MEASURING RESILIENT COMMUNITIES

An analytical and predictive tool

SILVIO CARTA[1], TOMMASO TURCHI[2] and LUIGI PINTACUDA[3]
[1,2,3] *University of Hertfordshire, UK.*
[1]*s.carta@herts.ac.uk, https://orcid.org/0000-0002-7586-3121*
[2]*t.turchi@herts.ac.uk, https://orcid.org/0000-0001-6826-9688*
[3]*l.pintacuda@herts.ac.uk, https://orcid.org/0000-0002-9422-1810*

Abstract. This work presents the initial results of an analytical tool designed to quantitatively assess the level of resilience of urban areas. We use Deep Neural Networks to extract features of resilience from a trained model that classifies urban areas using a pre-assigned value range of resilience. The model returns the resilience value for any urban area, indicating the distance between the centre of the selected area and relevant typologies, including green areas, buildings, natural elements and infrastructures. Our tool also indicates the urban morphological characteristics that have a larger impact on the resilience score. In this way we can learn why a neighbourhood is successful (or not) and how to improve its level of resilience. The model employs Convolutional Neural Networks (CNNs) with Keras on Tensorflow for the computation. The outputs are loaded onto a Node.JS environment and bootstrapped with React.js to generate the online demo.

Keywords. Sustainable Cities and Communities, Resilient Communities, CNN, urban morphology, SDG 11, SDG 13.

1. Introduction

The configuration of the built environment has a significant impact on the ways in which people inhabit the urban space. Within the scope of this study, we consider resilience in terms of the ways in which urban communities respond to any event on the basis of the presence, location and configuration of the resources in their neighbourhoods and cities. In case of adverse circumstances (that can happen at a sudden as an earthquake or flooding, or more slowly like climate change), communities adapt, withstand and thrive in and around the physical elements of their urban space. This work is underpinned by the assumption that there is a significant correlation between urban morphology and the resilience of communities.

In previous work (Carta et al., 2021), we developed a quantitative method to evaluate the resilience of net-zero communities, based on position, density and proximity of physical resources. In this paper we present further findings where we apply our method to generate an online tool to automatically evaluate the level of

resilience of any neighbour and urban area based on their urban configuration.

The end-goal of our research is to develop a reliable method to calculate resilience values of urban communities based on urban morphology. In particular, we aim to use satellite images using models for object detection, so that our model can automatically assess and consistently provide a resilience value for any urban area in the world. The long-term plan is to train our model to classify resilience values directly from the images, by training it with a robust list of values of resilience for each urban area. In this study, we present the first step towards this plan where we use object-detection to identify relevant urban typologies in satellite imagery and evaluate resilience values directly on the web app.

2. Methods

We generated two datasets of satellite images. The first one is based on the DeepGlobe Land Cover Classification Dataset (Demir et al., 2018) and has been used for the training of the model on Tensorflow. We used a sample of 100 images (out of the 803 images in the original set) selecting the most representative with regards to the typologies observed.

The second dataset has been created for this project using QGIS to extract the OpenStreetMap features related to the physical environment (detectable in aerial images) as for our previous work (Carta et al., 2021). This second set has been used for the validation of the model.

2.1. WORKFLOW

For the resilience predictive model, we followed the workflow below:

Dataset: 1) collect satellite images from the DeepGlobe Land Cover Classification Dataset; 2) object labelling with VoTT (Visual Object Tagging Tool); 3) export the labelled images (in JSON format) (VoTT-JSON); 4) Import the labelled data (JSON) into roboflow; 5) Pre-process the images in roboflow (including resizing and augmentation); 6) in roboflow generate a new dataset (with pre-processed images) and create train/test split (70/20/10%); 7) export new dataset to YOLOv5/PyTorch format for training.

Training: 1) Import the pre-processed dataset from roboflow to Colab; 2) Clone Yolov5 on Colab; 3) Define model configuration and architecture; 4) Train custom Yolov5 Detector (approx. 4hr – using CUDA tensor types running on GPU); 5) Evaluate Custom YOLOv5 Detector Performance; 6) Run inference with training weights; 7) Export weights (with the best weight model) for tensorflow.js.

Web app visualiser: 1) Import tensorflow.js weights from Colab; 2) Apply the model to the on-screen satellite image selected by the user; 3) Compute the distance between each identified cluster to the GPS location on the centre of the screen; 4) show the final resilience score.

2.2. DATA PREPARATION AND LABELLING

The satellite images from the first dataset (all of which were at the same altitude and resolution) have been labelled one by one using Microsoft VoTT: Visual Object

Tagging Tool (Microsoft, 2019). Our previous work on the city of Copenhagen (Carta et al., 2021) showed that resilience values calculated on proximity and density of physical elements are mostly affected by green areas, natural elements and entertainment venues. Based on these previous findings, we focused on the 4 visually recognisable typologies below to identify relevant classes for our training: 1) Green areas, 2) Buildings (built areas vs unbuilt), 3) Large infrastructures (train stations, stadiums etc.) and 4) Natural elements (lakes, rivers, coasts etc.).

The labelling on VoTT allowed having a clearly tagged dataset where all images are mapped to the 4 classes. All images and labels have been pre-processed, including resizing all set to 416×416 pixels to ensure consistency in the training. We augmented the initial 100 images to 559 images including horizontal flip, rotation (-15° to+ 15°), saturation (-50% to +50%) and exposure (-25% to +25%). The dataset has been split into 70% for the training set (489 images), 20% for the validation set (46 images) and 10% for testing (24). The set ready for training has been exported in Yolov5 format, as the YOLO architecture allows for tensors and the use of GPU perfect for Tensorflow.

2.3. TRAINING DATA ON TENSORFLOW

The main idea underpinning this experiment is to use the YOLOv5 Detector (Redmon et al. 2016), which is an object detection model based on DenseNet/CNN backbone and train the weights for the model using inference (Gavali and Banu, 2019). Our model uses Keras on Tensorflow.

The You Only Look Once (YOLO) model was developed in 2015 and has attracted significant attention in researchers in computer vision since then. YOLO is defined as "an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself" (Ultralytics, 2021). One of the advantages of YOLO is that its system "computes all the features of the image and makes predictions for all objects at the same time. That is the idea of "You Only Look Once" (Thuan, 2021).

Released in 2020 by Glenn Jocher on GitHub (Jocher, 2020), YOLOV5 has been introduced as 'a family of object detection architectures and models […] [that] represents […] research into future vision AI methods' (Jocher, 2020). YOLOV5 object detection model is based on a DenseNet architecture (cf. EfficientDet architecture, which employs 'EfficientNet as the backbone network, BiFPN (bi-directional feature pyramid network), as the feature network, and shared class/box prediction network' (Tan et al., 2020:5).

For the training of our model, the following method has been followed: 1) Install dependencies, including TORCH.CUDA. This package adds support for CUDA tensor types, that implement the same function as CPU tensors, but they utilize GPUs for computation (PyTorch, 2021); 2) Import labelled dataset from roboflow server; 3) Define model configuration and architecture; 4) Train custom Yolov5 Detector. The model has been trained with 50 epochs and the network had 283 layers, 7,263,185 parameters, 7,263,185 gradients, 16.8 GFLOPs (Giga flops, or floating-point operations per second); 5) Evaluate Custom YOLOv5 Detector Performance, as shown in Figure 1.

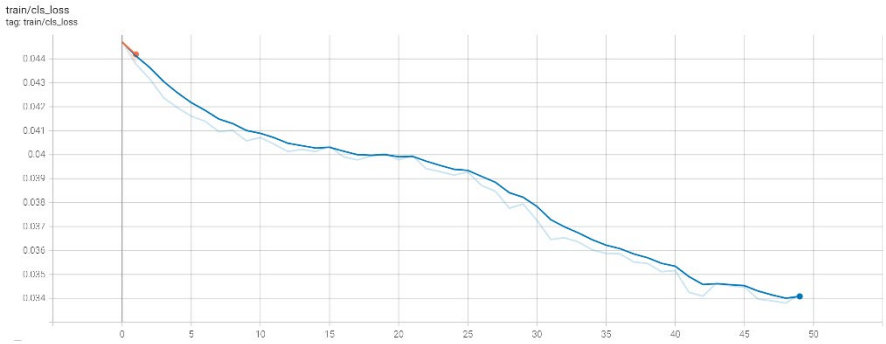

Figure 1. Visualising the performance of the object detector. The class-loss curve goes down to a value of 0.033 after around 40 epochs in this train.

6) Visualise training data with labels for control, as shown in Figure 2.

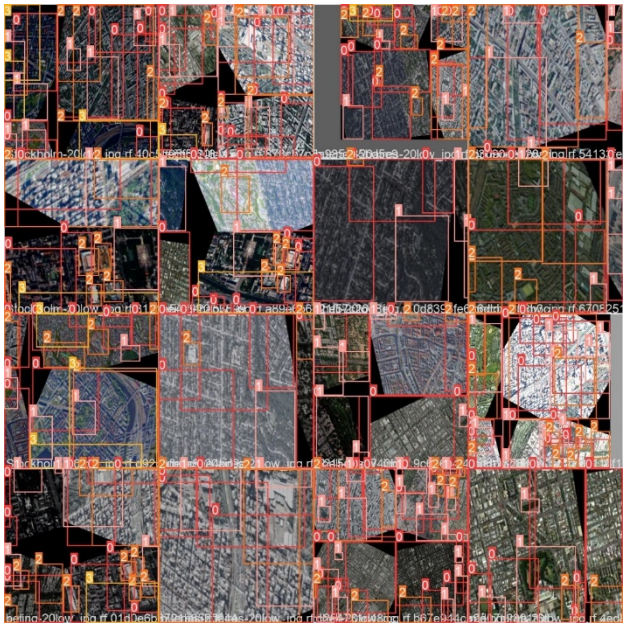

Figure 2. This figure shows the ground truth training data where the classifier has been initially tested with the training set. The classifier correctly assigned various classes (0 for green areas, 1 for buildings, 2 for large infrastructures and 3 for natural elements).

7) Run inference with training weights. The last part of this method is to use the best

weight in the training to run inference on a new set using the method detect.py. In this case, we use the testing set to initially evaluate the model within Colab. The best weight is used to evaluate the resilience value of any new map inputted to the model. Figure 3 illustrates how the model is able to detect the presence of buildings in a test image.

Figure 3. detection of building groups with the testing set.

2.4. VISUALISE RESULTS ON THE WEB APP

For this experiment we developed a web app to calculate the R values directly on browsers, where we evaluate the distance in pixels between the centre of the map (as selected by the user) and the centre of the rectangular selection identified by the Yolov5 model.

The Web app is developed with the front-end Javascript framework React.js and uses Tensorflow.js to load and run the Yolov5 model online. It presents users with a Google Maps-like interface, where one can move around a satellite map of the World, pan and zoom in and out, and look for specific places by label. A button on the lower right corner triggers the execution of the classification over the white-overlayed area on the map, as shown in Figure 4. A static satellite image of the GPS location on the centre of the screen is fetched from Google Maps, encoded, and run through the Yolov5 model.

The results are pixel coordinates of the identified markers for each class (i.e., green areas, buildings, infrastructures, natural elements). We convert these coordinates back to GPS locations in order to draw the right squares onto the onscreen map (each with a different shade depending on the class it belongs to) and calculate the line-of-sight distance between each point and the centre of the screen. Finally, we calculate the *R* value using these distances and the gamma values for each class, displaying the result on screen. On this initial prototype, we are not using the uncertainty given by the model

for each identified marker, but it could play a role in future instances of the model.

Figure 4. Web app interface where user can visualise R values based on area search.

3. Model Validation

In order to test the accuracy of our model, we used a sample of 15 cities, for which we compared the resilience values obtained by using two different methods. Firstly, we used data from existing literate and rankings, using data from the 2013 Grosvenor report (Barkham et al., 2013) and the Quality of Life global ranking from Numbeo (2021). Secondly, we evaluate the resilience values for the same cities using the method we developed in previous work (Carta et al., 2021), as detailed in Section 3.1. We then compared the three sets of values to estimate the precision of our model.

3.1. RESILIENCE BY PHYSICAL ELEMENTS

In order to test our model, we calculated the resilience value of each neighbourhood using the method developed in Carta et al. 2021 which can be summarised in (1):

$$R = \sum_{i=1}^{n} d(min)_n \, \gamma_n \qquad (1)$$

where R is the overall resilience value for eth observed neighbourhood, *d*(min) is the minimum distance between the centre of the neighbourhood and the urban typology considered (e.g. local school, park etc.) and Υ is a coefficient that considers the quality of the distance of the typology based on Kronberg et al. (2019), Government Office for Science (2019) and Knupfer et al. (2018) for the private/public transportation ratio and National Travel Survey (2014) for education settings. The Υ values are calculated following Table 1:

Good	Fair	Bad
< 15 min	15 min / 30 min	> 30 min
< 1,260 Km	1,260 to 2,520 Km	> 2,520 Km

Table 1. We considered walking distance in minutes and Kilometres: average of 1.4 metres per second or 5 km per hour.

As this method requires the georeferenced position of urban typologies (like schools, train stations, parks etc.), we created a specific map for each city using QGIS to import the OpenStreetMap features related to the physical environment at the scale of the city and the pin as from the Google Earth image. To calculate the values for each of the 4 typologies included in this study (green areas, buildings, large infrastructures and natural elements), we run a simple Grasshopper definition to compute the equation (1) using the maps created with OpenStreetMap and QGIS. The Grasshopper definition provided the values shown in the third column of Table 2.

3.2. VALUES COMPARISON

The resilience values obtained with these two methods have been compared with those calculated with the Yolov5 model as shown in Table 2 below.

	R (QoL)	R (GSVN)	R (GH)	R (Yolov5)
Munich	27	24	14.14	9.48
Seattle	26	11	7.56	3.08
Beijing	234	39	7.39	1.43
Tokyo	87	26	6.97	1.47
Melbourne	44	13	6.7	1.24
Rio	232	45	6.7	1.22
Pittsburgh	47	5	6.55	-
Cairo	227	48	5.89	5.48
Moscow	202	37	5.78	-
Delhi	236	42	4.92	0.79
London	149	18	4.82	-
Toronto	101	1	4.69	1.31
Singapore	113	32	4.53	-
Bno Aires	213	36	4.06	1.12
Stockholm	92	6	3.92	1.25

Table 2. Comparison of different values of resilience. QoL (1st column) from Numbeo (2021) indicating Quality of Life values, GSVN (2nd column) from Gosvernors (2014), R values obtained with our Grasshopper script (3rd column) and values with our Yolov5 model (4th column).

3.3. ANALYSIS OF RESULTS

The four sets of R values summarised in Table 2 differ significantly in relative magnitude between sets. However, they appear consistent throughout in relative values per city, as shown in Figure 5. For example, we see that Buenos Aires has lower *R* values than Delhi and higher than Melbourne in all 4 sets. This is consistent in all cities observed.

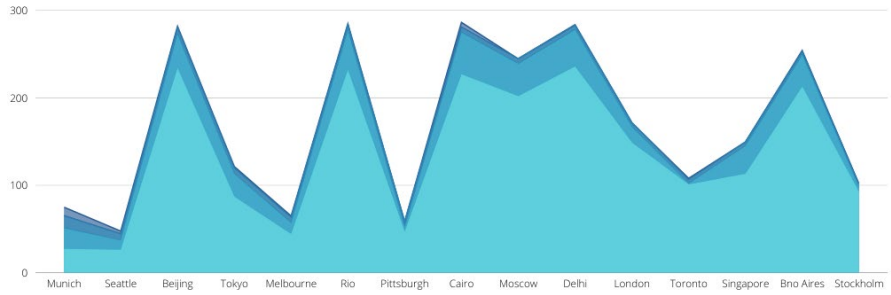

Figure 5. Plot of the R values with the 4 methods from Figure 2.

4. Limitations and Next Steps

In the development of this model, we identified the following limitations. Firstly, we used a limited dataset (around 500+ images) for model training. An architecture like that of Yolov5 would require a larger number of images per class (in the order of thousands). However, Yolov5 is able to provide accurate results even with low numbers of images (Ultralytics, 2021). Moreover, the dataset used for the web app is based on DigitalGlobe Basemap +Vivid (Maxar, 2021) and differs from those of Google Maps and this might lower the accuracy of our results.

Secondly, we employed a limited number of categories to define the urban morphology of the areas observed. We limited the classes to 4 (green areas, buildings etc.). A more comprehensive experiment should include more categories, for example distinction between buildings (schools, residential, museums etc.). This approach will move from the analysis of simple satellite imagery to augmented images through the integration with OpenStreetMap data.

Another point to consider for refinement of this work is the way in which different values in (1) for each typology are summated and averaged. In this experiment, we did not attribute different weights to points. By acknowledging the individual contribution of each element, using for example a coefficient that accounts for distance from the centre of the area, or the size of the typology, we could yield more accurate results. Also, the model returns a value of uncertainty of each identified cluster, which might also be included in the R measure.

We also noticed that the model predicts differently depending on the zoom of the visualised area on screen (larger zoom results in more features being detected). This is partially related to the scale of the images in our training set and to the resolution of

images loaded on the web app when calculating *R*. In future iterations of the app, we will consider constraining the zoom of the area, once the user has made the selection.

Finally, a limitation we observed in using Yolov5 was the difficulty in creating custom datasets without using roboflow to pre-process the dataset and labelling.

The next step for this project is to build a larger dataset with labels in order to refine the model and its accuracy.

5. Conclusions

The model we developed compute the resilience value *R* of any urban area in the world. For each selected area, the app returns an overall *R*, as well as a breakdown of the average of each cluster of urban typologies sorted by category, as shown in Figure 6. With this study, we introduce the early stages of a novel tool to analyse urban areas and their level of community resilience, as well as elements to suggest how to intervene to improve it.

Figure 6. Screen grab of the app showing the overall value R and 4 clusters of buildings detected by the model. The distance from the centre of the area to each cluster is used to calculate the individual contribution of that cluster to R (0.31, 0.28, 0.26).

This tool can be helpful to analyse areas of the world that are not currently included in world's resilience rankings, and to help designers to test design hypothesis for new schemes. For example, planners can analyse georeferenced master plan images to compare current levels of resilience to those resulting from the proposed schemes. We this initial results, we aim at sharing with colleagues our findings and gather feedback on how to improve the tool in the next stages of development.

Acknowledgements

Our YoloV5 model, the Colab notebook, the Grasshopper definition and the dataset used for training can be found at: https://anonymous.4open.science/r/caadria2022-4687. The web app can be found here: https://caadria2022.netlify.app/

References

Barkham, R.J., Brown, K., Parpa, C., Breen, C., Carver, S. and Hooton, C., (2013). *Resilient cities: A Grosvenor research report. Grosvenor Global Outlook*. Available at: https://www.alnap.org/system/files/content/resource/files/main/resilient-cities-a-grosvenor-research-report-2014.pdf

Carta S, Pintacuda L, Owen I. W. and Turchi T (2021) Resilient Communities: A Novel Workflow. *Frontiers of Built Environment*. 7:767779. https://doi.org/10.3389/fbuil.2021.767779

Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D. and Raskar, R., 2018. Deepglobe (2018). A challenge to parse the earth through satellite images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 172-181). From: https://www.kaggle.com/balraj98/deepglobe-land-cover-classification-dataset

Gavali, P. and Banu, J.S. (2019). Deep convolutional neural network for image classification on CUDA platform. In *Deep Learning and Parallel Computing Environment for Bioengineering Systems* (pp. 99-122). Academic Press. https://doi.org/10.1016/B978-0-12-816718-2.00013-0

Government Office for Science (2019). *A time of unprecedented change in the transport system*. GfOS Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/780868/future_of_mobility_final.pdf

Jocher, G. (2020). *YOLOV5*. Ultralytics/yolov5. GitHub. Available at: https://github.com/ultralytics/yolov5

Knupfer, Stefan M. (2018). *Elements of success: Urban transportation systems of 24 global cities*. McKinsey & Co.

Kronberg, Nico et al. (2019) *Transport Statistics Great Britain 2019*. Department for Transport. Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/870647/tsgb-2019.pdf

Maxar (2021). *DigitalGlobe Basemap +Vivid*. Maxar. https://www.maxar.com/products/imagery-basemaps.

Microsoft (2019) VoTT. Commercial Software Engineering (CSE) group. Available at: https://github.com/microsoft/VoTT.

National Travel Survey (2014). *Travel to school*. Department for Transport.

Numbeo (2021). Available at: https://www.numbeo.com/quality-of-life/rankings.jsp

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Roboflow (2020). Available at: https://roboflow.com/

Tan, M., Pang, R. and Le, Q.V., (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781-10790).

Thuan, D., (2021). *Evolution of yolo algorithm and yolov5: the state-of-the-art object detection algorithm*. https://www.theseus.fi/handle/10024/452552

Ultralytics (2021). *YOLOv5 Documentation*. From: https://docs.ultralytics.com/