Damage Detection using Neural Networks

-Sumedha Khatter
@Airware

Agenda

Explaining Neural Networks

Neural Network

1. It is a computing system inspired by the biological neural networks.
2. A neural network is based on the collection of connected units or nodes called artificial neurons and each connection between artificial neurons can transmit a signal from one to another.
3. The output is calculated by a non-linear function of the sum of its inputs.
4. An example neural network would compute s = W2 max(0, W1*x), where W1 and W2 are weight matrices and are learned via stochastic gradient descent.

Layers of a neural network

The layers in a neural network is organized as-

1. Input Layer
2. Hidden Layers
3. Output Layer -

Types of hidden layers in a network:

1. Densely connected layer- learn global patterns in their input feature space.
2. Convolutional layer- learn local features
3. Pooling- Reduces the spatial size of the representation to reduce the parameters and hence control overfitting.
4. Dropout- a regularization technique that randomly drops out a number of output features.

Activation Functions

1. Linear Function- f(x) = x
2. Sigmoid Function- It exists between 0 and 1. This function can cause a neural network to stuck during training due to the fact that is a strongly- negative input is provided to the logistic sigmoid, the output is very close to zero. And then it can cause the model parameters to get updated less regularly.
3. Tanh- Provides a range between -1 and 1. .
4. ReLu- outputs values from 0 to infinity.
5. Softmax- Provides probabilities between 0 and 1 . Most commonly used activation function for multiclass classification.

```
model = models.Sequential()
model.add(layers.Conv2D(100, (3,3), activation ='relu', input_shape = (512, 512, 3)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(200, (3,3), activation ='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(200, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation = 'relu'))
model.add(layers.Dense(1, activation = 'sigmoid'))
model.summary()
```

Layer 1-
Input Shape- 512 * 512 * 3
Output Shape- ((512 -3)/1 +1) * 510 *100
Number of params- 100 * (3*3*3 + 1) = 2800
Layer 2-
Input Size- 510 * 510 * 100
Output Size- 255 * 255 * 100
Layer 3-
Input Size- 255 * 255 * 100
Output Size- 253 * 253 * 200
Number of Params- 200 * (3*3*100 + 1) = 200 * (901)= 180200
Layer 4-
Input Size- 253 * 253 * 200
Output Size- 126 * 126 * 200

Layer 5-
Input Size- 126 * 126 * 200
Output Size- 124 * 124 * 200
Number of params- 200 * (3*3*200 + 1)= 200 * (1801)= 360200
Layer 6-
Input Size- 124 * 124 * 200
Output Size- 62 * 62 * 200
Layer 7-
Output- 62 * 62 * 200 = 768800
Layer 8-
Number of params- 512 * (76800 + 1) = 39322112
Output Size- 512
Layer 9-
Number of params- 1 * (512 + 1)= 513

Binary Classifier: Wind Damage or no-Wind Damage

Using simple Sequential Model

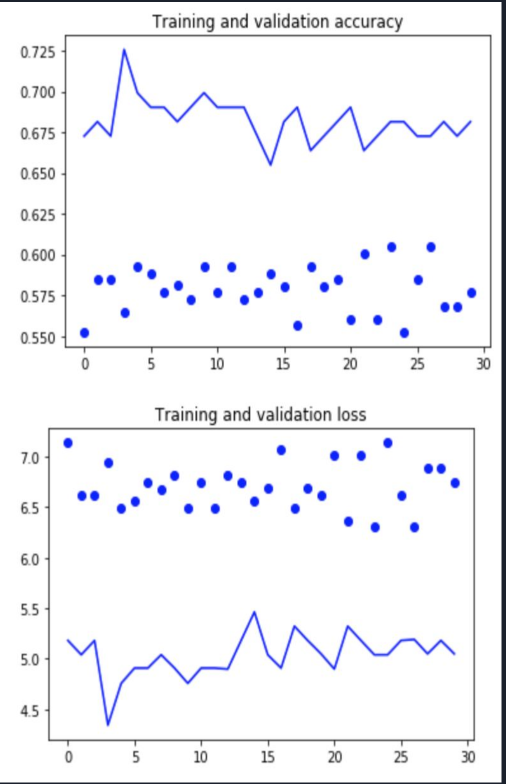
Model Summary

- The Sequential model consisted of 3 convolutional layers and two dense layers.

- Training Set Size- 243, Validation set size- 31

Results

- There is a huge difference between training and validation accuracy.



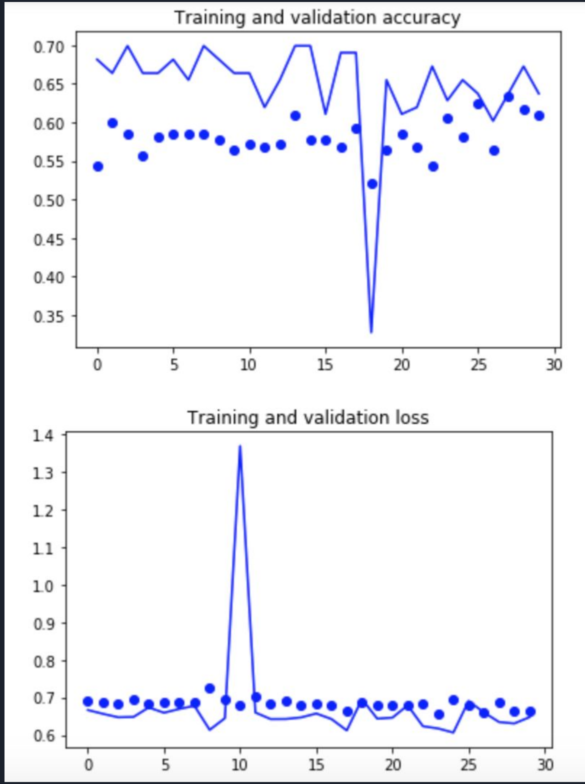
Using the Sequential Model with Data Augmentation

Model Summary

- The Sequential model consisted of 3 convolutional layers and two dense layers.
- Training Set Size- 243, Validation set size- 31

Results

- There is a huge difference between training and validation accuracy.



Using VGG16, without any fine tuning

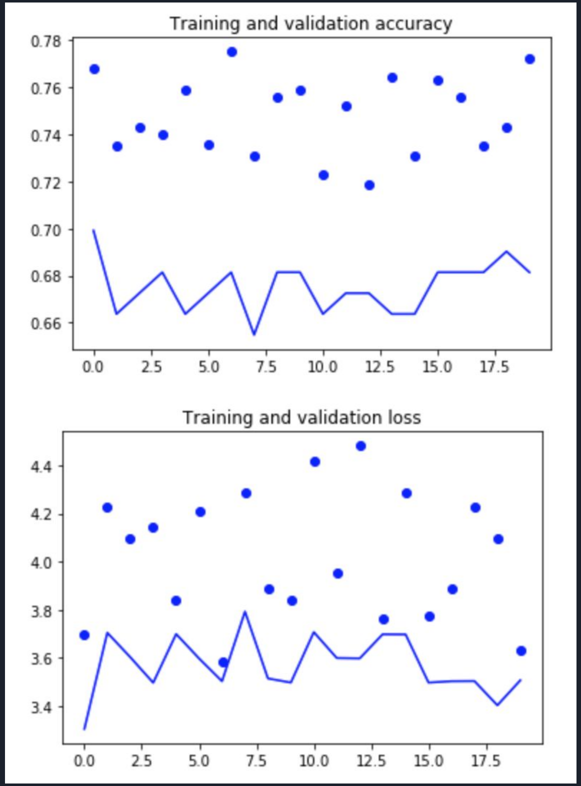
Model Summary

- The model was built using the convolutional base of VGG16 and then 2 dense layers on the top of it. The convolutional base was non-trainable.

- Training Set Size- 243, Validation set size- 31

Results

- There is a huge difference between training and validation accuracy and noise in the accuracy curves.



Using VGG16 with fine tuning of last conv_block

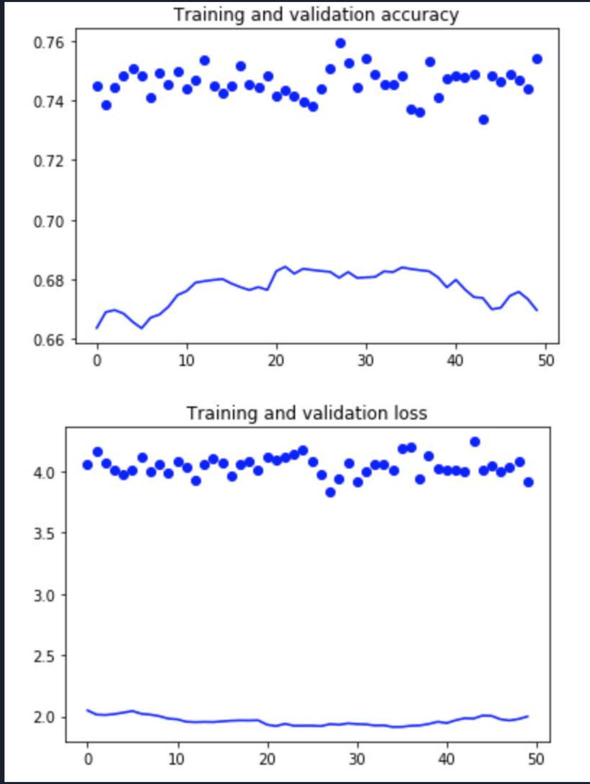
Model Summary

- The model consisted of convolutional base made from the VGG16 and then 2 dense layers on top of it. The last block of the Convolutional base was made trainable.

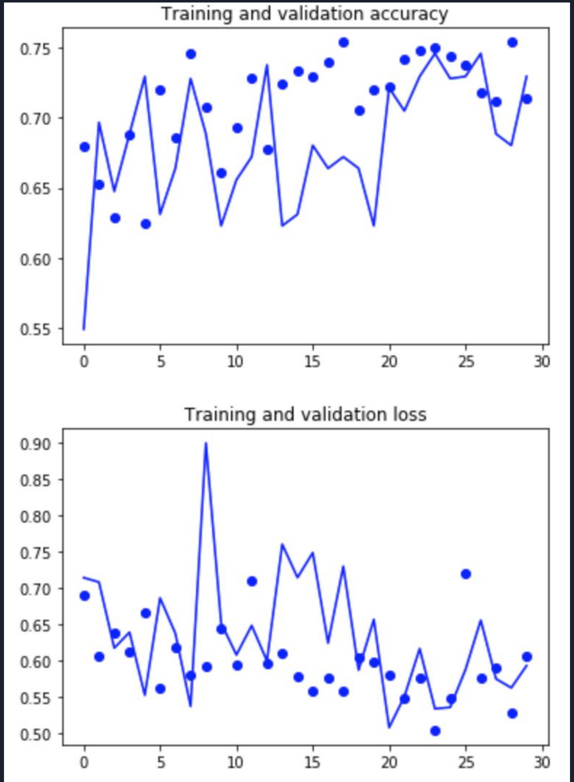
- Training Set Size- 243, Validation set size- 31

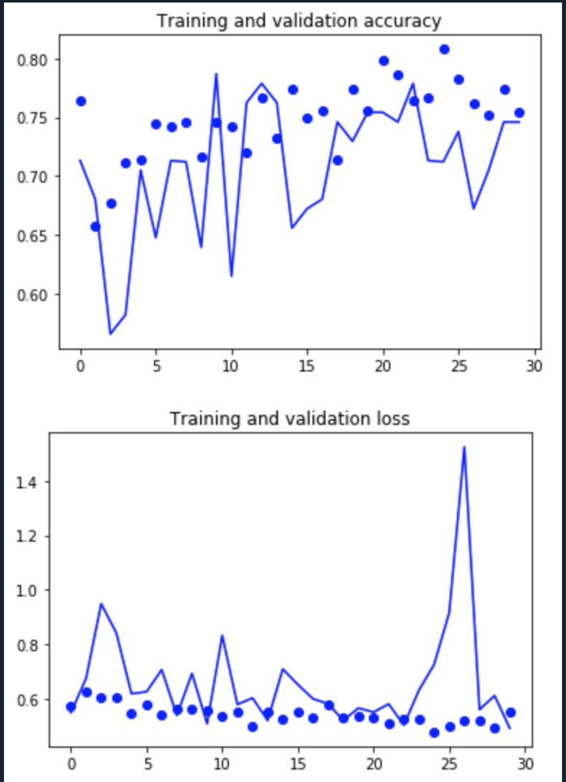
Results

- There is a huge difference between training and validation accuracy, but there is very less noise.



Data Augmentation: zoom_range 0.8 and 0.3





Results-

1. The same experiments were performed with a bigger dataset.
2. Data Augmentation changes the accuracy numbers.
3. There is an increase in accuracy with a moderate range of the augmentation parameters. Eg- zoom_range of 0.2- 0.3 increases the accuracy, a higher range like 0.8-1 gives a little less accuracy and adds noise to the accuracy curves.
4. Making the last block of a pre-trained network increases the accuracy.
5. Accuracy achieved were better and reached around 74%

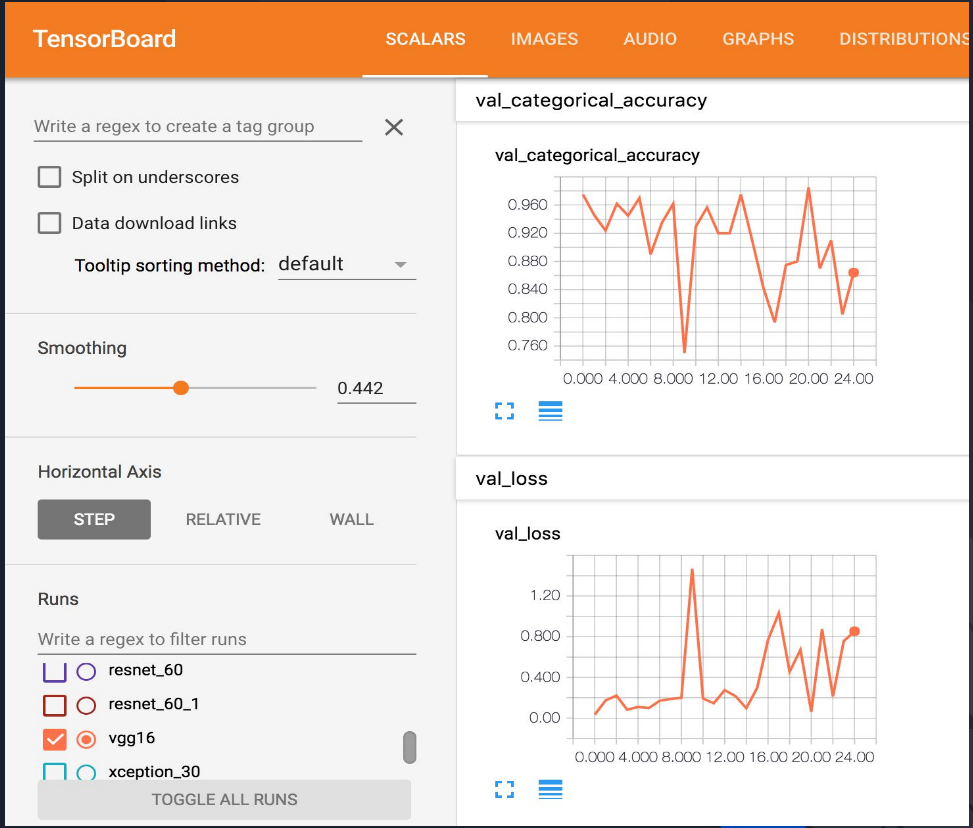
4 class classifier-

1. Wind-damaged roof
2. Roof
3. Not-roof
4. Objects

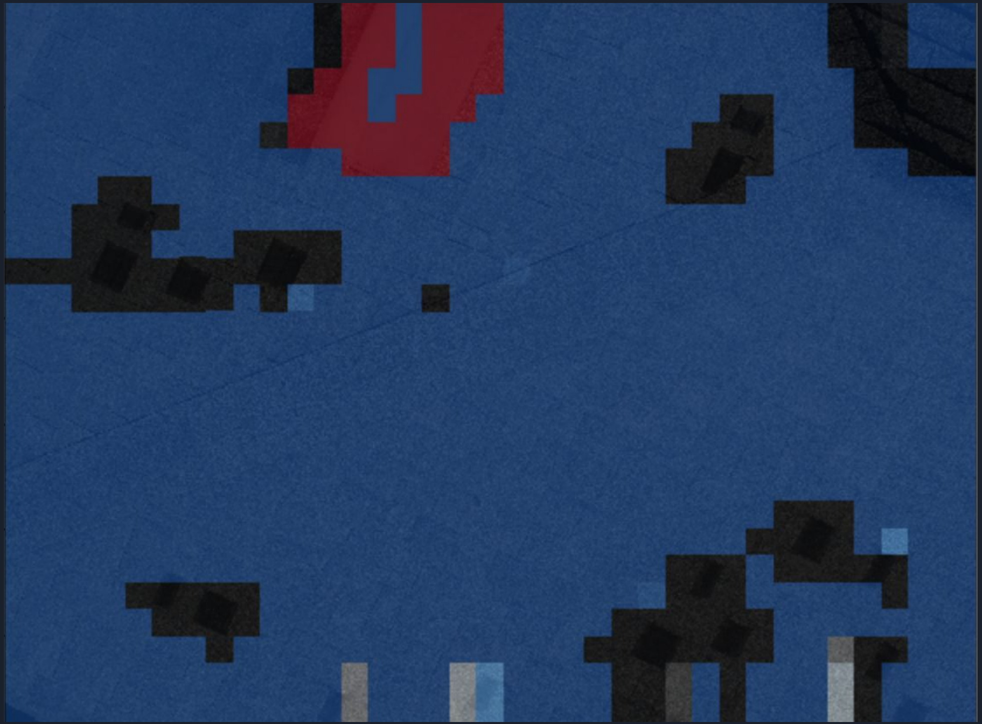
VGG

1. This is the keras model of the 16-layer network used by the VGG team in the ILSVRC-2014
2. This model is a thorough evaluation of networks of increasing depths using an architecture with a very small convolutional filters and achieves a significant numbers in accuracy
3. Number of weight layers are 16 and 19 in VGG16 and VGG19.

Results with VGG16



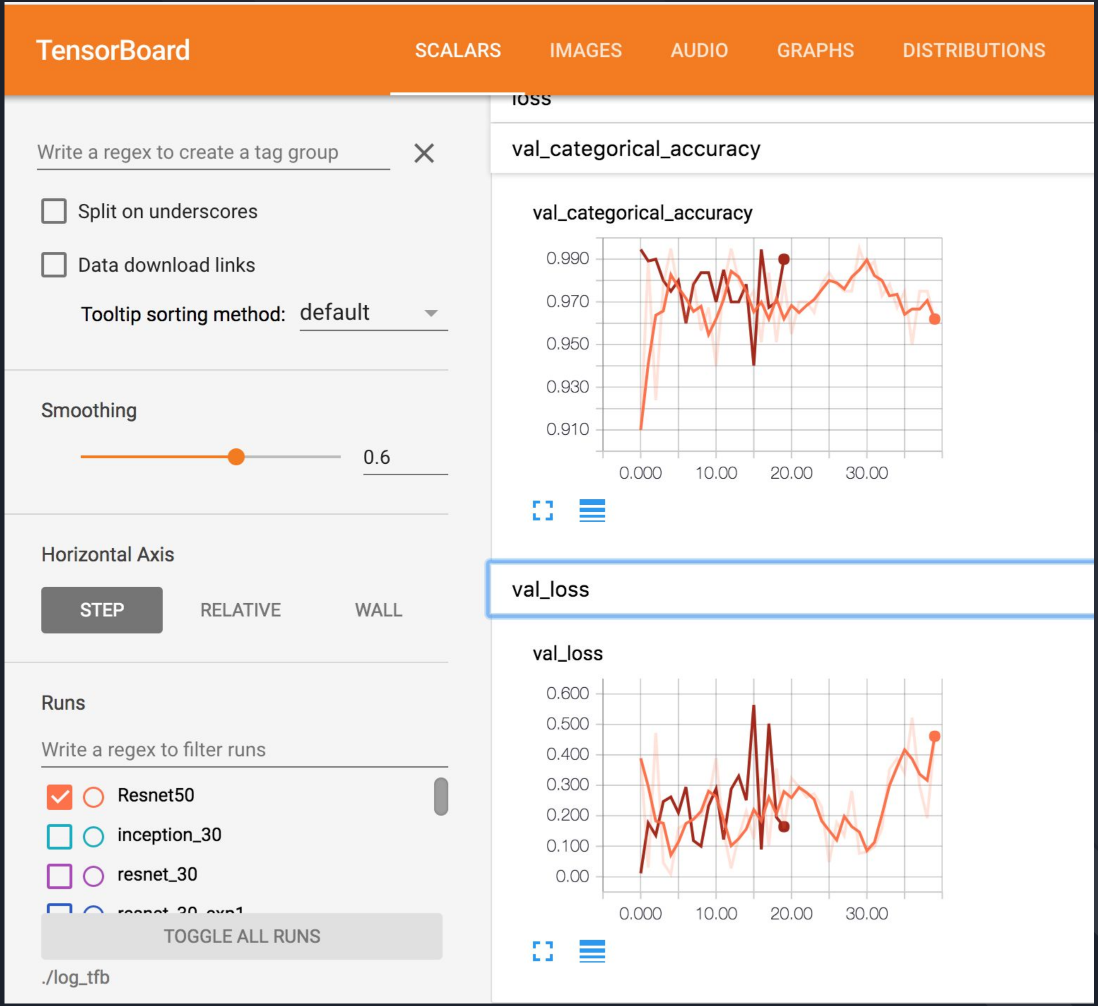
Test results with VGG 16



Resnet

1. A layer in the traditional neural network learns to calculate a function, y = f(x)
2. A residual network layer approximately calculates y = f(x) + id(x) = f(x) + x
3. The resnet used in my experiments consist of 50 layers. The model is called Resnet50.
4. The gradient signal vanishes with increasing network depth, but the identity connections in ResNets propagate throughout the model.

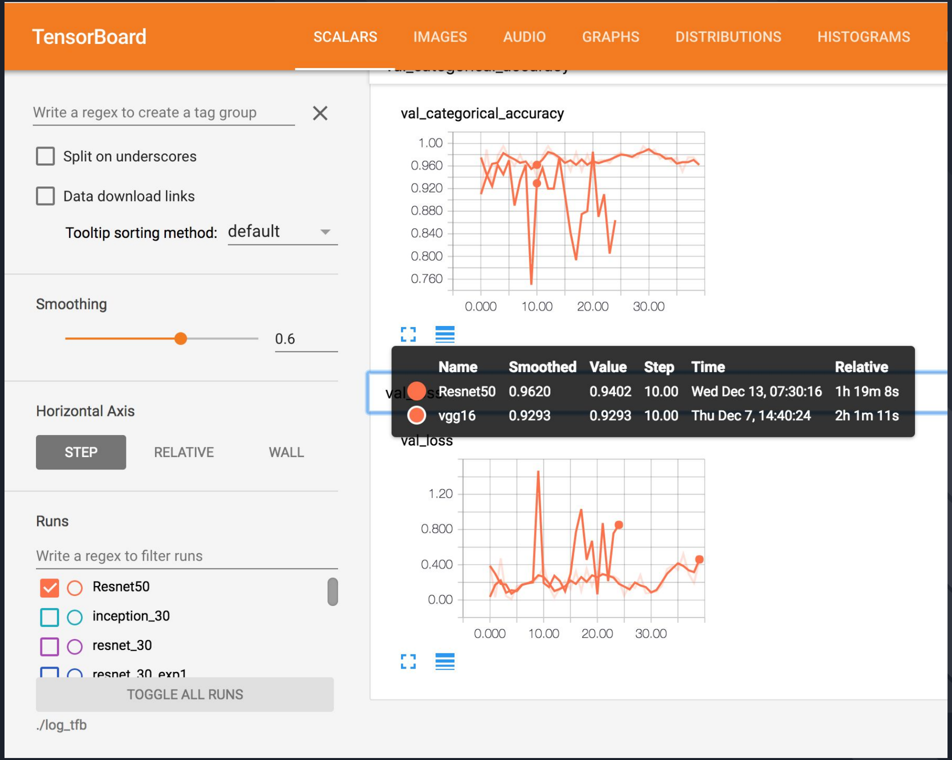
Results with ResNet50



Test results with Resnet.

Comparing results of Resnet and VGG16

1. On comparing minutely, it can be found that ResNet50 performs better than VGG16
2. There is less noise in the curves using ResNet60
3. Test image with VGG16 detects a non-damaged area as a damaged area.



Post- processing of the model tested image.

1. The image is processed after the damages are detected by the model.
2. Isolated damages are removed using this algorithm.
3. Algo:

```
For each pixel in the image:
    max_nb = max(neighbors)
    pixel_val = min(max_nb, pixel_val)
```

Post processing test results

Conclusions

1. Transfer learning with pre-trained networks are better when last block of the model is made trainable.
2. The experiments for the wind damage were performed with 4 classes and various results on the test images have showed overlaps of: not-roof and objects. May be, we can try a classifier with just 3 classes- wind-damaged roof, roof and not-roof.
3. We can try models like ResNet152 and Xception for classifiers to capture more complex patterns.

References

1. CNN for visual Recognition (http://cs231n.github.io/)
2. Wikipedia (https://en.wikipedia.org/wiki/Artificial_neural_network)
3. Keras for Sequential Models (https://keras.io/getting-started/sequential-model-guide/)
4. pyimageSearch- with Imagenets (https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/)
5. Deep Learning with Python (MEAP) by Francois Challote
6. Python Documentation (https://docs.python.org/3/)
7. Tensorflow(https://www.tensorflow.org/get_started/summaries_and_tensorboard)
8. Siraj Raval you tube videos (https://www.youtube.com/channel/UCWN3xxRkmTPmbKwht9FuE5A)

Thanks a lot for patient hearing!