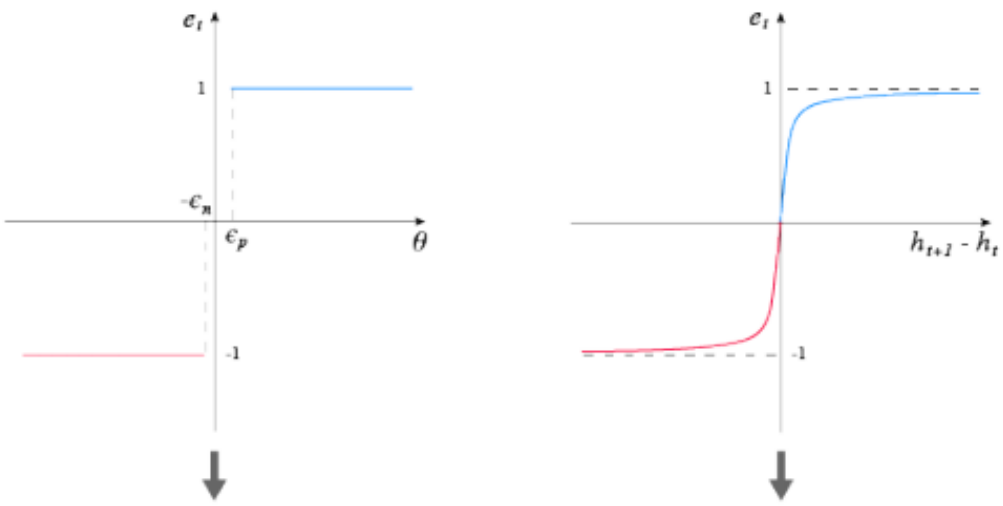


Figure 3.9: Event model

In our case, we have treated events as step function which makes it a discontinuous problem. The authors have modeled it as a continuous tangent function.

They show that the reconstruction is performed by minimizing a weighted combination of loss functions, including the pixel loss ($\mathcal{L}_{pix}$) and the sparsity loss ($\mathcal{L}_{TV}$). The objective function can be formed as,

$$\hat{\mathcal{H}} = \underset{\mathcal{H}}{\operatorname{argmin}}\, \mathcal{L}_{pix}(H, \mathcal{F}, \mathcal{E}) + \mathcal{L}_{TV}(H)$$

Pixel loss. The pixel loss includes per-pixel difference loss in ℓ_1 form with respect to the intensity and event pixels, i.e.,

$$\mathcal{L}_{pix}(\mathcal{H}, \mathcal{F}, \mathcal{E}) = \mathbb{E}_{fpix}\left[\|\mathcal{F} - \mathcal{A}(\mathcal{H})\|_1\right] + \lambda_e \mathbb{E}_{epix}\left[\|\mathcal{E} - \mathcal{B}(\mathcal{H})\|_1\right],$$

over the entire available data range. $\mathcal{F}$ and $\mathcal{E}$ denote respectively the ground truth of the frame and event data, and $\mathbb{E}_x$ represents expectation with respect to the observed pixels/events.

Sparsity loss. We employ total variation (TV) sparsity in the spatial and temporal dimensions of the high-res tensor $\mathcal{H}$. The TV sparsity loss is defined as:

$$\mathcal{L}_{TV}(\mathcal{H}) = \lambda_{xy}\mathbb{E}_{hpix}\left[\left\|\dot{\mathcal{H}}_{xy}\right\|_1\right] + \lambda_t \mathbb{E}_{hpix}[\left\|\dot{\mathcal{H}}_t\right\|_1],$$

where $\dot{\mathcal{H}}_{xy} = \frac{\partial \mathcal{H}}{\partial x} + \frac{\partial \mathcal{H}}{\partial y}$ and $\dot{\mathcal{H}}_t = \frac{\partial \mathcal{H}}{\partial t}$.

By forming the problem into a differentiable physical model, one is enabled to recover the latent high-res tensors using gradient descent based methods in large scale. They optimized the equations using stochastic gradient descent with TensorFlow implementation.

We could also have tried residual scheme for denoising the images instead of playing directly with the noisy images. We could have gone for similar residual networks like ResNet or DCNN similar to the implementation in the work (Wang *et al.*, 2019). They employ the CNN to find the residual $\mathcal{R}$ such that,

$$\mathcal{R} = \hat{\mathcal{H}} - \mathcal{H}_g,$$

In our case, we have finally achieved a frame rate of 60fps (2x 30). However, if we iteratively go on, such as denoising frames further at every 25% distance, we could achieve a higher frame rate. In the current approach, we approach halfway by multiplicative iteration from both sides. We could further extrapolate by storing raw image from at every $\frac{1}{4}^{th}$ distance. Subsequently, instead of just a pair of images, we'll have multiple pairs of images to denoise. Thus, higher framerate could be achieved.

CHAPTER 4

Conclusion

4.1 Human Activity Recognition

We have obtained a 75-80% accuracy on the UCF11 video activity dataset shot by an event camera. We have proposed an end to end robust deep network solution (CNN plus LSTM for spatio temporal feature learning). Unlike the previous method, this new framework doesn't need manual feature engineering.
However, there are sectors we could have faired better. Image slicing for sampling is a novice way for undersampling. As suggested above, there are better ways like segmentation sampling. Two streams ConvNets could also be explored.

4.2 Video Intensity reconstruction

We were able to reconstruct the intermediate intensity frame reconstruction between two consecutive image frames of a video. We used event streams/frames to interpolate raw image frames. The noisy image frames interpolated from both sides are then denoised using a Noise2Noise autoencoder. We were able to achieve fairly good results with good PSNR measurements.

We had modeled events as a step function and as a result, we directly superimposed multiplicatively. However, as other works show, modelling them using differential approximations (tanh function), we could obtain better results by solving the objective function. Our system fails to perform well on highly dynamic videos. Better denoisers like residual schemes could help our cause.

REFERENCES

1. **Baby, S. A.**, **B. Vinod**, **C. Chinni**, and **K. Mitra** (2018). Dynamic vision sensors for human activity recognition. *CoRR*, **abs/1803.04667**. URL `http://arxiv.org/abs/1803.04667`.

2. **Diederik P. Kingma, J. B.** (2015). Adam: A method for stochastic optimization. *Conference for Learning Representations*.

3. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun** (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, **abs/1502.01852**. URL `http://arxiv.org/abs/1502.01852`.

4. **Lehtinen, J.**, **J. Munkberg**, **J. Hasselgren**, **S. Laine**, **T. Karras**, **M. Aittala**, and **T. Aila** (2018). Noise2noise: Learning image restoration without clean data. *CoRR*, **abs/1803.04189**. URL `http://arxiv.org/abs/1803.04189`.

5. **Lo, S.-L.** and **A. C. Tsoi**, Motion boundary trajectory for human action recognition. *In ACCV Workshops*. 2014.

6. **Ma, C.**, **M. Chen**, **Z. Kira**, and **G. AlRegib** (2017). TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. *CoRR*, **abs/1703.10667**. URL `http://arxiv.org/abs/1703.10667`.

7. **Mueggler, E.**, **H. Rebecq**, **G. Gallego**, **T. Delbrück**, and **D. Scaramuzza** (2016). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *CoRR*, **abs/1610.08336**. URL `http://arxiv.org/abs/1610.08336`.

8. **Ng, J. Y.**, **M. J. Hausknecht**, **S. Vijayanarasimhan**, **O. Vinyals**, **R. Monga**, and **G. Toderici** (2015). Beyond short snippets: Deep networks for video classification. *CoRR*, **abs/1503.08909**. URL `http://arxiv.org/abs/1503.08909`.

9. **Reinbacher, C.**, **G. Graber**, and **T. Pock** (2016). Real-time intensity-image reconstruction for event cameras using manifold regularisation. *CoRR*, **abs/1607.06283**. URL `http://arxiv.org/abs/1607.06283`.

10. **Ronneberger, O.**, **P. Fischer**, and **T. Brox** (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, **abs/1505.04597**. URL `http://arxiv.org/abs/1505.04597`.

11. **Scheerlinck, C.**, **N. Barnes**, and **R. E. Mahony** (2018). Continuous-time intensity estimation using event cameras. *CoRR*, **abs/1811.00386**. URL `http://arxiv.org/abs/1811.00386`.

12. **Wang, L.**, **Y. Xiong**, **Z. Wang**, **Y. Qiao**, **D. Lin**, **X. Tang**, and **L. V. Gool** (2017). Temporal segment networks for action recognition in videos. *CoRR*, **abs/1705.02953**. URL `http://arxiv.org/abs/1705.02953`.

13. **Wang, Z.**, **W. Jiang**, **A. K. Katsaggelos**, and **O. Cossairt** (2019). Event-driven video frame synthesis. *CoRR*, **abs/1902.09680**. URL http://arxiv.org/abs/1902.09680.