Object Counting by Leveraging CNN and LSTM with Multi-Source of Input

Xuan Li
xuanli1@andrew.cmu.edu

Fabricio Flores
wflores@andrew.cmu.edu

1 Introduction

Counting objects instances in images or sequence of image (video) is a challenging problem to solve in computer vision. A great many solutions have been developed to count people, cars and other objects, while approaches using canical vision features (for example, SIFT) are not capable to achieve state-of-art performance. Nowadays, neural network architectures has been heavily exploited instead, and have been showed to outperform the traditional approaches for various objects including traffic, animal, and occupancy [9]. On the other hand, LSTM have been showed to model sequential features well as compared to traditional time series models such autoregressive models [8].

In general, the obstacles encountered in the object counting fields includes: variation of object shapes, overlapping, perspective view, variable scale, and model generalization efficiency. For instance, the approach taken to count vehicle in highways or people in crowded scenarios might be fail to count cats on a photo. Similarly, counting objects on a single photo could be of different mechanism as counting objects in real time video. The problems faced in the previous examples have to deal with object occlusion (multiple objects overlaps) and the perspective, where very small objects are present in the distance.

In this work we are interested in the problem of vehicle counting, using images that suffer from low frame rate, low resolution, high occlusion and large perspective [1,2,8,9]. An example is listed in Figure 1. Not only restricted in the number of objects actually present in the image, we are also interested to extract density of objects which contains rich spatial features. In other words, for this project we would build a model that could output the density as well as volume of vehicle given a sequence of frames, which leverages the task of objection detection and counting.

Inspired by work[1,2,8,9], in this paper we implemented our model by incorporating Convolutional Neural Networks (CNN) for spatial density extracting, and Long Short Term Memory (LSTM) for temporal count prediction, and calibrate our model on a open source traffic dataset for vehicle counting. Also, to tackle the vanishing gradient issues [11], in the model we adopt identity passing in CNN, and also in LSTM to learn counting in a residual fashion. Thus, during training we require multi-source of input, including raw RGB image and its corresponding density map image with same dimension, while ground truth of counts could be acquired by summing over the pixels of the density map. The detailed architecture and configuration of the model is illustrated in Section 3, while results are compared in Section 5 and discussed in Section 6.

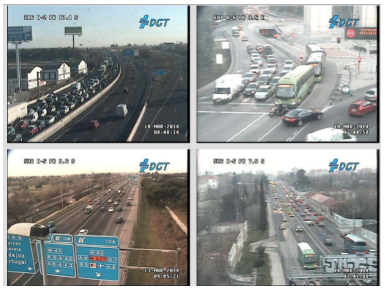

Figure 1: Example of data: low resolution, high occlusion

2 Related Work

2.1 Counting by clustering

This approach relies on the assumption that individual motion field or visual features are relatively uniform, hence coherent feature trajectories can be grouped together to represent independently moving entities[7]. One of the problems of this approach is that relies on the assumption of motion coherency, thus estimation is not adequate when objects remain static throughout frames or counting is not performed in continuous images frames. On the other hand, the algorithm might fail to handle the case that objects suffering from high occlusion [2].

2.2 Counting by detection

In the problem of localization and classification, the task is to detect and classify multiple objects at the same time. Object detection is the problem of finding and classifying a variable number of objects on an image. By a variable number we imply that the output in object detection could be different from image to image.

The current methods used for object detection-counting relies on the approach known as Fast RCNN and Faster CNN [6]. These methods classify objects by regions proposals using convolutional networks. This approach work in a multi-stage sequence, first detecting object boundaries, then performing identification and finally counting if necessary. However, if a high number of objects are present in an image these methods would perform rather slow and thus not suited for real time processing. The drawback of this approach is mainly its scalability and real-time stability, both during the training and during the actual testing while object detection was performed.

2.3 Counting by regression

For images with crowded objects, counting by regression would avoids actual segregation of objects and estimate the crowd density directly based on holistic and collective description of crowd patterns [9, 10]. Counting by regression is a feasible method for crowded environments where detection and tracking are difficult to be implemented in real-time. These methods work by defining a mapping from the input image features to the object counts. In other words, the process accounts for learning a map from local image features to object density maps. Once the algorithm is trained, the object count is obtained by simply integrating over regions in the estimated density map.

2.4 Deep learning and LSTM based counting methods

Attempts have been made to combine CNN with recurrent neural networks (RNN) in order to capture the spatio-temporal information of tasks such as video description and multi label classification[1, 9, 10]. There models integrated the essence of CNN for spatial object detection, and LSTM for temporal regression, which has been regarded as the state-of-the-art architecture for spatial-temporal prediction tasks for various type of objects [8]. For our project we implement our model following the CNN-LSTM outline for traffic counting tasks.

3 Models and Training

3.1 Model Architecture

As it is shown in Figure 2, we refer the implementation in [1] as our basic structure, with CNN $\Rightarrow$ LSTM $\Rightarrow$ residual-learning modules in sequence. This architecture takes raw RGB data image and density map (upper image in the dashed box) as input during training, and only require RGB images when making predictions. Basically, the functions and components in each of the modules are

- CNN module (mainly conv, deconv, maxpooling and atrous layers) with identity passing (vertical black arrow) for spatial feature extraction. Atrous layers are helpful for upsampling by inserting holes between nonzero filters [1], and extract dense feature for deconv layer for density map reconstruction.
- a stacked LSTM module to generate refined temporal features given the extracted spatial features. Given a time step of 5, the module takes the flatten density map predicted from CNN module in a time distributed manner.
- Finally a fully connected (FC) regression layers to predict object counts based on the combined spatial-temporal features (sum over density map, and output of LSTM) in residual fashion.

To see the detailed parameter setting of the network please refer to our repository in github [1]. Basically, we select a similar configuration as VGG16 for the 'conv' and 'atrous' layers, in terms of filter size and numbers. After upsampling and 'deconv', we use a $1 \times 1 \times 1$ 'conv' layers (feature reweighting) to predict density map. We also apply 'L2' regularization on all layers to prevent overfitting.

3.1.1 Baseline Model 1: CNN Only

The configuration of our first baseline model is showed in Figure 8. Ignoring the spatial features and count regression, it only learns the 2D density representation of the objects, and by summing over the predicted density map we retrieve the count statistics. The detail of this model could be found in 'CNN_baseline.py'.

3.1.2 Baseline Model 2: CNN with Residual Regression

The configuration of the second baseline model is listed in Figure 9. In addition to baseline model 1, it has a fully connected regression layer to predict counting based on the residual passing from density map. The total loss would be a weighted sum of density map and counts. The detail of this model could be found in 'CNN_baseline_residual.py'.

3.1.3 Proposed Model: CNN-rLSTM

Our proposed model is illustrated in Figure 10. Given a time step of 5, the LSTM layers take the time distributed output of CNN part, and a FC layer is attached at the end of LSTM to learn the residual of true counts given the sum of density map. Also, the total loss is a weighted combination of density map and counts. The detail of this model could be found in 'CNN_rLSTM.py'.

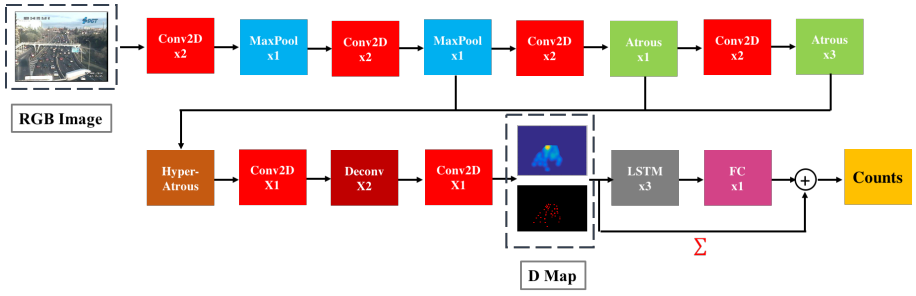

Figure 2: Network Architecture

[1] https://github.com/kkkacfly/10707

3.2 Training

3.2.1 Loss Functions

In order to train and evaluate the performance of the architecture the following loss function is selected

$$L = L_D + \lambda L_C,$$

where L_D accounts for the Euclidean distance that measures the difference between the estimated density and the ground truth. This loss function function L_D for the density map estimation is given by

$$L_D = \frac{1}{2N} \sum_{i=1}^{N} \sum_{p=1}^{P} \|F_i(p, \Theta) - F_i^0(p)\|_2^2$$

where N is the batch size and $F_i(p)$ is the estimated density for pixel p in image i and Θ is the parameter of the CNN module.

The second term of the loss function L_C accounts for the count estimation and is given by

$$L_C = \frac{1}{2N} \sum_{i=1}^{N} (C_i - C_i^0)^2$$

where C_i^0 is the ground truth counting number of frame i and C_i is the estimates count of frame i as well. Finally the value λ is a parameter that account for the importance of the count loss. Thus by simultaneously learning the CNN for density map and the LSTM for counting each task can be trained with fewer parameters.[1] In all, our objective function could be regarded as the summed Mean Square Error (MSE) of density map and count. We will refer the loss as 'MSE' in the following paragraphs.

3.2.2 Evluation Metrics

In order to evaluate the performance of different configurations in the network architecture three evaluation metrics are used: (i) Mean Absolute Error (MAE); (ii) Mean square error (MSE). Particularly, we use MAE to evaluate the loss of counts, and use MSE for the loss of density map.

3.2.3 Training Algorithm

For baseline model 1 and 2 the training could be realized with standard back propagation. For the proposed model, the training procedure is explained in **Algorithm 1**. Basically, the input of the model should be a sequence of image and density map. Traing in batches, the input is sequentially processed, the loss is accumulated at the last time step and then model parameters are updated using ADAM learner. Initially we set the learning rate as 0.001, and λ as 0.001. All our test are conducted on a NVIDIA 1070 GTX GPU, and the average running time per epoch are 120s, 140s, 230s for each of the model respectively.

4 Dataset

4.1 Description

TRANCOS [3]: [2] It has 1244 RGB images of traffic volume in shape 480×640, with ground truth of vehicle counts, density map, ROI map. The average number of vehicle per image is around 30, and following the same setting as in [1,2,3,9], we segmented the first 831 images as training set, and take the latter 421 images as testing set. Samples of the RGB images and density map are showed in Figure 1 and 3. It is observed that there exists a high variation camera angle, vehicle scale, and occlusion.

[2] http://agamenon.tsc.uah.es/Personales/rlopez/data/trancos

Algorithm 1: Batched CNN-rLSTM Training Algorithm

Input : sequential RGB images: $\{X_{11}, ..., X_{nm}\}$, $X_{ij} \in \mathbb{R}^{h \times w \times c}$
Output : sequential density map $\{D_{11}, ..., D_{nm}\}$, $D_{ij} \in \mathbb{R}^{h \times w}$
Parameter : Θ, Γ, Φ

// initialize parameters
for $i = 1 : n$ **do**
  for $j = 1 : m$ **do**
    $\hat{D_{ij}} = \text{FCN}(X_{ij}; \Theta)$
    $L_{Dj} = L_2(D_{ij}, \hat{D_{ij}})$
    $C_r = \text{FC}(\text{LSTM}(\hat{D_{ij}}; \Gamma); \Phi)$
    $C_{ij} = \texttt{TensorSum}(\hat{D_{ij}}) + C_r$
    $L_{Cj} = L_2(\texttt{TensorSum}(D_{ij}, C_{ij}))$
  end
  $L = \sum_j L_{Dj} + \lambda \sum_j L_{Cj}$
  update
  $\Theta, \Gamma, \Phi \leftarrow \texttt{ADAM}(\Theta, \Gamma, \Phi)$
end

4.2 Preprocessing

As it is seen in left column in Figure 3, there exist ambiguous regions in the original raw images, which is hard even for hand-labeling of vehicle. Thus, besides raw images, in the original dataset the researchers also included a mask map for each of the image to indicate the region of interests, and the ground truth density is also corresponding to the masked images. Thus, we generated the input image by pointwise multiplication with the mask, and a sample is showed in the second column in Figure 3. Also, inspired by VGG 16, we subtract the global mean of RGB channel separately to reduce variation.

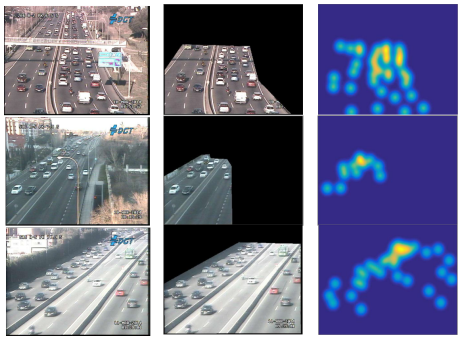

Figure 3: Sample of Processed Image and Density Map: TRANCOS

4.3 Data Augmentation

Since the number of original training set is relatively small compared with the number of parameter of model, we tried several image generation methods using 'cv2' package including:

- flipped images: horizontally flip the processed training set including raw RGB image and density map
- cropped images: randomly crop the training set using a fixed scale, on both of RGB image and density map
- adjust contrast/brightness: randomly adjust the contrast/brightness within scale range $[0.2, 0.8]$ on the training set (raw RGB image only)

5 Result

5.1 Comparison of Different Data Augmentation

First we evaluate the performance of different approaches of data augmentation mentioned in Section **4.3**, using baseline model 1. The convergence property is showed in Figure 4, and the comparison of loss is listed in Table 1. Here Case 1, 2, 3 refers to flipped, cropped and adjusted brightness processing. It is observed that in general the flipped case converges faster, but eventually the other two cases would converge to a local optimal with minimum MAE of count. Due to the limited computation resource we have at hand, for the following experiment we augmented the training set only using random brightness.

	MSE	MAE
Case 1	**0.06**	5.46
Case 2	0.09	4.75
Case 3	0.08	**4.47**

Table 1: Comparison on Different Data Augmentation

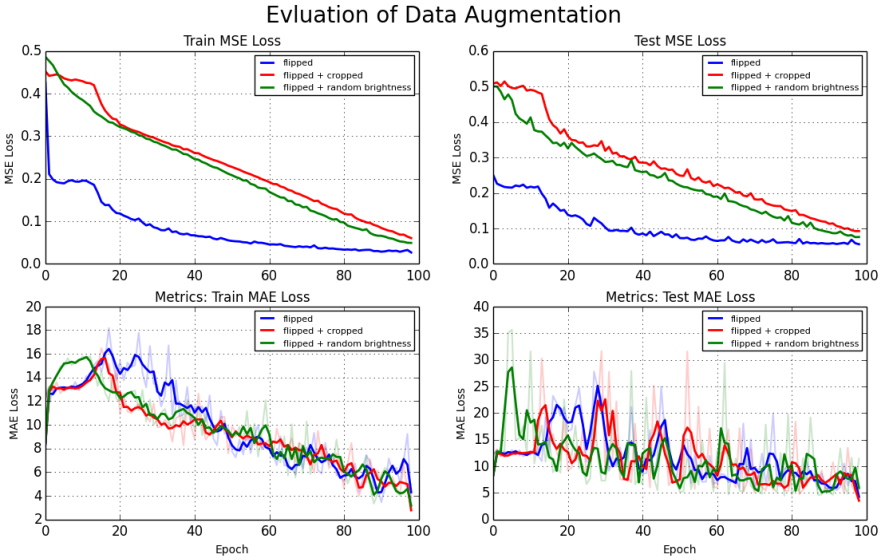

Figure 4: Comparison of Different Data Augmentation

5.2 Comparison of Different Models

We run 100 epochs of each of our model With the augmented training set, The convergence property (training set), prediction of count trend (test set), prediction of density map (test set) are showed in Figure 5, 6 and 7 respectively. It is observed that CNN-rSLTM converges asymptotically faster than the two baseline models, and it attain a lower error rate in terms of both MSE and MAE for training. However, as it is reflected in Table 2, within 100 epochs the MAE on the test set does not converge to a lower range compared with Baseline model 2, which might be due to insufficient training. Thus, for next stage would try to fine-tune the model with another 100 epochs for evaluation.

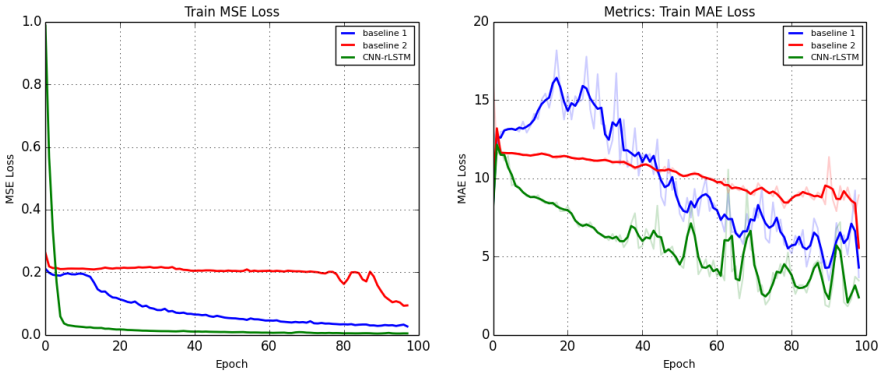

Figure 5: Comparison of Different Models: Loss

Second, in Figure 6 it is observed that in general Baseline 2 and CNN-rLSTM match the count trend visually better than Baseline 1, which might be due to the additional residual regression output layer. In 7, it is seen that all of the models can basically learn the shape of density map, but none of them can assign proper density (weights) to the corresponding pixel. In other words, the occlusion problem has not been fully tackled given the current CNN settings.

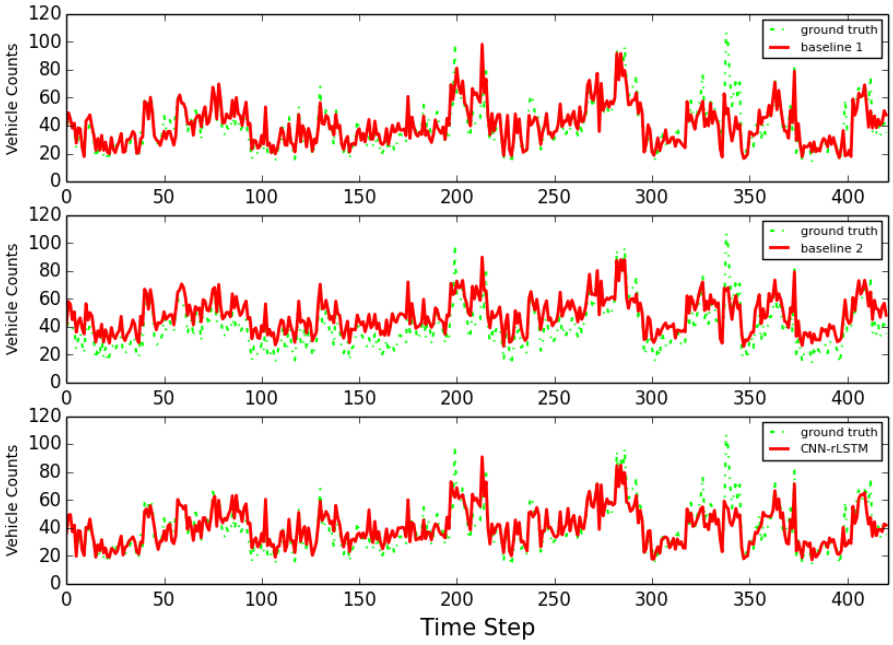

Figure 6: Comparison of Different Models: Count Prediction

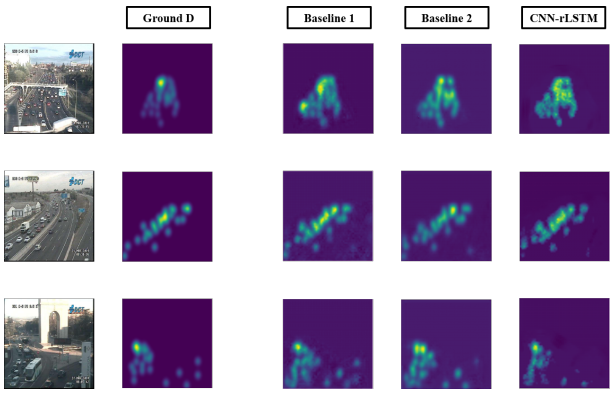

Figure 7: Comparison of Models: Density

Finally, besides the three models proposed in this paper, in Table 2 we also compared with three other similar models implemented in work[1,2]. The 'FCN-MT', 'FCN HA' are similar like Baseline model 1, but are following the architecture of Resnet50 and achieve state-of-the-art performance on this dataset; the 'FCN-rLSTM' has a similar structure as 'CNN-rLSTM', but with different layer settings and identity passing. First it is seen that our models can achieve comparable results as in [1,2]. However, the most complicated 'CNN-rLSTM' did not outperform the baseline ones, but interestingly, this observation is in accordance with the phenomenon in work [1,2], which is also reflected in Table 2 as well.

	MSE	MAE
FCN-MT [2]	-	5.31
FCN-HA [1]	-	**4.21**
FCN-rLSTM [1]	-	4.38
Baseline 1	0.08	**4.47**
Baseline 2	0.04	5.73
CNN-rLSTM	**0.02**	4.63

Table 2: Comparison on Different Models

6 Conclusion

In this paper, we have built, trained and tested three combination of 'CNN', 'LSTM', 'identity pass', 'residual regression' to predict volume counts as well as density map on a public traffic vehicle dataset. We were able to achieve comparable results with state-of-the-art methods, and observe similar behaviors as mentioned in work[1]. In all, we have concluded that

(a) A proper data augmentation would benefit the model in terms of convergence as well as achieving lower error rate.

(b) 'CNN' module is able to extract the general outline of density map (spatial distribution), but can not learn to assign weight properly, which might be due to the frequent up/down sampling.

(c) 'CNN' framework only might be sufficient for count prediction purpose only, and within 100 epochs the more complicated spatial-temporal 'CNN-rLSTM' does not show significant improvement in terms of MAE.

(d) Although the more complicated 'CNN-rLSTM' would converge faster than the baseline models, to achieve a comparable performance it might require more training epochs as well as a learner with variable learning rate.

References

[1] Zhang, Shanghang, et al. (2017) FCN-rLSTM: Deep Spatio-Temporal Neural Networks for Vehicle Counting in City Cameras. *arXiv preprint arXiv:1707.09476*

[2] Zhang, Shanghang, et al. (2017) Understanding Traffic Density from Large-Scale Web Camera Data." *arXiv preprint arXiv:1703.05868*

[3] Onoro-Rubio, Daniel, & Roberto J. López-Sastre. (2016) Towards perspective-free object counting with deep learning. *European Conference on Computer Vision.* Springer International Publishing.

[4] Chan, Antoni B., et al. (2008) Privacy preserving crowd monitoring: Counting people without people models or tracking. *Computer Vision and Pattern Recognition, 2008.* CVPR 2008.

[5] Lempitsky, Victor, & Andrew Zisserman. (2010) Learning to count objects in images. *Advances in Neural Information Processing Systems.*

[6] Yali Amit, Pedro Felzenszwalb. https://cs.brown.edu/p̃ff/papers/detection.pdf

[7] Loy C.C., Chen K., Gong S., Xiang T. (2013) *Crowd Counting and Profiling: Methodology and Evaluation*. In: Ali S., Nishino K., Manocha D., Shah M. (eds) Modeling, Simulation and Visual Analysis of Crowds. The International Series in Video Computing, vol 11. Springer, New York, NY

[8] Zhang, Yingying Zhou, Desen Chen, Siqin Gao, & Shenghua Ma, Yi. (2016). Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. *589-597. 10.1109/CVPR.2016.70.*

[9] Xiong, Feng, Xingjian Shi, & Dit-Yan Yeung. Spatiotemporal modeling for crowd counting in videos. *arXiv preprint arXiv:1707.07890 (2017).*

[10] Sindagi, Vishwanath A., & Vishal M. Patel. Generating high-quality crowd density maps using contextual pyramid cnns. *IEEE International Conference on Computer Vision. 2017.*

[11] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).*

Appendix

Model Architectures

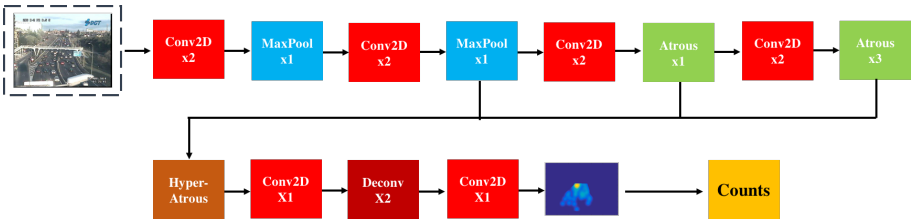

Figure 8: Baseline Model 1: CNN Only

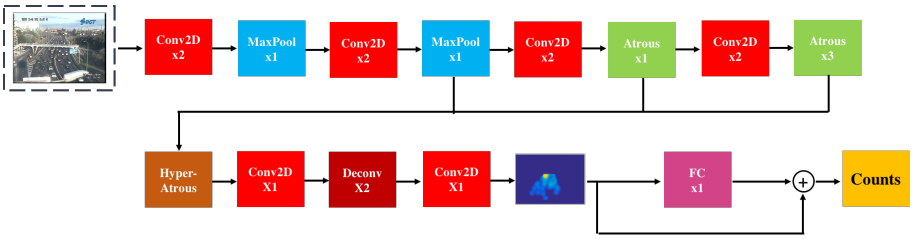

Figure 9: Baseline Model 2: CNN with Residual Regression

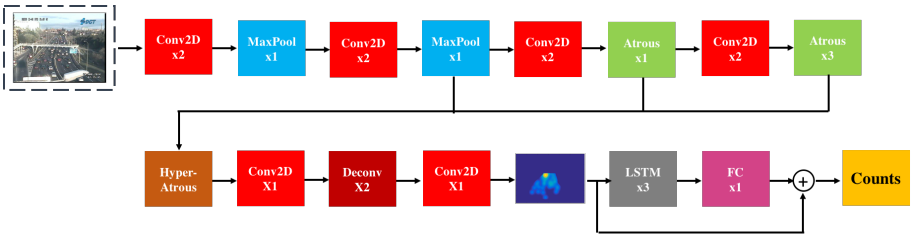

Figure 10: Proposed Model: CNN-rLSTM