Xiaoran Wang, Yujing Wu, Hanyong Xu

MUSA 650 Final Project Report

Argentina Slums Classification and Identification

1. Introduction

Urbanization has been a global trend in the past few decades, bringing a large number of new residents to cities and leading to the number of urban settlements to exceed the rural settlements for the first time. This remarkable urban expansion further caused the inadequate developments of infrastructure, such as public transport, affordable housing, and utility (Kit & Reckien, 2012). The basic housing needs and living necessities of a significant amount of low-income urban residents forced the emergence of informal settlements, known as slums. Slums are densely populated urban residential areas with inadequate infrastructures and low-quality housing units. They are inhabited by low-income residents of the city, who suffer from poor living conditions such as insufficient living area, low accessibility to clean water, and poor tenure security. Currently, around one-quarter of the world's population lives in slums (Kuffer et al. 2016). UN-habitat indicates that from 2003 to 2010, many cities in the Global South experienced more than 70% increase in the slum area (Kuffer et al. 2016).

The slum-identification within a given city not only allows the government and other organizations to provide assistance and management accordingly but also serves as a metric for the effect of the overall city planning efforts. The identification task is mostly driven by the growth and persistence of existing slums, as well as the emergence of new slums. However, the research on slum identification usually lacks detailed information about slums. Considering the features of informal and rapid expansion, one can hardly acquire up-to-date data about slum extent, building conditions, and population density. Until recent years, remote sensing techniques have been used in large-scale slum classification to enhance our understanding of slum temporal dynamics. Due to the high cost of high-resolution remote sensing images, related research uses remote sensing images as the data source in machine learning to identify the slum areas (Baylé, 2016).

In this project, our goal is to identify the slum areas in the city of Buenos Aires, Argentina through remote sensing images with machine-learning approaches. We will first use satellite images from two districts in Argentina: Buenos Aires and Córdoba, to train a machine-learning algorithm that identifies which regions are slums, then we will use the algorithm to detect slums of the Buenos Aires over the span of past few years using the district's satellite images of different years.

2. Data

2.1 Training Data

The training data comes from the Kaggle Competition: Slums and informal settlements detection (https://www.kaggle.com/fedebayle/slums-argentina). This dataset contains georeferenced images of urban slums and informal settlements for two districts in Argentina: Buenos Aires and Córdoba. The image of Cordoba was taken on 2017-06-09 and the images of Buenos Aires on 2017-05-04. Each image comes from the Sentinel-2 sensor, with 32 by 32 pixels, 4 bands (bands 2, 3, 4, 8A), and 10-meter resolution. Images are in .tif format. For the Córdoba district, 37 out of 13,004 images are labeled as slums; for Buenos Aires, 1,008 out of 46,047 images are labeled as slums. The images of slums and non-slums in the two regions are shown below.
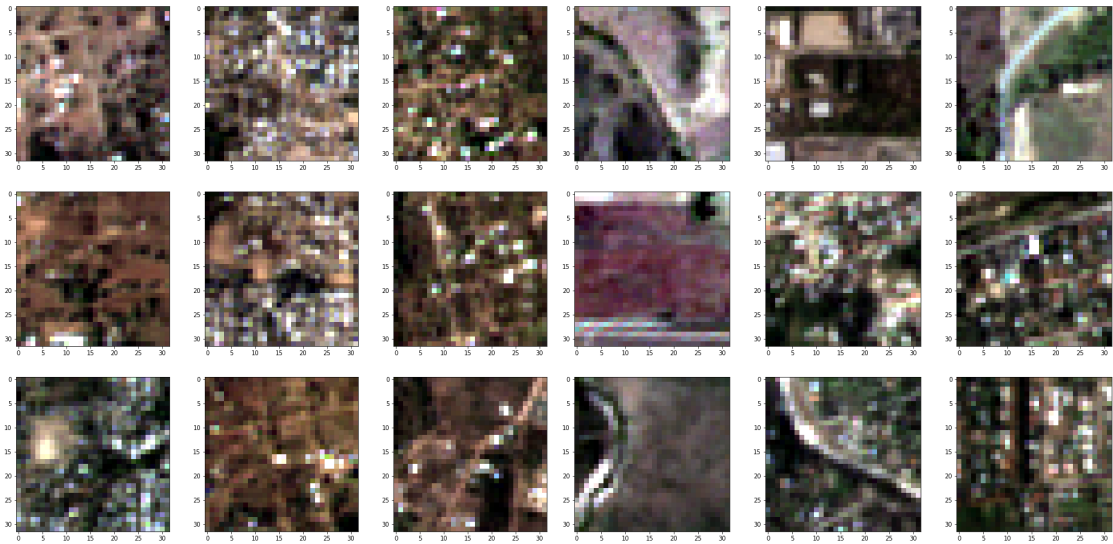
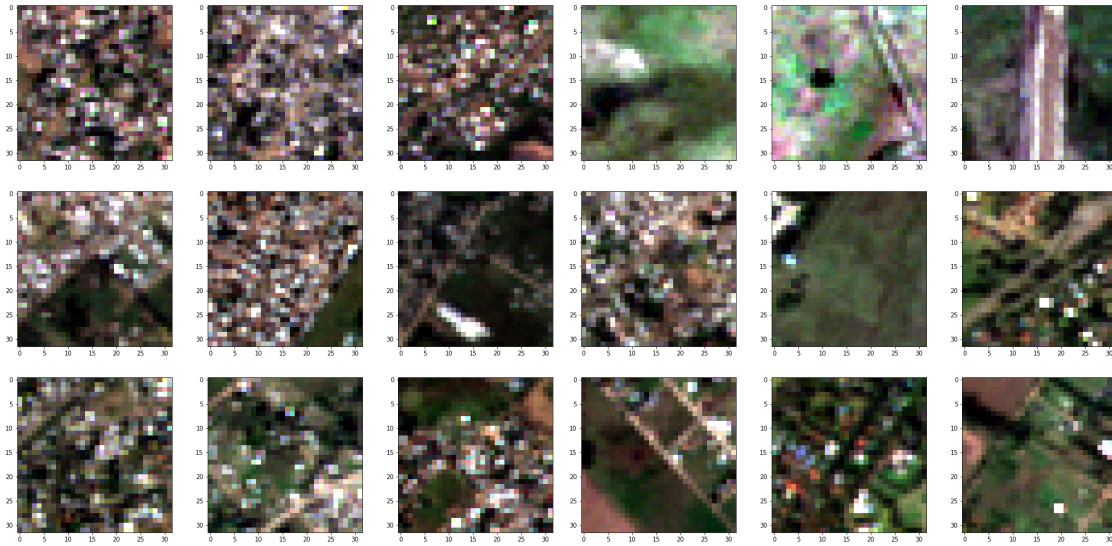Figure 1. Slums (left) and non-slums (right) in Cordoba

Figure 2. Slums (left) and non-slums (right) in Buenos Aires

2.2 Data for Application

We applied our model to track how the slums and informal settlements changed in these Buenos Aires over the years. We accessed Sentinel-2 images of the region taken on December 20th, 2017 and April 18th, 2020 from the USGS Earth Explorer (https://earthexplorer.usgs.gov/). After training our model, we applied it to these two images to examine if the slums and informal settlements expanded, gradually disappeared, or stayed the same between 2017 and 2020. In addition, we also compared our findings to the maps created by *Caminos de la Villa*, a project that tracks the location and development of slums and informal settlements in Buenos Aires. The two satellite images and the map created by the project are shown below.
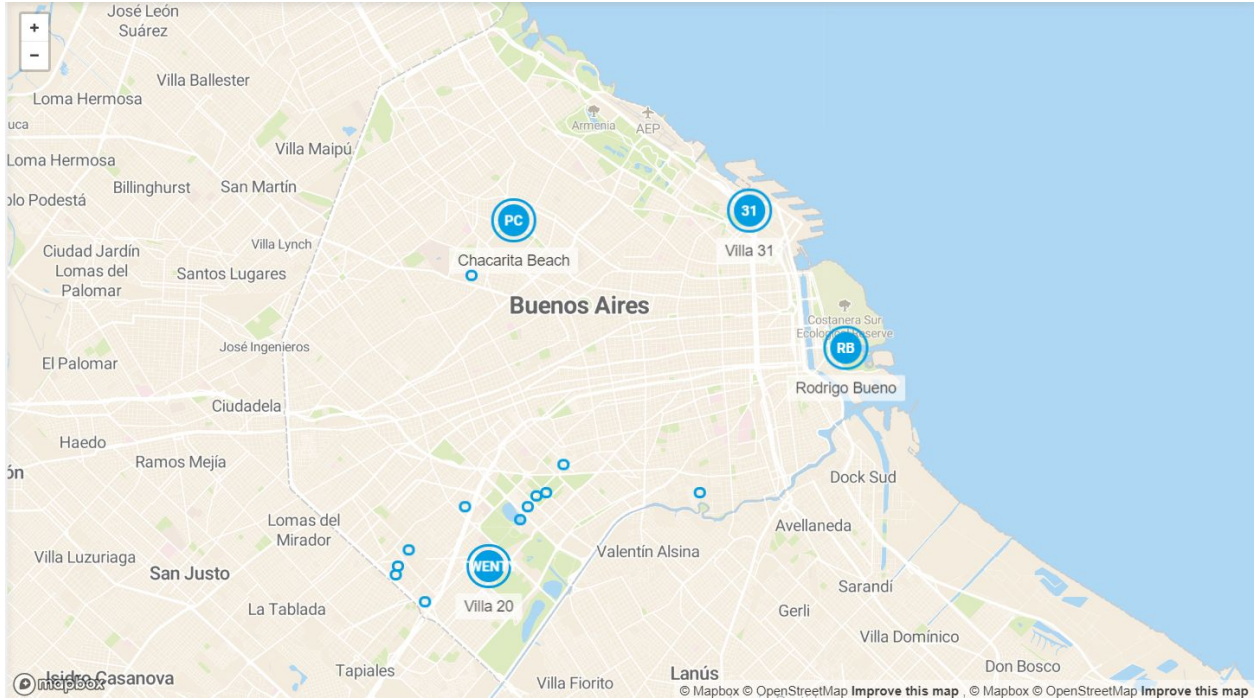
Figure 3. The locations of slums/informal settlements in Buenos Aires (those represented by large circles are undergoing urbanization)
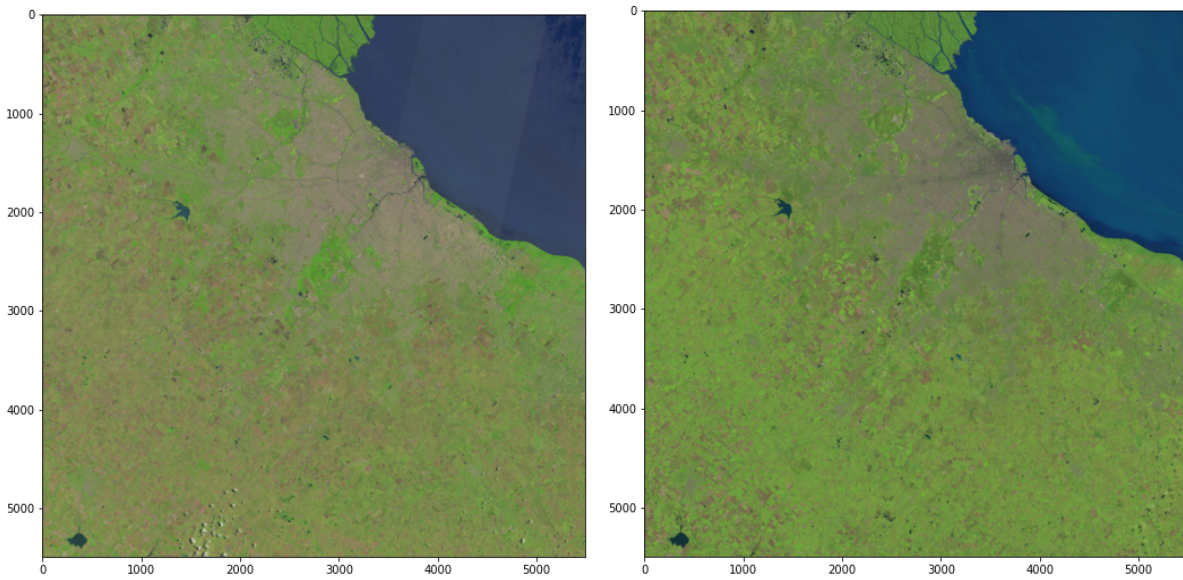

Figure 4. Satellite imageries of Buenos Aires taken in 2017 (left) and 2020 (right)

3. Methods

3.0 Overview

4

Our first step was to convert our data into a single JSON file to make it more convenient to read and process the data in Google Colab. We only used the RGB bands of our data since the larger images to which we wanted to apply our models only have three bands. Since our data is fairly unbalanced, we then subsetted our data to ensure we have equal amounts of slums and non-slum images. We used a variety of machine learning algorithms, including the support vector machine (SVM), convolutional neural networks (CNN), ResNet, and K-Means clustering. We evaluated the performance of these algorithms through different matrices including accuracy, receiver operating characteristic (ROC) curve, and confusion matrix.

After we finished training the model, we applied our model to the satellite imageries. Since the number of images of Cordoba is quite limited in the Kaggle dataset we used, we only applied our model to the satellite imageries of Buenos Aires. We cropped these images to the rough extent of the region. We then followed a similar process to the notebook DLBasics_SHIPS.ipynb showed in class to cut the images into smaller pieces of the size of our training data (32 * 32 pixels) and then normalized each piece. Finally, we applied our model on these smaller images and label the positions of images predicted to be slums/informal settlements in the satellite imageries.

3.1 SVM

Since this is a classification problem, we chose to start with the simple SVM model. First, the multi-channel images were normalized and combined together into single-band gray images. The images were then reshaped into single arrays to feed into the SVM model. The SVM model parameters were selected based on AUC values using GridsearchCV from the following combinations: kernel were selected from 'rbf', 'poly', 'sigmoid', gamma were selected from 1e-3, 'scale', and C were selected from 1, 2, 3, 10, 100. The combination with the best performance was 'rbf', 'scale', and C=1. Then we used this model to train on the whole dataset. Finally, we ran the trained model on the big satellite images and checked the locations of the informal settlements/slums.

3.2 CNN

5

Since the CNN is known to work well with image data, our second attempt was to train a CNN model. Before we trained the CNN, we split the balanced data into a training dataset (60% of the data) and a testing dataset (the remaining 40% of the data). We then normalized the data by dividing all data with the maximum pixel value of the training dataset. Besides, we binarized our label data into two columns: non-slums/ informal settlements and slums/ informal settlements.

As seen in Figure 5 below, we trained a CNN model to predict whether an image is a slum or not. We used the optimizer that implements the Adam algorithm and trained the model with 32 as the batch size and 30 as the number of epochs. Besides, we also used a function to adjust our learning rate as the number of epochs increases.

After we finished training the model, we applied our model to the two satellite imageries of Buenos Aires and identified the locations of the urban informal settlements/ slums as we mentioned above.

```
Model: "sequential_1"

Layer (type)                    Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)               (None, 32, 32, 64)        1792
_____
conv2d_2 (Conv2D)               (None, 32, 32, 64)        36928
_____
max_pooling2d_1 (MaxPooling2    (None, 16, 16, 64)        0
_____
conv2d_3 (Conv2D)               (None, 16, 16, 128)       73856
_____
max_pooling2d_2 (MaxPooling2    (None, 8, 8, 128)         0
_____
dropout_1 (Dropout)             (None, 8, 8, 128)         0
_____
flatten_1 (Flatten)             (None, 8192)              0
_____
dense_1 (Dense)                 (None, 512)               4194816
_____
dropout_2 (Dropout)             (None, 512)               0
_____
dense_2 (Dense)                 (None, 2)                 1026
=================================================================
Total params: 4,308,418
Trainable params: 4,308,418
Non-trainable params: 0
```

Figure 5. CNN model summary

3.4 ResNet

To try out a new type of model and compare its performance with the other approaches, we chose to implement the ResNet model. We splitted the balanced data into a 60% training set

and a 40% testing set. We then normalized the data by scaling the images into a range between 0 to 255, and then divided all pixel values by the maximum values. Besides, we binarized our label data into two columns: non-slums/ informal settlements and slums/ informal settlements.

As seen in Figure 6 below, we trained a ResNet model based on ResNet14v1 to predict whether an image is a slum or not. The figure below shows the model summary. We used the optimizer that implements the Adam algorithm and trained the model with batch size 32 and 80 epochs. Besides, we also used a function to adjust our learning rate as the number of epochs increases. Specifically, the learning rate starts as 0.001 and reduces to 0.0001 after 50 epochs. We also included real-time data augmentation, including rotating between 0-180 degrees, width and height shift between 0-0.1, as well as horizontal flip. After we trained the model, we applied the model to the two big satellite images and identified the locations of the informal urban settlements/ slums.

```
Layer (type)                     Output Shape         Param #    Connected to
==================================================================================
input_1 (InputLayer)             (None, 32, 32, 3)    0

conv2d_1 (Conv2D)                (None, 32, 32, 16)   448        input_1[0][0]

batch_normalization_1 (BatchNor  (None, 32, 32, 16)   64         conv2d_1[0][0]

activation_1 (Activation)        (None, 32, 32, 16)   0          batch_normalization_1[0][0]

conv2d_2 (Conv2D)                (None, 32, 32, 16)   2320       activation_1[0][0]

batch_normalization_2 (BatchNor  (None, 32, 32, 16)   64         conv2d_2[0][0]

activation_2 (Activation)        (None, 32, 32, 16)   0          batch_normalization_2[0][0]

conv2d_3 (Conv2D)                (None, 32, 32, 16)   2320       activation_2[0][0]

batch_normalization_3 (BatchNor  (None, 32, 32, 16)   64         conv2d_3[0][0]

add_1 (Add)                      (None, 32, 32, 16)   0          activation_1[0][0]
                                                                 batch_normalization_3[0][0]

activation_3 (Activation)        (None, 32, 32, 16)   0          add_1[0][0]

conv2d_4 (Conv2D)                (None, 32, 32, 16)   2320       activation_3[0][0]

batch_normalization_4 (BatchNor  (None, 32, 32, 16)   64         conv2d_4[0][0]

activation_4 (Activation)        (None, 32, 32, 16)   0          batch_normalization_4[0][0]

conv2d_5 (Conv2D)                (None, 32, 32, 16)   2320       activation_4[0][0]

batch_normalization_5 (BatchNor  (None, 32, 32, 16)   64         conv2d_5[0][0]

add_2 (Add)                      (None, 32, 32, 16)   0          activation_3[0][0]
                                                                 batch_normalization_5[0][0]

activation_5 (Activation)        (None, 32, 32, 16)   0          add_2[0][0]

conv2d_6 (Conv2D)                (None, 16, 16, 32)   4640       activation_5[0][0]

batch_normalization_6 (BatchNor  (None, 16, 16, 32)   128        conv2d_6[0][0]

activation_6 (Activation)        (None, 16, 16, 32)   0          batch_normalization_6[0][0]

conv2d_7 (Conv2D)                (None, 16, 16, 32)   9248       activation_6[0][0]

conv2d_8 (Conv2D)                (None, 16, 16, 32)   544        activation_5[0][0]

batch_normalization_7 (BatchNor  (None, 16, 16, 32)   128        conv2d_7[0][0]

add_3 (Add)                      (None, 16, 16, 32)   0          conv2d_8[0][0]
                                                                 batch_normalization_7[0][0]

activation_7 (Activation)        (None, 16, 16, 32)   0          add_3[0][0]
```

conv2d_9 (Conv2D)	(None, 16, 16, 32)	9248	activation_7[0][0]
batch_normalization_8 (BatchNor	(None, 16, 16, 32)	128	conv2d_9[0][0]
activation_8 (Activation)	(None, 16, 16, 32)	0	batch_normalization_8[0][0]
conv2d_10 (Conv2D)	(None, 16, 16, 32)	9248	activation_8[0][0]
batch_normalization_9 (BatchNor	(None, 16, 16, 32)	128	conv2d_10[0][0]
add_4 (Add)	(None, 16, 16, 32)	0	activation_7[0][0] batch_normalization_9[0][0]
activation_9 (Activation)	(None, 16, 16, 32)	0	add_4[0][0]
conv2d_11 (Conv2D)	(None, 8, 8, 64)	18496	activation_9[0][0]
batch_normalization_10 (BatchNo	(None, 8, 8, 64)	256	conv2d_11[0][0]
activation_10 (Activation)	(None, 8, 8, 64)	0	batch_normalization_10[0][0]
conv2d_12 (Conv2D)	(None, 8, 8, 64)	36928	activation_10[0][0]
conv2d_13 (Conv2D)	(None, 8, 8, 64)	2112	activation_9[0][0]
batch_normalization_11 (BatchNo	(None, 8, 8, 64)	256	conv2d_12[0][0]
add_5 (Add)	(None, 8, 8, 64)	0	conv2d_13[0][0] batch_normalization_11[0][0]
activation_11 (Activation)	(None, 8, 8, 64)	0	add_5[0][0]
conv2d_14 (Conv2D)	(None, 8, 8, 64)	36928	activation_11[0][0]
batch_normalization_12 (BatchNo	(None, 8, 8, 64)	256	conv2d_14[0][0]
activation_12 (Activation)	(None, 8, 8, 64)	0	batch_normalization_12[0][0]
conv2d_15 (Conv2D)	(None, 8, 8, 64)	36928	activation_12[0][0]
batch_normalization_13 (BatchNo	(None, 8, 8, 64)	256	conv2d_15[0][0]
add_6 (Add)	(None, 8, 8, 64)	0	activation_11[0][0] batch_normalization_13[0][0]
activation_13 (Activation)	(None, 8, 8, 64)	0	add_6[0][0]
average_pooling2d_1 (AveragePoo	(None, 1, 1, 64)	0	activation_13[0][0]
flatten_1 (Flatten)	(None, 64)	0	average_pooling2d_1[0][0]
dense_1 (Dense)	(None, 2)	130	flatten_1[0][0]

Total params: 176,034
Trainable params: 175,106
Non-trainable params: 928

Figure 6. ResNet model summary

3.5 K-Means

Finally, we would like to see how unsupervised classification works in this problem. Thus, we conducted a K-Means clustering analysis on the Kaggle dataset. Since the data is heavily unbalanced, we subset the data before classifying. The ratio between positive and negative labeled images in the dataset that was used in the K-Means clustering classification was almost 1:1. It includes all 1045 slum-labeled images and the first 1050 non-slum labeled images. Since the goal was to identify two types of land use, a K-Means clustering for 2 clusters was applied to all images from the subsetted dataset. We then calculate the confusion matrix to

examine the classification accuracy. We then classified one image from each label and applied 2-cluster and 5-cluster K-Means segmentation to see how the pixel distributed pattern for these two types of land uses.

After that, we applied K-Means clustering to the satellite images from USGS to see whether the unsupervised approach can identify the difference between slum and non-slum areas in the mixed land use images. Additionally, we also want to observe whether the slums have expanded in these three years. The original image covers a fairly large area, but K-Means works better for the small-scale site. Thus, we selected the location of *Villa 20*, where it appeared as a slum land use in both CNN predicted results and the slum map. We zoomed the images to this target slum site by cropping the images to the size of (100 * 100 pixels). The 2-cluster and 5-cluster classifications were applied to the new dataset.
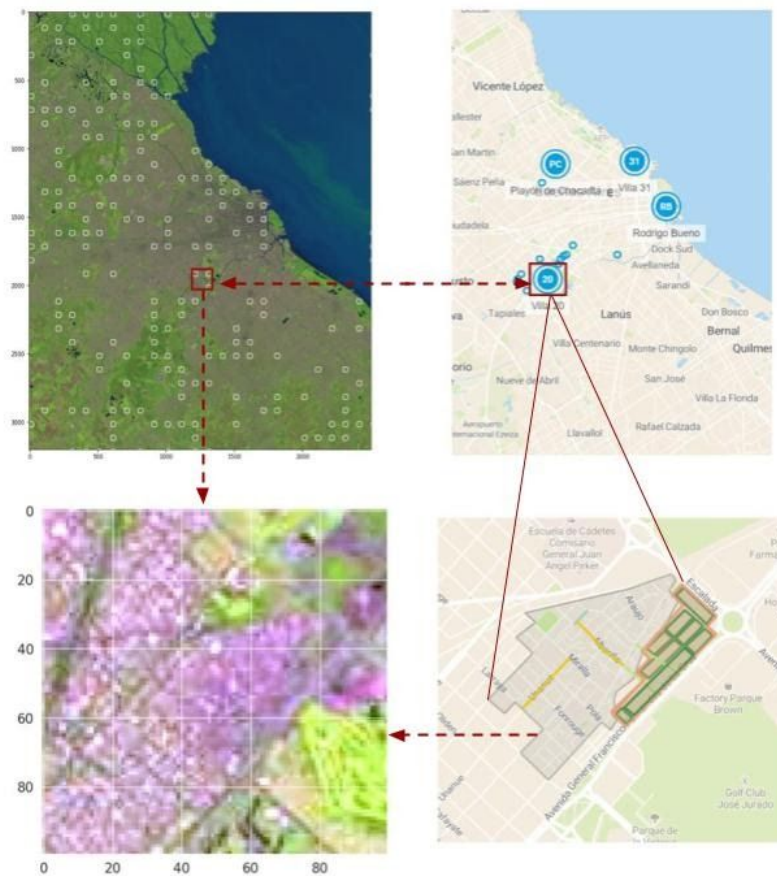
Figure 7. Predicted result of 2020 USGS image from CNN model (top left); Argentina slum map as reference of slum location (top right and bottom right); Cropped clum area from 2020 USGS image for K-Means Classification (bottom left).

4. Results

4.1 SVM

      After selecting the best parameters based on GridsearchCV, we ran the model with that combination. The 5 fold cross-validation gives an accuracy score of 68%. Figure 8 below shows the confusion matrix of the SVM model. Based on the normalized matrix value, one can see that the model predicts quite well, with high true positive and true negatives. Based on the ROC plot below, the model has an AUC value of 0.75.
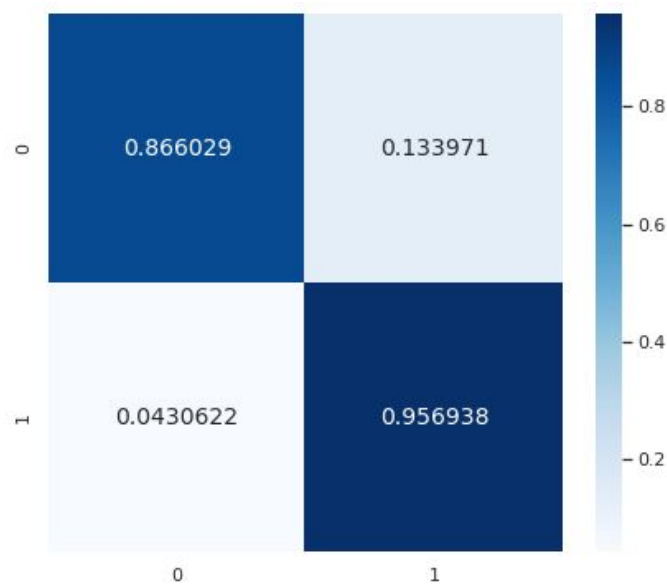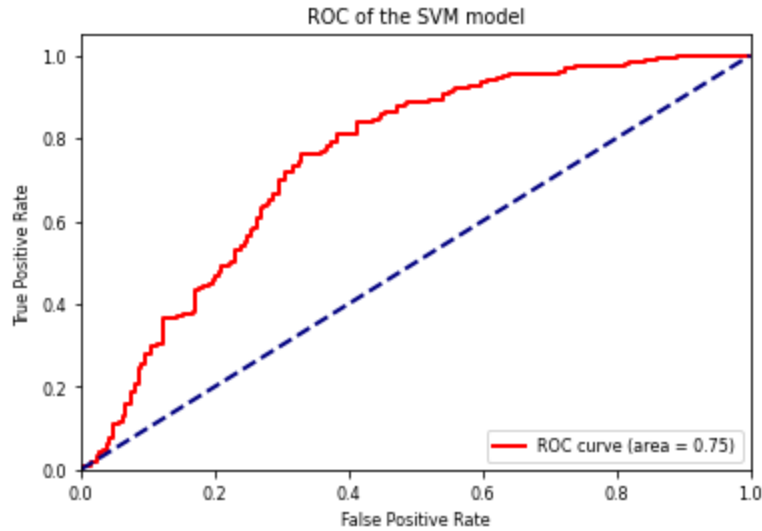

Figure 8. Confusion matrix for the SVM model

Figure 9. ROC curve of the SVM model

Although the model had a good performance based on cross-validation, the model did not work well on unseen data. Specifically, as the prediction images shown below, when we applied the model on the large satellite image and tried to classify its patches into urban slums and non-slums, the SVM model failed to identify the ocean and predicted too many positive cases. On the other hand, the predicted positive patches on lands did cluster around urban areas, which makes sense.

In terms of the change in urban slum over the years, our model predicted 161 urban slum patches in 2017, while 148 patches in 2020. This means over the three-year span, the number of urban slums has been decreasing. However, if we only compare the number of positive patches excluding the ones in the ocean, there seems to be more slums in 2020 than in 2017.
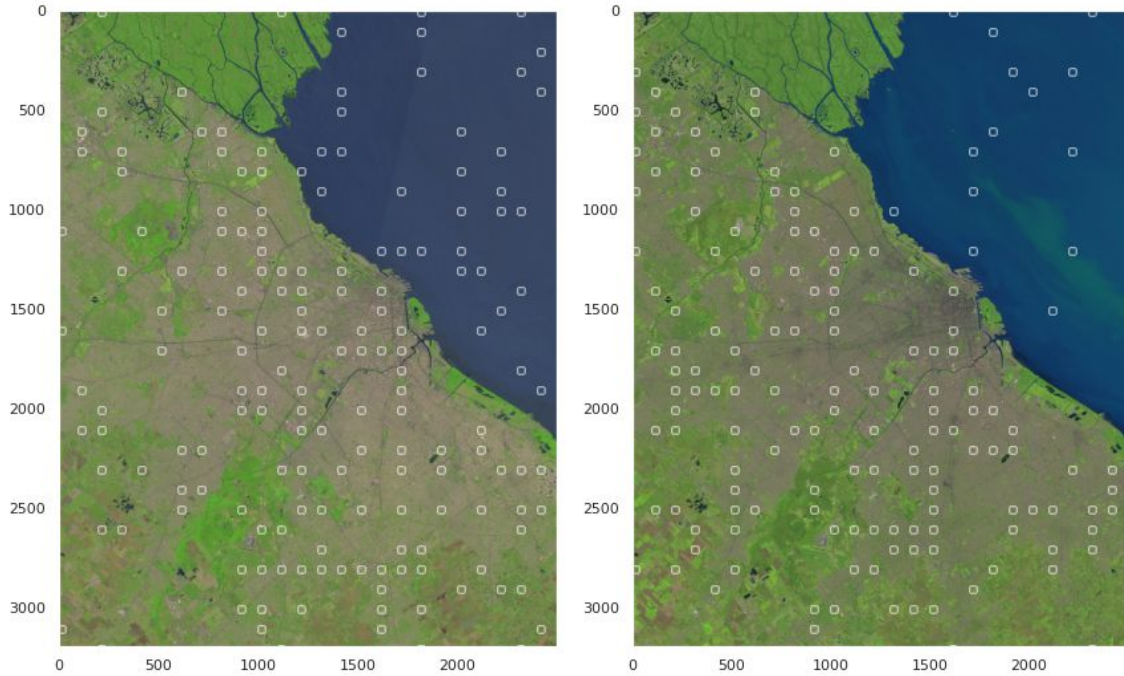
Figure 10. SVM applied to the satellite imageries of Buenos Aires taken in 2017 and 2020

4.2 CNN

Figure 11 and 12 below show the change in model accuracy and loss when we trained the CNN model. As the number of epochs increased, the training accuracy kept increasing while the testing accuracy started to flatten at around 0.75. Similarly, the training loss kept decreasing while the validation loss started increasing at around the 15th epoch.
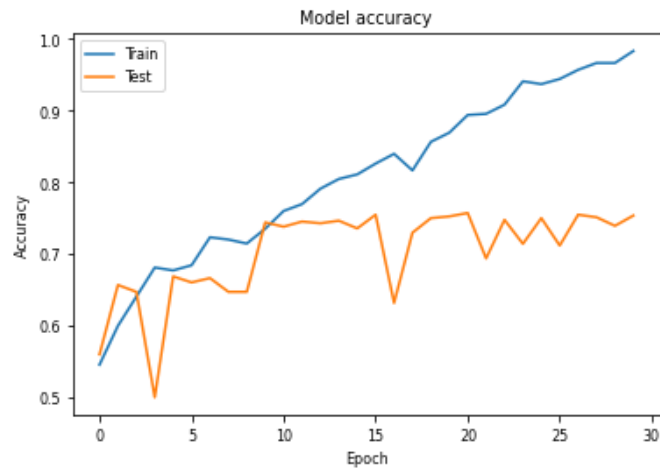
Figure 11. The trend in model accuracy when we trained the CNN model
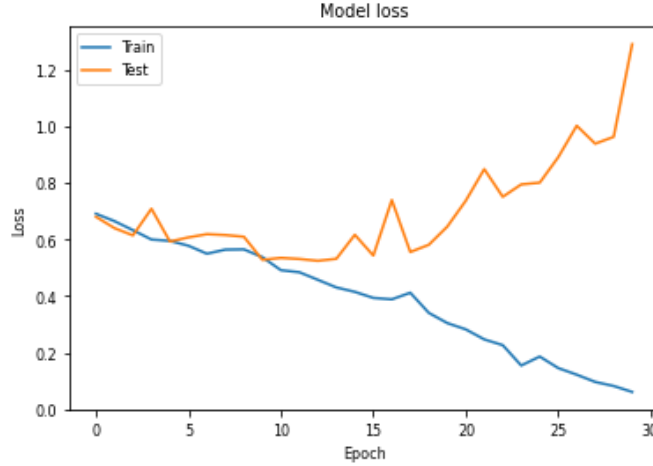
Figure 12. The trend in model loss when we trained the CNN model

Our CNN model achieved an accuracy of 0.754. As seen in Figure 13 below, the AUC of the model is 0.78.
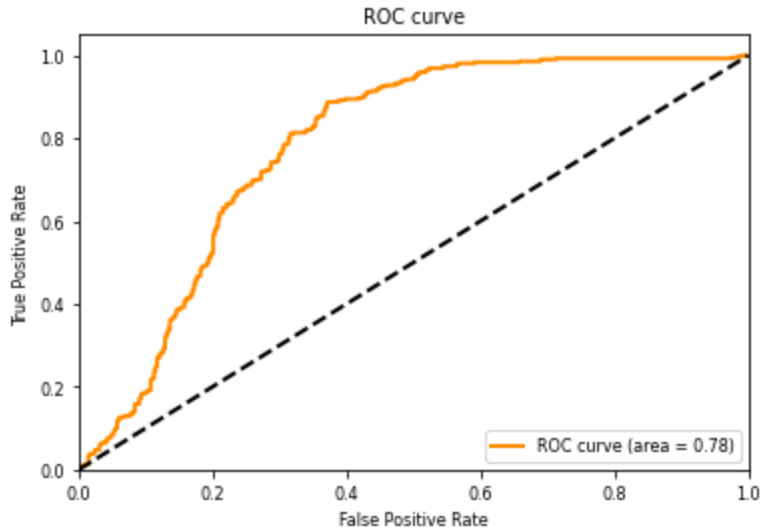
Figure 13. ROC curve of the CNN model

Below in Figure 14 is the confusion matrix of our CNN model. The matrix shows that we have a fairly high true positive rate and a slightly lower true negative rate. Some non-slum images were falsely classified as slums while only a few slum images were falsely classified as non-slums.
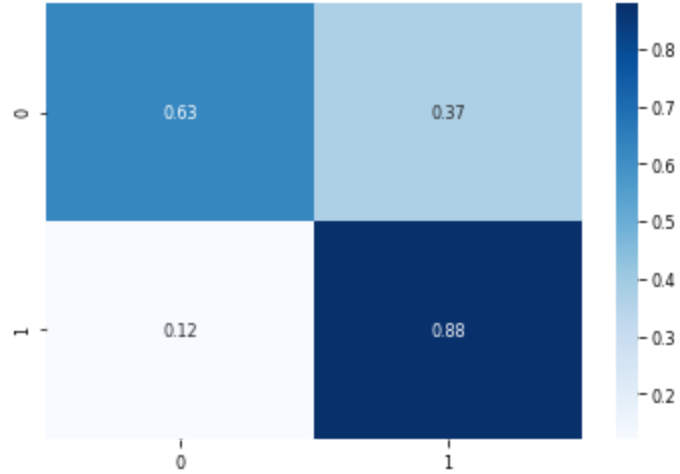
Figure 14. The confusion matrix of the CNN model

The predictions made by our model for the satellite imageries are shown in Figure 15 below. In the map created by *Caminos de la Villa*, most slums/informal settlements are located in southern as well as eastern part of Buenos Aires. There are also a couple of slums near the center of the city. As seen in Figure 15 below, our model managed to predict the slums in eastern as well as the center of Buenos Aires. Between 2017 and 2020, the predictions of our model indicated there may have been an increase in the number of slums, especially in the suburban areas around Buenos Aires.
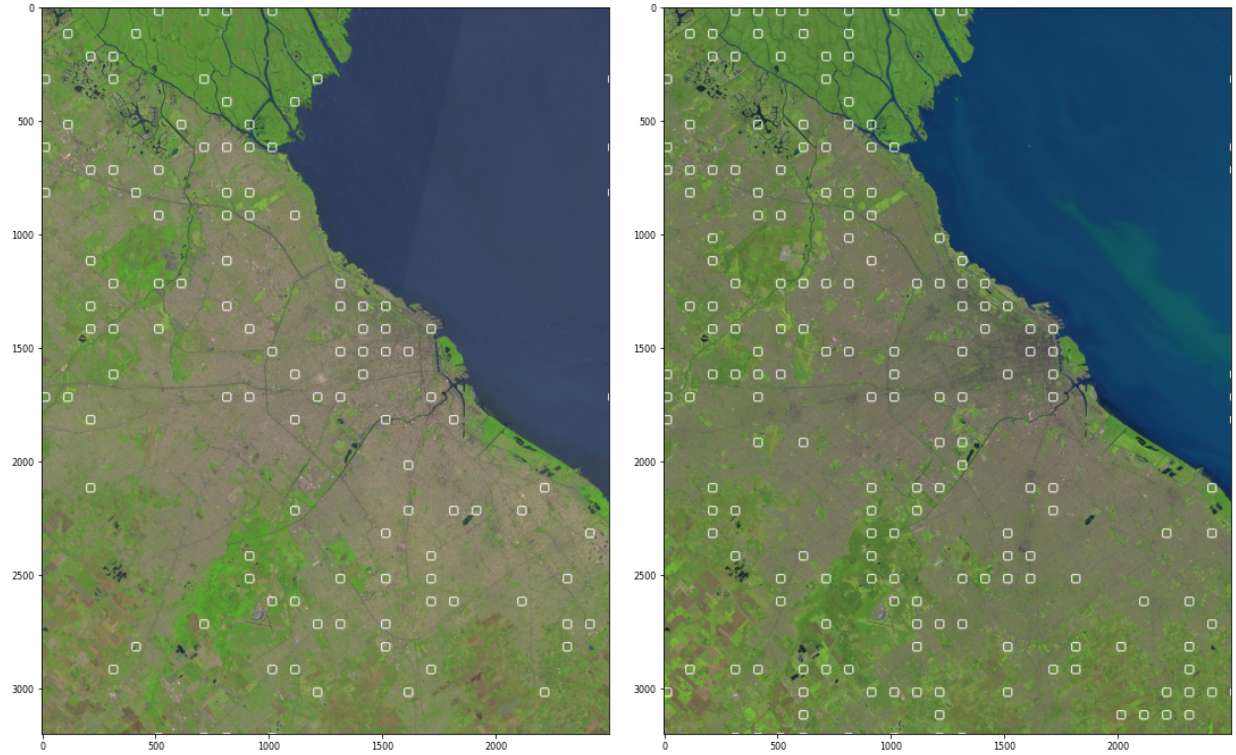
Figure 15. CNN applied to the satellite imageries of Buenos Aires taken in 2017 and 2020

4.4 ResNet

Figure 16 below shows the change in model loss when we trained the ResNet model. As the number of epochs increased, the training accuracy kept increasing while the testing accuracy started to flatten to around 0.79 starting at epoch 50 when we reduced the learning rate. The validation loss first fluctuated but then gradually decreased while the training loss is always decreasing.
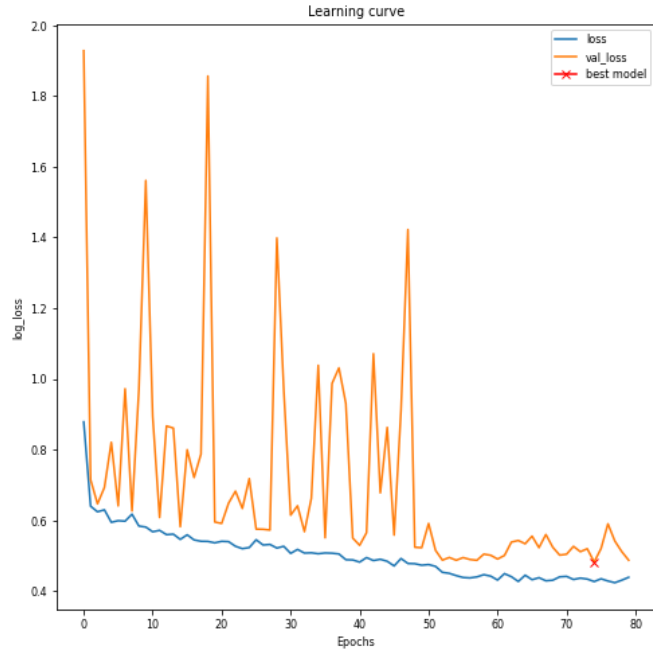
Figure 16. The loss trend when we trained the ResNet model

Our ResNet model achieved an accuracy of 0.807. As seen in Figure 17 below, the AUC of the model is 0.89.
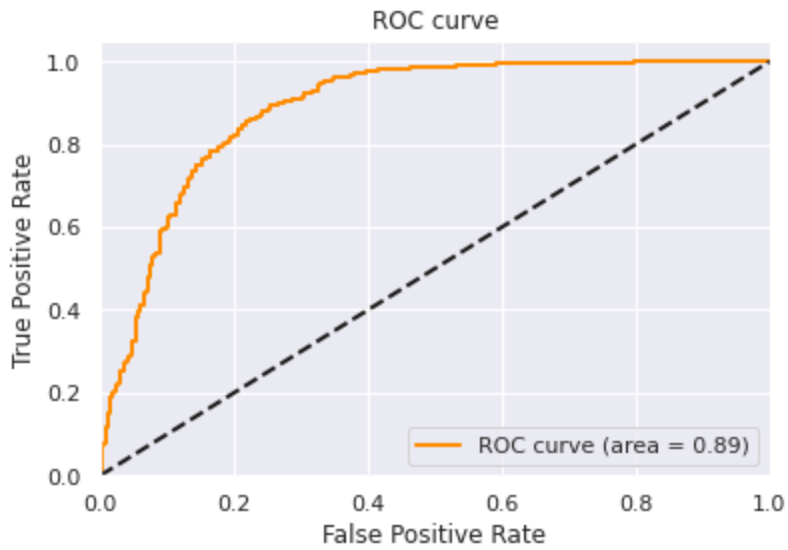

Figure 17. The ROC of the ResNet model

Below in Figure 18 is the confusion matrix of our ResNet model. The matrix shows that we have a fairly high true positive rate and a slightly lower true negative rate. Some non-slum images were falsely classified as slums while only a few slum images were falsely classified as non-slums.
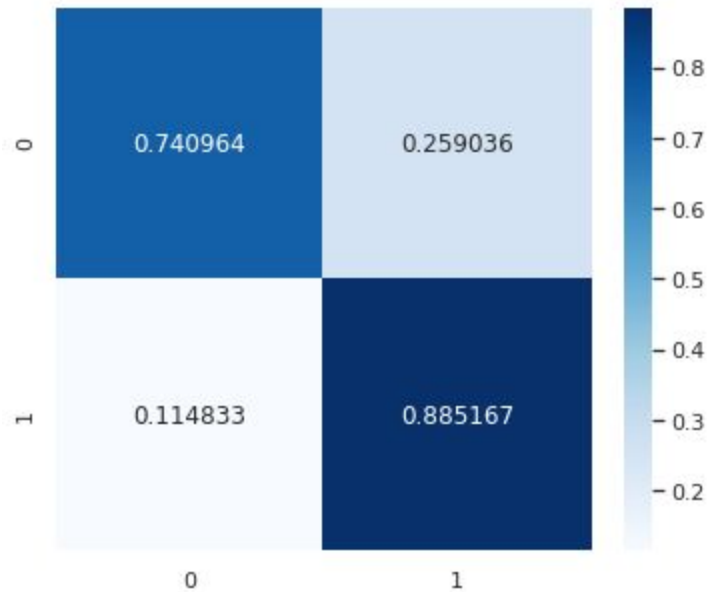

Figure 18. The confusion matrix of the ResNet model

Despite the good performance of this model, when we applied it to the big satellite images to find the locations of the urban informal settlements/ slums, we could not find any positively labeled prediction. In other words, the model could not find any informal settlements or slums in the image, which is contrary to our initial assumption. We will further discuss this issue in the discussion section below.

4.5 K-Means

Below, Figure 19 is the confusion matrix of the K-Means clustering results. It shows a high true negative rate, so the classification worked well in identifying the non-slum images. Both true positive and false positive rates are fairly low, which indicates that K-Means hardly classified any image as a slum image.
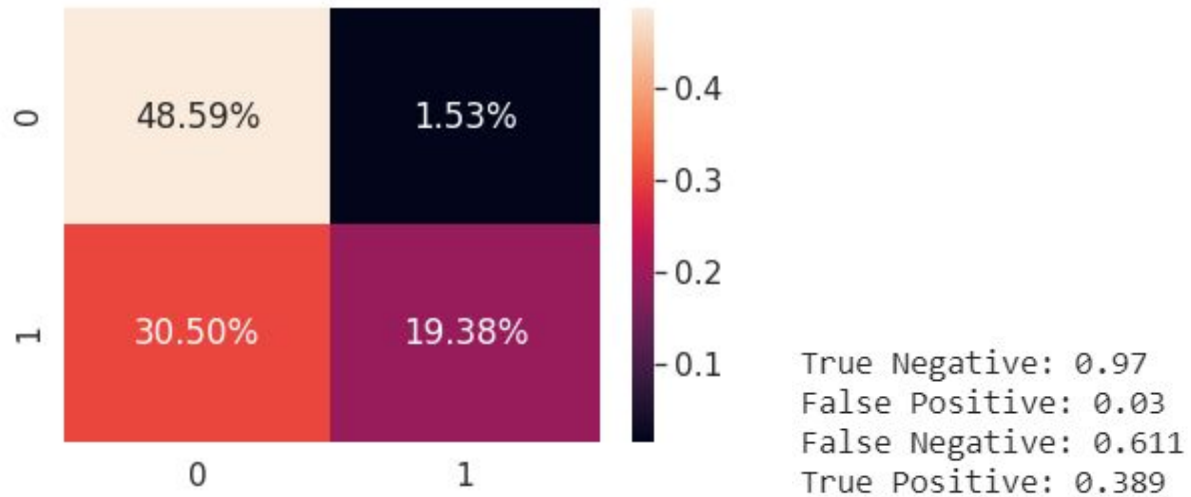
Figure 19. Confusion matrix for K-Means clustering analysis.

The six images in Figure 20 are the sample images from slum images and non-slum images. As mentioned in the caption, the top row are the slum image and the bottom row are the non-slum images under 2-cluster and 5-cluster K-Means classification. One can observe that the distribution of pixel value in slum images is more segmented, but in non-slum images, the pixel values distribution tends to be continuous and clustered. The images below indicate that K-Means is a feasible approach to differentiate slum and non-slum land uses by showing the pixel distributed patterns.
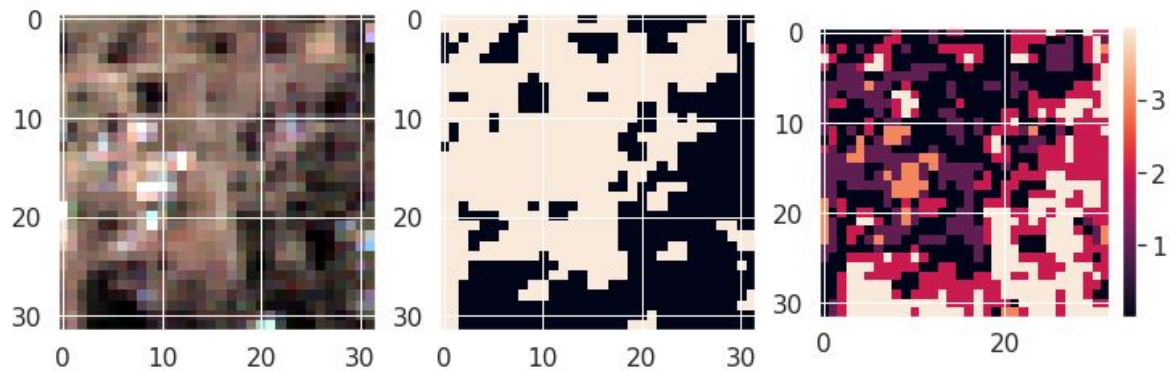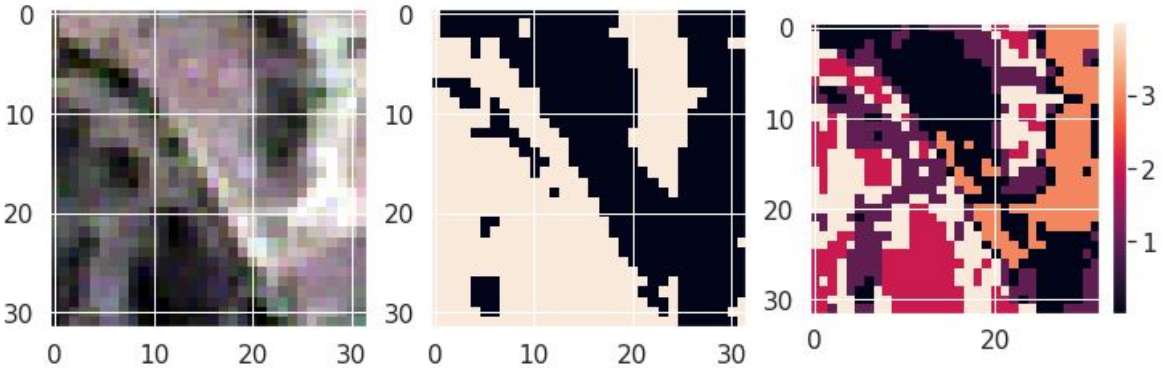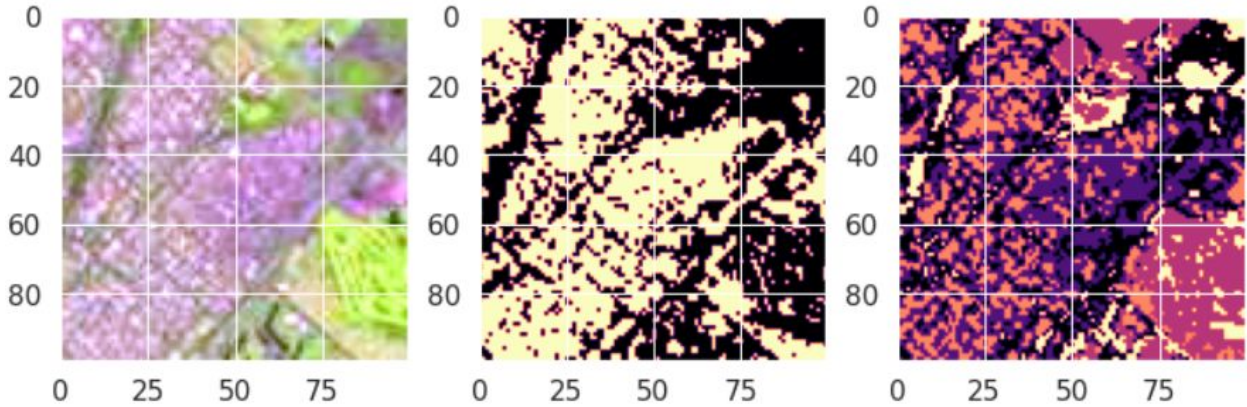
Figure 20. K-Means segmentation with 2 and 5 clusters. Top row: sample images for slum; bottom row: sample images for non-slum.

We applied 2-cluster and 5-cluster K-Means classifications to a Buenos Aires slum site in 2017 and 2020 images for two purposes: one is to explore the performance of K-Means on the mixed land uses images, and the second is to see if the slum expanded or shrunk over the three years. Figure 21 below is the results. The 2017 5-cluster image (top-right) provides a fair identification of the slum area: the large proportion of purple pixels at the center of the image corresponds to the slum. For the rest of the images, slum areas are largely mixed with the other types of land use, which also make it hard to observe a clear expanding or shrinking tendency through these images. Thus, for the slum identification, K-Means works better when the image land type is binary (slum or non-slum). The performance and validity would be affected in the mixed land uses images. K-Means can barely identify slum out of other normal constructed land uses since slum is usually adjacent or embedded in the urban area.
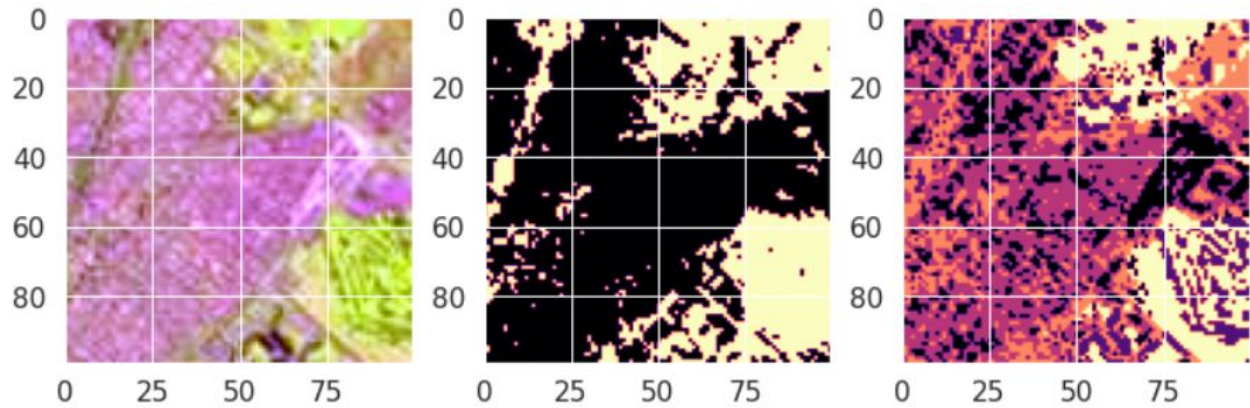
Figure 21. K-Means segmentation with 2 and 5 clusters for the predicted slum area: 2017 and 2020 scaled images (left column); images with 2-clustered classification (middle column); images with 5-clustered classification (right column).

5. Discussion

5.1 Conclusion

K-Means clustering as an unsupervised approach of classification was able to classify the land pattern for binary images, but it was no longer applicable when classifying slums in images with mixed land uses. The urban environment largely reduced its applicability.

The SVM model was able to conduct classification of slums and non-slums fairly, although it failed to identify the ocean and predicted many false positives. Thus, either a better model needs to be used or more diverse data should be gathered in order for the model to perform well.

Within our training dataset, CNN did a relatively good job in identifying images that are slums. However, in the larger satellite images, the model seemed to have predicted more slums in Buenos Aires than there actually are. In conclusion, the model should be able to help governments and other organizations identify areas that are potentially slums or informal settlements. But ground truth data still needs to be collected to confirm the presence and locations of the slums.

By comparing the predictions produced by SVM and CNN models on 2017 and 2020 Buenos Aires images, we see relatively different results in the change in slums and informal settlements in the area. SVM predicted a decrease in slums while CNN predicted an increase in the slums. Since the training data we have is not georeferenced, we do not have enough ground

truth data of the actual locations of the slums in the suburban areas of Buenos Aires. Thus, based on the performance of the model within the city of Buenos Aires, we conclude that the prediction made by CNN is more reliable and the slums may have increased over the past three years.

5.2 Future Improvements

We have identified several ways to improve our project in the future. Since the resolution of the data we used this time is quite low, it became quite challenging to extract useful information from the images. Thus, we would like to obtain image data with higher resolution that contains more information in the future. In addition, while we were able to split our data and evaluate the performance of our models on the validation datasets, we had trouble assessing the quality of our predictions on the larger satellite imageries. Though the *Caminos de la Villa* provides a map of the slums in Buenos Aires, we are also interested in the slums in suburban areas around the city. Thus, in the future we would like to find datasets from other sources such as OpenStreetMap to locate the slums in the area and better assess the performance of our model. Lastly, the ResNet model, in spite of its good performance on the training and validation data, it fails to identify any urban slums with the large satellite images. We suspect this might be because of overfit and lack of data. Future improvements include using more data as mentioned above, and use other regularization methods to control overfit.

References

Baylé, F. (2016). *Detección de Villas y Asentamientos Informales en el partido de La Matanza mediante teledetección y sistemas de información geográfica* (Doctoral dissertation, Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales).

Kit, O., Lüdeke, M., & Reckien, D. (2012). Texture-based identification of urban slums in Hyderabad, India using remote sensing data. *Applied Geography*, *32*(2), 660-667.

Kuffer, M., Pfeffer, K., & Sliuzas, R. (2016). Slums from space—15 years of slum mapping using remote sensing. *Remote Sensing*, *8*(6), 455.