Wheat segmentation with Neural Networks

Kristina Rančić, Hunor Tot-Bagi

Agenda

01

Labeling Dataset

Description of hand labeling data set using Photoshop and its uploading and division in Google Colab Notebook.

02

Preprocessing

Several methods we applied on all original images in order of better recognizing wheat grains.

03

Model

Understanding of Keras Functional Model and our layers implementation.

04

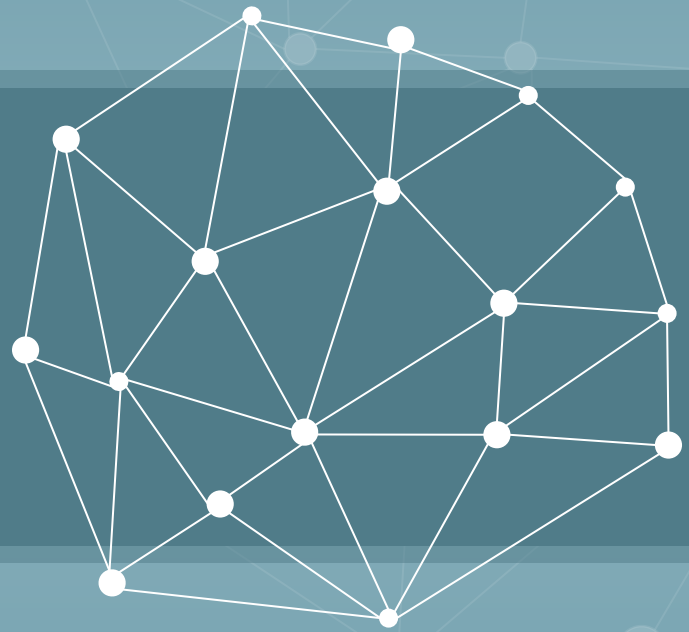
Results

Presentation of output images and counting function.

05

Modules overview

Python modules used in the project.



Labeling Dataset

Via Adobe Photoshop CC 2015

01

Brush tool 40px

02

Masks on new layer

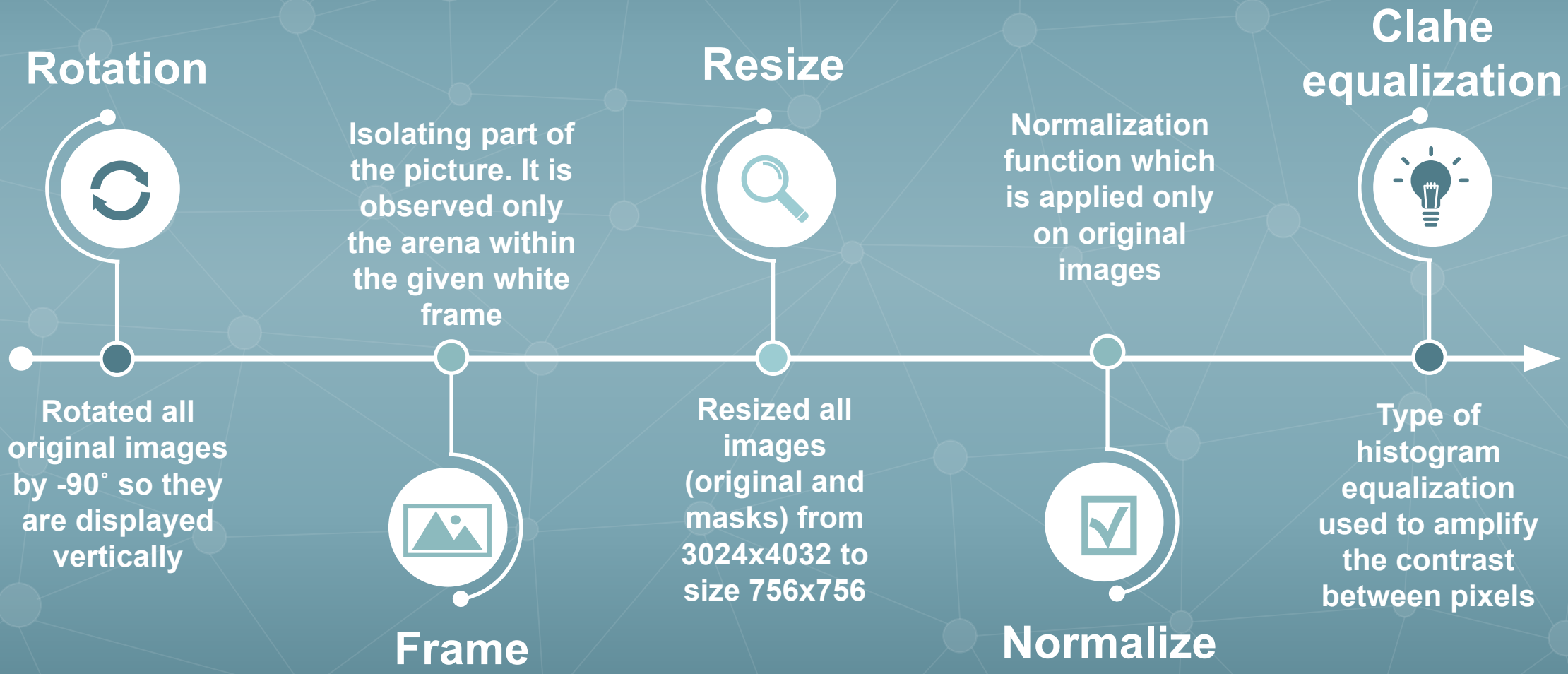
Wheat dataset

62 images

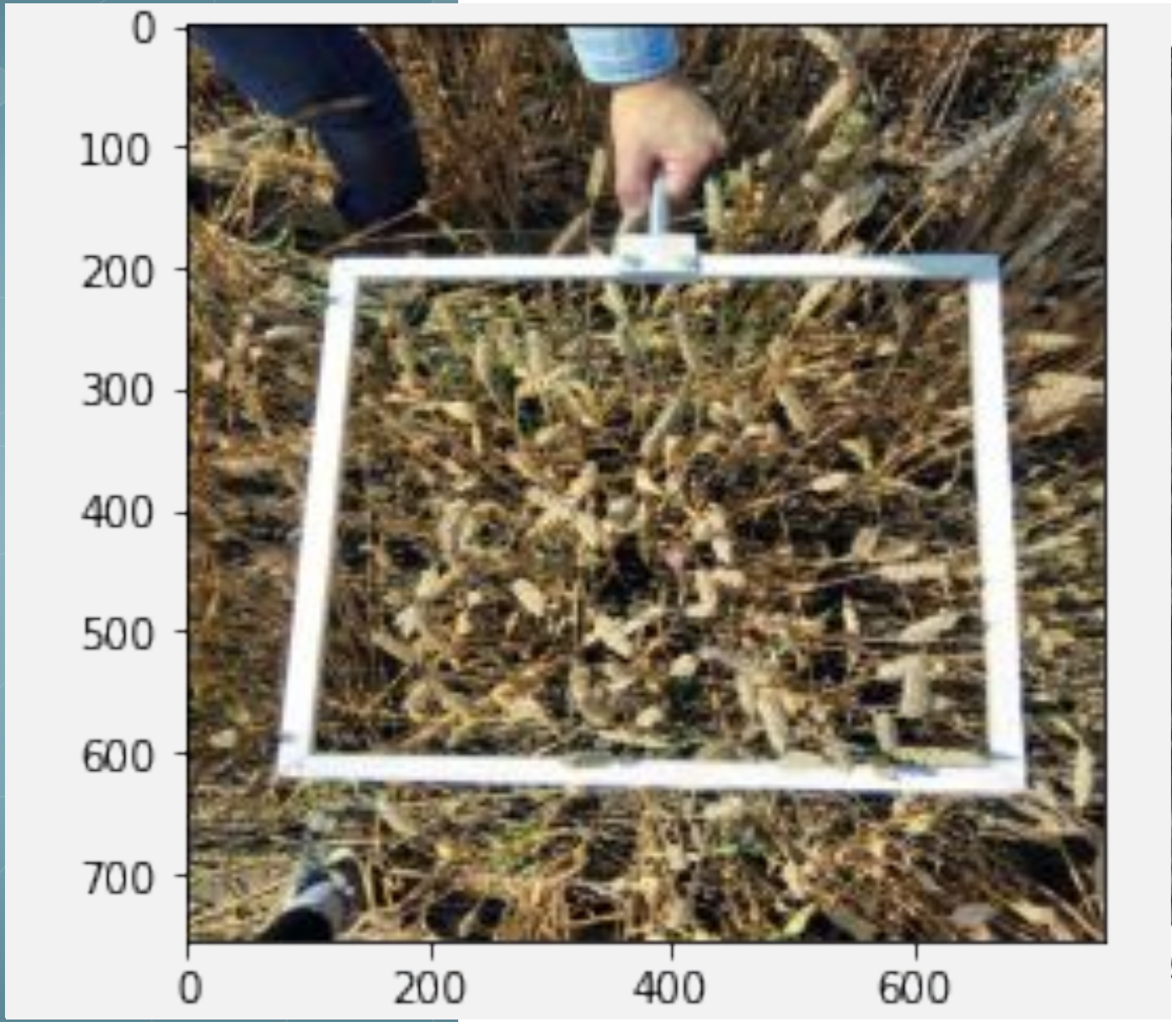
Division of dataset into training and testing parts:

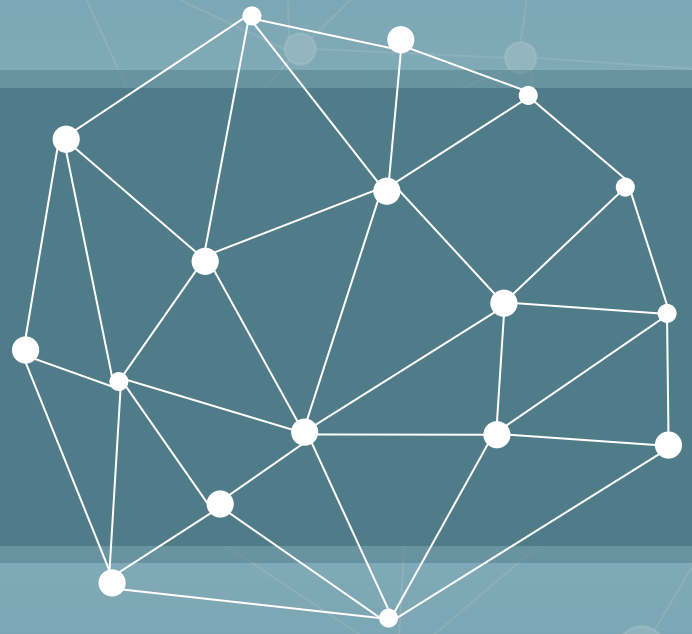
training - 50 images
testing - 12 images

Preprocessing functions



1) Original
2) Rotated
3) Resized
4) Normalized
5) Clahe equalization





Convolutional Neural Network

Keras Functional Model

```
input_size=(756, 756, 3)
inputs = Input(input_size)

conv1 = Conv2D(128, 3, activation='relu', padding='same', kernel_initializer='he_normal')(inputs)
conv1 = Conv2D(128, 3, activation='relu', padding='same', kernel_initializer='he_normal')(conv1)
pool1 = MaxPool2D(pool_size=(2, 2))(conv1)

conv2 = Conv2D(256, 3, activation='relu', padding='same', kernel_initializer='he_normal')(pool1)
conv2 = Conv2D(256, 3, activation='relu', padding='same', kernel_initializer='he_normal')(conv2)
pool2 = MaxPool2D(pool_size=(2, 2))(conv2)

...

conv8 = Conv2D(128, 3, activation='relu', padding='same', kernel_initializer='he_normal')(up7)
conv9 = Conv2D(64, 3, activation='relu', padding='same', kernel_initializer='he_normal')(conv8)
conv9 = Conv2D(1, 1, activation='sigmoid')(conv8)

model = Model(inputs=inputs, outputs=conv9)

model.compile(loss='binary_crossentropy', optimizer=Adam(learning_rate=0.0001),metrics=['accuracy'])
model.summary()

model.fit(tr_data, tr_mask, batch_size=1,
                    epochs=30,
                    shuffle=True,
                    verbose=1)
```

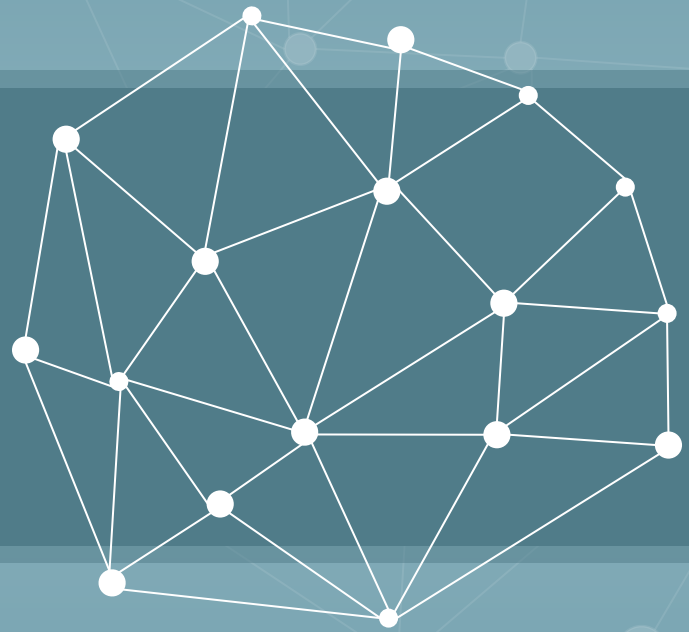
Define input

Connecting layers

Creating the Model

Model compile

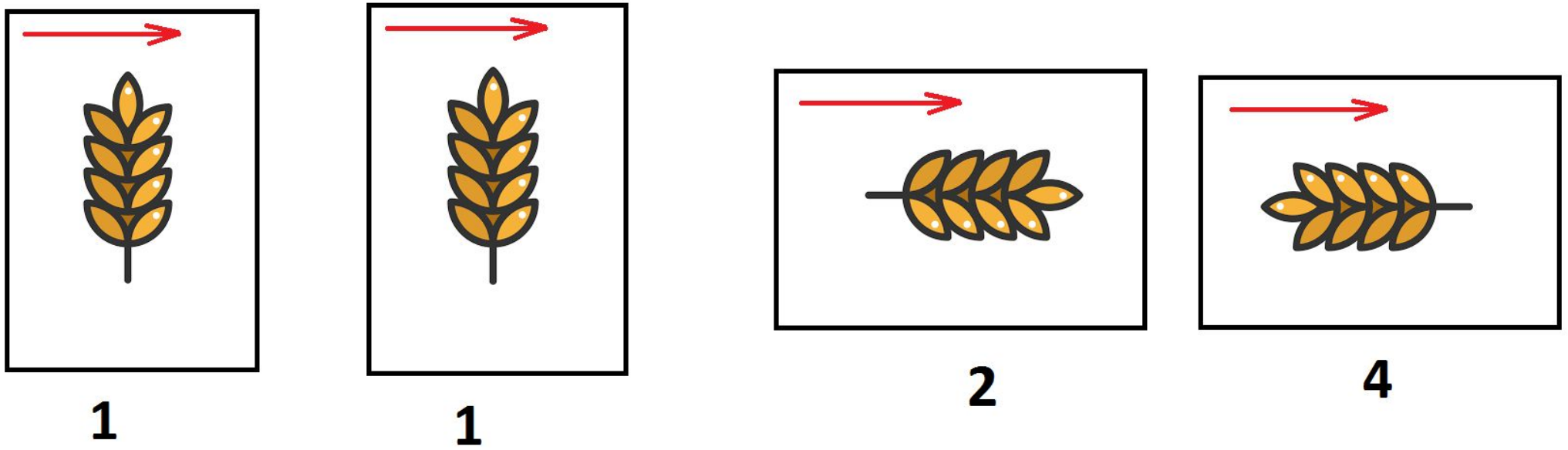
Model fit



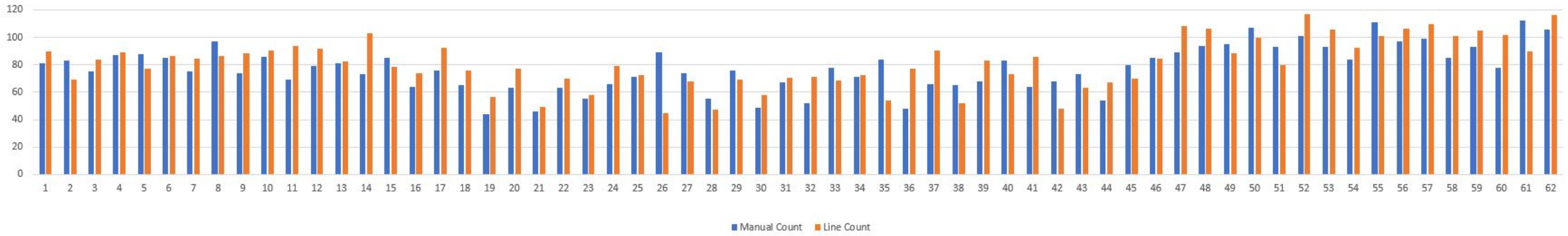
Wheat counting

Horizontal Line Scan

How to improve this algorithm?



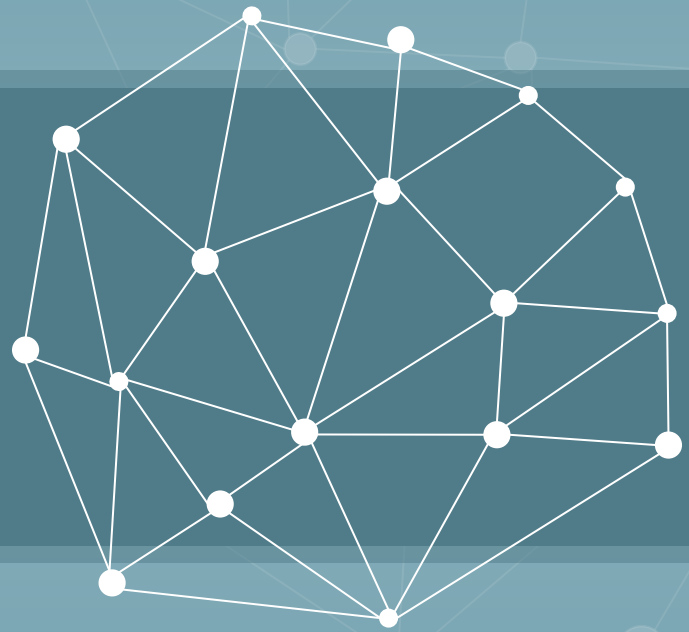
Absolute error on Masks



Min 1

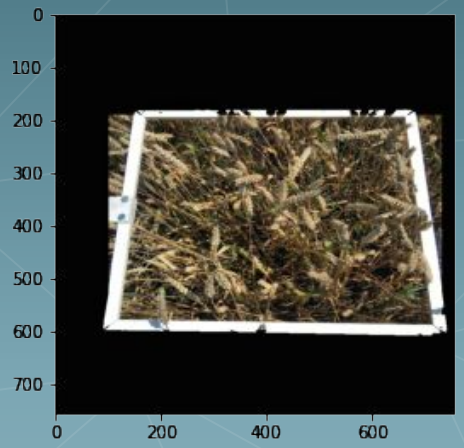
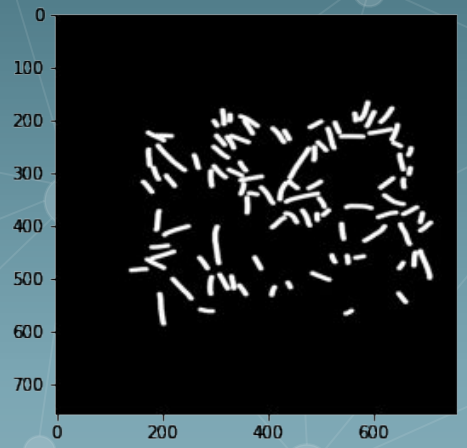
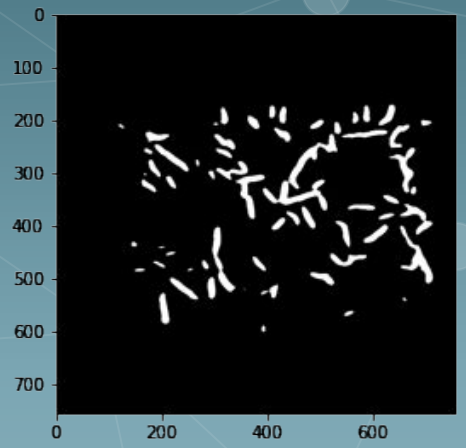
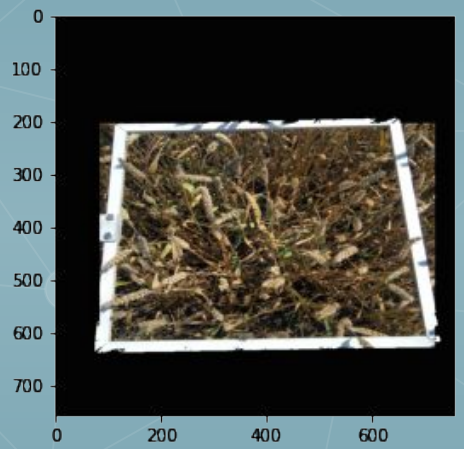
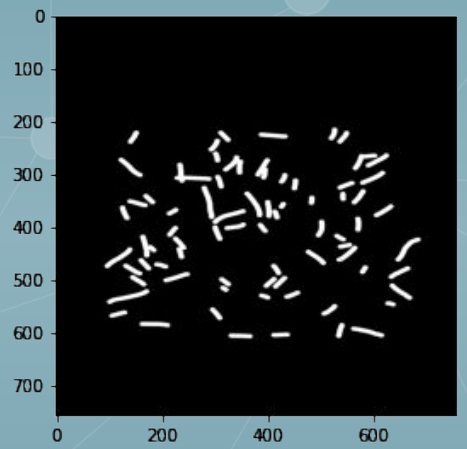
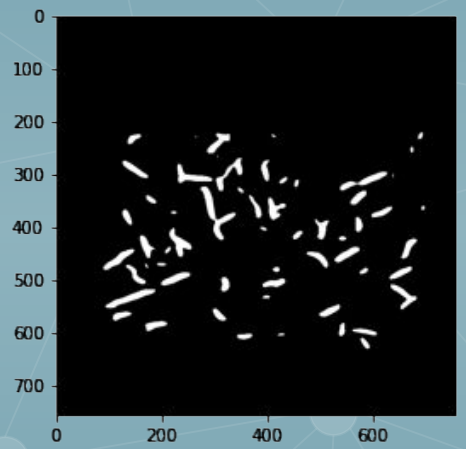
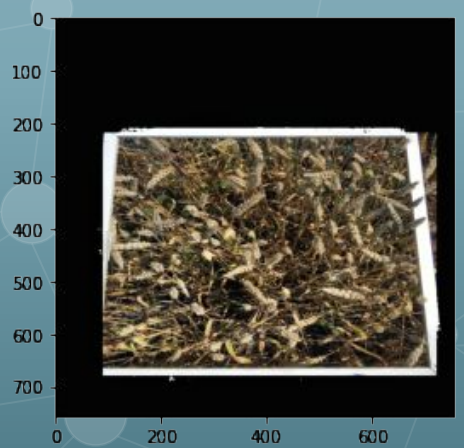
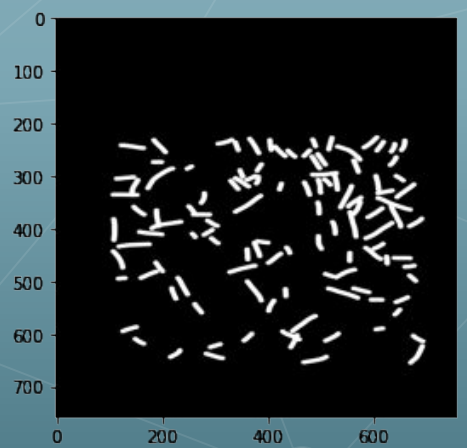
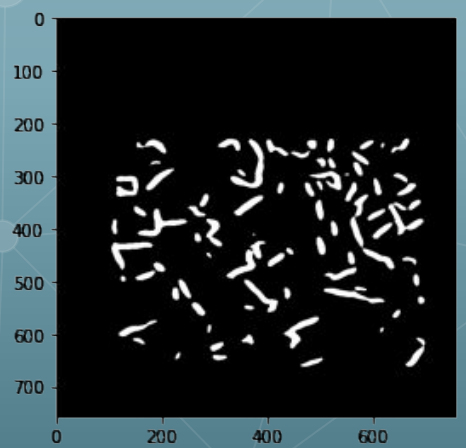
Max 44

Average 12



Results

```
Epoch 5/20
50/50 [==============================] - 124s 2s/step - loss: 0.1249 - accuracy: 0.9436 - val_loss: 0.2346 - val_accuracy: 0.9289
Epoch 6/20
50/50 [==============================] - 124s 2s/step - loss: 0.1115 - accuracy: 0.9455 - val_loss: 0.1369 - val_accuracy: 0.9289
Epoch 7/20
50/50 [==============================] - 124s 2s/step - loss: 0.1071 - accuracy: 0.9474 - val_loss: 0.1172 - val_accuracy: 0.9364
Epoch 8/20
50/50 [==============================] - 124s 2s/step - loss: 0.1003 - accuracy: 0.9497 - val_loss: 0.1207 - val_accuracy: 0.9352
Epoch 9/20
50/50 [==============================] - 124s 2s/step - loss: 0.0995 - accuracy: 0.9497 - val_loss: 0.1383 - val_accuracy: 0.9296
Epoch 10/20
50/50 [==============================] - 124s 2s/step - loss: 0.0986 - accuracy: 0.9501 - val_loss: 0.1245 - val_accuracy: 0.9365
Epoch 11/20
50/50 [==============================] - 124s 2s/step - loss: 0.0970 - accuracy: 0.9505 - val_loss: 0.1262 - val_accuracy: 0.9312
Epoch 12/20
50/50 [==============================] - 124s 2s/step - loss: 0.0945 - accuracy: 0.9511 - val_loss: 0.1259 - val_accuracy: 0.9322
Epoch 13/20
50/50 [==============================] - 124s 2s/step - loss: 0.0925 - accuracy: 0.9522 - val_loss: 0.1078 - val_accuracy: 0.9404
Epoch 14/20
50/50 [==============================] - 124s 2s/step - loss: 0.0928 - accuracy: 0.9521 - val_loss: 0.1133 - val_accuracy: 0.9393
Epoch 15/20
50/50 [==============================] - 124s 2s/step - loss: 0.0899 - accuracy: 0.9528 - val_loss: 0.1026 - val_accuracy: 0.9397
Epoch 16/20
50/50 [==============================] - 124s 2s/step - loss: 0.0889 - accuracy: 0.9533 - val_loss: 0.1216 - val_accuracy: 0.9350
Epoch 17/20
50/50 [==============================] - 124s 2s/step - loss: 0.0877 - accuracy: 0.9538 - val_loss: 0.1406 - val_accuracy: 0.9323
Epoch 18/20
50/50 [==============================] - 124s 2s/step - loss: 0.0864 - accuracy: 0.9542 - val_loss: 0.1062 - val_accuracy: 0.9404
Epoch 19/20
50/50 [==============================] - 124s 2s/step - loss: 0.0829 - accuracy: 0.9551 - val_loss: 0.1043 - val_accuracy: 0.9407
Epoch 20/20
50/50 [==============================] - 124s 2s/step - loss: 0.0826 - accuracy: 0.9552 - val_loss: 0.0992 - val_accuracy: 0.9428
```

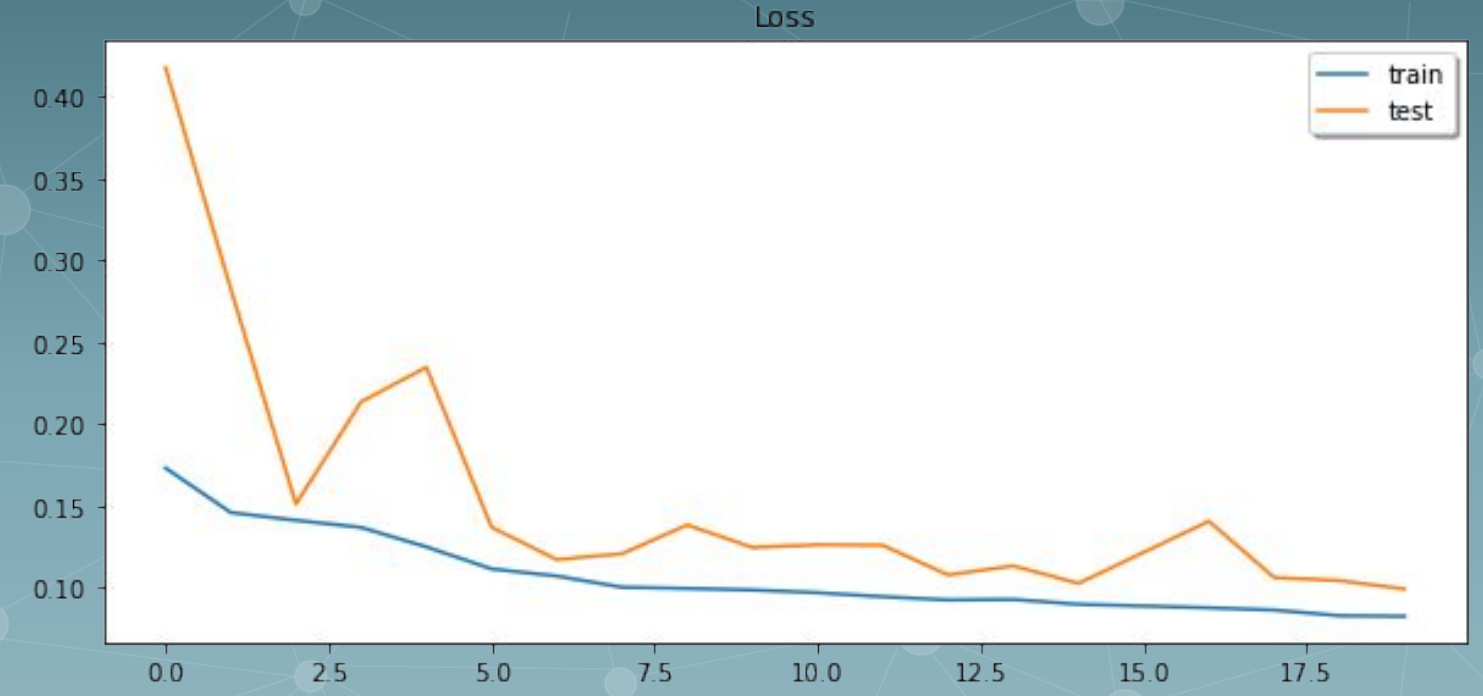










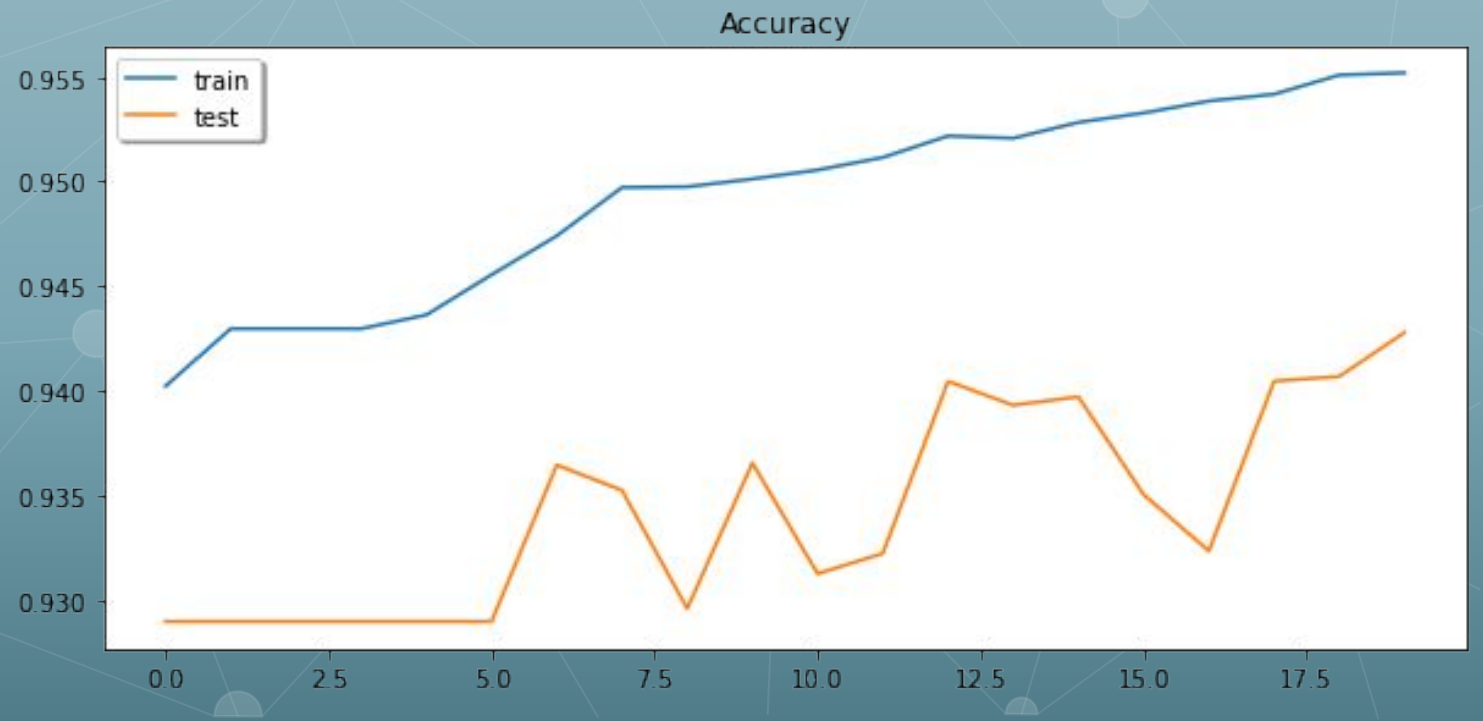
Statistics:

- 20 epochs
- Loss: 0.1
- Accuracy: 0.94

Metrics:

- Precision score: 0.64
- Recall score: 0.59
- F1 score: 0.61





Modules overview

- tensorflow.keras.layers - Conv2D, MaxPool2D, Dropout
- Pillow – Image, Interpolation
- Matplotlib.pyplot
- Cv2 – inRange, createCLAHE
- Numpy

THANK YOU
FOR YOUR ATTENTION