7 Supplementary Material

7.1 Summary of Notation

Dimensions	
D	Physical dimensionality of point cloud
Q	Number of input channels
P	Number of output channels
S	Size of input cloud
S'	Size of output cloud
E	Number of edges
M	Number of basis functions
Sets	
$\mathcal{X} \subset \mathbb{R}^D$	Input cloud coordinates
$\mathcal{X}' \subset \mathbb{R}^D$	Output cloud coordinates
$\mathcal{N}_i \subseteq \mathcal{X}$	Set of inputs in neighborhood of x'_i
Tensors	
$x_j \in \mathcal{X}$	j^{th} input coordinate
$x'_i \in \mathcal{X}'$	i^{th} output coordinate
$\Delta x_{ij} \in \mathbb{R}^D$	Edge vector: $x'_i - x_j$, $x_j \in \mathcal{N}_i$
$f_j \in \mathbb{R}$	Input feature associated with x_j
$f'_i \in \mathbb{R}$	Output feature associates with x'_i
$f \in \mathbb{R}^S$	Single-channel input feature for input cloud $\mathcal{X}$
$f' \in \mathbb{R}^{S'}$	Single-channel output feature for output clouse $\mathcal{X}'$
$F \in \mathbb{R}^{S \times Q}$	Multi-channel input features
$F' \in \mathbb{R}^{S' \times P}$	Multi-channel output feature
$\Theta^{(m)} \in \mathbb{R}^{Q \times P}$	kernel parameters associated with m^{th} basis fn
$N^{(m)} \in \mathbb{R}^{S' \times S}$	Neighborhood matrix

Table 6: Summary of notation.

7.2 Additional Point Cloud Network Details

Pseudo-code for Iterative Farthest Point (IFP) variants and rejection sampling are given in Algorithms 1 through refalg:approx-ifp-rej.

Select differences between rejection sampling and random sampling are given in Figure 3.

A diagram of our large point cloud network is given in Figure 4.

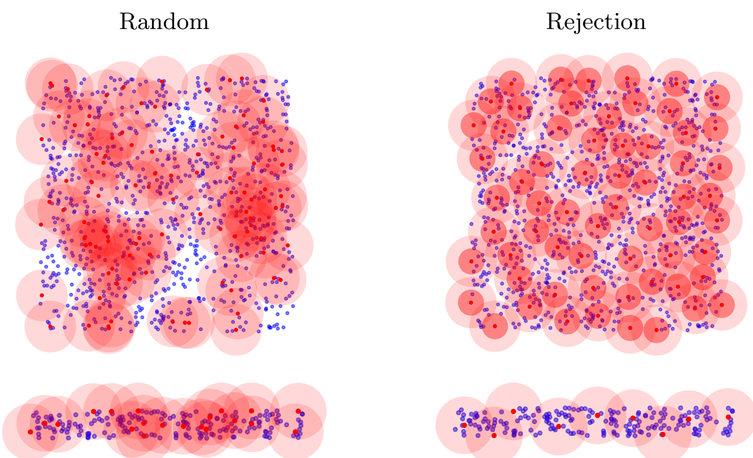


Fig. 3: Output cloud (red dots) resulting from different sampling schemes applied to input clouds (blue) and the corresponding neighborhoods (light red circles). From the top left image, we can see random sampling can result in some regions being under-sampled. This is particularly problematic for networks with subsequent up-sampling, where some blue points have no red points in their own neighborhood. The number of sampled points is not fixed for rejection sampling, so significantly less points will be sampled from pointy surfaces (bottom). By construction, none of the dark red circles (top right, half base radius) overlap, so the total number of possible sample points is limited by ball packing theorems.

Algorithm 1: IFP

Inputs: $\mathcal{X}$ input point cloud
S' output size
Result: $\mathcal{X}'$: sampled points
$\mathcal{X}' \leftarrow []$;
$S \leftarrow \text{size}(\mathcal{X})$;
$d_{\min} \leftarrow \infty \times \text{ones}(S)$;
for i *in range*(S') **do**
  $j \leftarrow \text{argmin}(d_{\min})$;
  $\mathcal{X}'.\text{append}(x_j)$;
  $d_{\min} \leftarrow \min(d_{min}, d(\mathcal{X}, x_j))$;
end

Algorithm 2: Approx. IFP

Inputs: $\mathcal{X}$ input point cloud
S' output size
$\mathcal{N}(\cdot)$ neighborhood fn
Q priority queue
Result: $\mathcal{X}'$: sampled points
$\mathcal{X}' \leftarrow []$;
for j *in range*(S') **do**
  $x'_i \leftarrow Q.\text{pop}()$;
  $\mathcal{X}'.\text{append}(x'_i)$;
  for x_n *in* $\mathcal{N}(x'_i)$ **do**
    $Q.\text{update}(x_n,\ d(x_n, x'_i))$
  end
end

Algorithm 3: Approx. IFP (without rej.)

Inputs: $\mathcal{X}$ input point cloud
S' output size
$\mathcal{N}(\cdot)$ neighborhood fn
Result: $\mathcal{X}'$: sampled points
$S \leftarrow \text{size}(\mathcal{X})$;
$Q \leftarrow \text{Priority Queue}(\infty \times \text{ones}(S), \mathcal{X})$;
$\mathcal{X}' \leftarrow \text{Approx. IFP}(\mathcal{X}, S', \mathcal{N}, Q)$

Algorithm 4: Rejection Sampling

Inputs: $\mathcal{X}$ input point cloud
$\mathcal{N}(\cdot)$ neighborhood fn
Result:
$\mathcal{X}'$: sampled points
d_{min}: distance from each input
point to closest output point
$\mathcal{X}' \leftarrow []$;
$S \leftarrow \text{size}(\mathcal{X})$;
$d_{\min} \leftarrow \infty \times \text{ones}(S)$;
visited $\leftarrow$ False $\times$ ones(S);
for x'_i *in* $\mathcal{X}$ **do**
  if *visited*$[i]$ **then**
    continue;
  end
  $\mathcal{X}'.\text{append}(x'_i)$;
  $\mathcal{N}_i \leftarrow \mathcal{N}(x'_i)$;
  for x_j *in* $\mathcal{N}_i$ **do**
    visited$[j] \leftarrow$ True;
    $d_{\min}[j] \leftarrow \min(d_{min}[j], d(x'_i, x_j))$;
  end
end

Algorithm 5: Approx. IFP (with rej.)

Inputs: $\mathcal{X}$ input point cloud
S' output size
$\mathcal{N}(\cdot)$ neighborhood fn
Result: $\mathcal{X}'$: sampled points
$\mathcal{X}'_0, d_{\min} \leftarrow \text{Rejection Sampling}(\mathcal{X}, S', \mathcal{N})$;

$Q \leftarrow \text{Priority Queue}(d_{\min}, \mathcal{X})$;
$S'_1 \leftarrow S' - \text{size}(\mathcal{X}'_0)$;
$\mathcal{X}'_1 \leftarrow \text{Approx. IFP}(\mathcal{X}, S'_1, \mathcal{N}, Q)$;
$\mathcal{X}' \leftarrow \text{concatenate}(\mathcal{X}'_0, \mathcal{X}'_1)$;

7.3 Additional Event Stream Network Details

The Leaky Integrate and Fire (LIF) algorithm we used is given in Algorithm 6.

Algorithm 6: Leaky Integrate and Fire (LIF)

Inputs: X input grid shape
t times for input events, sorted ascending
x coordinates for input events, same order as t
$\mathcal{N}(\cdot)$ spatial neighborhood fn giving coordinates of receptive field
$\tilde{t}$ decay time
v_{thresh} spike threshold
v_{reset} reset potential
Result: $t_{\text{out}}, x_{\text{out}}$: time and coordinates of output stream.

```
x_out ← [];
t_out ← [];
V ← zeros(X);
T ← zeros(X);
S ← size(x);
for i in range(S) do
    t_i ← t[i];
    x_i ← x[i];
    n ← size(N(x_i));
    for x_j in N(x_i) do
        v ← V[x_j] exp(−(t_i − T[x_j])/t̃) + 1/n;
        if v > v_thresh then
            v ← v_reset;
            t_out.append(t_i);
            x_out.append(x_j);
        end
        V[x_j] ← v;
        T[x_j] ← t_i;
    end
end
```

We down-sampled examples from the two highest-resolution datasets – N-Caltech101 and ASL-DVS – by a factor of 2 in each dimension. We performed basic data augmentation involving small rotations ($-22.5°$ to $22.5°$), time/polarity reversal for all datasets except ASL-DVS and left-right flips for CIFAR-10-DVS and N-Caltech101. No data augmentation was applied to ASL-DVS. We computed neighborhood information for N-MNIST online and offline with 8 augmented repeats for MNIST-DVS, CIFAR10-DVS and NCaltech101-DVS.

For the small number of examples with more than 300,000 events we took the first 300,000. Apart from this infrequent cropping, we use all events in all examples.

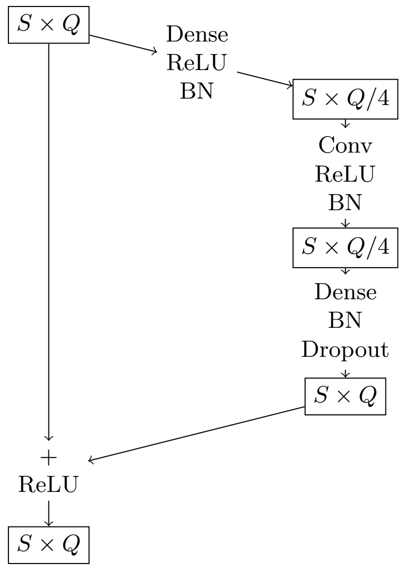
All models were trained with Adam optimizer, initial learning rate $1e-3$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-7$. We trained our ASL-DVS model for 100 epochs with a fixed learning rate. For all others, we decay the learning rate by a factor of 5 after the training accuracy fails to increase for 10 epochs, and run until learning ceases as a result of several such decays.

Dataset summary statistics and select model hyper-parameters parameters given in Table 7.

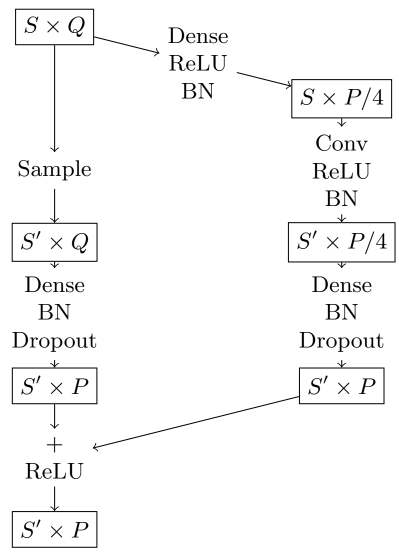
A diagram of the model used for CIFAR10-DVS is given in Figure 5.

Dataset	N-MNIST	MNIST-DVS	CIFAR10-DVS	N-Caltech101	ASL-DVS
# Classes	10	10	10	101	24
Resolution	34×34	128×128	128×128	174×234	180×240
# Train examples	60,000	9,000	9,000	7,838	80,640
Median # events	4,196	70,613	203,301	104,904	17,078
Mean # events	4,171	73,704	204,979	115,382	28,120
Max # events	8,183	151,124	422,550	428,595	470,435
Batch Size	32	32	16	8	8
Spike Threshold, v_{thresh}	1.5	1.5	1.6	1.25	1.0
Reset Potential, v_{reset}	-3.0	-2.0	-3.0	-2.0	-3.0
Initial Decay Time, t_0	2,000	10,000	4,000	1,000	1,000
Initial Filters, f_0	32	8	8	16	16
# Down Samples	3	5	5	5	5
Data Augmentation					
Rotation up to $\pm 22.5^\circ$	Yes	Yes	Yes	Yes	No
Flip left-right	No	No	Yes	Yes	No
Flip time/polarity	Yes	Yes	Yes	Yes	No
Preprocessing repeats	∞ (online)	8	8	8	1

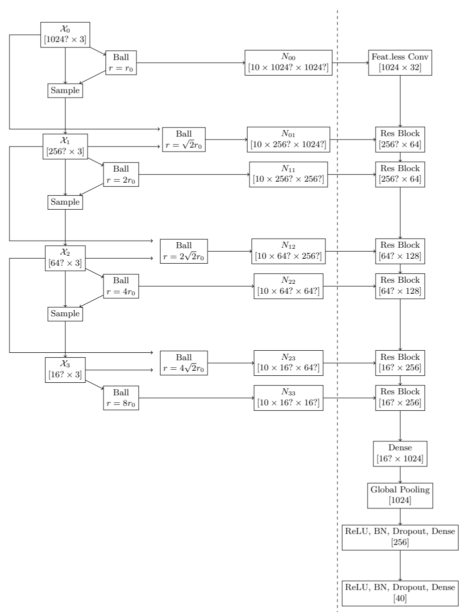
Table 7: Event stream dataset summary statistics and model/data augmentation hyperparameters.



(a) In-place residual block



(b) Down-sample residual block



(c) Large Point Cloud Network, $r_0 = 0.1125$. Numbers in brackets represent output example dimensions. Dimensions with question marks (?) correspond to approximate number of points when using no point dropout. Dashed line corresponds to preprocessing/batching divide. BN is batch normalization, and Dropout uses a rate of 0.5

Fig. 4

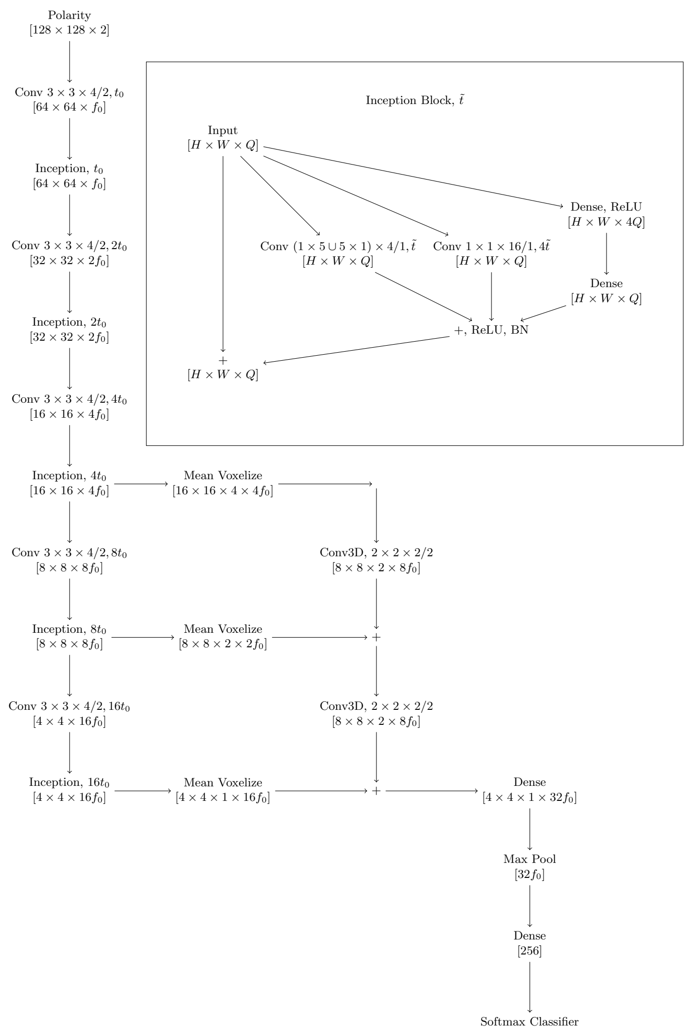


Fig. 5: Network architecture for event stream inference for CIFAR10-DVS. Conv $h \times w \times t/S, \tilde{t}$ is a down-sampling convolution with spatial stride S, spatial kernel shape $h \times w$ and temporal kernel size t, $i.e. M_u = hw$, $M_v = t$. The output stream is the result of LIF subsampling with the same spatial kernel size and decay time $\tilde{t}$. Edges with $\Delta t > 4\tilde{t}$ are cropped, and convolutions use Δt scaled by $\tilde{t}$. Each down-sampling convolution, the pre-max-pooling dense layer and the final dense layer are all followed by ReLU, batch normalization and dropout with rate 0.5. Each mean voxelization is followd by batch normalization, and each Conv3D is followed by ReLU and batch normalization.