Flood Damage Detection Using Convolutional Neural Network

Abdulaziz Gebril

April 27, 2021

Machine Learning II Final Project

Introduction

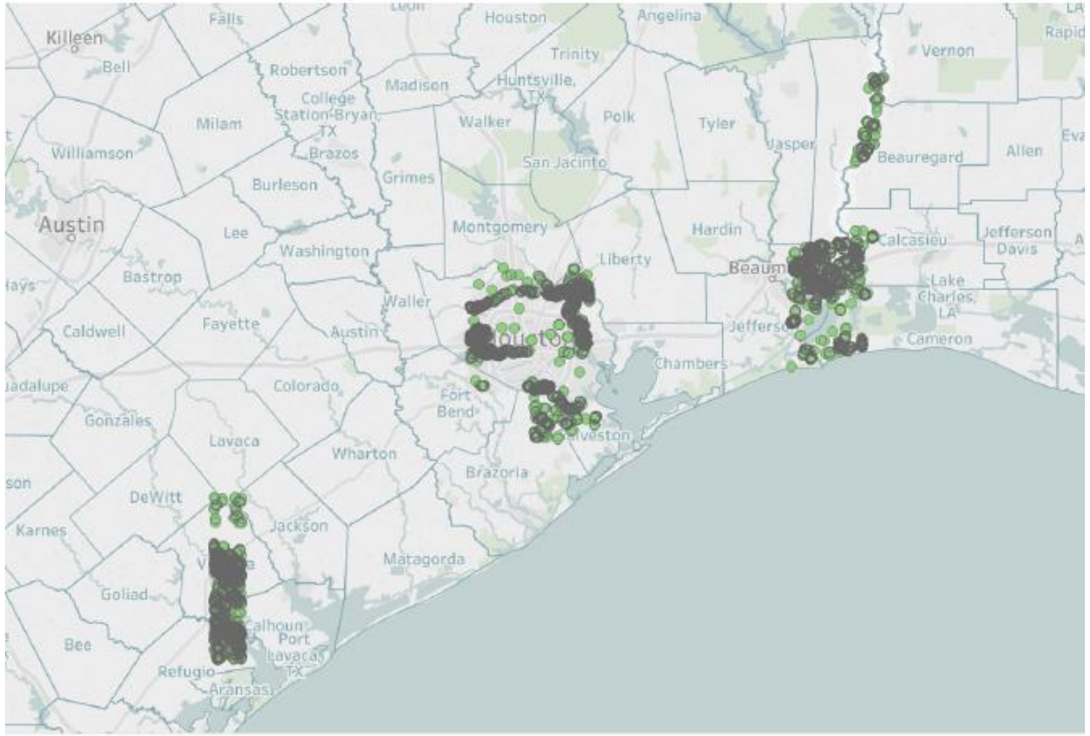
After a hurricane, damage assessment is critical to emergency managers and first responders to plan and allocate resources appropriately.

Main aim is to automate the process of quantifying damaged buildings

Utilizing different state-of-art Convolutional Neural Network

Dataset

- Satellite imagery of the Greater Huston area before and after Hurricane Harvey in 2017 obtained from Kaagle
- The flooded/damaged buildings were labeled by volunteers through crowd-sourcing project.



Dataset

- The dataset consists of 23,000 3-Pixel RGB satellite images labeled into two classes, either “Damaged” or “No Damage”.

Set	Damaged	No Damage
Training set	5,000	5,000
Validation	1,000	1,000
Unbalanced test set	8,000	1,000
Balanced test set	1,000	1,000

Damaged/Flooded Buildings

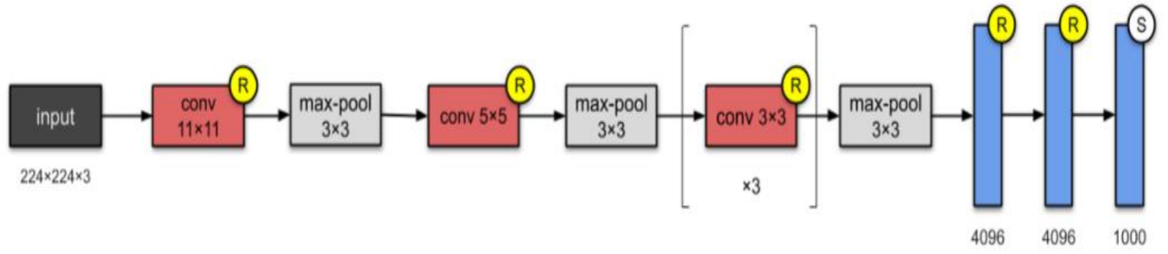
Undamaged Buildings

Pretrained models

- AlexNet
- VGG-16
- ResNet50

AlexNet

- Input size: 224
- Loss Function: Cross Entropy Loss
- Optimizer: SGD
- Batch Size: 300
- Mini-Batch Size: 3
- Learning Rate: 0.00002

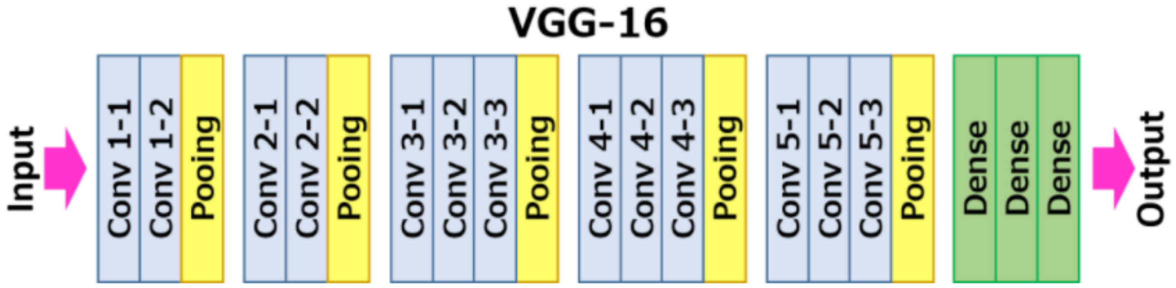



```
for param in model.parameters():
    param.requires_grad = False
n_inputs = model.classifier[6].in_features
n_classes = 2

# Add on classifier
model.classifier[6] = nn.Sequential(
    nn.Linear(n_inputs, 256), nn.ReLU(), nn.Dropout(0.2),
    nn.Linear(256, n_classes))
```

VGG-16

- Input size: 224
- Loss Function: Cross Entropy Loss
- Optimizer: SGD
- Batch Size: 300
- Mini-Batch Size: 5
- Learning Rate: 0.00001

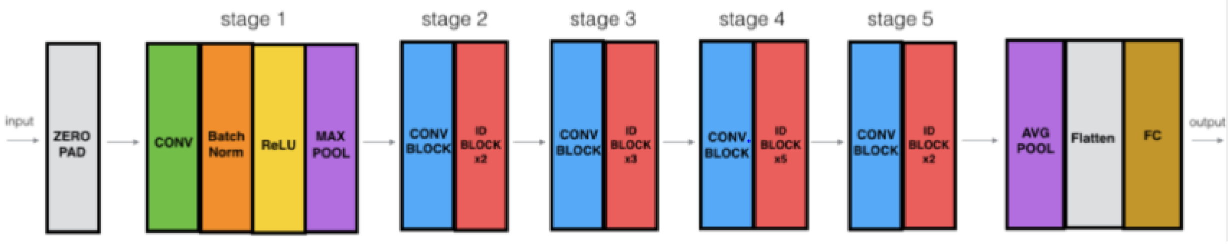



```
for param in model.parameters():
    param.requires_grad = False
n_inputs = model.classifier[6].in_features
n_classes = 2

# Add on classifier
model.classifier[6] = nn.Sequential(
    nn.Linear(n_inputs, 256), nn.ReLU(), nn.Dropout(0.2),
    nn.Linear(256, n_classes))
```

ResNet50

- Input size: 224
- Loss Function: Cross Entropy Loss
- Optimizer: SGD
- Batch Size: 300
- Mini-Batch Size: 10
- Learning Rate: 0.0002




```python
# Freeze early layers
for param in model.parameters():
    param.requires_grad = False

n_inputs = model.fc.in_features
n_classes = 2

# Add on classifier
model.fc = nn.Sequential(
    nn.Linear(n_inputs, 256), nn.ReLU(), nn.Dropout(0.25),
    nn.Linear(256, n_classes))
```

Custom Model

- Input size: 128
- Loss Function: Cross Entropy Loss
- Optimizer: SGD
- Batch Size: 300
- Mini-Batch Size: 10
- Learning Rate: 0.01

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size = 5, stride = 1, padding = 1)
        self.convnorm1 = nn.BatchNorm2d(32)
        self.pool1 = nn.MaxPool2d(2, 2)

        self.conv2 = nn.Conv2d(32, 64, kernel_size=2, stride=1, padding=1)
        self.convnorm2 = nn.BatchNorm2d(64)
        self.pool2 = nn.MaxPool2d((2, 2))

        self.conv3 = nn.Conv2d(64, 64, kernel_size = 3, stride = 1, padding = 1)
        self.convnorm3 = nn.BatchNorm2d(64)
        self.pool3 = nn.AvgPool2d((2, 2))

        self.dropout = nn.Dropout(DROPOUT)
        self.linear1 = nn.Linear(64 * 16 * 16, 32)
        self.linear1_bn = nn.BatchNorm1d(32)
        self.linear2 = nn.Linear(32, 2)
        self.linear2_bn = nn.BatchNorm1d(2)
        self.sigmoid = torch.sigmoid
        self.relu = torch.relu

    def forward(self, x):
        x = self.pool1(self.convnorm1(self.relu(self.conv1(x))))
        x = self.pool2(self.convnorm2(self.relu(self.conv2(x))))
        x = self.pool3(self.convnorm3(self.relu(self.conv3(x))))
        # print(x.shape)
        x = self.dropout(self.linear1_bn(self.relu(self.linear1(x.view(-1, 64 * 16 * 16)))))
        x = self.dropout(self.linear2_bn(self.relu(self.linear2(x))))
        x = self.sigmoid(x)
        return x
```

UnBalanced Test Results

- Unbalanced Test set consists of 8,000 Damaged labels and 1,000 undamaged labels.

Model	Accuracy	Precision	Recall	F1-score
Gebril model	94.14%	0.957	0.976	0.8608
AlexNet	92.3%	0.926	0.986	0.839
ResNet50	90.74%	0.917	0.977	0.80
VGG-16	82.02%	0.813	0.981	0.70

Balanced Test Results

- Balanced Test set consists of 1,000 Damaged labels and 1,000 undamaged labels.

Model	Accuracy	Precision	Recall	F1-score
Gebril model	88.65%	0.952	0.84	0.886
AlexNet	90.8%	0.915	0.902	0.97
ResNet50	86.9%	0.903	0.84	0.868
VGG-16	83.8%	0.806	0.861	0.837

Conclusion

Pretrained and custom models were trained to address binary classification problem (Detecting flood damage based on satellite images).

Custom model (Gebril) yielded highest accuracy on unbalanced test data, While AlexNet yielded highest overall accuracy.

Enhance model by better Hyperparameter tuning and adding more regularization techniques.

Widen model usage to include more areas and more infrastructure objects such as road and bridges.