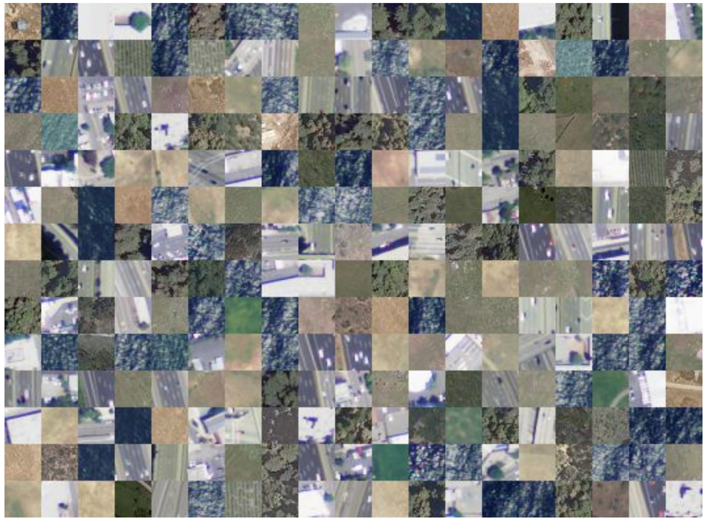
Classifying DeepSat-6 Satellite Images Using Convolution Neural Networks

Varsha Gopalakrishnan

Project Overview

- Mapping land cover types is one of the primary methods to track land use activity across the earth's surface. Satellite images and GIS are some of the most commonly used data sources.
- In this project, I'm using the DeepSat-6 satellite dataset to build a multi-class classification algorithm using **Convolution Neural Networks**.
- This project will be useful for
  - Environmental conservationists to identify areas that have been converted to barren land, crops or urban areas.
  - Identify barren lands and grasslands that can be reforested.
  - Construction agencies and real estate management companies to identify urban areas for future development.



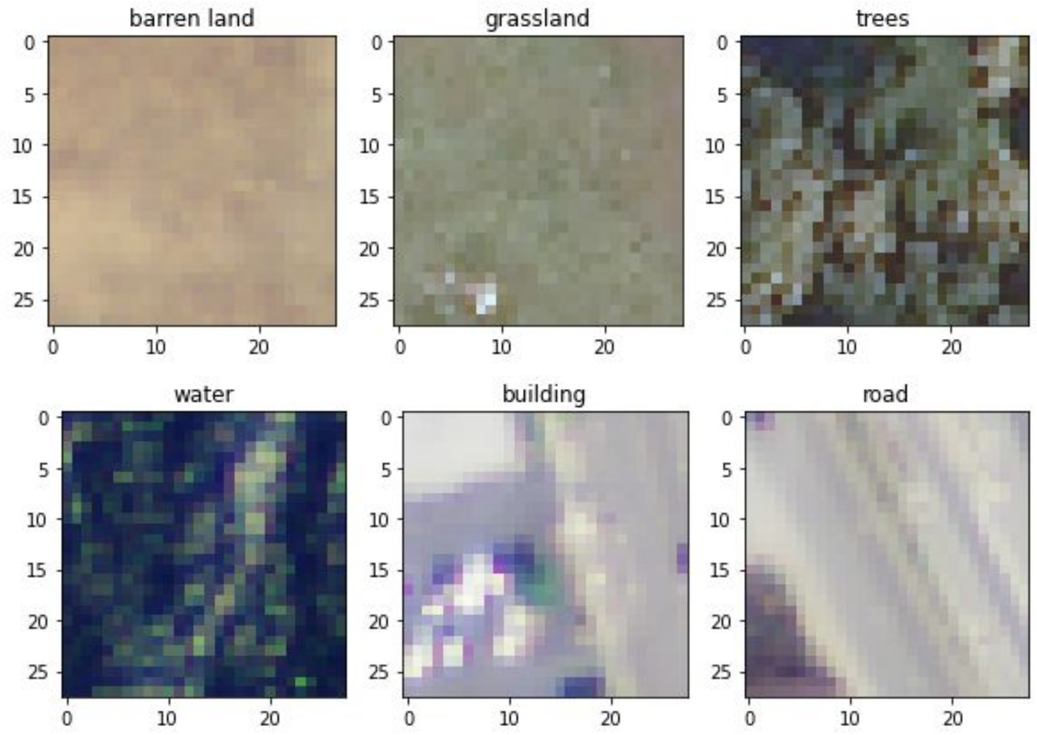
Sample Images from DeepSat-6

Deepdive into DeepSat-6

- Dataset obtained from Kaggle
- Contains images obtained from National Agriculture Imagery Program (NAIP) dataset taken in California [Ref].
    - 405,000 image patches each of size 28x28
    - 6 land cover classes - barren land, trees, grassland, roads, buildings and water bodies
    - 324,000 training images with labels
    - 81,000 test images with labels consisting of one-hot encoded vectors
    - Each image is a 28 by 28 pixel image with 4 channels - Red, Green, Blue and Near InfraRed (NIR).

Exploratory Data Analysis

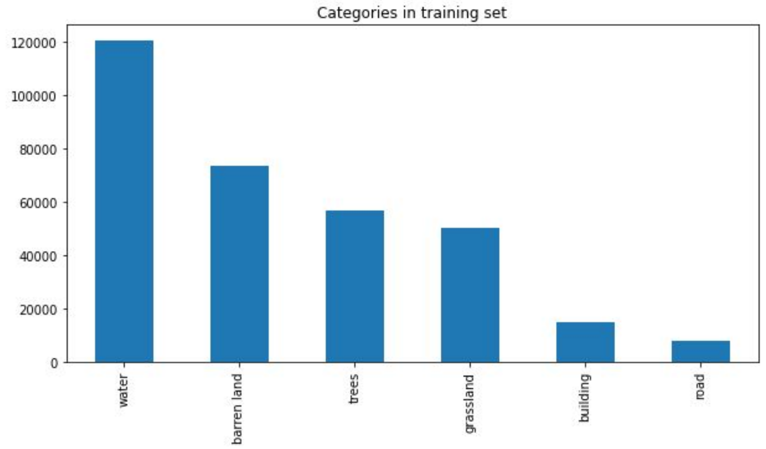
1) Sample images from dataset:

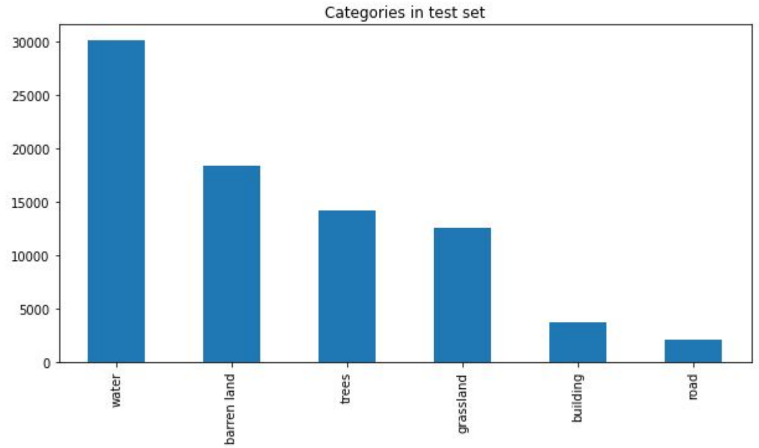


Exploratory Data Analysis

2) Occurance of Labels:

- Water is the most commonly occuring image in the dataset
- Unequal distribution of images by land use type will likely result in a class imbalance issue while building the Convolution Neural Network (CNN) [Ref]

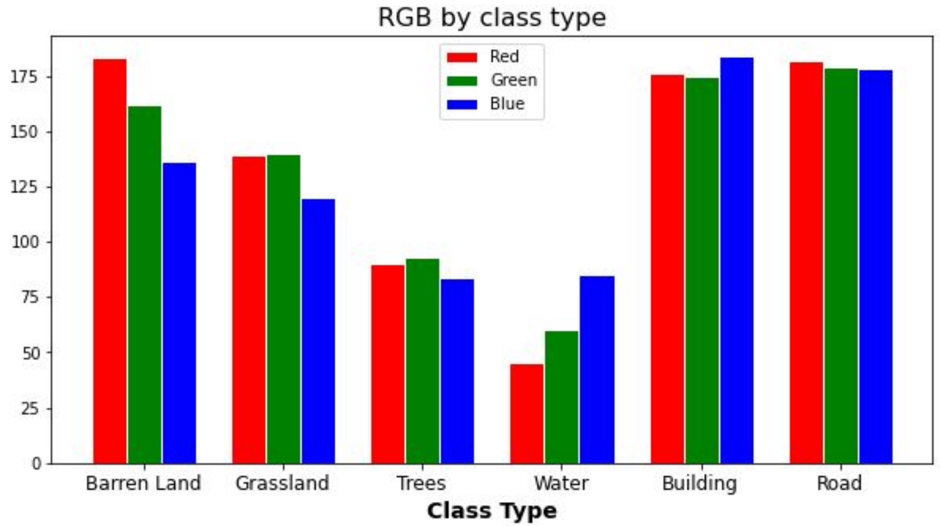




Exploratory Data Analysis

3) Decomposing Images:

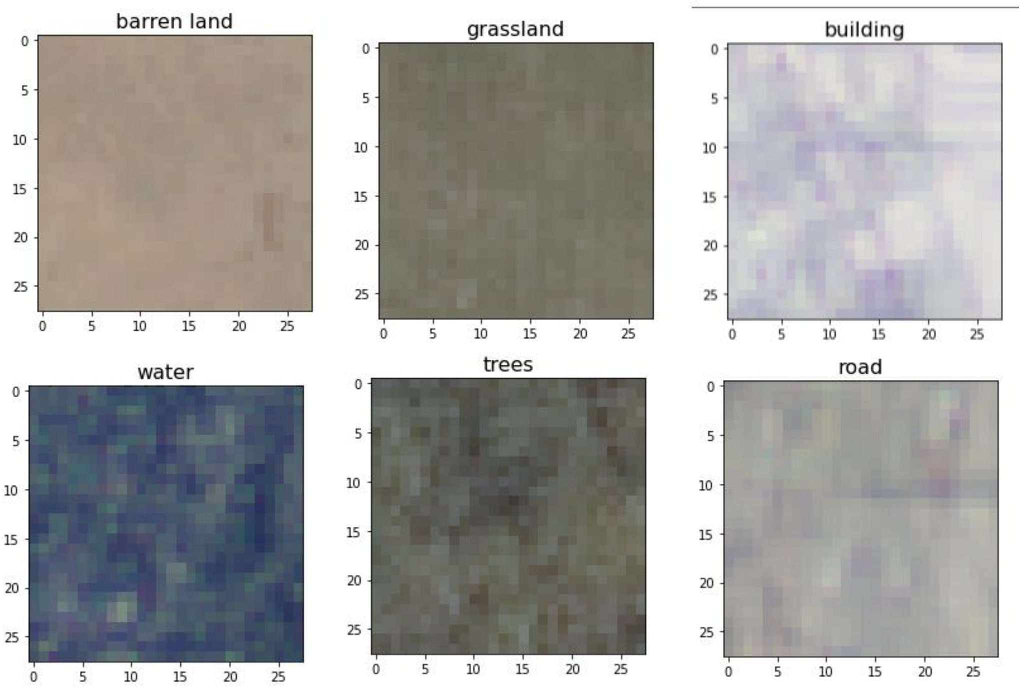
- Average RGB values of all images belonging to a class type
- Red is more dominant in barren lands, blue is more dominant in water land class
- Buildings and roads have equal shades of RGB



Exploratory Data Analysis

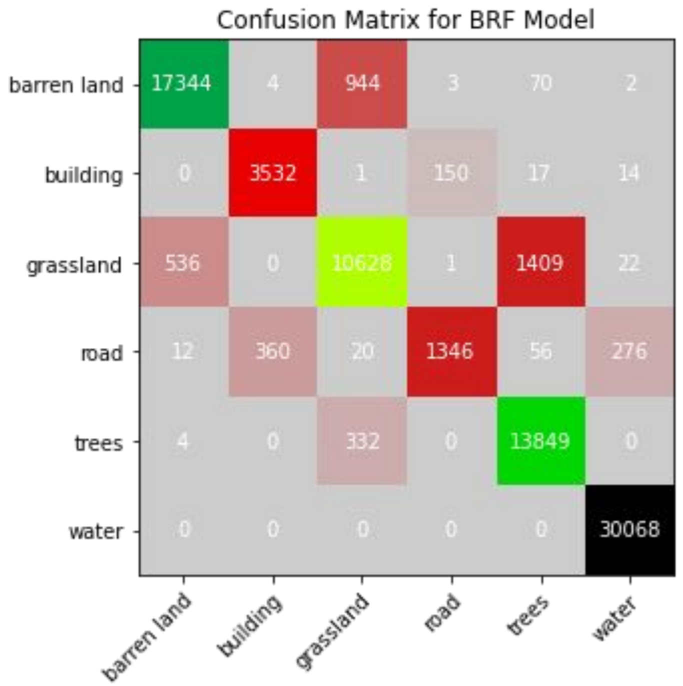
4) Average Image by Class Type:

- Average RGB channels calculated on a pixel by pixel basis for each label in training data
- Water land class has more shades of blue while buildings and road have equal shades of RGB appearing as a light color



Machine Learning for Image Classification

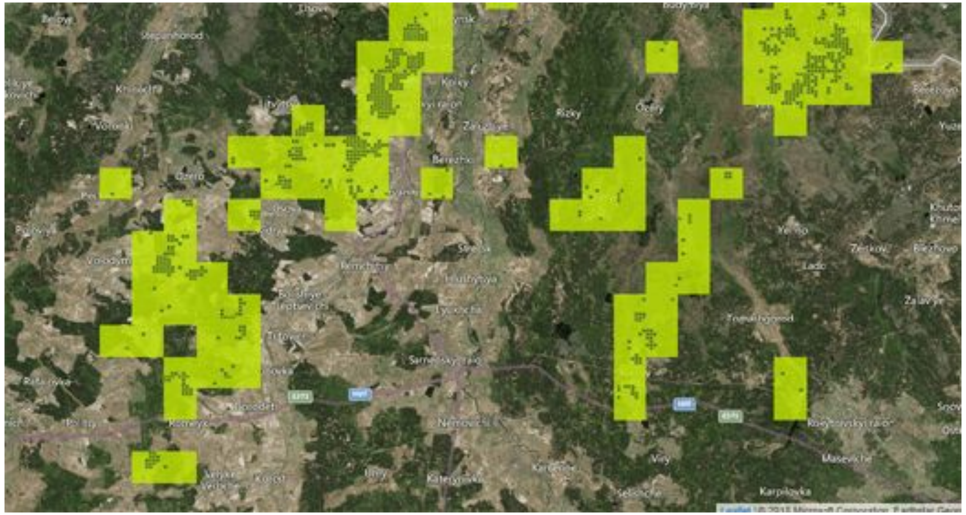
- As a first step, a simple Random Forest approach was used to build a baseline model to classify images
- Performance of Baseline Random Forest (BRF) will be compared against CNN models
- Model trained on a subset of training data with 20,000 images
    - `Max depth = 10`
    - `Number of estimators = 100`
- Overall balanced accuracy= **89.69%**
- Several mis-classifications of land classes


Confusion Matrix for BRF Model

	barren land	building	grassland	road	trees	water
barren land	17344	4	944	3	70	2
building	0	3532	1	150	17	14
grassland	536	0	10628	1	1409	22
road	12	360	20	1346	56	276
trees	4	0	332	0	13849	0
water	0	0	0	0	0	30068

Application of Deep Learning - Image Classification

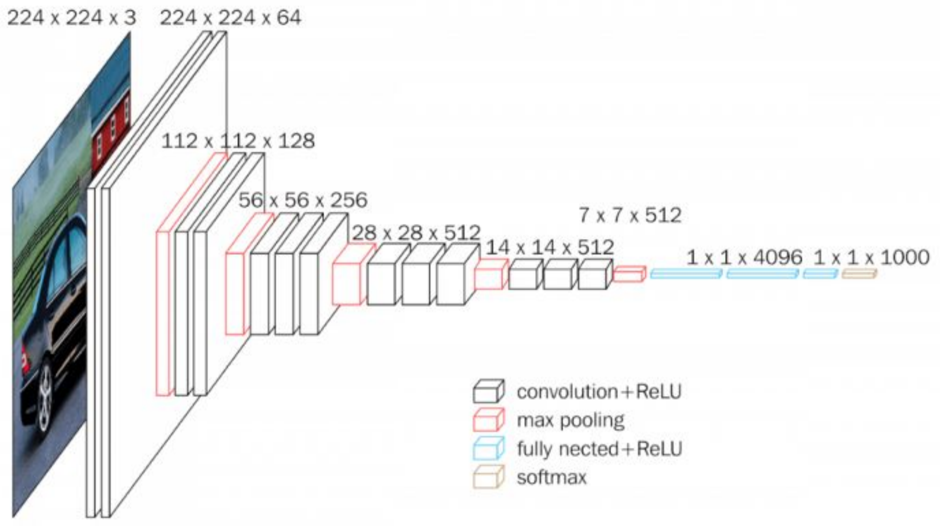
- Convolution Neural Networks (ConvNet/CNNs) is a Deep Learning algorithm which takes an image as input, assigns weights and biases to various aspects of the image and classifies images into different categories to differentiate them from one another.
- Image recognition through CNN is now the foundation of modern computer vision.
- CNNs have been successfully applied in satellite image classification problems [Ref].



Satellite spying on the Earth

Convolution Neural Network Models

- Four different CNNs were built and tested for classifying the DeepSat-6 dataset using Keras with Tensorflow backend**.
  1) Baseline CNN - *architecture built from scratch*
  2) Transfer Learning by Padding Input Image - *VGG16 architecture*
  3) Transfer Learning with Fine Tuning Layers and Padding Input Image - *VGG16 architecture*



VGG-16 architecture

*** The fourth CNN model is similar to transfer learning with padding but instead of padding the input image, Upsampling is used. This approach is not discussed in the presentation but please refer to the report for more information.*

Hyperparameters

Training/Validation Set

- '`ImageDataGenerator`' to generate batches of tensor image data with real-time augmentation to split training dataset into training and validation sets.
- Image rescaled by dividing RGB value by 255

Model compilation, checkpoint

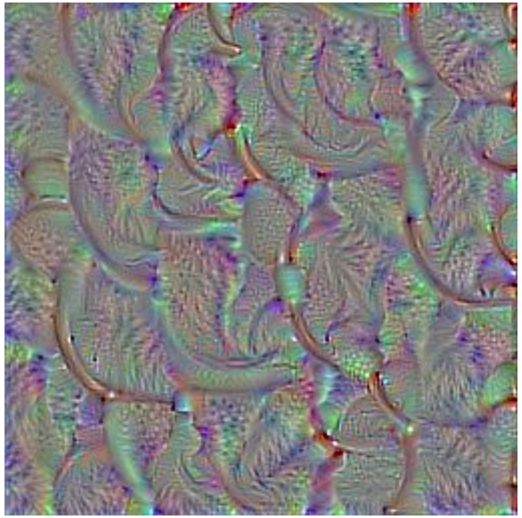
- `Adam optimizer`
- `Categorical crossentropy` loss function
- Steps per epoch = 150
- Epoch = 20

Evaluating Model Performance

- Balanced accuracy (Average recall of all classes)
- Other metrics evaluated:
  - Number of predicted vs actual images
  - Classification report and confusion matrix
  - Prediction error

Feature Visualization

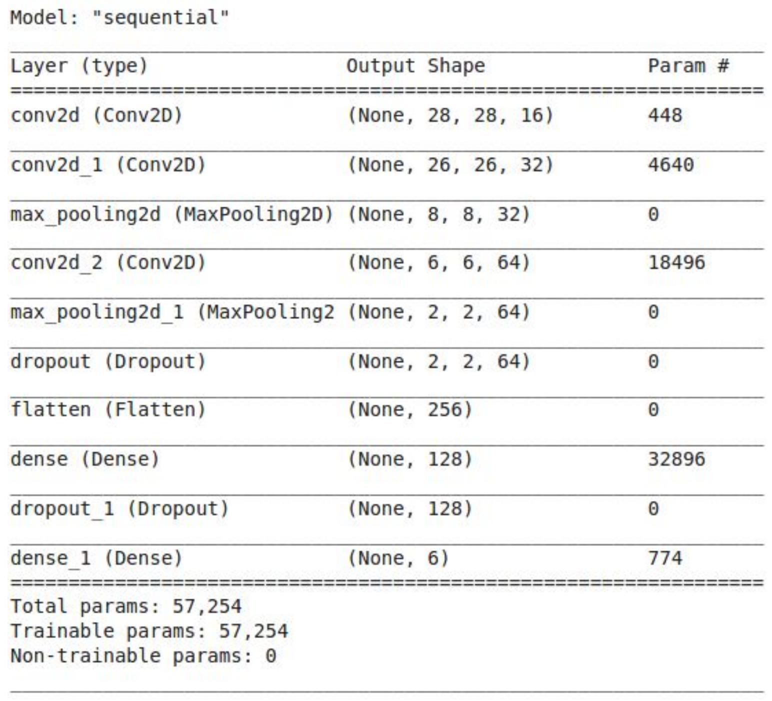
- Maximize activation of a specific filter in a target layer.
- Represent a visualization of the pattern that filter corresponds to.
- Resulting image gives us an insight into what the network’s fitler is “looking for” or identifying in each image.



Sample filter in Vgg16 model

Baseline CNN Model

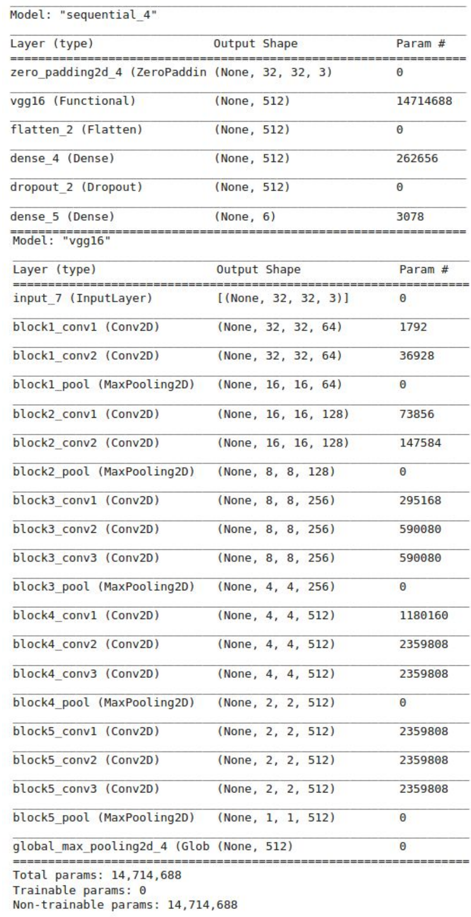
- Input image = (28,28,3)
- Architecture consists of convolution, maxpooling, dense, flatten and drop out layers
- Final fully connected dense output of (6x1) estimated using 'softmax' activation
- Highest probability of occurrence of a class assigned to each image
- 57,524 trainable parameters



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        448
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 26, 26, 32)        4640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 6, 64)          18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 2, 2, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 57,254
Trainable params: 57,254
Non-trainable params: 0
_________________________________________________________________
```

Transfer Learning with VGG16 (Padding)

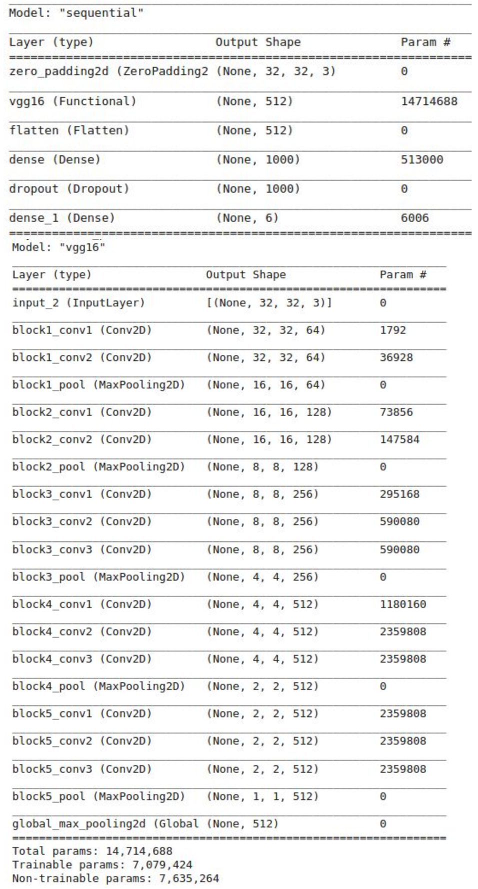
- Input image = (28,28,3)
- First layer is a ZeroPadding layer to convert input image to (32,32,3) (smallest image size for VGG16)
- Entire VGG16 network with weights and biases, all parameters set as non-trainable
- Flatten layer to (512x1) format followed by dense layer
- Final fully connected dense output of (6x1) estimated using ‘softmax’ activation
- 265,734 trainable parameters



```
_________________________________________________________________
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
zero_padding2d_4 (ZeroPaddin (None, 32, 32, 3)         0
_________________________________________________________________
vgg16 (Functional)           (None, 512)               14714688
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 3078
=================================================================
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
_________________________________________________________________
global_max_pooling2d_4 (Glob (None, 512)               0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
```

Transfer Learning with Fine Tuning VGG16

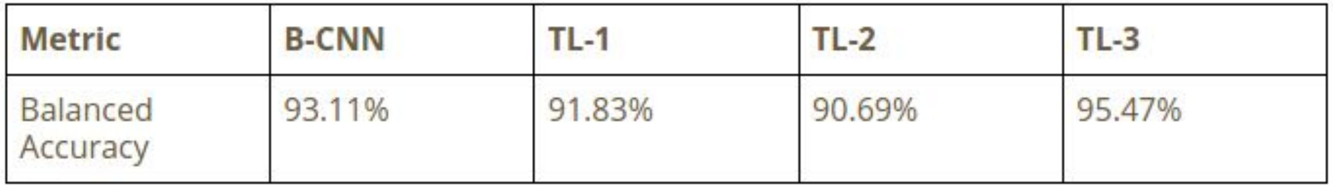
- Input image = (28,28,3)
- Same architecture as previous transfer learning model
- Changing last dense layer to match our dataset; allow last convolution layer (block 5) and classification stages to train (set parameters as trainable) but retain the weights of the rest of the layers
- Final fully connected dense output of (6x1) estimated using ‘softmax’ activation
- 7,079,424 trainable parameters



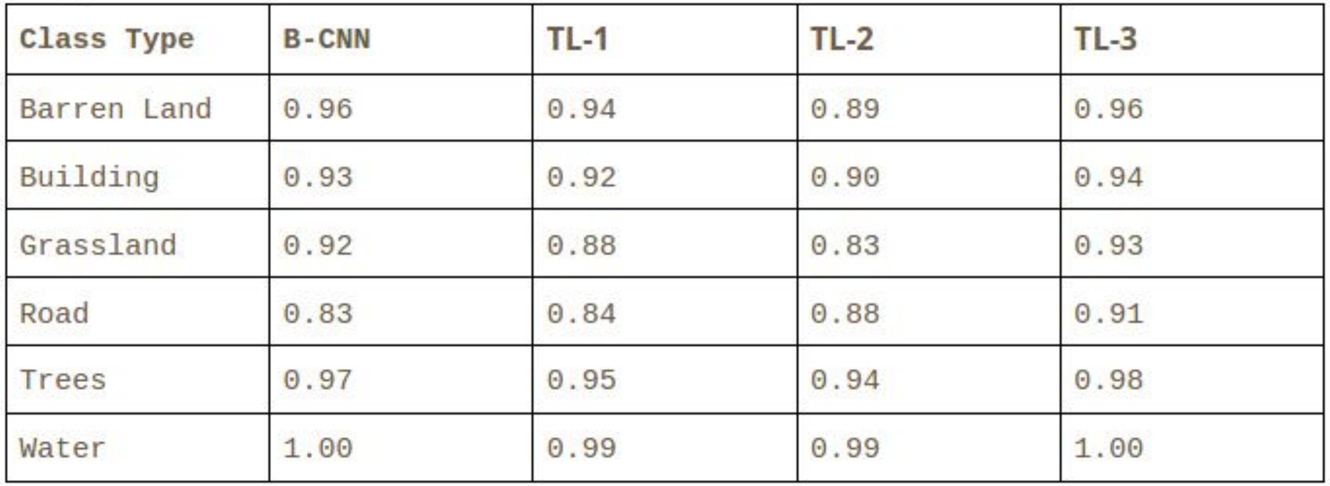
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
zero_padding2d (ZeroPadding2 (None, 32, 32, 3)         0
_________________________________________________________________
vgg16 (Functional)           (None, 512)               14714688
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 1000)              513000
_________________________________________________________________
dropout (Dropout)            (None, 1000)              0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 6006
=================================================================
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
_________________________________________________________________
global_max_pooling2d (Global (None, 512)               0
=================================================================
Total params: 14,714,688
Trainable params: 7,079,424
Non-trainable params: 7,635,264
```

Evaluating Model Performance

- Overall balanced accuracy and F1 scores for the different land classes are presented below.
- Model that yielded the highest balance accuracy score is the Transfer Learning approach with VGG16 and fine tuning, followed by the Baseline CNN model built from scratch.



Metric	B-CNN	TL-1	TL-2	TL-3
Balanced Accuracy	93.11%	91.83%	90.69%	95.47%

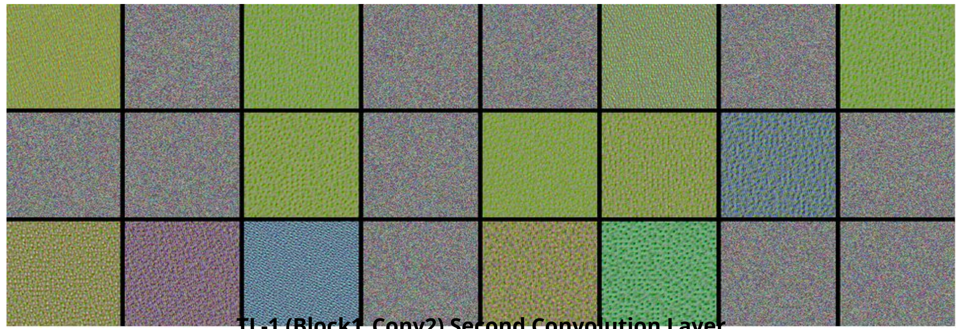


Class Type	B-CNN	TL-1	TL-2	TL-3
Barren Land	0.96	0.94	0.89	0.96
Building	0.93	0.92	0.90	0.94
Grassland	0.92	0.88	0.83	0.93
Road	0.83	0.84	0.88	0.91
Trees	0.97	0.95	0.94	0.98
Water	1.00	0.99	0.99	1.00

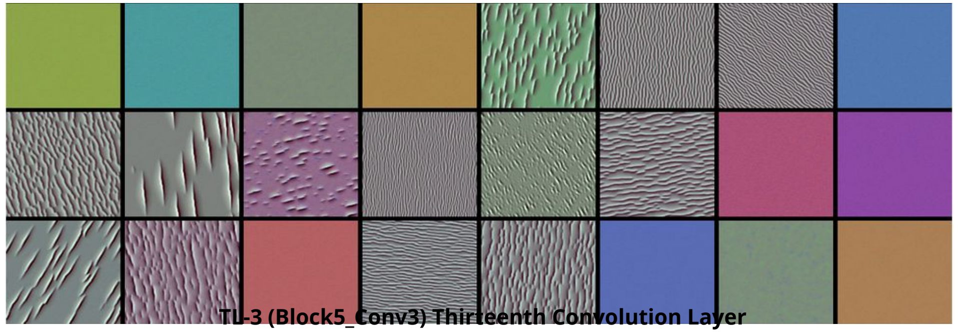
Feature Visualization

- **Baseline CNN Model**
  - Second convolution layer
  - Detects basic features such as colors and to a certain extent directions
- **Transfer learning with Padding Model**
  - Second convolution layer in first block of Vgg16
  - Encode directions more distinctly as well as colors. The horizontal lines could represent water and the vertical lines could be roads
- **Transfer learning with Padding Model**
  - Thirteenth convolution layer - represents a much deeper layer in the architecture; filter is detecting more complex patterns
  - Encode directions more distinctly as well as colors. The horizontal lines could represent water and the vertical lines could be roads

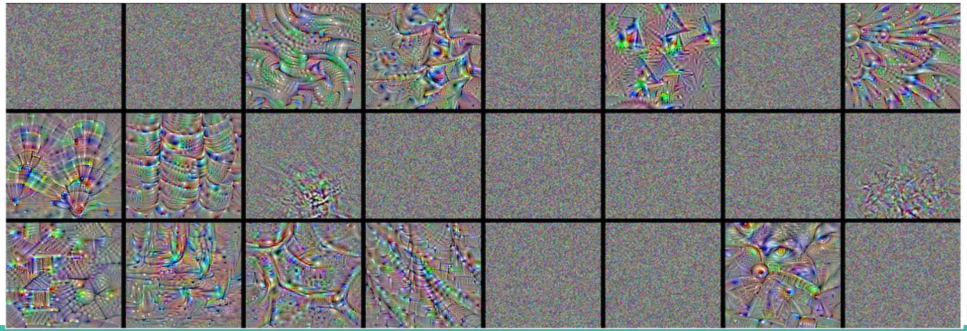
Baseline CNN (Conv2d_2) Second Convolution Layer



TL-1 (Block1_Conv2) Second Convolution Layer

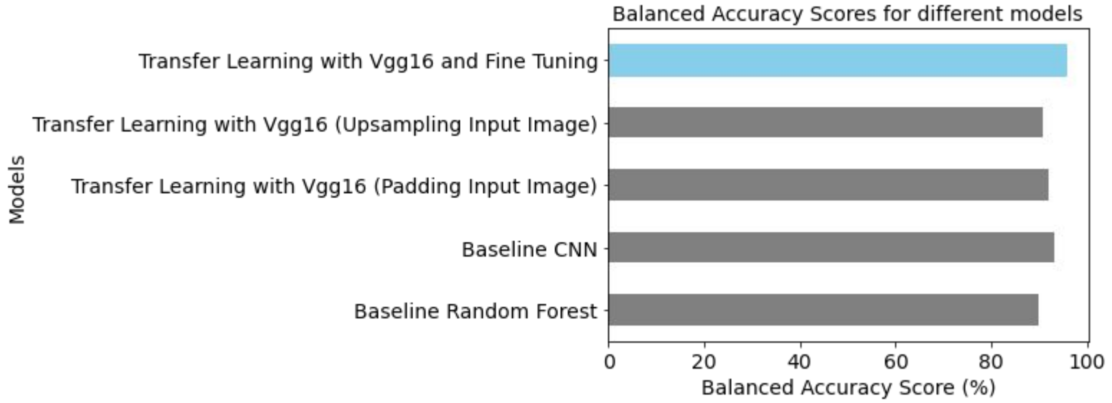


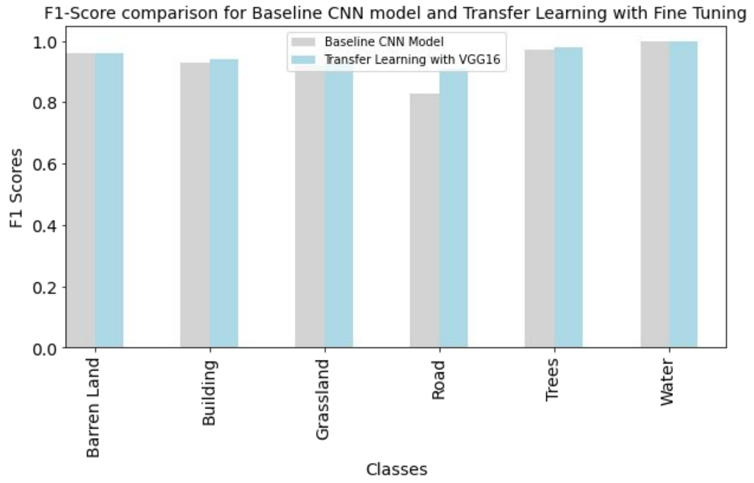
TL-3 (Block5_Conv3) Thirteenth Convolution Layer



Model comparison

- The model that yielded the highest balance accuracy score is the Transfer Learning approach with VGG16 and fine tuning
- Second best model is the Baseline CNN model
- F1 scores for all land classes are higher or similar for the transfer learning with fine tuning model compared to the baseline CNN model.



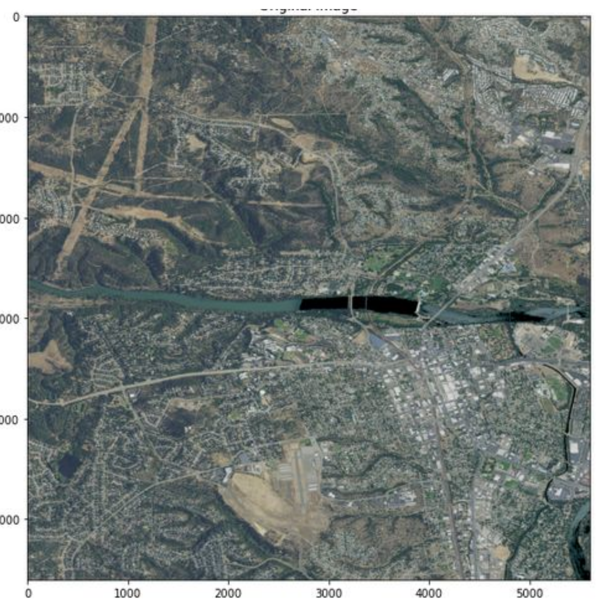


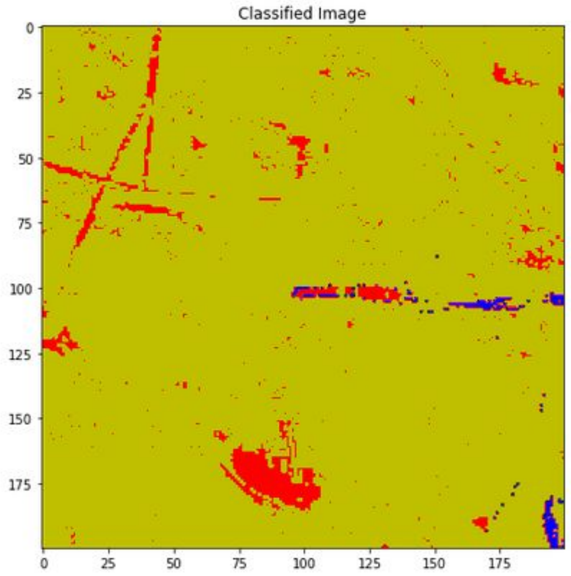
Making Predictions on a new NAIP image

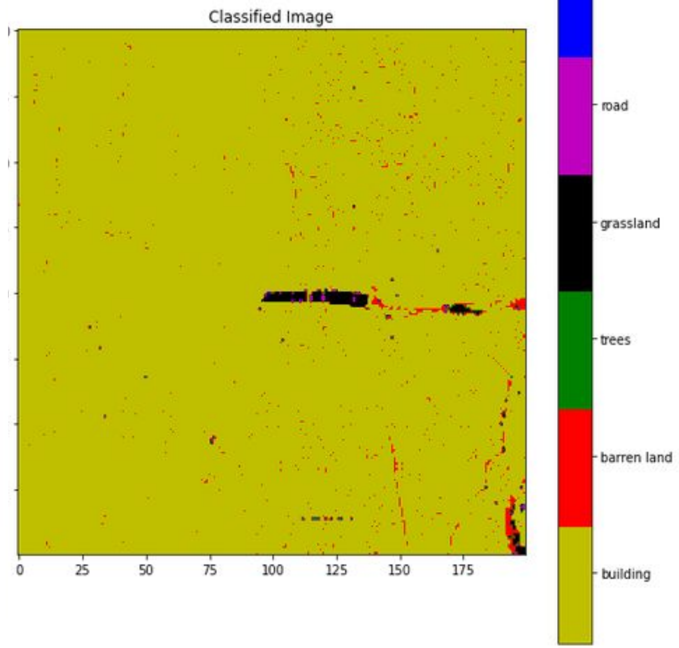
Original Image

Baseline CNN Model

Transfer Learning Model







Conclusions and Future Work

- Explored different CNN models for classifying the DeepSat-6 satellite image dataset
- Balanced accuracy score was used as the metric to evaluate and compare model performance since the training dataset contains an imbalance of classes.
    - Overall, the transfer learning model with fine tuning of one of the layers resulted in the highest balanced accuracy of **95.47%**.
- Transfer learning model with fine tuning had a deeper and a more complex network compared to the baseline CNN model.
- The transfer learning model however did not do a great job of making predictions on new images at different resolutions.
- Future work :
    - Identifying optimal hyperparameters that can improve model performance
    - Using the transfer learning model built here to build an app that can classify satellite images