MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences

Xingyu Liu
Stanford University

Mengyuan Yan
Stanford University

Jeannette Bohg
Stanford University

Abstract

Understanding dynamic 3D environment is crucial for robotic agents and many other applications. We propose a novel neural network architecture called MeteorNet for learning representations for dynamic 3D point cloud sequences. Different from previous work that adopts a grid-based representation and applies 3D or 4D convolutions, our network directly processes point clouds. We propose two ways to construct spatiotemporal neighborhoods for each point in the point cloud sequence. Information from these neighborhoods is aggregated to learn features per point. We benchmark our network on a variety of 3D recognition tasks including action recognition, semantic segmentation and scene flow estimation. MeteorNet shows stronger performance than previous grid-based methods while achieving state-of-the-art performance on Synthia. MeteorNet also outperforms previous baseline methods that are able to process at most two consecutive point clouds. To the best of our knowledge, this is the first work on deep learning for dynamic raw point cloud sequences.

1. Introduction

Our world is three dimensional. In many applications such as autonomous driving and robotic manipulation, the autonomous agent needs to understand its 3D environment. Among various 3D geometric representations, point clouds are the closest to raw sensory data from LiDAR or RGB-D cameras. Point clouds do not suffer as much from quantization errors compared to other geometric representations such as grids. Recently, deep architectures have been proposed that directly consume a single or a pair of point clouds for various 3D recognition tasks [21, 22, 20, 14, 24]. These architectures have outperformed methods based on other geometric representations.

Our world is also dynamic. Many recognition tasks benefit from sequences of temporal data that are longer than two frames. Examples include estimating the acceleration of a moving object or recognizing human actions. Unlike 2D image videos which can be represented as a regular spatiotemporal 3D grid and learned by 3D convolutional neural networks (CNNs) [27, 10, 2], dynamic 3D point cloud sequences have an entirely different structure and are more challenging to learn. Recently, deep architectures were proposed for dynamic point cloud sequences that convert the irregular 3D data to a grid representation and leverage 3D or 4D convolutions along the time dimension [15, 4]. However, the grid quantization error is inevitable in these methods and can be fatal for robotic applications that require precise localization. Moreover, performing convolution with a shared and fixed kernel everywhere in the 3D or 4D space is either inefficient or requires special engineering efforts such as sparse convolution [4].

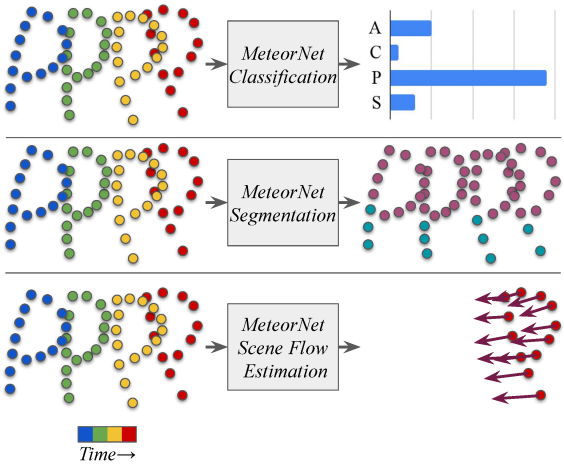


Figure 1: **MeteorNet applications.** Our model directly consumes a dynamic sequence of raw point clouds, and learns both global and local point features for various applications including classification, semantic segmentation and scene flow estimation.

In this work, we present a novel method for learning representations for dynamic 3D point cloud sequences. The key to our approach is a novel neural network module named the *Meteor* module. This module takes in point cloud sequences and aggregates information in spatiotemporal neighborhoods to learn features for each point. The module can be stacked on top of each other, where per-point features from the previous module are input to the next module. The stacked modules hierarchically aggregate information from larger neighborhoods. Inspired by [21, 22, 33], the aggregation process is implemented by

applying the same multi-layer perceptrons (MLPs) to each point in the neighborhood and max pooling afterwards.

We propose two methods for determining the spatial-temporal neighborhoods to address the motion range of an object: direct grouping and chained-flow grouping. The former method directly increases the grouping radius over time. The latter method tracks object motions and uses offline estimated scene flow to construct efficient and effective neighborhoods. We conduct intensive experiments to compare these two methods.

As visualized in Figure 1, learned features from Meteor modules can be used for downstream tasks such as classification, segmentation or scene flow estimation. We name the resulting deep neural network *MeteorNet*. For semantic segmentation, MeteorNet achieved state-of-the-art results on the dataset Synthia [23]. It also outperforms previous methods on a semantic segmentation dataset derived from KITTI [18]. MeteorNet specifically showed its advantage for recognizing movable objects on these two datasets. For scene flow estimation, MeteorNet beats previous baselines and achieves leading performance on FlyingThings3D [17] and the KITTI scene flow dataset [19]. It also achieves leading performance on the action recognition dataset MSRAction3D [12]. To the best of our knowledge, this is the very first work on deep learning for dynamic *raw* point cloud sequences. We expect that our MeteorNet can benefit research and applications in autonomous driving, robotic manipulation and related domains.

2. Related Work

Deep learning for RGB videos Existing approaches towards deep learning on videos can be categorized by how the temporal relationship between frames is modelled. The first family of approaches extracts a global feature for each video frame with a shared CNN and uses recurrent neural nets to model temporal relations [6, 36]. The second family of approaches learns temporal relations from offline-estimated optical flow [5] or optical flow trajectories [25] with a separate branch of the network besides the RGB branch. The third family of approaches uses 3D CNNs and learns temporal relations implicitly [27, 2, 10, 31, 38]. The fourth family of approaches uses non-local operations [32] or correspondence proposals [13] to learn long-range dependencies. Our work is a deep learning method for 3D videos and is inspired by the above methods.

Grid-based 3D deep learning Different representations for 3D geometry have been discussed in the literature [9]. A 3D occupancy grid is one of the most popular representations. Previous works have explored 3D convolution [34, 16] or sparse 3D convolution [35] for various 3D recognition tasks. Recent works on deep learning for 3D sequences used a 4D occupancy grid representation by adding an additional time dimension. Fast-and-Furious [15] proposed to view the vertical dimension as feature channels and apply 3D convolutions on the remaining three dimensions. MinkowskiNet [4] explicitly used sparse 4D convolution on a 4D occupancy grid. Instead of quantizing the raw point clouds into an occupancy grid, our method directly processes point clouds.

Deep learning on 3D point clouds Another popular representation of 3D geometry is 3D point clouds. Two major categories of methods have been explored. The first category is based on PointNet [21]. The core idea is a symmetric function constructed with shared-weight deep neural networks applied to every point followed by an element-wise max pooling. Follow-up work is PointNet++ [22] which extracts local features of local point sets within a neighborhood in Euclidean space and hierarchically aggregates features. Dynamic graph CNN [33] proposed a similar idea. The difference is that the neural network processes point pairs instead of individual points. FlowNet3D [14] lets the shared neural network take mixed types of modalities, i.e. geometric features and displacement, as inputs to learn scene flow between two point clouds.

The second category of methods combines the grid and point representation. VoxelNet [37] divides the space into voxels, uses local PointNets within each voxel and applies 3D convolution to voxel grids. SPLATNet [26] interpolates the point values to grids and applies 3D convolution before interpolating back to the original point cloud. Our work lies in the first category and focuses on learning representations for point cloud sequences.

3. Deep Learning on 3D Point Cloud Sequences

Our proposed method addresses the following three properties of point cloud sequences:

1. **Unordered intra-frame.** Points within the same frame should be treated as an unordered set. Change of feeding order of points within frames should not change the output of the deep learning model.

2. **Ordered inter-frame.** Points in different frames should be distinguished by their time stamps. Changing the time stamp of a point means moving the point to a different frame and should change the resulting feature vector.

3. **Spatiotemporal metric space.** Neighboring points form a meaningful local structure. For point cloud sequences, points that are close spatially *and* temporally should be considered neighbors. Therefore, the metric space that defines the neighborhood should include both the spatial and temporal domains.

In this section, we first briefly review point cloud deep learning techniques. Then we describe the Meteor module, the core module of our network, which serves the above three properties. We also explain the overall architecture design choices for various downstream applications. Finally,

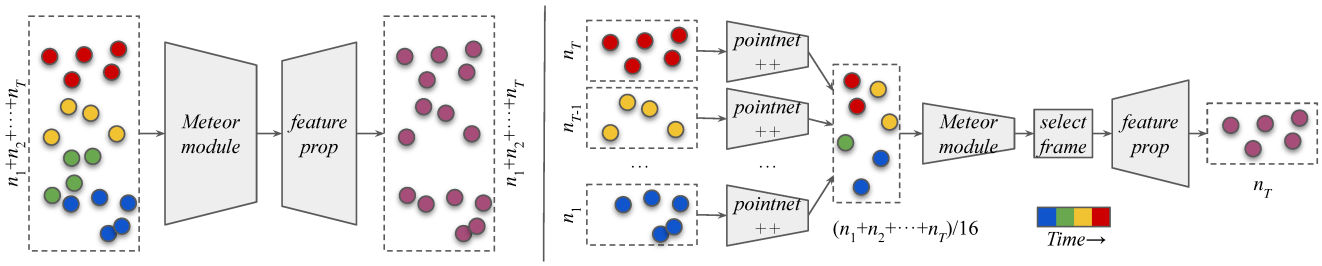


Figure 2: Architecture design choices for MeteorNet. Left: Early fusion with per-point output for all frames. Right: Late fusion with per-point output for the last frame.

we present the theoretical foundation of universal approximation for our architecture.

3.1. Review of PointNet

Our method is inspired by the seminal work of PointNet [21], a neural network architecture for deep learning on a single point cloud. It has been proven in [21] that given a set of point clouds $\mathcal{X} \subseteq \{\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \mid n \in \mathbb{Z}^+, \mathbf{x}_i \in [0,1]^m\}$ and any continuous set function $f : \mathcal{X} \to \mathbb{R}^c$ w.r.t the Hausdorff distance, there exists a set function in the form of

$$g(S) = \gamma \circ \underset{\mathbf{x}_i \in S}{MAX}\{\eta(\mathbf{x}_i)\}$$

that can approximate f on $\mathcal{X}$ arbitrarily closely. $\eta : \mathbb{R}^m \to \mathbb{R}^r$ and $\gamma : \mathbb{R}^r \to \mathbb{R}^c$ are two continuous functions and MAX is the element-wise maximum operation. In practice, η and γ are instantiated to be MLPs. The follow-up work of PointNet++ [22] extracts features in local point sets within a neighborhood and hierarchically aggregates features. For a point $\mathbf{x}_i \in S$, its learned feature is

$$g'(\mathbf{x}_i) = \gamma \circ \underset{\{\mathbf{x}_j | \mathbf{x}_j \in S, \mathbf{x}_j \in \mathcal{N}_0(\mathbf{x}_i)\}}{MAX}\{\eta(\mathbf{x}_j)\}$$

where the neighborhood $\mathcal{N}_0(\mathbf{x}_i)$ can be decided by a fixed radius r or the k nearest neighbors of $\mathbf{x}_i$.

3.2. Meteor Module

A point cloud sequence of length T is defined as a T-tuple of 3D point clouds $S = (S_1, S_2, \ldots, S_T)$. Each element of this tuple is a 3D point set $S_t = \{p_i^{(t)} \mid i = 1, 2, \ldots, n_t\}$, where the point $p_i^{(t)}$ is represented by its Euclidean coordinates $\mathbf{x}_i^{(t)} \in \mathbb{R}^3$ and a feature vector $f_i^{(t)} \in \mathbb{R}^c$. The feature vector can be attributes obtained from sensors, e.g. color, or output from the previous Meteor module. Our proposed Meteor module consumes S as input and produces an updated feature vector $h(p_i^{(t)})$ for every point $p_i^{(t)}$ in S. The first step of Meteor module is to find neighboring points of $p_i^{(t)}$ in the same or nearby frames to form a local spatiotemporal neighborhood $\mathcal{N}(p_i^{(t)})$.

Given $\mathcal{N}$, we introduce two instantiations of h for Meteor module. The first instantiation is for applications where the correspondence across frame is important (e.g. scene flow estimation). For each $(p_j^{(t')}, p_i^{(t)})$ pair, we pass the feature vectors of two points and difference of their spatiotemporal positions into to an MLP with shared weights ζ, followed by an element-wise max pooling. The updated feature of $p_i^{(t)}$ is then obtained by

$$h(p_i^{(t)}) = \underset{p_j^{(t')} \in \mathcal{N}(p_i^{(t)})}{MAX}\{\zeta(f_j^{(t')}, f_i^{(t)}, \mathbf{x}_j^{(t')} - \mathbf{x}_i^{(t)}, t' - t)\}$$

The second instantiation is for applications where the correspondence between points across frames is not important (e.g. semantic segmentation), we pass the feature vector of $p_j^{(t')}$ and difference of the spatiotemporal positions between $p_j^{(t')}$ and $p_i^{(t)}$ to ζ followed by a max pooling layer

$$h(p_i^{(t)}) = \underset{p_j^{(t')} \in \mathcal{N}(p_i^{(t)})}{MAX}\{\zeta(f_j^{(t')}, \mathbf{x}_j^{(t')} - \mathbf{x}_i^{(t)}, t' - t)\}$$

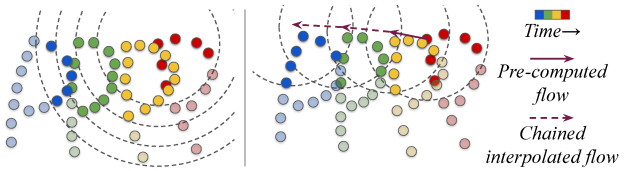
In terms of the spatiotemporal interaction range of a point $\mathcal{N}$, we introduce two types of point grouping methods for deciding $\mathcal{N}$, which is crucial for the Meteor module: *direct grouping* and *chained-flow grouping*.

Direct grouping. Our intuition is that, the maximum distance an object can travel increases as time increases. Thus we directly increase grouping radius in 3D spatial space as $|t - t'|$ increases, to cover motion range across time. Formally, the neighborhood is decided by

$$\mathcal{N}_d(p_i^{(t)}; r) = \{p_j^{(t')} \mid ||\mathbf{x}_j^{(t')} - \mathbf{x}_i^{(t)}|| < r(|t' - t|)\}$$

where r is a monotonically increasing function. Note that $r(0) > 0$ so that points within the same frame can also be grouped. Direct grouping is illustrated in the Figure 3(a).

Chained-flow grouping. In real-world dynamic point cloud sequences, the object represented by points in one frame usually has its corresponding points spatially close in neighboring frames. The motion trajectory of the object that is established through these correspondences is crucial for temporal understanding. In dynamic point cloud sequences, the interaction between a point and the other points in its spatiotemporal neighborhood should follow the direction of



(a) (b)

Figure 3: Two types of grouping methods: (a) direct grouping; (b) chained-flow grouping.

its motion. Such motion can be described by *scene flow* — dense 3D motion field [28].

In practice, for all t, we first estimate the backward scene flow $\mathfrak{f}_i^{(t,t-1)} \in \mathbb{R}^3$ from frames t to $t-1$

$$\{\mathfrak{f}_i^{(t,t-1)}\}_i = \mathcal{F}_0(\{p_i^{(t)}\}, \{p_j^{(t-1)}\})$$

where $\mathcal{F}_0$ is a scene flow estimator between point clouds $\{p_i^{(t)}\}$ and $\{p_j^{(t-1)}\}$, e.g. FlowNet3D [14]. Then $\mathbf{x}_i'^{(t-1)} = \mathbf{x}_i^{(t)} + \mathfrak{f}_i^{(t,t-1)}$ is the estimated position of the virtual corresponding point of $p_i^{(t)}$ in frame $t-1$.

To estimate the corresponding position of $p_i^{(t)}$ in frame $t-2$, we first interpolate the scene flow estimation results $\{\mathfrak{f}_j^{(t-1,t-2)}\}_j$ at the position of virtual point $p_i'^{(t-1)}$. Among choices for interpolation, we use the simple inverse distance weighted average of k nearest neighbors

$$\mathfrak{f}_i'^{(t-1,t-2)} = \frac{\sum_{j=1}^{k} w(\mathbf{x}_j^{(t-1)}, \mathbf{x}_i'^{(t-1)}) \mathfrak{f}_j^{(t-1,t-2)}}{\sum_{j=1}^{k} w(\mathbf{x}_j^{(t-1)}, \mathbf{x}_i'^{(t-1)})}$$

where $w(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{d(\mathbf{x}_1,\mathbf{x}_2)^p}$ is the interpolation weight. We use $p=2, k=2$ by default. Then we chain the flow to estimate the corresponding position of $p_i^{(t)}$ in frame $t-2$: $\mathbf{x}_i'^{(t-2)} = \mathbf{x}_i^{(t)} + \mathfrak{f}_i^{(t,t-1)} + \mathfrak{f}_i'^{(t-1,t-2)}$. Its position in frames beyond $t-2$ can be further interpolated and chained by repeating the above process. The pre-computed flow vectors $\mathfrak{f}$ and chained interpolated flow vectors $\mathfrak{f}'$ are illustrated in Figure 3(b). Then the neighborhood for chained-flow grouping is determined by

$$\mathcal{N}_c(p_i^{(t)}; r) = \{p_j^{(t')} \mid ||\mathbf{x}_j^{(t')} - \mathbf{x}_i'^{(t')}|| < r\}$$

where r is a constant value by default. One can also use a monotonically increasing function of $|t-t'|$ for r to compensate for scene flow estimation error.

Compared to direct grouping, chained-flow grouping focuses on tracking the motion trajectories and correspondences of each point so that a smaller grouping radius can be applied. It can also be more computationally efficient.

3.3. Overall Architecture and Applications

The feature vectors output by Meteor module can be further processed to obtain a quantity for the entire sequence, such as class scores (e.g. for classification); or propagated back to each point to obtain a per-point quantity, such as the class scores for all points (e.g. for semantic segmentation); or a per-point quantity for the points in a particular frame (e.g. scene flow estimation). We name the overall architecture for classification, semantic segmentation and scene flow estimation as MeteorNet-cls, MeteorNet-seg and MeteorNet-flow respectively.

There are generally two types of design choices for including Meteor modules in the architecture: Early fusion and Late fusion.

Early fusion We apply Meteor module at the first layer so that the points from different frames are mixed from the beginning, as illustrated in left part of Figure 2. In the following experiment section, MeteorNet-cls and MeteorNet-seg used early fusion.

Late fusion We apply a few layers of feature learning (e.g. PointNet++) individually to points in each frame before mixing them in the Meteor module, as illustrated in right part of Figure 2. It allows the model to capture higher level semantic features. In the follwing experiment section, MeteorNet-flow used late fusion.

3.4. Theoretical Foundation

We provide a theoretical foundation for our MeteorNet by showing the universal approximation ability of Meteor module to continuous functions on point cloud sequences.

Suppose $\forall t, \mathcal{X}_t = \{S_t \mid S_t \subseteq [0,1]^m, |S_t| = n, n \in \mathbb{Z}^+\}$ is the set of m-dimensional point clouds inside an m-dimensional unit cube at time $t \in \mathbb{Z}$. We define single-frame Hausdorff distance $d_H(S_i, S_j)$ for $S_i \in \mathcal{X}_i$ and $S_j \in \mathcal{X}_j$. $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_T$ is the set of point cloud sequences of length T. Suppose $f : \mathcal{X} \to \mathbb{R}$ is a continuous function on $\mathcal{X}$ w.r.t $d_{seq}(\cdot,\cdot)$, i.e. $\forall \epsilon > 0, \exists \delta > 0$, for any $S, S' \in \mathcal{X}$, if $d_{seq}(S, S') < \delta$, $|f(S) - f(S')| < \epsilon$. Here, we define the distance of point cloud sequences $d_{seq}(\cdot,\cdot)$ as the maximum per-frame Hausdorff distance among all respective frame pairs, i.e. $d_{seq}(S, S') = \max_t\{d_H(S_t, S_t')\}$. Our theorem says that f can be approximated arbitrarily closely by a large-enough neural network and a max pooling layer with enough neurons.

Theorem 1. *Suppose $f : \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_T \to \mathbb{R}$ is a continuous function w.r.t $d_{seq}(\cdot,\cdot)$. $\forall \epsilon > 0$, $\exists$ a continuous function $\zeta(\cdot,\cdot)$ and a continuous function γ, such that for any $S = (S_1, S_2, \ldots, S_T) \in \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_T$,*

$$\left| f(S) - \gamma \circ \left(\underset{\mathbf{x}_i^{(t)} \in S_t, t \in \{1,2,\ldots,T\}}{MAX} \{\zeta(\mathbf{x}_i^{(t)}, t)\} \right) \right| < \epsilon$$

where MAX is a vector max operator that takes a set of vectors as input and returns a new vector of the element-wise maximum.

The proof of this theorem is in the supplementary material. The key idea is to prove the homeomorphism between a point cloud sequence and a single-frame point cloud by designing a continuous bijection between them so that the conclusions for a single-frame point cloud can be used. We explain the two implications of the theorem. First, for a point cloud sequence of length T, function ζ essentially processes $T\times$ more points identically and independently than for a single-frame point cloud. Thus compared to single-frame PointNet [21], it may need a larger network as well as a larger bottleneck dimension for MAX to maintain the expressive power for a point cloud sequence. We will show this in later experiments. Second, simply adding an additional time stamp t as input to ζ, the network is able to distinguish the points from different frames but still treats the points from the same frame as an unordered set.

4. Grids versus Point Clouds: A Toy Example

We constructed a toy dataset of point cloud sequences where grid-based methods fail in learning a simple classification task. Through this simple dataset, we show the drawbacks of these grid-based methods and the advantage of our architecture which works on raw point clouds.

The dataset consists of sequences of a randomly positioned particle moving inside a $100 \times 100 \times 100$ cube. The moving direction is randomly chosen from 6 possible directions, all parallel to one of the edges of the cube. Each sequence has four frames. There are four categories for the motion speed of a particle: "static", "slow", "medium" and "fast". The moving distance of the particle at each step is zero for the "static" category and randomly chosen between $[0.09, 0.11]$, $[0.9, 1.1]$ and $[9, 11]$, respectively, for the other three categories. The dataset has 2000 training and 200 validation sequences, each with an equal number of sequences per label. Figure 4 illustrates an example data point in our dataset with the "fast" label.

We used a toy version of two recent grid-based deep architectures for dynamic 3D scenes, FaF [15] and MinkNet [4], as well as our MeteorNet. We only allow three layers of neurons within the three network architectures and convolution kernel sizes of no larger than 3. The architecture details are listed in Table 1. The particle is represented by grid occupancy for grid-based methods, and by a 3D point for MeteorNet. The experiment settings are designed to simulate situations where stacking convolution layers to increase expressive power is insufficient or inefficient. No data augmentation is used. The training and validation results are listed in Table 1. Our model can perfectly learn the toy dataset, while previous grid-based methods cannot avoid significant errors regardless of grid size chosen.

We provide an explanation as follows. If the grid size is chosen to be comparable to the "slow" step size, "static" and "slow" particles can be correctly classified. However, both "medium" or "fast" particles require huge convolution receptive fields to cover the moving distance. Thus, the toy grid-based networks failed to classify them. If the grid size is chosen to be comparable to the "fast" (or "medium") step size, "fast" (or "medium") particles can be correctly classified. However, both "slow" and "static" particles mostly yield similar stationary grid occupancy. Thus, the toy grid-based networks failed to classify them. Through this experiment, we show the advantage of our MeteorNet: it is grid size agnostic and can accurately learn both long- and short-range motion at the same time.

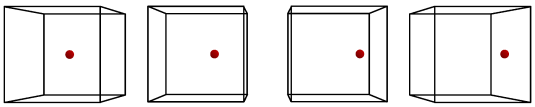


Figure 4: A "fast" example in our dataset.

		FaF [15]	MinkNet [4]	**MeteorNet (ours)**
arch	layer 1	conv16, 3×3×3	conv16, 3×3×3×3	mlp [16,16]
	layer 2	conv16, 3×3×3	conv16, 3×3×3×3	
	layer 3	max pool, fc		
grid size	0.05	N/A, N/A	76.41, 75.00	**100.00**, **100.00**
	0.10	N/A, N/A	76.16, 75.50	
	0.50	91.05, 79.00	89.31, 87.50	
	1.00	84.20, 78.50	79.73, 82.00	
	5.00	61.15, 63.50	59.87, 61.00	
	10.0	59.65, 60.00	51.41, 53.00	

Table 1: Toy experiment settings and results. Accuracy results (%) are shown in "*train*, *val*". N/A denotes resource (e.g.memory) not enough.

5. Experiments

The design of our MeteorNet is motivated by two hypotheses. First, deep architectures that process raw point clouds suffer less from quantization error than grid-based methods. Therefore, they achieve a better performance in a variety of recognition tasks. Second, many recognition tasks benefit from longer input sequences of 3D data. To show this, we apply MeteorNet to three 3D recognition tasks: action recognition, semantic segmentation and scene flow estimation, which are typical classification and regression tasks. We compare the performance to a variety of baselines including grid-based and single-frame methods. We also visualize some example results.

5.1. Classification

We first conducted an experiment on point cloud sequence classification. We use the MSRAction3D dataset [12]. It consists of 567 Kinect depth map sequences of 10 different people performing 20 categories of actions. There are 23,797 frames in total. We reconstructed point cloud sequences from the depth maps. We use the same train/test

Method	Input	# of Frames	Accuracy
Vieira et al. [29]	depth	20	78.20
Kläser et al. [11]	depth	18	81.43
Actionlet [30]	groundtruth skeleton	full	88.21
PointNet++ [22]	point	1	61.61
MeteorNet-cls	point	4	78.11
		8	81.14
		12	86.53
		16	88.21
		24	**88.50**

Table 2: Classification accuracy on MSRAction3D (%).

split as previous work [30]. The classification results are listed in Table 2. The baselines are descriptor-based methods on depth maps [29, 11] and skeletons [30], and PointNet++ [22]. MeteorNet-cls significantly outperforms all baselines on this dataset. We also show that action recognition accuracy of MeteorNet-cls benefits from longer point cloud sequences.

5.2. Semantic Segmentation

We conduct two semantic segmentation experiments. We first test MeteorNet-seg on large-scale synthetic dataset Synthia [23] to perform an ablation study and compare with a sparse 4D CNN baseline. Then we test MeteorNet-seg on real LiDAR scans from KITTI dataset [8].

Synthia dataset It consists of six sequences of driving scenarios in nine different weather conditions. Each sequence consists of four stereo RGBD images from four viewpoints captured from the top of a moving car. We reconstruct 3D point clouds from RGB and depth images and create 3D point cloud sequences. The scene is cropped by a 50m×50m×50m bounding box centered at the car. We then use farthest point sampling to downsample the scene to 16,384 points per frame. We used the same train/validation/test split as [4]: sequences 1-4 with weather conditions other than sunset, spring and fog are used as train set; sequence 5 with foggy weather are used as validation set; and sequence 6 with sunset and spring weather are used as test set. The train, validation and test set contain 19,888, 815 and 1,886 frames respectively.

As an ablation study, we used MeteorNet-seg with various settings of model sizes and number of input frames. Compared to MeteorNet-seg-s, MeteorNet-seg-m has a larger bottleneck dimension at each max pooling layer. Compared to MeteorNet-seg-m, MeteorNet-seg-l has the same max pooling dimensions but larger dimensions in non-bottleneck layers. Among the baselines, 3D and 4D MinkNet [4] voxelize the space and use 3D spatial or 4D spatiotemporal sparse convolution to perform segmentation. The evaluation metrics are the per-class and overall mean IoU as well as overall segmentation accuracy. The results are listed in Table 3.

Our multi-frame MeteorNet-seg outperforms the single-frame PointNet++. Furthermore, point-based methods outperform sparse-convolution-based methods by a significant margin in most categories as well as in the overall metrics. We also found interesting results in the ablation study. First, increasing the max pooling bottleneck dimension is more effective than increasing the non-bottleneck layer dimension for multi-frame input. Second, increasing the number of frames without increasing the model size can negatively impact the network performance. Third, MeteorNet-seg with chained-flow grouping achieves the best performance on movable objects such as "Car" and "Pedestrian" among all point-based methods. This shows its advantage in detecting motion. Figure 5 visualizes two example results. Our model can accurately segment most objects.

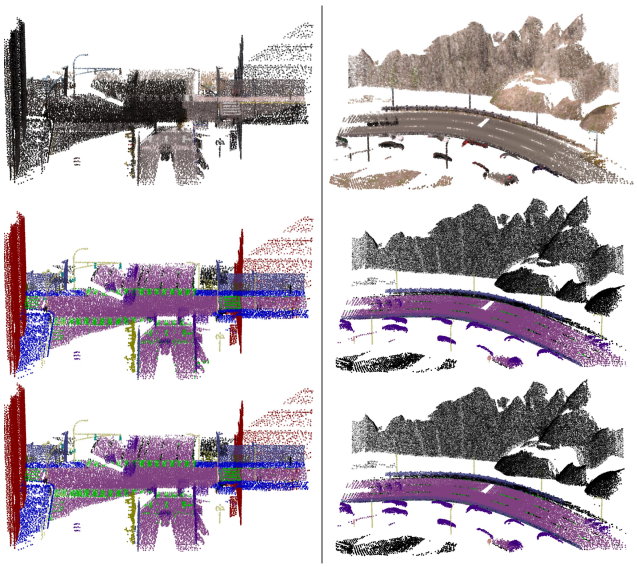


Figure 5: Visualization of two example results from the Synthia dataset. From the top: RGB input, ground truth, predictions.

KITTI dataset We derived a semantic segmentation dataset from the KITTI object detection dataset [1] by converting bounding box labels to per-point labels. Additional frames are from the corresponding LiDAR data in the KITTI raw set [8]. We used GPS/IMU data to transform the point cloud into world coordinates to eliminate ego motion. There are three semantic classes: "Car", "Pedestrain/Cyclist" and "Background". The "Car" class mostly consists of points of parked and static cars[1] while the "Pedestrain/Cyclist" class mostly consists of points of moving people. We used the default train/val split as in [20]. The segmentation results are listed in Table 4.

MeteorNet yields similar segmentation accuracy as PointNet++ on "Car" but significantly improves on "Pedestrian/Cyclist" with multiple frames as input. As expected,

[1]We provide an approximate statistics by randomly sampling 200 point cloud sequences from all 7481 train/val sequences and count "Car" instances. There are totally 884 "Car" instances. Among them, 654 (73.9%) are static cars, 230 (26.1%) are moving cars.

Method	Params (M)	# of Frames	IoU												mIoU	mAcc
			Bldng	Road	Sdwlk	Fence	Vegitn	Pole	Car	T.sign	Pdstr	Bicyc	Lane	T.light		
3D MinkNet14 [4]	19.31	1	89.39	97.68	69.43	86.52	98.11	97.26	93.50	79.45	92.27	0.00	44.61	66.69	76.24	89.31
4D MinkNet14 [4]	23.72	3	90.13	**98.26**	73.47	87.19	**99.10**	97.50	94.01	79.04	**92.62**	0.00	50.01	68.14	77.46	88.01
PointNet++ [22]	0.88	1	96.88	97.72	86.20	92.75	97.12	97.09	90.85	66.87	78.64	0.00	72.93	75.17	79.35	97.83
MeteorNet-seg-s (direct)	0.88	2	98.08	97.77	87.16	93.53	96.91	97.47	94.04	77.22	72.19	0.00	**73.59**	75.75	80.31	98.11
MeteorNet-seg-m (direct)	1.36	2	97.65	97.83	90.03	94.06	97.41	**97.79**	94.15	82.01	79.14	0.00	72.59	**77.92**	81.72	98.17
MeteorNet-seg-m (chain)	1.36	2	98.22	97.79	90.98	93.18	98.31	97.45	**94.30**	76.35	81.05	0.00	74.09	75.92	81.47	98.13
MeteorNet-seg-m (direct)	1.36	3	**98.45**	97.92	**91.57**	**94.40**	97.54	97.46	94.11	79.04	75.04	0.00	73.17	74.93	81.13	**98.28**
MeteorNet-seg-l (direct)	1.78	3	98.10	97.72	88.65	94.00	97.98	97.65	93.83	**84.07**	80.90	0.00	71.14	77.60	**81.80**	98.15

Table 3: Semantic Segmentation results on the Synthia dataset. Metrics are mean IoU and mean accuracy (%).

Method	# of Frames	Car	Pdstr/ Cyclst	Bkgrd	mIoU
PointNet++ [22]	1	74.06	36.43	98.19	69.56
MeteorNet-seg (direct)	2	**74.53**	42.19	98.24	71.65
	3	71.22	**50.93**	98.12	**73.42**

Table 4: **Semantic Segmentation results on KITTI dataset.** Metrics are per-class and average IoU (%).

Method	Input	Frames	mean	std	acc	outlier
FlowNet-C [7]	depth	2	0.473	0.275	10.75	11.60
FlowNet-S [7]	depth	3	0.437	0.281	22.25	10.62
FlowNet3D [14]	points	2	0.218	0.196	49.46	2.37
MeteorNet-flow (direct)	points	3	0.219	0.187	47.44	2.30
	points	4	0.214	0.190	**52.12**	2.40
MeteorNet-flow (chained-flow)	points	3	0.215	0.194	49.63	2.44
	points	4	**0.209**	**0.184**	49.91	**2.28**

Table 5: **Flow estimation results on the FlyingThings3D dataset.** Metrics are for the end-point-error (EPE) of scene flow: mean, standard deviation, accuracy (%, ratio of estimations with EPE $<$0.1 or 10%), and outlier ratio (%, of estimations with EPE $>$1.0).

the accuracy on "Pedestrian/Cyclist" continues to increase as the number of input frames increases. This underlines the advantage of MeteorNet for learning object motion.

5.3. Scene Flow Estimation

Labelling dense scene flow in real point cloud data is very expensive. To the best of our knowledge, there does not exist any large-scale real-world point cloud dataset with per-point scene flow annotations. A common approach is to train on a large-scale synthetic dataset and then to finetune on a smaller real dataset [14]. To this end, we conducted two experiments: we first train MeteorNet on the FlyingThings3D dataset [17], then finetune MeteorNet on the KITTI scene flow dataset [19].

FlyingThings3D dataset This synthetic dataset consists of RGB and disparity image videos rendered from scenes of multiple randomly moving objects from ShapeNet [3]. It provides 8,955 training videos and 1,747 test videos where each video has 10 frames. We reconstructed 3D point clouds from disparity maps. Maps of optical flow and disparity change are provided for consecutive frames, from which 3D scene flow can be reconstructed. This dataset is challenging because of large displacements and strong occlusions. We randomly sampled 20,000 4-frame sequences from training videos as our training set and 2,000 4-frame sequences from testing videos as our test set. We used random rotation for data augmentation.

We evaluate 3D end-point-error (EPE) of scene flow, which is defined as the L_2 distance between the estimated flow vectors and the ground truth flow vectors. We report four aspects of EPE: mean, standard deviation, accuracy with threshold 10% or 0.1, and outlier ratio with threshold 1.0, as our evaluation metrics. Among the baselines, FlowNet-C/FlowNet-S are convolutional architectures adapted from [7] that take two/three dense depth maps (converted to xyz coordinate maps) as input to estimate per-pixel scene flow instead of optical flow as originally done in [7]. FlowNet3D [14] is a PointNet-based architecture that takes two point clouds as input to estimate per-point scene flow. The results are listed in Table 5.

We see that convolution-based methods have a hard time capturing accurate scene flow probably due to occlusion. Point based methods such as FlowNet3D can effectively capture the accurate motion in point clouds. MeteorNet-flow can further improve scene flow estimation with more frames as input. MeteorNet-flow with direct grouping shows better accuracy for small displacements while MeteorNet-flow with chained-flow grouping shows better overall performance.

By using chained flow, points are grouped into a neighborhood which are located close to the proposed corresponding position as predicted by flow from past frames. Therefore, the model is provided more evidence to estimate the correct flow direction and magnitude. Furthermore, as the number of input frames increases for MeteorNet-flow, the performance gain is consistent.

KITTI scene flow dataset This dataset provides ground truth disparity maps and optical flow for 200 frame pairs [19]. From this, we reconstructed 3D scene flow. Only 142 out of 200 pairs have corresponding raw LiDAR point cloud data and thus allow us to use preceding frames from the

KITTI raw dataset [8]. We project the raw point clouds onto the groundtruth maps to obtain the groundtruth 3D scene flow. We first train the models on the FlyingThings3D dataset until convergence and use first 100 pairs to finetune the models. Then we use the remaining 42 pairs for testing. We report two aspects of EPE of scene flow: mean and standard deviation, as our evaluation metrics. The baseline FlowNet3D is trained and finetuned the same way as MeteorNet-flow. The results are listed in Table 6.

Method	Input	Frames	mean	std
FlowNet3D [14]	points	2	0.287	0.250
MeteorNet-flow (direct)	points	3	0.282	**0.204**
	points	4	0.263	0.210
MeteorNet-flow (chained-flow)	points	3	0.277	0.244
	points	4	**0.251**	0.227

Table 6: **Flow estimation results on KITTI sceneflow dataset.** Metrics are the mean and standard deviation of the End-point-error (EPE) of scene flow.

Our MeteorNet-flow with three and four frames as input outperforms the baselines. As the number of frame increases, the performance gain is consistent. The version using chained-flow for grouping is better in mean error while the direct grouping version has lower standard deviation and is more robust. Our hypothesis is that when points are sparse, the initial flow estimation may have larger error which propagates through time and thereby affects the final prediction. Figure 6 visualizes some examples of the resulting flow estimates.

6. Discussion

Relation to Other Architectures MeteorNet can be seen as a generalization of several previous architectures. When the input is one point cloud frame, MeteorNet can be reduced to either PointNet++[22] or DGCNN [33], depending on the instantiation of the h function in Subsection 3.2. When the input has two frames of point clouds, MeteorNet can be reduced to FlowNet3D [14].

Direct Grouping vs. Chained-flow Grouping In subsection 3.2, we discussed the two grouping methods. One possible advantage of chained-flow grouping compared to direct grouping is its computational efficiency. Intuitively for direct grouping, the spatial neighborhood radius r should grow linearly with $t - t'$. Therefore, for a point cloud sequence with length T, the total number of points in the spatiotemporal neighborhood of a point is $\mathcal{O}(T^4)$. For chained-flow grouping, the spatial radius r does not need to grow with time, thus the number of points in the spatiotemporal neighborhood is only $\mathcal{O}(T^2)$. This limitation of the direct grouping can be mitigated by limiting the temporal radius of neighborhoods while increasing the number of stacked Meteor modules. This is similar to using smaller convolution kernels while increasing the number of layers in convolutional neural networks.

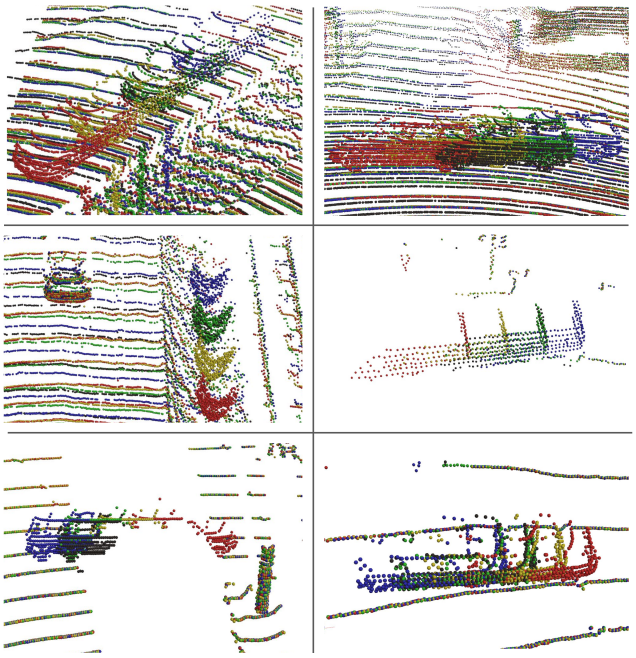


Figure 6: **Visualization of MeteorNet example results on the KITTI scene flow dataset.** Point are colored to indicate which frames they belong to: frames $t-3$, frame $t-2$, frame $t-1$, frame t. **Translated points** (frame $t-3$ + estimated scene flow) is in black. Green and black shapes are supposed to overlap for perfect estimation.

A potential problem for chained-flow grouping is when point clouds are too sparse and the flow estimation is inaccurate. In this case, errors may accumulate during chaining, and the resulting spatiotemporal neighborhood may deviate from the true corresponding points across time. Studying the effect of initial scene flow error on final performance will be left as future work.

7. Conclusion

In this work, we proposed a novel deep neural network architecture MeteorNet that directly consumes dynamic 3D point cloud sequences. We show how this new architecture outperforms grid-based and single-frame methods on a variety of 3D recognition tasks including activity recognition, semantic segmentation and scene flow estimation. We also show the universal approximation ability of our network and provide visualizations of example results.

Acknowledgements

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] Kitti 3d object detection benchmark leader board. http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d. Accessed: 2017-11-14 12PM. 6

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012, 2015. 7

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019. 1, 2, 5, 6, 7

[5] Andrew Zisserman Christoph Feichtenhofer, Axel Pinz. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2

[6] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2

[7] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 7

[8] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013. 6, 8

[9] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. *CSUR*, 2017. 2

[10] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 1, 2

[11] Alexander Kläser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. 6

[12] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *CVPR Workshops*, 2010. 2, 5

[13] Xingyu Liu, Joon-Young Lee, and Hailin Jin. Learning video representations from correspondence proposals. In *CVPR*, 2019. 2

[14] Xingyu Liu, Charles. R. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, 2019. 1, 2, 4, 7, 8

[15] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018. 1, 2, 5

[16] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 2

[17] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 2, 7

[18] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015. 2

[19] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 2, 7

[20] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, June 2018. 1, 6

[21] Charles R. Qi, H. Su, Kaichun Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, July 2017. 1, 2, 3, 5

[22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 5099–5108. Curran Associates, Inc., 2017. 1, 2, 3, 6, 7, 8

[23] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016. 2, 6

[24] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1

[25] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2

[26] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018. 2

[27] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2

[28] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *ICCV*, 1999. 4

[29] Antonio W Vieira, Erickson R Nascimento, Gabriel L Oliveira, Zicheng Liu, and Mario FM Campos. Stop: Space-time occupancy patterns for 3d action recognition from depth map sequences. In *CIARP*. Springer, 2012. 6

[30] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012. 6

[31] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. In *CVPR*, 2018. 2

[32] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2

[33] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic

graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 1, 2, 8

[34] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2

[35] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 2

[36] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[37] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 2

[38] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *ECCV*, 2018. 2