IACV Project

S.12 - Parcel Classification

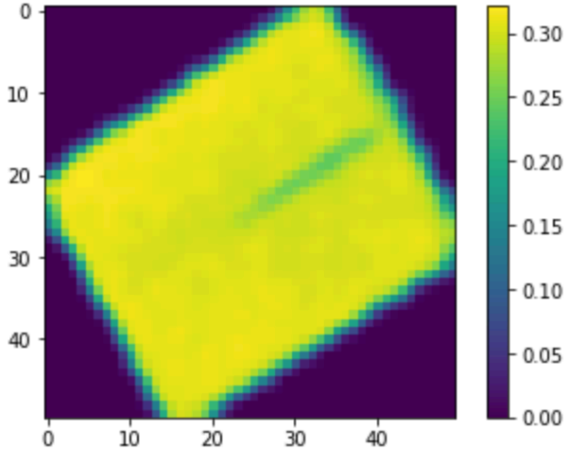
Ghitti Marco

S.12 Parcel Classification

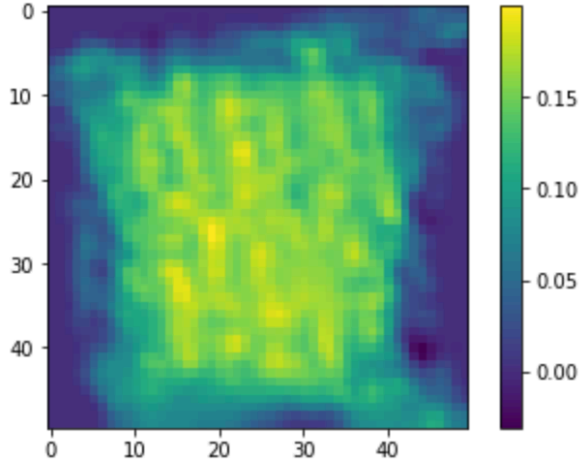
Classifying Images of different sizes acquired from a RGB-D sensor. Image characteristics:

1. No color information provided
2. A few pixels report depth measures
3. Images belong to 3 classes

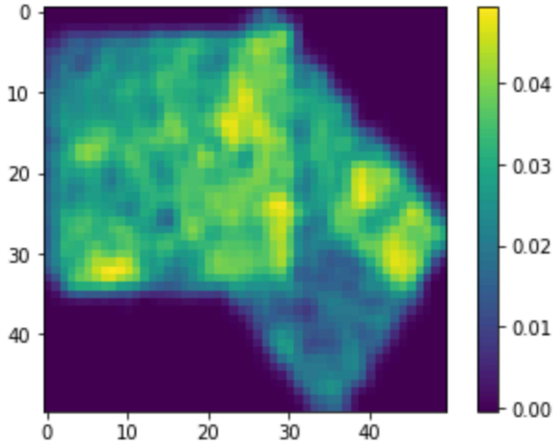
PARCEL



ENVELOPE



DOUBLE



Goal:

1. Train a CNN able to classify these images in the three classes.
2. Possibly compare the CNN performance with an hand-crafted classifier

Dataset and preprocessing

- Dataset composed of 2052 images
    - Double: 471
    - Envelope: 879
    - Parcel: 702

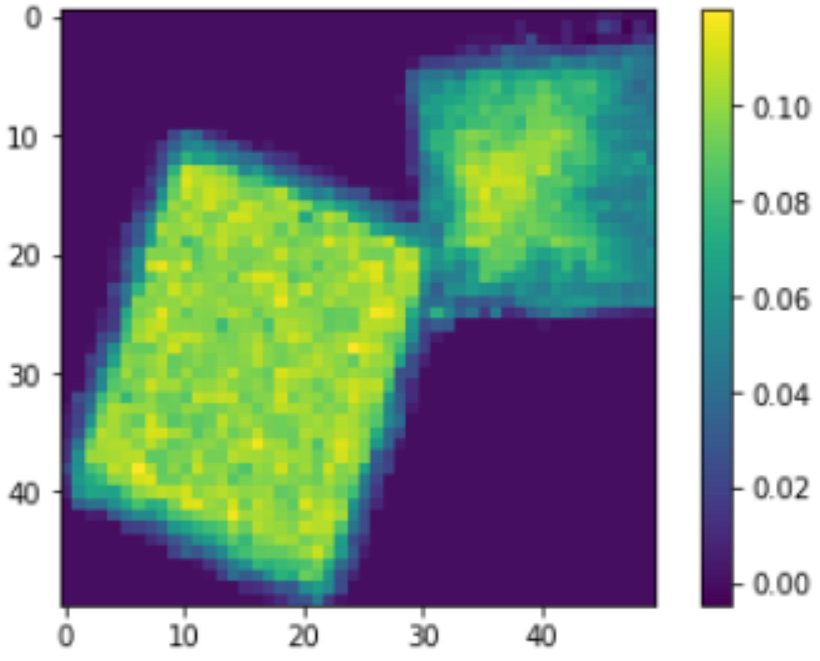
- All images have been resized to 50x50 pixels.

Hand-Crafted Classifier

Selected features: H5

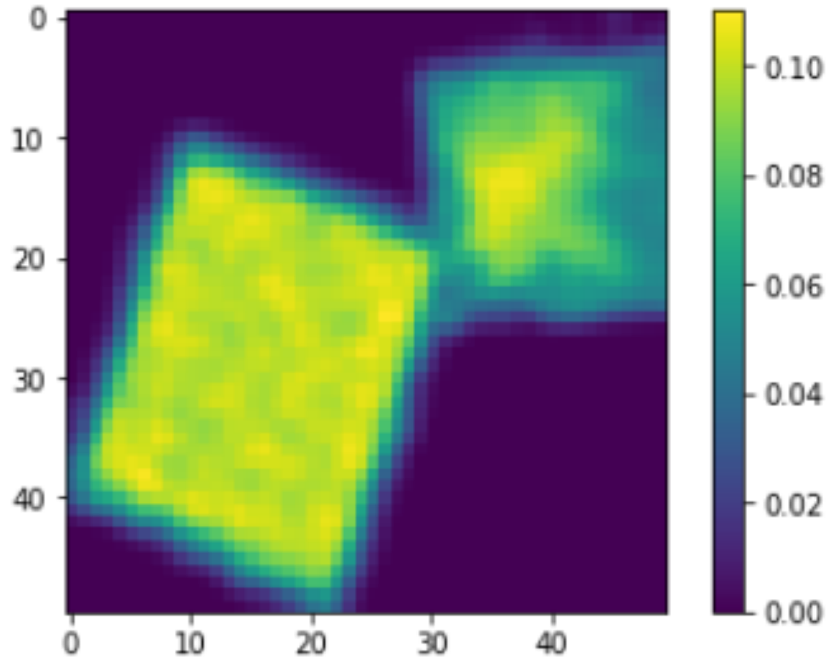
Hand-Crafted Classifier
Selected features: H5

1. Load the image



Hand-Crafted Classifier
Selected features: H5

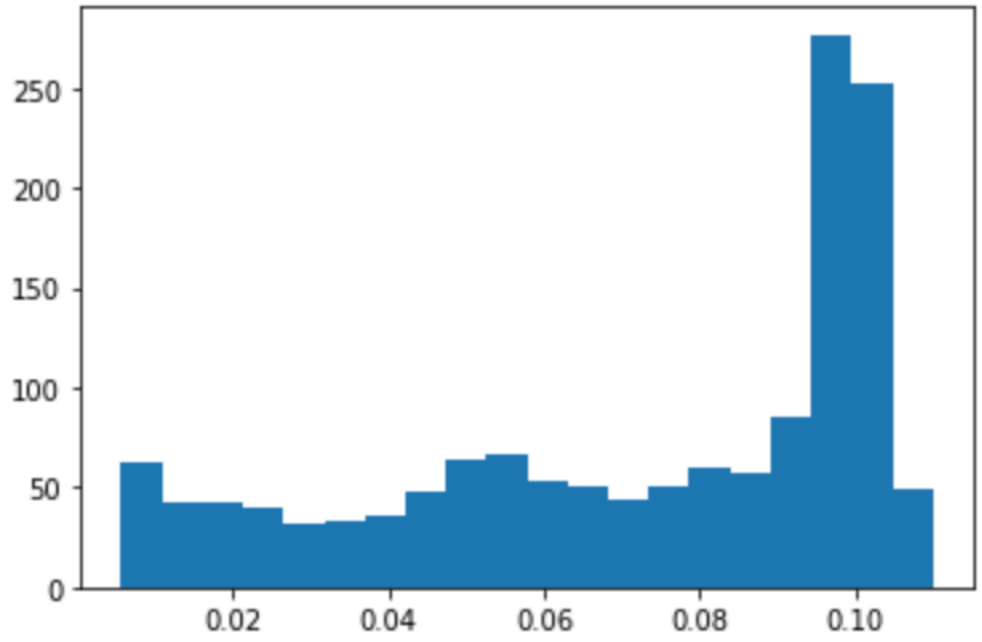
1. Load the image
2. Blur 3x3 kernel



Hand-Crafted Classifier
Selected features: H5

1. Load the image
2. Blur 3x3 kernel
3. Compute np.histogram

```
hist = np.histogram(blur3[np.where(blur3 > rng*0.05)].flatten(),bins=20)[0];
```



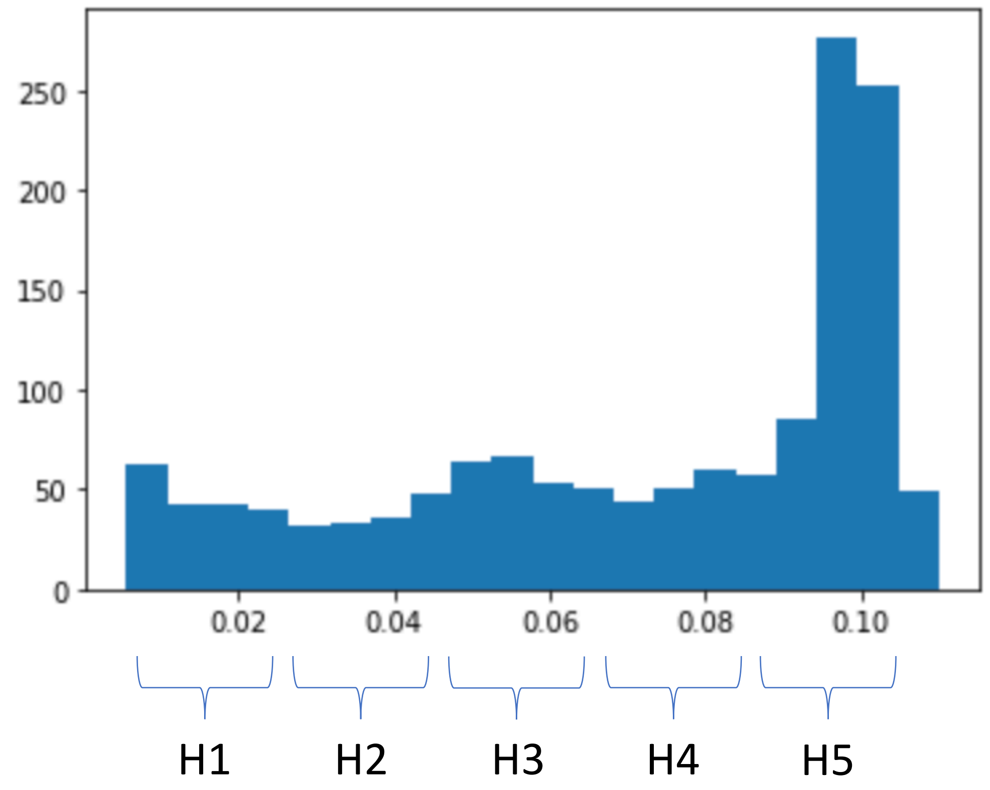
Hand-Crafted Classifier

Selected features: H5

1. Load the image
2. Blur 3x3 kernel
3. Compute np.histogram

```
hist = np.histogram(blur3[np.where(blur3 > rng*0.05)].flatten(),bins=20)[0];
```

4. Divide the histogram in 5 bins (H1,…,H5)

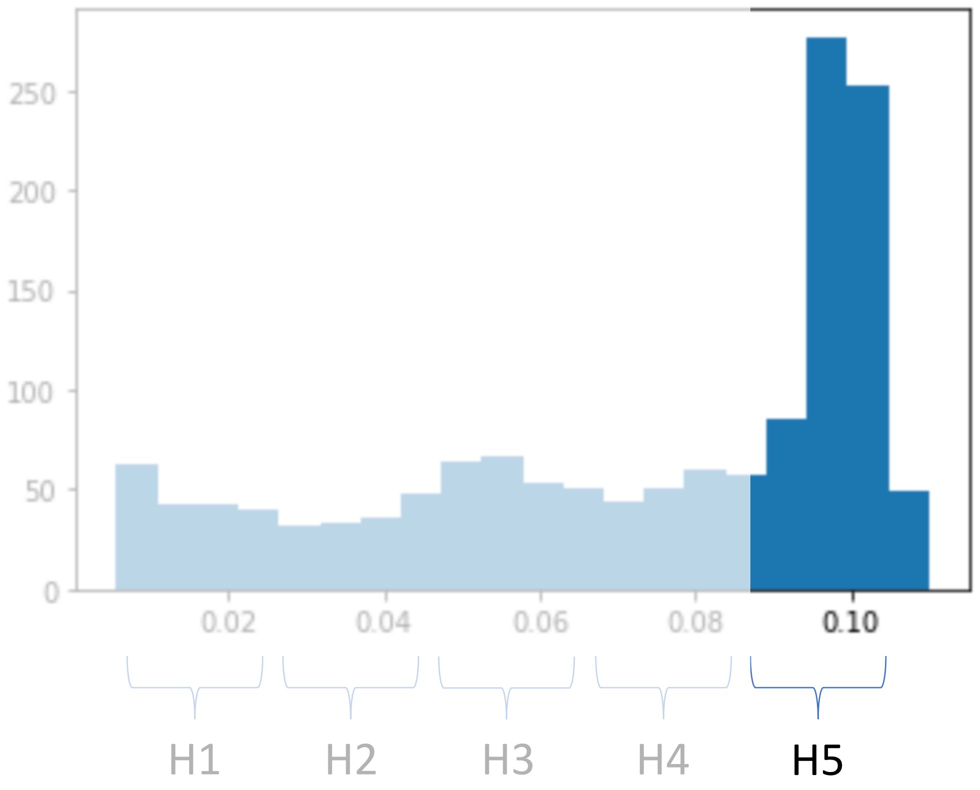


Hand-Crafted Classifier
Selected features: H5

1. Load the image
2. Blur 3x3 kernel
3. Compute np.histogram

```
hist = np.histogram(blur3[np.where(blur3 > rng*0.05)].flatten(),bins=20)[0];
```

4. Divide the histogram in 5 bins (H1,…,H5)
5. Take H5

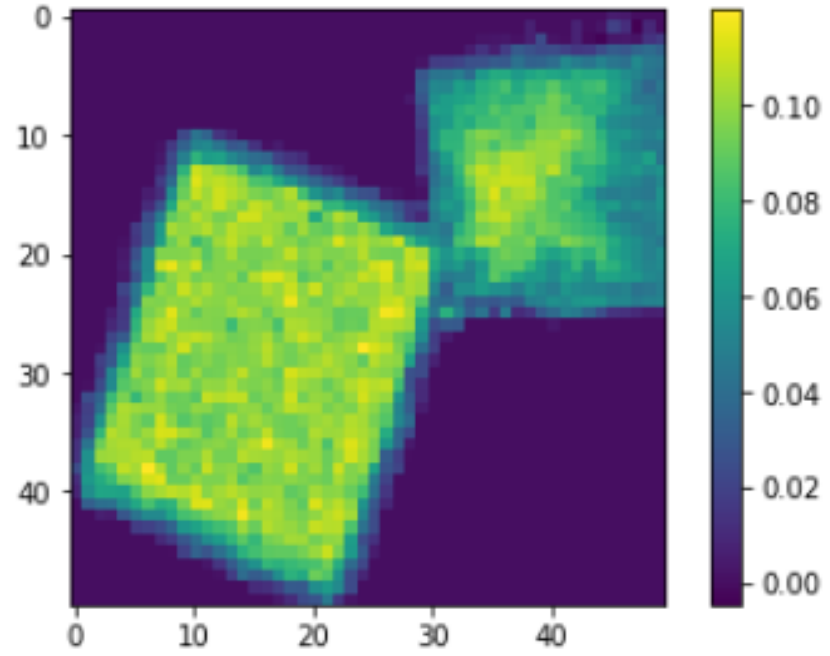


Hand-Crafted Classifier

Selected features: MaxMagOnArea

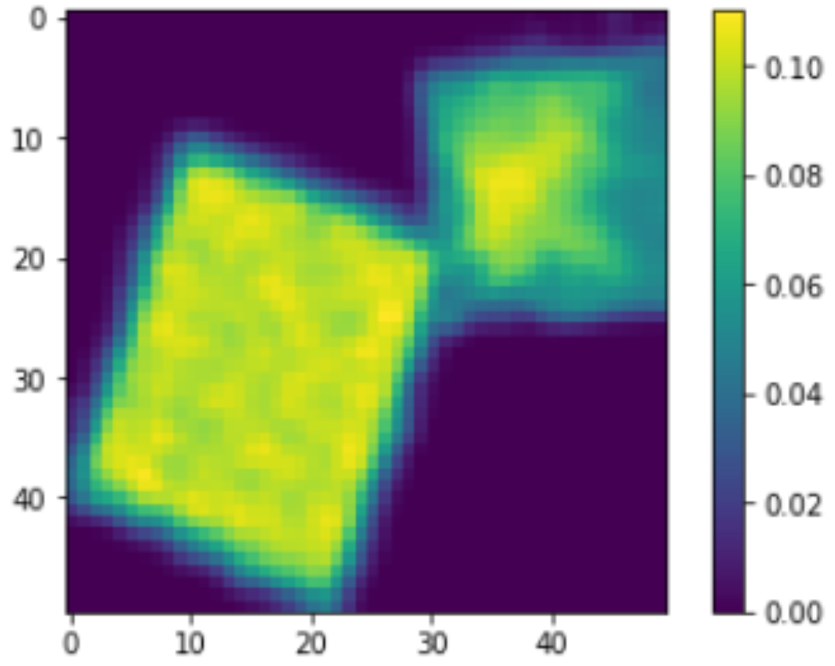
Hand-Crafted Classifier
Selected features: MaxMagOnArea

1. Load the image



Hand-Crafted Classifier
Selected features: MaxMagOnArea

1. Load the image
2. Blur 3x3 kernel

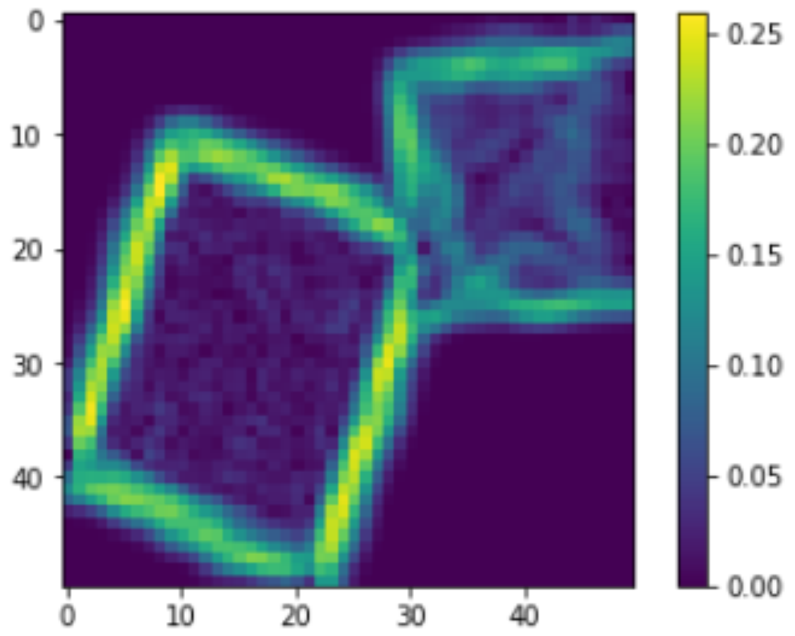


Hand-Crafted Classifier
Selected features: MaxMagOnArea

1. Load the image
2. Blur 3x3 kernel
3. Compute derivatives and gradient magnitude

```
dx = cv2.Sobel(blur3, cv2.CV_64F,1,0,ksize=3);
dy = cv2.Sobel(blur3, cv2.CV_64F,0,1,ksize=3);
```

```
mag, deg = cv2.cartToPolar(dx, dy, angleInDegrees=True);
```



Hand-Crafted Classifier

Selected features: MaxMagOnArea

1. Load the image
2. Blur 3x3 kernel
3. Compute derivatives and gradient magnitude

```
dx = cv2.Sobel(blur3, cv2.CV_64F,1,0,ksize=3);
dy = cv2.Sobel(blur3, cv2.CV_64F,0,1,ksize=3);
```

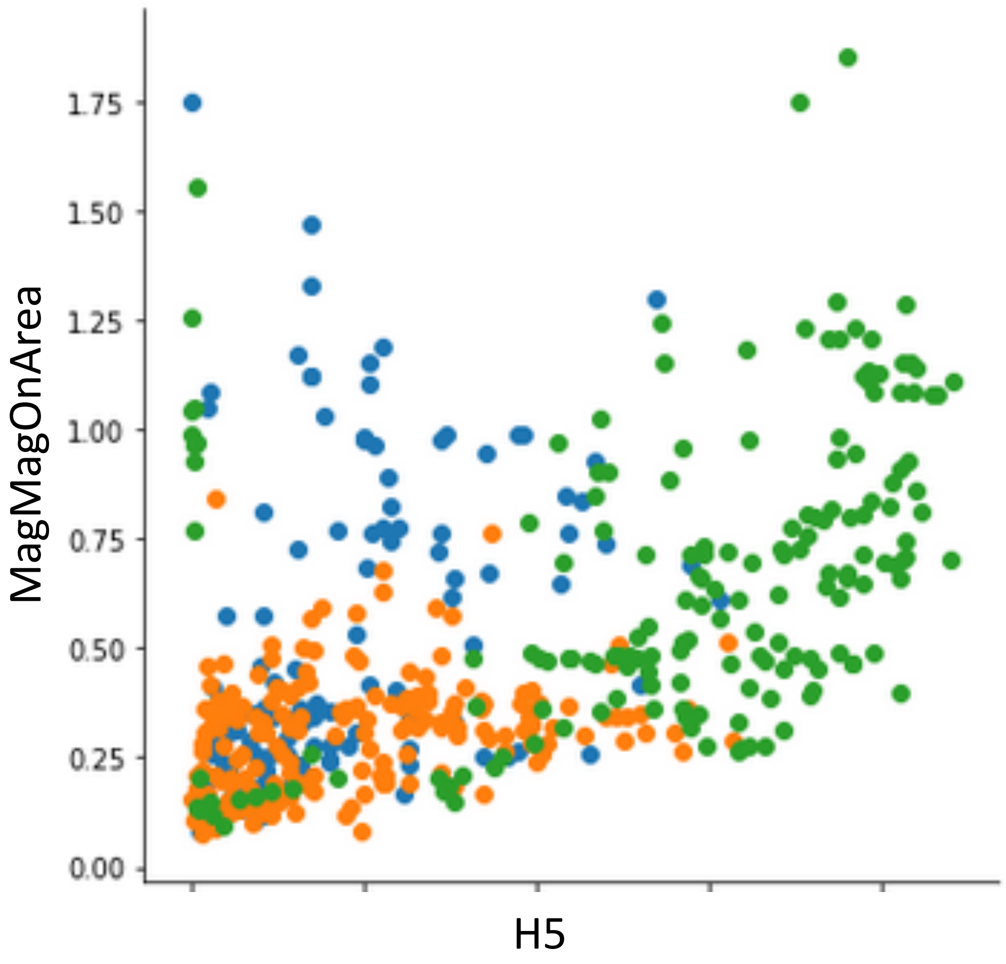
```
mag, deg = cv2.cartToPolar(dx, dy, angleInDegrees=True);
```

Max ≈ 0.26

4. Take the max value

```
maxMagOnArea = np.max(mag[np.where(blur3 > rng*0.05)])
```

Hand-Crafted Classifier
Scatterplot



Hand-Crafted Classifier
Training

Split Train and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    df[['h5','maxMagOnArea']], df[['Target']], test_size=0.33, random_state=seed)
```

Hand-Crafted Classifier

Training

Grid Search to find good Hyperparameters

```
rf = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=0)

parameters = {'max_depth':[1,2,3,4,5],}
gsCV = GridSearchCV(rf, parameters, cv=100, n_jobs=-1)
gsCV = gsCV.fit(X_train, y_train)
```

Found best hyperparameter

```
gsCV.best_params_
```

```
{'max_depth': 4}
```

Hand-Crafted Classifier
Training

CV score on training set

```
rf = RandomForestClassifier(n_estimators=1000, max_depth=4, random_state=seed)
scores = cross_val_score(rf, X_train, y_train, n_jobs=-1, cv=100)
np.mean(scores)
```

```
0.7483919413919415
```

Final score on test set

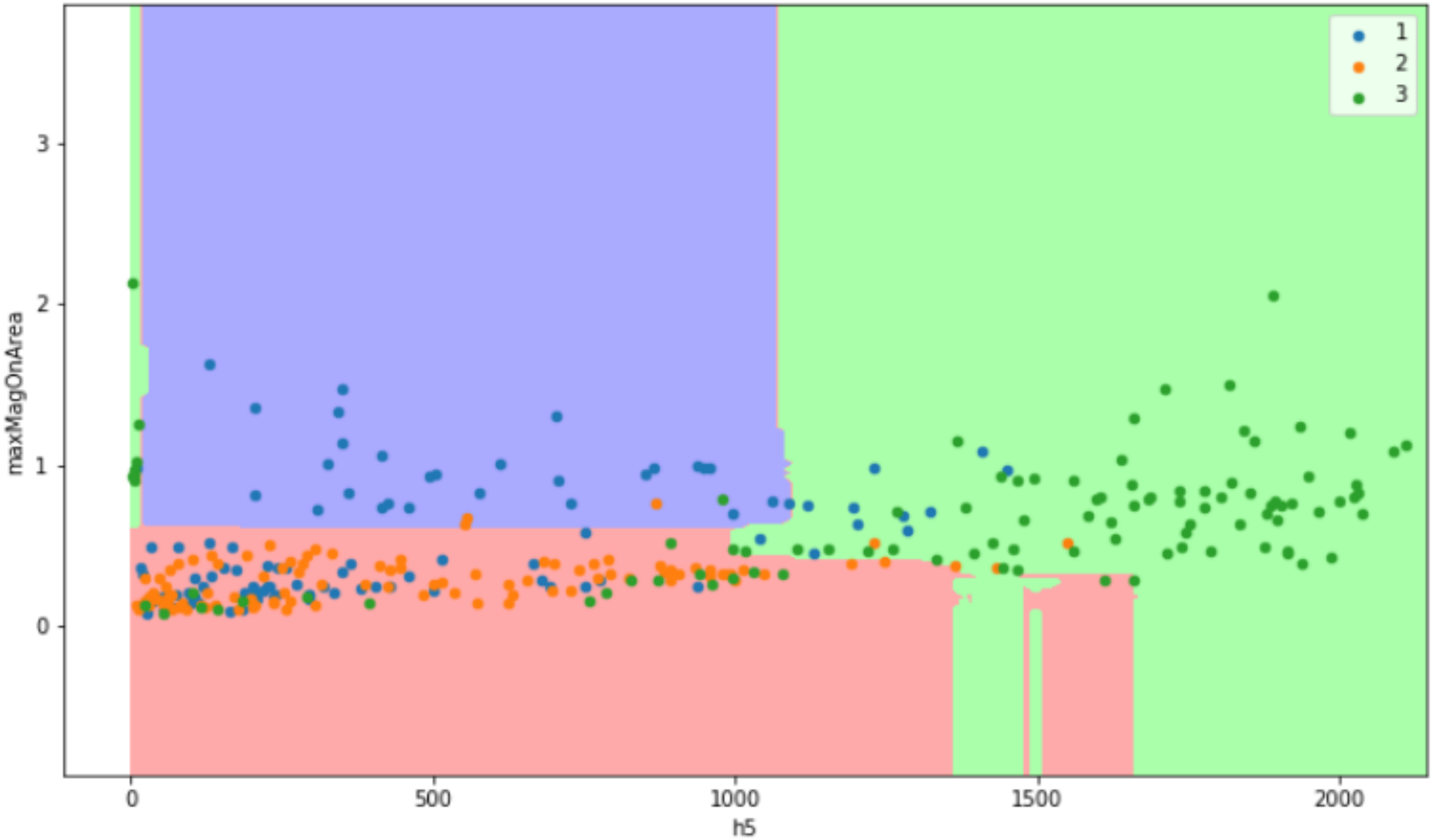
```
rf.score(X_test,y_test)
```

```
0.7330383480825958
```

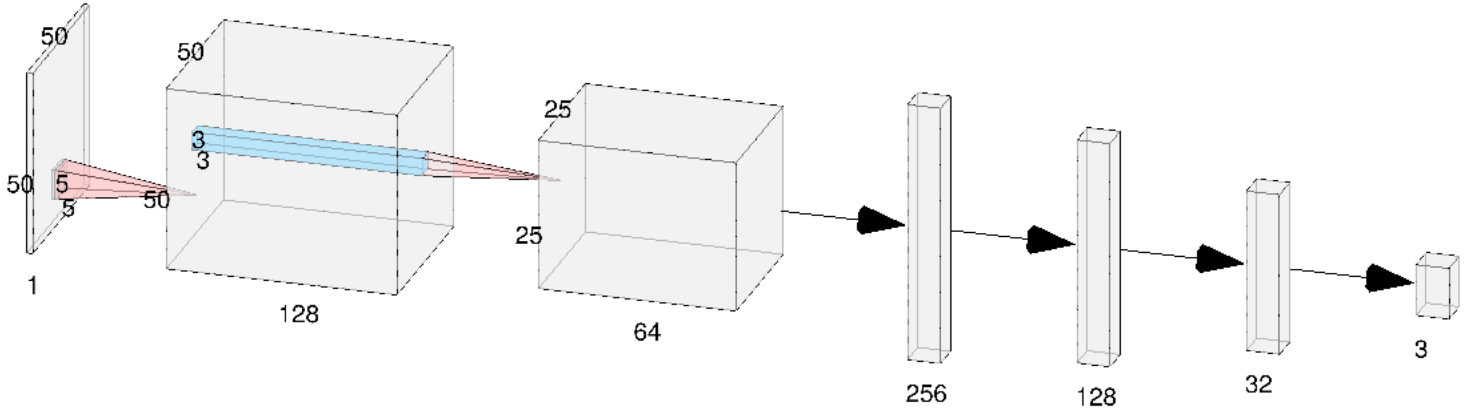
Accuracy $\approx$ 74%

Hand-Crafted Classifier

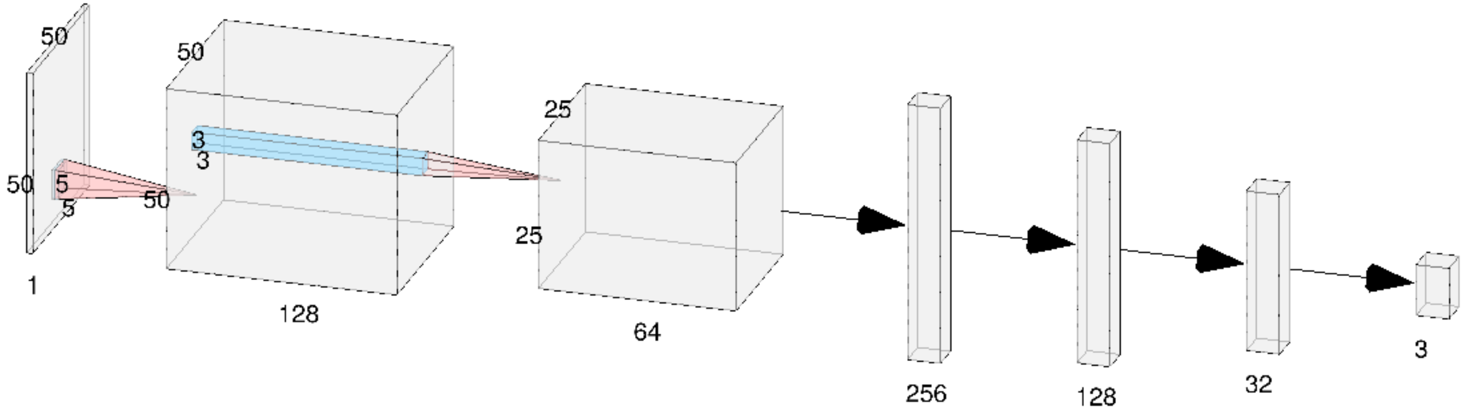
Explainability



CNN
Architecture



CNN
Architecture



- 2 Conv2D layers and 4 dense
- Regularized with L1,L2 and dropout.
- Relu activations except for the last two layers
- Adam optimizer
- Crossentropy loss

```
cnnInput = Input(shape=(50,50,1), name="input")

model = Conv2D(filters=128, kernel_size=(5,5), activation = 'relu',activity_regularizer=l1_l2(l1=0.00001, l2=0.00001))(
model = MaxPooling2D(pool_size = (2, 2))(model)
model = Dropout(rate = 0.5)(model)

model = Conv2D(filters=64, kernel_size=(3,3), activation = 'relu',activity_regularizer=l1_l2(l1=0.00001, l2=0.00001))(c
model = MaxPooling2D(pool_size = (2, 2))(model)
model = Dropout(rate = 0.5)(model)

model = Flatten()(model)
model = Dense(units = 256, activation = 'relu',activity_regularizer=l1_l2(l1=0.00001, l2=0.00001))(model)
model = Dropout(rate = 0.5)(model)
model = Dense(units = 128, activation = 'relu',activity_regularizer=l1_l2(l1=0.00001, l2=0.00001))(model)
model = Dropout(rate = 0.5)(model)

model = Dense(units = 32, activation = 'sigmoid',activity_regularizer=l1_l2(l1=0.00001, l2=0.00001))(model)

preds = Dense(units = 3, activation = 'softmax', name='output')(model)

model = Model(inputs=cnnInput, outputs=preds)

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

CNN
Train-Dev-Test split

As before one third of the dataset is kept as Test dataset.

```
X_train, X_eval, y_train, y_eval = train_test_split(
    df.drop(2500, axis=1, inplace=False).values.reshape(nImages,50,50,1), df.loc[:,2500], test_size=0.333, random_stat

X_train, X_test, y_train, y_test = train_test_split(
    X_train, y_train, test_size=0.2, random_state=seed)
```

One fifth of the remaining test datatset is kept to perform early stopping.

CNN
Training procedure

- Model trained until the validation loss do not improve for 30 epochs
- Batch size 128
- Best model saved in .h5 file

```
bestModelLoss = 9999.0;
sensitivity = 0.001
lastBestIter = 0;

for i in range(1,9999):
    hist = model.fit_generator(datagen.flow(X_train, y_train, batch_size=128), epochs=1, validation_data=(X_test, y_tes

    predictions = model.predict(X_eval)
    classes = np.argmax(predictions, axis=1)
    print(np.mean(classes == y_eval))

    valLoss = hist.history['val_loss'][0]
    if valLoss < bestModelLoss -  sensitivity:
        bestModelLoss = valLoss
        lastBestIter = i
        model.save('bestModel.h5')

    print(i - lastBestIter)
    if i > lastBestIter+30:
        print("Model didn't improve in the last 30 epochs.. break")
        break;

model = load_model('bestModel.h5')
```

CNN
Data augmentation

Data augmentation has been used to enhance the training dataset.

```
datagen = ImageDataGenerator(
    rotation_range=45,
    width_shift_range=0.0,
    height_shift_range=0.0,
    shear_range= 0,
    horizontal_flip=True,
    vertical_flip=True)

datagen.fit(X_train)
```

CNN
Test dataset results

The trained model have an accuracy of ≈ 85%

```
predictions = model.predict(X_eval)
classes = np.argmax(predictions, axis=1)
```

```
np.mean(classes == y_eval)
```

```
0.8552631578947368
```

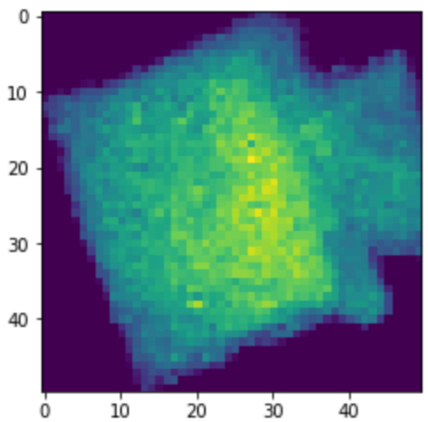
Errors evenly spread across classes.

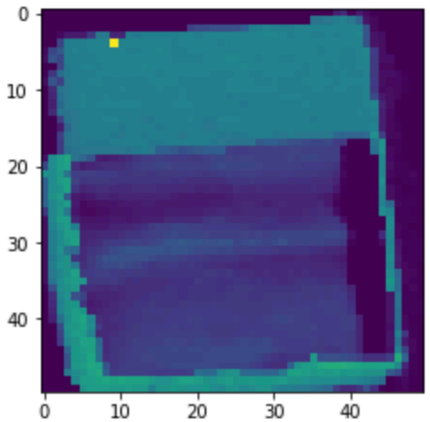
```
confusion_matrix(y_true=y_eval, y_pred=classes,labels=[0,1,2])
```

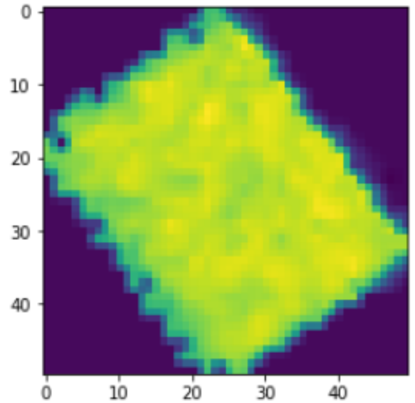
```
array([[121,  28,   9],
       [ 15, 275,   7],
       [ 11,  29, 189]])
```

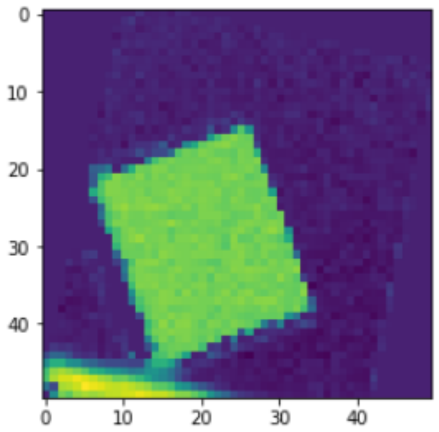
CNN

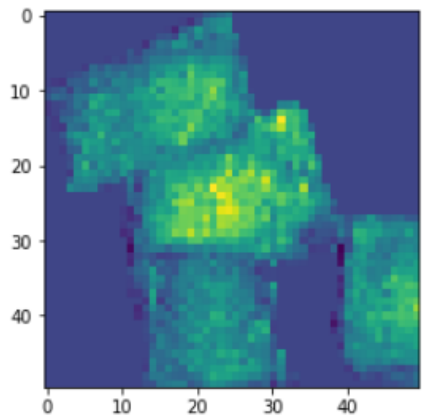
False negatives - Double

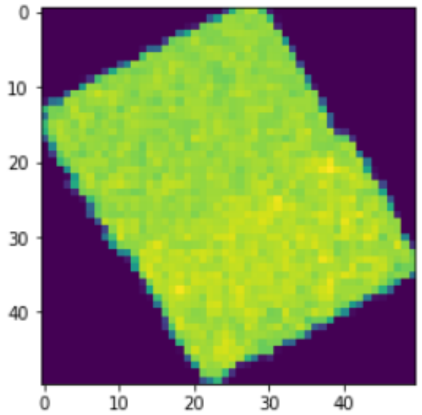




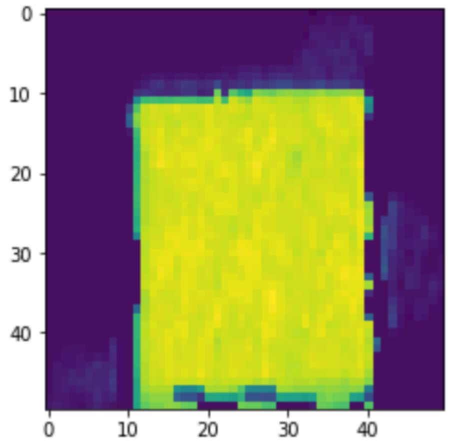


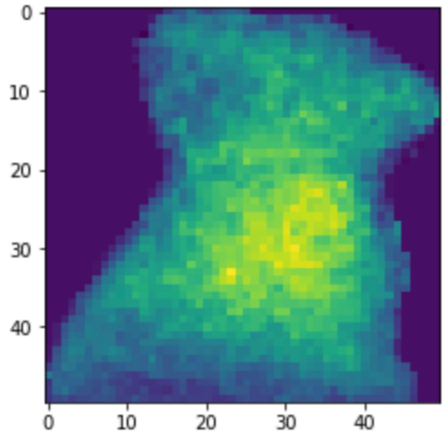


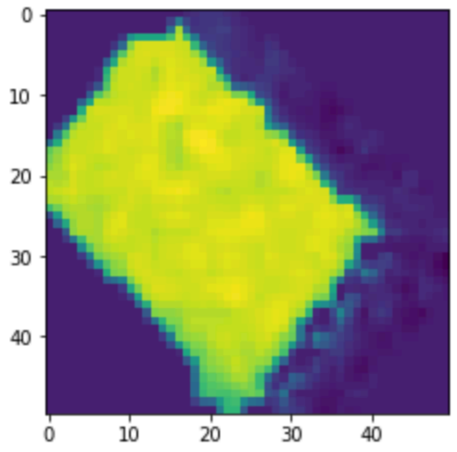


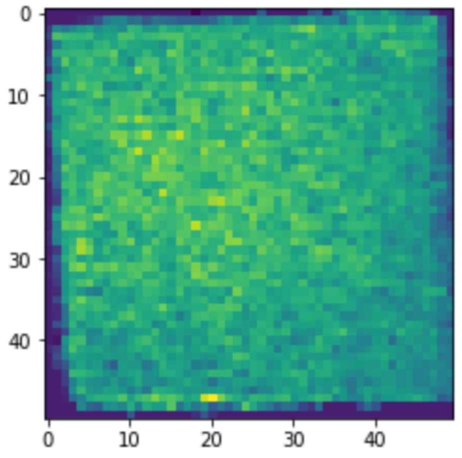


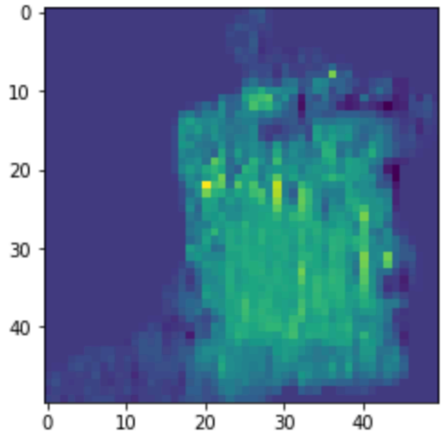
CNN
False positives - Double

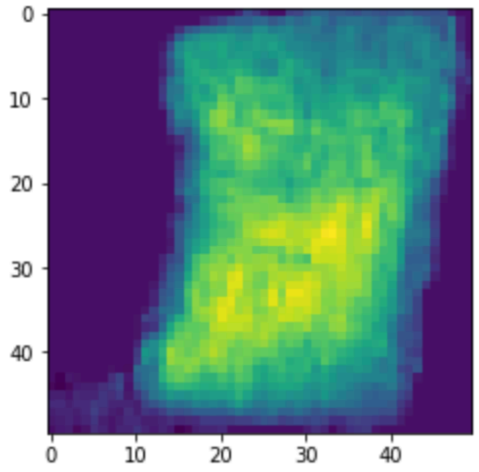






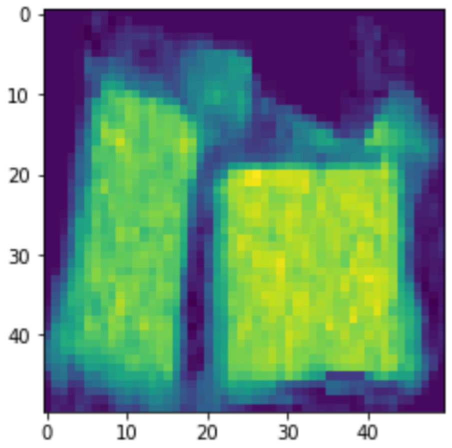


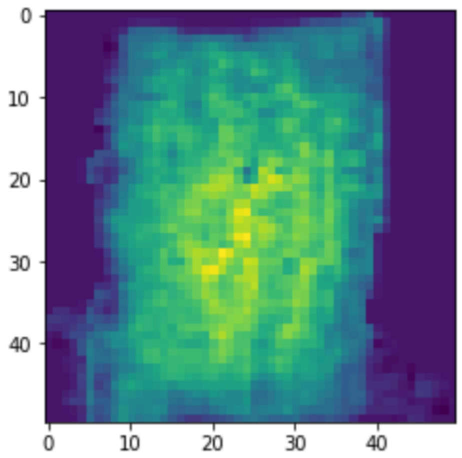


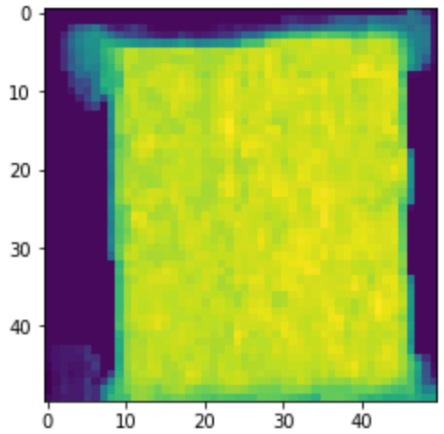


CNN

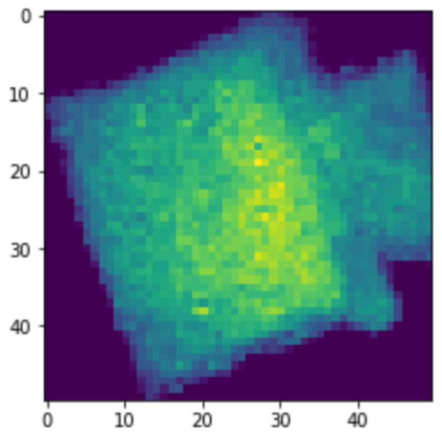
False negatives - Envelope

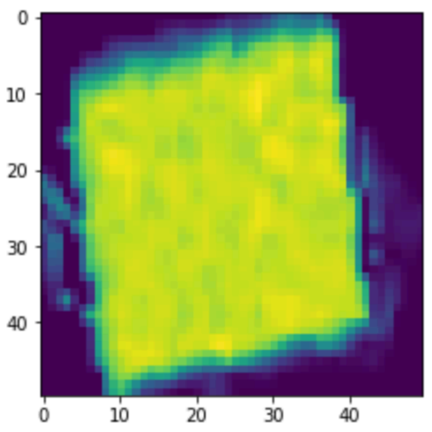


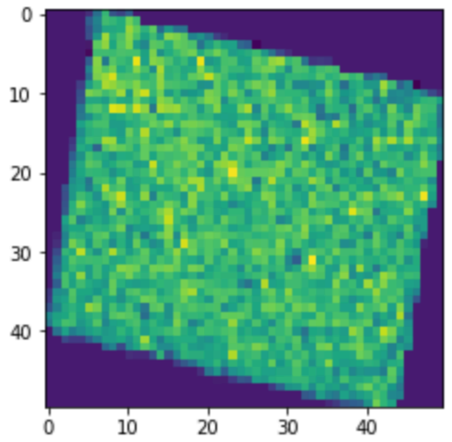




CNN
False positives - Envelope

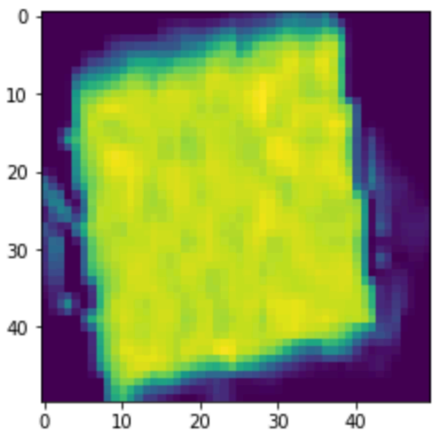


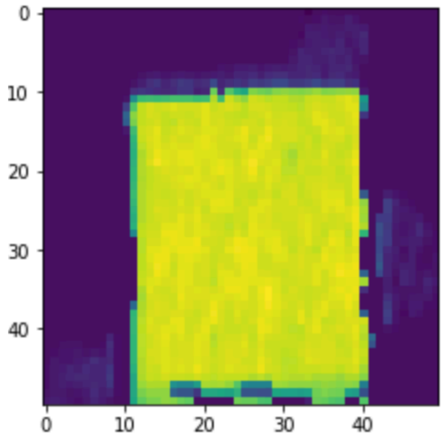


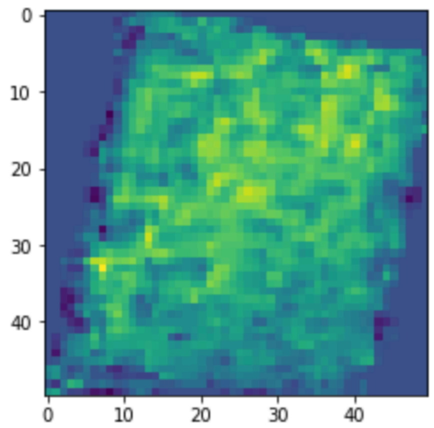


CNN

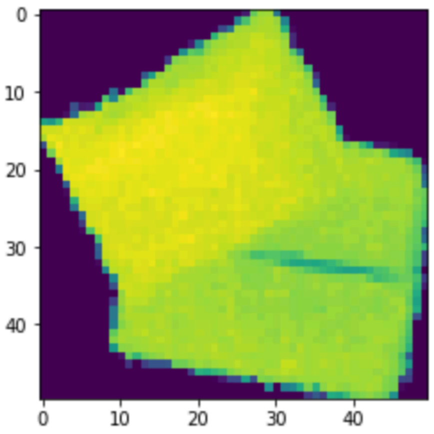
False negatives - Parcel

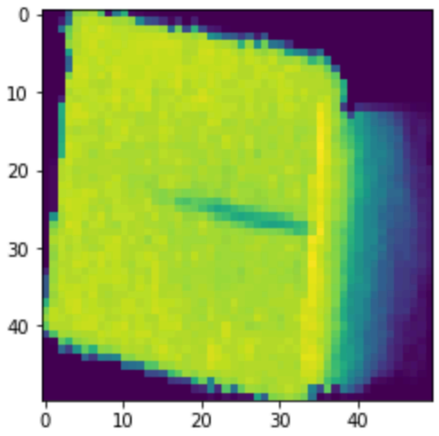


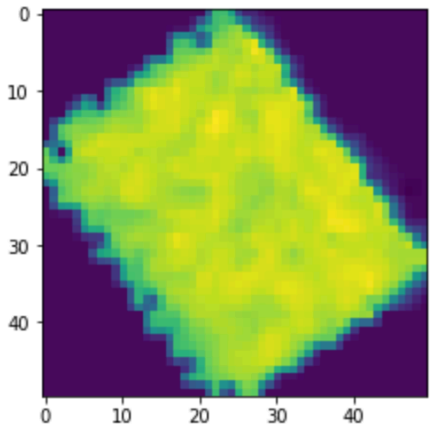




CNN
False positives - Parcel







Final thoughts

- CNN models lead to better models at the expense of the possibility to explain the results.
- Better hand-crafted features might improve the model (e.g. Clustering).
- More data points whould improve performances of CNN model.
- To deploy the final models we should retrain both models on the whole dataset without the test set, that would improve accuracy.