# Automatic Deep Learning-based monitoring of security fences in drone orthophotos

*Alessio* Brugiavini,[1] *Marco* Romanelli[1]

**Abstract.** Nowadays, during the safe demarcation of work areas, it's necessary to recognize and monitor the yard's perimeter due to safety pourpose. In this scenario, CNN's Classification Network is the perfect choice in order to satisfy this request.We compute square patches representing the yard perimeter, these patches will form the training and validation dataset for a VGG16 CNN's, which is perfect for object recognition and classification of fence presence in the yard field. The next step will be testing the algorithm with another dataset used for the prediction phase. Result show that the proposed approach is a good method to identify security flaw in a yard fence, with less effort in term of time.

## 1 Introduction

Related to this project, the domain is to use a Deep Learning Architecture for images classification about working areas, for safety pourpose. The task consist in taking orthophotos [1], which are aerial photograph or satellite imagery geometrically corrected ("orthorectified") used to measure true distances, because it is an accurate representation of the Earth's surface, having been adjusted for topographic relief, lens distortion, and camera tilt.Given this ortophotos we crop the perimeter of the yard manually with a QGIS [2] tool,QGIS is a free and open-source cross-platform desktop geographic information system (GIS) application that supports viewing, editing, printing, and analysis of geospatial data, to detect image features. the cropped yard perimeter is input data to our software, which provide to track square patches which are usefull to find image features. The project could be splitted in two phases,the first phase is the patch extraction phase, and the second is the Deep Learning phase. In order to extract the patch we used Python to code the related scripts. We had to use specific libraries because OrthoRGB was the image format taken by a drone from aloft of the yard. Photos has been taken by a camera with a 45 degrees angle. After that, we manually track the yard perimeter with a QGIS tool, and give it as input for the specific script. Square patches has been draw and automatically saved in a folder. The deep learning phase consist in taking the extracted patches as the dataset needed to train e validate the network. VGG16 is the CNN's classification network used for this project. For testing the network, we choose another dataset, consisting in another yard's set of images. After that we put the prediction's result on QGIS again, to clearly see how the patches has bees identified by the network.

## 2 Related Work

### 2.1 ChangeNet: A Deep Learning Architecture for Visual Change Detection

The increasing urban population in cities necessitates the need for the development of smart cities that can offer better services to its citizens. Drone technology plays a crucial role in the smart city environment and is already involved in a number of functions in smart cities such as traffic control and construction monitoring. A major challenge in fast growing cities is the encroachment of public spaces. A robotic solution using visual change detection can be used for such purposes. For the detection of encroachment, a drone can monitor outdoor urban areas over a period of time to infer the visual changes. Visual change detection is a higher level inference task that aims at accurately identifying variations between a reference image (historical) and a new test image depicting the current scenario. In case of images, the challenges are complex considering the variations caused by environmental conditions that are actually unchanged events. Human mind interprets the change by comparing the current status with historical data at intelligence level rather than using only visual information. In this paper, we present a deep architecture called ChangeNet [3] for detecting changes between pairs of images and express the same semantically (label the change). A parallel deep convolutional neural network (CNN) architecture for localizing and identifying the changes between image pair has been proposed in this paper. The architecture is evaluated with VL-CMU-CD street view change detection, TSUNAMI and Google Street View (GSV) datasets that resemble drone captured images. The performance of the model for different lighting and seasonal conditions are experimented quantitatively and qualitatively.

### 2.2 Change Detection in Unmanned Aerial Vehicle Images for Progress Monitoring of Road Construction

Currently, unmanned aerial vehicles are increasingly being used in various construction projects such as housing developments, road construction, and bridge maintenance. If a drone is used at a road construction site, elevation information and orthoimages can be generated to acquire the construction status quantitatively. However, the detection of detailed changes in the site owing to construction depends on visual video interpretation. This study [4] develops a method for automatic detection of the construction area using multitemporal images and a deep learning method. First, a deep learning model was trained using images of the changing area as reference. Second, we obtained an effective application method by applying various parameters to the deep learning process. The application of the time-series images of a construction site to the selected deep learning model enabled more effective identification of the changed areas than the existing pixel-based change detection. The proposed method is expected to be very helpful in construction management by aiding in the development of smart construction technology.

## 3 Materials and Methods

The material for this project consist in three dataset for training, testing and validation. Dataset contain square patches of a yard's fence with 133x133 dimension. The dataset it's been uploaded, data augmentation was applied and two label, 0 and 1, has been attached to the patches. 0 if the fence was not present, 1 if yes. So the dataset consist in 2 class. Subsequently we used a VGG16 CNN's to train, test and validate the model. Description of the instruments used, the dataset contents and the preprocessing procedures applied is provided.

### 3.1 Dataset

We have a total of 3 dataset, 1 for each drone flying. Each dataset contain 2 sets of photos, with yard's fence (label 1), and without yard's fence (label 0). the first dataset, refering to the yard1, contain 745 photos, 311 with label 0 and 434 with label 1. the second dataset, refering to the yard1 as before, but in different weather conditions, contain 842 photos, 448 with label 0 and 394 with label 1. the third dataset,refering to a new yard, yard2, contain 333 photos, 81 with label 0 and 252 with label 1, there are less photos, but used only for testing. in order to balance the dataset, we made sure that label 0 and label 1 had the same number of images, 477 each, for a total of 954 elements before data augmentation. Data augmentation are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model. It is closely related to oversampling in data analysis. Horizontal and vertical flip has been applied to the training set, after that, 2862 elements

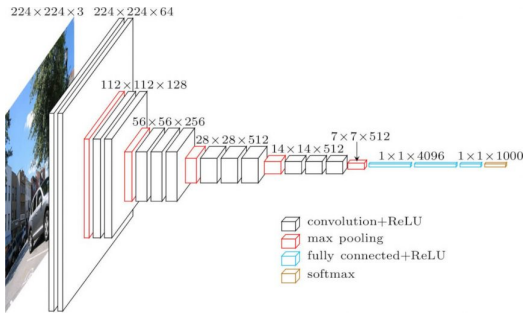



Figure 1: VGG-16 architecture

of 133x133 pixel was the new dimension of it. it is needed to be specified that the dataset used for train and validation, consist in two set of photos, took in the same yard, but in different weather conditions, which is crucial for our pourpose because changes in intensity and other factors, for example, the different type of field, could reduce the prediction accuracy during the prediction phase.

### 3.2 VGG16 CNN's

VGG16 [5] is a convolutional neural network model. The input to convolutional 1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of convolutional layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 convolutional layers. Spatial pooling is carried out by five max-pooling [6] layers, which follow some of the convolutional layers (not all the convolutional layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2. Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer, Softmax [7] assigns decimal probabilities to each class in a multi-class problem. Those decimal probabilities must add up to 1.0. This additional constraint helps training converge more quickly than it otherwise would. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU) non-linearity.

## 4 Experimental Procedure

For this task, we used a VGG16 CNN's to predict if, given a dataset composed by square patches, an orange fence was present or not in the image. If the fence is in the image, the classification label is settled to 1, if not, the classification label is settled to 0. Fine-tuning is simple method.

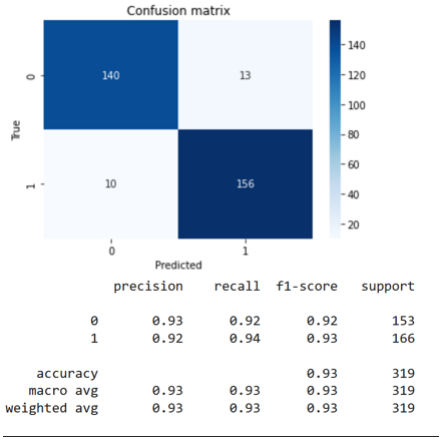
```
input_1 (InputLayer)         [(None, 133, 133, 3)]
block1_conv1 (Conv2D)        (None, 133, 133, 64)
block1_conv2 (Conv2D)        (None, 133, 133, 64)
block1_pool (MaxPooling2D)   (None, 66, 66, 64)
block2_conv1 (Conv2D)        (None, 66, 66, 128)
block2_conv2 (Conv2D)        (None, 66, 66, 128)
block2_pool (MaxPooling2D)   (None, 33, 33, 128)
block3_conv1 (Conv2D)        (None, 33, 33, 256)
block3_conv2 (Conv2D)        (None, 33, 33, 256)
block3_conv3 (Conv2D)        (None, 33, 33, 256)
block3_pool (MaxPooling2D)   (None, 16, 16, 256)
block4_conv1 (Conv2D)        (None, 16, 16, 512)
block4_conv2 (Conv2D)        (None, 16, 16, 512)
block4_conv3 (Conv2D)        (None, 16, 16, 512)
block4_pool (MaxPooling2D)   (None, 8, 8, 512)
block5_conv1 (Conv2D)        (None, 8, 8, 512)
block5_conv2 (Conv2D)        (None, 8, 8, 512)
block5_conv3 (Conv2D)        (None, 8, 8, 512)
block5_pool (MaxPooling2D)   (None, 4, 4, 512)
```

Figure 2: fine tuning layers

It uses already trained network and re-trains the part of that by the new dataset to adjust the weights in order to improve the classification performance. In this case, fine tuning is made by freezing [8] all VGG16 layers except the last 4 CONV layers, Freeze means that CNN's layers are not trainable because does not change. The network hyperparameters are the same for all the performed test: number of epochs = 30, no batch size settled, learning rate = 0.0001, momentum=0.9. We also implemented a callback function called early stopping, which stop the training if accuracy does not improve for 5 consecutive epochs. Another implemented function is the model checkpoint, which save the model with the best accuracy instead of the last model trained. It's important because sometimes, despite the early stopping, the last model trained is not necessarily the best to use. All the procedure is done on Google COLAB.

### 4.1 Fine-Tuning Pre-trained Model VGG-16 approach

Transfer learning for VGG16 starts with freezing [9] the model weights that had been obtained by training the model on a big dataset previously mentioned. These learned weights and filters provide the network with great feature extraction capabilities, which will help us boost its performance when it is trained to classify fence presence. Hence, only the Fully Connected layers will be trained while keeping the feature extraction part of the model almost frozen. If the network performed well during training, it should also perform well when tested on yard's images that it had not seen before, We will test it on our testing set of images.




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.93 | 0.92 | 0.92 | 153 |
| 1 | 0.92 | 0.94 | 0.93 | 166 |
| accuracy | | | 0.93 | 319 |
| macro avg | 0.93 | 0.93 | 0.93 | 319 |
| weighted avg | 0.93 | 0.93 | 0.93 | 319 |

Figure 3: 60-20-20 test confusion matrix

### 4.2 Training setting

Related to the dataset split, we opted for different settings, in order to obtain as many result as possible. Firt test was performed with a 60-20-20 split combining yard1 and yard1 with different weather, 60 percent of the dataset of for the training, 20 for the validation and 20 for the test. In the second test we choose a 80-20 split, 80 percent of the dataset of the first yard for training, 20 for validation and another whole dataset of photos, the third fly, refering to a new yard's fence, used for testing.

### 4.3 Testing

to test the model we used a new whole dataset of a completely different yard.

#### *4.3.1 Test number 1*

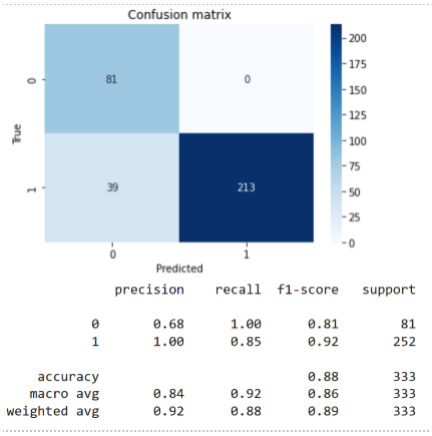
This test was performed with a 60-20-20 split combining yard1 and yard1 with different weather, 60 percent of the dataset of for the training, 20 for the validation and 20 for the test.The network's hyperparameter are the same described in the section 4.

#### *4.3.2 Test number 2*

In this second test we choose a 80-20 split, 80 percent of the dataset of yard1 plus yard1 in different weather conditions for training and 20 for validation. And another whole dataset of photos, the third fly, refering to yard2, used for testing. The network's hyperparameter are the same described in the section 4.

## 5 Results

Results show that the final test, performed with a 80-20 split, is good. the model stop the train after 8 epochs, with a val loss of 0.1735 considering the confusion matrix, there are 0 false positive and 39 false negative over a 333 images dataset. Considering 39 false negative prediction, is not as bad as we think, because in the worst




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.68 | 1.00 | 0.81 | 81 |
| 1 | 1.00 | 0.85 | 0.92 | 252 |
| accuracy | | | 0.88 | 333 |
| macro avg | 0.84 | 0.92 | 0.86 | 333 |
| weighted avg | 0.92 | 0.88 | 0.89 | 333 |

Figure 4: 80-20 test confusion matrix

```
Predicted : 0
Correct label : 1
Percentage 0: 99.992%
Percentage 1: 0.008%
```

Figure 5: example of false positive test

case, we should check for an higher number of patch in the yard. the important thing is that no false positive prediction is done. So in term of security there's not an high impact. The future improvement of this network should be to avoid the false negative prediction also. Obviously, the 60 20 20 test, utilizing the same dataset is slightly accurate.

The limits of this study are related to the low number of drone fly done (3 only), so we don't have enough data to test the model in different ways. We only have access to 2 yards which are not enough to train the model properly.

For future development, the monitoring of security fences in drone orthophotos could be tested with a segmentation [10] procedure.

## References


[1] Wikipedia, *Ortofoto*, wikipedia.org (2020), https://it.wikipedia.org/wiki/Ortofotografia

[2] Wikipedia, *Qgis*, wikipedia.org (2020), https://it.wikipedia.org/wiki/QGIS

[3] A. Vargheseand, *Changenet: A deep learning architecture for visual change detection*, ECCVV (2018), https://openaccess.thecvf.com/content_eccv_2018_workshops/w7/html/Varghese_ChangeNet_A_Deep_Learning_Architecture_for_Visual_Change_Detection_ECCVW_2018_paper.html

[4] M. Song, *Change detection in unmanned aerial vehicle images for progress monitoring of road construction*, mdpi (2021), https://www.mdpi.com/2075-5309/11/4/150

[5] Keras, *vgg16 and vgg19*, keras.io (2018), https://keras.io/api/applications/vgg/

[6] Wikipedia, *Max pooling*, wikipedia.org (2020), https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

[7] Wikipedia, *Funzione softmax*, wikipedia.org (2020), https://it.wikipedia.org/wiki/Funzione_softmax

[8] Roshan, *Transfer learning and fine tunign model using vgg16*, medium.com (2020), https://medium.com/@roshankg96/transfer-learning-and-fine-tuning-model-using-vgg-

[9] R.K. Gupta, *Transfer learning and fine tuning model using vgg 16*, Medium.com (2020), https://medium.com/@roshankg96/transfer-learning-and-fine-tuning-model-using-vgg-

[10] I. Das, *segmentation*, arxiv.org (2019), https://arxiv.org/pdf/1907.06119.pdf