The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Encoding Neighbor Information into Geographical Embeddings Using Convolutional Neural Networks

Christopher Blier-Wong,[1 2] **Jean-Thomas Baillargeon,**[2]
Hélène Cossette,[1] **Luc Lamontagne,**[2] **Etienne Marceau**[1]
[1] École d'actuariat, Université Laval, Canada (christopher.blier-wong.1@ulaval.ca)
[2] Département d'informatique et de génie logiciel, Université Laval, Canada (jean-thomas.baillargeon@ift.ulaval.ca)

Abstract

Geographic information is crucial for estimating the future costs of an insurance contract. It helps identify regions exposed to weather-related events and regions exhibiting higher concentrations of socio-demographic risks such as flood or theft. In actuarial science, the current approach of estimating future costs in a territory is through one-hot encoding of zip codes, postal codes or company-defined polygon levels in statistical learning models. This method has two main drawbacks: it does not share information from similar risk territories and does not share information from neighboring areas. We propose the Convolutional Regional Autoencoder model, a method for generating geographical risk encodings using convolutional neural networks. We aim to replace the traditional territory variable for estimating future costs of insurance contracts. Experimental results demonstrate that encodings generated by our approach provide more useful features to predict risk-related regression tasks.

1 Introduction

Insurance plays an essential role in society since it enables the transfer of risks from individuals to insurers. Insurers accept this risk transfer in exchange for a fixed premium calculated before knowing the actual cost of the risk. In non-life insurance, actuaries predict the total costs associated with short term risks, such as single-year car insurance and home insurance contracts. Historical data is used to infer relationships between an insurance client's information and his expected future claim costs in a process called ratemaking.

Geographical information is useful in property and casualty insurance ratemaking since it helps contextualize risks. For weather-related perils such as flooding, geographic information is crucial since location represents vital information to predict claim frequency. An insurance company might want to limit the exposure to risks in the same street because if a flood occurs, the insurance company will need to pay a lot of claims at the same time, placing it in a difficult financial situation. For socio-demographic risks such as driving, habits depend on where drivers live: in rural areas, they are less likely to have accidents since they use rarely frequented roads. When they do have accidents, they tend to generate higher claim costs since they are more severe.

For this reason, insurance companies define territory levels, which act as one-hot encoding in their ratemaking models. Due to the nature of insurance data (low frequency, heavy-tailed severity), the variance of claim costs is high. Therefore, a high volume of historical data is needed for models to yield reliable predictions and actuaries have to deal with a variance-bias tradeoff in determining the size of territory levels. Larger territories contain more observations and lower variance but high bias, as opposed to smaller regions with fewer observations and lower bias but higher variance. Some spatial effects occur at a granular scale while some at a grander scale. This challenge implies that a spatial risk effect appearing on a small scale could go undetected to an insurer using territory boundaries of larger size. Therefore, insurance companies tend to choose smaller territories, leading to neighboring territories that look alike, sometimes having a different insurance premium due to high variance.

In this paper, we propose a method to generate low-dimensional encodings (embeddings) of geographic risks across a large territory. These encodings are compact representations of the geographical context of a spatial coordinate. Since the dimensions of the embeddings may be smaller than one-hot encoding of rating territories, we have better control of the variance-bias tradeoff of geographic risk segmentation. Additionally, since geographic embeddings are calculated at coordinate-level and not at territory-level, they may be smoother in space, avoiding harsh spikes when crossing territory limits.

This work aspires to be a reference on geographical risk segmentation with the use of external data sources. We focus on an application to actuarial science, but this research is relevant to other fields such as geomatic and geostatistical sciences. Our contributions include:

- The exploration of using external data at different spatial scales to create an encoding of geographic risks.
- The introduction of the geographic data cube generator, an algorithm for generating matrix grids of spatial data usable by vision techniques.
- The introduction of the Convolutional Regional AutoEn-

coder (CRAE) as a method of generating geographical encodings that exploit information from neighboring points.

- The source code to generate geographical data cubes and to train the neural network [1].

2 Related Work

The work most similar to ours is by (Saeidi, Riedel, and Capra 2015), who examine methods of dimension reduction of UK census data with principal component analysis and autoencoders. The learned representations are used for downstream regression tasks. Although they examine neighbouring effects by adding the longitude and latitude coordinates to the embeddings, results show that adding coordinates does not improve model accuracy in downstream tasks. Our model tackles the spatial continuity directly.

Convolutional neural networks and multi-source geographical information are used by (Yao et al. 2018) for house price mapping. This work was for a task-specific application and trained in a supervised way.

Other geographical embeddings have been studied in specific applications. For example, (Cocos and Callison-Burch 2017) build on the distributional semantics theory that similar words appear in similar contexts. Additionally, people use different vocabularies at different geographical locations. Therefore, tweets sent from a particular location are context-dependent. (Levy and Goldberg 2014) train word embeddings with the skip-gram model using geographic tags based on locations identified from Google Places and OpenStreetMap. Other specific applications of geographical embeddings in the literature include point-of-interests sequencing (Yao et al. 2017), generation of natural environment embeddings (Jeawak, Jones, and Schockaert 2019) and analysis of lexical variation (Eisenstein et al. 2010). Our work stands out by proposing embeddings that are unsupervised and agnostic of downstream tasks.

The approach proposed in this paper designs representations of geographical information such that they may easily be used as features in a machine learning model. These representations are useful since feature engineering of important or significant geographic variables is hard due to their high volume and variety. Our work therefore fits within the field of representation learning (see, e.g. (Bengio, Courville, and Vincent 2013)). Our model is based on convolutional autoencoders, where the encoder is composed of convolutions and max-pooling, while the decoder is composed of deconvolutions and max unpooling. Such ideas are inspired by (Zeiler, Taylor, and Fergus 2011) and (Noh, Hong, and Han 2015), who apply these techniques in vision tasks.

3 Proposed Method

In this section, we present our approach and evaluation strategy. First, we present the models that serve as baselines for our experiment. We proceed with our approach, the Convolutional Regional Autoencoder (CRAE). Vector data cubes are presented, followed by the evaluation scheme.

[1] https://github.com/jtbai/census-compression

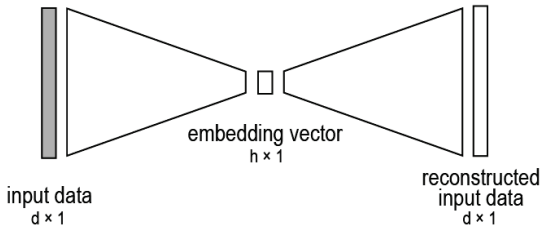

Figure 1: Fully Connected Autoencoder

3.1 Baseline models

To compare the effectiveness of our approach, we establish two baseline algorithms: principal component analysis (PCA) and fully-connected autoencoder (FCAE).

Principle component analysis is a popular linear dimension reduction technique in statistics. An important advantage of PCA is that the percentage of variance conserved from the input features is easily controlled. Additionally, latent variables are linearly uncorrelated, making the problem unique and easily interpretable. Since linear features are highly restricting, the latent variables might not be abstract enough to contain enough information in a low-dimensional feature space.

An autoencoder is a neural network designed to learn a function such that the output values are equal to the input values. In this paper, the FCAE is an undercomplete (bottleneck) autoencoder composed of 4 main components: input data x of size d, an encoder f, a bottleneck layer $f(x)$ (containing the latent variables of size h that will be used as embedding) and a decoder g. Encoders are often composed of one linear layer, followed by an activation function. More layers may generate better representations but are harder to train (see (Bengio, Courville, and Vincent 2013)). The embeddings from this architecture are extracted by applying the encoder function to the initial data, thus projecting the input space to a latent space of a reduced dimension. We present a FCAE architecture in Figure 1.

3.2 Convolutional Regional Autoencoder

The proposed approach in this paper builds on the undercomplete autoencoder by expanding the spatial context. The limitation of the FCAE and PCA methods is that embeddings for a location are created using only features of that same location, i.e. the encoder output is $f(x)$. Since weather or social effects vary smoothly through space, we want to alleviate this constraint and consider supplementing information from nearby locations. This idea is based on the first law of geography: everything is related to everything else, but near things are more related than distant things (see (Tobler 1970)). Using neighbor information will create embeddings using a function of the type $f(x \in \delta_x)$, where δ_x corresponds to the neighbors of location x. In our CRAE model, the neigborhood of x is defined by generating a square grid centered at coordinate x, sampling points at a uniform distance in longitude and latitude.

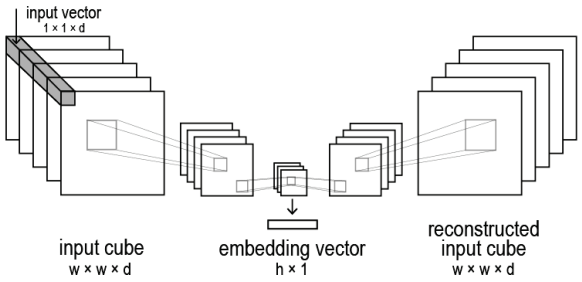

Figure 2: Convolutional Regional Autoencoder

Convolutional neural networks (CNNs), a deep neural network algorithm, are particularly efficient at solving problems of spatial dependence. Their success initially came from vision tasks (see, e.g. (LeCun and Bengio 1995), (Krizhevsky, Sutskever, and Hinton 2012)). We postulate that CNNs can be used for other tasks where data can be formed into a grid. We redefine the data used in our algorithm to convert vectors of data into vector data cubes. This cube can, in turn, be handled in the same way as an image in vision tasks. Although an image is a data cube of depth 3 (where slices consist of R, G and B channels), CNNs may handle data cubes of arbitrary depth. In our CRAE model, data cubes represent geographical information around a center coordinate. Each slice of the data cube is composed of a raster for a specific input variable. The cube is a combination of raster slices, and the number of slices is the number of features to train embeddings. Hence, the depth d of the cube is the number of features available from input vectors used in baseline models. The algorithm to generate data cubes and an example data cube is presented in the next subsection.

In CRAE, the bottleneck comes from the reduction of the depth of convolutions (number of feature maps) and the application of max pooling. Once the CRAE is trained, regional geographical embeddings are generated by applying the encoder to the input data, creating a cube of smaller dimensions. As illustrated in Figure 2, embeddings are created from the values in the middle layer. The cube is flattened into a vector of dimension $h \times 1$.

To illustrate this idea, suppose we have a CRAE model with one encoder layer and an input cube of size $8\times8\times100$. The convolution has zero-padding such that the raster size does not change and has depth 5. After the convolution, the data cube size is $8\times8\times5$. Then, we apply 2×2 max-pooling with stride 2, so the data cube size becomes $4\times4\times5$ and contains 80 values. The embeddings for this example are obtained by flattening the $4\times4\times5$ cube into a 80×1 vector.

3.3 Generating Vector Data Cubes

Since CRAE handles vector data cubes, these cubes first need to be generated. This subsection presents the method for creating vector data cubes from vector data.

Two parameters regulate the generation of data cubes: the width in pixels of the data cube p and the resolution w, the distance in meters between two vertical or horizontal points.

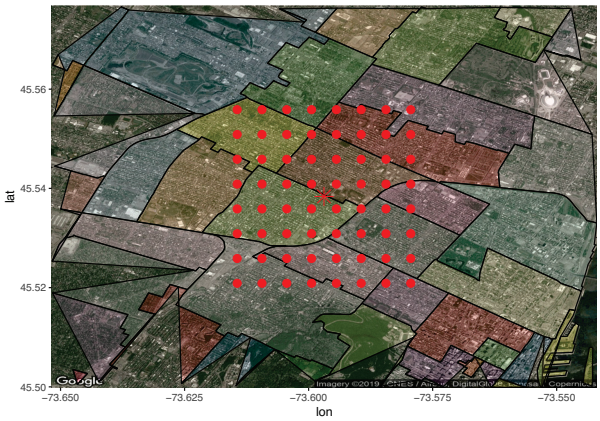

Figure 3: Example Data Grid

The grid is given by the union of coordinates in the formula

$$\delta_{\mathrm{x}} = \bigcup_{\mathrm{i}=0}^{\mathrm{p}-1} \bigcup_{\mathrm{j}=0}^{\mathrm{p}-1} \left[\mathrm{lon(x)} + \frac{\mathrm{w}}{2} \left(2\mathrm{i} - \mathrm{p} + 1\right), \mathrm{lat(x)} + \frac{\mathrm{w}}{2} \left(2\mathrm{j} - \mathrm{p} + 1\right) \right],$$

where $lon(x)$ and $lat(x)$ are respectively the longitude and latitude coordinates of the center point x, where x belongs to the set of locations in the territory where geographical embeddings are calculated. If the grid size p is even, the center point x is not included in δ_x. The general algorithm to create a geographic data cube is presented in Algorithm 1. The random rotation at step 4 is to make the model rotation invariant.

Algorithm 1: Generating a geographic data cube

Input: Center coordinates, cube width w, pixel size p
Output: Geographic data cube

1 project coordinates to Lambert coordinate system;
2 generate grid of size $p \times p$ with pixel width w meters;
3 center grid at input coordinates;
4 apply random rotation to grid;
5 **foreach** *pixel in grid* **do**
6 allocate pixel to corresponding index;
7 assign vector data to pixel

An example of resulting grid (steps 1-4 in Algorithm 1) is presented in Figure 3. The parameters used for this grid are $w = 100$ meters and $p = 8$. The red star corresponds to the center location. In this example, we used a central location close to the border between two polygons. With baseline models, only the data associated to the polygon in which the point is located would be considered. With our model, 12 points would contain data from the original polygon, 8 points would use data from the neighbor polygon, and the remaining 44 are sampled from other, farther, neighboring polygons. As distance increases, more neighbors contribute to the data but with lower importance to each new neighbor.

The corresponding vector data cube from the previous example is presented in Figure 4. Each slice of the cube represents an input variable.

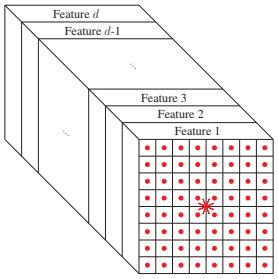

Figure 4: Example Vector Data Cube

3.4 Model Evaluation

We evaluate the baseline models and the proposed approach with two metrics. The first is the reconstruction error of unsupervised models, and the second is the prediction error on downstream regression tasks.

Reconstruction Error The reconstruction error is the objective function to minimize the learning algorithm. This metric represents the inability of the algorithm to reconstruct its input from the latent space. This metric is pertinent since a model with a low reconstruction error of input in the output layer implies that the bottleneck layer contains salient information.

Prediction Error on Downstream Tasks The prediction error on downstream tasks is the metric used to evaluate the usefulness of the embeddings generated by our approach. These tasks use generalized linear models (GLMs) with a Poisson distribution. We use GLMs due to their prevalence in non-life actuarial practice.

Downstream were selected according to two criteria. First, they should be related to insurable risks since embeddings are to be used as geographic risk segmentation variables in actuarial (insurance) ratemaking. Second, the data needs to span large regions: since open datasets are often published by cities or municipalities, comparing tasks across vast territories is complex since data collection methods may differ. Downstream tasks are presented in the results section.

4 Experiment

Two sources of data are needed for our experiment. The first is data which describes the territories, which are used to generate geographic embeddings. The second will be presented in the Results section. The data used to generate geographic embeddings is the 2016 Canadian Census [2]. This dataset contains rich, heterogeneous summary statistics from inhabitants, at the FSA level (an aggregation of neighboring postal codes). The census dataset contains 2,247 variables for each of the 1,620 FSAs. Types of variables include amounts, counts and proportions. Information such as salary, age and field of study is available aggregated by the individual or by household. Since FSAs are aggregation of postal codes, they may be too large to draw useful geographic insights. Our model indirectly performs spatial disagregation by providing features at a higher resolution that the available data. Each polygon in Figure 3 represents a different FSA.

4.1 Generating Census Value Vectors

We first preprocess the Canadian census data. Preprocessing was done such that reconstruction errors from autoencoders did not significantly attempt to reconstruct values on larger scales and did not use discriminatory attributes.

First, we pruned the variables to remove the ones explicitly referring to ethnic attributes. Out of 2,247 variables, 512 were kept as they were judged not related to ethnicity. Examples of removed attributes include language spoken at home and country of birth. Second, the variables were normalized on a scale of $[0, 1]$. This prevented the model from focusing on large reconstruction errors such as salaries and neglecting small reconstruction errors such as inhabitants per household. The code used to normalize census data is provided in a public repository [3].

4.2 Selecting the FCAE architecture

We selected the FCAE architecture on its ability to generate embeddings that yield a low prediction error when used as features in downstream regression tasks. For comparison purposes, we selected an embedding (bottleneck) size of 32. The hyperparameters of this model were the number of layers and feature size decay speed from the input to the bottleneck. We experimented with the following methods for selecting these hyperparameters:

- exponential interpolation where the size of layer l is given by $\lfloor 32 \times 16^{1-\frac{l}{n}} \rfloor, l = 0, \ldots, n$ and n is the number of layers in the encoder. Values of n = 1, 2, 3, 4 and 5 were tested;
- quadratic interpolation where the size of layer l is given by $\lfloor [\sqrt{512}(1 - \frac{l}{n}) + \sqrt{32}(\frac{l}{n})]^2 \rfloor$, $l = 0, \ldots, n$ and n is the number of layers in the encoder. Values of n = 1, 2, 3 and 4 were tested.

We observed that deeper networks do not necessarily lead to a more useful embedding, and we selected an encoder with a fully connected layer going directly from 512 dimensions (the number attributes from the pruned census) to 32 dimensions. We did not use any intermediate representation between the input layer and the bottleneck layer, but used a ReLU activation function to add non-linearity to the network. Figure 1 shows the architecture of the autoencoder.

To train the autoencoder, we used Adam optimizer, with an initial learning rate of 0.0001 and reducing the learning rate by a factor of 4 when validation loss plateaus for 100 epochs. The batch size is 32, and the network is trained for 800 epochs.

4.3 Generating Census Data Cubes

To use the census data in a convolutional neural network, we create a data cube containing information around a coordinate. We use a data cube with census value vectors to generate data to train the CRAE.

[2] https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/index-eng.cfm

[3] https://github.com/jtbai/census-data-transformation

In our experiment, we selected $p = 16$ and $w = 50$ meters for generating data cubes. These parameters were empirically estimated, extrapolating conclusions from the city of Montreal to train the model for Canada.

4.4 Selecting the CRAE architecture

Working with geographic data, we must manipulate polygons and point patterns. Attributing a point to a polygon (postal code to FSA) is a task that requires a long time if data cubes are generated on each epoch. Since there are close to 1,000,000 postal codes in Canada, and each data cube has a depth of 512 features, data size becomes an issue. For a dataset of data cubes of dimension $16 \times 16 \times 512$, generating data cubes in advance takes 36 hours. The alternative, generating data cubes during training, results in 25 hours per epoch. We elected the former approach.

This approach was not without drawbacks: the dataset for 1,000,000 postal codes is composed of 1,000,000 data points, each 16×16 pixels $\times$ 512 features $\times$ 16 bits. This resulted in a 2 Terabyte dataset for a single data cube generation hyperparameter combination. Each epoch represented a challenge and took between 1.5 and 3 hours, depending on the computer hardware. For this reason, we tested embeddings from only one architecture using the entire Canadian dataset. To select the best architecture hyperparameters, we tested various combinations and retained the one that yields a better reconstruction error after 20 epochs. However, since embeddings are trained offline, this training only has to be done once and are within reach for insurance companies.

The two hyperparameters to test were the size of the convolution filter in the first layer and the decay rate for feature maps. We tested convolution filter kernel sizes of 3×3 and 5×5 and intermediate representation depths of 52 and 256.

The selected CRAE architecture trained with the Canadian dataset has two layers of convolution, each followed by a max-pooling of kernel size 2×2 and stride 2. The first layer of convolutions includes 256 channels and a filter of kernel size 5×5 and the second consists of 8 channels, with a filter of kernel size 3×3. The latent space is $2\times2\times8$, generating a bottleneck of 32 values. These 32 values are then flattened into a vector of 32×1 and used in the GLMs for downstream regression tasks. To train the CRAE, we use the stochastic gradient descent over 50 epochs. The learning rate used is 0.001, and we reduce the learning rate on a plateau.

5 Results and Analysis

5.1 Reconstruction Errors

Reconstruction errors determine if the decoder may reconstruct the input vectors from the latent space. Although a good reconstruction error does not directly translate to better representation for embeddings, this is the most convenient optimizer loss function to use to keep the task unsupervised. Generally, embeddings of size 32 have the best performance on downstream tasks. For consistency, we select an embeddings size of 32 for all vectors. Reconstruction errors were 4.6%, 4.7% and 6.9% for PCA, FCAE and CRAE respectively. We note that the PCA and FCAE reconstruct 512 values as CRAE reconstructs $16\times16\times512$ values, hence baseline reconstruction errors can't be compared directly with CRAE. We conclude important information is adequately encoded since all reconstruction errors are negligible.

Table 1: Observed vs Expected on Frequency Predictions

Sev. Level	Observed	PCA	FCAE	CRAE
1	19,010	18,939	*18,934*	**18,949**
2	16,437	16,308	*16,301*	**16,329**
3	9,118	*8,728*	8,728	**8,740**
4	372	*370*	370	**372**
5	45	*47.7*	47.6	**47.4**

Table 2: Exceeded losses (in thousands CAD)

Sev. Level	PCA	FCAE	CRAE
2	1,414	1,495	1,189
3	4,286	4,286	4,157
4	14	13	-4
Total	5,719	5,795	5,342

5.2 Downstream Regression Tasks

The second dataset needed for this experiment are response variables associated with territories to be used to calculate prediction errors on downstream tasks. In Québec, a public institution (SAAQ) handles bodily injury payments resulting from automobile accidents. The SAAQ publishes locations of accidents, which are then geocoded by Montreal Open Data [4], providing a dataset of coordinate-level observations for the location of car accidents. The dataset used to fit our model used in downstream regression tasks are the coordinates of 171,271 car accidents in Montreal between 2014 and the 2018. For each accident, we have a severity level from low damage (level 1) to deadly accident (level 5). Each coordinate is associated to the closest postal code centroid and the downstream task is to predict the number of accidents within postal codes. The metrics we consider are predictions on downstream regression tasks. Embeddings are used as covariates in the regression task. These metrics allow us to determine if the information carried by the embeddings are useful to accomplish generic tasks and establishes the relevance of our approach in other domains. GLMs are trained using 70% of postal codes. Table 1 presents the results for the three sets of embeddings for each downstream task using the remaining 30% validation set. Bold values show the smallest difference with the observed count (best performance), and italic values show the largest difference with the observed count (worse performance). Accident levels 2, 3 and 4 come with indemnity payments. The average payment from the SAAQ was 11,009.70 CAD in 2017[5]. We present the expected exceeded losses (amount exceeding the expected value, assuming each loss is 11,009.70 CAD) for each model in Table 2.

We observe that CRAE outperforms the baseline on all

[4] http://donnees.ville.montreal.qc.ca/

[5] https://saaq.gouv.qc.ca/fileadmin/documents/publications/rapport-annuel-gestion-2017.pdf

tasks. This confirms our intuition that smoothing regional demographic profiles using a CNN approach yields better embeddings. When accident severity is considered, the impact of this improvement is accentuated. Using CRAE would have improved the loss prediction by 400,000 CAD for the 30% test set. The SAAQ could have priced insurance contracts more accurately and lowered it's loss. PCA and FCAE yield similar predictions since the embeddings generated are of similar complexity. In an actuarial science context, this represents a significant improvement since it gives more flexibility for actuaries to determine higher concentrations of risky behaviour.

6 Conclusions and Future Work

In this paper, we presented a method of using external data sources to create geographical embeddings to predict downstream tasks related to socio-demographic risks. Our approach surpasses the baseline models since they generally lead to smaller errors in generalized linear model predictions. Our method successfully smooths regional effects of higher scale datasets for granular use in risk-related tasks. This tool is practical for insurance companies seeking to improve spatial risk analysis. Summarizing complex and heterogeneous data in an embedding gives actuaries more flexibility in risk modeling and will be in a better position to understand the risks in it's insurance portfolio. To the best of our knowledge, we have proposed the first geographic embeddings based on convolutional neural networks and have placed the CRAE as state of the art for territorial risk classification of property and casualty insurance.

The next steps could be incorporating more sources of data such as textual data from geolocalized tweets, or sentiment analysis from local newspapers. To better capture natural insurance risks, weather information could also be added. Algorithm 1 may already deal with cases of additional data sources (appenders) that may apply at different geographical scales by adding a loop over all data sources before line 5.

Other improvements to the model would be scale and rotation invariance, achieved by applying random rotations or scaling when generating the geographic data cube. Another approach would be to sample w uniformly between a predefined inverval of values or using geostatistical tools such as the variogram.

Acknowledgments

Neural networks were trained with Pytorch (Paszke, Gross, and et. al 2019) with the Poutyne framework (Paradis 2018). Maps were generated with Google maps R api with the ggmap library. This work was financially supported by the Society of Actuaries Hickman Scholar program and by the Chaire en actuariat de l'Université Laval (Blier-Wong, Baillargeon and Marceau : FO502320).

References

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8):1798–1828.

Cocos, A., and Callison-Burch, C. 2017. The language of place: Semantic value from geospatial context. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 99–104.

Eisenstein, J.; O'Connor, B.; Smith, N. A.; and Xing, E. P. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 1277–1287.

Jeawak, S. S.; Jones, C. B.; and Schockaert, S. 2019. Embedding geographic locations for modelling the natural environment using Flickr tags and structured data. In *European Conference on Information Retrieval*, 51–66.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105.

LeCun, Y., and Bengio, Y. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361(10):1995.

Levy, O., and Goldberg, Y. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, 302–308.

Noh, H.; Hong, S.; and Han, B. 2015. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 1520–1528.

Paradis, F. 2018. Poutyne: Keras-like framework for PyTorch.

Paszke, A.; Gross, S.; and et. al, M. 2019. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc. 8024–8035.

Saeidi, M.; Riedel, S.; and Capra, L. 2015. Lower dimensional representations of city neighbourhoods. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Tobler, W. R. 1970. A computer movie simulating urban growth in the Detroit region. *Economic Geography* 46(sup1):234–240.

Yao, Y.; Li, X.; Liu, X.; Liu, P.; Liang, Z.; Zhang, J.; and Mai, K. 2017. Sensing spatial distribution of urban land use by integrating points-of-interest and Google Word2Vec model. *International Journal of Geographical Information Science* 31(4):825–848.

Yao, Y.; Zhang, J.; Hong, Y.; Liang, H.; and He, J. 2018. Mapping fine-scale urban housing prices by fusing remotely sensed imagery and social media data. *Transactions in GIS* 22(2):561–581.

Zeiler, M. D.; Taylor, G. W.; and Fergus, R. 2011. Adaptive deconvolutional networks for mid and high level feature learning. In *Proceedings of the IEEE International Conference on Computer Vision*.