Remote Sensing Image Classification Based on CNN Network

——Data Science Capstone

Qing Ruan

May 3, 2021

Table of Contents

1. Introduction

1.1 Background

With the development of earth observation technologies, large amount of remote sensing data can be acquired. How to effectively mine and classify these images have become a challenging problem.

As one of the deep learning models, convolutional neural networks (CNNs) show a good performance in exploiting information of imagery data compared with other convention learning algorithms.

1.2 Problem Statement

Shallow learning methods are limited to distinguishing images with more complex information. Remote sensing images are usually high-resolution.

Among neural networks, several hot pre-trained models for image classification such as GoogleLenNet, ResNet and VGGNet have been proposed and they have achieved record-breaking success on ImageNet Large Scale Visual Recognition Challenge (LSVRC) contest. However, different from ImageNet dataset, remote sense images have more particular features and the neural network models mentioned above might not be totally suitable. Therefore, in this project, a newly designed CNN architecture will be constructed.

1.3 Project Scope

This project aims to conduct a remote sensing image classification system to monitor natural environment, land use and vegetation ecology.

Specifically, I applied normalization to process data and implemented different CNN architecture consisting of convolution layers, pooling layers, drop-out rate and batch normalization. Then, several self-designed models in different structures, such as number of layers and

number of kernels will be compared. Among these architectures, the best model with highest accuracy will be selected and optimized furtherly to eliminate the overfitting and increase the accuracy. In the end, transfer learning basing on classical pre-trained models from ImageNet will be performed and compared with the customized CNN model.

2. Literature Review

Deep neural networks easily fit random labels and use data augmentation, regularization and batch normalization can easily eliminate overfitting.

AlexNet neural networks are conducted by 5 convolutional layers, 3 max pooling layers, 2 fully connected layers and 1 SoftMax layer. Relu as activation function and 60,000,000 parameters totally.

VGG neural network architecture used 3*3 convolutional layers. Kernels and stride 1which guaranteed no loss of information. Other parameters: Relu function, 5 max pooling layers, no normalization and 3 fully-connected layers.

3. Methodology

3.1 Data Description

This is a 21-class land use image dataset meant for research purposes. There are 100 images for each of the following classes: agricultural, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium residential, mobile home park, overpass, parking lot, river, runway, spare residential, storage tanks and tennis court.

Each image measures 256 * 256 pixels. The images were manually extracted from large images from the USGS National Map Urban Area

Imagery collection for various urban areas around the country. The pixel resolution of this public domain imagery is 1 foot.

3.2 Data Preprocessing

To save training time and simplify the complexity of models, images were firstly reduced into 64 * 64 pixels for each one. Then implementing min-max scaling project range into (0,1) to do image normalization. Next is one-hot encoding, convert numerical labels into categorical via `keras.utils.to_categorical`. The datasets of X(features) and Y(labels) are saved into .npy files respectively.

The result of dataset splitting is training set 60%, testing data 20% and validation 20%.

3.3 Model Building

In this part, we will build several different models basing on the number of layers and kernels and then select the best one which has the highest accuracy. Then the accuracy of the one selected will be improved further.

3.3.1 Model Selection

There are several variables being used to adjust the parameters of different models. They are the number of convolutional layers (2,3,4), number of kernels (32, 64, 128, 256). Other parameters are fixed.

For architecture parameters, for example, the filter size is 3 * 3, same padding pattern and max pooling. Two fully connected layers in each model and dropping out rate is 0.5. The selection of activation function is Relu and SoftMax.

As shown above, there are datasets available for training CNN-based remote sensing images classification models, the category types and numbers of labels in the datasets are still extremely limited and often fail to meet the data scale requirement of models training. Acquiring samples with manual visual interpretation has very low efficiency and a

relatively high cost. Therefore, to prevent overfitting, data augmentation is used to replace training data with new produced random data. For training data, we use `ImageDataGenerator` from keras to load data in batches and for validation and testing data without it. The specific parameters of data augmentation are as follow.

```
image_gen = ImageDataGenerator(
    #zoom_range = 0.1,
    width_shift_range=5,   # pixel
    height_shift_range=5,  # pixel
    horizontal_flip=True,
    vertical_flip=True,
    rotation_range=20)

# At Test/Validation time without data augmentation
image_gen_test = ImageDataGenerator()

train_gen = image_gen.flow(np.array(X_train),
                           y_train,
                           batch_size = 32,
                           shuffle = True)
val_gen = image_gen_test.flow(np.array(X_val),
                              y_val,
                              batch_size = 32,
                              shuffle = True)
```

The following ones show the images after data augmentation.

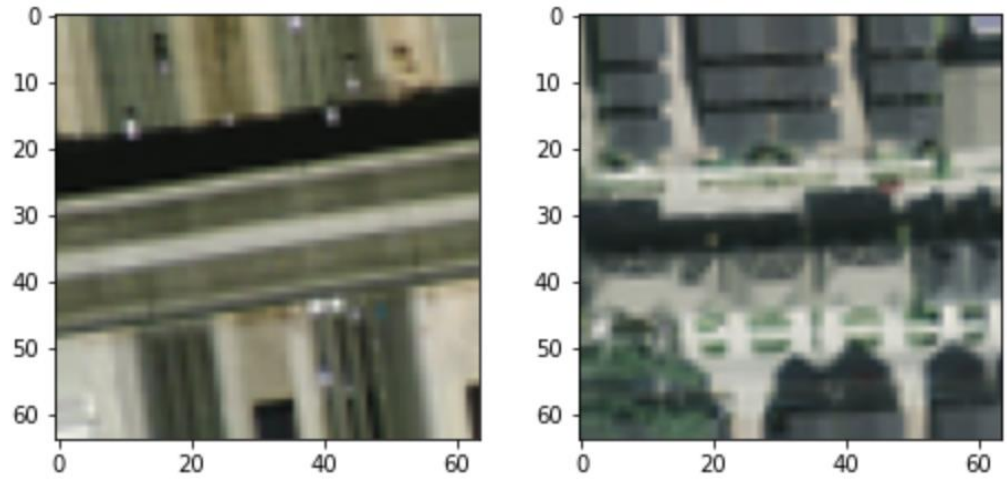


Fig 1. Images after data augmentation

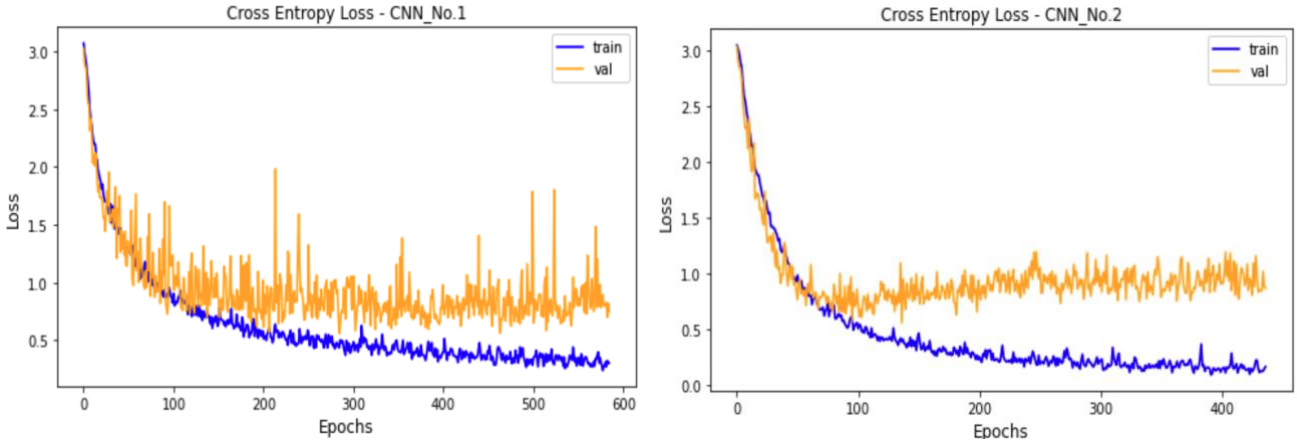
For compiling parameter, we use Adam optimizer, and the learning rate is 0.001. `categorical_crossentropy` as loss function. As the dataset is balanced, the metric is `accuracy`. `val_loss` is used to check earlystopping and `val_accuracy` is used to check best model.

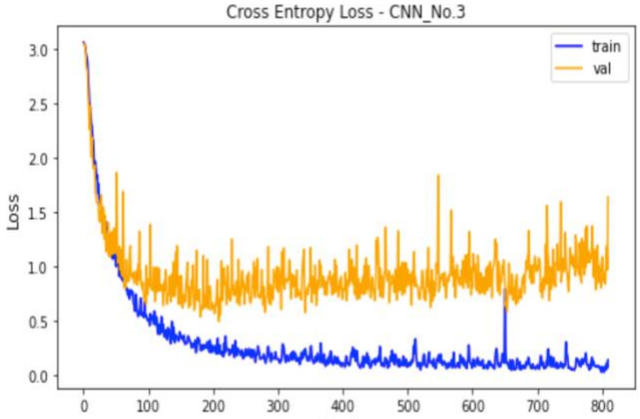
For training parameters, the number of epochs is 1000, the patience is 600 and batch size is 32.

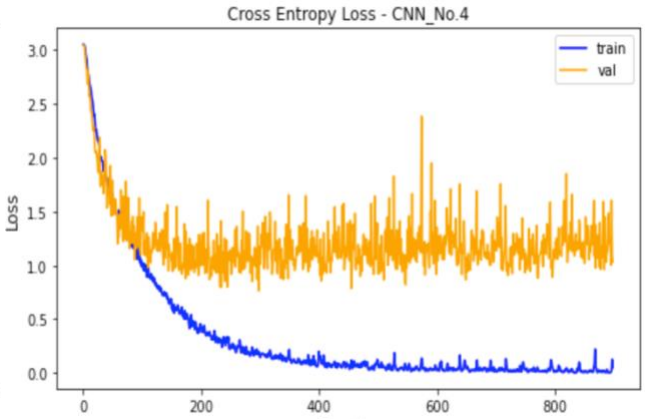
Table 1. Architecture with different models

No.	Architecture	Kernels	Conv layers
1	Conv + Pool + Conv + Pool	64 - 128	2
2	Conv + Pool +Conv + Pool	64 – 128 - 256	3
3	Conv + Pool + Conv + Pool	64 – 128 – 256 – 512	4
4	Conv + Pool + Conv + Pool + Conv + Pool +Conv + Pool	32 - 64 – 128 – 256 - 512	5
5	Conv + Pool +Conv + Pool	64 – 128 – 256	3
6	Conv + Conv + Pool + Conv + Conv + Pool	64 – 128 - 128 - 256	4

Here are the models I have tried on training datasets. And the results are the followings. The first one and the second one occurred early stopping in advance. All of these six models have overfitted, and for the fourth one, the value of validation cross entropy loss is the largest. It occurred overfitting first.







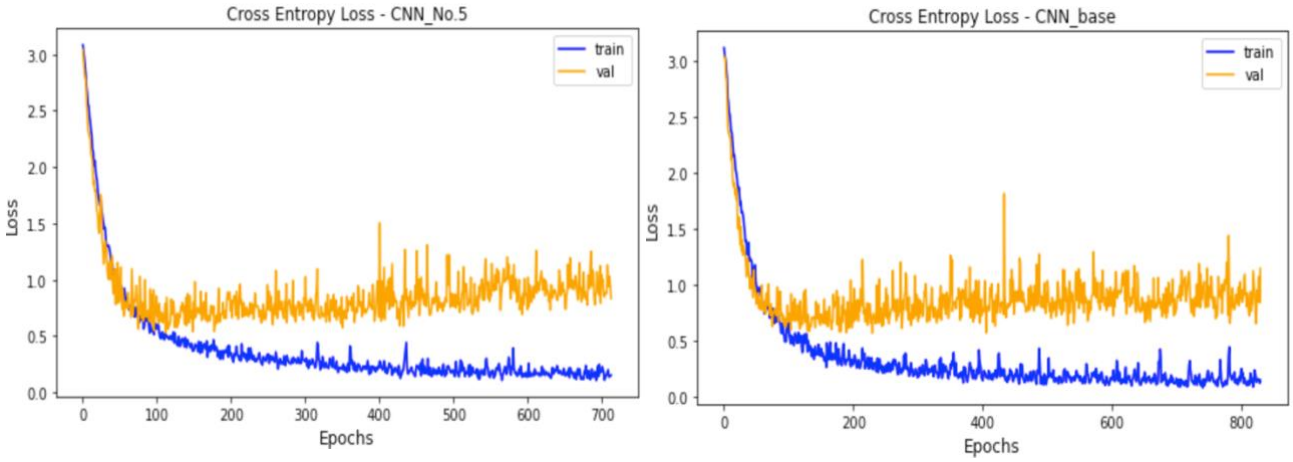


Fig2. Cross entropy with epochs in different models

These figures show the differences between train accuracy and validation accuracy in these six models. The differences in these models are 11.825, 12.698, 10.210, 13.810, 12.936 and 10.158, respectively. The third one and the sixth one overfitted less which represents 1(conv layer)-1-1-1 model and 2(conv layer)-2 model.

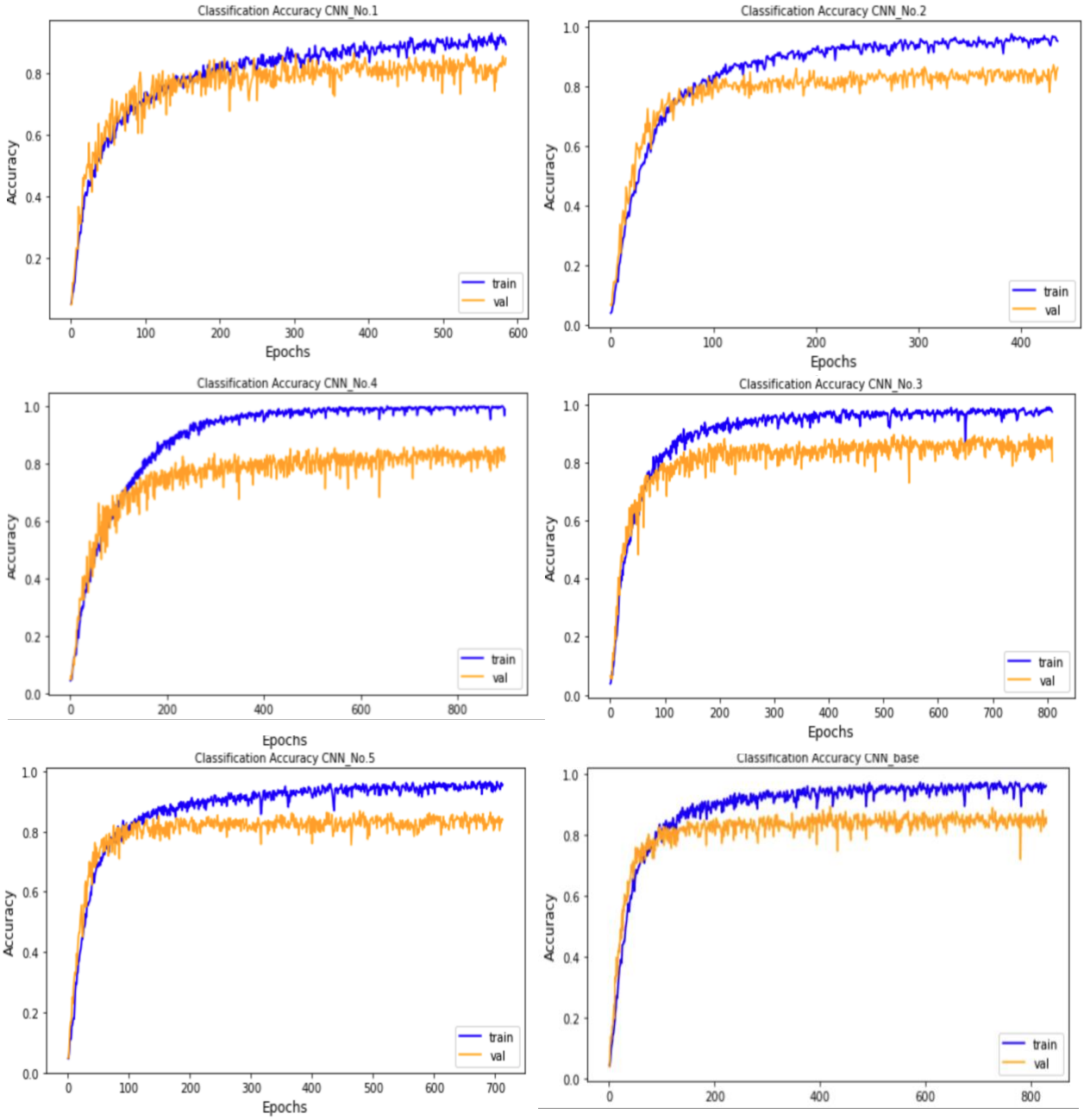


Fig 3. Classification accuracy with epochs in different models

This chart shows the results with different models.

Table 2. Results with different models

No.	Architecture	Kernels	Conv layers	Accuracy
1	Conv + Pool + Conv + Pool	64 - 128	2	83.333

2	Conv + Pool +Conv + Pool	64 – 128 - 256	3	84.286
3	Conv + Pool + Conv + Pool	64 – 128 – 256 – 512	4	85.238
4	Conv + Pool + Conv + Pool + Conv + Pool +Conv + Pool	32 - 64 – 128 – 256 - 512	5	83.571
5	Conv + Pool +Conv + Pool	64 – 128 – 256	3	83.810
6	Conv + Conv + Pool + Conv + Conv + Pool	64 – 128 - 128 - 256	4	84.524

The highest accuracy acquired is 85.238 and it is from the third one. Therefore, the third one is selected as the best one.

3.3.2 Model Architecture

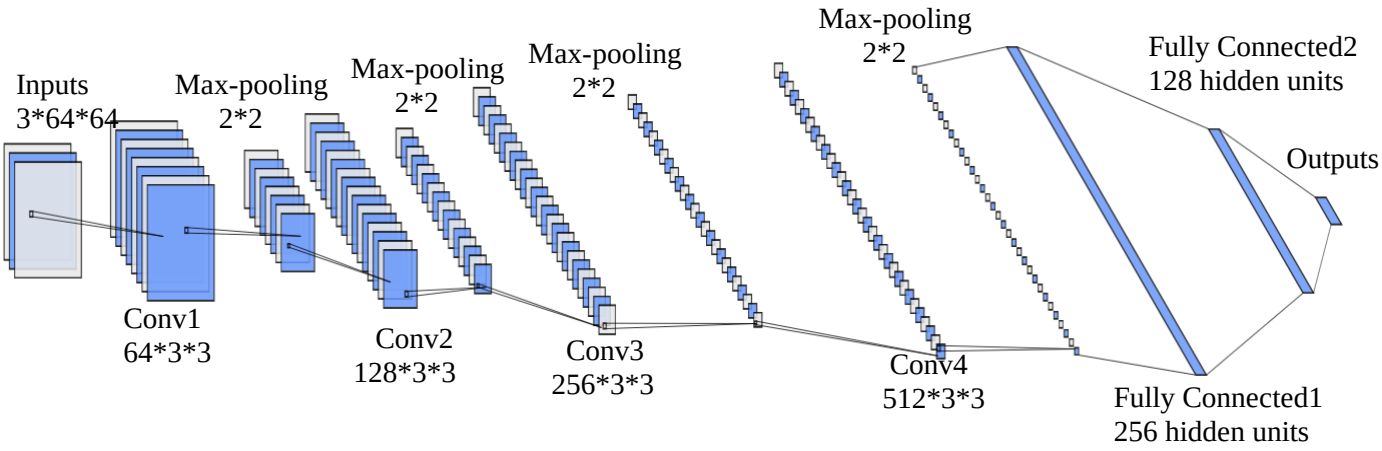


Fig 4. The architecture for the best model

The figure above shows the architecture for the best model. Firstly, the graphs of 3*64*64 pixels as the inputs. Then they are followed by 64*3*3 convolutional layer, 2*2 max-pooling layer, 128*3*3 convolutional layer, 2*2 max-pooling layer, 256*3*3 convolutional layer, 2*2 max-pooling layer, 512*3*3 convolutional layer and 2*2 max-pooling layer. Finally, they are followed by two fully connected layers.

The parameters are as follow:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 64)        1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 256)       295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 512)         1180160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 256)               2097408
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 21)                2709
=================================================================
Total params: 3,683,989
Trainable params: 3,683,989
Non-trainable params: 0
_________________________________________________________________
```

3.3.3 Model Improvement

We found that in this model, there is still overfitting in the fig 5. Therefore, the batch normalization has been implemented to improve the model. Specifically, each convolutional layer is followed by a batch normalization before Relu activation. The value of momentum here is set 0.9. The compiling parameters have been adjusted a little: the number of epochs increased from 1000 to 2000, and the learning rate is 0.0001which is decreased to get higher accuracy.

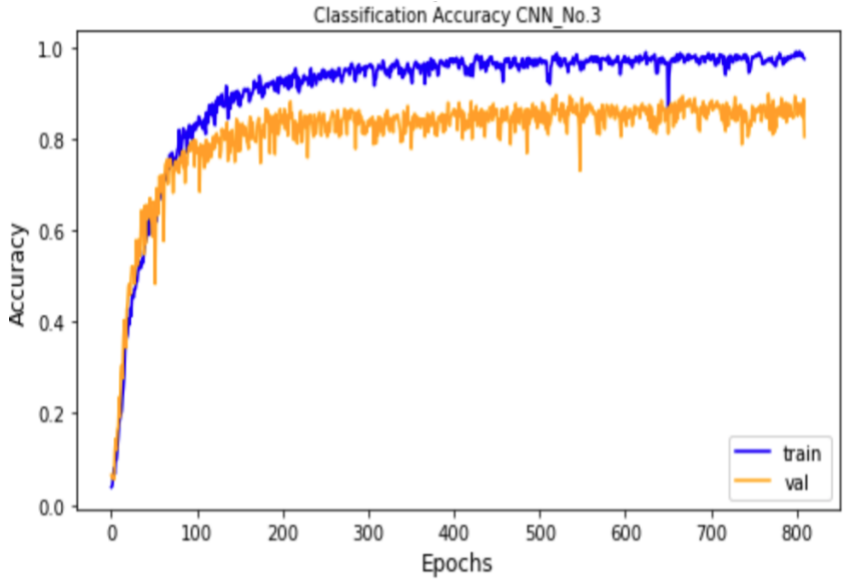


Fig 5. Classification accuracy before batch normalization

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 64, 64, 64)	1792
batch_normalization (BatchNo	(None, 64, 64, 64)	256
activation (Activation)	(None, 64, 64, 64)	0
max_pooling2d (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_1 (Conv2D)	(None, 32, 32, 128)	73856
batch_normalization_1 (Batch	(None, 32, 32, 128)	512
activation_1 (Activation)	(None, 32, 32, 128)	0
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 128)	0
conv2d_2 (Conv2D)	(None, 16, 16, 256)	295168
batch_normalization_2 (Batch	(None, 16, 16, 256)	1024
activation_2 (Activation)	(None, 16, 16, 256)	0
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 256)	0
conv2d_3 (Conv2D)	(None, 8, 8, 512)	1180160
batch_normalization_3 (Batch	(None, 8, 8, 512)	2048
activation_3 (Activation)	(None, 8, 8, 512)	0
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 256)	2097408
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 128)	32896
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 21)	2709

Total params: 3,687,829
Trainable params: 3,685,909
Non-trainable params: 1,920

The total parameters are as above.

4. Results and Analysis

These pictures show the results. The first two are the ones with no batch normalization and the next two are the ones with batch normalization.

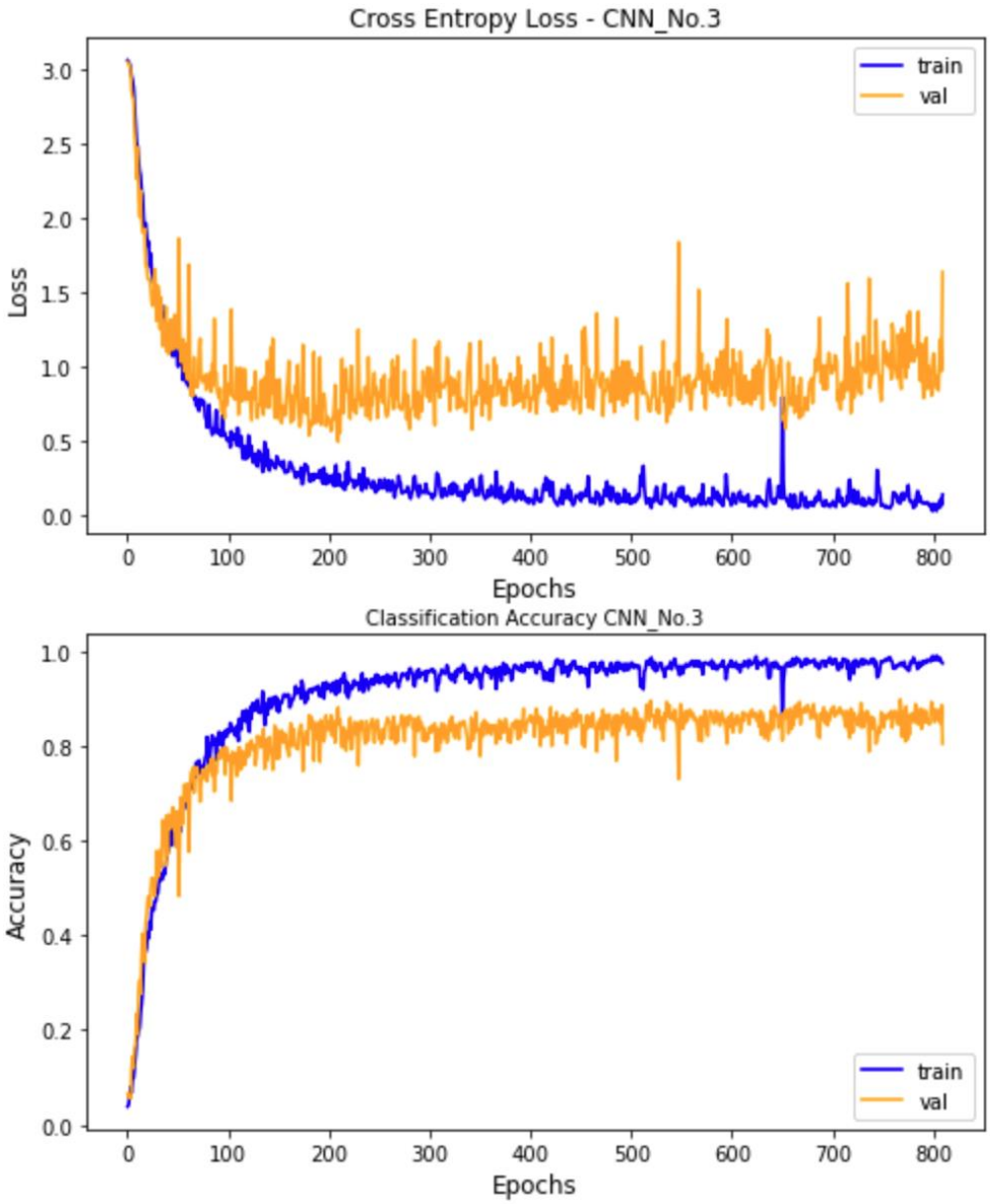


Fig 6. results with no batch normalization

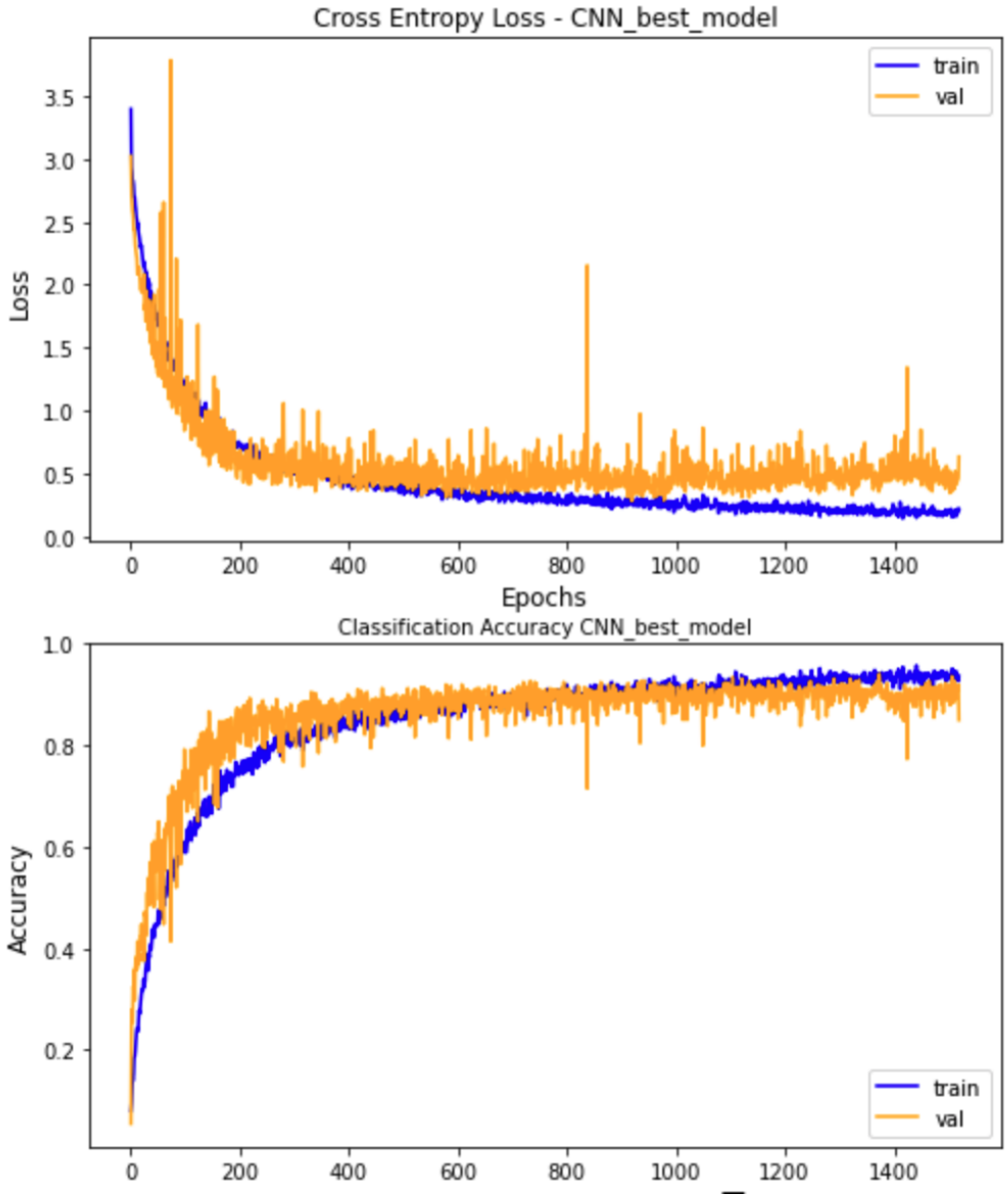


Fig 7. results after batch normalization

From the history plots, we can find that after batch normalization, the problem of overfitting has been eliminated.

Next one shows the classification report. The number of prediction samples is 420. The accuracy of training set is 100%. The validation set is 94.286%. And for the testing set, the micro avg is 0.92 which means the accuracy. It has improved from 85 to 92 after batch normalization.

	precision	recall	f1-score	support
agricultural	0.95	1.00	0.98	20
airplane	0.78	0.70	0.74	20
baseballdiamond	1.00	1.00	1.00	20
beach	0.95	0.95	0.95	20
buildings	1.00	1.00	1.00	20
chaparral	0.94	0.85	0.89	20
denseresidential	1.00	1.00	1.00	20
forest	0.95	0.95	0.95	20
freeway	0.83	0.95	0.88	20
golfcourse	0.95	1.00	0.98	20
harbor	1.00	1.00	1.00	20
intersection	0.95	0.90	0.92	20
mediumresidential	0.95	0.90	0.92	20
mobilehomepark	0.95	1.00	0.98	20
overpass	1.00	1.00	1.00	20
parkinglot	0.84	0.80	0.82	20
river	0.74	0.85	0.79	20
runway	0.71	0.75	0.73	20
sparseresidential	1.00	1.00	1.00	20
storagetanks	0.95	0.95	0.95	20
tenniscourt	0.82	0.70	0.76	20
micro avg	0.92	0.92	0.92	420
macro avg	0.92	0.92	0.92	420
weighted avg	0.92	0.92	0.92	420
samples avg	0.92	0.92	0.92	420

Fig 8. Classification report

This one shows the confusion matrix. We can acquire specific results of prediction for each image. Here, we take `building` and `airplane` as examples.

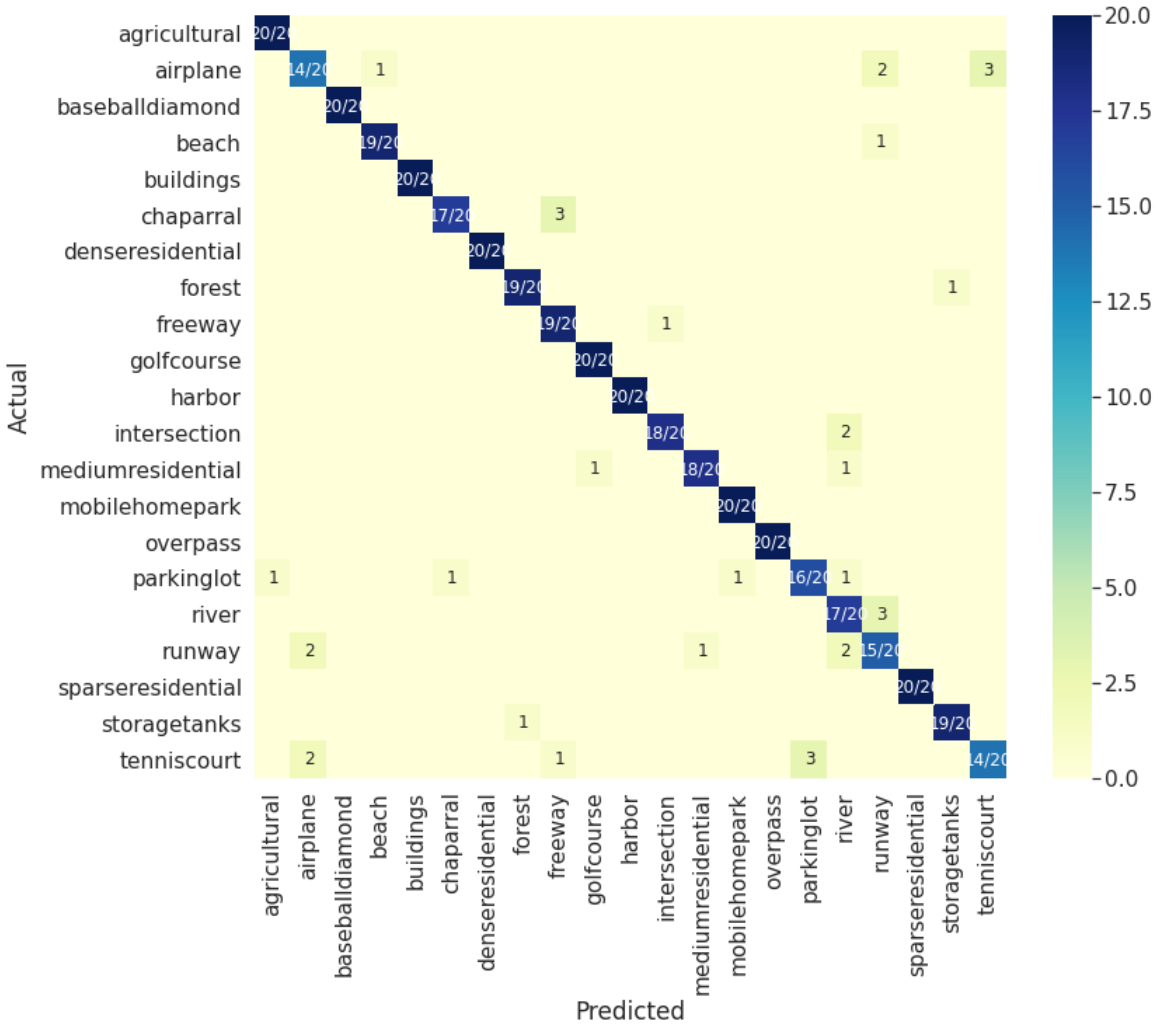


Fig 9. Confusion matrix

The next two pictures represent images prediction results for the type of `building` and `airplane`. Images surrounded by red rectangle means wrong prediction. In the building dataset, all of predictions are correct. However, in airplane ones, 6 wrong predictions occurred and four of them are predicted into runways and tennis courts.

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: beach

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: tenniscourt

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: runway

LABEL:airplane
PRED: tenniscourt

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: runway

LABEL:airplane
PRED: airplane

LABEL:airplane
PRED: tenniscourt

LABEL:airplane
PRED: airplane

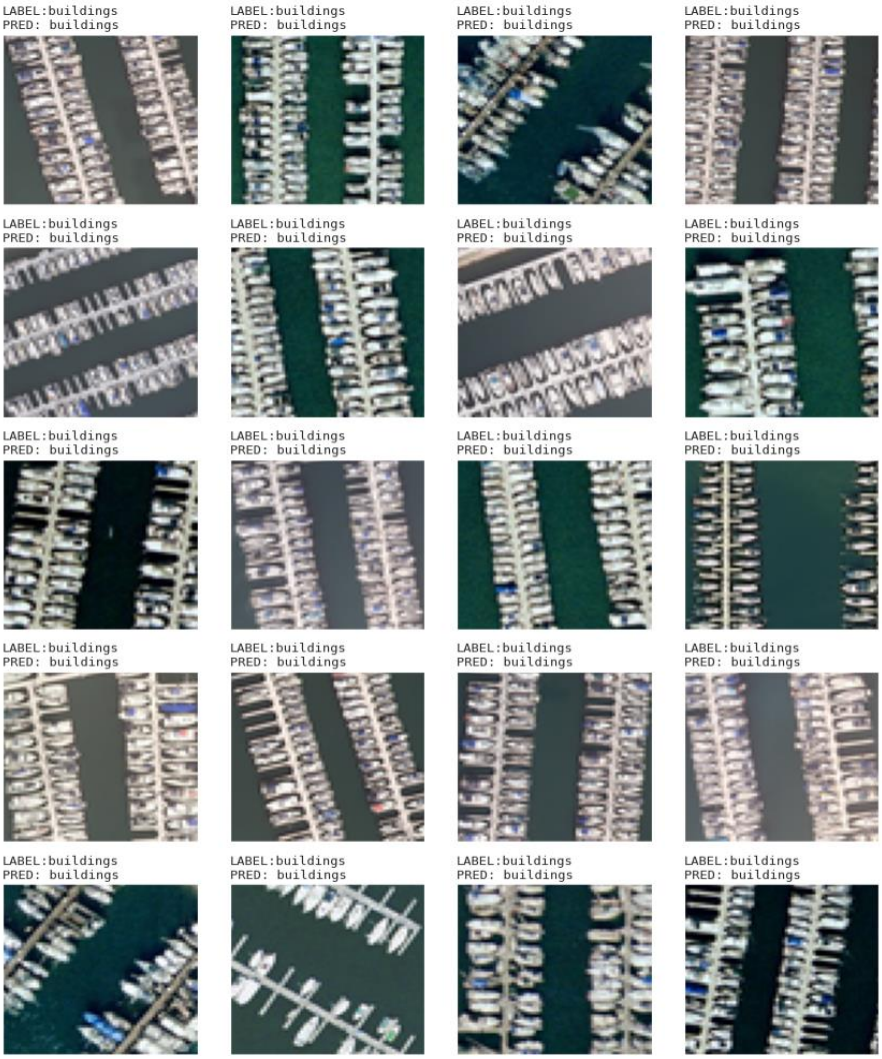


Fig 10. Image visualization for airplane and building

6. Transfer Learning

In this project, transfer learning has also been tried. The selected pretrained model is VGG and the model has been extracted without fully connected layers, in which keep original parameters of this model except for the fc layers. Therefore, the parameter of `trainable` is false.

To fit into the pretrained model, there are some parameters needed to be adjusted. The images are resized into (224, 224), the learning rate is 0.0001 and the number of epochs is 10.

Here are the total parameters for transfer learning.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
flatten_1 (Flatten)	(None, 25088)	0
dense_2 (Dense)	(None, 1024)	25691136
dense_3 (Dense)	(None, 21)	21525

Total params: 40,427,349
Trainable params: 25,712,661
Non-trainable params: 14,714,688

The following shows the classification report for the results of transfer learning. The final accuracy is 0.8 which is less than the one in our customized model.

	precision	recall	f1-score	support
agricultural	0.95	0.90	0.92	20
airplane	1.00	1.00	1.00	20
baseballdiamond	0.89	0.85	0.87	20
beach	0.77	1.00	0.87	20
buildings	0.77	0.85	0.81	20
chaparral	1.00	0.95	0.97	20
denseresidential	0.65	0.55	0.59	20
forest	0.90	0.95	0.93	20
freeway	0.95	1.00	0.98	20
golfcourse	0.89	0.80	0.84	20
harbor	1.00	1.00	1.00	20
intersection	0.90	0.95	0.93	20
mediumresidential	0.76	0.80	0.78	20
mobilehomepark	0.67	0.60	0.63	20
overpass	1.00	0.90	0.95	20
parkinglot	0.95	1.00	0.98	20
river	0.89	0.80	0.84	20
runway	0.91	1.00	0.95	20
sparseresidential	0.86	0.95	0.90	20
storagetanks	0.94	0.75	0.83	20
tenniscourt	0.76	0.80	0.78	20
micro avg	0.88	0.88	0.88	420
macro avg	0.88	0.88	0.87	420
weighted avg	0.88	0.88	0.87	420
samples avg	0.88	0.88	0.88	420

Fig 11. Classification report for transfer learning models

7. Conclusions

In this project, a remote sensing image classification system has been conducted by selecting the best model from different architectures and the accuracy reaches up to 92%. To prevent eliminating, here we used batch normalization and data augmentation and it improved accuracy a lot. Finally, I performed transfer learning by extracting VGG without fully connected layers and the test accuracy is 88%.

8. Limitations and Later Work

There are some limitations that can be improved later.

Firstly, the data size is relatively small and less is considered on the ways of data augmentation. Secondly, for transfer learning, only trained fully connected layers without pretrained layers to train this model. Besides, we lack the discussion of different hyperparameters.

Therefore, the model can be explored more in these ways. Firstly, we can try different ways of data augmentation and optimize hyperparameters to train models. Secondly, we can explore other pretrained models and train all layers to improve the accuracy.

9. References

[1] Song, J., Gao, S., Zhu, Y., & Ma, C. (2019). A survey of remote sensing image classification based on CNNs. Big earth data, 3(3), 232-254.
[2] Shafaey, M. A., Salem, M. A. M., Al-Berry, M. N., Ebied, H. M., & Tolba, M. F. (2020, April). Remote Sensing Image Classification Based on Convolutional Neural Networks. In Joint European-US Workshop on Applications of Invariance in Computer Vision (pp. 353-361). Springer, Cham.
[3]Zhang, W., Tang, P., & Zhao, L. (2019). Remote sensing image scene classification using CNN-CapsNet. Remote Sensing, 11(5), 494.
[4]Li, R., Zheng, S., & Duan, C. (2020). Land cover classification from remote sensing images based on multi-scale fully convolutional network. arXiv preprint arXiv:2008.00168.
[5] ("7 Popular Image Classification Models in ImageNet Challenge (ILSVRC) Competition History | MLK - Machine Learning Knowledge", 2021)

10. Appendixes

https://github.com/carrieqing/capstone_project/