Dendritic Spine Detection

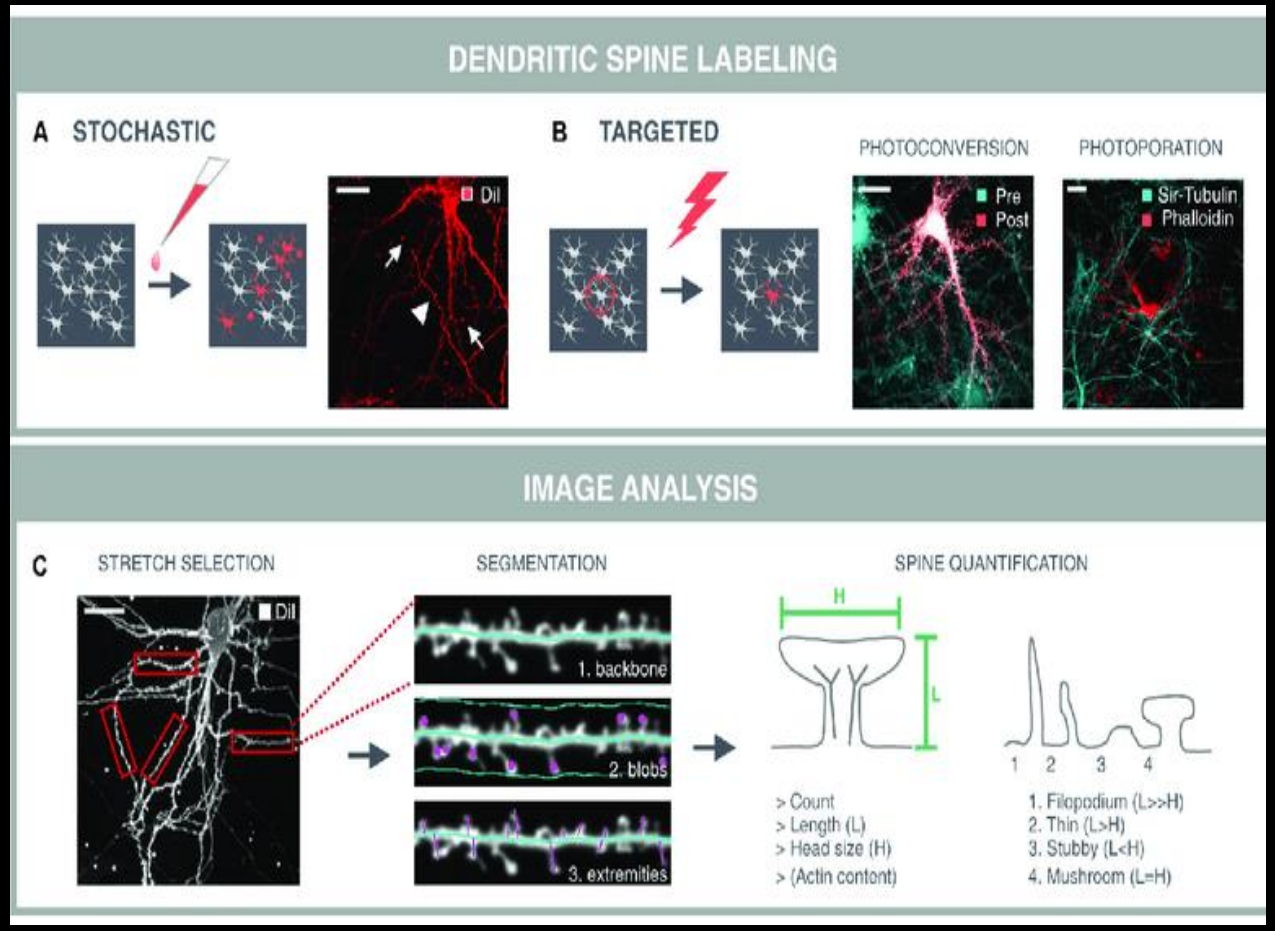
By: Alexa Alvarez and Lauren Pearl

Outline

- Introduction
- Github repository set up
- Image scanning
- Algorithms used:
  - **Random Forest**
  - **K-means Clustering**
- Errors and debugging
- Supervised vs unsupervised systems
- Conclusion

Refresh on what dendritic spines detection is

- Dendritic spine labelling is a technique used in neuroscience to selectively label and visualize dendritic spines (little protrusions on the dendrites of neurons).



Github Repository Setup

Setup Github respository for version control and collaboration

Random Forest

```
!git clone https://github.com/ryoheiyasuda/SpineDetector.git
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from PIL import Image
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

K-Means Clustering

```
import subprocess

repository_url = "https://github.com/alexaalvarez2021/new-.git"


try:
    subprocess.run(["git", "clone", repository_url], check=True)
    print("Repository cloned successfully.")
except subprocess.CalledProcessError as e:
    print(f"Error: {e}")
```

```
Repository cloned successfully.
```

Image Scanning

scan images of dendritic spines for analysis

Random Forest

```
image_fpath = os.path.join(folder, 'train', 'images', images[idx])
img = Image.open(image_fpath)
img = img.resize((256, 256)).convert("L")  # Resize and convert to grayscale

label_fname = images[idx].strip(".jpg") + ".txt"
label_fpath = os.path.join(folder, 'train', 'labels', label_fname)
lbl = np.loadtxt(label_fpath)

x = lbl[:, 1] * 255
y = lbl[:, 2] * 255

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7))
ax1.imshow(img, cmap="gray")
ax2.imshow(img, cmap="gray")
ax2.scatter(x, y, color="r", marker="x")
plt.show()

threshold = 65
img_thresh = img.point(lambda value: 255 if value > threshold else 0)

x = np.array(img_thresh).reshape(-1)
y = lbl[:, 0].astype(int)

# Match lengths of x and y
min_length = min(len(x), len(y))
x = x[:min_length
y = y[:min_length
    ndarray: y_train
    ndarray with shape (1,)

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
X_train = X_train.reshape(-1, 1)
```

K-Means Clustering

```
[ ] uploaded_image = files.upload()
```

Choose Files | Labeled_S...EtAl2018.zip

- **Labeled_Spines_SmirnovEtAl2018.zip**(application/zip) - 31807059 bytes, last modified: 4/15/2024 - 100% done

Saving Labeled_Spines_SmirnovEtAl2018.zip to Labeled_Spines_SmirnovEtAl2018.zip

```
uploaded_spine_info = files.upload()
```

Choose Files | Labeled_S...EtAl2018.zip

- Labeled_Spines_SmirnovEtAl2018.zip(application/zip) - 31807059 bytes, last modified: 4/15/2024 - 100% done

Saving Labeled_Spines_SmirnovEtAl2018.zip to Labeled_Spines_SmirnovEtAl2018 (1).zip

```
[ ] uploaded_bounding_box = files.upload()
```

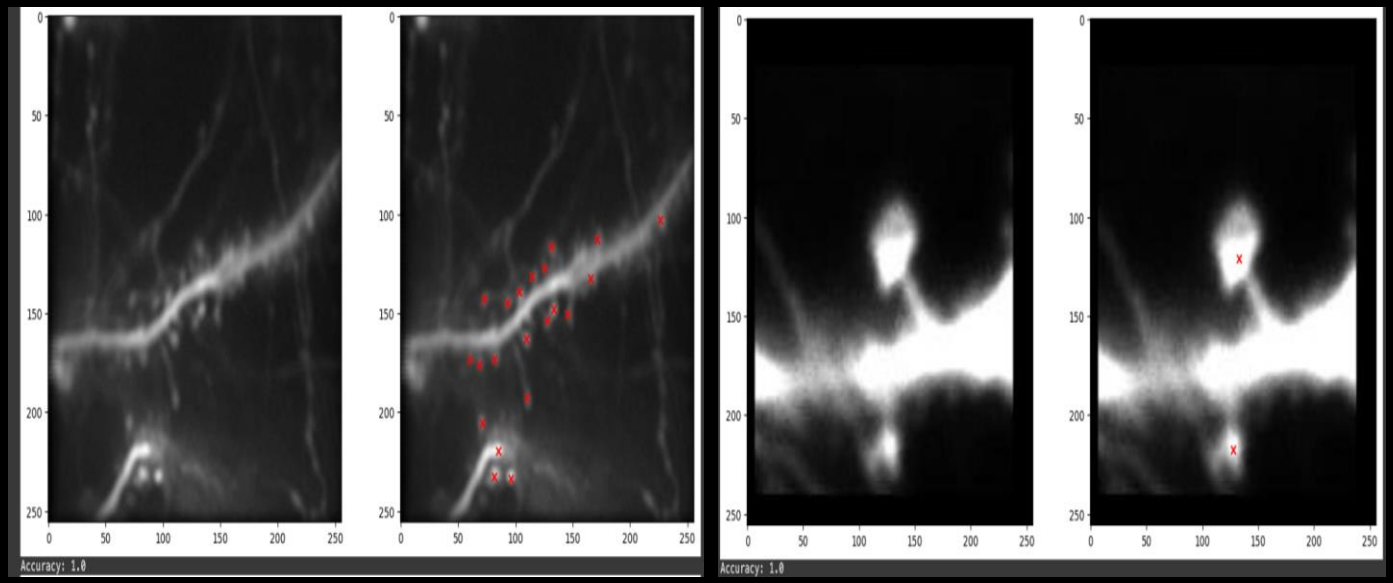
Choose Files | Labeled_S...EtAl2018.zip

- Labeled_Spines_SmirnovEtAl2018.zip(application/zip) - 31807059 bytes, last modified: 4/15/2024 - 100% done

Saving Labeled_Spines_SmirnovEtAl2018.zip to Labeled_Spines_SmirnovEtAl2018 (2).zip

```
[29] import pandas as pd
     import zipfile

     # Path to the ZIP archive
     zip_file_path = "/Labeled_Spines_SmirnovEtAl2018.zip"
     # Extract the CSV file from the ZIP archive
     with zipfile.ZipFile(zip_file_path, "r") as zip_ref:
         csv_file_name = zip_ref.namelist()[0]  # Assuming the CSV file is the first one in the archive
         with zip_ref.open(csv_file_name) as csv_file:
             # Read the CSV file with pandas
             data = pd.read_csv(csv_file)

[7] print(data.head())
    print(data.info())
```

```
        rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
        rf_classifier.fit(X_train, y_train)

        if X_test.ndim == 1:
            X_test = X_test.reshape(-1, 1)
        y_pred = rf_classifier.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        print("Accuracy:", accuracy)


file_paths = [127, 128, 126, 125, 124, 130]  # List of image indices
process_images(file_paths)
```



Algorithm #1: **Random Forest and Results**

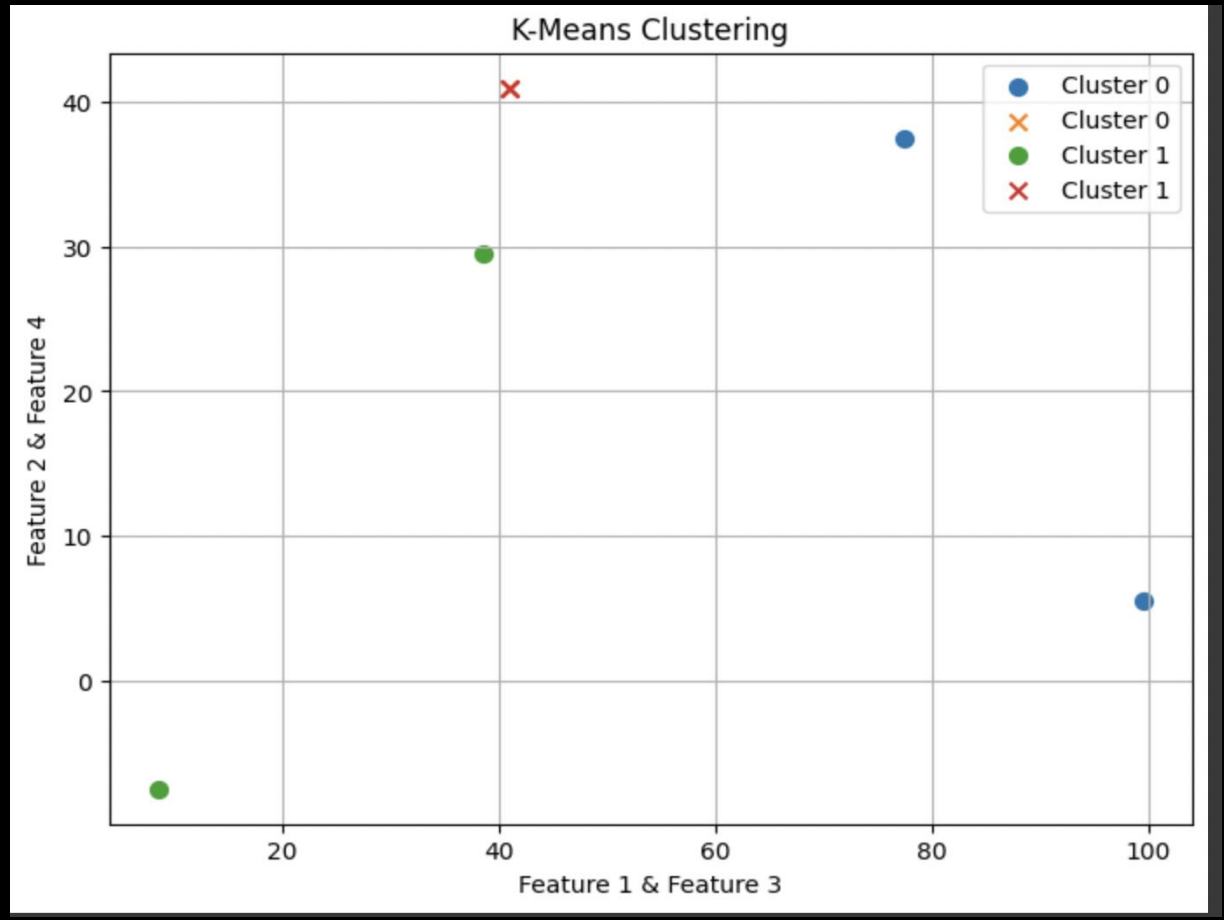
Algorithm #2: K-Means Clustering and results

Code + Text

```
9] # Add the cluster labels to the DataFrame
   data['cluster'] = cluster_labels

   # Print the clusters
   print(data)
```

```
   feature1  feature2  feature3  feature4  cluster
0     77.52     37.52     40.96     40.96        0
1     38.52     29.52     40.96     40.96        1
2     99.52      5.52     40.96     40.96        0
3      8.52     -7.48     40.96     40.96        1
```

```
import matplotlib.pyplot as plt

# Create scatter plot
plt.figure(figsize=(8, 6))

# Plot data points for each cluster
for cluster_id in data['cluster'].unique():
    cluster_data = data[data['cluster'] == cluster_id]
    plt.scatter(cluster_data['feature1'], cluster_data['feature2'], label=f'Cluster {cluster_id}', marker='o', s=50
    plt.scatter(cluster_data['feature3'], cluster_data['feature4'], label=f'Cluster {cluster_id}', marker='x', s=50

# Add labels and legend
plt.xlabel('Feature 1 & Feature 3')
plt.ylabel('Feature 2 & Feature 4')
plt.title('K-Means Clustering')
plt.legend()
plt.grid(True)

# Show plot
plt.show()
```



Errors and Debugging

- #1- Missing Dependencies: Making sure all necessary libraries were installed
- #2- File path errors: verifying the path is specified
  - Example: ('/content/SpineDetector/New_Dataset3') contained subfolders ('train/images')
- #3- Image Loading and Processing: making sure images are in correct format/not distorted and can be opened
  - Example: Image resizing ('img.resize((256,256))')

#4 - Incorrect Number of Clusters (K): important in ensuring optimal clustering results

Supervised vs Unsupervised

Supervised machine learning algorithms like random forest require labeled training data to learn patterns and make predictions. Unsupervised algorithms like k-means clustering can find hidden patterns and groupings without labeled data.

Conclusion

Implementing and comparing random forest and k-means clustering machine learning models provided insights into detecting dendritic spines in neuron images. The supervised random forest model produced more accurate spine segmentations but required time-consuming manual image labeling. Overall, this project established framework for iterative improvement of dendritic spine detection through open-source code.