Satellite Imagery Classification with Deep Learning

Lisa Taylor

Completed as Capstone Assignment 2

Springboard Data Science Bootcamp

Overview

Satellite imagery

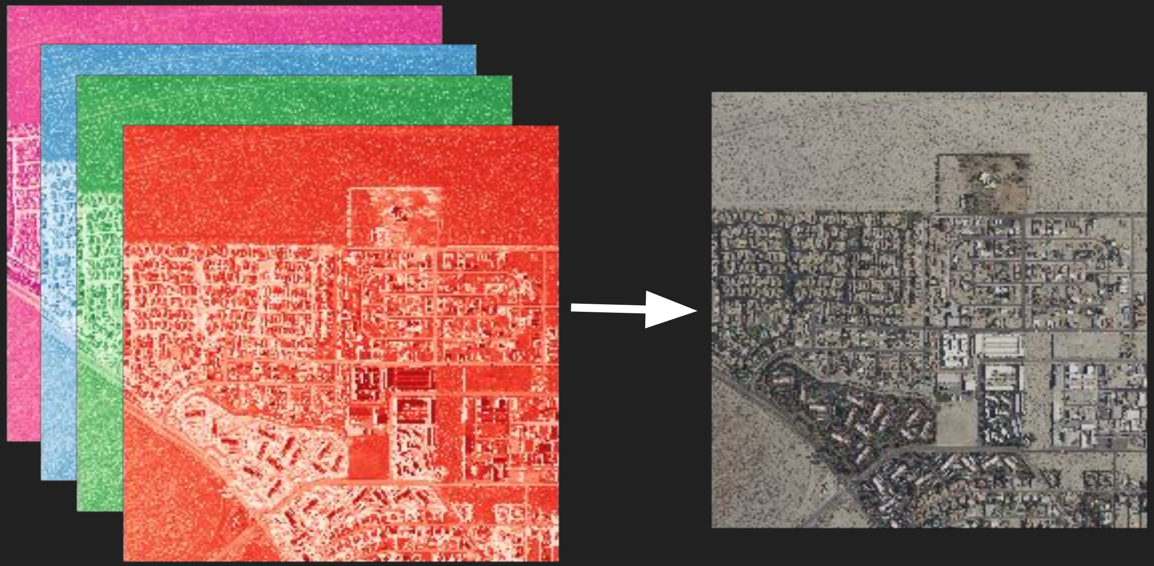
- Rapid growth in data acquisition
- Applications
    - Agriculture
    - Insurance
    - Disaster Response
    - Land Use Management
    - Environmental Management
- Common use-case: classify land uses within area of interest

Raw data → Process→ Interpret → Decisions

Screen captures from Google Earth

About Satellite Imagery

- Resolution
- Bands/Spectra
- Extent
- Temporal

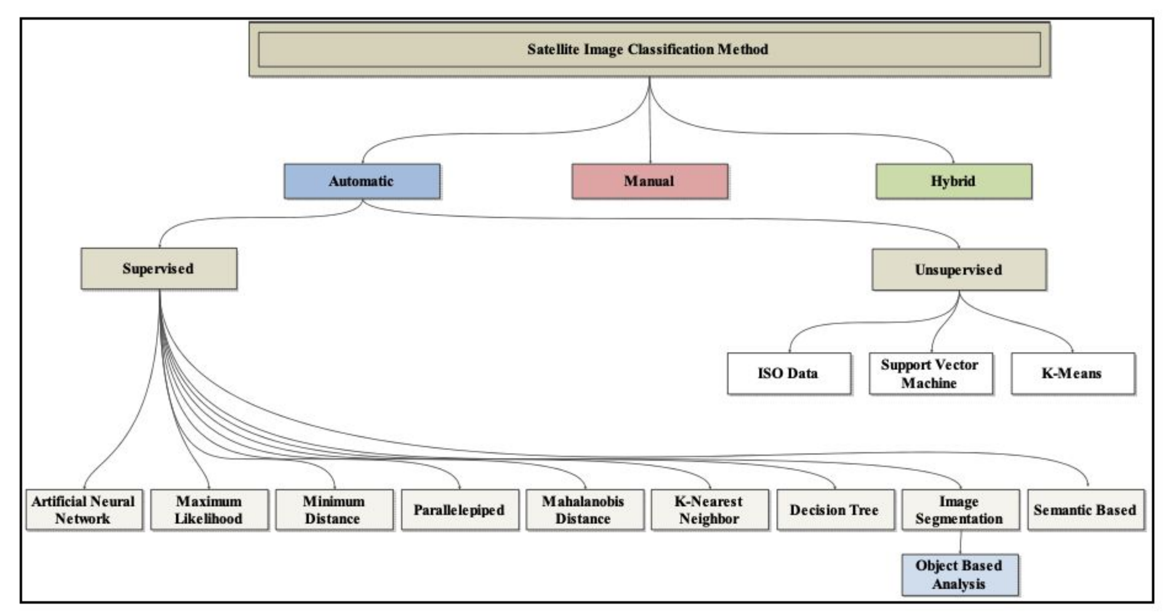


Satellite Imagery - Computer Vision Tasks

- Patch-based classification
- Semantic segmentation (per-pixel classification)
- Object counting
- Object detection (presence/absence, bounding box)
- Change detection

Satellite Image Classification with ML

GOAL: Group image pixels into meaningful categories



Ref: Abburu et al. Satellite Image Classificaiton Methods and Techniques, a Review. International Journal of Computer Applications Vol 119, No 8, June 2015.

Satellite Imagery Analysis With ML

Challenges:

- Big data
- Significant intra-class variability

Why Deep Learning?

- Traditional supervised learning methods like random forest do not scale well to big data.
- CNNs can use the underlying structure in images for classification. Context.

DeepSat-6 (Basu et al, 2015)

Labelled satellite imagery dataset, used for benchmarking classification models

405,000 patches sampled from NAIP

No spatial context

28x28 pixel tiles

1 meter resolution

4 bands: R,G,B,IR

6 class labels: building, barren land, tree, grassland, road, water

Training: 324,000 tiles (80%), Test: 81,000 tiles (20%)

Deep Learning for Computer Vision

Convolutional Neural Network (CNN)

- Deep learning model widely used in computer vision applications

Convolution

- Small tensor multiplied over sections of a larger image, like a filter
- Learn local patterns
- Convolutional layer applies multiple convolutions to the input
- Training learns weights of the convolutions that are most informative

Computer Vision Approaches

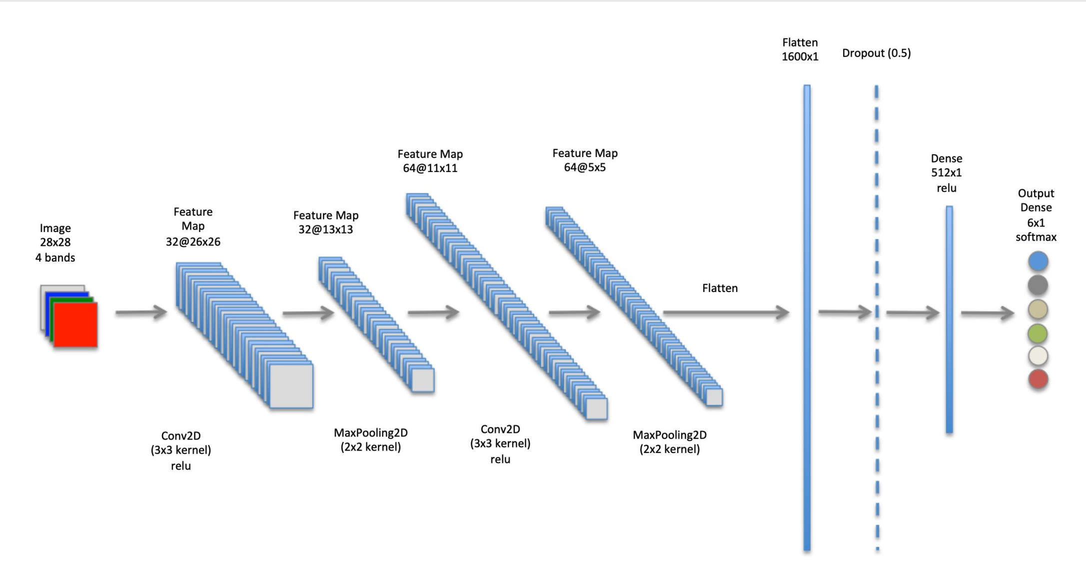
- Basic Architecture
  - Convolutional Base
  - Dense Classifier
- Approaches
  - Build and train CNN from scratch
  - Transfer Learning
    - Repurpose pre-trained CNN base (VGG16, Resnet) + custom classifier
    - Training options:
      - Apply convolutional base to dataset, fit classifier to numpy array output
      - Attach classifier to frozen convolutional base, train classifier model
      - Like prior, also re-train top layers of base (fine-tuning)
      - Unfreeze base and retrain entire

Baseline CNN - Architecture

Input
- ImageDataGenerator (train,val)

Convolutional Base
- Maxpooling - downsampling
- Relu - nonlinearity
- Learns patterns at different scales
- Translation invariant

Dense Classifier
- Learn global pattern
- Softmax
- Dropout
- Output classification



Baseline CNN - Training

Backpropagation - gradient descent from output to input

Loss Function	*How performance is measured on training data*	Categorical Cross-Entropy (Softmax Loss) -log(softmax(s)) for positive class
Optimizer	*How network parameters are updated during training*	SGD (lr=0.01)
Output Metrics	*What measures to record during training*	Accuracy

Deep Learning Platform: Keras on Colab

Keras

- High-level library for building deep learning models
- Multiple backend deep learning frameworks (Tensorflow, Theano, CNTK) for handling tensor operations
- Modular approach
- Leverage backend engine to compute on GPU

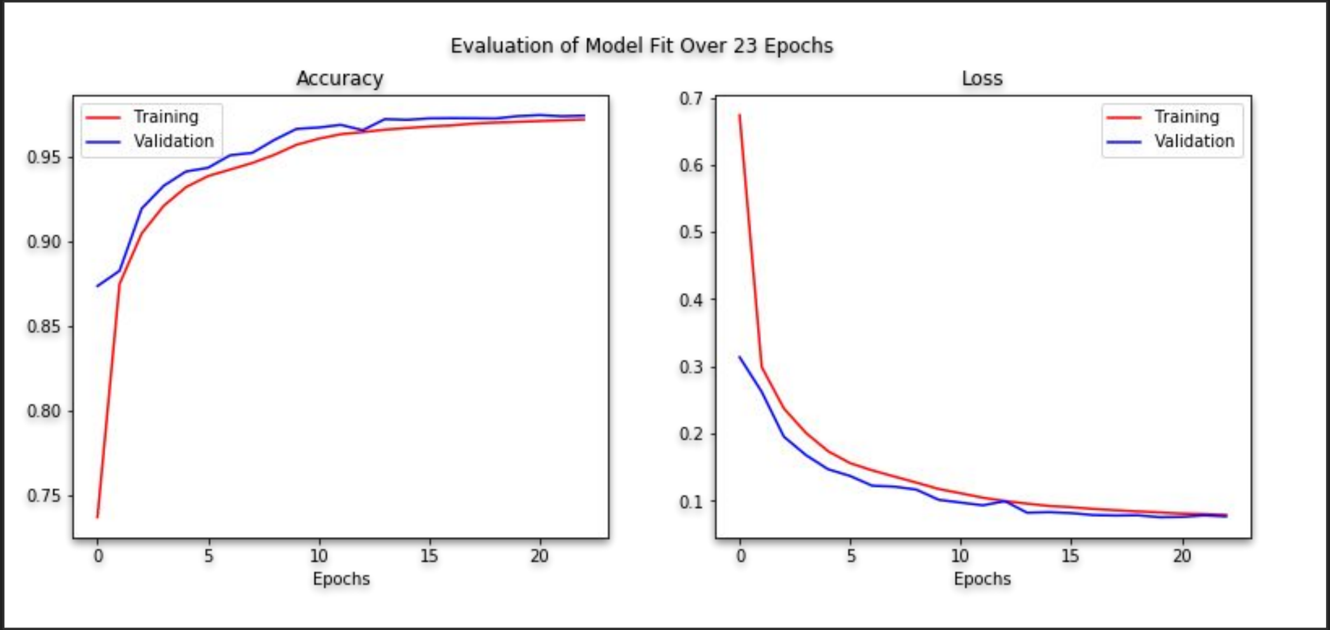
Colab

- Jupyter notebook environment running python in cloud
- Access to GPU resources
- Free

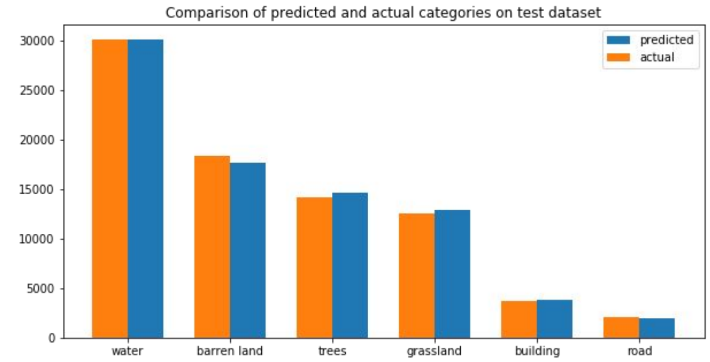
Building a CNN in Keras

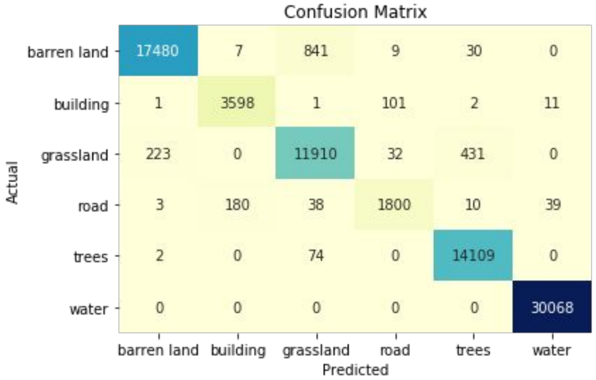
```
# Set up the model
from tensorflow.keras import models,layers
model = models.Sequential()
model.add(layers.Conv2D( 32, (3, 3), activation='relu', input_shape=(28,28, 4))) #  RGB+IR images.
model.add(layers.MaxPooling2D(( 2, 2)))
model.add(layers.Conv2D( 64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D(( 2, 2)))
model.add(layers.Flatten())   #collapse to 1D
model.add(layers.Dropout( 0.5)) #added to reduce overfitting
model.add(layers.Dense( 512,activation='relu')) #reduce after flatten
model.add(layers.Dense( 6,activation='softmax')) #final 6-way classification, predict class
model.summary()

# Compile the model
model.compile(loss='categorical_crossentropy' ,optimizer=optimizers.SGD(lr= 0.01),metrics=['accuracy'])

# Train the model
history=model.fit_generator(train_generator,steps_per_epoch= 545,epochs=30,validation_data=val_generator,validati
on_steps=126,callbacks=[early_stop_mon])
```

Baseline Model - Training Performance



Baseline Model - Test Performance





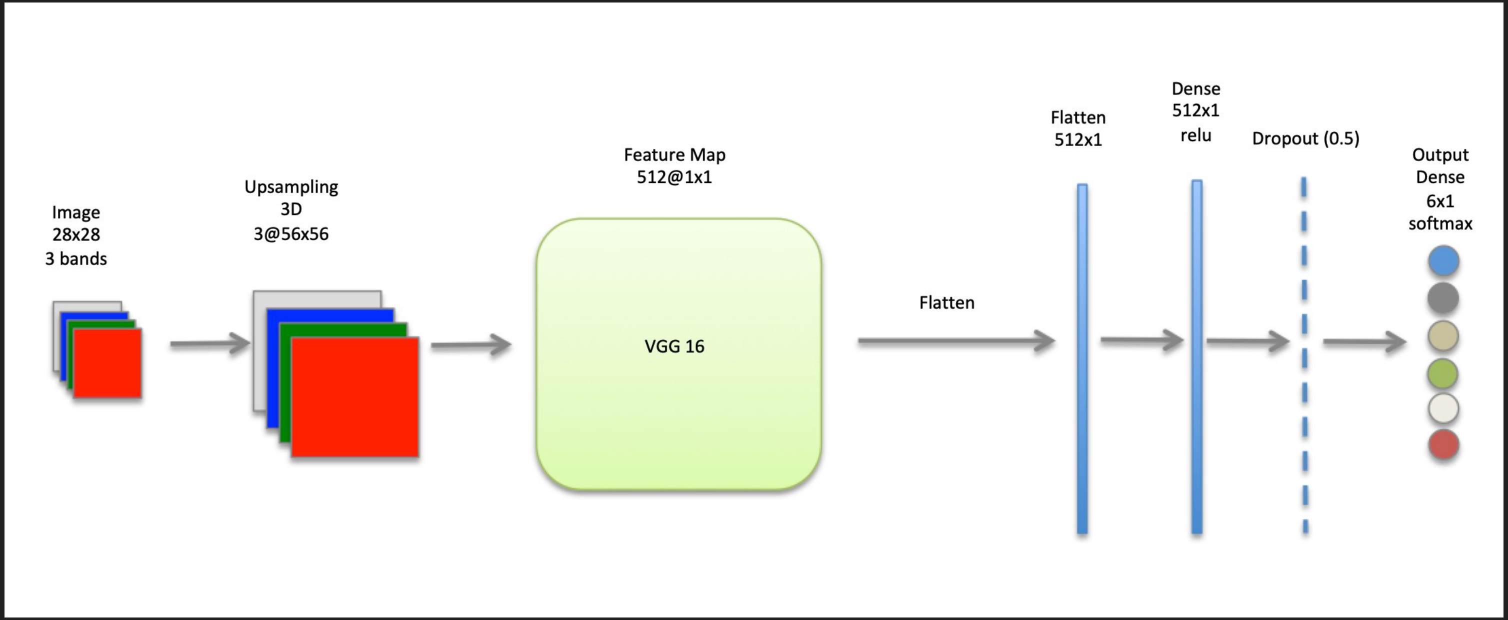
Overall accuracy on test dataset: 0.975

Classification Report:

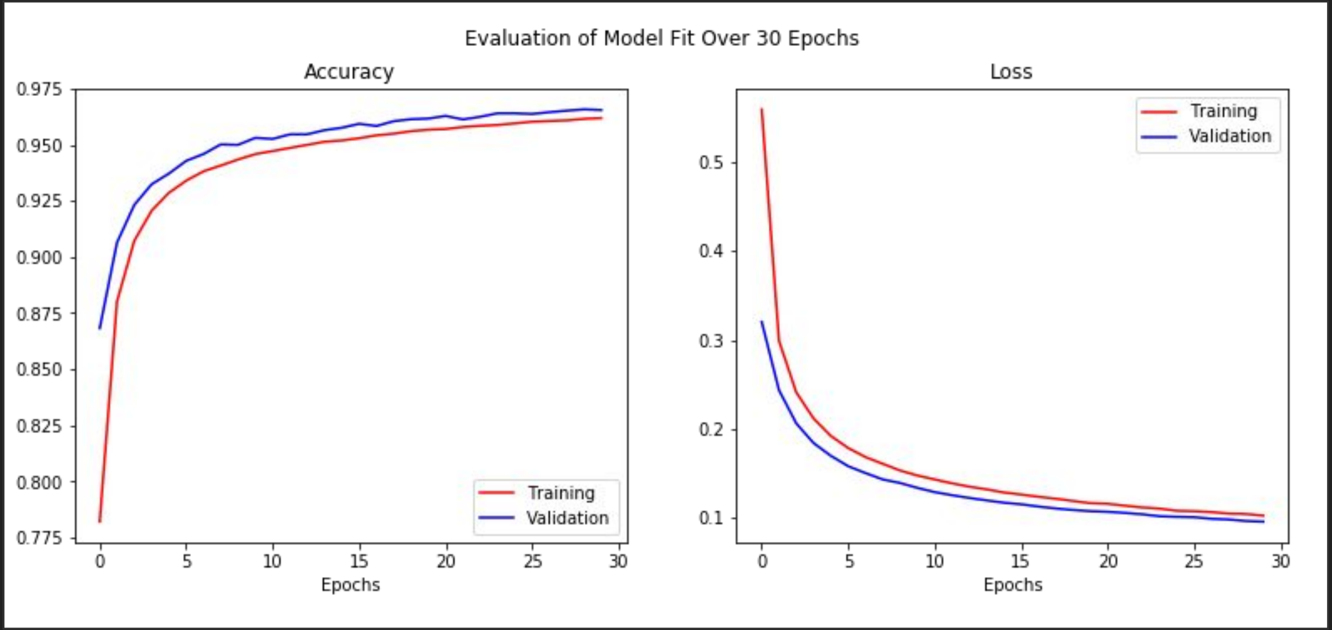
	precision	recall	f1-score	support
barren land	0.99	0.95	0.97	18367
building	0.95	0.97	0.96	3714
grassland	0.93	0.95	0.94	12596
road	0.93	0.87	0.90	2070
trees	0.97	0.99	0.98	14185
water	1.00	1.00	1.00	30068
accuracy			0.97	81000
macro avg	0.96	0.96	0.96	81000
weighted avg	0.98	0.97	0.97	81000

Transfer Learning with VGG16

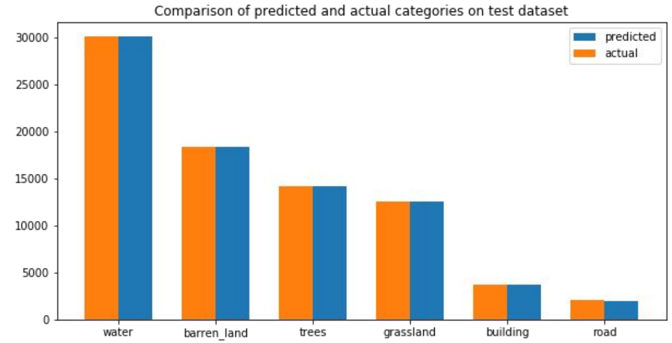
Architecture:

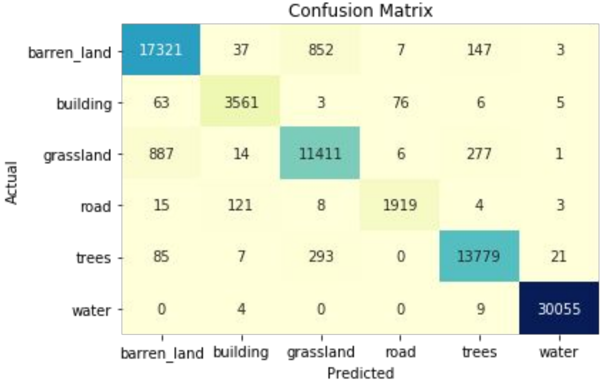


VGG16 - Training Performance



VGG16 Model - Test Performance



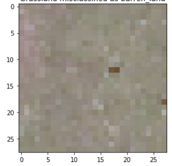


Overall accuracy on test dataset: 0.964

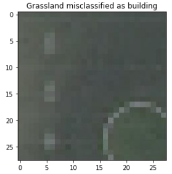
Classification Report:

	precision	recall	f1-score	support
barren_land	0.94	0.94	0.94	18367
building	0.95	0.96	0.95	3714
grassland	0.91	0.91	0.91	12596
road	0.96	0.93	0.94	2070
trees	0.97	0.97	0.97	14185
water	1.00	1.00	1.00	30068
accuracy			0.96	81000
macro avg	0.95	0.95	0.95	81000
weighted avg	0.96	0.96	0.96	81000

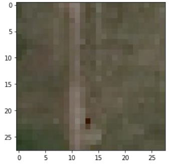
Grassland misclassified as barren_land



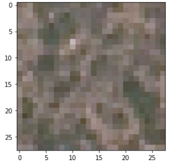
Grassland misclassified as building



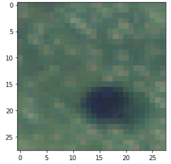
Grassland misclassified as road



Grassland misclassified as trees



Grassland misclassified as water



Misclassifications - Class Probabilities

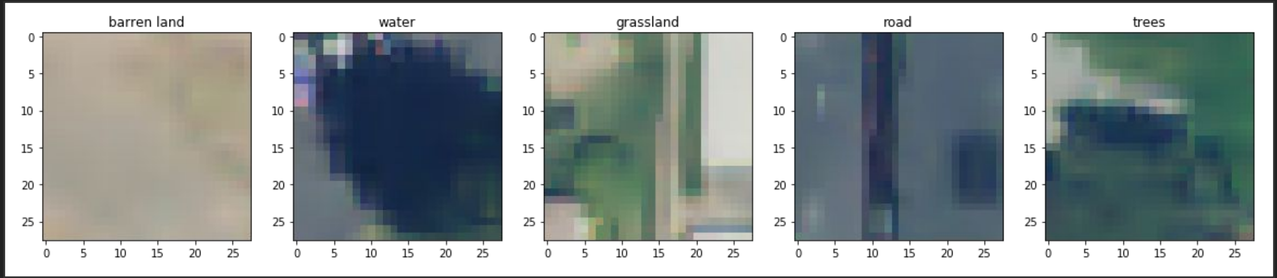
Class Probabilities							
building	**barren_land**	**trees**	**grassland**	**road**	**water**	**Actual**	**Predicted**
0.482	0.505	0.000	0.012	0.001	0.000	building	barren_land
0.000	0.151	0.022	0.827	0.000	0.000	barren_land	grassland
0.000	0.598	0.001	0.402	0.000	0.000	grassland	barren_land
0.000	0.125	0.003	0.872	0.000	0.000	barren_land	grassland
0.000	0.119	0.001	0.881	0.000	0.000	barren_land	grassland
0.101	0.891	0.000	0.000	0.007	0.000	building	barren_land
0.000	0.816	0.002	0.182	0.000	0.000	grassland	barren_land
0.763	0.000	0.000	0.000	0.237	0.000	road	building
0.000	0.113	0.059	0.828	0.000	0.000	trees	grassland
0.001	0.257	0.003	0.740	0.000	0.000	barren_land	grassland

Apply Baseline Model to New Image

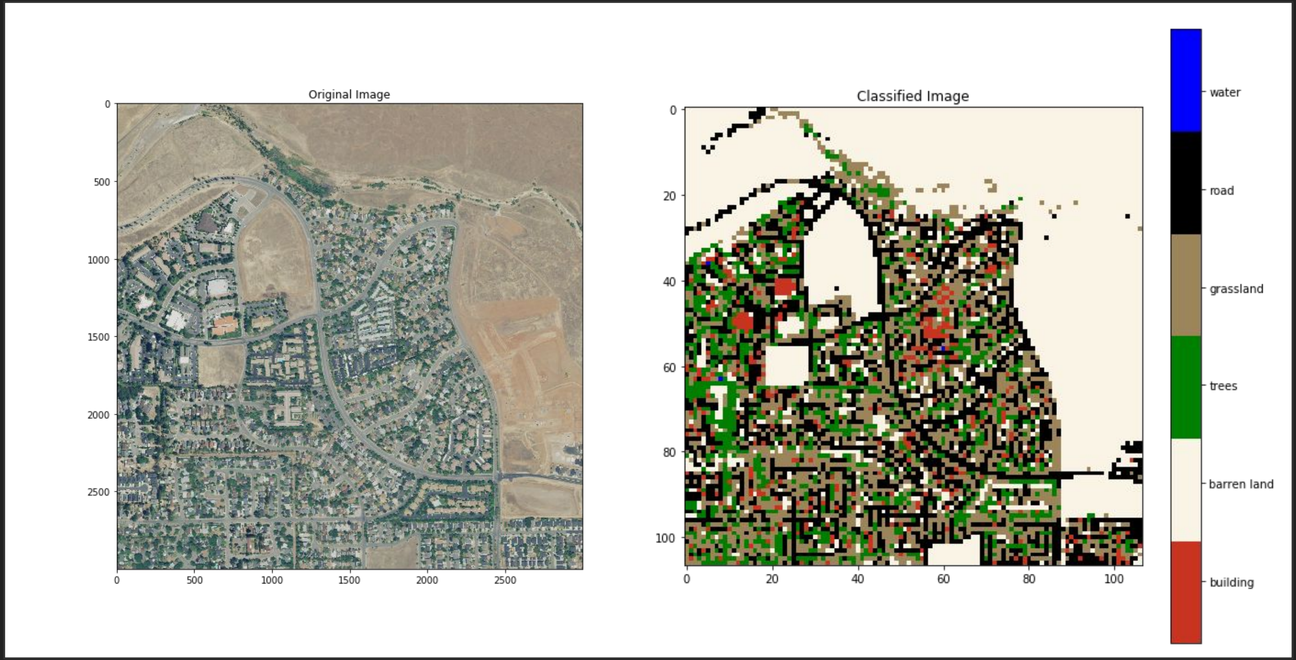
Download new NAIP image

Split into tiles, Reshape to channels-first ndarray

Input to model.predict()



Reassemble Classified Image



Summary of Results

- Applied Deep Learning with CNN to classify satellite image tiles
- Used DeepSat6 benchmarking dataset
- Two methods:
  - Simple Baseline CNN (97.5% accuracy)
  - Transfer learning with VGG16 (96.4% accuracy)
- Applied Baseline CNN to new NAIP imagery

Follow-on Work

- Additional testing on NAIP tiles
- Georeference output tile for additional spatial analysis
- Address resolution loss
  - Classify overlapping tiles, assign class to central pixels
  - Image segmentation with U-Net