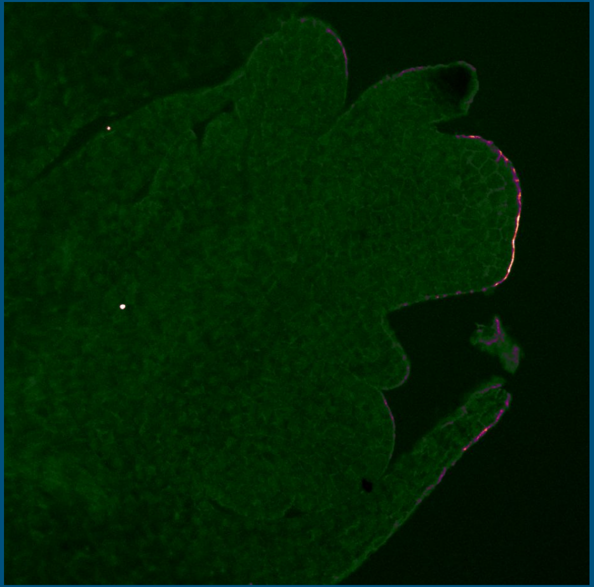
Computer Model for Classifying Cell Wall Antibody Staining

By Charles Maus

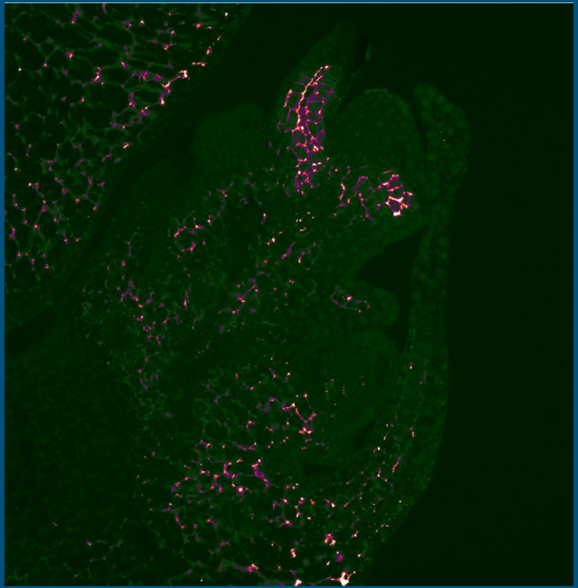
Localization of cell wall homogalacturonans can be visualized through immunofluorescence microscopy

LM20



Methylesterified HG

LM19



De-methylesterified HG

Raw microscope images can be modified to form data set

Download ImageMagick -

```
##Magick Downloaded (Used in Bash/Command Line)
aining\LM19\
magick mogrify -format jpg LM19*.tif
## Change color
magick LM19*.jpg -colorspace Gray LM19*BW.jpg
##Repeat for LM20
cd .
cd LM20
 magick mogrify -format jpg LM20*.tif
 magick LM20*.jpg -colorspace Gray LM20*BW.jpg
```

There are python packages for image analysis and classification

TensorFlow

matplotlib

```
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import pathlib
```

We have to define the data set for the model

```
data_dir = "C:/Users/cccsm/praccomp2024/FinalProject/ImagesforModelTraining/ModelImages"
pathlib.Path(data_dir).with_suffix('')
```

✓ 0.0s

```
WindowsPath('C:/Users/cccsm/praccomp2024/FinalProject/ImagesforModelTraining/ModelImages')
```

```
batch_size = 4 ## Should we do 1 at a time?
img_height = 1024
img_width = 1024
```

✓ 0.0s

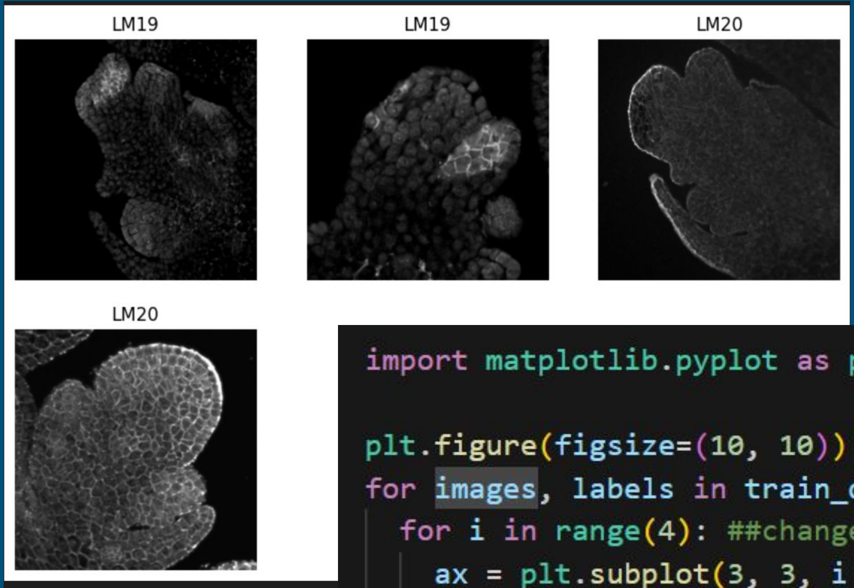
```
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

✓ 0.1s

```
Found 24 files belonging to 2 classes.
Using 20 files for training.
```

```
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

We can use matplotlib.pyplot to visualize what the model sees




```python
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(4): ##change range from 9-4; based on Batch size
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Once we have the data, it is compiled and run through the model

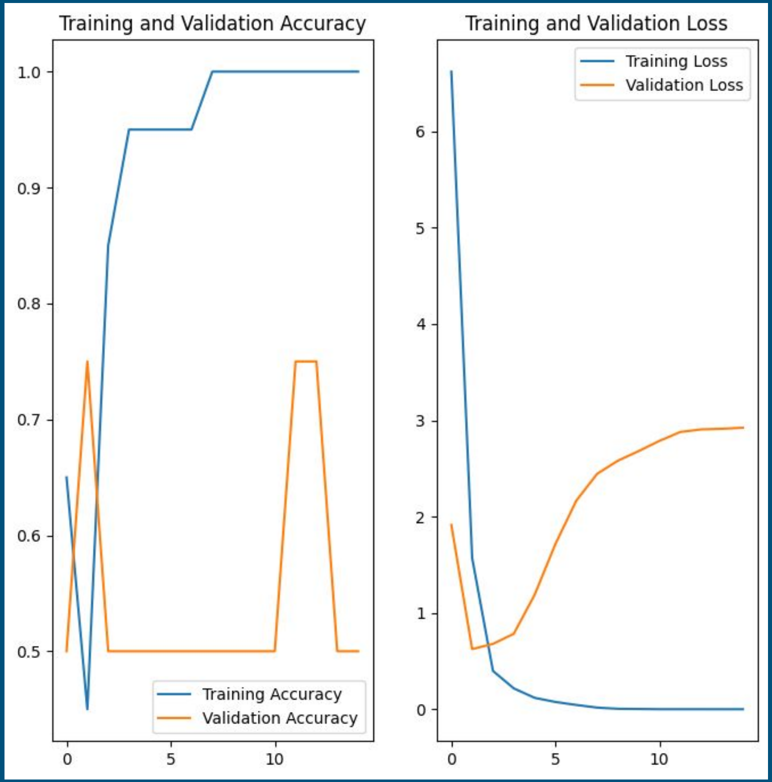
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.summary()
```

Layer (type)	Output Shape	Param #
rescaling_1 (Rescaling)	(None, 1024, 1024, 3)	0
conv2d (Conv2D)	(None, 1024, 1024, 16)	448
max_pooling2d (MaxPooling2D)	(None, 512, 512, 16)	0
conv2d_1 (Conv2D)	(None, 512, 512, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 256, 256, 32)	0
conv2d_2 (Conv2D)	(None, 256, 256, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 128, 128, 64)	0
flatten (Flatten)	(None, 1048576)	0
dense (Dense)	(None, 128)	134,217,856
dense_1 (Dense)	(None, 2)	258

```
epochs=15
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

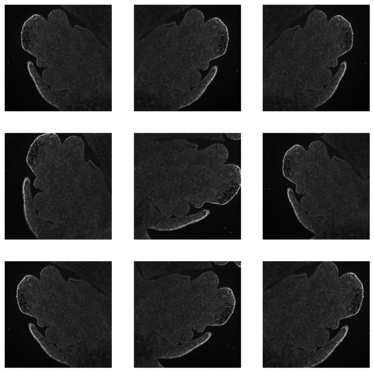
The model displays training success, but has limited validation success

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



To account for validation issues, we can optimize and augment the data

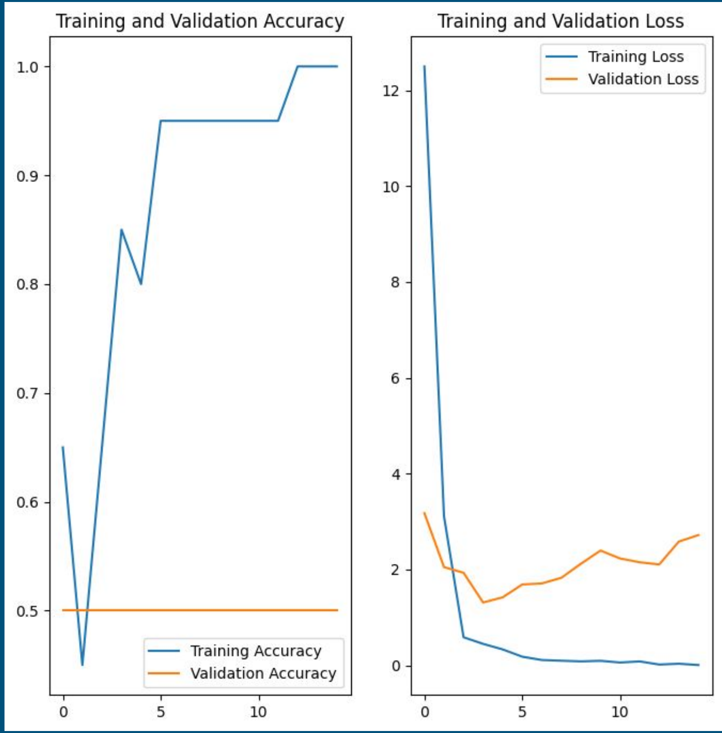
```
data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal",
                      input_shape=(img_height,
                                  img_width,
                                  3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)
plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
  for i in range(9):
    augmented_images = data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```



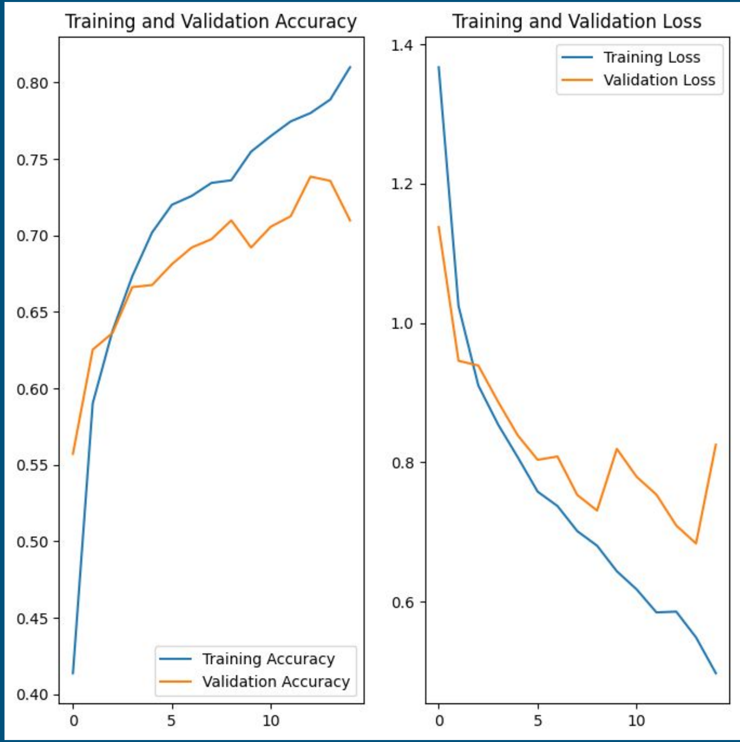
```
model = Sequential([
  data_augmentation,
  layers.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes, name="outputs")
```

The new validation and training plot displays the success of this augmentation

My Model:



Expected:



Let's Test it!

```python
LM20_path = "C:/Users/cccsm/praccomp2024/FinalProject/TestImages/NTestLM20bBW.jpg"
img = tf.keras.utils.load_img(
    LM20_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

✓ 0.1s

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step
This image most likely belongs to LM20 with a 99.79 percent confidence.
```

```python
LM20_path = "C:/Users/cccsm/praccomp2024/FinalProject/TestImages/NTestLM20aBW.jpg"
img = tf.keras.utils.load_img(
    LM20_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

✓ 0.1s

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
This image most likely belongs to LM20 with a 71.36 percent confidence.
```

Possible edits and ways to increase accuracy of classification & Future Applications

Additional Data - Increasing Data Set Size!

Randomizing Validation vs Training Images

Applying the model to other meristem types (SAMs, SPMs, etc.)