Satellite Image Classification

Grace Yu

12/5/2020

Motivation

Build a machine learning framework to differentiate any satellite image by trained models

Highlights

- Different machine learning algorithms including deep neural network, CNN, and GANs models were deployed on a huge satellite image dataset.
- Model architecture and performance were explicitly evaluated.
- The best model, CNN model, was applied to classify new images successfully.

Data Sources

- Original Paper

Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert Dibiano, Manohar Karki and Ramakrishna Nemani, DeepSat - A Learning framework for Satellite Imagery, ACM SIGSPATIAL 2015.

http://csc.lsu.edu/~saikat/deepsat/

- Satellite Images CSV file

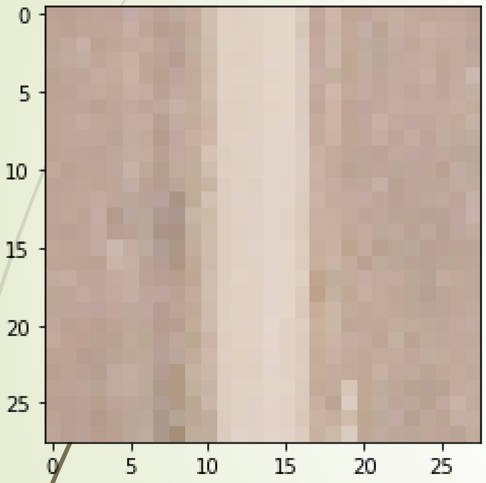
https://www.kaggle.com/arpandhatt/satellite-image-classification

Exploratory Data Analysis

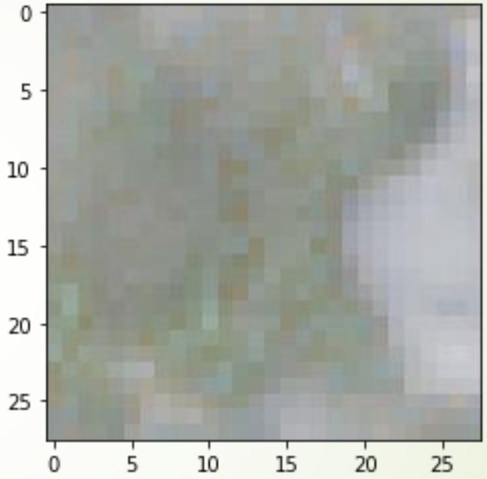
SAT-4 Image: 500,000, training: 400,000 (4/5), test: 100,000 (1/5)

Four categories: barren land, trees, grassland, other (in this order)

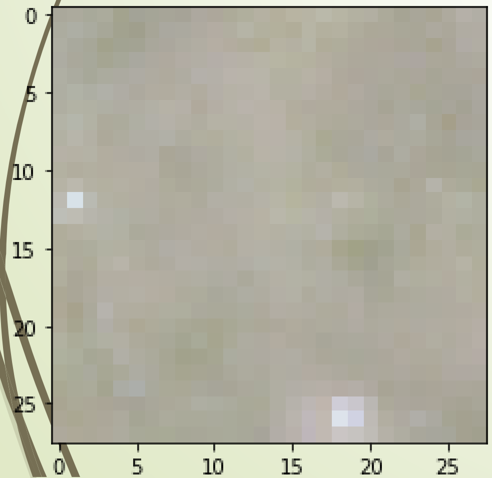
Each image: 28x28x4 (RGB and Infra).



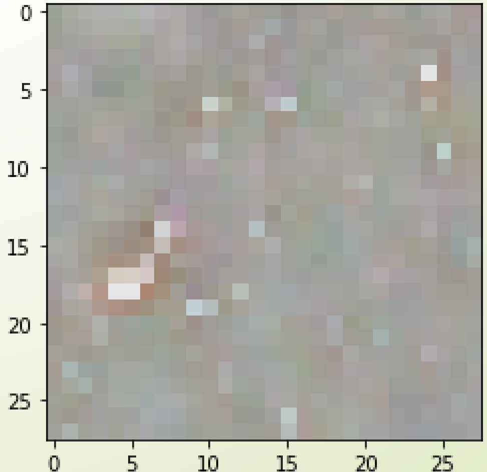
Barren land
(label: 0)
(104,465)



Trees
(label: 1)
(81,118)

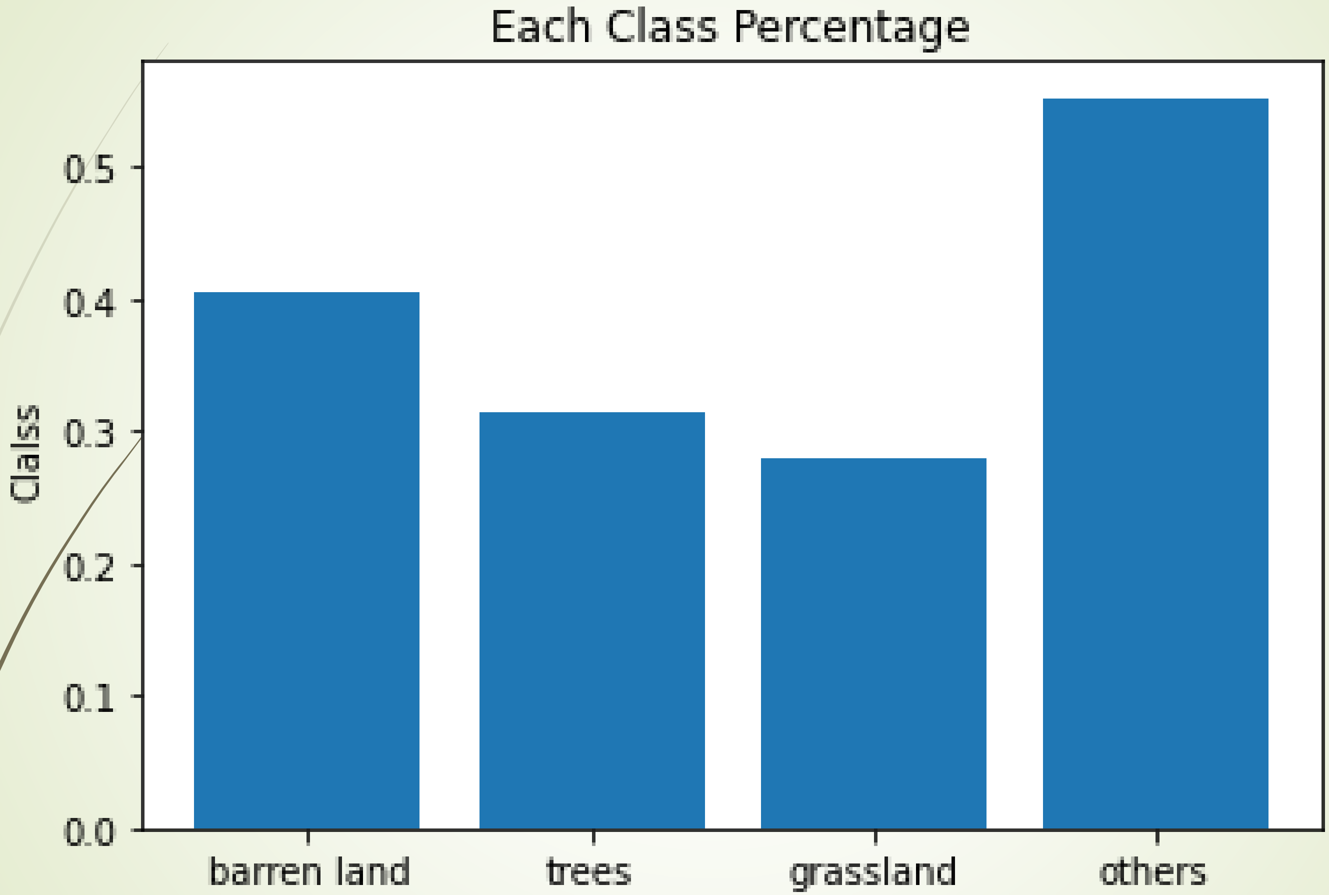


Grassland
(lable: 2)
(72,017)



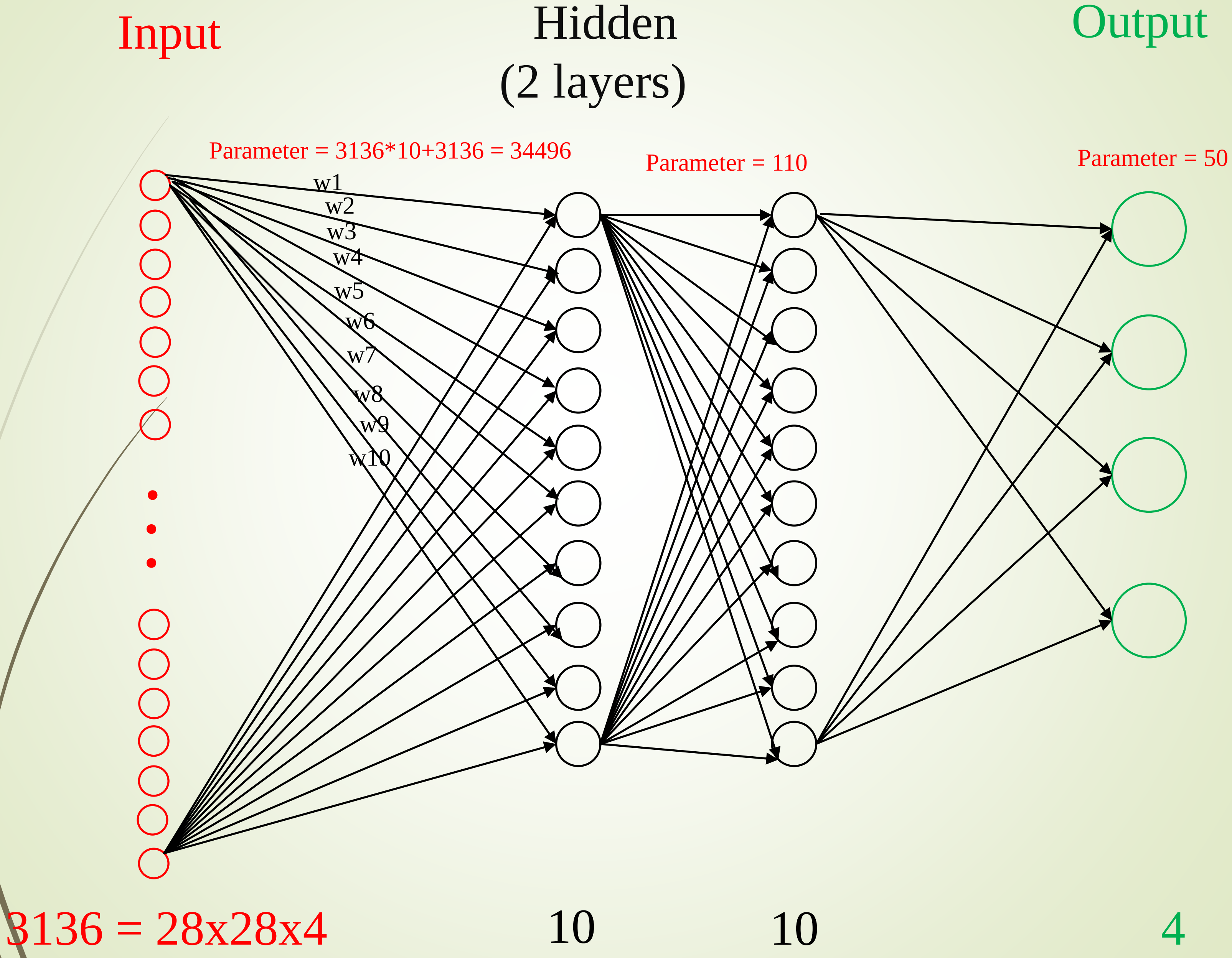
Other
(lable: 3)
(142,400)

Samples are balanced



0 1 2 3

Model 1 - Deep Neural Network



Model Architecture

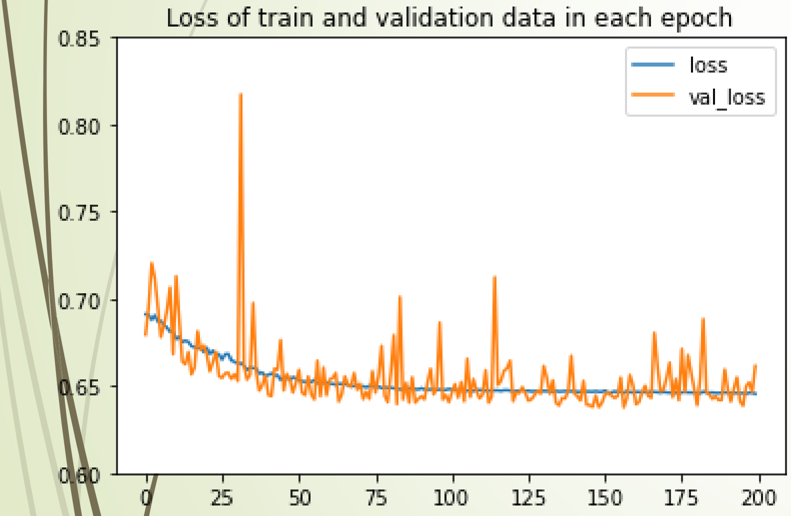
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=======================================================
==========
dense (Dense)                  (None, 10)                31370
_________________________________________________________________
dense_1 (Dense)                (None, 10)                 110
_________________________________________________________________
dense_2 (Dense)                (None, 4)                 44
=======================================================
==========
Total params: 31,524
Trainable params: 31,524
Non-trainable params: 0
```

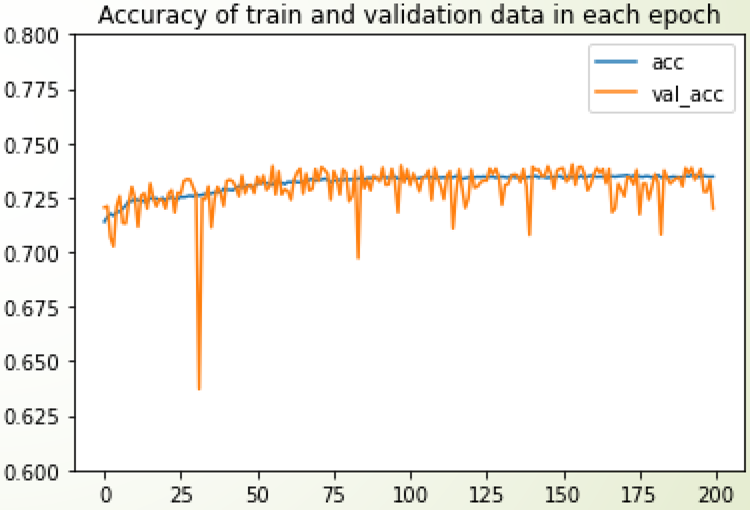
Model Performance

Training Data

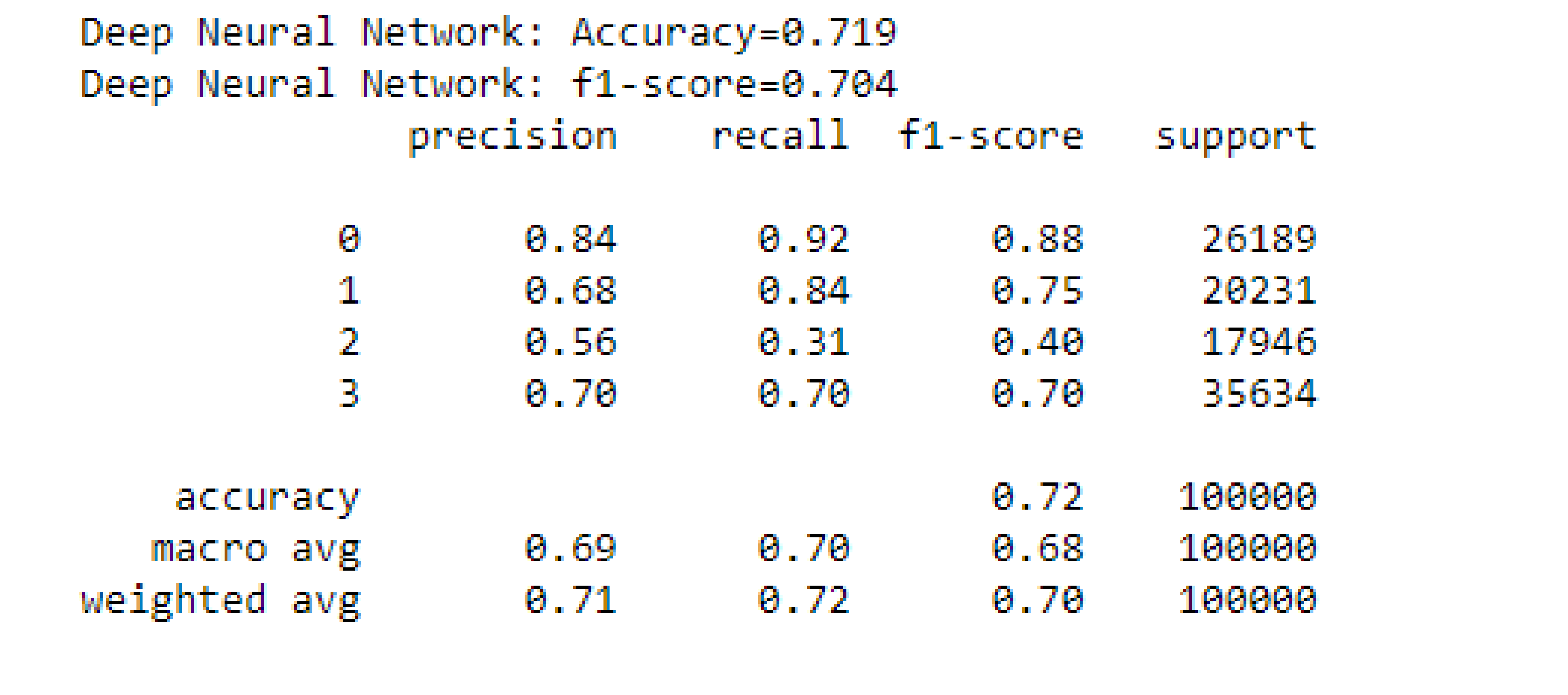
Loss Function

Accuracy



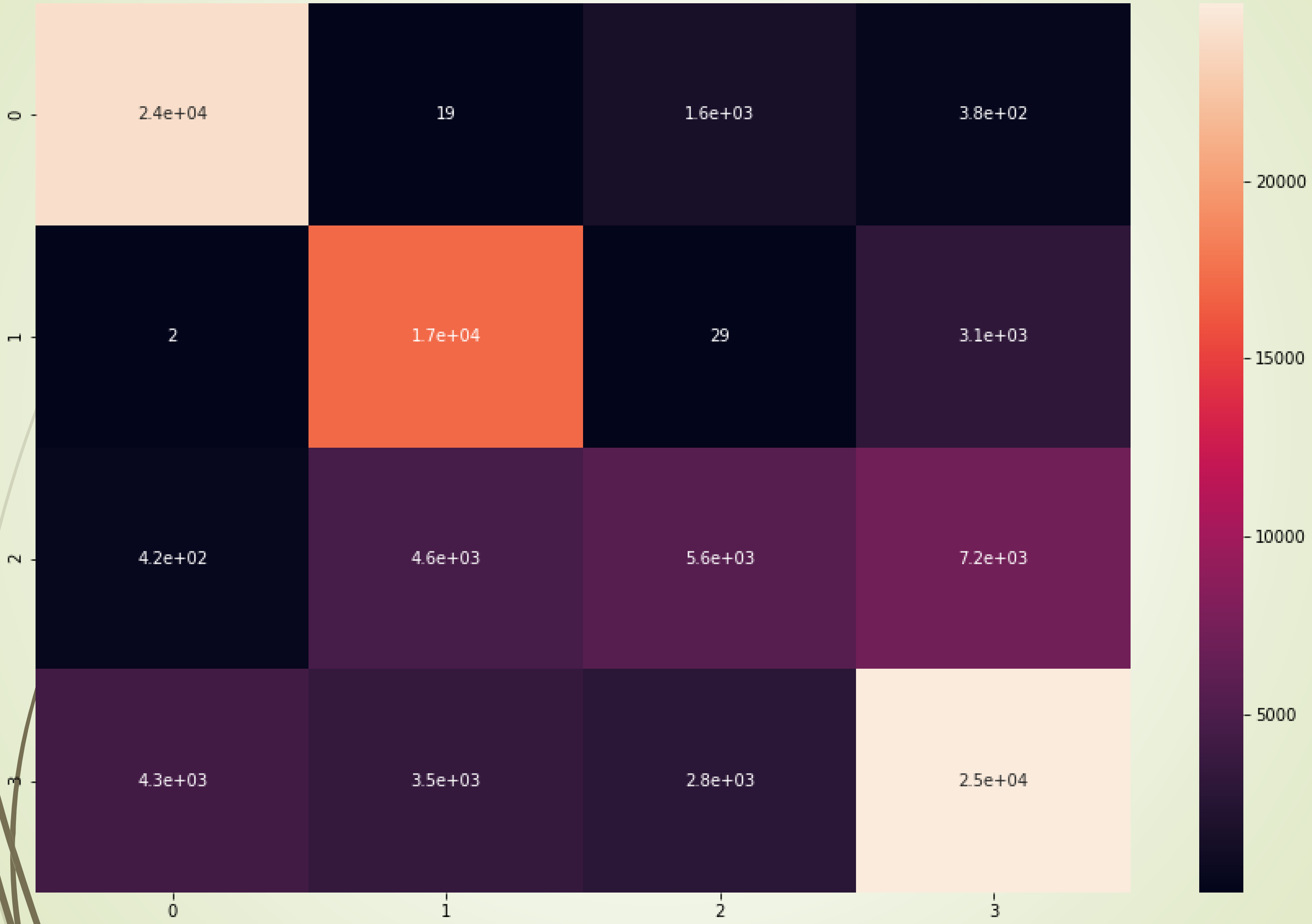


Model Evaluation



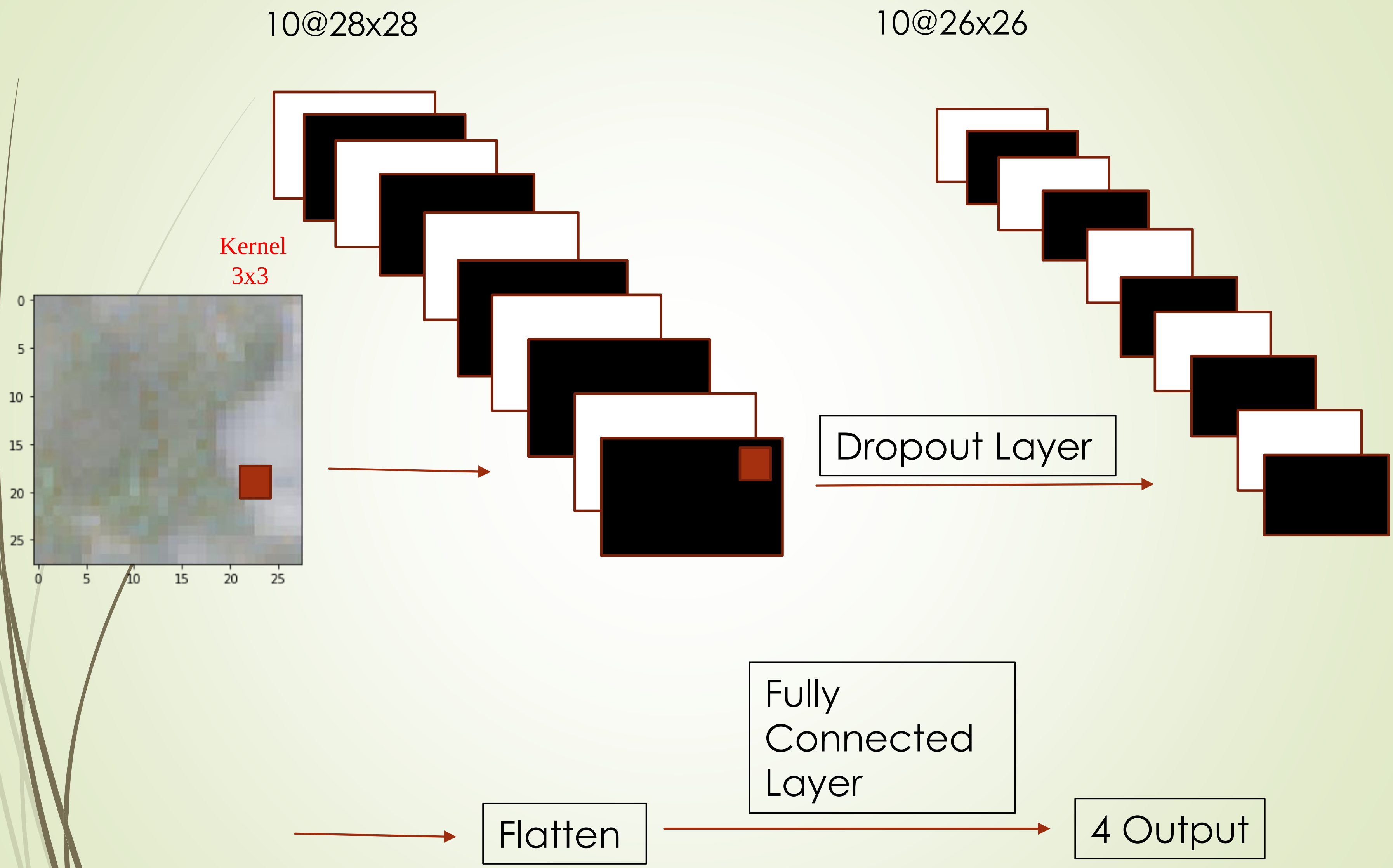
```
Deep Neural Network: Accuracy=0.719
Deep Neural Network: f1-score=0.704
              precision    recall  f1-score   support

           0       0.84      0.92      0.88     26189
           1       0.68      0.84      0.75     20231
           2       0.56      0.31      0.40     17946
           3       0.70      0.70      0.70     35634

    accuracy                           0.72    100000
   macro avg       0.69      0.70      0.68    100000
weighted avg       0.71      0.72      0.70    100000
```

Confusion Matrix



	0	1	2	3
0	2.4e+04	19	1.6e+03	3.8e+02
1	2	1.7e+04	29	3.1e+03
2	4.2e+02	4.6e+03	5.6e+03	7.2e+03
3	4.3e+03	3.5e+03	2.8e+03	2.5e+04

Model 2 – Convolutional Neural Network (CNN)



10@28x28

10@26x26

Kernel 3x3

Dropout Layer

Flatten

Fully Connected Layer

4 Output

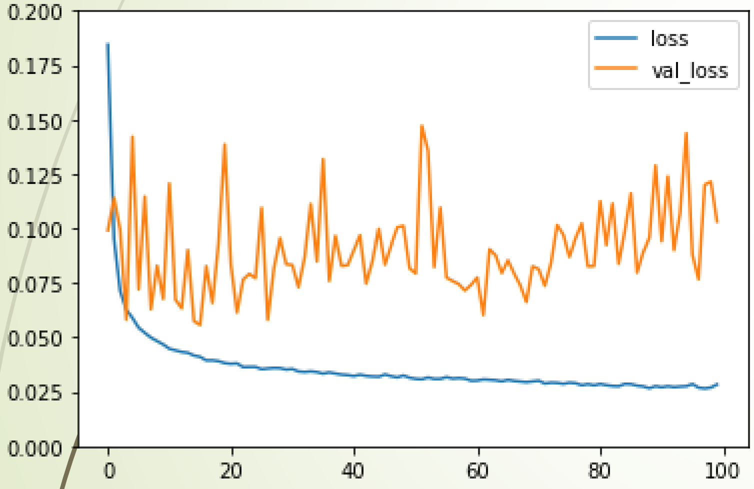
Model Architecture

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 10)        370
_________________________________________________________________
dropout (Dropout)            (None, 28, 28, 10)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 26, 26, 10)        910
_________________________________________________________________
flatten (Flatten)            (None, 6760)              0
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 27044
=================================================================
Total params: 28,324
Trainable params: 28,324
Non-trainable params: 0
_________________________________________________________________
```

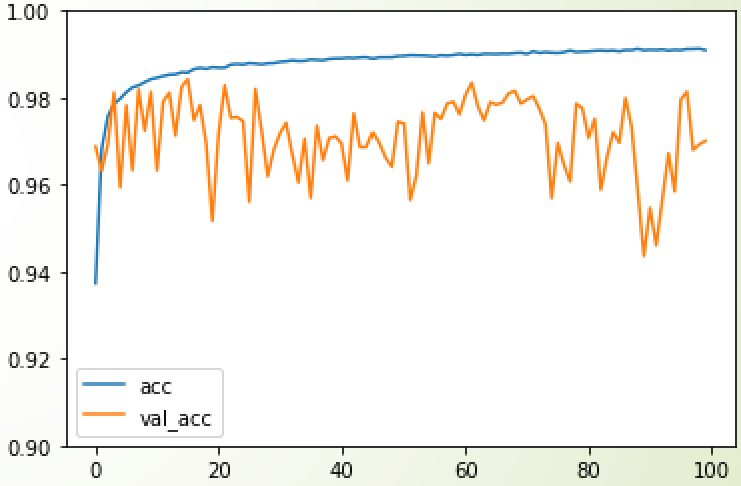
Model Performance

Training Data

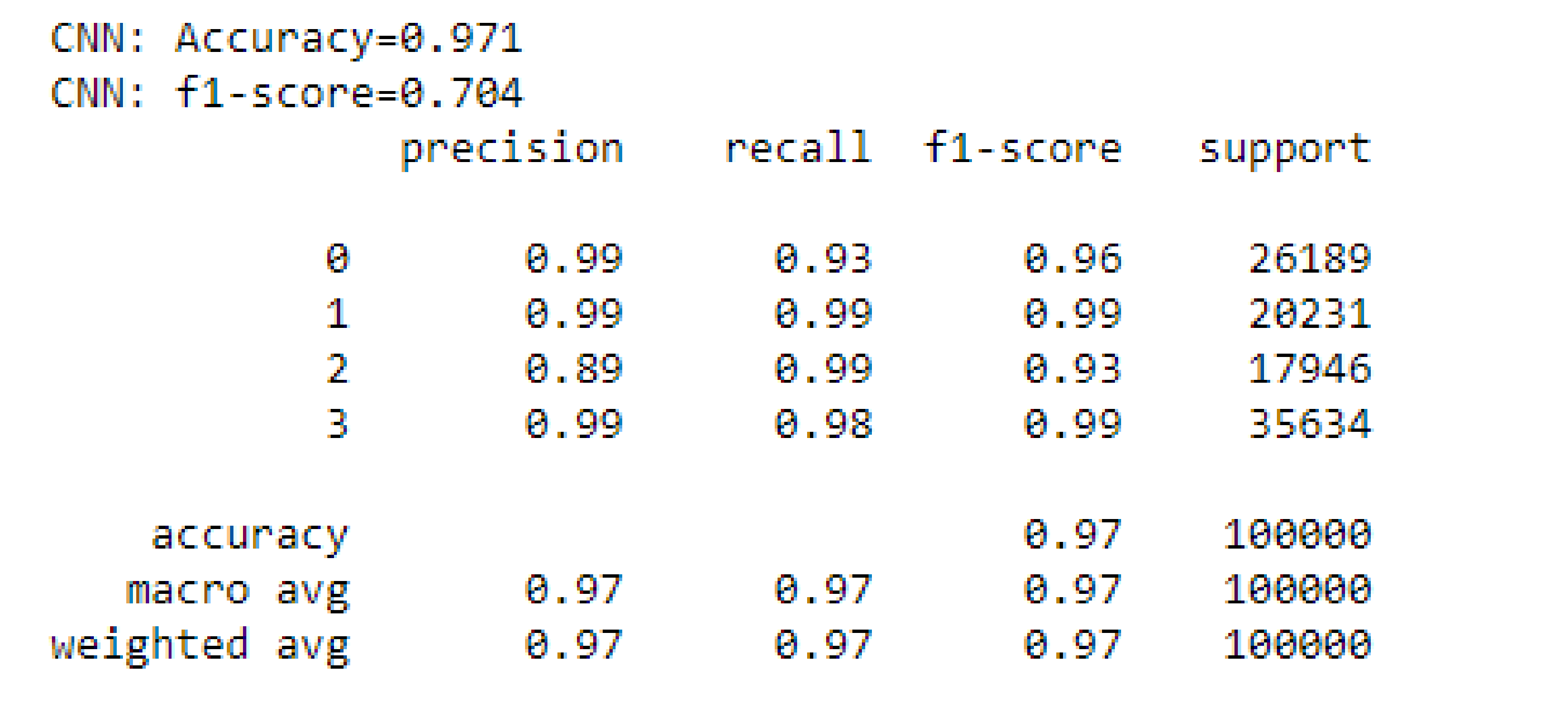
Loss Function



Accuracy

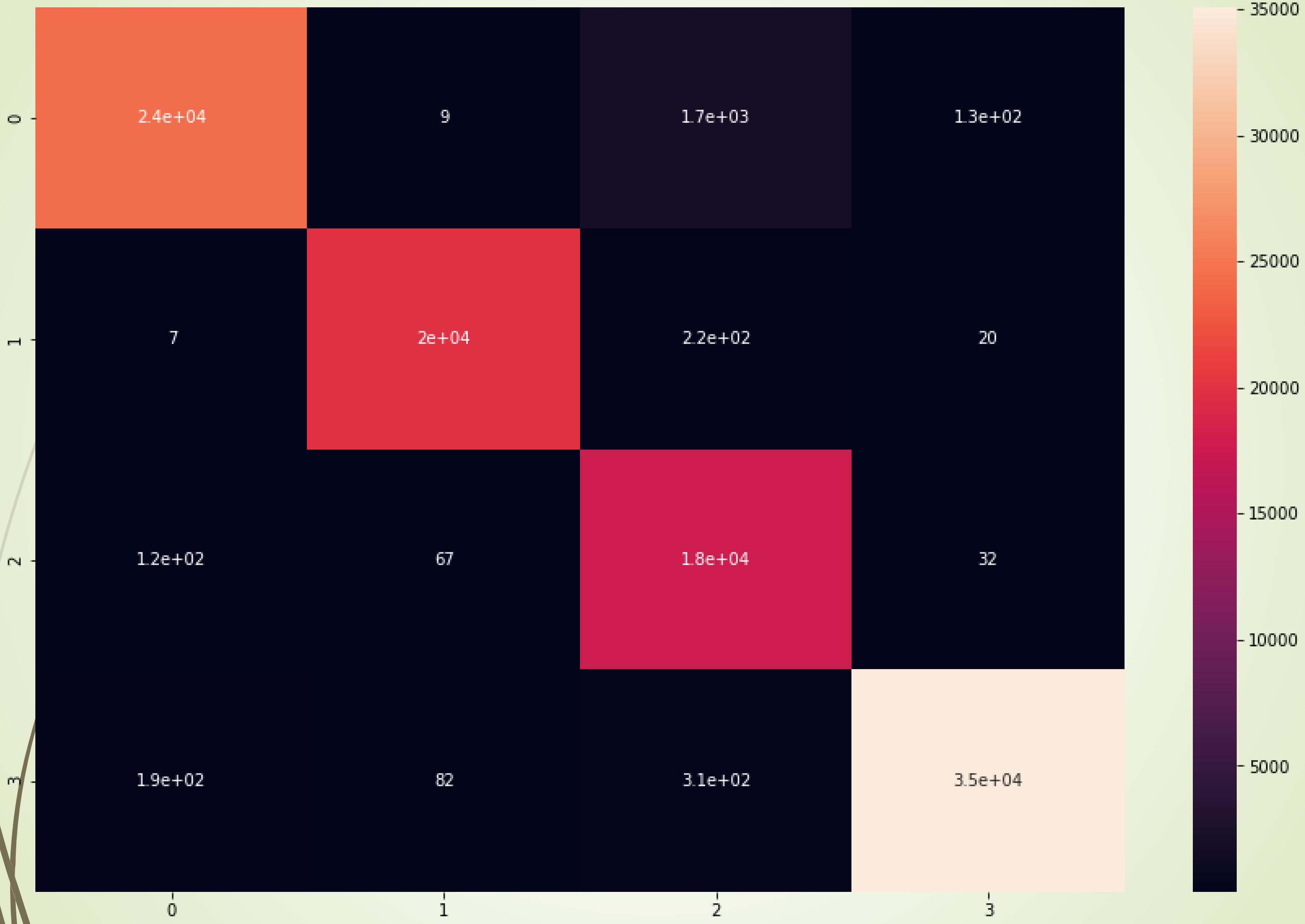


Model Evaluation



```
CNN: Accuracy=0.971
CNN: f1-score=0.704
              precision    recall  f1-score   support

           0       0.99      0.93      0.96     26189
           1       0.99      0.99      0.99     20231
           2       0.89      0.99      0.93     17946
           3       0.99      0.98      0.99     35634

    accuracy                           0.97    100000
   macro avg       0.97      0.97      0.97    100000
weighted avg       0.97      0.97      0.97    100000
```

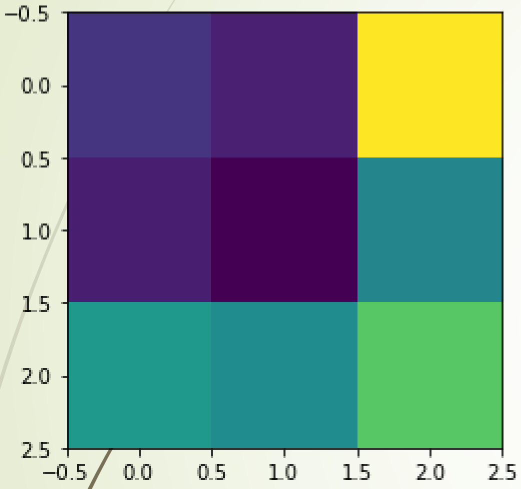
Confusion Matrix



	0	1	2	3
0	2.4e+04	9	1.7e+03	1.3e+02
1	7	2e+04	2.2e+02	20
2	1.2e+02	67	1.8e+04	32
3	1.9e+02	82	3.1e+02	3.5e+04

Model Interpretation

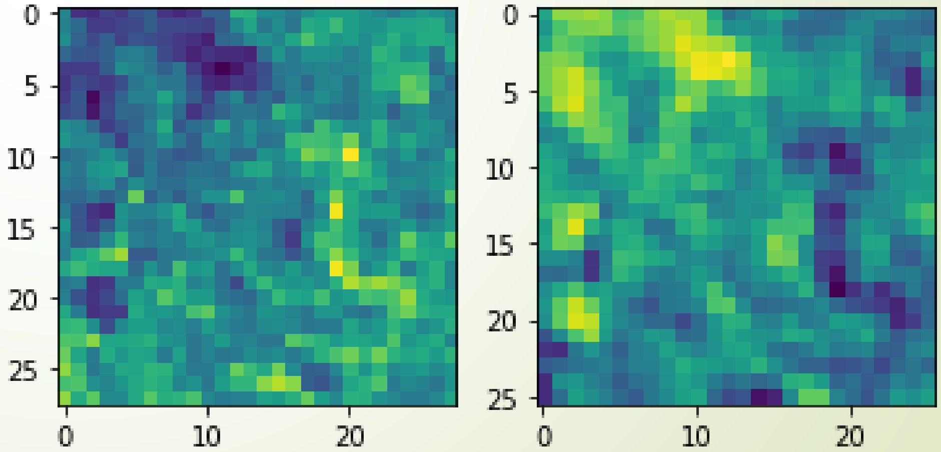
Kernel – Layer1



Image

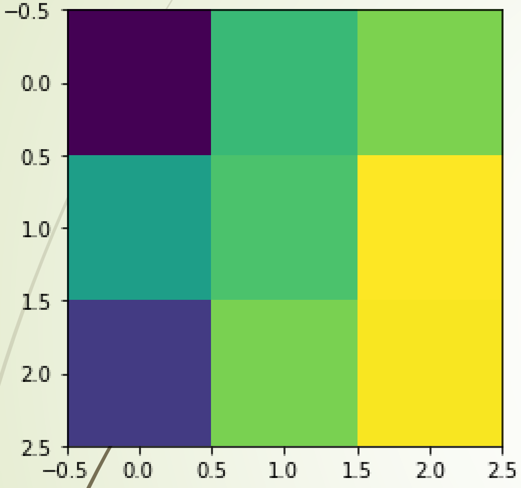
Before

After



Model Interpretation

Kernel – Layer2



Image

Before

After

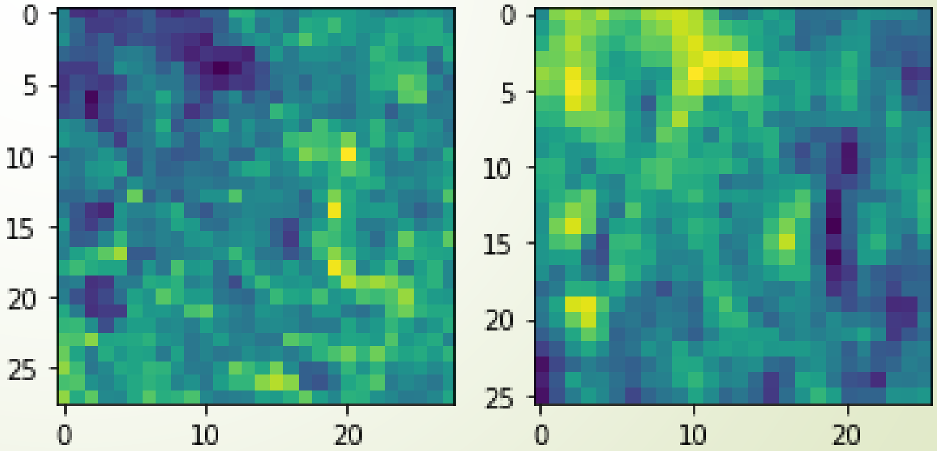
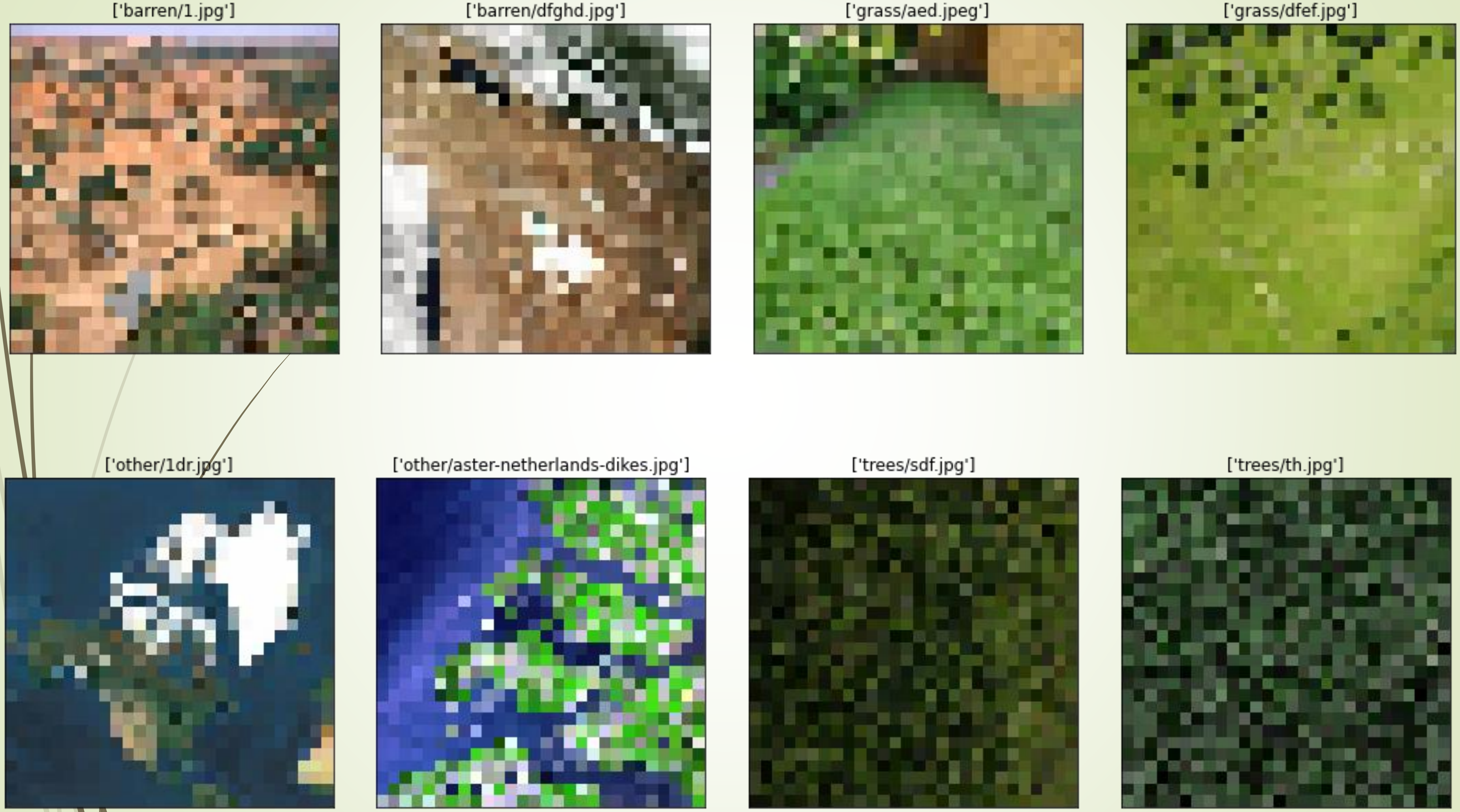


Image for Testing



['barren/1.jpg']

['barren/dfghd.jpg']

['grass/aed.jpeg']

['grass/dfef.jpg']

['other/1dr.jpg']

['other/aster-netherlands-dikes.jpg']

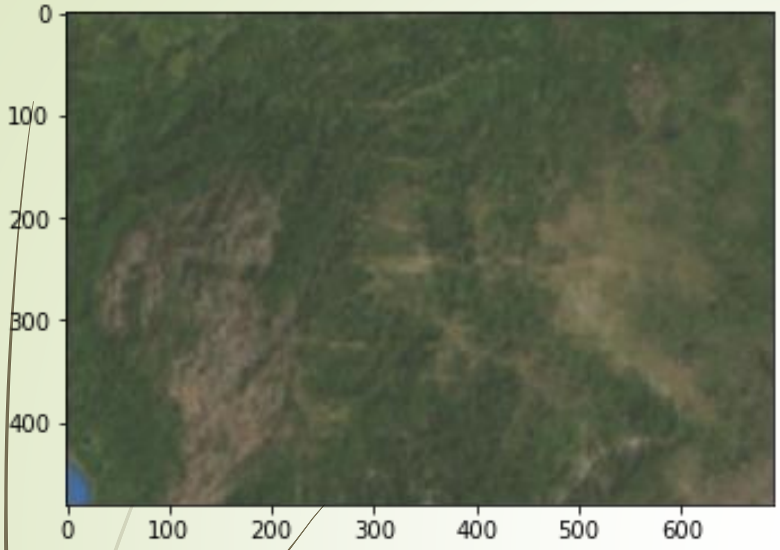
['trees/sdf.jpg']

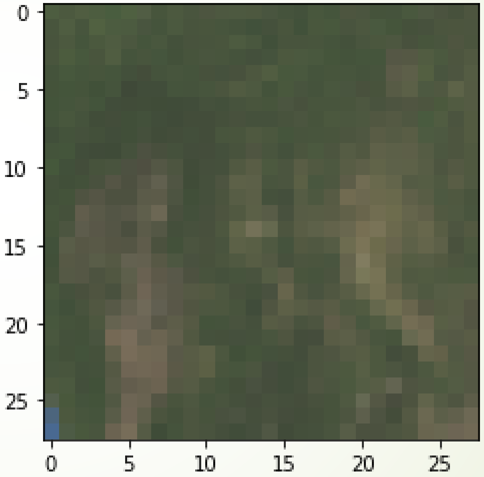
['trees/th.jpg']

Image Prediction

- Download 9 satellite images from internet
- Feed them into the CNN model and predict
- Results: Predict all 9 images are others.
- The problem here is all input data is 3 color channels, I artificially add 4th color channel as zero or mean of 3 channels. I think this cause the CNN model confused.

Results

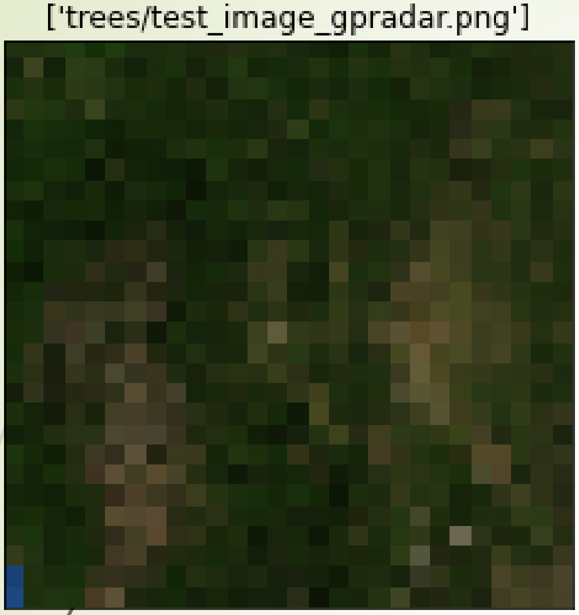




Prediction: 2 - Grassland

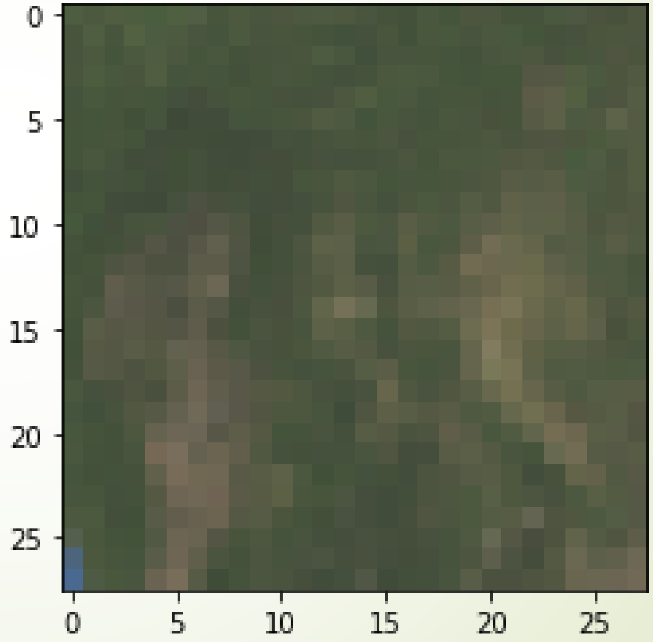
Discussion

3 channels



Information loss

4 channels



Prediction: 3 – Other

Prediction: 2 – Grassland

Summary

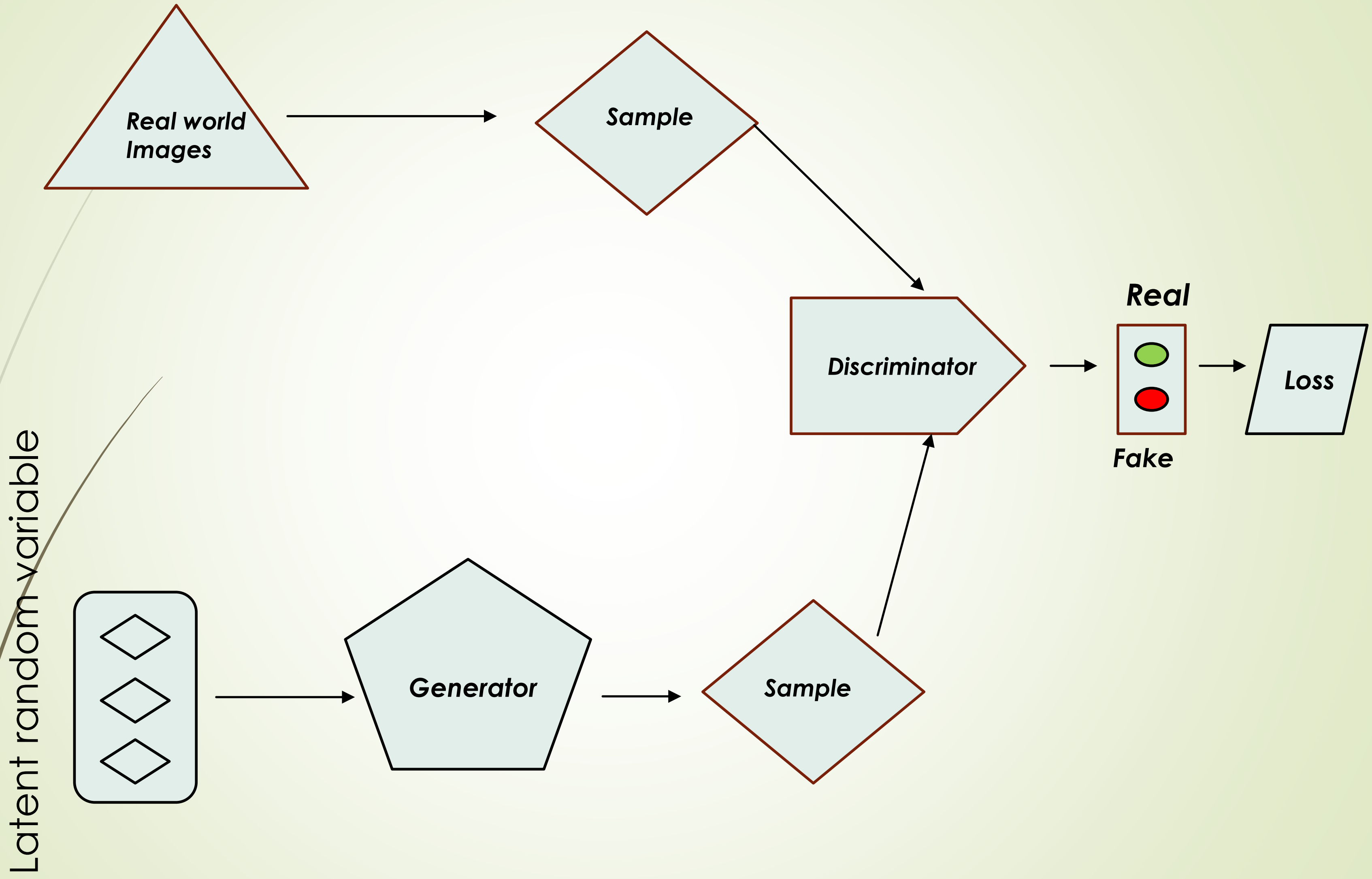
- Trained a large dataset using general deep neural network and CNN model, CNN is much better in terms of accuracy
- Tested model on random pictures. It demonstrated the importance to have the infra color information. Samples need to be prepared and pre-processed to match the training input for more accurate prediction.

Future Work

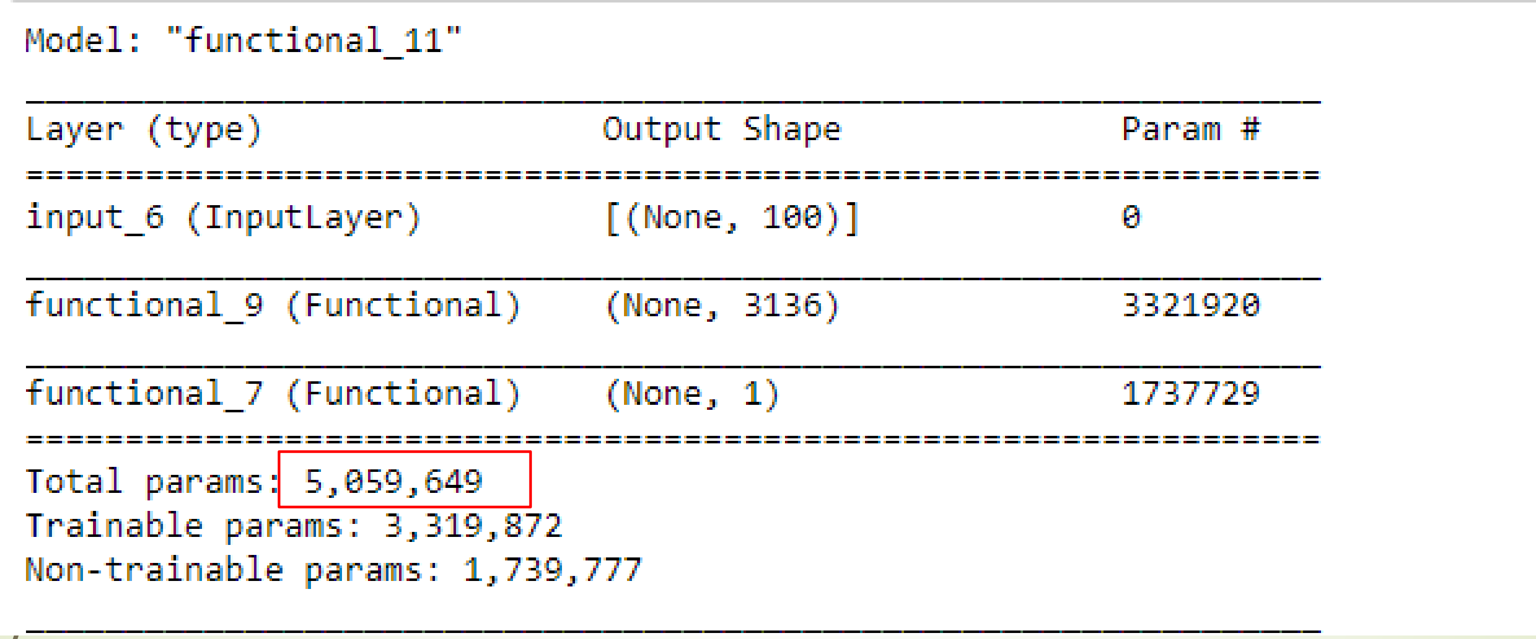
- Label more features, can train SAT-6 data;
- Train model with 3 color channels only;
- Use satellite image with more channels to train classification model;
- Try U-Net to do object detection;
- Test GANs model to generate synthetic images for training.

Preliminary GANs Results

Generative Adversarial Networks (GANs)



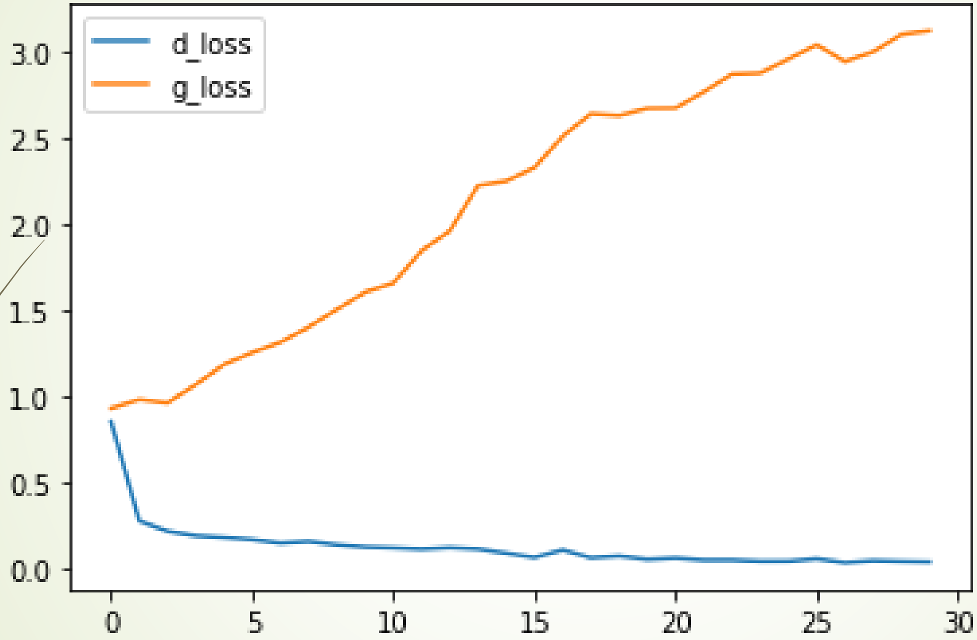
Model Architecture



```
Model: "functional_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
functional_9 (Functional)    (None, 3136)              3321920
_________________________________________________________________
functional_7 (Functional)    (None, 1)                 1737729
=================================================================
Total params: 5,059,649
Trainable params: 3,319,872
Non-trainable params: 1,739,777
_________________________________________________________________
```

Model Performance

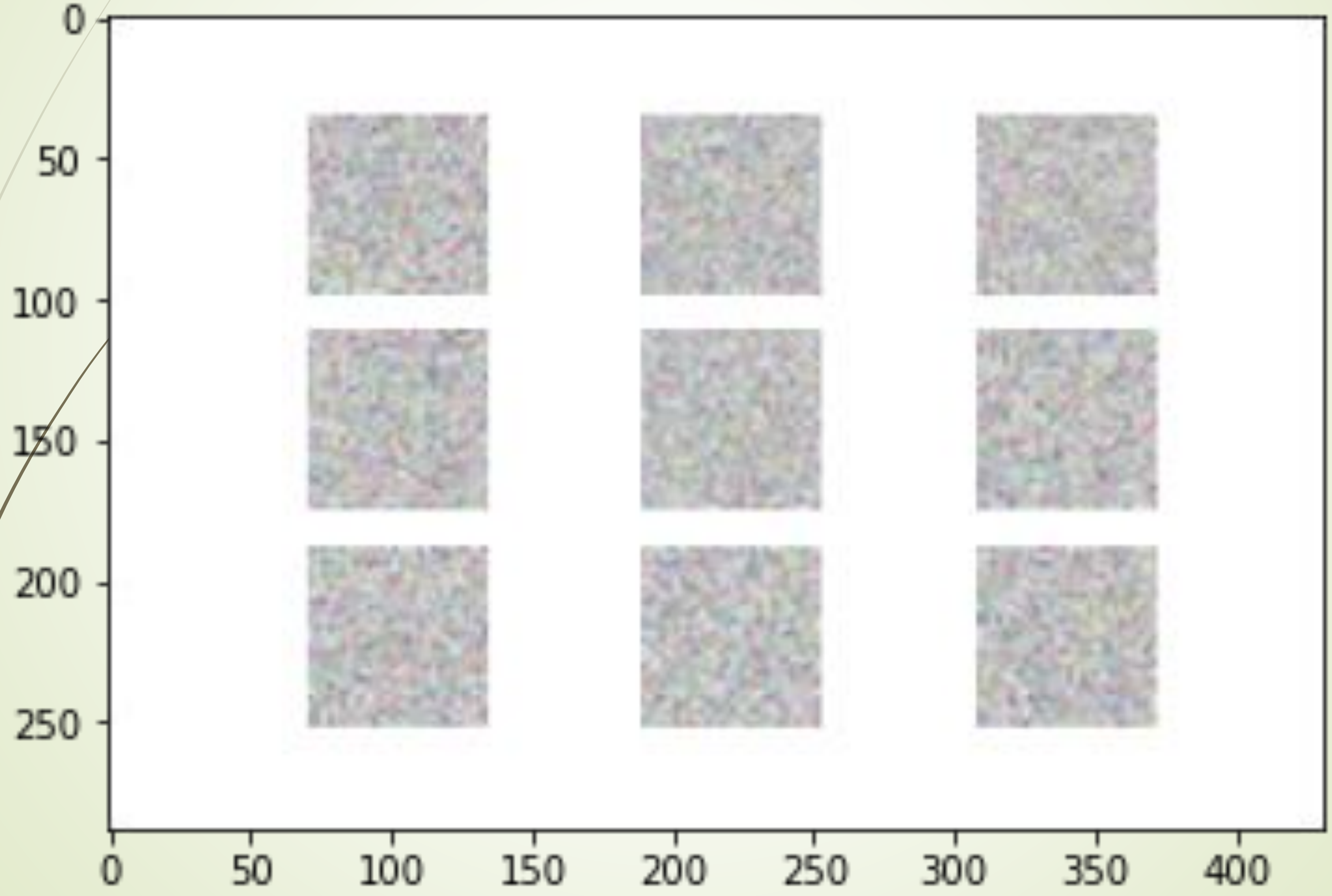
Loss Function



Discriminator converge very quickly, while generator is far from converging. Only run 300 epochs, need more epochs. Need cloud computation or GPU.

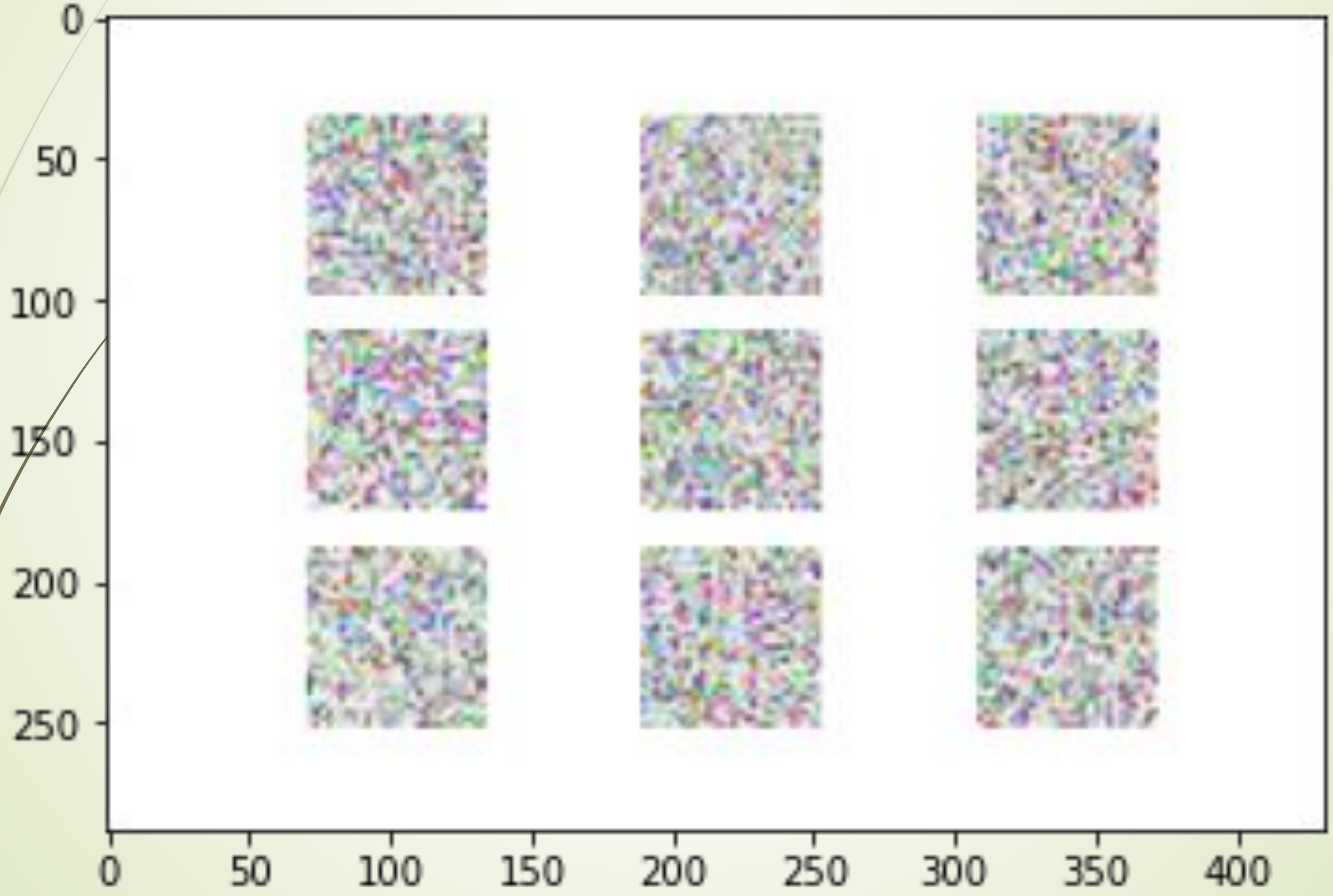
Synthetic Images

Beginning



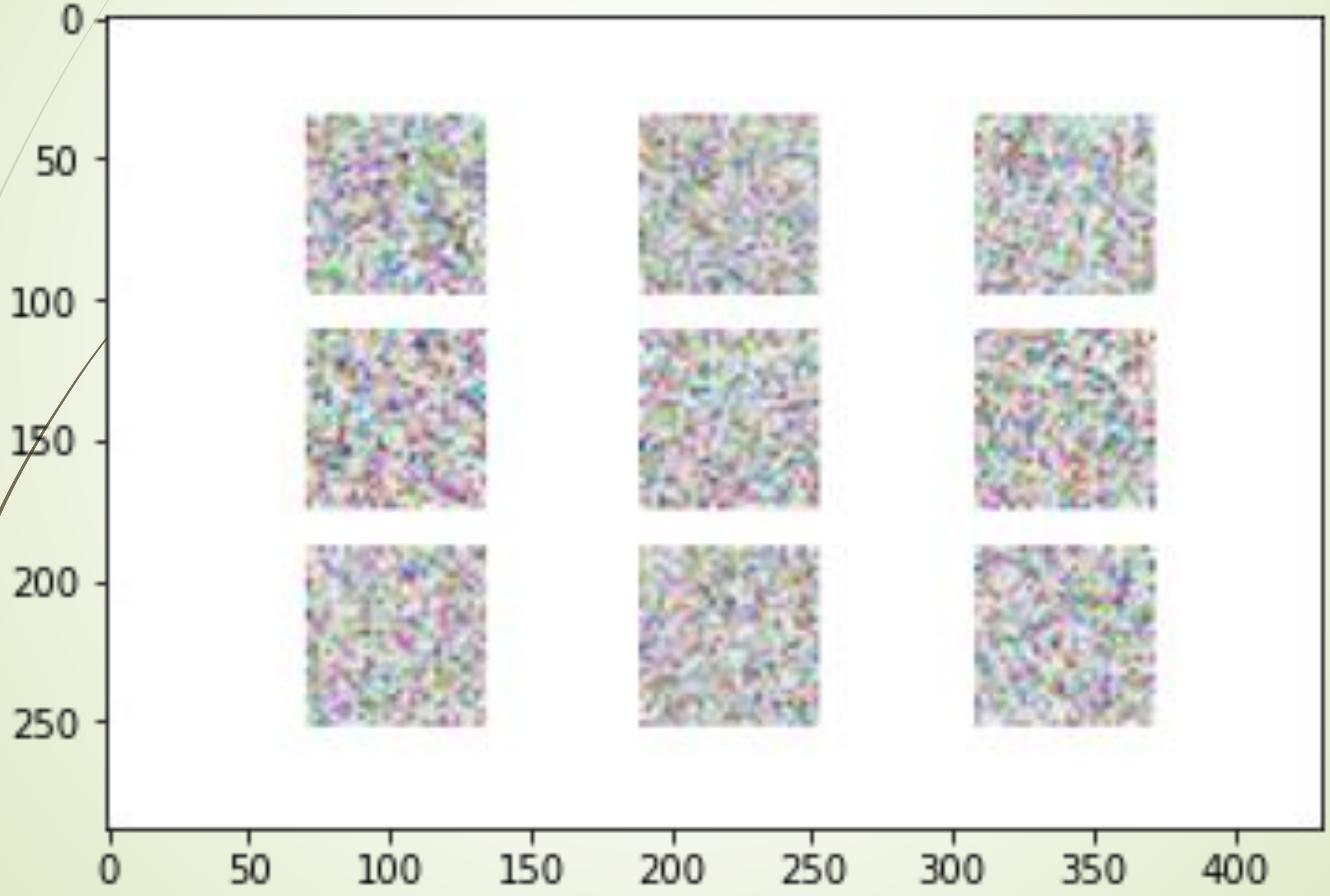
Synthetic Images

Middle



Synthetic Images

End


0
50
100
150
200
250
0
50
100
150
200
250
300
350
400