Predicting User Routines with Masked Dilated Convolutions

Renzhong Wang
Microsoft
Bellevue, WA
rewan@microsoft.com

Dragomir Yankov
Microsoft
Sunnyvale, CA
dragoy@microsoft.com

Michael R. Evans
Microsoft
London, UK
mievans@microsoft.com

Senthil Palanisamy
Microsoft
Bellevue, WA
senthilp@microsoft.com

Siddhartha Arora
Microsoft
Bellevue, WA
sidaro@microsoft.com

Wei Wu
Microsoft
Bellevue, WA
weiwu@microsoft.com

ABSTRACT

Predicting users daily location visits - when and where they will go, and how long they will stay - is key for making effective location-based recommendations. Knowledge of an upcoming day allows the suggestion of relevant alternatives (e.g., a new coffee shop on the way to work) in advance, prior to a visit. This helps users make informed decisions and plan accordingly.

People's *visit routines*, or just *routines*, can vary significantly from day to day, and visits from earlier in the day, week, or month may affect subsequent choices. Traditionally, *routine prediction* has been modeled with sequence methods, such as HMMs or more recently with RNN-based architectures. However, the problem with such architectures is that their predictive performance degrades when increasing the number of historical observations in the routine sequence. In this paper, we propose Masked-TCN (MTCN), a novel method based on time-dilated convolutional networks. The method implements *custom dilations* and *masking* which can process effectively long routine sequences, identifying recurring patterns at different resolution - hourly, daily, weekly, monthly. We demonstrate that MTCN achieves 8% improvement in accuracy over current state-of-the-art solutions on a large data set of visit routines.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**;

KEYWORDS

Location Context; Sequence Prediction; Location Recommendation

ACM Reference Format:
Renzhong Wang, Dragomir Yankov, Michael R. Evans, Senthil Palanisamy, Siddhartha Arora, and Wei Wu. 2019. Predicting User Routines with Masked Dilated Convolutions. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3347025

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347025

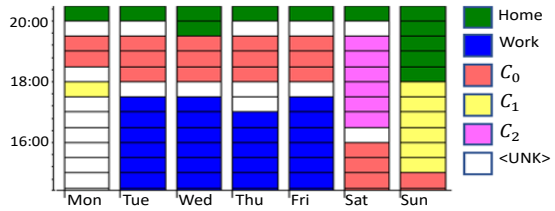


Figure 1: Visualization of a partial routine. Colors depict visits to different locations during the corresponding times.

1 INTRODUCTION

People's lives are filled with routine - we do similar activities during similar times of the day, week, or month: on business days we commute to work or school, on certain days we go to gym; weekends we visit favorite locations (Figure 1[1]).

We call the process of detecting patterns in the sequence of location visits for a given user *Routine Inference*, and the process of using the inferred patterns to predict future visits *Routine Prediction*. Accurate routine prediction is essential for many location-aware mobile apps and digital assistants. They use routine predictions to power multiple applications, with *location-based recommendations* being one of the most important among them.

Location-Based Recommendations (LBR) [8, 10, 13, 15] utilize user interests and location context to suggest events or alternate venues. Some systems focus on group-based recommendations, e.g., when are certain locations more likely to be visited by groups of people[13]. As expected, longer histories of personalized geographic and temporal signals have been shown to improve the effectiveness of LBR [8]. An important aspect in sequence and time-aware recommendations [9] is detecting when users will be most receptive to suggestions. In LBR, one of the primary factors for triggering user response is whether it will fit into the anticipated user routine. For instance, if we recommend that a user visits a restaurant on a day which they are likely to be busy with work, then most likely this recommendation will be discarded and create a negative user experience.

Hidden Markov Models (HMMs) have been the traditional approach to modeling routine prediction [1]. More recently, researchers have focused on RNN-based architectures. In particular, attention-based sequence-to-sequence (S2S) models were applied by [5, 6].

[1]User data depicted displayed with user consent.

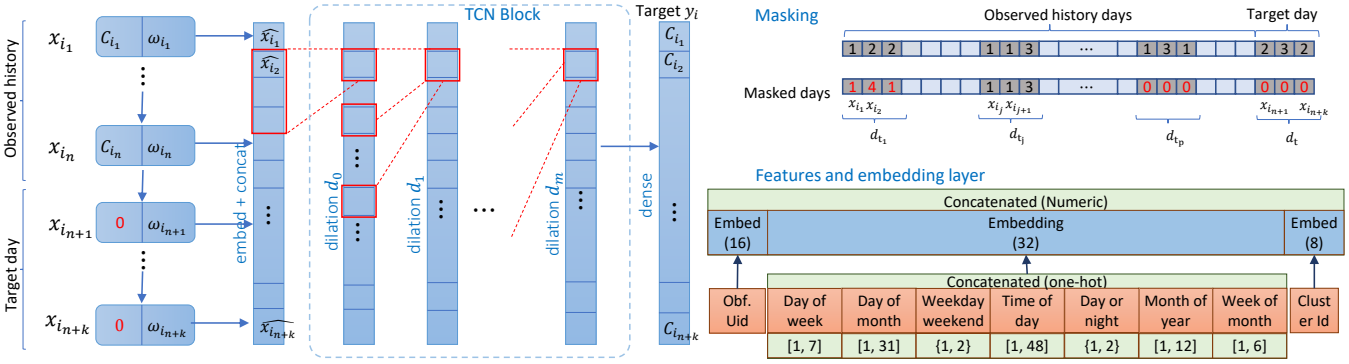


Figure 2: MTCN architecture for routines - embedding followed by custom dilation layers. Right top: example of masked sequence. Right bottom: state features and embedding used in the first layer of the network.

Both works show improvement in certain cases over the HMM approaches, with [5] also stating that the attention layer is suitable to capture patterns at varying time resolutions. The presented results, however, have been inconclusive, with the authors of [6] stating that "the S2S model is better than the LSTM model, with the Markov model slightly outperforming both of them" on one task.

Recently, [2] studied the degrading performance of RNN-based sequence models in the case of long sequence lengths, even when using attention mechanisms. The authors also point out the tendency of these methods to converge to trivial dominant state predictions. They proposed an alternative method based on *Time-dilated Convolutional Networks* (TCNs). Kernel dilation in TCNs create an exponential receptive field with just a few layers, making them suitable for handling very long sequences. We propose a modification of TCN which has two distinct features - *custom dilations* and *masking*. Kernel dilations, when customized correctly, are shown to capture variable-length recurrences - things that we do hourly, daily, weekly, or monthly, while *masking* [4] is used to improve the generalization performance of the network.

2 REPRESENTATION AND FORMULATION

Routine prediction has two core elements - spatial and temporal. We want to predict *where* users are likely to go and *when*. These components are reflected in the two concepts: *locations* and *visits*.

Locations (also *Location Clusters*) are dense clusters of point data from GPS traces observed for a user. In some cases, locations can be uniquely mapped, e.g., through a reverse geocoding service [11], to a *semantic geo-entity*, such as park, school, or restaurant. The GPS points are clustered using algorithms such as time-augmented DBSCAN [3], which takes into account both the geo-spatial and the temporal proximity of the points. The noise associated with GPS readings often makes it difficult to identify accurate clusters, and even more so to map them uniquely to a semantic location [14].

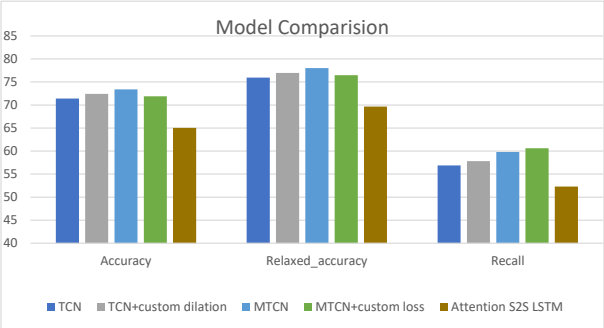
Visits are associated with locations. A visit to a location has a start time and a duration. Detecting visits is also difficult as not all locations have GPS coverage or because devices may fall into low power mode and therefore not send any GPS signals in an effort to save battery life. Due to the above inaccuracies, associated with both location and visit identification, inferring routines can often be challenging.

Table 1: Obfuscated location visit logs.

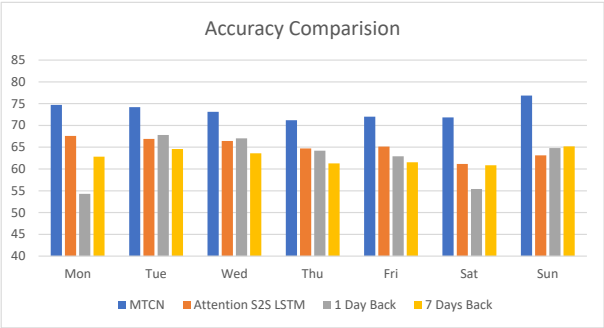
Obf. Uid	ClusterId C_i	Date/Time t_i	Duration
uid1	*Home*	2019-01-01 00:30 am	380 min
uid1	C_0	2019-01-01 8:10 am	22 min
uid1	*Work*	2019-01-01 8:40 am	450 min
uid1	C_7	2019-01-01 6:15 pm	112 min

Routine logs: The data used in this paper is assumed to have both locations and visits inferred, within some reasonable error. The system removes all user and raw identifiable information (e.g., all latitude/longitude information is removed). It generates meta logs as per Table 1, which we use for modeling the routine prediction problem. In our data, some users have the option of assigning labels to their location clusters (e.g., *Home*, *Work*). There are different ways to formalize the routine prediction problem. One approach could be to use the entries in Table 1 directly to define sequence states and predict future entries in the logs table. In this approach the sequences can be compressed and therefore relatively short. This representation, however, requires structured prediction of next visit, i.e. the tuple (C_i, t_i). Defining loss functions for such targets is not intuitive because they require trade-off between two heterogeneous components - location and time.

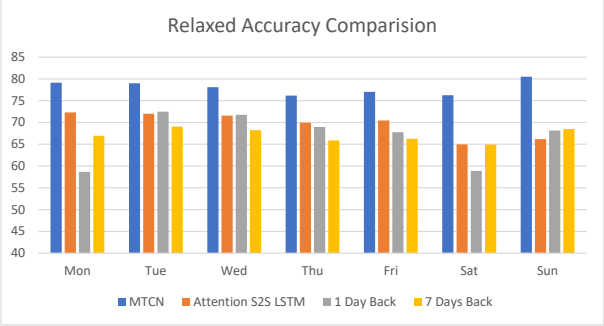
Instead, we opt for denser but easier to work with and more intuitive representation. A day is divided into regular small intervals, e.g. of half an hour each as shown in Figure 1. Every interval becomes a state in the sequence. It is labeled with the location of the user at that time (or the location with the longest duration during that time interval, if more than one location was visited). If no location is known at a particular interval, it is labeled with the special location <*UNK*> (Unknown). The location logs get transformed into a sequence i with states: $\{x_{i_j} = (C_{i_j}, \omega_{i_j})\}_{j=1}^{n}$, where C_{i_j} is the location cluster for the j-th interval, and $\omega_{i_j} = (u_i, t_{i_j}, \ldots)$ is a context vector having features capturing the obfuscated user id u_i and different temporal aspects associated with the j-th interval, such as: day of month, day of week, business day, weekend, time of day (Figure 2 right bottom). Other predictive features, such as the semantic labels (business categories) for the clusters or the



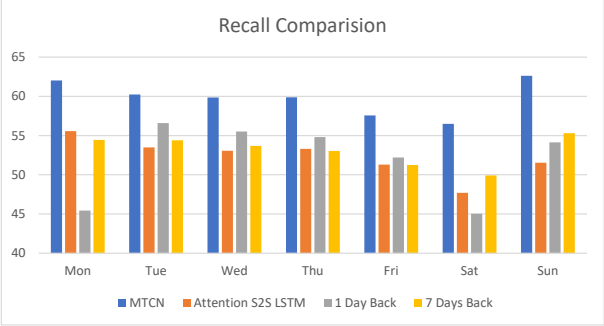
(a) Average accuracy and recall: MTCN vs TCN vs Attention S2S LSTM



(b) Daily accuracy: MTCN vs Attention S2S vs two trivial baselines



(c) Daily relaxed accuracy: MTCN vs Attention S2S vs two trivial baselines



(d) Daily recall: MTCN vs Attention S2S vs two trivial baselines

Figure 3: Evaluation metrics MTCN vs TCN vs Attention S2S and two baselines.

distances between them, can easily be added to the representation if available, but are not used in the current evaluation. The representation is dense and the input sequences can be long, however, the target is a simple categorical label which is easier to learn.

Routine Prediction can now be formalized as the sequence-to-sequence (S2S) problem: Given a dense sequence of regular intervals $\{x_{i_j} = (C_{i_j}, \omega_{i_j})\}_{j=1}^{n}$, predict the location (cluster ID) of the next k regular intervals $\{C_{i_j}\}_{j=n+1}^{n+k}$.

3 ROUTINE PREDICTION WITH MTCN

Bai et al. [2] propose a S2S architecture based on 1D convolutions, the *Time-dilated convolution network (TCN)*[2], and show that it has better accuracy on longer sequences compared to traditional RNN architectures commonly used for S2S problems.

Our approach, *Masked TCN (MTCN)*, has two simple modifications over TCN - custom dilations and masking (Figure 2). The network starts with embedding layer (Figure 2 right bottom), followed by a number of custom dilation layers.

Dilations [16] are increments in the convolution filter (red boxes Figure 2). They allow (M)TCN to handle long sequences. Every dilation layer increases the receptive field in TCN exponentially. In MTCN, the dilation size and convolution filter lengths are set to capture the regularities in user routines. E.g., assuming states representing 30 min, we use dilations of size 1, 2, 4, 8 to detect correlated visits within few hours; 48, 96, 144 for visits at this time the previous or few days ago; 336 for visits at this time a week ago. In the evaluation, custom dilations are shown to predict routines better compared to default power of two dilations.

The idea behind *masking* [4, 12] is simple - perturb a small random sample of the input sequence states, thus achieving a form of regularization and avoiding overfitting. MTCN uses masking in training in the following way (Figure 2 right top): the input has $(n + k)$ states combining the n observed and k target states. The targets are the $(n + k)$ actual cluster ids for these states. In the input, the cluster ids C_{i_j} for the k last states are replaced with *padding* 0. Among the rest, we choose 15% (similar to [4]) of the observed days and mask their states. 80% of the C_{i_j} for the masked days are again replaced with 0, 10% are kept unchanged, and in 10% the cluster id for every visit is randomly replaced by another cluster id. In scoring, the cluster ids for the observed initial n states are kept unchanged and the target k are padded with 0.

Loss Functions. We experiment with two loss functions: *default loss* - cross-entropy with softmax for every interval x_{i_j} in the target and masked days; and *custom loss* - instead of equal weights per interval, it associates equal weights per visit. If there are dominant locations the default loss tends to choose them as safe predictions missing short visits. The custom loss focuses on recalling all visits, as mis-predicting both long and short visits incurs equal penalty.

[2]For detailed description of the method we refer the readers to [2], here we outline only the components related to our modification.

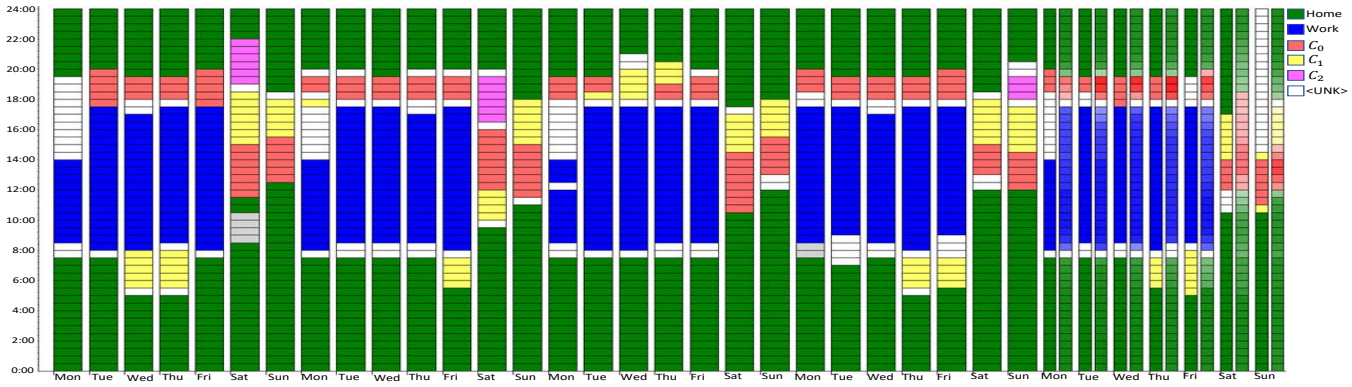


Figure 4: Routine history and prediction (last seven days) for one user. On prediction days, we show both the truth and the prediction (left/right), with the color intensity of the prediction indicating the model's confidence.

4 EXPERIMENTS

Evaluation is done using data logs with structure as of Table 1, unfolded into interval representation as discussed in Section 2. Each sequence state represents a 30 minutes interval. The prediction horizon is set to one day ahead, i.e., $k = 48$ states (24 hours, two intervals per hour). Users that have less than three weeks of history are removed from training and predictions. Also removed are users that are in <UNK> state, more than 50% of the training time. For each user, the last week of data is used for testing, i.e., for a user with data spanning over days $d_1, \ldots, d_t$ we construct a sequence from $d_1, \ldots, d_{t-7}$ to predict d_{t-6}, then $d_1, \ldots, d_{t-6}$ to predict d_{t-5}, etc. For training, we remove the last test week and generate similar sequences where the target is the last but one week of user history. 10% of the users and their sequences are present only in the test set. This imitates the production system where most users are seen before, and only a small fraction of new users appear daily. There are approximately 60K users and 420K sequences in the train set, and 12K users and 84K sequences in the test. The length of the input sequences varies from seven to thirty days.

We compute three measures: *accuracy* - how many intervals are predicted correctly; *relaxed accuracy* - here we assume that an interval is predicted correctly even if the target is shifted by half an hour (e.g. we predict a user will go to work at 9:00 am and instead they go at 9:30 am); and *recall* of visits - sum of fraction of identified visits over the number of all visits in target.

There are five methods compared - TCN, TCN + custom dilation, MTCN, Attention S2S LSTM (AttS2S) [7], and two trivial but very effective baselines - *1 Day Back* and *7 Days Back*. The former baseline simply copies as prediction the last observed day, i.e. we assume people will do tomorrow exactly what they did today. The latter copies as prediction the last observed similar day of week, e.g. we predict on Monday a user will do what they did last Monday.

Figure 3 (a) shows our main result - the previous state of the art, AttS2S, achieves 65.02% average accuracy. In comparison TCN is **71.40%** or **6.38%** improvement, and best of all is MTCN with **73.45%**, i.e. **8.43%** improvement over AttS2S. Adding custom dilations without masking gives 72.42%. The relaxed accuracy for MTCN is 78.03% vs 69.65% AttS2S. Similar are the recall improvements - 60.61% MTCN vs 52.28% AttS2S, i.e. 8.33% improvement. When comparing recall both methods are trained with the custom loss which we can see gives better results for recalling visits compared to default loss (M)TCN.

The metrics per day of week (b)-(d) show a number of interesting effects. Intuitively 1 Day Back does not do well on the boundary of business week and weekend, yet both baselines show that people tend to stick to daily or weekly habits. Though AttS2S is better on average than the two baselines, on certain days it in fact does worse! AttS2S tends to over-predict the dominant state and also does not pick correctly the different periodicity in the data. MTCN does consistently better on all days and all metrics. It is interesting to point out that Sunday through Wednesday are easier to predict. Thursday through Saturday people have more variance in their daily schedules which makes predicting them harder but also opens opportunities for interesting recommendations.

We finish with an example routine for a user (Figure 4[3]). Each column shows the routine for one day, different colors indicating different locations where the user was observed. For the last seven days we show the ground truth (first narrow column) and the prediction (second narrow column). The gradient in prediction shows the confidence with which the location is predicted. We can see how MTCN does quite well, e.g. around actual commute to work times, the method becomes more confident that the user should be at work. It is also able to capture that they visit briefly some other locations (C_0, C_1 and C_2) after work and during the weekend.

5 CONCLUSION

Our proposed method, *Masked TCN (MTCN)*, is a convolution-based approach to finding routine predictions using custom dilations and masking. Via experimental validation it was demonstrated to have better accuracy for routine prediction compared to previous state of the art: Attention S2S recurrent models.

Our current focus is exploring semantic information about the visited locations. Knowing the category of a location (golf course, restaurant, etc.), its working hours, or popular visit times can change significantly our belief whether a user will visit it. It also provides opportunity for more relevant time-wise and category-wise location-based recommendations.

[3]User data depicted displayed with user consent.

REFERENCES

[1] Jorge Alvarez-Lozano, J. Antonio García-Macías, and Edgar Chávez. 2013. Learning and User Adaptation in Location Forecasting. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp '13 Adjunct)*. 461–470.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. (03 2018).

[3] Derya Birant and Alp Kut. 2007. ST-DBSCAN: An Algorithm for Clustering Spatial-temporal Data. *Data Knowl. Eng.* 60 (01 2007), 208–221.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* (2018). http://arxiv.org/abs/1810.04805

[5] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. 1459–1468.

[6] Antonios Karatzoglou, Adrian Jablonski, and Michael Beigl. 2018. A Seq2Seq Learning Approach for Modeling Semantic Trajectories and Predicting the Next Location. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18)*. 528–531.

[7] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1412–1421.

[8] Augusto Q. Macedo, Leandro B. Marinho, and Rodrygo L.T. Santos. 2015. Context-Aware Event Recommendation in Event-based Social Networks. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. 123–130.

[9] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4 (July 2018), 66:1–66:36.

[10] Faisal Rehman, Osman Khalid, and Sajjad Madani. 2017. A comparative study of location-based recommendation systems. *The Knowledge Engineering Review* 32 (01 2017), 1–30.

[11] Shashi Shekhar and Sanjay Chawla. 2003. *Spatial databases - a tour.* Prentice Hall.

[12] Wilson Taylor. 1953. Cloze Procedure: A New Tool for Measuring Readability. *Journalism Quarterly* 30 (1953), 415–433.

[13] Stewart Whiting, Omar Alonso, Vasileios Kandylas, and Serge-Eric Tremblay. 2018. Urban Maps of Social Activity. In *Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM 2018, Stanford, California, USA, June 25-28, 2018*.

[14] Fei Wu and Zhenhui Li. 2016. Where Did You Go: Personalized Annotation of Mobility Records. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. 589–598.

[15] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. 2011. Exploiting Geographical Influence for Collaborative Point-of-interest Recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*. 325–334.

[16] Fisher Yu and Vladen Koltun. 2016. Multi-Scale Context Aggregation by Dilated Convolutions *(ICLR)*.