Predicting City Statistics from Aerial Imagery

James Cooney cooneyj@umich.edu
Sahil Farishta sahilf@umich.edu
Rupal Nigam rupaln@umich.edu
Shivam Sharma sshivam@umich.edu

University of Michigan
Ann Arbor

Abstract

The project's goal is to predict statistics about the area by analyzing aerial images of cities. The project aims to predict statistics including, but not limited to, crime rates, unemployment rates, and education level. The team plans to accomplish this by analyzing features such as lighting, land usage, streets, and other relevant features. The feature data will be fed into a convolutional neural network that will consider both the images with statistical data. The network will be used to find correlations between features and the statistics of interest and allow the team to determine what properties of a city lead to different statistics.

1. Introduction

This project's goal is to predict statistics about urban areas by analyzing aerial images. This will help determine whether urban city designs and geography are a factor when it comes to statistics about the area. These insights could shape future design decisions when it comes to urban city planning. If a correlation were to be found between murder rates and number of alleys, or land usage and education level, cities could be designed in a way to promote desirable statistics. We believe this is possible since we correlate certain features in cities. Narrow alleyways are correlated to crime while a large sprawling city with tall buildings has more people than a small flat town. Many of these descriptors can be extracted through images. Today, we have access to plenty of satellite imagery of cities in both daylight and night-time conditions. These images are processed to find information about the city, such as land usage and street layouts. Using this data, we believe it should be possible to pick up on trends that similar cities have. This problem is perfect to tackle using computer vision techniques since we are able to extract the features of cities through the processing of aerial images.

2. Contributions

Our project was a neural network designed to take in multiple aerial images of cities and use them in order to generate statistics for the input city. This was done by training the neural network using sample images and datasets. These images and data were provided via external sources. Our contributions were extracting and processing the data, designing the neural network, and training the algorithm. We performed validation experiments on the network, finding good hyperparameter values before producing test results and comparing the outputs to the true values in order to determine model performance.

3. Data

For this project, we used aerial imagery and statistical data from 25 of the most populous 30 cities in Germany.

3.1. Aerial Photos

We used four images of each city. These included a satellite daytime view, satellite night view, street map, and land use map. The data from the land use map comes from the Urban Atlas. This is a part of the European Union's Copernicus Program, an Earth observation initiative that monitors atmosphere, oceans, land-use, and climate change from several satellites. This program provides data from 2012 for 300 European cities in the form of shapefiles that can be viewed with ARCmap. Land-use is divided into 30 categories ranging from wetlands and pastures to dense urban usage. The other three views are all generated with GIS data in ARCmap. This allows all four aerial photos to be of the exact same area. Each of these four layers needs to be loaded and aligned to ensure they all show the same land area. Then the colors have to be manually added to the land-use map. This ensures that each category of land will be the same color across all cities used in the network. Finally, all layers are exported as a png. Each photo is 1446 x 787 pixels and covers an area roughly 30 x 17 miles centered on each chosen city.

Figure 1. Daytime satellite view of Munich

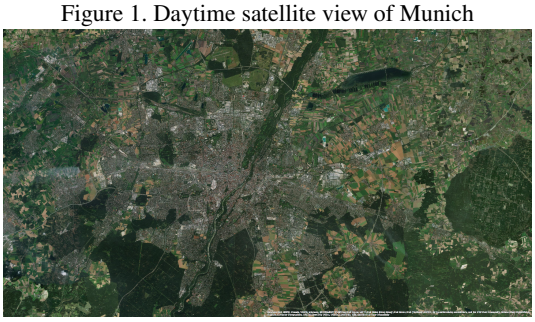


Figure 2. Land-use map of Munich from Urban Atlas

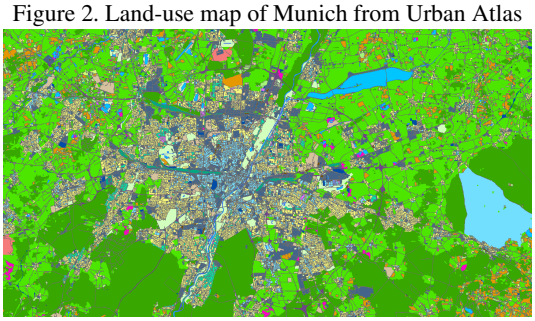


Figure 3. Nighttime satellite view of Munich

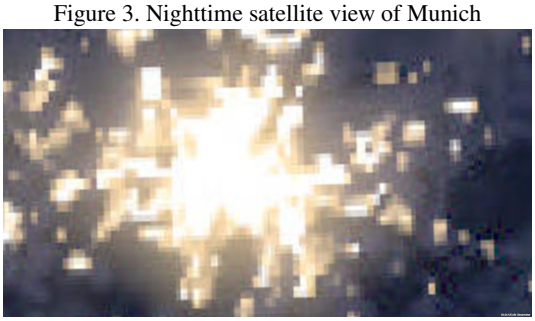
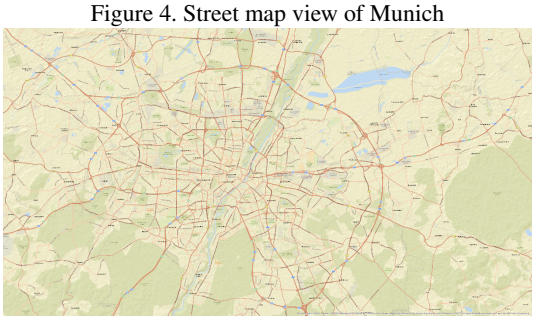


Figure 4. Street map view of Munich



3.2. Statistical Data

We found a database hosted by Eurostat, a directorate that provides statistics to the European Union, that contains city statistics. The database provides many options for querying, including by city, year, and statistic. These statistics include data for the largest 30 cities in Germany, which is a superset of the cities we are concerned with in this project. Some of the data available include education, living conditions, and population. We will use these statistics as the output from the neural net and treat them as the labels for the network. We plan on using the statistics for population, unemployment rate, percent of population who have completed college, number of murders and violent deaths, and average size of households. We are getting the data from the 2012 survey which matches the year our image data was provided.

Figure 5. Education statistics for German Cities

TIME	CITIES	INDIC_UR	Value
2012	Germany	Students in higher education	2,384,809
2012	Berlin	Students in higher education	159,658
2012	Hamburg	Students in higher education	91,439
2012	München	Students in higher education	107,103
2012	Köln	Students in higher education	86,764
2012	Frankfurt am Main	Students in higher education	56,360
2012	Stuttgart	Students in higher education	43,518
2012	Leipzig	Students in higher education	37,415
2012	Dresden	Students in higher education	42,739
2012	Düsseldorf	Students in higher education	36,119
2012	Bremen	Students in higher education	31,291
2012	Hannover	Students in higher education	39,330
2012	Nürnberg	Students in higher education	20,313
2012	Bielefeld	Students in higher education	30,341
2012	Wiesbaden	Students in higher education	14,159
2012	Augsburg	Students in higher education	23,040
2012	Bonn	Students in higher education	33,533
2012	Karlsruhe	Students in higher education	39,615
2012	Mönchengladbach	Students in higher education	7,013
2012	Kiel	Students in higher education	31,261
2012	Mannheim	Students in higher education	25,309
2012	Münster	Students in higher education	50,432
2012	Chemnitz	Students in higher education	10,331
2012	Braunschweig	Students in higher education	17,369
2012	Aachen	Students in higher education	46,977
2012	Wuppertal	Students in higher education	17,739

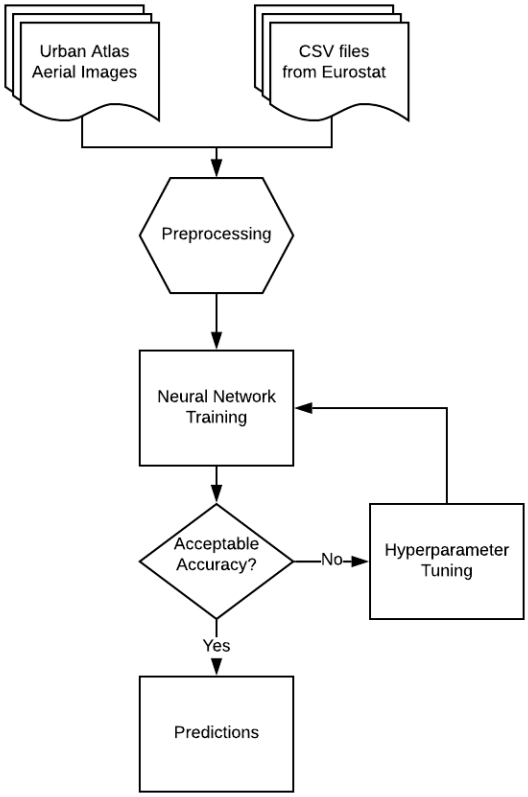
3.3. Data Usage

Our network takes in the imagery and processes it, producing estimates for the 5 statistics we are measuring. The output statistics are then compared with the true statistical values provided by the dataset. Our team was concerned the model would simply learn to identify the cities instead of generalizing features within the city imagery. To combat this, we split our data into 3 sets. 15 cities were used for training, 5 images were used for validation, and 5 were used for testing. The 5 validation images were used for determining model robustness and hyperparameter tuning and the network never trained on these images during the design stage. Once the network was fully designed and hyperparameters were selected, the model would then train on the training and validation sets and evaluated on the test set. Since the model had never seen the cities in the validation and test sets, the model would not perform well on these cities if it had simply learned to identify the city it was presented with. Thus we believe our system was able to generalize on features within the imagery in order to predict statistics about the cities.

4. Methods

Our project consisted of taking the aerial images and the data files and pre-processing them before feeding them into the neural network. We checked the validation results of the output of the model and adjusted the hyperparameters as necessary. Finally, when the results were within the tolerable range, the hyperparameters were finalized and the model was run on the test data to create the predictions.

Figure 6. Pipeline for Project



4.1. Pre-processing

The land-use, daytime satellite, and street map images were read in as RGB images, the night time satellite view was read as grayscale. These four images were all stacked into a single 10 channel image then each channel was normalized to zero-mean and a standard deviation of 1 across the entire data set. These normalized image layers were downscaled to be 512×512 in order to have square images to operate on. We also normalized the data by dividing each statistical value by the maximum of each category. This was done in order to ensure that all data lay between 0 and 1.

4.2. Neural Network

The neural network consists of 4 convolutional layers and 2 fully connected layers. Each convolutional layer consists of a convolutional, a max pool, a batch norm, and a dropout, followed by a ReLU activation function. The loss function used was Mean Squared Error and we used the Adam Optimizer with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-5}.

Figure 7. Convolutional Neural Network Architecture

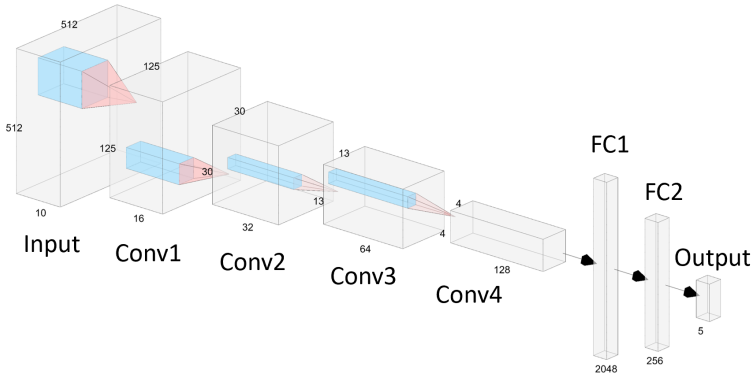
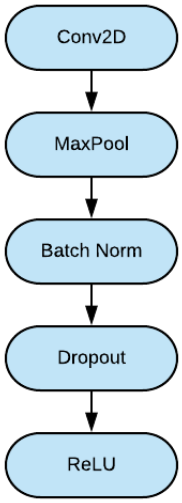


Figure 8. Structure of Convolutional Layers



The first convolutional layer takes in a $10 \times 512 \times 512$ image and uses a kernel size of 16×16 with a stride and padding of 2×2. The first max pool layer uses a kernel size of 3×3 and a stride of 2×2 with no padding. After the batch norm is applied, a dropout is applied with probability of dropping a node being 0.05. Finally the ReLU activation function is applied at the end. This same batch norm, droupout, and ReLU activation function end each convolutional layer. The output of this layer is $16 \times 125 \times 125$.

The second convolutional layer uses a kernel size of 9×9 with a stride and padding of 2×2. The second max pool layer uses a kernel size of 3×3 and a stride of 2×2 with no padding. The same batchnorm, droupout, and ReLU activation is applied at the end. The output of this layer is $32 \times 30 \times 30$.

The third convolutional layer uses a kernel size of 5×5 with a stride of 1×1 and no padding. The third max pool layer uses a kernel size of 2×2 and a stride of 2×2 with no padding. The same batchnorm, droupout, and ReLU activation is applied at the end. The output of this layer is $64 \times 13 \times 13$.

The fourth and final convolutional layer uses a kernel size of 5×5 with a stride of 1×1 and no padding. The fourth max pool layer uses a kernel size of 3×3 and a stride of 2×2 with no padding. The same batchnorm, droupout, and ReLU activation is applied at the end. The output of this layer is $64 \times 4 \times 4$.

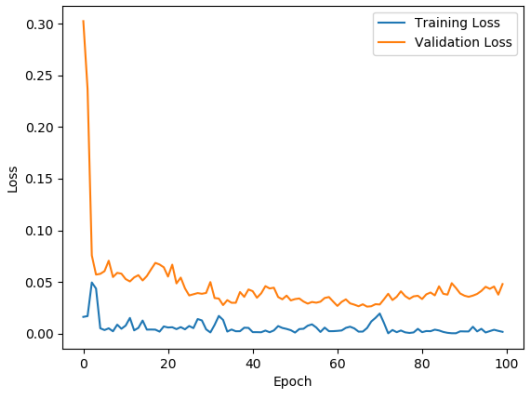
After this, the output is flattened and shaped into a 1x2048 vector that then is fed into the first fully connected layer that outputs a 1x256 vector. This is then put through a ReLU activation function before being fed into the second fully connected layer that outputs a 1x5. These are the 5 statistical values that are used as the predictions by the system.

The system was trained by splitting the 25 cities up into 15 training cities, 5 validation cities, and 5 test cities. Each epoch consisted of training on the training set with a batch size of 1. This meant seeing each city in the training set once per epoch. The validation phase consisted of running the network on the validation cities and determining the loss based on the outputs and the true known values. Again, a batch size of 1 used, so each image was seen once during the validation process. Finally, during the testing phase, each city was run through the network and the outputs were printed out to a file. We then analyzed each output file to determine the robustness of the model.

5. Experiments

To train the system, we ran the network on the 15 training cities for 100 epochs, using the validation city set to select hyperparameters. Once the hyperparameters were chosen, the system was run for 100 epochs using a combined set of the training and validation data. After 100 epochs, the model was analyzed using the graph shown in Figure 9. Since the validation loss was lowest at epoch 60, we then retrained a model for 60 epochs and used output of the test images as the final outputs for our project.

Figure 9. Training and Validation Loss for Model over 100 Epochs



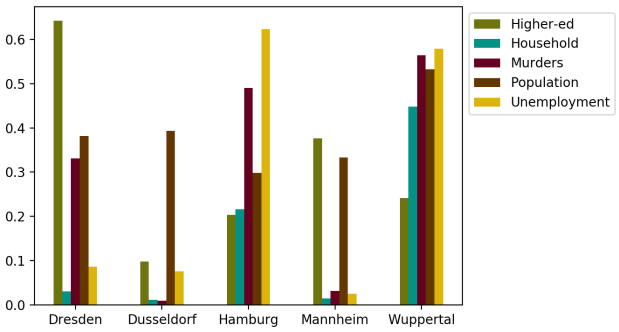
6. Results

We analyzed the performance of our network by studying various metrics. First, we looked at the normalized root mean squared error between the true value and the prediction. The normalized root mean squared is definied as follows:

$$NRMSE = \frac{\sqrt{(x_{true} - x_{prediction})^2}}{x_{true}} \quad (1)$$

Normalizing the error by the true value allows us to compare errors across the different statistics. Figure 10 displays the NRMSE for each city. We can see that the network clearly struggles with 'Wuppertal' and has the highest NRMSE for that city. This could be because Wuppertal is a smaller mountain city, which is distinct from the other cities in the dataset.

Figure 10. Normalized root mean squared error for each test city



To ensure we weren't introducing bias into the testing, we made sure to pick test data from uniformly throughout the training data state space. Figure 11 shows scatter density plots of the training data points and the test data points. The two subplots are the statistics with typically low values, and high values - "household" statistics were omitted for this plot because they have a very small variance.

Table 1. Test city CNN predictions, and true values

	Higher-ed		**Household**		**Murders**		**Population**		**Unemployment**	
	True	*Predicted*	*True*	*Predicted*	*True*	*Predicted*	*True*	*Predicted*	*True*	*Predicted*
Dresden	42,738.99	15,299.26	1.80	1.74	10.00	6.68	517,764.98	320,468.22	7.50	6.84
Dusseldorf	36,119.00	32,587.03	1.80	1.82	10.00	9.90	589,648.97	357,537.43	6.50	6.00
Hamburg	91,438.99	72,874.05	1.80	2.18	74.99	38.20	1,718,186.90	1,206,099.01	5.50	8.92
Mannheim	25,309.00	34,833.22	1.90	1.92	12.00	11.61	2,91,457.98	3,88,410.15	5.30	5.16
Wuppertal	17,738.99	22,015.50	2.00	1.10	6.00	9.38	3,42,570.00	1,60,241.60	8.90	3.75

Figure 11. Scatter density plot for test, and training data to ensure the test data was spread across the training data state space. Lighter colors are denser regions.

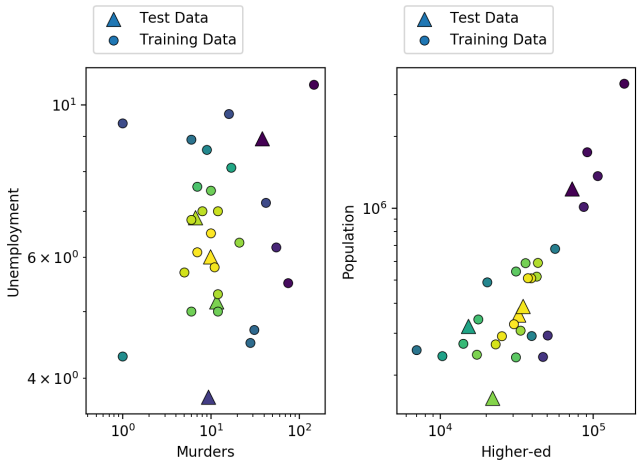
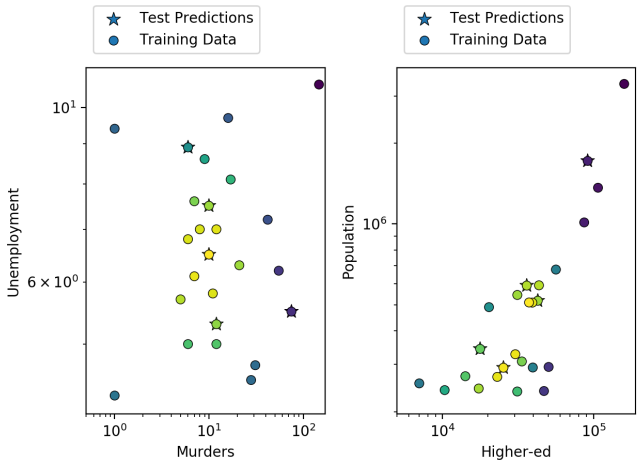


Figure 12. Scatter density plot for test *predictions*, and training data shows some predictions were almost the same as training data points. Lighter colors are denser regions.



We conducted a similar analysis to have a look at the distribution of the test data *predictions*. From Figure 12 we can see the distribution of the predicted statistics. Interestingly, some of the predictions are very close (almost overlapping) with test data. This leads us to believe that the CNN possibly overfit the data slightly.

Taking a look at the NRMSE, we will try to compare our model towards a random model. Since we normalize the data from 0-1, an average random error would be 0.5 in this range. Similarly, a random true value would be 0.5. Thus, a random NRMSE would be:

$$NRMSE_{random} = \frac{\sqrt{(0.5)^2}}{0.5} = 1 \tag{2}$$

Looking at our normalized root mean squared error for our 5 test cities as shown in Figure 10, we see that our model did indeed perform better than a random model, with our NRMSE values ranging from 0 to 0.65.

Finally, we took the data and calculated how many standard deviations it was from the true value. This was used to get a sense of how far off the predicted data was from the true value based on how scattered each statistic was across the entire dataset. The results are seen in Table 2. As we can see, the majority of the data was within 1 standard deviation of the correct value. Additionally, the only statistic that had results more than 2 standard deviations away from the true value was household size. This may come from the fact that household size has very little variance across our datasets which means small errors in household size are not penalized enough while the model is being trained. This results in the data being off by a large amount, even though to human eyes, 2.18 doesn't seem too different than 1.8.

7. Conclusion

Looking forward, we would like to expand our data collection to encompass more cities in order to improve learning abilities. Having only 25 cities to use severely limited our training data and the wide variance in city sizes made it difficult for the model to generalize. We would also like to have more city features within the data collected, such as population and building density maps and height maps for the buildings. This would provide a 3-dimensional view of the city to the model instead of relying only on the flat 2D map. Additionally, we would like to expand more on the land use statistics, so that we could identify specific build-

Table 2. Test city CNN predictions, and number of standard deviations from true values

	Higher-ed		**Household**		**Murders**		**Population**		**Unemployment**	
	Predicted	*# Standard Deviations Off*	*Predicted*	*# Standard Deviations Off*	*Predicted*	*# Standard Deviations Off*	*Predicted*	*# Standard Deviations Off*	*Predicted*	*# Standard Deviations Off*
Dresden	15,299.26	0.80	1.74	0.65	6.68	0.11	320,468.22	0.29	6.84	0.38
Dusseldorf	32,587.03	0.10	1.82	0.22	9.9	0.00	357,537.43	0.35	6	0.29
Hamburg	72,874.05	0.54	2.18	4.11	38.2	1.18	1,206,099.01	0.76	8.92	1.99
Mannheim	34,833.22	0.28	1.92	0.22	11.61	0.01	388,410.15	0.14	5.16	0.08
Wuppertal	22,015.50	0.12	1.1	9.72	9.38	0.11	160,241.60	0.27	3.75	3.00

ings within the city. This would allow the model to correspond features such as universities to more educated people. We would also like to include an activation map that visualizes what the model learns from each image and how these features correspond to various statistics. This would allow us to find potential flaws in the model and draw conclusions. These conclusions could demonstrate what features in a city lead to higher numbers of college graduates or a lower number of violent crimes and could inspire changes in future urban design and planning.

References

[1] Adrian Albert, Jasleen Kaur, and Marta C. González. Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. *CoRR*, abs/1704.02965, 2017.

[2] Copernicus. Urban atlas 2012. `https://land.copernicus.eu/local/urban-atlas/urban-atlas-2012`, 2012.

[3] European Commission. Eurostat. `https://ec.europa.eu/eurostat/web/cities/data/database`, 2012.

[4] Wikipedia contributors. List of cities in germany by population — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=List_of_cities_in_Germany_by_population&oldid=924101522`, 2019.