FIA 2020/22
XII CONGRESSO/CONGRESO IBEROAMERICANO DE ACÚSTICA
XXIX ENCONTRO DA SOCIEDADE BRASILEIRA DE ACÚSTICA - SOBRAC
Florianópolis, SC, Brasil

Sound Pressure Estimation Method Using Traffic Cameras and Artificial Neural Networks

Lima, M. S.[1]; **Pinto, P. C. C.**[2]; **Gomes, J. G. R. C.**[3]; **Torres, J. C. B.**[4]

[1] Universidade Federal do Rio de Janeiro/PEE/PADS, Brazil, mathlima@poli.ufrj.br
[2] Universidade Federal do Rio de Janeiro/PEE/PADS, Brazil, pedrocayres@poli.ufrj.br
[3] Universidade Federal do Rio de Janeiro/PEE/PADS, Brazil, gabriel@pads.ufrj.br
[4] Universidade Federal do Rio de Janeiro/PEE/PADS, Brazil, julio@poli.ufrj.br

Abstract

Sound pressure in cities is an important information for urban planning. Measuring this information directly usually requires a complex microphone infrastructure, which may in turn require a considerable monetary investment and long term measuring periods. Cities usually have closed-circuit TV infrastructure to monitor traffic on streets, but such cameras do not have microphones. A new solution for sound pressure inference using closed-circuit TV infrastructure, *i.e.* for estimating sound pressure level from traffic images, is considered in this work. Specifically, we make a comparison between previously proposed non-temporal convolutional neural networks and our presently proposed solution, which uses long short-term memories (LSTM) to exploit temporal relationships within image blocks. The dataset is composed of 38 videos representing traffic under different conditions (day time, weather, and so forth), with an overall length of 995 minutes. In order to predict the average sound pressure from unseen image sequences (without audio), 130 model variations are trained using the images and audio signals from training videos within the dataset. Model performance is assessed using mean squared error and Pearson correlation between the predicted and target output signals, across ten different folds. The LSTM-based neural networks yield better results in comparison to the non-temporal architectures. The newly proposed models achieve estimation errors below those of the previously proposed models: 71.3% average correlation and mean-squared error equal to 1.14 (LSTM), in comparison to 60.2% and 1.47 (non-temporal). Regularization methods are essential for training, and the VGG16 convolutional neural networks yield the best results. Object detection architectures such as Faster R-CNN seem to have a potential for improving the prediction results. We conclude that predicting urban sound pressure from image sequences is possible. Future work topics include preparing new datasets, that represent different places and traffic viewpoints, and investigating how object detection may improve the results.

Keywords: Sound pressure, machine learning, convolutional neural networks, audiovisual signal processing, smart cities.

PACS: 43.50.Lj; 43.60.Lq; 43.60.-c.

1. INTRODUCTION

As the urge for the development of sustainable cities and communities arises [1], so does the need to monitor variables related to environmental pollution. Among those variables, noise pollution is of special interest. At first sight, noise pollution may not seen as harmful, but it has been shown to directly impact the physical and mental health of citizens [2–4], children development [5] and surrounding wild life [6].

Noise pollution can be caused by, among other factors, road traffic, railways, air traffic and construction sites. Knowing which city areas have strongest sound pollution, at specific moments of the day, is crucial for an effective city management, as well as other aspects of urban engineering. It is known that the most important source of noise pollution in cities is road traffic [7], and major cities usually have an infrastructure of closed-circuit television cameras in order to monitor traffic. The usage of this infrastructure can be a cost-effective solution for sound pressure estimation on streets. Closed-circuit

television cameras, however, are not allowed to capture audio in many places, and therefore an audio processing solution cannot satisfy all needs. For this reason, a task of estimating sound pressure from traffic images is proposed.

In previous works [8, 9], we had used convolutional neural networks in order to extract features from individual video frames, and we used those feature vectors to estimate sound pressure levels. In the present work[1], video frame sequences are used as inputs for sound pressure prediction, so that the temporal relationship between successive video frames improves the regression error. More specifically, convolutional neural networks extract features from the frames in the input sequence, and the sequence of feature vectors then feeds a long short-term memory, which in turn generates an output vector that is used for predicting a scalar sound pressure level.

The paper is structured as follows: in Section II, we present the basic concepts that are required for a proper definition of convolutional neural networks, long short-term memories and transfer learning. The proposed methodology, including dataset generation, model training and evaluation, and conclusive tests, is described in Section III. The main results are presented in Section IV, and the conclusions are presented in Section V.

2. FUNDAMENTALS

In this section, some basic concepts necessary for the understanding of this work are introduced. We briefly discuss convolutional neural networks, long short-term memory cells and the concept of transfer learning.

2.1 Convolutional Neural Networks

A convolutional neural network (CNN) is a class of artificial neural networks that uses a convolution operation within its layers. Each convolutional layer is responsible for applying such operation between the input data and a matrix kernel. In image processing applications, these layers are often responsible for extracting feature maps from an image, and are therefore commonly known as feature extractors. Convolutional layers are generally used alongside with pooling layers, where a down-sampling operation of such feature maps occur. The global max pooling (GMP) operation consists of calculating the maximum value within each feature map, while the global average pooling (GAP) consists of calculating the average value.

Three different convolutional neural networks topology are considered in this work: the VGG16 [10] is a network composed of stacked convolutional and pooling layers; the ResNet50 [11] is a residual network composed of stacked convolutional/pooling layers and direct information paths between layers; and the InceptionV3 [12] is composed of convolutional and pooling layers, as well as a number of techniques that exploit computationally efficient operations.

2.2 Long Short-term Memory

Classical recurrent neural networks have problems retaining information from long sequences [13]. The long short-term memory (LSTM) is a recurrent neural network layer proposed to address this problem. It uses four weight matrices in its internal structure, in such a way that it optimizes which information should be forgotten, stored, updated or sent to the output. The reader can learn more about the LSTM and its applications on the original paper by Hochreiter et al [14].

2.3 Transfer Learning

Transfer learning is the process of using a model, or part of a model, previously optimized for an original task, in another task. Usually, the original task is optimized over a bigger dataset, so that the model for the new task can benefit from the previous model generalization.

In regard to convolutional neural networks, it is common to load (*i.e.* to transfer) their convolutional layers and pre-trained weights into the new model, using it as an image feature extractor. These convolutional layers can also be frozen, which means their weights are not updated during the training algorithm, only updating the subsequent layers. This allows shorter training and evaluation times, while preserving generalization

[1]Source code of this project can be found at github.com/ma-ath/tcc-matheuslima

of the model.

3. METHODOLOGY

In this work, a dataset composed of audiovisual recordings from a traffic intersection is used. This dataset was generated from a camera and its internal microphone, placed at the window of a nearby building. Recordings were made from about 50 meters away from the target location. The resulting dataset is composed of 38 videos, with an overall length of 995 minutes, recorded in a variety of weather and traffic conditions, as well as different day times. This dataset is an expanded version of the one used in [8] and [9].

Our proposed architecture expects a dataset of images as inputs, and sound pressures as targets. Video frames are therefore extracted from the dataset, and targets are calculated from the extracted audio data, as:

$$S_t[i] = \ln\left(\frac{1}{M}\sum_{k=iM}^{(i+1)M-1} I^2[k]\right), \qquad (1)$$

where S_t is the sound pressure, M is the number of audio samples that are assigned to each frame that is extracted from the video, and $I[k]$ is the k-th audio sample.

From the original video sequence, every 10th frame is sampled as an *extracted* frame, which means that the time interval between successive extracted frames is approximately 330 ms. The summation in Eq. 1 uses audio samples from the 330 ms immediately before, and from the 330 ms immediately after, the i-th extracted frame, in order to compute the i-th sound pressure sample. Image resizing and a z-score normalization is applied to each color channel on all networks input. The process of dataset generation is depicted in Figure 1.

Transfer learning is used on all convolutional feature extractors that are used in the present work: we import weights that were pre-trained for image classification on the ILSVRC (ImageNet) dataset [15]. All pre-trained convolutional layers are frozen during training.

In order to apply a cross-fold validation, the dataset is manually divided into 10 different folds. Each fold contains a selection of the 38 videos from the dataset, with the ratio between the overall evaluation video length and the overall training video length ranging from 216 minutes/712 minutes = 30.3% (fold 5) to 268 minutes/659 minutes = 40.6% (fold 1). Video selection for each fold is based on day time and weather condition, in order to diversify the training and evaluation environments.

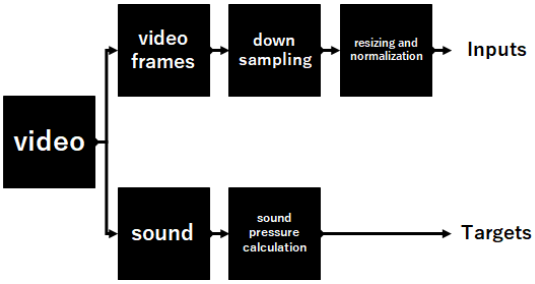
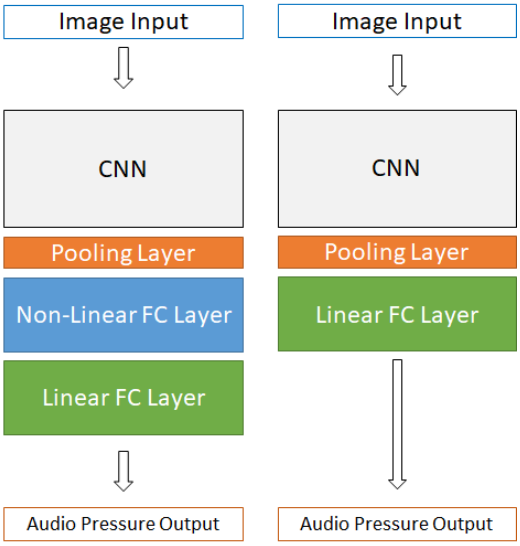


Figure 1: Dataset generation (simplified).

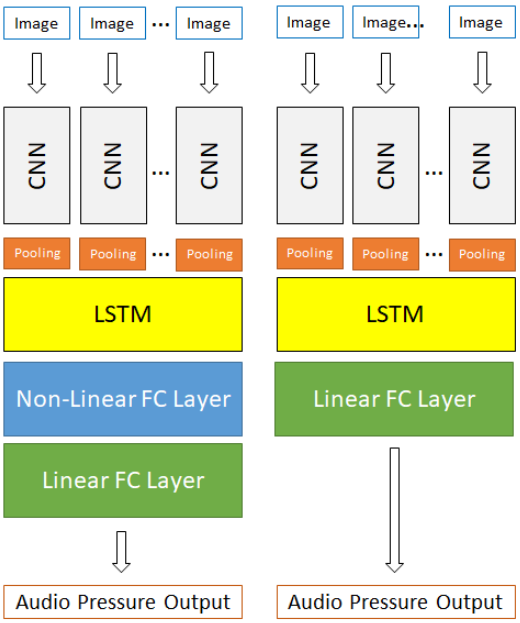
3.1 Network Architectures

In the present work, we compare neural networks that can be grouped into two main categories according to the absence or presence of recurrent layers, as depicted in Figure 2. The neural networks without a recurrent layer, which are shown in Figure 2a, take a single input image at a time, extract its features using convolutional layers (ImageNet pre-trained), and use one or two dense layers (right and left, respectively) to predict the current sound pressure. The neural networks with a recurrent layer, which are shown in Figure 2b, take multiple input images at once, extract their features using the same pre-trained convolutional network (run in parallel for each input image), and then feed the feature vectors through an LSTM layer, in order to exploit the temporal information flow between successive video frames.

Based on previous works [8, 9], we decided to use 128 neurons on both non-linear fully-connect layers that are shown in Figures 2a and 2b. Regarding the LSTM layer, we use an input size of 512×1 when using the VGG16 CNN, and 2048×1 when using both ResNet50 and InceptionV3 CNNs. LSTM output size is 128×1. Regarding the linear fully-connected layer, the input size is 128×1, and the output size is 1×1.



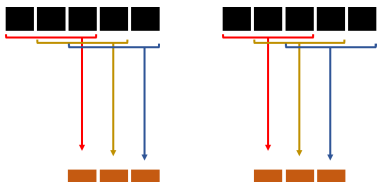
(a) networks using no recurrent layer



(b) networks using an LSTM recurrent layer

Figure 2: 2a networks using a single input image and no recurrent layer; and 2b networks taking multiple successive input images at once and using an LSTM recurrent layer.

It is important to notice that the multiple-input recurrent networks in Figure 2b still maintain a unique output. As such, a choice regarding how to feed data into the network has to be made. One option is to use causal prediction, using past information in order to predict the current target. Another would be to use a non-causal one, using past and future information to predict the current target. Both options are depicted in Figure 3.



(a) Causal feeding

(b) Non causal feeding

Figure 3: Different data feeding approaches: 3a causal, and 3b non-causal.

Either for neural networks with or without the recurrent layer, an experiment using features extracted by an object detection neural network is conducted as well. These features are extracted using a publicly available Faster R-CNN [16] that had been pre-trained over the COCO [17] dataset. Extracted information includes the object class, confidence score and its bounding boxes coordinates.

Processing of these features is conducted by removing objects that are detected with a low confidence score, or objects whose classes are either irrelevant or misidentified for the task, or both. Two tensor representations of these features are compared. The first one is a 6×1 vector whose components count the relevant class instances within the image. The second one is a sparse 20×8 matrix containing, in each row, the detection confidence score, the square root of the bounding box area, and a one-hot representation of the identified class. Lines are ordered according to confidence score. These features are applied to the network as auxiliary inputs, as depicted in Figure 4. The non-linear fully-connected layer that receives these features has 64 neurons, while all others have 128 neurons.

In all tests, both L2 and dropout normalization methods are tried, in order to mitigate over-fitting during network training. We run gradient descent using the Adam optimizer and MSE loss function for all models. The figures of merit used to evaluate the models performance are the mean MSE and the mean correlation between target and predicted outputs.

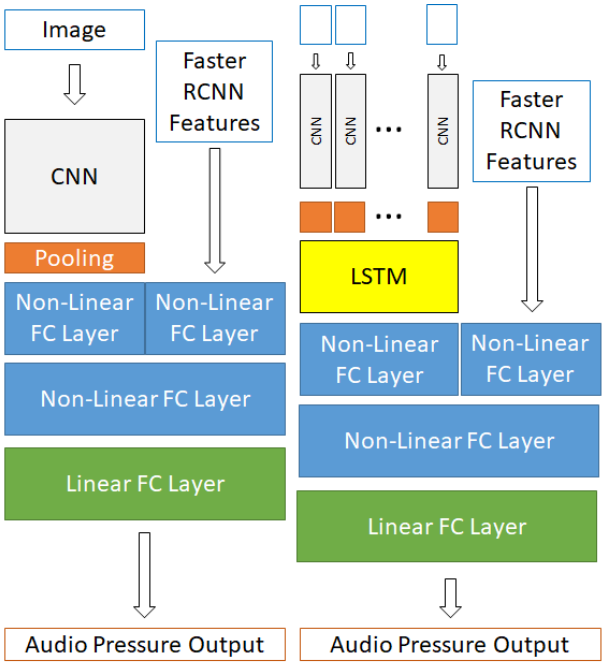


Figure 4: Networks using an auxiliary Faster R-CNN input.

4. RESULTS

As the number of possible models grows exponentially with the number of hyperparameters variations, optimizing all possible models in multiple folds is unfeasible. Therefore, in order to choose which hyperparameters should be considered in a multiple-fold training session, we select the VGG16 as default CNN, and run a selection of models with varying hyperparameters on a single fold. A list of hyperparameters, as well as its possible values, are shown in Table 1.

Table 1: Hyperparameters experimented in one fold.

Hyperparameter	Possible values
Pooling layer	GMP / GAP
Number of images input	9 / 17
LSTM dropout rate	0% / 20% / 50%
FC layer regularization	none / L2
data serving approach	causal / non-causal
FC activation function	tanh / relu

4.1 Results on a single fold

Among the selected hyperparameters in this training session, it is observed that using a global average pooling layer is better than using a global max pooling, as it usually yields a smoother training curve.

In respect to the LSTM layer, using 17 image inputs usually leads to better minimization results when compared to using only 9 images. This led us to expand the number of inputs on the LSTM layer in Section 4.2, from 17 up to 32 images. Non-causal data feeding also yields better minimization results, in comparison to causal feeding. The usage of dropout on this layer proves to be essential, as not using dropout consistently leads to overfitting. There was no considerable difference between using 20% or 50% dropout rates.

In respect to non-linear fully-connected layers, there is no significant difference between using a hyperbolic tangent or a ReLU activation function. Hyperbolic tangent is chosen here based on previous works [9]. We choose not to apply an L2 regularization method in these layers, and instead apply a 20% dropout rate in Section 4.3.

4.2 Results on multiple folds

In this training session, multiple networks variations are optimized over the previously prepared set of folds. Network average behavior is assessed, and hyperparameter variations are tested. In particular, we test the VGG16, ResNet50 and InceptionV3 CNNs as feature extractors. The usage of a fully-connected hidden layer on all networks is also assessed. Finally, models using the extracted Faster R-CNN features are evaluated.

Best models without the usage of a recurrent layer can be seen on Table 2. Best results are achieved by the VGG16 CNN, and a non-linear hidden fully-connected layer is consistently beneficial. This extra layer however is prone to overfitting, likely due to the increased number of parameters. Auxiliary inputs with Faster R-CNN extracted features further improve these results, with a MSE decrease from 1% up to 8%. Using such auxiliary inputs, however, proves to be challenging, as the model tends to easily overfit. The tensor representation that delivers the best results is the 6×1 counting vector representation.

Best results of models using an LSTM recurrent layer can be seen on Table 3. The VGG16 convolutional neural network once again achieves the best minimization results, and a hidden fully-connected layer is beneficial. Using longer input sequences is consistently better when compared

Table 2: Models with no recurrent layer (ordered by best epoch MSE)

Model Number	Mean MSE	Mean Correlation	CNN	Hidden layer
12	1.46	59.73%	VGG16	Yes
15	1.56	57.62%	VGG16	No
13	1.58	57.29%	ResNet50	Yes
16	1.68	55.15%	ResNet50	No
14	2.06	41.83%	InceptionV3	Yes
17	2.46	32.84%	InceptionV3	No

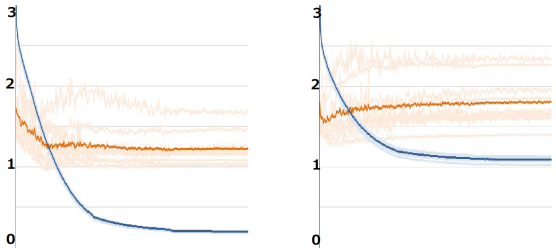
to shorter sequences. Faster R-CNN extracted features, as auxiliary inputs, improve results by around 1%. Once again, hidden fully-connected layers are prone to overfitting. However, a 20% dropout applied on the LSTM layer seems to mitigate this problem, and networks using an LSTM are less prone to overfitting.

Table 3: Models with LSTM recurrent layer (ordered by best epoch MSE)

Model Number	Mean MSE	Mean Correlation	CNN	Hidden layer	Input Size
1	1.15	70.53%	VGG16	Yes	32
7	1.21	69.45%	VGG16	No	32
6	1.40	62.24%	VGG16	No	9
0	1.42	62.21%	VGG16	Yes	9
3	1.51	61.04%	ResNet50	Yes	32

4.3 Direct comparative study

It is evident that models using a recurrent layer surpass models without it. The best model using an LSTM recurrent layer (model 1) has a MSE approximately 21% lower than its counterpart without a recurrent layer (model 12). A direct comparison of the best LSTM and non-LSTM models is conducted, by retraining them for 300 epochs. Different from Sections 4.1 and 4.2, a 20% dropout regularization is applied after the hidden layer, in order to mitigate overfitting issues. Loss minimization curves are shown in Figure 5. MSE and correlation figures for these experiments can be seen in Table 4. Time domain responses for both models are shown in Figure 6.

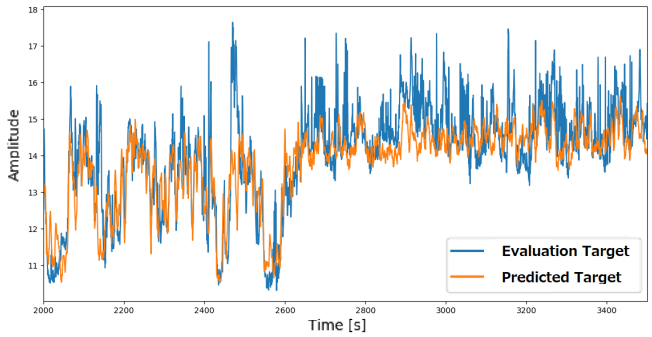


(a) With LSTM **(b)** Without LSTM

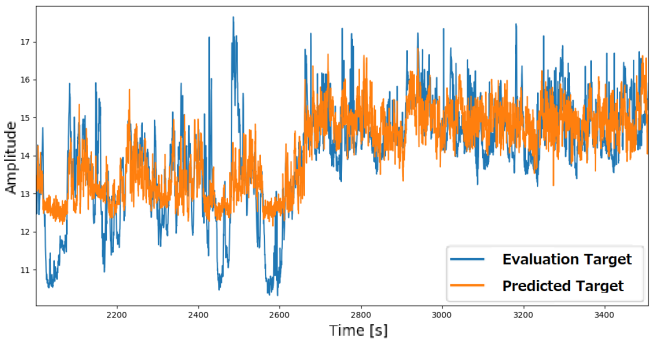
Figure 5: Loss minimization curves for 300 epochs. Figure 5a shows the loss curve for model using an LSTM. Figure 5b shows the loss for model without LSTM. In both figures: blue represents the training loss; orange represents the evaluation loss; thin lines represent the loss for each individual fold; thicker lines indicate average loss curves computed over all folds.

Table 4: Comparison of best models with and without the LSTM layer.

Model Description	Mean MSE (best epoch)	Mean Correlation (best epoch)
Best LSTM	1.14	71.34%
Best non-LSTM	1.47	60.19%



(a) With LSTM



(b) Without LSTM

Figure 6: Temporal sound pressure prediction sequences on the validation datasets: 6a using one LSTM layer, and 6b without any LSTM layer.

5. CONCLUSIONS

In the present work, models using LSTM layers consistently outperform those who do not use them. Using longer input sequences with a non-causal dataset feeding is preferable. The model that achieved the best results is detailed in Figure 7. We explored the use of auxiliary inputs containing Faster R-CNN features, using two different tensor representations. The features improved prediction results marginally, with the disadvantage of increased overfitting challenges. All models tended to overfit during training, and a dropout regularization method was tested.

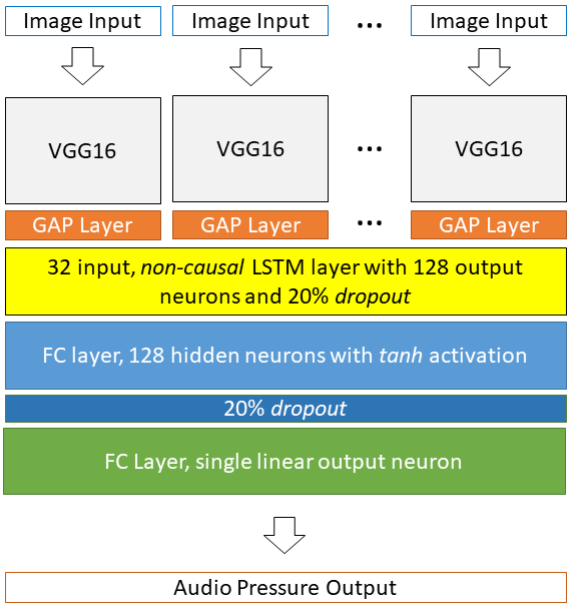


Figure 7: Detailed view of the best model achieved.

5.1 Future Works

During this research, a number of possible future works were identified. A summary of the most promising ideas are presented as follows.

- Generation of a new dataset, containing images from multiple locations. Using images from real CCTV cameras is preferable. Also, recording of audio closer to the framed scene is also preferable.
- Exploration of the convolutional layer. A possibility to further improve results might be experimenting with 3D convolutional blocks [18, 19] or Convolutional LSTMs [20].
- The usage of multiple input models showed promising results. Further exploring how to integrate classification networks such as the Faster R-CNN or the YoloV4 [21] into the architecture might be a way of improving results.
- Development of a visualization tool integrating video cameras, sound pressure data, sound maps and traffic reports.

6. ACKNOWLEDGEMENTS

This work has been supported by *Conselho Nacional de Desenvolvimento Científico e Tecnológico* - CNPq, in Brazil, under research grants 432997/2018-0, 310481/2019-4 and 440074/2020-7. The first author's work was supported by CNPq.

REFERENCES

[1] UN General Assembly. Un general assembly, transforming our world : the 2030 agenda for sustainable development, 2015. URL https://www.refworld.org/docid/57b6e3e44.html.

[2] Lisa Goines and Louis Hagler. Noise pollution: A modern plague, 2007. ISSN 00384348.

[3] Jing Ma, Chunjiang Li, Mei Po Kwan, and Yanwei Chai. A multilevel analysis of perceived noise pollution, geographic contexts and mental health in beijing. *International Journal of Environmental Research and Public Health*, 15, 2018. ISSN 16604601. doi:10.3390/ijerph15071479.

[4] Ravindra Khaiwal, Tanbir Singh, Jaya Prasad Tripathy, Suman Mor, Sanjay Munjal, Binod Patro, and Naresh Panda. Assessment of noise pollution in and around a sensitive zone in north india and its non-auditory impacts. *Science of the Total Environment*, 566-567, 2016. ISSN 18791026. doi:10.1016/j.scitotenv.2016.05.070.

[5] Alok Gupta, Anant Gupta, Khushbu Jain, and Sweta Gupta. Noise pollution and impact on children health, 2018. ISSN 09737693.

[6] Masayuki Senzaki, Taku Kadoya, and Clinton D. Francis. Direct and indirect effects of noise pollution alter biological communities in and near noise-exposed environments. *Proceedings of the Royal Society B: Biological Sciences*, 287, 2020. ISSN 14712954. doi:10.1098/rspb.2020.0176.

[7] Juan Miguel Barrigón Morillas, Guillermo Rey Gozalo, David Montes González, Pedro Atanasio Moraga, and Rosendo Vílchez-Gómez. Noise pollution and urban planning, 2018. ISSN 21986592.

[8] L. O. Mazza, J. G. R. C. Gomes, and J. C. B. Torres. Noise intensity prediction from video frames using deep convolutional neural networks. *Anais 23rd International Congress on Acoustics, Aachen, Alemanha*, pages 4999–5006, 9 2019.

[9] L. O. Mazza. Sound pressure level prediction from video frames using deep convolutional neural networks. Master's thesis, Federal University of Rio de Janeiro, Rio de Janeiro, 2019.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 2016. ISSN 10636919. doi:10.1109/CVPR.2016.90.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December, pages 2818–2826, 2016. ISBN 9781467388504. doi:10.1109/CVPR.2016.308.

[13] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. ISSN 19410093. doi:10.1109/72.279181.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. ISSN 08997667. doi:10.1162/neco.1997.9.8.1735.

[15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. ISSN 15731405. doi:10.1007/s11263-015-0816-y.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi:10.1109/TPAMI.2016.2577031.

[17] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. volume 8693 LNCS, pages 740–755. Springer Verlag, 2014. doi:10.1007/978-3-319-10602-1$_4$8.

[18] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:4724–4733, 2017. doi:10.1109/CVPR.2017.502.

[19] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann Lecun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. ISSN 10636919. doi:10.1109/CVPR.2018.00675.

[20] Xingjian Shi, Zhourong Chen, Hao Wang, Dit Yan Yeung, Wai Kin Wong, and Wang Chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, 2015-Janua:802–810, 2015. ISSN 10495258.

[21] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. URL https://arxiv.org/abs/2004.10934.