HIdentifier: A Method in Agriculture CPS Framework to Automatically Identify Disease Hotspots Using Message Passing in Graph

Presenter: Kiran Kumar Kethineni

Kiran Kumar Kethineni[1], Saraju P. Mohanty[2], E. Kougianos[3]

University of North Texas, USA.[1,2,3], USA

Email: **Kethineni.kirankumar@unt.edu[1], saraju.mohanty@unt.edu[2], elias.kougianos@unt.edu[3]**

Why This Matters

- Agriculture is the foundation of the food system.
- Agriculture is a major contributor to the global economy.
- The global human population is projected to reach 9.7 billion by 2050 and 10.9 billion by 2100.
- Ensured Food security and food safety.

Source of Food

Source of Livelihood

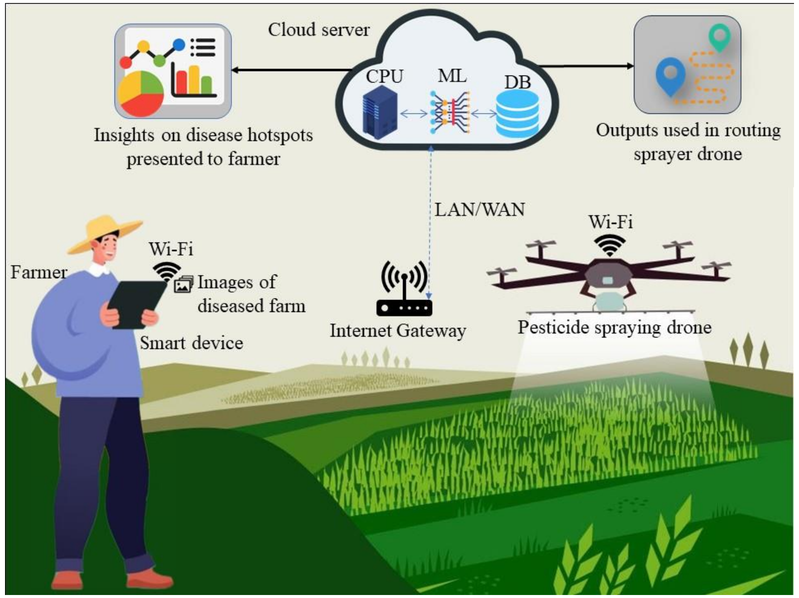
Source of Employment

Source of Income

Contribution to Trade

Why This Matters

- Impact on crop yield.
- Economic consequences.
- Food security concerns.
- Environmental impact.



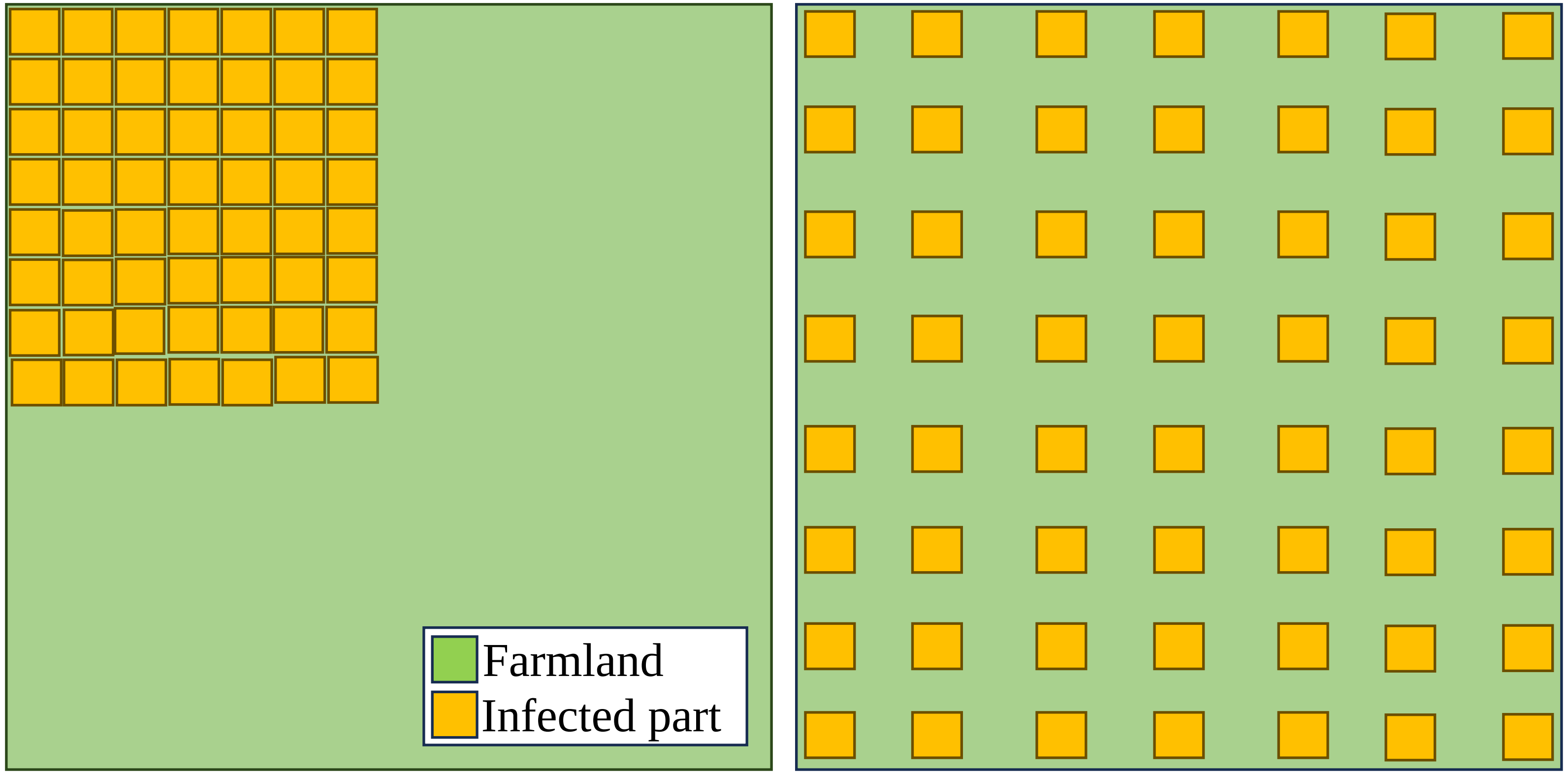
- Need for early detection and management ACPS.
- Hotspot detection for better disease management.

Related Works

Work	Factors considered	Remark
Mitra et al.	Proportion of area damaged by climate.	Does not consider spatial spread.
Parikh et al.	Percentage of area affected by disease.	Does not consider spatial spread.
Jamadar et al.	Area occupied by lesion due to disease.	Does not consider spatial spread.
Ratnasari et al.	Area covered by spots due to diseased.	Does not consider spatial spread.
Divyanth et al.	Percentage of area occupied by diseased segments.	Does not consider spatial spread.
Stimator	**Percentage of area affected by disease and its spatial spread.**	**Considers spatial spread for effective estimation.**

Problem Statement

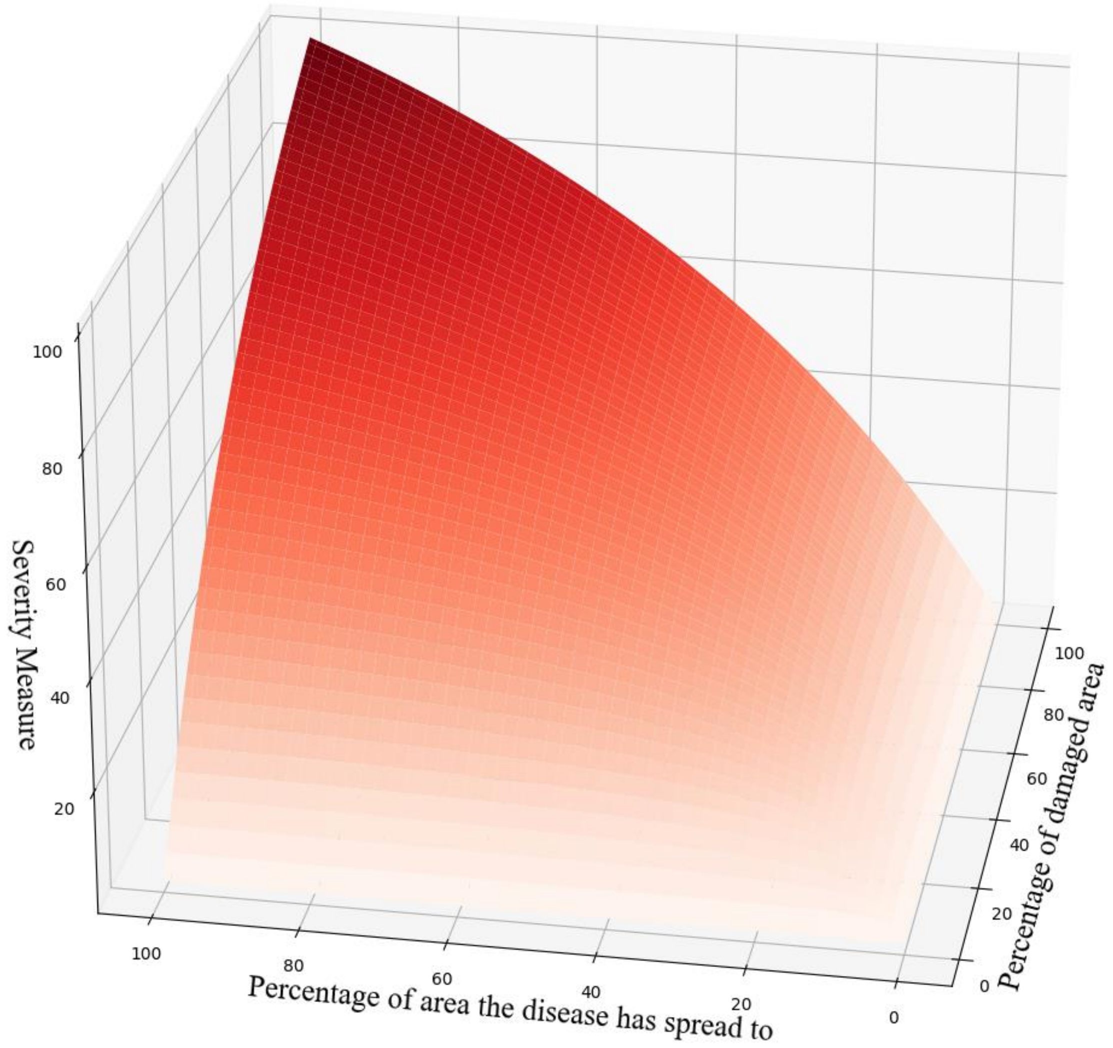
- Spatial spread must be included in estimating the severity of the disease.



(a) Disease confined to a corner. (b) Disease spread throughout farmland.

Proposed Solution

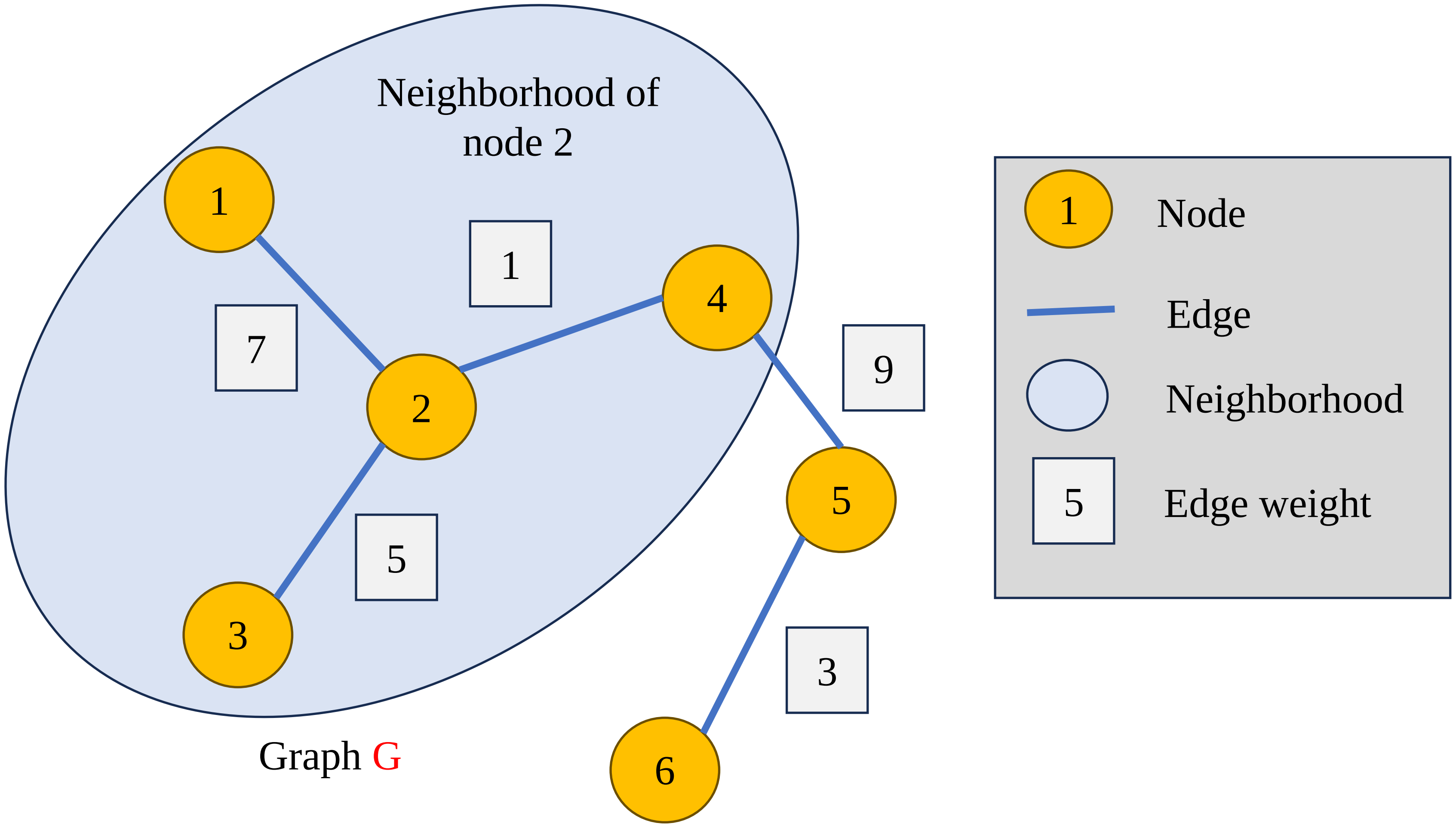
- Harmonic mean of percentage of damaged area and percentage of area the disease has spread.



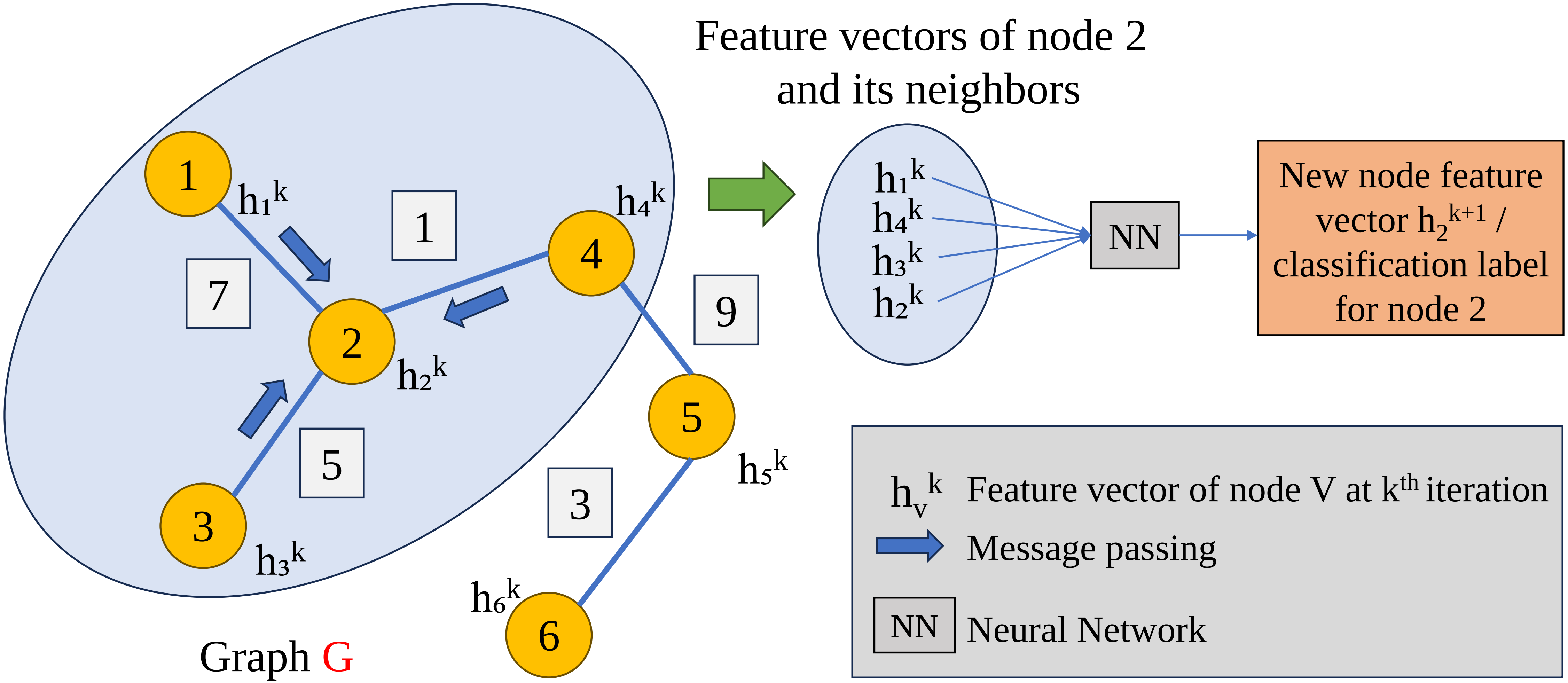
Novel Contributions

- ✓ Stimator considers the area affected by disease and its spatial spread to effectively estimate is severity.
- ✓ The Proposed method represents data of the diseased locations as a graph to capture their spatial relationship.
- ✓ The severity measure proposed is robust and is not affected by single extreme values.
- ✓ It introduces a GNN based method to measure the percentage of the area the disease has spread to for estimating the severity.

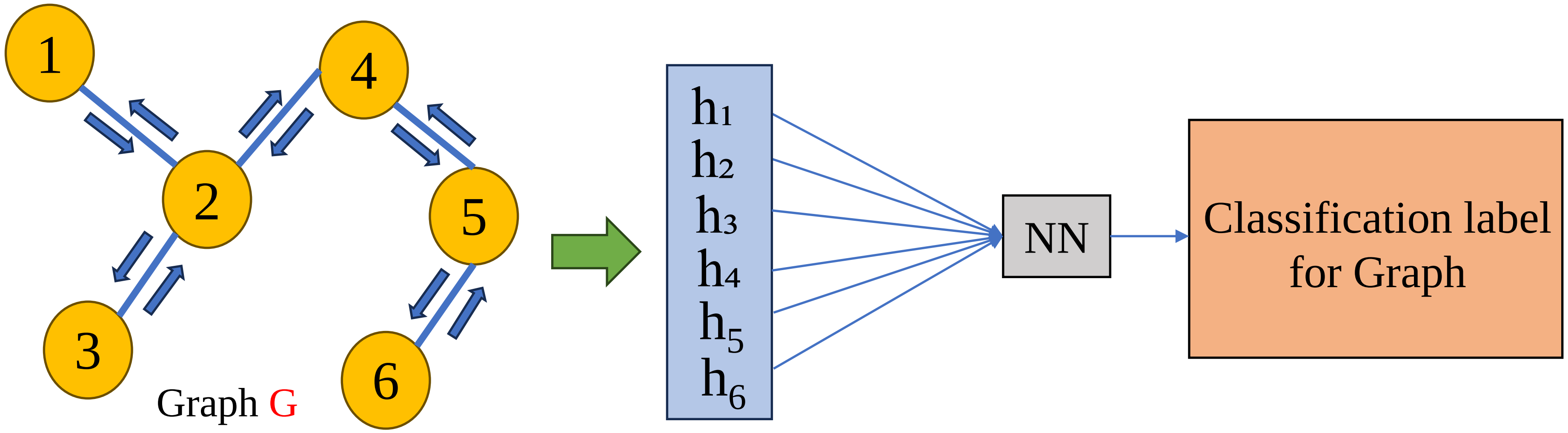
Introduction to Graph



Introduction to Graph Neural Networks



Introduction to Graph Classification

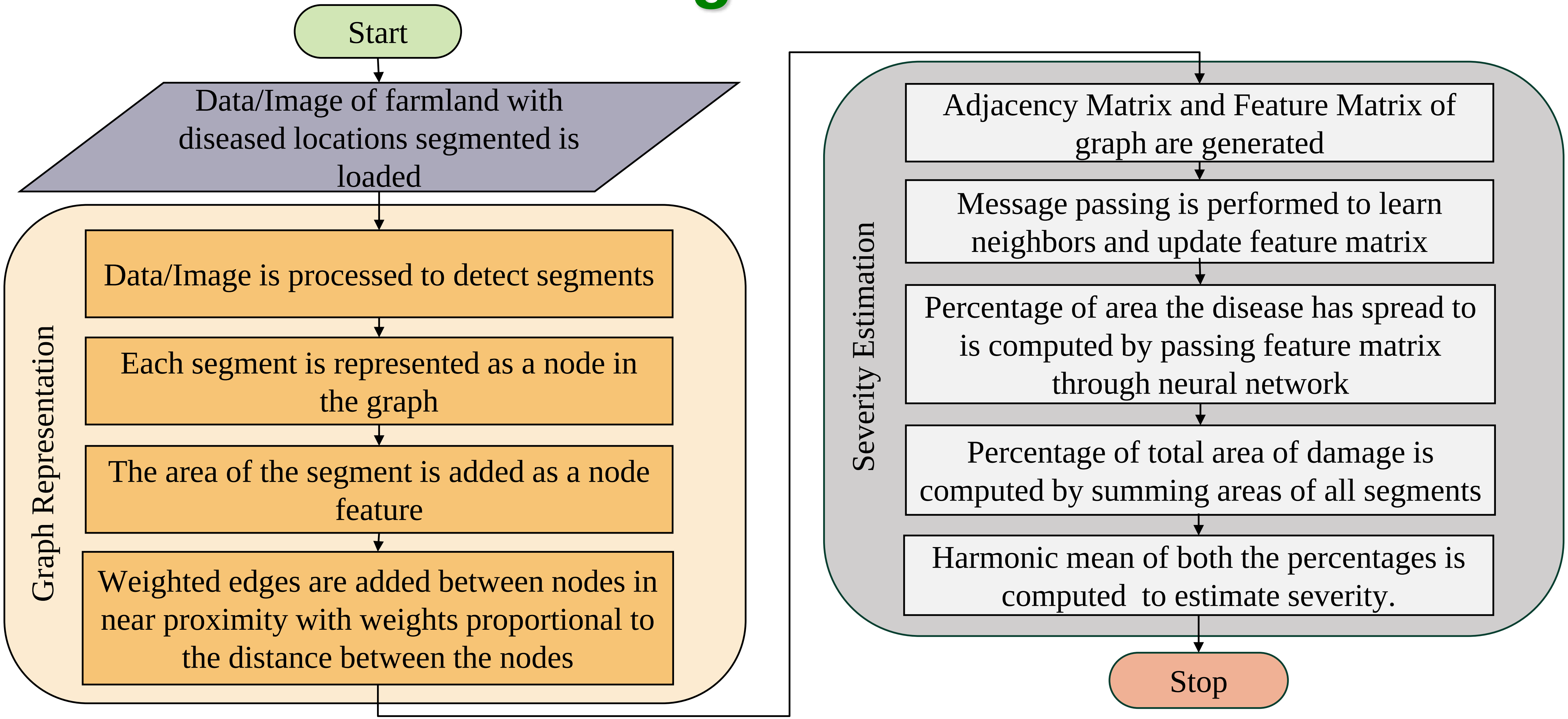


Message passing performed to update node features

Updated node features

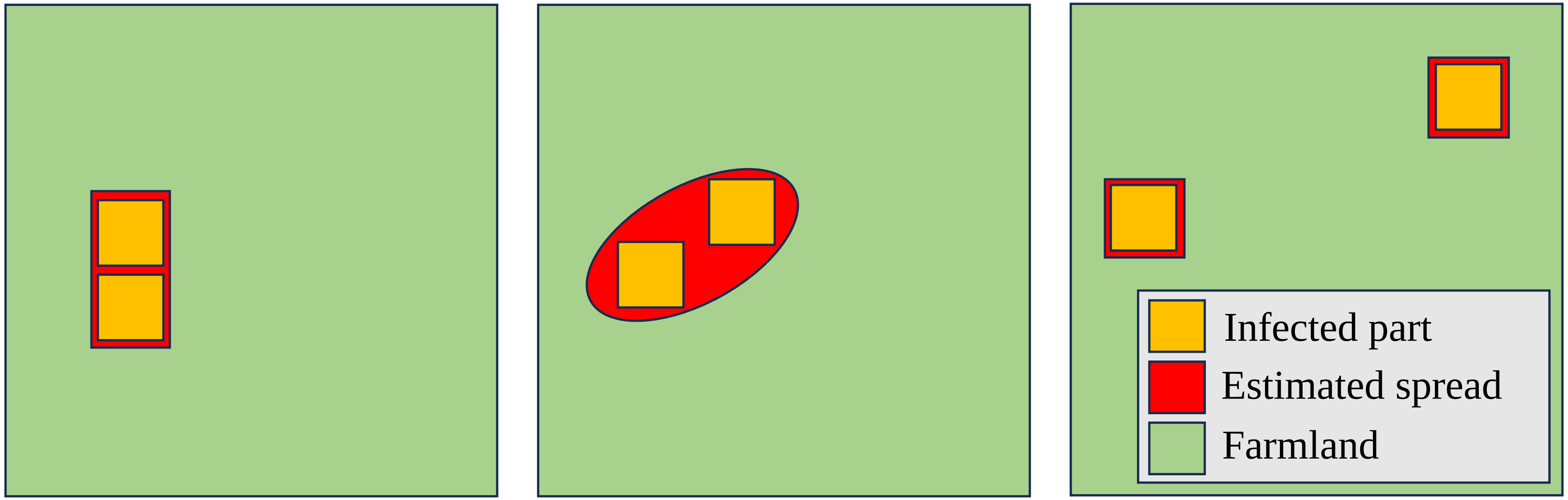
Graph classified using Neural Network

Working of Stimator



Working of Stimator

- The area of the segment is added as a node feature.
- Weighted edges are added between nodes in near proximity with weights proportional to the distance between the nodes.
- Spread at node u $=\Sigma(h_u,\Sigma(\{h_v \times W(uv) : v \in N(u)\}))$



(a) Right next to each other (b)Within near proximity (C) Far apart from each other

Implementation and Results

- The GNN solution proposed has been experimentally validated on a data set from Kaggle which contains images of apple leaves with diseased parts of the leaf annotated.
- But in real-time, images of farmland will be used.
- The solution was developed in Python using NetworkX and Keras libraries.

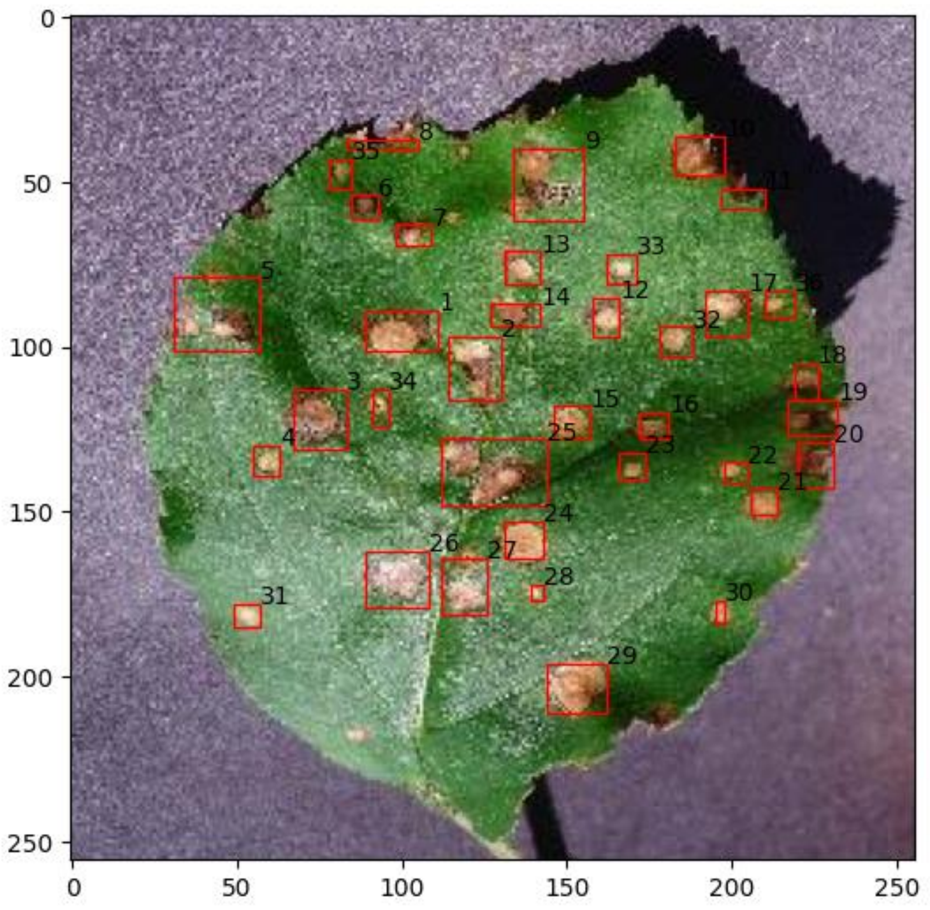
```python
picname = "108.JPG"
G = nx.Graph()
nodeC = 0;
spc = 1
for _,row in train[train.filename == picname].iterrows():
    xmin = row.xmin
    xmax = row.xmax
    ymin = row.ymin
    ymax = row.ymax
    width = xmax - xmin
    height = ymax - ymin
    xpos = (xmax + xmin)/2
    ypos = (ymax + ymin)/2
    area = width * height
    nodeC += 1
    G.add_nodes_from([(nodeC, {"area": area , "x" : xpos , "y" : ypos})])
    for u in G.nodes():
        for v in G.nodes():
            if u != v:
                dist = math.sqrt((G.nodes[u]['x'] - G.nodes[v]['x']) ** 2 + (G.nodes[u]['y'] - G.nodes[
                if(dist < 60):
                    G.add_edge(u, v , weight= (dist/60) )
            else:
                G.add_edge(u, v)
adj_matrix = nx.adjacency_matrix(G)
# Convert the sparse matrix to a dense matrix
adj_matrix_dense = adj_matrix.todense()

# Convert the dense matrix to a NumPy array
adj_matrix_array = np.array(adj_matrix_dense)
```

(a) Graph creation

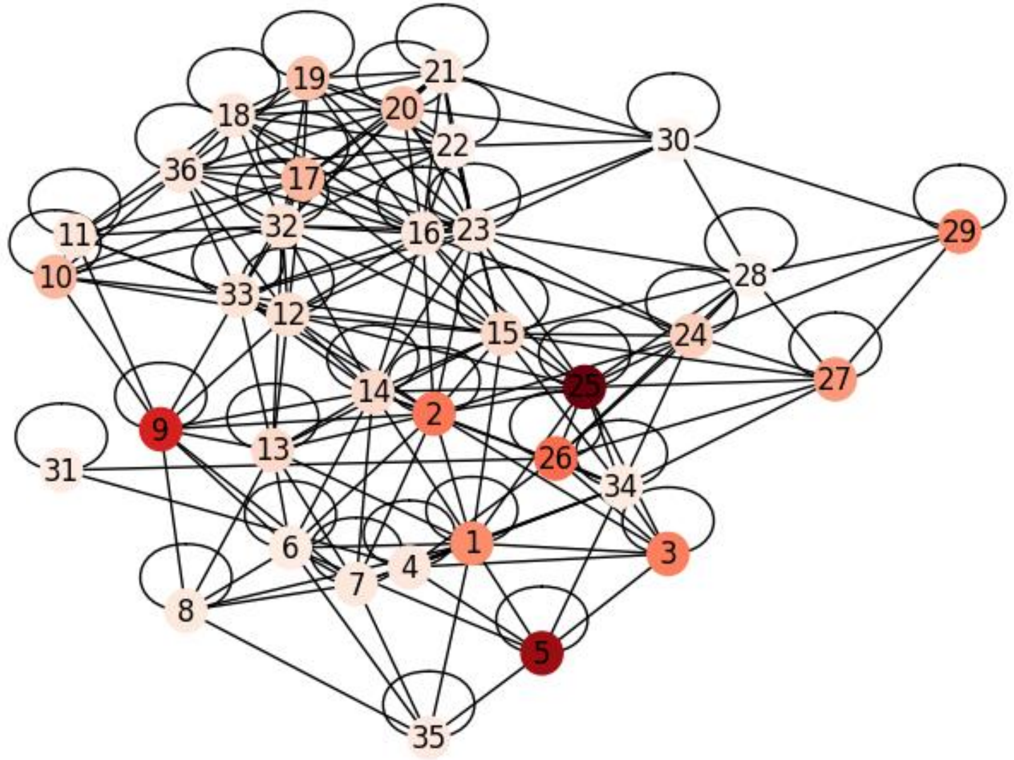
```
Number of graphs generated: 400
Number of convolved outputs: 400
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_87 (Dense)            (None, 40)                1640

 dense_88 (Dense)            (None, 20)                820

 dense_89 (Dense)            (None, 10)                210

 dense_90 (Dense)            (None, 5)                 55

 dense_91 (Dense)            (None, 1)                 6

=================================================================
Total params: 2,731
Trainable params: 2,731
Non-trainable params: 0
```

(a) Neural network model

Implementation and Results



(a) Inputted Image 1



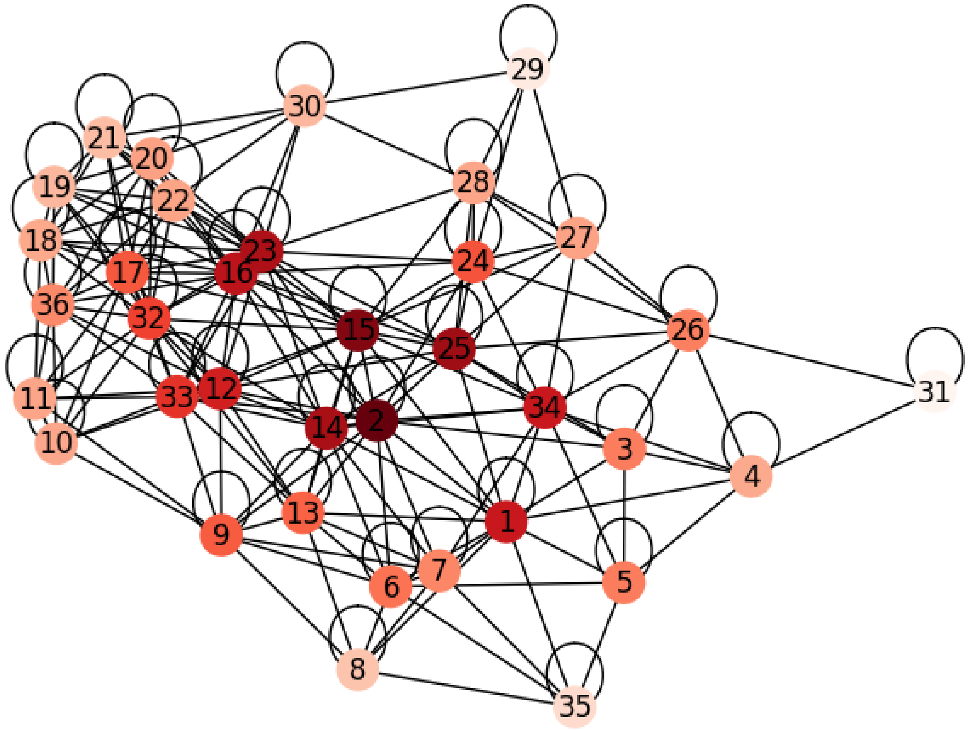
(b) Graph representation of inputted Image 1

Implementation and Results

```
Number of nodes:  36
Size of adjacency matrix:  (36, 36)
Node features befor message passing:
[264 304 288  72 572  56  66  66 462 180  78  96 110 105 110  72 182  77
 165 154  64  42  64 132 640 323 238  20 270  21  56  90  81  55  63  72]
Adjacency Matrix:
[[1.         0.41373972 0.61327898 ... 0.40637284 0.84959141 0.        ]
 [0.41373972 1.         0.82483163 ... 0.5153882  0.         0.        ]
 [0.61327898 0.82483163 1.         ... 0.31380284 0.         0.        ]
 ...
 [0.40637284 0.5153882  0.31380284 ... 1.         0.         0.        ]
 [0.84959141 0.         0.         ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.         0.         1.        ]]
Node features after message passing
[[2168.82223242 2134.76750797 2047.67910733 1234.43523463 1253.26804097
  1731.77122599 1015.51274091  887.00854593 1375.07780277 1006.55069032
  1029.43541061 1993.78055222 1093.52495067 1712.52611886 1825.88283327
  1927.02453698 1122.33135851  622.52203214  623.76043466  709.4556963
   616.93269243  692.46502495 1894.90937099 1402.99539704 2103.44970787
  1390.22899702 1157.0818617  1095.54483279  585.26511162  593.01438577
   364.85579985 1236.64774881 1351.34317877 2322.65625994  887.86007242
   879.2660503 ]]
```

(a) Information related to graph generated from Image 1

Implementation and Results

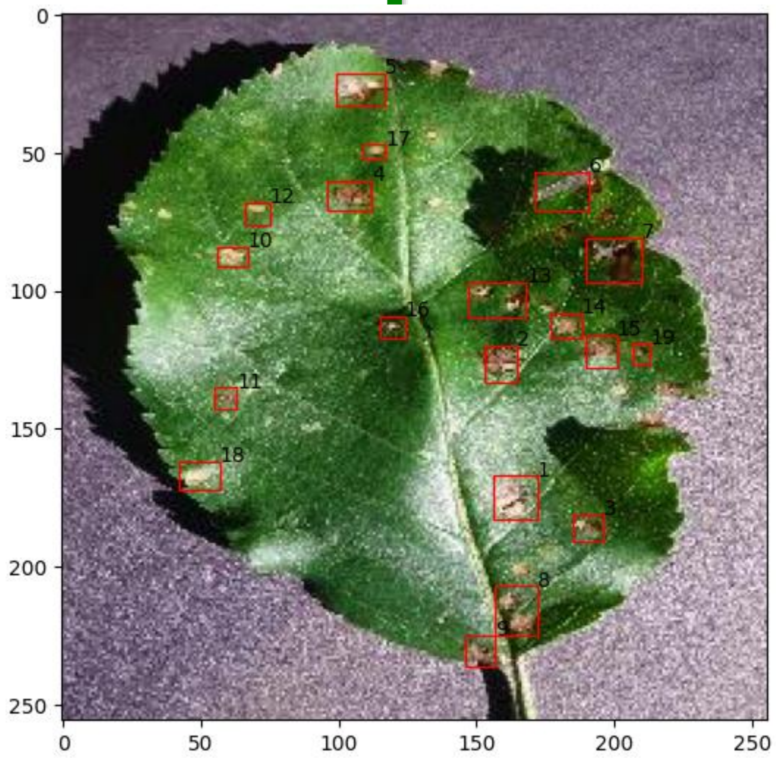


(a) Graph representation of Image 1 after message passing

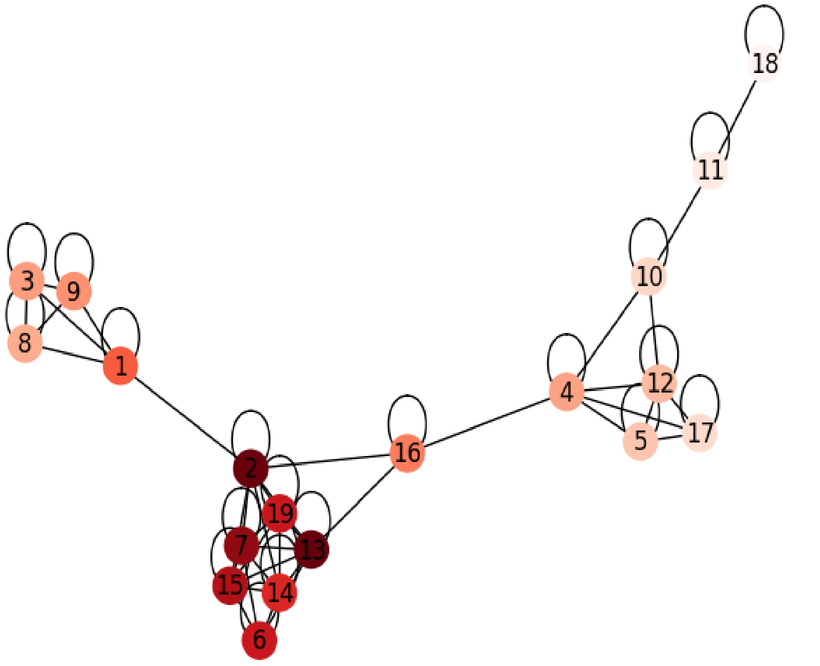
```
1/1 [==============================] - 0s 76ms/step
Predicted percentage of area the disease has spread to:  77.372986
Computed percentage of area damaged by disease:  30.453333333333333
Computed severity measure:  43.704827297159326
```

(b) Estimated severity of Image 1

Implementation and Results



(a) Inputted Image 2



(b) Graph representation of Image 2 after message passing

```
1/1 [==============================] - 0s 19ms/step
Predicted percentage of area the disease has spread to:  26.885326
Computed percentage of area damaged by disease:  15.637333333333334
Computed severity measure:  19.773683642809008
```

(c) Estimated severity of Image 2

Conclusion and Future Work

- This article presented a novel method to efficiently estimate the severity of disease in farmland considering the spatial spread of the disease for better resource planning.
- It can perform spatial analysis on existing conditions but cannot predict the further spread of the disease.
- Development of an A-CPS with the help of IoAT solutions that can efficiently predict areas that can be affected subsequently and propose efficient routes for sprayers to effectively manage the disease can be sought for future research

Thank You !!