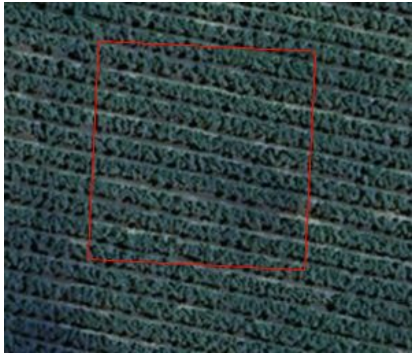
Land Cover Classification (one dimensional analysis)

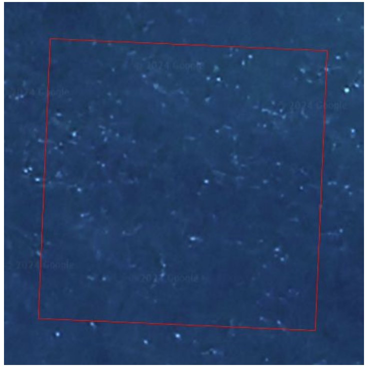
Ben Harris
Ian Swallow
Sam Hobbs
Collins Senaya

What is Land Cover Classification?

- Taking input images and determining what type of geological features are shown
  - Each image consists of pixels that contain 432 “bands” of information



These are “Permanent Crops”



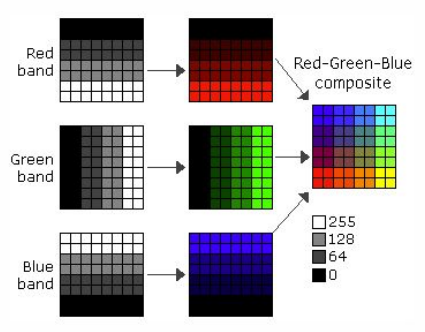
This is a “Waterbody”

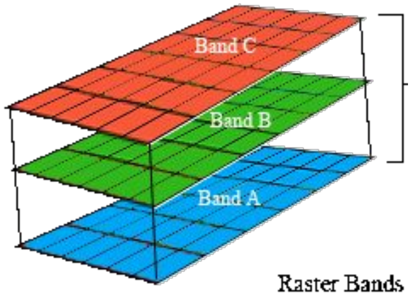
This is “built-up”

etc…

What is a “band” of data?

- Each band represents a particular characteristic of a pixel
  - Each band is a different infrared frequency that acts differently when they collide with different materials like rocks, water, and tree canopies
- Each pixel has 432 bands
  - Put together, that’s A LOT of data! Millions upon millions of data points to consider
    - Our .csv had 115 million+ cells…





Why do we care?

- It's useful information!
  - Gives insight to regions without having a human needing to label everything
  - Can track trends in landscape
- Broad applications
  - Cuts down on Labor & Time for classification
  - Can be repurposed for other classifications projects.
- BioSCape Project
  - NASA backed Biodiversity research of the South Cape of Africa

BIOSCAPE

Machine Learning

- Convolutional Neural Network (CNN)
    - Learns trends from previous data, and performs predictions on new input data based on those trends
- PCA Analysis
    - Reduces the dimensionality of the data
        - In this case, reduce the usage of 373 bands down to the most important bands
        - Useful for KNN and Logistic Regression
            - K-Nearest-Neighbors (KNN)
                - A method of classification where each point is categorized based on similar characteristics to other data points
            - Logistic Regression (LR)
                - Predicts class probabilities using a curve that maps values between 0 and 1.
- Linear Discriminant Analysis (LDA)
    - Reducing the feature separability between classes

Pre-Processing

- ~2,200 images to train and test on
  - We labelled each one a minimum of three times to get a majority vote on classification
- Create a CSV file so the information is usable
  - Rasterio library
- Filtering of data
  - Not all data given from the images is useable (Noisy, Flight patterns not in images)
- Split data
  - Set aside some of the data to test our models, ensuring they are predicting accurately
  - Splitting data on an Image-level

The Data

```
with rasterio.open(join(path_samples_1, '1_ang20231028t101421_014_L2A_OE_
    print("resolution: ", ds.res)
    print("Shape: ", ds.shape)
    ds = ds.read()
    print("Array Shape: ", ds.shape)
```

```
resolution:  (4.9, 4.9)
Shape:  (10, 10)
Array Shape:  (373, 10, 10)
```

```
ds[:2]
```

```
array([[[0.01890998, 0.01346918, 0.01348204, 0.01305762, 0.01523122,
         0.01332604, 0.01514322, 0.01834574, 0.02253461, 0.01906751],
        [0.01368855, 0.01443155, 0.01409643, 0.01305762, 0.01382202,
         0.01505167, 0.01472478, 0.02301237, 0.02253461, 0.01505403],
        [0.01236837, 0.01443155, 0.01614338, 0.01670057, 0.01455318,
         0.01783718, 0.02160607, 0.01760162, 0.01775908, 0.02026838],
        [0.01362917, 0.01361732, 0.01668102, 0.01670057, 0.01703556,
         0.01783718, 0.0172121 , 0.02297779, 0.0228613 , 0.01885407],
        [0.01463799, 0.01478598, 0.01668102, 0.01939409, 0.0157516 ,
         0.01670882, 0.0236931 , 0.02297779, 0.01963201, 0.01885407],
        [0.01463799, 0.01867374, 0.01362365, 0.01178232, 0.01936085,
         0.02315897, 0.01927681, 0.02260808, 0.02180685, 0.02403733],
        [0.01814001, 0.01284469, 0.01362365, 0.01917251, 0.01936085,
         0.01768679, 0.02118265, 0.02131751, 0.02081405, 0.02403733],
        [0.01167966, 0.01284469, 0.01994975, 0.01380918, 0.01937063,
         0.02057219, 0.02118265, 0.02184269, 0.01808047, 0.02090366],
        [0.01900949, 0.01603037, 0.01310998, 0.0180847 , 0.01823605,
         0.02115246, 0.01919487, 0.01851987, 0.03319117, 0.03277779],
        [0.01561393, 0.01603037, 0.01906566, 0.0180847 , 0.01972781,
         0.02115144, 0.01761609, 0.03469272, 0.03319117, 0.01112588]],

       [[0.02545341, 0.01544228, 0.01315289, 0.02426513, 0.02481231,
         0.01259124, 0.016026  , 0.02379542, 0.02572341, 0.02041227],
        [0.01314393, 0.02362236, 0.02386183, 0.02426513, 0.01412701,
         0.0188639 , 0.02014118, 0.02294535, 0.02572341, 0.01742779],
        [0.01366109, 0.02362236, 0.01425563, 0.02078871, 0.01966477,
         0.02169595, 0.02104024, 0.0201913 , 0.02098918, 0.0252691 ],
        [0.01401854, 0.01519072, 0.02062858, 0.02078871, 0.02069796,
         0.02169595, 0.01877324, 0.02584995, 0.02414699, 0.02508363],
        [0.01997662, 0.01848179, 0.02062858, 0.02275201, 0.01975328,
         0.01931756, 0.02566899, 0.02584995, 0.02519147, 0.02508363],
        [0.01997662, 0.02065543, 0.01635433, 0.01752375, 0.02255032,
         0.02576256, 0.02576471, 0.02615343, 0.0252151 , 0.02634802],
        [0.02006628, 0.01995645, 0.01635433, 0.02071472, 0.02255032,
         0.02199314, 0.02280192, 0.02158698, 0.02769558, 0.02634802],
        [0.02018143, 0.01995645, 0.02136969, 0.02076253, 0.02532449,
         0.02164077, 0.02280192, 0.02558043, 0.0249496 , 0.02617808],
        [0.02235128, 0.01876087, 0.02072179, 0.01765709, 0.01896966,
         0.02741962, 0.02632417, 0.02493823, 0.03072576, 0.02804623],
        [0.02054779, 0.01876087, 0.01931456, 0.01765709, 0.02597334,
         0.02437998, 0.02519206, 0.03100185, 0.03072576, 0.02500539]]],
      dtype=float32)
```

```
ds[:,0,0]
```

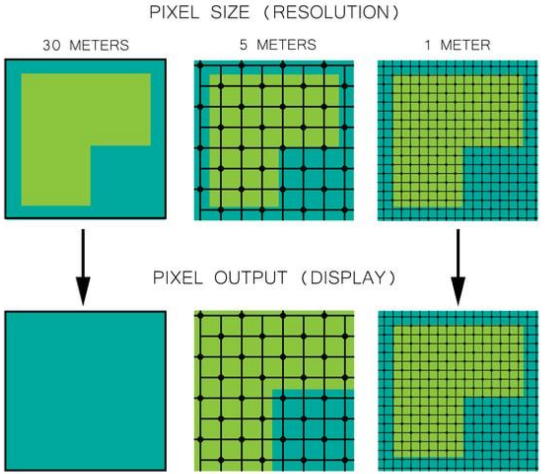
```
array([0.01890998, 0.02545341, 0.02680097, 0.02993097, 0.02599587,
       0.03024603, 0.0335446 , 0.03533418, 0.03959136, 0.04471725,
       0.0455179 , 0.04883455, 0.05512206, 0.05713886, 0.05850134,
       0.0604089 , 0.06354472, 0.06456087, 0.0672814 , 0.06948043,
       0.0717717 , 0.0741334 , 0.07750069, 0.08012719, 0.08448848,
       0.08652212, 0.0882066 , 0.09174882, 0.09516714, 0.09985641,
       0.10164141, 0.1064215 , 0.10933152, 0.11346705, 0.1165515 ,
       0.12054856, 0.12479144, 0.12961487, 0.13030073, 0.13482675,
       0.13788415, 0.14160955, 0.14314076, 0.14686692, 0.149538  ,
       0.15258297, 0.15248564, 0.15502334, 0.15808015, 0.15862162,
       0.15969668, 0.16181406, 0.16407768, 0.1657265 , 0.16629   ,
       0.16814084, 0.16890872, 0.17019424, 0.17379284, 0.17333238,
       0.17406647, 0.17814532, 0.17953834, 0.18176067, 0.18272781,
       0.1834855 , 0.18612902, 0.18751615, 0.18906523, 0.19024874,
       0.1917163 , 0.19001693, 0.19160187, 0.19335295, 0.19413543,
       0.19483502, 0.19755726, 0.19867922, 0.19996437, 0.20099814,
       0.20214118, 0.20263393, 0.20330463, 0.20369096, 0.20404252,
       0.20452131, 0.20437716, 0.20497514, 0.20501728, 0.2052141 ,
       0.20488924, 0.20498465, 0.2050034 , 0.20501372, 0.2047943 ,
       0.20518495, 0.20511875, 0.20532925, 0.20510574, 0.20527156,
       0.20527452, 0.20565525, 0.20557435, 0.20582004, 0.20577618,
       0.2062191 , 0.20651065, 0.20694658, 0.20650375, 0.20624648,
       0.2068108 , 0.20744722, 0.20799698, 0.20891519, 0.20935382,
       0.20962028, 0.21017186, 0.21028551, 0.21049055, 0.21062678,
       0.21094926, 0.21114174, 0.21162863, 0.21176453, 0.2125633 ,
       0.21277024, 0.2134506 , 0.21362649, 0.21375926, 0.21413802,
       0.21470168, 0.21510811, 0.2150898 , 0.21531324, 0.21576542,
       0.21603495, 0.21641973, 0.21692301, 0.2172111 , 0.21749038,
       0.2177547 , 0.21821797, 0.21819209, 0.21850899, 0.2186227 ,
       0.21848688, 0.21777861, 0.21821775, 0.21926801, 0.21992302,
       0.2202801 , 0.2202776 , 0.22137943, 0.22186528, 0.22224756,
       0.22223805, 0.22211303, 0.22197753, 0.22224544, 0.22242865,
       0.22225359, 0.22246292, 0.22227718, 0.22248977, 0.22217195,
       0.22230974, 0.22255547, 0.2225306 , 0.22288856, 0.22275372,
       0.22303213, 0.2230684 , 0.2233477 , 0.22342639, 0.22353706,
       0.22362292, 0.22381166, 0.22388117, 0.2239641 , 0.22406025,
       0.22409841, 0.22428608, 0.22438212, 0.22447124, 0.22458172,
       0.22460735, 0.22456993, 0.22964764, 0.23076896, 0.23032713,
       0.23101993, 0.22950476, 0.21936573, 0.22100355, 0.2047616 ,
       0.19834167, 0.20658147, 0.20407388, 0.207656  , 0.21409866,
       0.21480392, 0.21861537, 0.22009368, 0.21959104, 0.23305155,
       0.22419012, 0.23147507, 0.23374437, 0.23041707, 0.22923528,
       0.22707179, 0.22886106, 0.22985736, 0.23002939, 0.23271665,
       0.23123933, 0.23082343, 0.23149094, 0.23168851, 0.23358727,
       0.23312116, 0.23360333, 0.23524658, 0.23577288, 0.23448013,
       0.23503126, 0.2359351 , 0.23588014, 0.2367796 , 0.23566782,
       0.2357772 , 0.23576581, 0.23595645, 0.23696153, 0.23533691,
       0.2367774 , 0.2381562 , 0.23607953, 0.23793504, 0.23774202,
       0.23425336, 0.23646721, 0.23834394, 0.23774333, 0.23462862,
       0.23439829, 0.23416257, 0.23730813, 0.23561446, 0.2348301 ,
       0.23585773, 0.23580278, 0.23565786, 0.23507704, 0.23408103,
       0.23622216, 0.23347102, 0.23574919, 0.23736489, 0.23834434,
       0.23525819, 0.23479976, 0.23572464, 0.23516756, 0.23522907,
       0.22805622, 0.23136322, 0.23146066, 0.2286876 , 0.22891048,
       0.20746154, 0.2066988 , 0.20556666, 0.20252727, 0.19813007,
       0.19873855, 0.19984524, 0.2012015 , 0.2053631 , 0.20763178,
       0.20808567, 0.20837338, 0.20904459, 0.20970081, 0.2097946 ,
       0.2096821 , 0.21029104, 0.21023534, 0.20811655, 0.2085872 ,
```

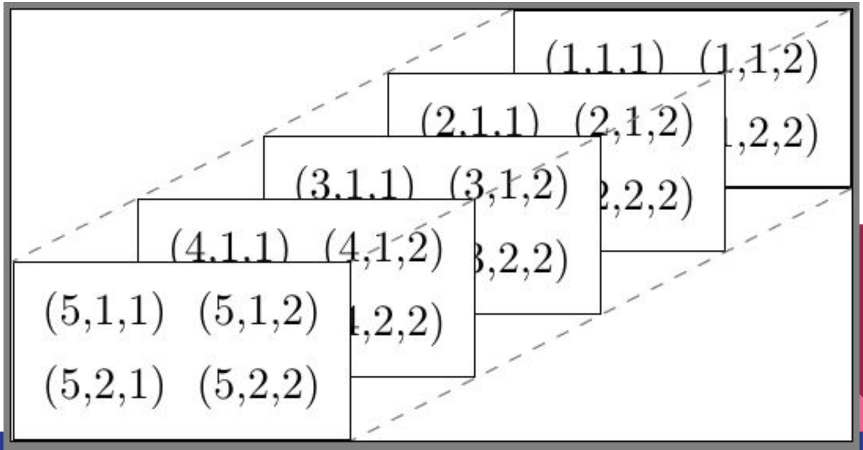
```
ds[:2].shape
```

```
(2, 10, 10)
```

```
ds[:,0,0].shape
```

```
(373,)
```





Developing a CSV

	img_pxl_index	frq0	frq372	Label	Shape	File_UID_Num	File	img_pos
0	0	0.018910	0.190160	Unconsolidated Barren	(10, 10)	1	1_ang20231028t101421_014_L2A_OE_main_27577724_...	(0, 0)
1	1	0.013469	0.169750	Unconsolidated Barren	(10, 10)	1	1_ang20231028t101421_014_L2A_OE_main_27577724_...	(0, 1)
2	2	0.013482	0.167964	Unconsolidated Barren	(10, 10)	1	1_ang20231028t101421_014_L2A_OE_main_27577724_...	(0, 2)
3	3	0.013058	0.165022	Unconsolidated Barren	(10, 10)	1	1_ang20231028t101421_014_L2A_OE_main_27577724_...	(0, 3)
4	4	0.015231	0.178857	Unconsolidated Barren	(10, 10)	1	1_ang20231028t101421_014_L2A_OE_main_27577724_...	(0, 4)
...	...	...	...	...	...	...	...	...
398279	59	0.002211	0.069898	Natural Wooded Land	(8, 8)	4177	28499_ang20231109t071216_015_L2A_OE_main_27577...	(7, 3)
398280	60	0.004726	0.069710	Natural Wooded Land	(8, 8)	4177	28499_ang20231109t071216_015_L2A_OE_main_27577...	(7, 4)
398281	61	0.004768	0.078389	Natural Wooded Land	(8, 8)	4177	28499_ang20231109t071216_015_L2A_OE_main_27577...	(7, 5)
398282	62	0.002905	0.072073	Natural Wooded Land	(8, 8)	4177	28499_ang20231109t071216_015_L2A_OE_main_27577...	(7, 6)
398283	63	0.009146	0.044159	Natural Wooded Land	(8, 8)	4177	28499_ang20231109t071216_015_L2A_OE_main_27577...	(7, 7)

398284 rows × 379 columns

```
def make_pandas_dataframe(dir_path, filename, col_labels, label=pd.NA, uid = 0, min_res=4.5, max_res=6.5):
  '''
  Description:
    Converts a tiff file to a pandas dataframe where each row is a pixel of the
    tiff image and every column is the frequency bands of that pixel along with
    various meta data like pixel location, label of image, shape of image, and
    filename. The images are also filtered for acceptable resolution range.
  Input:
    dir_path        : String; This is the directory path to where the .tiff
                      files are stored. (ex: '/content/drive/.../files/')
    filename        : String; This is the name of the .tiff file you want to
                      convert to a pandas dataframe.
    col_labels      : List; A list of strings that are the names of the columns
                      of the bands in the .tiff file.(ex: ['frq1', ..., 'frqN'])
    label           : String; The label of the image of the tiff, all pixels
                      will be assigned this label. Defaults to NaN.
    uid             : Int; This is an integer value that is suposed to be used
                      as an alternative unique identifier for the individual
                      tiff files, that is not the string filename. Defaults to 0
                      if not provided.
    min_res         : Float; The minimum accepted resolution of a pixel in the
                      tiff file. Inclusive. Defaults to 4.5
    max_res         : Float; The maximum accepted resolution of a pixel in the
                      tiff file. Inclusive. Defaults to 6.5
  Output:
    Pandas DataFrame: This is the pandas dataframe of the tiff file provided or
                      None if the .tiff file fails the resolution check of the
                      pixels from the min_res and max_res provided.
    Bool            : Boolean that represents if the tiff file is of appropriate
                      resolution. Dependent on the provided min_res and max_res
                      provided.
  '''
  arr, res_check, shape = tiff_to_arr(join(dir_path, filename), min_res=min_res, max_res=max_res)
  #if filename is in label data for labeling.
  if (res_check):
    ds = convert_3D_to_1D(arr)
    df = pd.DataFrame(ds, columns=col_labels)
    df['Label'] = label
    df['Shape'] = shape
    df['File_UID_Num'] = uid
    df['File'] = filename
    return df, True
  return None, False #None could be arr here but for readability it is None. Also to make sure nothing weird
```

samples.csv	Dec 2, 2024	Samuel Hobbs	1.5 GB

Pre-Processing

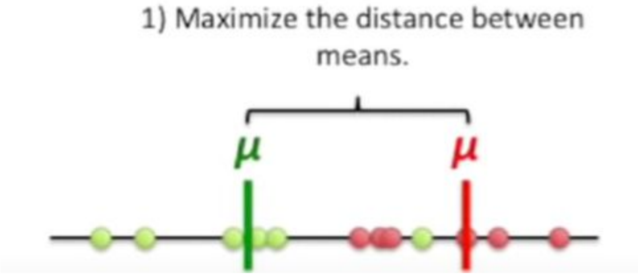
```python
def preprocess_data(samples_df, labels_df):
  # Extract sample number
  samples_df['Sample_num'] = samples_df['File'].str.split('_').str[0].astype(int)

  # Clean up labels in both DataFrames (removing the extras in the names (e.g wheat)...)
  labels_df['Class'] = labels_df['Class'].str.split('(').str[0].str.strip()
  samples_df['Label'] = samples_df['Label'].str.split('(').str[0].str.strip()

  # Remove rows with "Mixed or Not Classified" (Assuming this was the plan)
  samples_df = samples_df[samples_df['Label'] != 'Mixed or Not Classified']
  labels_df = labels_df[labels_df['Class'] != 'Mixed or Not Classified']

  # Filter labels_df to include only Sample_num values present in samples_df
  labels_df = labels_df[labels_df['Sample_num'].isin(samples_df['Sample_num'])]

  # Reset the index to be consecutive after removing "Mixed or Not Classified"
  samples_df.reset_index(drop=True, inplace=True)
  labels_df.reset_index(drop=True, inplace=True)

  # Make syre of the unique labels after cleaning
  assert set(samples_df['Label'].unique()) == set(labels_df['Class'].unique()), "Mismatch in unique labels between samples_df and labels_df"

  # Define frequency columns for targeted NaN replacement
  frequency_columns = [col for col in samples_df.columns if col.startswith('frq')]

  # Make all those found with NaN's to be changed to -9999.000..
  samples_df.loc[:, frequency_columns] = samples_df[frequency_columns].fillna(-9999)
```

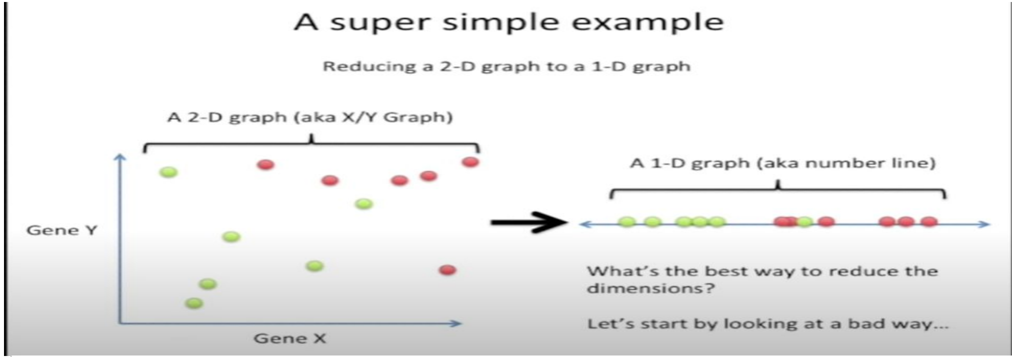
```python
  # Filter out rows where all values in frequency columns are -9999
  filtered_samples_df = samples_df[~samples_df[frequency_columns].eq(-9999).all(axis=1)]

  # Make sure labels_df are aligned properly by keeping only matching Sample_num values
  filtered_samples_df = filtered_samples_df[filtered_samples_df['Sample_num'].isin(labels_df['Sample_num'])]

  # Filter out rows where all values in frequency columns are -9999
  filtered_samples_df = samples_df[~samples_df.filter(like='frq').eq(-9999).any(axis=1)]

  # Make sure filtered samples_df and labels_df are matching up by keeping only matching Sample_num
  filtered_samples_df = filtered_samples_df[filtered_samples_df['Sample_num'].isin(labels_df['Sample_num'])]

  # Check that all NaNs in frequency columns have been replaced
  nan_counts = filtered_samples_df[frequency_columns].isna().sum().sum()
  assert(nan_counts == 0)  # Should be 0 if all NaNs were replaced

  # Check for -9999 in the dataframe
  count_negative_9999 = (filtered_samples_df == -9999).sum().sum()
  assert(count_negative_9999 == 0)

  # Make a copy of samples_df to avoid Warnings after filtering
  samples_df = samples_df.copy()

  # Label Encoding for consistency
  label_encoder = LabelEncoder()
  filtered_samples_df['Label_Encoded'] = label_encoder.fit_transform(filtered_samples_df['Label'])

  return filtered_samples_df, labels_df, label_encoder
```

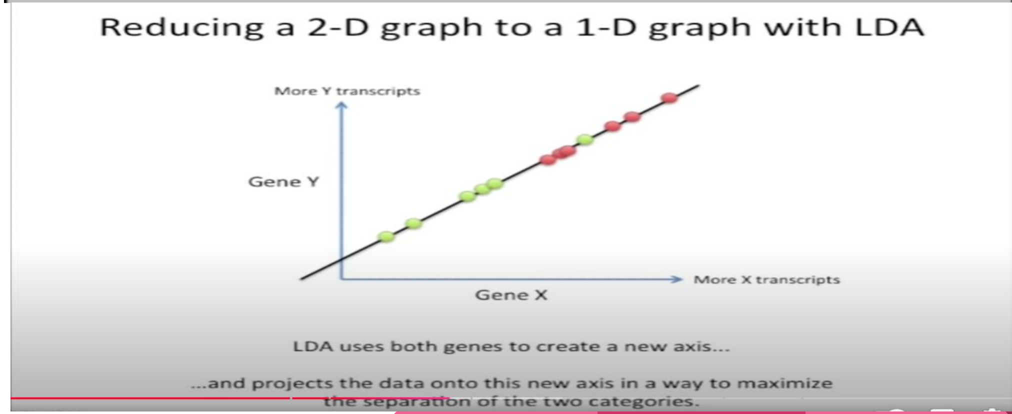
Improvements

- Further filtering of data
  - Removing elements that were noisy
  - Re-formatting data
- Grid-Search
  - A mechanized method to optimize parameters
- Random Forest
  - Improve class separability using advanced techniques like:
    - Balancing the dataset
    - Adding or transforming features
- Reformatting
  - Using the algorithms in different ways to achieve slightly different but more applicable results
    - This reduced our accuracy, but gave us more applicable information

Linear Discrimination Analysis (LDA)







LDA Matrix

Purpose: This matrix quantifies the performance of the classification model by showing the counts of correct and incorrect predictions for each class.

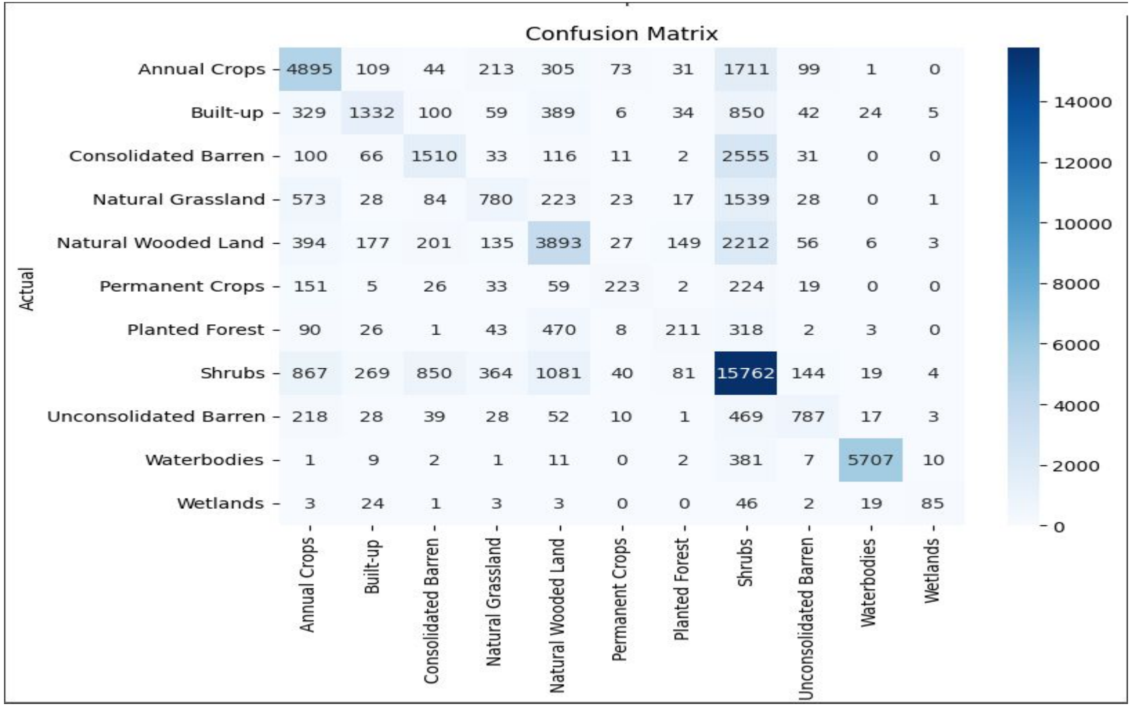
Rows: Represent the actual land cover classes.

Columns: Represent the predicted land cover classes.

Diagonal Cells: Indicate correct classifications (e.g., 4895 "Annual Crops" correctly classified).

Off-Diagonal Cells: Indicate misclassifications (e.g., 1711 "Annual Crops" misclassified as "Unconsolidated Barren").

High Misclassification: The large values in the "Shrubs" row (both correct and incorrect) and the corresponding "Unconsolidated Barren" column confirm the overlap observed in the scatter plot, indicating potential confusion between these classes.



LDA Matrix

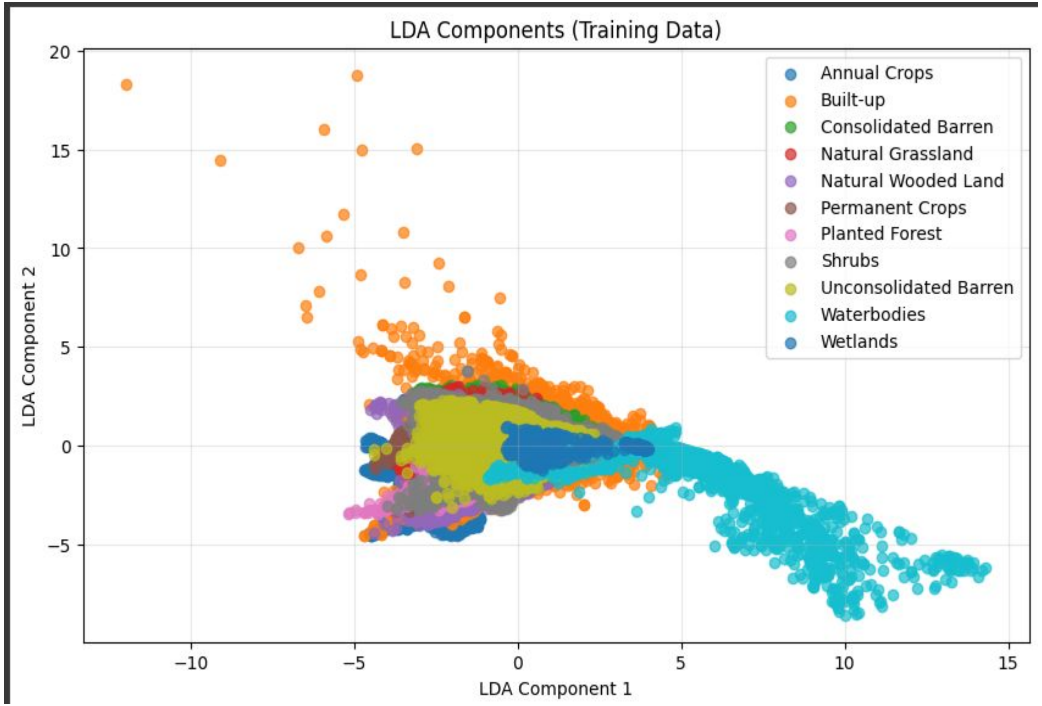
Scatter Plot (LDA Components):

Purpose: This plot visualizes how well LDA separates different land cover classes in a lower-dimensional space. Each point represents a data point (e.g., a pixel or region) from the training dataset.

Axes: The axes represent the two most significant LDA components. These components are linear combinations of the original input features (e.g., spectral bands from satellite imagery) chosen to maximize the separation between classes.

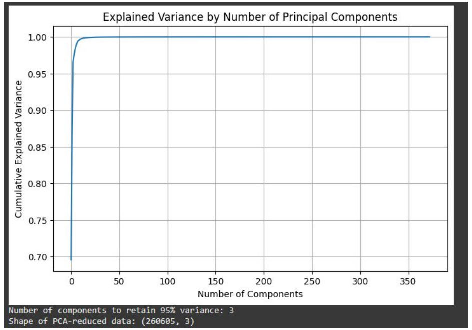
Clusters: Each color represents a different land cover class (e.g., "Waterbodies," "Shrubs"). Ideally, classes should form distinct, well-separated clusters.

Overlapping Clusters: Some overlap exists, particularly between "Shrubs" and "Unconsolidated Barren." This overlap suggests potential misclassification in those areas.



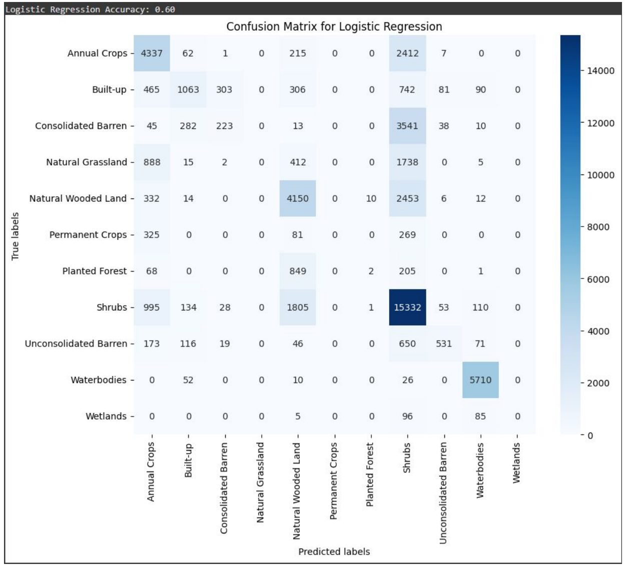
Primary Component Analysis (PCA)

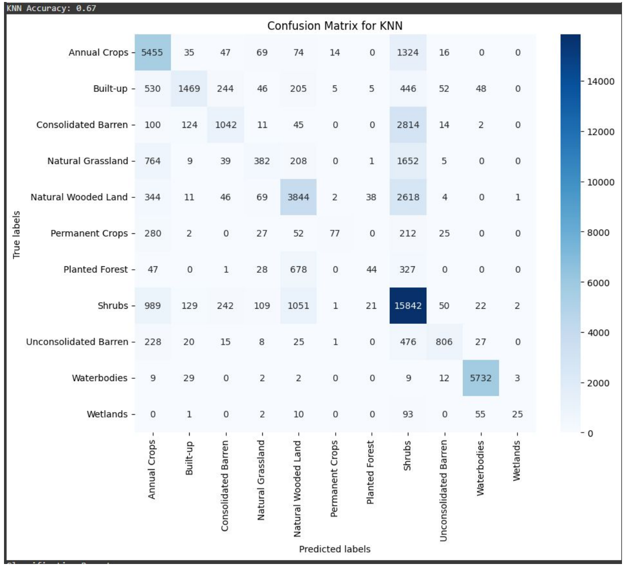
- Each image has 373 bands, way too much useless data
  - PCA reduces the dimensionality of the data, forming axes between bands based on variance
    - These axes are called “components”
- Reducing dimensionality allows us to analyze the components
  - KNN and Regression
  - SVM was initially attempted, but still took a long time with disappointing initial results
- In some cases, reducing the dimensionality actually improves accuracy
  - Noise reduction
  - Overfitting concerns



For Posterity:
https://www.youtube.com/watch?v=tU4Rm0Y_jQI

PCA Matrices





KNN CR

Classification Report:

	precision	recall	f1-score	support
Annual Crops	0.62	0.78	0.69	7034
Built-up	0.80	0.48	0.60	3050
Consolidated Barren	0.62	0.25	0.36	4152
Natural Grassland	0.51	0.12	0.20	3060
Natural Wooded Land	0.62	0.55	0.58	6977
Permanent Crops	0.77	0.11	0.20	675
Planted Forest	0.40	0.04	0.07	1125
Shrubs	0.61	0.86	0.72	18458
Unconsolidated Barren	0.82	0.50	0.62	1606
Waterbodies	0.97	0.99	0.98	5798
Wetlands	0.81	0.13	0.23	186
accuracy			0.67	52121
macro avg	0.69	0.44	0.48	52121
weighted avg	0.67	0.67	0.63	52121

CNN

Image-Level CM

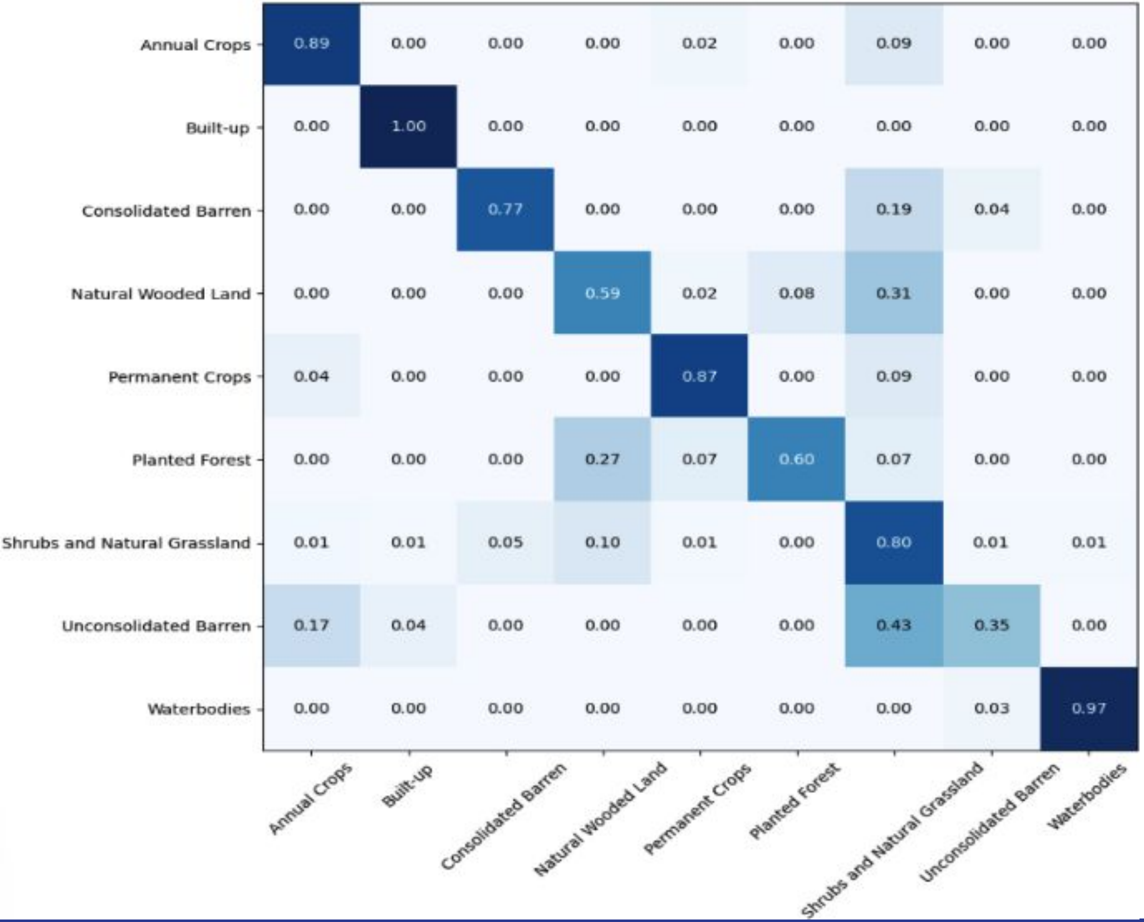
	Annual Crops	Built-up	Consolidated Barren	Natural Grassland	Natural Wooded Land	Permanent Crops	Planted Forest	Shrubs	Unconsolidated Barren	Waterbodies
Annual Crops	0.77	0.00	0.00	0.02	0.00	0.02	0.00	0.19	0.00	0.00
Built-up	0.04	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Consolidated Barren	0.00	0.00	0.67	0.00	0.00	0.00	0.00	0.33	0.00	0.00
Natural Grassland	0.29	0.00	0.00	0.18	0.00	0.00	0.00	0.47	0.06	0.00
Natural Wooded Land	0.00	0.00	0.02	0.00	0.78	0.00	0.05	0.16	0.00	0.00
Permanent Crops	0.07	0.00	0.00	0.00	0.00	0.89	0.00	0.04	0.00	0.00
Planted Forest	0.00	0.00	0.00	0.00	0.35	0.00	0.52	0.13	0.00	0.00
Shrubs	0.02	0.01	0.03	0.00	0.10	0.00	0.00	0.84	0.01	0.00
Unconsolidated Barren	0.06	0.00	0.11	0.00	0.00	0.06	0.00	0.11	0.67	0.00
Waterbodies	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.97

Pixel-Level Metrics

	precision	recall	f1-score
Annual Crops	0.71	0.71	0.71
Built-up	0.89	0.79	0.84
Cons. Barren	0.68	0.57	0.62
Natural Grassland	0.53	0.26	0.35
Natural Wooded Land	0.64	0.67	0.65
Permanent Crops	0.85	0.70	0.77
Planted Forest	0.62	0.44	0.51
Shrubs	0.63	0.79	0.70
Uncon. Barren	0.64	0.55	0.59
Waterbodies	1.00	0.92	0.96
accuracy	0.70		
macro avg	0.72	0.64	0.67
weighted avg	0.70	0.70	0.70

Test Acc: 70.58%

CNN (Combined Classes) Image-Level CM

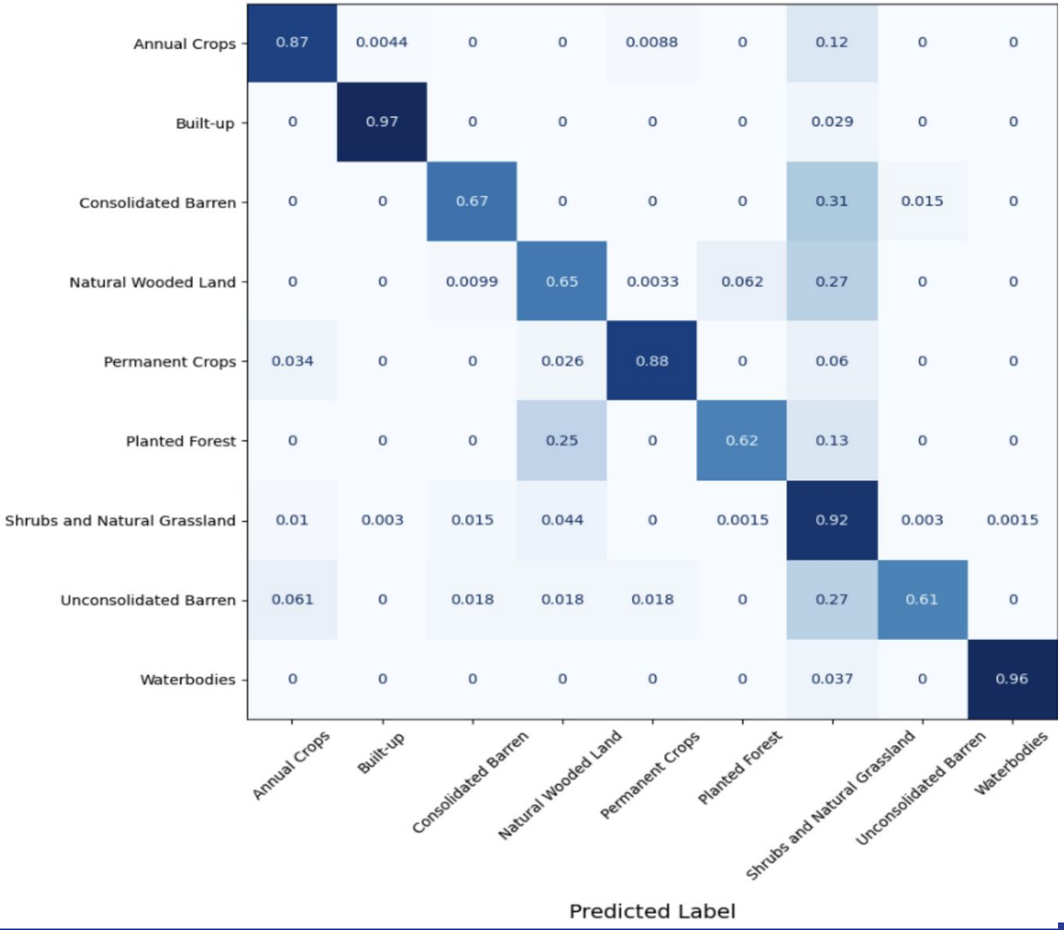


Pixel-Level Metrics

	precision	recall	f1-score
Annual Crops	0.80	0.71	0.75
Built-up	0.83	0.85	0.84
Cons. Barren	0.80	0.54	0.64
Wooded Land	0.64	0.56	0.60
Permanent Crops	0.81	0.75	0.78
Planted Forest	0.58	0.40	0.47
Shrubs & Grassland	0.67	0.85	0.75
Uncon. Barren	0.60	0.34	0.44
Waterbodies	0.96	0.91	0.94
accuracy	0.72		
macro avg	0.74	0.66	0.69
weighted avg	0.72	0.72	0.71

Test Acc: 71.69%

CNN (Combined Classes & Morphology) Image-Level CM



Pixel-Level Metrics

	precision	recall	f1-score
Annual Crops	0.91	0.85	0.88
Built-up	0.96	0.97	0.96
Cons. Barren	0.84	0.66	0.74
Wooded Land	0.78	0.65	0.71
Permanent Crops	0.95	0.88	0.91
Planted Forest	0.73	0.63	0.68
Shrubs & Grassland	0.76	0.92	0.83
Uncon. Barren	0.95	0.64	0.76
Waterbodies	0.99	0.96	0.98
accuracy	0.83		
macro avg	0.87	0.80	0.83
weighted avg	0.83	0.83	0.82

Test Acc: 80.65%

CNN

Overview for Image-Level Accuracy

Without combining Classes

Test Acc: 76.68%

	precision	recall	f1-score
Annual Crops	0.71	0.71	0.71
Built-up	0.89	0.79	0.84
Cons. Barren	0.68	0.57	0.62
Natural Grassland	0.53	0.26	0.35
Natural Wooded Land	0.64	0.67	0.65
Permanent Crops	0.85	0.70	0.77
Planted Forest	0.62	0.44	0.51
Shrubs	0.63	0.79	0.70
Uncon. Barren	0.64	0.55	0.59
Waterbodies	1.00	0.92	0.96
accuracy	0.70		
macro avg	0.72	0.64	0.67
weighted avg	0.70	0.70	0.70

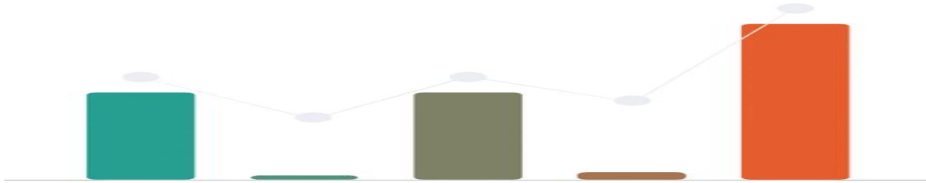
Combining Classes

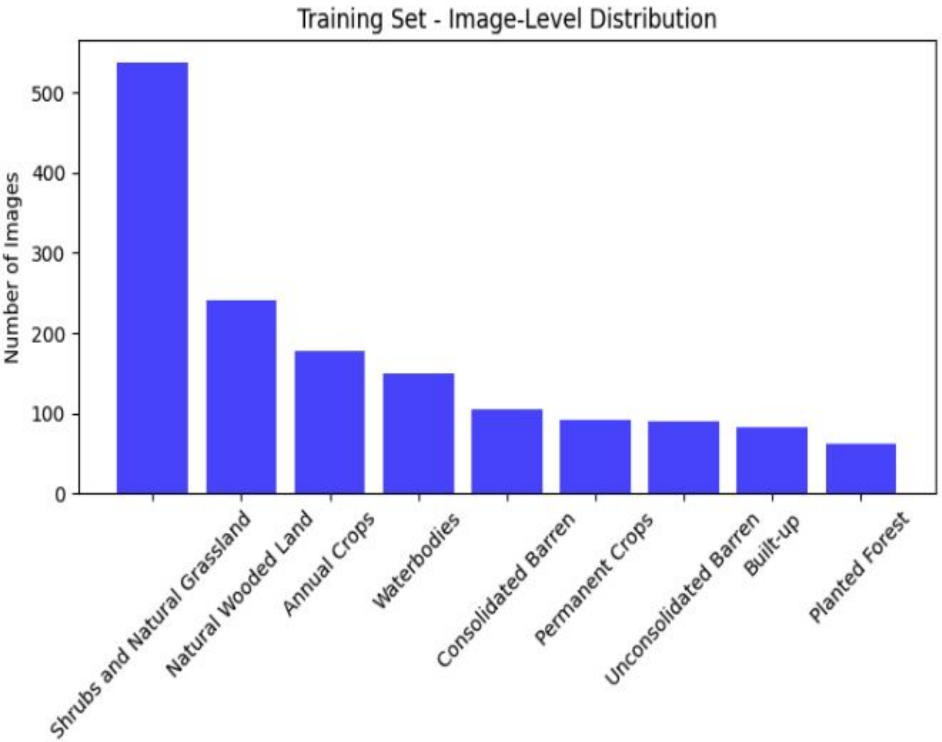
Test Acc: 82.65

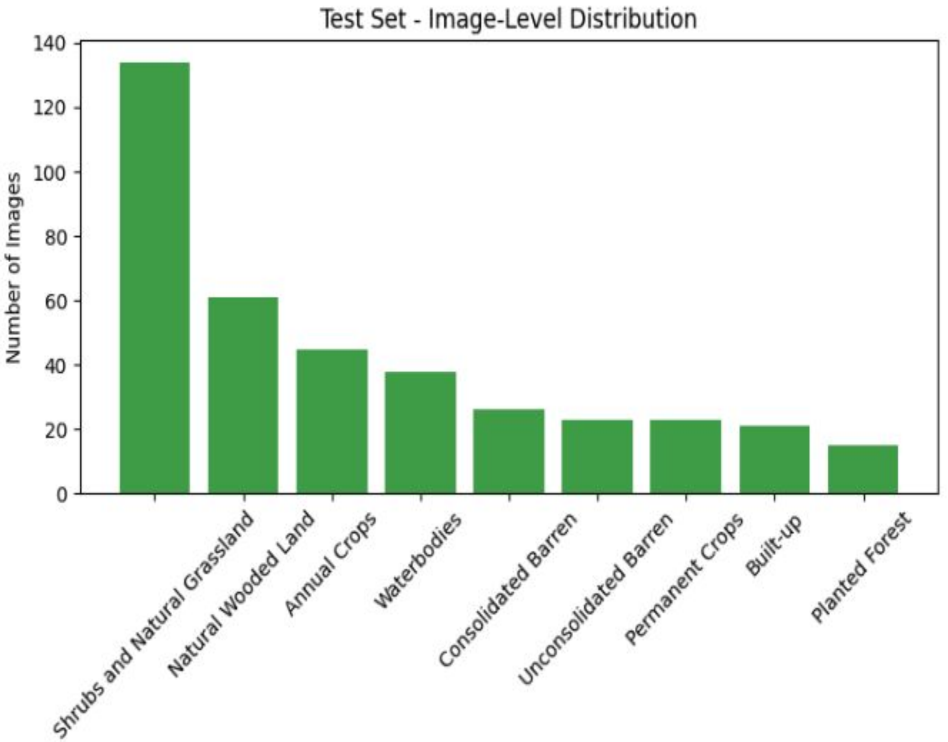
	precision	recall	f1-score
Annual Crops	0.91	0.85	0.88
Built-up	0.96	0.97	0.96
Cons. Barren	0.84	0.66	0.74
Wooded Land	0.78	0.65	0.71
Permanent Crops	0.95	0.88	0.91
Planted Forest	0.73	0.63	0.68
Shrubs & Grassland	0.76	0.92	0.83
Uncon. Barren	0.95	0.64	0.76
Waterbodies	0.99	0.96	0.98
accuracy	0.83		
macro avg	0.87	0.80	0.83
weighted avg	0.83	0.83	0.82

Data Imbalance

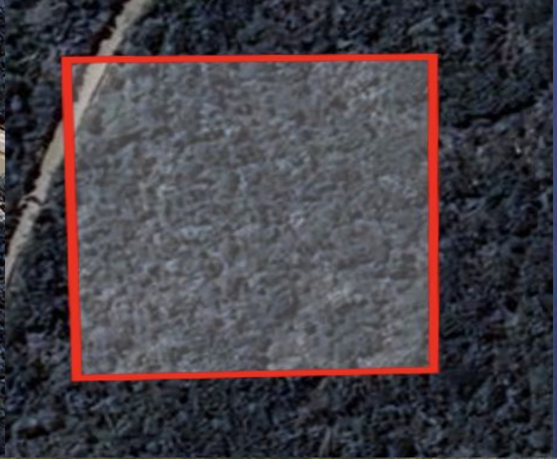
- Oversampling
- Class Weights

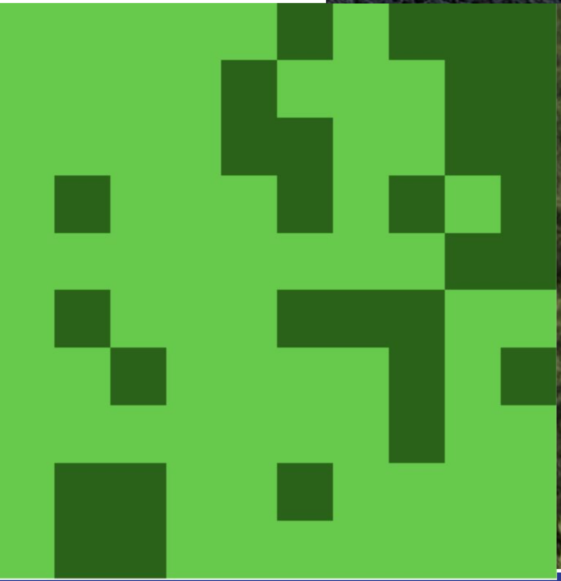






Ground Truth Labeling





Post-Processing Confusion Matrix (CNN)

- Morphology: Based a pixel's prediction on its neighbors
- Process: 2D post-processing technique in which each resulting image produced from the predicted pixels has a small amount of Dilation applied to them by expanding regions of predicted classes by adding neighboring pixels to the class in order to smooth the predictions and reduce noise.

Legend:

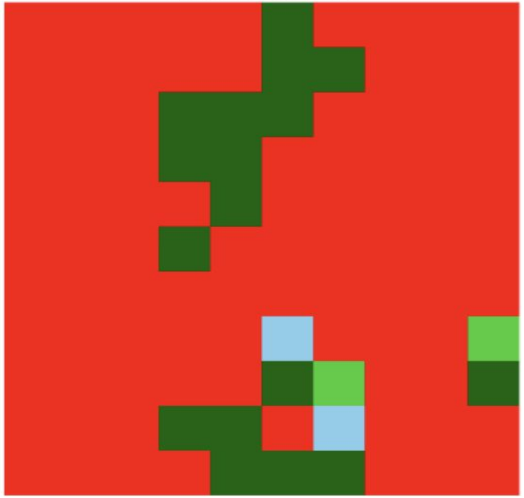
Built-Up

Natural Wooded Area

Permanent Crops

Natural Grassland

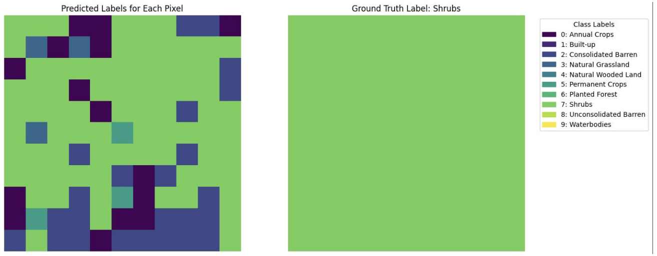
Original Pred Labels
Sample_num: 5699
Ground Truth: Built-up



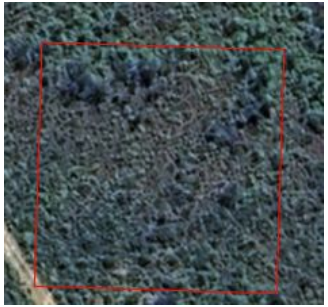
Filtered Pred Labels
Sample_num: 5699
Ground Truth: Built-up

Recognizing Trends

- We do extremely well in the “built-up” and “waterbodies” categories
  - This is likely because they are easily distinguishable compared to other categories
    - Water is a solid blue with consistent waves, buildings have rigid lines
- We perform worst on “natural grassland”
  - This is likely due to how seemingly mixed a lot of the grassland images were
    - We classified these images as “shrubs” more often than not
      - Consequently, we perform very well on shrubs!



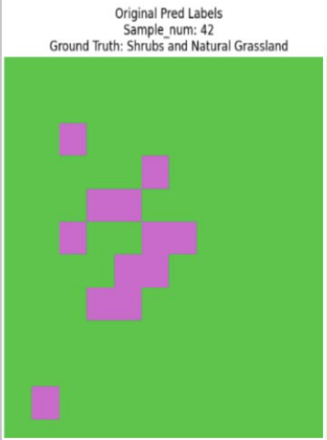
Example of grassland

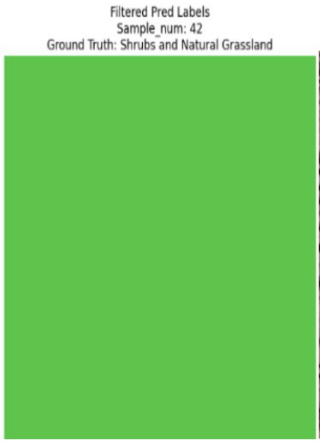


Example of shrubs

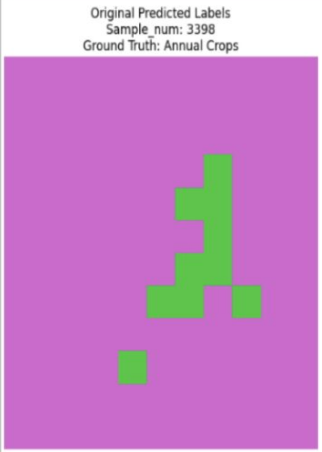
Pixel Analysis

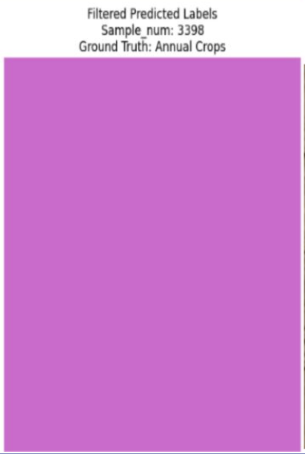
Sample 42

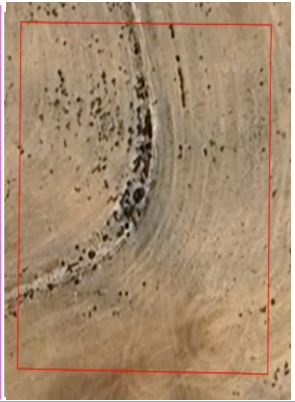


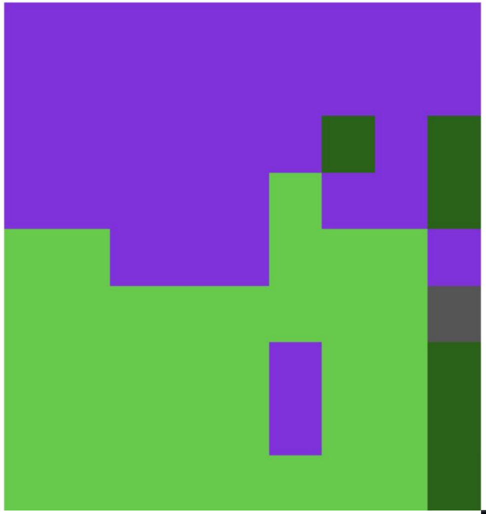


Sample 3398









Legend:

- Shrubs
- Annual Crops
- Planted Forest
- Natural Forest

Sample 2853