Spatial-Temporal Consistency Network for Low-Latency Trajectory Forecasting

Shijie Li, Yanying Zhou, Jinhui Yi, and Juergen Gall
University of Bonn

Abstract

Trajectory forecasting is a crucial step for autonomous vehicles and mobile robots in order to navigate and interact safely. In order to handle the spatial interactions between objects, graph-based approaches have been proposed. These methods, however, model motion on a frame-to-frame basis and do not provide a strong temporal model. To overcome this limitation, we propose a compact model called Spatial-Temporal Consistency Network (STC-Net). In STC-Net, dilated temporal convolutions are introduced to model long-range dependencies along each trajectory for better temporal modeling while graph convolutions are employed to model the spatial interaction among different trajectories. Furthermore, we propose a feature-wise convolution to generate the predicted trajectories in one pass and refine the forecast trajectories together with the reconstructed observed trajectories. We demonstrate that STC-Net generates spatially and temporally consistent trajectories and outperforms other graph-based methods. Since STC-Net requires only 0.7k parameters and forecasts the future with a latency of only 1.3ms, it advances the state-of-the-art and satisfies the requirements for realistic applications.

1. Introduction

Intelligent agents like autonomous vehicles and mobile robots navigate and interact in spaces that are shared with humans. The safety of the humans is therefore of the highest priority. To this end, agents are required to understand and forecast the trajectories of surrounding pedestrians or vehicles such that they can make the right decisions and for instance navigate safely and smoothly through a crowd. Trajectory forecasting, however, is challenging due to the complex behavior and interactions of humans in crowds. Furthermore, the available resources will always be a limiting factor as the agents are equipped with energy-efficient hardware. Hence, there is a need for compact forecasting models with a very low latency. In practice, a low latency is required for most applications since there is otherwise not enough time to adjust the motion and any millisecond can save lives.

(a) Social-STGCNN [24] (b) Ours

Figure 1: State-of-the-art graph-based approaches like [24] do not model the temporal motion of the trajectories well, which results in shaky and unrealistic trajectories (dashed yellow) as shown in (a). In this figure, red denotes the observed trajectory and blue the ground-truth future trajectory. Our approach addresses this issue and forecasts spatially and temporally consistent trajectories (b).

Over the last years, several approaches have been proposed for trajectory forecasting. RNN-based methods [1, 20, 36] utilize a recurrent model to model the trajectory of each pedestrian in the scene and their interactions are taken into account by a pooling operation. GAN-based methods [13, 17, 29] enable RNN-based methods to produce multiple socially plausible trajectories. Graph-based methods [15, 34, 24] model spatial relations as a graph and utilize graph convolutions. These methods, however, have in common that they use recurrent neural networks to generate the future trajectories, which results in a relatively high latency. To address this issue, Social-STGCNN [24] proposed a CNN for time-extrapolation. As a result, Social-STGCNN achieves a very low latency of 2ms. This, however, comes at the cost that the forecast trajectories are noisy and not temporally consistent as shown in Fig. 1a.

In this work, we therefore address this issue and present an approach that generates spatially and temporally consistent trajectories with a latency of less than 2ms. To achieve this, the proposed Spatial-Temporal Consistency Network (STC-Net), which is shown in Fig. 2, utilizes graph convolutions for spatial modeling and dilated temporal convolutions for temporal modeling. Our network generates trajectories that are consistent over the past and the future. This consistency is implicitly enforced by reconstructing the past, jointly refining the reconstructed past and forecast future, and computing the loss over the entire trajectory. In order

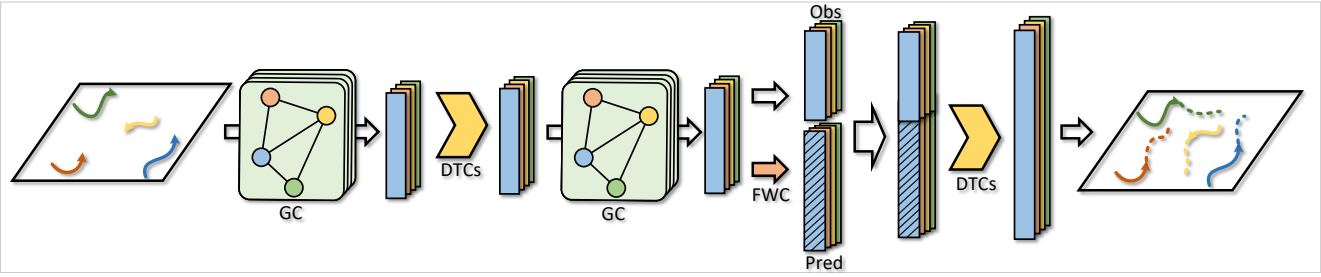


Figure 2: The proposed spatial-temporal consistency network takes observed trajectories (solid curves) of N objects as input and forecasts the future trajectories (dashed curves). It combines graph convolutions (GC) for modeling interactions of trajectories in close proximity and dilated temporal convolutions (DTCs) for capturing temporal relations over the entire trajectories. The feature-wise convolution (FWC) forecasts the features in one pass and the final refinement ensures the consistency of the reconstructed and forecast part of a trajectory.

to achieve a very low latency, we do not use any recurrent layers, but propose feature-wise convolutions that generate the future trajectories with variable length in one pass. The proposed network design yields very compact networks with only 0.7k parameters and 1.3ms latency.

We thoroughly evaluate the approach on three datasets with six different scenes and analyze its generalization ability by performing a cross-dataset experiment and evaluating the approach for different temporal sampling rates. The proposed approach outperforms other graph-based methods in terms of accuracy, model size, and latency. It is also important to note that most approaches cannot keep up with the frame-rate, i.e., the forecasting cannot be performed until the next frame is captured. This means that these methods are not fast enough for real-world applications. Even for a very slow frame-rate of 2.5 frames per second, only [13, 25, 24] are able to process the data fast enough. Compared to these methods, our approach is not only faster, but it is also by far more accurate.

2. Related Work

We briefly discuss the recent progress in trajectory forecasting and network architecture search. RNN-based methods model the motion of each pedestrian by a recurrent architecture and the context of nearby trajectories by pooling operations. Social-LSTM [1] is one of the earliest methods following this approach and several methods explore different strategies to improve the performance. PIF [20] takes human behavioral information into consideration and introduces rich visual features. SR-LSTM [36] proposes a social-aware information selection mechanism to select useful information from neighboring pedestrians. FvTraj [4] focuses on first-person view based trajectory forecasting with multi-head attention mechanisms. Recently, [23] proposed a variational auto-encoder to predict first the end-points of the trajectories and then the corresponding trajectories in a second step. While it achieves impressive results, it is too expensive for most applications. In contrast, our approach requires 3,000 times less parameters and is 467 times faster.

GAN-based methods take into account that many future trajectories might be plausible. To handle this problem, Social-LSTM [1] is converted into a generative model in Social-GAN [13]. Based on it, SoPhie [29] combines a social attention mechanism with a physical attention mechanism. CGNS [17] replaces the LSTMs in SoPhie [29] by GRUs. Similar to [23], Goal-GAN [9] uses a two-stage process for trajectory forecasting. Since these works also rely on recurrent architectures, they are as inefficient as RNN-based methods.

Graph-based methods use graph convolutions instead of simple pooling operations for modeling the spatial interactions. Social-BiGAT [15] is a graph-based generative model that uses a graph attention network to encode social interactions between pedestrians and a recurrent encoder-decoder architecture to predict future trajectories. Similar to Social-BiGAT [15], STGAT [14] captures spatial interactions by a graph attention mechanism at each time-step and uses LSTMs for forecasting. RSBG [30] proposes a social behavior graph which encodes the social representations. DAG-Net [25] uses a double attentive graph neural network for trajectory forecasting. These methods, however, remain inefficient since they still use a recurrent architecture. Social-STGCNN [24] omits the RNNs and uses a CNN for time extrapolation. While this results in a very low latency, it comes at the cost of a very weak temporal model.

Many approaches have been proposed for neural architecture search using, *e.g.,* reinforcement learning [37, 38] or evolutionary search [21]. The large search space, however, makes these approaches extremely expensive and resource demanding. Hence, a well-designed search space is of vital importance. Aiming at this, some new methods have been proposed, including differentiable architecture search [22], single-path one-shot sampling [8, 12], path-level binarization [6], and weight sharing [5, 33]. Recently, the design of efficient models with neural architecture search has been investigated for 2D image tasks [31, 32].

3. Spatial-Temporal Consistency Network

The goal of trajectory forecasting is predicting the upcoming trajectories given past observed trajectories. We denote the observed trajectory of each object n, which are commonly pedestrians but can also be other objects like vehicles, by $\mathcal{T}_o^n = \{\mathbf{p}_t^n = (x_t^n, y_t^n) \mid t \in \{1, ..., T_o\}\}$ and the future trajectory by $\mathcal{T}_f^n = \{\mathbf{p}_t^n = (x_t^n, y_t^n) \mid t \in \{T_o + 1, ..., T_o + T_f\}\}$, where $\mathbf{p}_t^n$ is the location of the object n at time t.

For forecasting, we use a Gaussian distribution $\mathcal{N}_{\mathbf{p}_t^n}(\hat{\mu}_t^n, \hat{\Sigma}_t^n)$, *i.e.,* we estimate for each $\mathbf{p}_t^n \in \mathcal{T}_f^n$ the mean $\hat{\mu}_t^n$ and the covariance matrix $\hat{\Sigma}_t^n$. In order to vectorize the covariance matrix, we use

$$\hat{\Sigma} = \begin{pmatrix} \hat{\sigma}_1^2 & \hat{\rho}\hat{\sigma}_1\hat{\sigma}_2 \\ \hat{\rho}\hat{\sigma}_1\hat{\sigma}_2 & \hat{\sigma}_2^2 \end{pmatrix}. \tag{1}$$

The covariance matrix is then defined by the three-dimensional vector $(\hat{\sigma}_1, \hat{\sigma}_2, \hat{\rho})$.

For training, we minimize the negative log-likelihood, *i.e.,* given for each trajectory the ground-truth position $\mathbf{p}_t^n$, we minimize

$$L_f = -\sum_{n=1}^{N} \sum_{t=T_o+1}^{T_o+T_f} \log\left(\mathcal{N}_{\mathbf{p}_t^n}(\hat{\mu}_t^n, \hat{\Sigma}_t^n)\right), \tag{2}$$

where N is the number of present objects.

In contrast to previous works, our proposed network not only estimates the future trajectory $\mathcal{T}_f^n$, but also reconstructs the observed trajectory $\mathcal{T}_o^n$ and refines them both as a single trajectory as it is illustrated in Fig. 2. This helps to learn a better representation that is spatially and temporally consistent as we will show in the experiments. For the observed trajectory $\mathcal{T}_o^n$, we use the squared error as reconstruction loss:

$$L_o = \sum_{n=1}^{N} \sum_{t=1}^{T_o} \left\|\hat{\mathbf{p}}_t^n - \mathbf{p}_t^n\right\|^2, \tag{3}$$

where $\hat{\mathbf{p}}_t^n$ is the reconstructed position for object n at frame t and $\mathbf{p}_t^n$ is the observed position.

For training the network, we sum the two loss functions Equ. (2) and Equ. (3) with equal weights, *i.e.,*

$$L = L_f + wL_o. \tag{4}$$

where $w = 1$. For the ablation study, however, we also investigate different values of w.

3.1. Spatial Graph Representation

As in other graph-based approaches [15, 34, 24], we use a graph structure to model the spatial relations between the objects. Since we focus on the temporal modeling, we use for a fair comparison the graph representation that has been proposed in [24]. At each timestamp t, a spatial graph $G_t = (V_t, E_t)$ is constructed where $V_t = \{v_t^n \mid n \in \{1, ..., N\}\}$ are the vertices corresponding to the N objects and $E_t = \{e_t^{nm} \mid n, m \in \{1, ..., N\}\}$ denote the edges that encode the spatial relationship between object n and m. For each edge, the affinity a_t^{nm} is computed by

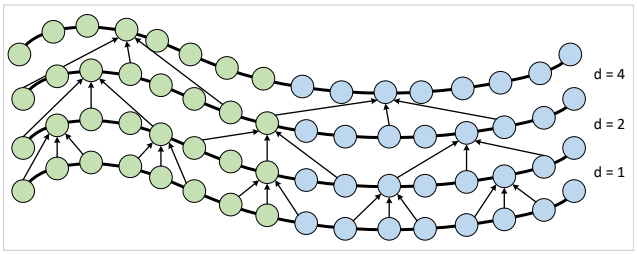


Figure 3: We model the observed trajectories (green) and forecast trajectories (blue) together by using dilated temporal convolutions that are stacked with increasing dilation rates. This provides a very strong temporal model for trajectories since the motion is modeled over a large temporal receptive field.

$$a_t^{nm} = \begin{cases} \left\|\mathbf{p}_t^n - \mathbf{p}_t^m\right\|^{-1} & \text{, if } \left\|\mathbf{p}_t^n - \mathbf{p}_t^m\right\| > 0 \\ 0 & \text{, otherwise.} \end{cases} \tag{5}$$

This means that the objects influence each other when they are close. Suppose $V^{(l)}$ is the stack of $V_t^{(l)}$, that is the vertices at time step t and network layer l. The features of the vertices $f(V^{(l)})$ are updated in a graph convolution layer as follows:

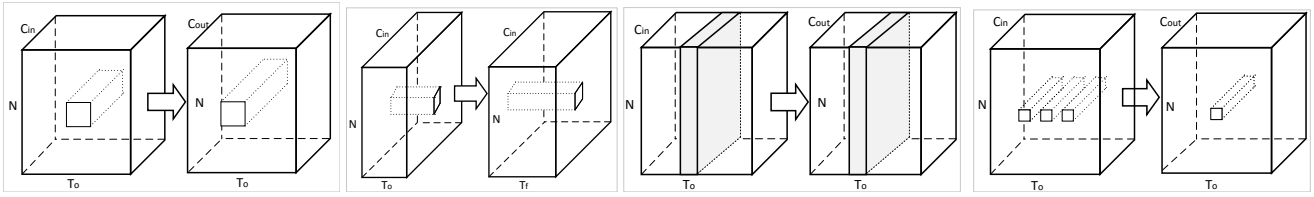
$$f\left(V^{(l+1)}\right) = ReLU\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} f\left(V^{(l)}\right) W^{(l)}\right) \tag{6}$$

where $W^{(l)}$ are the learnable convolution weights, $\tilde{A}$ is the stack of $\tilde{A}_t = A_t + I$, and the diagonal matrix $\tilde{D}$ is the stack of $\tilde{d}_t^{nn} = \sum_m \tilde{a}_t^{nm}$.

While Equ. (6) describes the layer for a single frame t, some works simply stack the features and matrices for multiple frames in order to extend the graph to the temporal domain. This, however, results in a poor temporal model that generates noisy trajectories which are not very consistent over time as we show in the experiments. In this work, we therefore propose to use dilated temporal and feature-wise convolutions for forecasting trajectories and combine them with graph convolutional layers. In the experiments, we will show that this results in forecast trajectories that are spatially and temporally more consistent.

3.2. Architecture

In order to forecast spatially and temporally consistent trajectories, we propose to combine different convolutions as shown in Fig. 4. While graph convolutions model the relations of objects that are close, they are not the best choice for temporal modeling as discussed in Section 3.1. We therefore propose graph convolutions for spatial modeling and



(a) Vanilla 1×1 Convolution (b) Feature-wise Convolution (c) Graph Convolution (d) Dilated Temporal Convolution

Figure 4: The four different convolutions that are used in STC-Net. For the illustrations, the input tensor is (C_{in}, T_o, N) where T_o are the observed frames, N denotes the number of objects, and C_{in} is the dimensionality of the input features. (a) Vanilla 1×1 convolutions map the feature to another space with dimensionality C_{out}. The output tensor is thus (C_{out}, T_o, N). (b) In contrast, the proposed feature-wise convolutions keep the feature dimensionality but adjust the temporal dimension. Since the predicted length T_f is variable, it is suitable for forecasting and we use them to forecast the future features. The output tensor is (C_{in}, T_f, N). (c) Graph convolutions aggregate the features at a frame based on the spatial relations of the objects and they are used to model interactions between objects. Depending on the number of kernels, the feature dimensionality can change and the output tensor is (C_{out}, T_o, N). (d) Temporal convolutions operate over time and aggregate for each trajectory temporal information. Due to dilated convolution kernels, the temporal convolutions can take the entire trajectory into account. While graph convolutions model spatial relations, the temporal convolutions model temporal relations.

dilated temporal convolutions for temporal modeling. In contrast to graph convolutions that only consider neighbors based on the affinity matrix A, dilated temporal convolutions with gradual increasing dilation rates capture long range dependencies as it is illustrated in Fig. 3.

For our network, which is shown in Fig. 2, we first aggregate spatial information among different observed trajectories at each frame using graph convolutions. We then aggregate temporal information using dilated temporal convolutions [18]. We use 3 dilated temporal convolutions with kernel size 3 and dilation rates 2^{l-1}, where l is the layer number, and apply it to each trajectory. Finally, we use a graph convolution to aggregate again the spatial information. In this way, the network is able to consider the interaction of trajectories that are in close proximity as well as the full temporal information that is available for each trajectory.

After the interaction of both spatial and temporal information, the initial trajectory prediction is generated by a feature-wise convolution, which is illustrated in Fig. 4. It is similar to a 1×1 convolution, but the main difference is the information aggregation dimension. At this stage of the network, the size of the feature tensor is (C_{in}, T_o, N) where C_{in} is the feature dimension, T_o is the length of the observed trajectories, and N is the number of objects in the current scene. While a vanilla 1×1 convolution changes the feature dimensionality and aggregates the information across the features, a feature-wise convolution aggregates for each channel in the feature vector all observed information along the temporal dimension and can be used to predict the future for each channel with variable length. After the feature-wise convolution, the size of the feature tensor is (C_{in}, T_f, N) where T_f is the length of the future trajectories. In contrast to recurrent or auto-regressive approaches, the feature-wise convolution predicts the future in one pass.

Different from previous works that only predict the future trajectories, we also reconstruct the observed trajectories to ensure that the observed and forecast part of a trajectory are consistent. As shown in Fig. 2, we refine both parts as a single consistent trajectory. Specially, the features of the observed and forecast trajectories are concatenated and refined by dilated temporal convolutions as before. The only difference is that we use 5 layers since the concatenated trajectory is longer than the observed one. For training, we use the loss Equ. (4).

3.3. Network Architecture Search

The architecture shown in Fig. 2 is one instance of spatial-temporal consistency networks. Depending on the number of layers, we obtain different instances. While we evaluate the impact of each component of the network as part of the ablation studies, we additionally perform a network architecture search to analyze which instance performs best. For the search algorithm, we adopt ENAS [27] which is a one-shot search algorithm. As search space, we allow different types of convolutions (temporal or graph convolution), temporal convolutions with different dilation rates, and a varying depth of the network. ENAS models all combinations as large directed acyclic graph and any valid network is a subgraph of it. A controller is then trained with

$\mathcal{T}_o$	NLL	NLL	L2	L2
$\mathcal{T}_f$	NLL	L2	NLL	L2
ADE / FDE	0.63 / 1.09	1.00 / 1.95	**0.51 / 0.85**	0.93 / 1.79

Table 1: Impact of different loss terms for the reconstructed ($\mathcal{T}_o$) and forecast ($\mathcal{T}_f$) trajectory. While NLL denotes the negative log-likelihood in Equ. (2), L2 denotes the L2 loss in Equ. (3).

w	0.5	0.7	1.0	1.3	w/o obs	w/o refine
ADE / FDE	0.55 / 0.91	0.54 / 0.87	**0.51 / 0.85**	0.54 / 0.91	0.58 / 0.97	0.60 / 1.04

Table 2: Impact of w in Equ. (4). While 'w/o obs' denotes a setting where the reconstructed observed trajectories are not concatenated with the forecast trajectories, 'w/o refine' uses the forecast trajectories directly after the feature-wise convolution without the last layer of dilated temporal convolutions.

	w/o GC	w/o DTC	w/o FWC	Ours
ADE	0.64	0.59	0.55	**0.51**
FDE	1.02	0.99	0.91	**0.85**
Latency	0.7ms	1.4ms	2.3ms	1.5ms

Table 3: Impact of the different types of convolutions.

policy gradient to select a subgraph that maximizes the expected reward on the validation set. In order to accelerate the search process, ENAS applies a parameter sharing strategy between child models.

4. Experiment

4.1. Datasets

We evaluate our method on three common trajectory forecasting datasets which are described below. As in previous works, 8 frames are used as observation and the following 12 frames need to be forecast. All trajectories are provided in 2D coordinates.

The **ETH** [26] and **UCY** [16] datasets aim at pedestrian trajectory forecasting. ETH contains 2 top view scenes with 750 trajectories and UCY contains 3 scenes close to the top view with 786 trajectories. The data processing is done as [13]. The sampling rate is 2.5 frames per second (one frame per 0.4s), which means that the model observes 3.2 seconds and predicts the trajectories for the next 4.8 seconds.

The **Stanford Drone Dataset** [28] includes a series of videos captured by a hovering drone from 8 different college campuses with more than 10,000 trajectories. The dataset is much larger than ETH and UCY and includes other objects like bikes or cars apart from pedestrians. The frame rate is 2.5 frames per second, which is the same as for ETH and UCY. We evaluate our approach as setting [25] for two protocols, namely 2D world and 2D image coordinates.

4.2. Metrics

There are two commonly used metrics to evaluate the performance of trajectory forecasting methods: average displacement error (ADE) [26] and final displacement error (FDE) [1]. The average displacement error is defined as:

$$ADE = \frac{\sum_{n=1}^{N} \sum_{t=T_o+1}^{T_o+T_f} \left\| \hat{\mathbf{p}}_{\mathbf{t}}^{\mathbf{n}} - \mathbf{p}_{\mathbf{t}}^{\mathbf{n}} \right\|_2}{N \times T_f}, \tag{7}$$

where $\hat{\mathbf{p}}_{\mathbf{t}}^{\mathbf{n}}$ is the estimated position of object n at frame t and $\mathbf{p}_{\mathbf{t}}^{\mathbf{n}}$ is the ground-truth position. It measures the average prediction performance along the whole trajectory. The final displacement error (FDE) is defined as:

$$FDE = \frac{\sum_{n=1}^{N} \left\| \hat{\mathbf{p}}_{\mathbf{T_o+T_f}}^{\mathbf{n}} - \mathbf{p}_{\mathbf{T_o+T_f}}^{\mathbf{n}} \right\|_2}{N}. \tag{8}$$

In contrast to ADE, it only measures the prediction accuracy at the end point. Because the prediction of our method is a distribution, we follow the evaluation protocol used in Social-LSTM [1] to compare the predicted distribution with the ground truth value. Specially, we sample 20 predictions from the predicted distribution and take the closest sample to the ground truth to compute ADE and FDE.

4.3. Ablation Study

We conduct the ablation study on the Stanford Drone Dataset [28] since it is the largest dataset and has the highest diversity. We first evaluate the proposed loss function Equ. (4), which uses the L2 loss for reconstructed trajectory in Equ. (3) and negative log-likelihood (NLL) for the forecast trajectory in Equ. (2). The results are shown in Tab. 1.

From the table we can see that applying the L2 loss on the reconstructed observed trajectory and the negative log-likelihood on the forecast trajectory achieves the lowest error (col 3). This is reasonable because the observed trajectory is determined, which means that there is no uncertainty in the ground-truth. In contrast, the future trajectory can be uncertain since the future is not necessarily determined and several trajectories might be plausible. If we use the L2 loss for the forecast trajectory (col 2 and 4), the error drastically increases since the L2 loss does not handle the uncertainty of the future. On the other hand, if we use only the negative log-likelihood (col 1), the error is higher than for the proposed setting (col 3). This is expected since the L2 error enforces a more accurate reconstruction of the observed trajectory, which is beneficial for the forecasting task.

As discussed in Sec. 3, we do not weight the two loss terms in Equ. (4). Nevertheless, we evaluate in Tab. 2 the impact of an additional weighting parameter w. We can see that the error first decreases as w increases to 1.0 and then increases again when w is larger than 1.0. This shows that there is no need to weight the two loss functions. In the second to last column, we show why it is important to take the observed trajectory for the final estimate into account. If we do not concatenate the reconstructed observed and the forecast trajectories (w/o obs) but refine only the forecast trajectory, the error increases. If we do not refine the trajectories at all (w/o refine), the error even increases further. This shows the importance of the additional refinement layers.

	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
S-LSTM [1]	1.09 / 2.35	0.79 / 1.76	0.67 / 1.40	0.47 / 1.00	0.56 / 1.17	0.72 / 1.54
PIF [20]	0.73 / 1.65	0.30 / 0.59	0.60 / 1.27	0.38 / 0.81	0.31 / 0.68	0.46 / 1.00
SR-LSTM [36]	0.63 / 1.25	0.37 / 0.74	0.51 / 1.10	0.41 / 0.90	0.32 / 0.70	0.45 / 0.94
DS-LSTM [7]	0.66 / 1.21	0.27 / 0.46	0.50 / 1.07	0.33 / 0.68	0.28 / 0.60	0.41 / 0.80
FvTraj [4]	0.56 / 1.14	0.28 / 0.55	0.52 / 1.12	0.37 / 0.78	0.32 / 0.68	0.41 / 0.85
S-GAN [13]	0.81 / 1.52	0.72 / 1.61	0.60 / 1.26	0.34 / 0.69	0.42 / 0.84	0.58 / 1.18
SoPhie [29]	0.70 / 1.43	0.76 / 1.67	0.54 / 1.24	0.30 / 0.63	0.38 / 0.78	0.54 / 1.15
CGNS [17]	0.62 / 1.40	0.70 / 0.93	0.48 / 1.22	0.32 / 0.59	0.35 / 0.71	0.49 / 0.97
Social-Ways [2]	0.39 / 0.64	0.39 / 0.66	0.55 / 1.31	0.44 / 0.64	0.51 / 0.92	0.46 / 0.83
Goal-GAN [9]	0.59 / 1.18	0.19 / 0.35	0.60 / 1.19	0.43 / 0.87	0.32 / 0.65	0.43 / 0.85
TPNet [11]	0.84 / 1.73	0.24 / 0.46	0.42 / 0.94	0.33 / 0.75	0.26 / 0.60	0.42 / 0.90
PECNet [23]	**0.54 / 0.87**	**0.18 / 0.24**	**0.35 / 0.60**	**0.22 / 0.39**	**0.17 / 0.30**	**0.29 / 0.48**
STSGN [35]	0.75 / 1.63	0.63 / 1.01	0.48 / 1.08	0.30 / 0.65	0.26 / 0.57	0.48 / 0.99
Social-BiGAT [15]	0.69 / 1.29	0.49 / 1.01	0.55 / 1.32	0.30 / 0.62	0.36 / 0.75	0.48 / 1.00
RSBG [30]	0.80 / 1.53	0.33 / 0.64	0.59 / 1.25	0.40 / 0.86	0.30 / 0.65	0.48 / 0.99
Social-STGCNN [24]	0.64 / 1.11	0.49 / 0.85	0.44 / 0.79	0.34 / 0.53	0.30 / 0.48	0.44 / 0.75
STGAT [14]	0.65 / 1.12	0.35 / 0.66	0.52 / 1.10	0.34 / 0.69	0.29 / 0.60	0.43 / 0.83
STC-Net	0.66 / 1.28	0.33 / 0.58	0.41 / 0.78	0.29 / 0.51	0.27 / 0.45	0.39 / 0.72
STC-Net-NAS	0.64 / 1.18	0.33 / 0.54	0.39 / 0.74	0.29 / 0.49	0.26 / 0.45	0.38 / 0.68

Table 4: Results on the ETH [26] and UCY [16] datasets.

	ADE	FDE
Social LSTM$^+$ [1]	0.73	0.96
Social-STGCNN$^+$ [24]	0.71	1.14
Social-Ways [2]	0.62	1.16
STGAT [14]	0.58	1.11
STGAT$^+$ [14]	0.63	1.19
DAG-Net [25]	0.54	1.05
DAG-Net* [25]	0.53	1.02
STC-Net	0.51	0.85
STC-Net-NAS	**0.49**	**0.82**

Table 5: Results on the Stanford Drone dataset (world coordinates) [28]. $^+$ means that we train the model with the code offered by the authors. * denotes that we use the pre-tained model offered by the authors.

	ADE	FDE
SoPhie [29]	16.27	29.38
Social GAN [13]	27.23	41.44
CF-VAE [3]	12.6	22.3
P2TIRL [10]	12.58	22.07
SimAug$^+$ [19]	10.27	19.71
PECNet [23]	**9.96**	**15.88**
STC-Net	11.96	20.12
STC-Net-NAS	11.88	20.08

Table 6: Results on the Stanford Drone dataset (image coordinates) [28]. $^+$ the approach uses additional training data.

In Tab. 3, we analyze the impact of the used convolutions on the latency as well as accuracy. Since we cannot simply remove the convolutions, we replaced them by other operations. We used pooling instead of graph convolutions (w/o GC), temporal convolutions instead of dilated temporal convolutions (w/o DTC), and the TXP-CNN of Social-STGCNN [24] instead of the feature-wise convolution (w/o FWC). The results show that both graph convolutions and dilated temporal convolutions are required in order to achieve accurate predictions. It also shows that the graph convolutions take more than 50% of the processing time. Furthermore, the proposed feature-wise convolution is much more efficient and effective compared to TXP-CNN.

4.4. Comparison with state-of-the-art methods

Accuracy. We compare our method with other state-of-the-art methods on the ETH [26], UCY [16], and Stanford Drone [28] datasets. The results for the ETH [26] and UCY [16] datasets are shown in Tab. 4. When we compare our approach to other graph-based methods [35, 15, 30, 24, 14], we observe that our approach achieves the lowest average (ADE) and final displacement error (FDE) among graph-based methods. When we compare our approach to the state-of-the-art, we observe that only PECNet [23] achieves a lower error. PECNet uses a two step approach that first predicts the end-points and then the trajectories based on the predicted end-points. While predicting end-points is complementary to our approach, the high accuracy comes at high computational cost as shown in Tab. 9. PECNet requires 3,000 times more parameters and is 467 times slower. In fact, a latency of over 600ms is too high for applications.

As discussed in Sec. 3.3, we also generated an instance of the spatial-temporal consistency networks by network architecture search. It needs to be noted that we used only the Stanford Drone Dataset [28] for the network optimization since it is the largest dataset. If we use this instance, which

	0.4s	0.8s	1.2s	1.6s	2.0s	AVG
Social-STGCNN [24]	0.71 / 1.15	0.73 / 1.14	0.69 / 1.10	0.61 / 0.92	0.67 / 1.01	0.68 / 1.06
DAG-Net [25]	0.53 / 1.02	0.53 / 1.01	0.58 / 1.14	0.56 / 1.11	0.82 / 1.64	0.60 / 1.18
STGAT [14]	0.63 / 1.19	0.64 / 1.17	0.61 / 1.14	0.51 / 0.92	0.57 / 1.05	0.59 / 1.09
STC-Net	0.51 / 0.85	0.50 / **0.80**	0.50 / 0.83	0.43 / 0.69	0.50 / 0.80	0.49 / 0.79
STC-Net-NAS	**0.49 / 0.82**	**0.49 / 0.80**	**0.48 / 0.82**	**0.41 / 0.67**	**0.47 / 0.78**	**0.47 / 0.78**

Table 7: Temporal robustness. The errors (ADE / FDE) for different sampling rates on the Stanford Drone dataset [28].

	ETH	HOTEL	UNIV	ZARA1	ZARA2	AVG
Social-STGCNN [24]	0.64 / **0.88**	0.44 / 0.67	0.50 / 0.79	0.53 / 0.71	0.48 / 0.70	0.52 / 0.75
STGAT [14]	0.71 / 1.31	0.33 / 0.61	0.61 / 1.26	0.43 / 0.83	0.38 / 0.75	0.49 / 0.95
DAG-Net [25]	0.71 / 1.23	0.22 / 0.41	0.70 / 1.47	0.43 / 0.88	0.29 / 0.60	0.47 / 0.92
STC-Net	0.61 / 1.10	**0.20** / 0.27	0.43 / 0.79	0.35 / 0.59	0.28 / 0.47	0.37 / 0.64
STC-Net-NAS	**0.59** / 1.12	**0.20 / 0.25**	**0.40 / 0.75**	**0.33 / 0.55**	**0.27 / 0.46**	**0.36 / 0.63**

Table 8: Cross-scene robustness. For this experiment, the approaches are trained on the Stanford Drone dataset [28] and evaluated on the ETH [26] and UCY [16] datasets.

	Parameters	Inference time
PECNet [23]	2.10M (3000x)	0.6070s (467x)
Social LSTM [1]	264K (377.1x)	1.1800s (907.7x)
SR-LSTM [36]	64.9K (92.7x)	1.1789s (906.8)
STGAT [14]	44.6K (63.7x)	1.3497s (1038.2x)
DAG-Net [25]	2.35M (3357.1x)	0.0463s (35.6x)
PIF [20]	360.3K (514.7x)	0.1145s (88.1x)
Social GAN [13]	46.3K (66.1x)	0.0968s (74.5x)
Social-STGCNN [24]	7.6K (11x)	0.0020s (1.5x)
STC-Net	0.8K (1.1x)	0.0015s (1.2x)
STC-Net-NAS	**0.7K**	**0.0013s**

Table 9: Model size and efficiency. Only [25, 20, 13, 24] achieve a latency that is lower than the frame-rate.

L	CH	ADE	FED	Params	Inf time
3	1×	11.96	20.12	0.8K	0.0015s
6	1×	11.61	18.79	1.0K	0.0017s
3	2×	11.65	19.79	2.6K	0.0015s
3	4×	11.49	19.04	10.0K	0.0015s
6	4×	**11.34**	**18.65**	13.6K	0.0017s

Table 10: Changing the size of the model. From top to bottom, we show the results of the original model, a deeper model, a wider model, and finally a combination of them. L denotes number of layers and CH means increase of number of channels, i.e., K× means increasing the number of channels by K times compared to the original model.

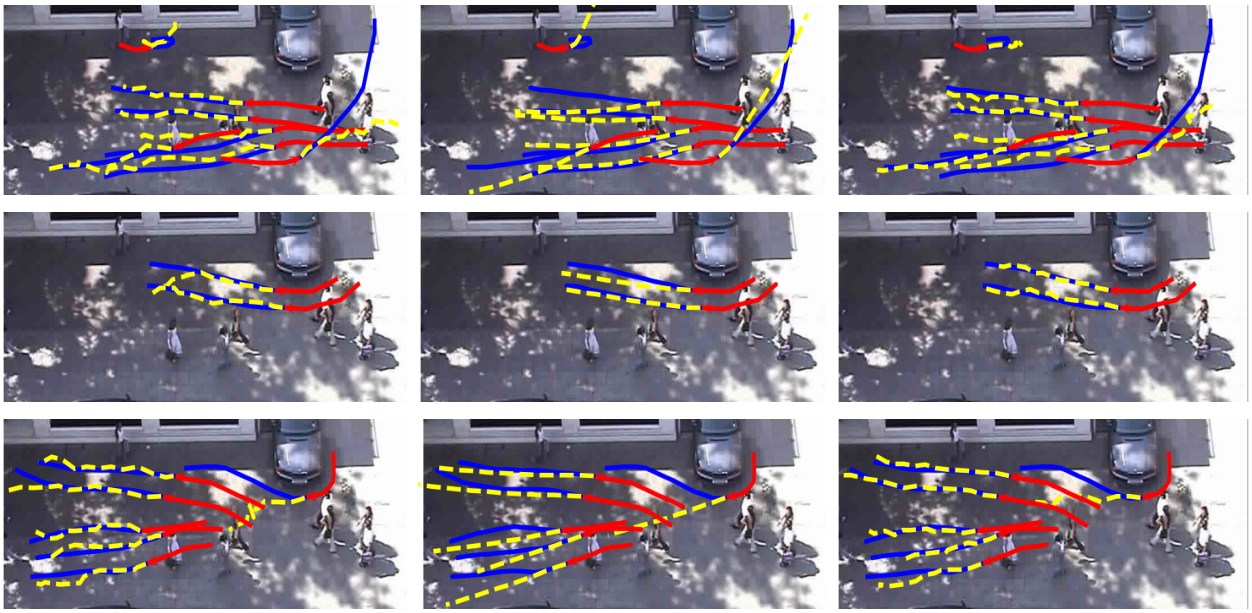
is denoted by STC-Net-NAS, for the other two datasets, we observe that the error is slightly reduced.

The results for the Stanford Drone dataset [28] are reported in Tab. 5 and Tab. 6, respectively. The results confirm the very good results from ETH and UCY. Our approach achieves a much lower error compared to other graph-based approaches. Only the recent approaches [19, 23] achieve a lower accuracy. While PECNet [23] uses a much bigger network and is much slower than our approach as discussed before, the numbers of [19] are not comparable since the approach uses additional training data.

Efficiency. Since methods for forecasting trajectories need to be compact and to have a very low inference time, we compare the size and inference time of STC-Net with other state-of-the-art methods in Tab. 9. For measuring the inference time, we used a single 1080Ti GPU. In contrast to previous works, our approach is very compact and comprises less than 1K parameters. From the state-of-the-art methods, only Social-STGCNN [24] achieves a very low inference time. STC-Net, however, not only achieves a much lower forecasting error than Social-STGCNN, it is even more compact and has a lower latency. It is interesting to note that the neural network search finds an instance that not only makes more accurate predictions, but it also reduces the model size and inference time. STC-Net-NAS runs with more than 750 frames per seconds on a low-budget GPU, yielding a latency of less than 2ms.

Furthermore, we evaluated on the Stanford Drone dataset (image coordinates) the impact of the model size by increasing the number of dilated convolutional layers before the feature-wise convolution and/or the number of channels. The results are shown in Tab. 10. The results show that the error can be further reduced by adding more layers or increasing the number of channels per layer at the cost of increasing the number of parameters or the inference time. Note that for all configurations the approach is still faster than previous works.

Temporal Robustness. We used so far a sampling rate of 2.5 frames per second, which corresponds to one frame every 0.4 seconds, for training and inference. In Tab. 7, we report the results when we use a lower sampling rate during inference while keeping the number of frames the same. When the sampling rate is higher, we simulate the case when the objects move faster than observed during training. Note that the numbers for different sampling rates are not directly comparable since the sampled trajectories also differ for the different sampling rates, but they indicate how robust the methods are. The results show that our and the other graph-



(a) Social-STGCNN [24] (b) STGAT [14] (c) STC-Net

Figure 5: Qualitative results. The **red** line is the observed trajectory, the **blue** line is the ground-truth of the future trajectory and the **yellow** dashed line is the prediction.

based approaches [24, 14] are very robust and can handle objects that are 5 times faster than the ones in the training set. Only for [25], we observe a drastic increase of the error for 2.0 seconds.

Cross-scene Robustness. In Tab. 8, we evaluate how robust the methods are across different scenes. This means that we train the approaches on the Stanford Drone dataset [28] and evaluate them on the ETH [26] and UCY [16] datasets. We use the Stanford Drone dataset for training since it is the largest dataset and contains not only pedestrians but also other objects. Compared with Tab. 4, the average error of Social-STGCNN [24] and STGAT [14] is only slightly higher. While the error increases for most scenes, it even decreases for the HOTEL scene. This is reasonable since the Stanford Drone dataset provides a larger dataset and the trajectories of the HOTEL scenes are relatively simple. This shows that these methods are quite robust to scene changes. Our approach, however, performs even better and the error even slightly decreases in average. This shows that our approach models the motion very well, is very robust to changes of the sampling rate, and generalizes to new scenes.

Qualitative Analysis. We show some qualitative results in Fig. 5 and compare our approach to the graph-based approaches Social-STGCNN [24] and STGAT [14]. From the images, we can see that the predictions of Social-STGCNN are not smooth and some trajectories even cross which is implausible. The predictions of STGAT are nearly linear, which results in larger errors when the direction changes. In contrast, our method forecasts more plausible trajectories.

5. Conclusion

In this paper, we proposed a highly efficient forecasting model that generates spatially and temporally consistent trajectories. The network utilizes graph and dilated temporal convolutions to model the spatial and temporal relations of each trajectory. Furthermore, a feature-wise convolution is utilized to forecast trajectories in one pass, and the reconstructed observed and forecast trajectories are jointly refined. By using a neural network architecture search, the network is further optimized. The proposed approach outperforms most other methods in terms of accuracy, is very compact with 0.7K parameters, and achieves a very low latency of 1.3ms. In our evaluation, we also demonstrate that the approach is robust to changes in the sampling rate and to scene variations. This makes the proposed approach perfectly suitable for realistic applications.

Acknowledgement The work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - GA 1927/5-2 (FOR 2535 Anticipating Human Behavior), under Germany's Excellence Strategy - EXC 2070 - 390732324, and the ERC Starting Grant ARCA (677650).

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. So-

cial lstm: Human trajectory prediction in crowded spaces. In *CVPR*, pages 961–971, 2016.

[2] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. In *CVPR Workshops*, pages 0–0, 2019.

[3] Apratim Bhattacharyya, Michael Hanselmann, Mario Fritz, Bernt Schiele, and Christoph-Nikolas Straehle. Conditional flow variational autoencoders for structured sequence prediction. *arXiv preprint arXiv:1908.09008*, 2019.

[4] Huikun Bi, Ruisi Zhang, Tianlu Mao, Zhigang Deng, and Zhaoqi Wang. How can i see my future? fvtraj: Using first-person view for pedestrian trajectory prediction. In *ECCV*, 2020.

[5] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[6] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[7] QJ Chaofan Tao and P Luo. Dynamic and static context-aware lstm for multi-agent motion prediction. In *ECCV*, 2020.

[8] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In *NIPS*, pages 6642–6652, 2019.

[9] Patrick Dendorfer, Aljoša Ošep, and Laura Leal-Taix'e. Goal-gan: Multimodal trajectory prediction based on goal position estimation. In *ACCV*, 2020.

[10] Nachiket Deo and Mohan M Trivedi. Trajectory forecasts in unknown environments conditioned on grid-based plans. *arXiv preprint arXiv:2001.00735*, 2020.

[11] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *CVPR*, pages 6797–6806, 2020.

[12] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

[13] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *CVPR*, pages 2255–2264, 2018.

[14] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *ICCV*, pages 6272–6281, 2019.

[15] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. In *NIPS*, pages 137–146, 2019.

[16] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by example. In *Computer graphics forum*, volume 26, pages 655–664. Wiley Online Library, 2007.

[17] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Conditional generative neural system for probabilistic trajectory prediction. *arXiv preprint arXiv:1905.01631*, 2019.

[18] Shi-Jie Li, Yazan Abu Farha, Yun Liu, Ming-Ming Cheng, and Juergen Gall. MS-TCN++: Multi-stage temporal convolutional network for action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[19] Junwei Liang, Lu Jiang, and Alexander Hauptmann. Simaug: Learning robust representations from 3d simulation for pedestrian trajectory prediction in unseen cameras. *arXiv preprint arXiv:2004.02022*, 2020.

[20] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *CVPR*, pages 5725–5734, 2019.

[21] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, pages 19–34, 2018.

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[23] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. In *ECCV*, pages 759–776. Springer, 2020.

[24] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *CVPR*, pages 14424–14432, 2020.

[25] Alessio Monti, Alessia Bertugli, Simone Calderara, and Rita Cucchiara. Dag-net: Double attentive graph neural network for trajectory forecasting. *arXiv preprint arXiv:2005.12661*, 2020.

[26] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *ICCV*, pages 261–268. IEEE, 2009.

[27] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[28] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, pages 549–565. Springer, 2016.

[29] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *CVPR*, pages 1349–1358, 2019.

[30] Jianhua Sun, Qinhong Jiang, and Cewu Lu. Recursive social behavior graph for trajectory prediction. In *CVPR*, pages 660–669, 2020.

[31] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019.

[32] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[33] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*, 2020.

[34] Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *ICLR*, 2018.

[35] Lidan Zhang, Qi She, and Ping Guo. Stochastic trajectory prediction with social graph network. *arXiv preprint arXiv:1907.10233*, 2019.

[36] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *CVPR*, pages 12085–12094, 2019.

[37] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[38] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018.