Land Use Classification Project Results

THAI LA, University of Saskatchewan, Canada

Additional Key Words and Phrases: Satellite, Image, Classification, LandUse, GeoSpatial

ACM Reference Format:
Thai La. 2023. Land Use Classification Project Results. 1, 1 (December 2023), 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

This report detail the results, the hyperparameter study for tuning, and road blocks that were encountered during this project..

The project repository can be found here.
The Kaggle Dataset can be found here.

2 RESULTS

The approach I took for this problem is to use a convolution neural network to learn good features for the images and then classified the images based on the learned features using a fully connected neural network.
I used transfer learning for my convolutional layers. The architecture I used for the convolution neural network was the VGG16 architecture. I only kept the convolutional layers and disposed of the dense layers that came along with the VGG16 architecture; this allowed me to adapt the pre-learned model to work on my data set because the images of my data set were of size 64 x 64. I froze the convolutional layers so that its weights were no longer trainable, then I added 2 final layers for the neural network, then were:

- Flatten: to flatten the output of the convolutional neural networks and input it into the dense layer.
- Dense: this dense layer is the output layer and contained only 10 units.

The activation function for the Dense layer was a softmax activation function. I used an epoch of 6, batch size of 16, a learning rate of 0.001, the Adam optimizer, the sparse categorical cross entropy loss, and the accuracy metric.
I was able to achieve an accuracy of 87% for the test set. Please reference the figures in the next page for the accuracy and loss graph, and the confusion matrix.

Author's address: Thai La, vtl932@usask.ca, University of Saskatchewan, 105 Administration Pl, Saskatoon, Saskatchewan, Canada, S7N 5A2.

XXXX-XXXX/2023/12-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

3 HYPERPARAMETER STUDY

I performed a hyperparameter study on the trained model. A total of 5 sweeps were performed, each sweep with different hyperparameters.

- Sweep 1: Epochs: 10, Learning Rate: 0.001, Batch Size: 16
- Sweep 2: Epochs: 10, Learning Rate: 0.001, Batch Size: 32
- Sweep 3: Epochs: 8, Learning Rate: 0.0001, Batch Size: 64
- Sweep 4: Epochs: 5, Learning Rate: 0.00001, Batch Size: 128
- Sweep 5: Epochs: 15, Learning Rate: 0.001, Batch Size: 128

From figure 3a and figure 3b, you can see that sweep 1 consistently performs the best, thus I used the hyperparameters of sweep 1 for the model. However, instead of 10 epochs, I had only trained the model for 6 epochs, since any further training will result in overfitting.

4 ROAD-BLOCKS

All the road blocks I ran into were implementation-based as it was my first time using tensorflow.
One roadblock was when I tuned the hyperparameters of my model, such as epochs, batch size, etc., it did not correctly fit the model again since the weights were persistent. To resolve this, I re-compiled the model every time I changed the hyperparameters.
Another road block I ran into was how to convert the numpy-representation of the image into tensor-representation of the image so that I can make use of hardware acceleration provided by Tensorflow. To resolve this, I created a utility function that converted a numpy array into a tensor.

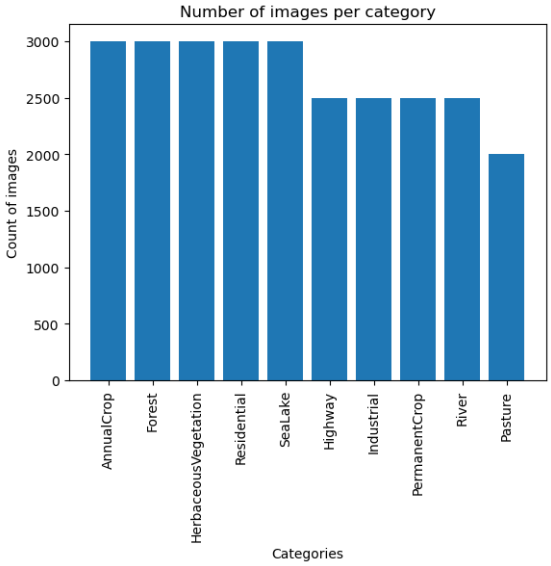

Figure 1. Final Dataset Figure

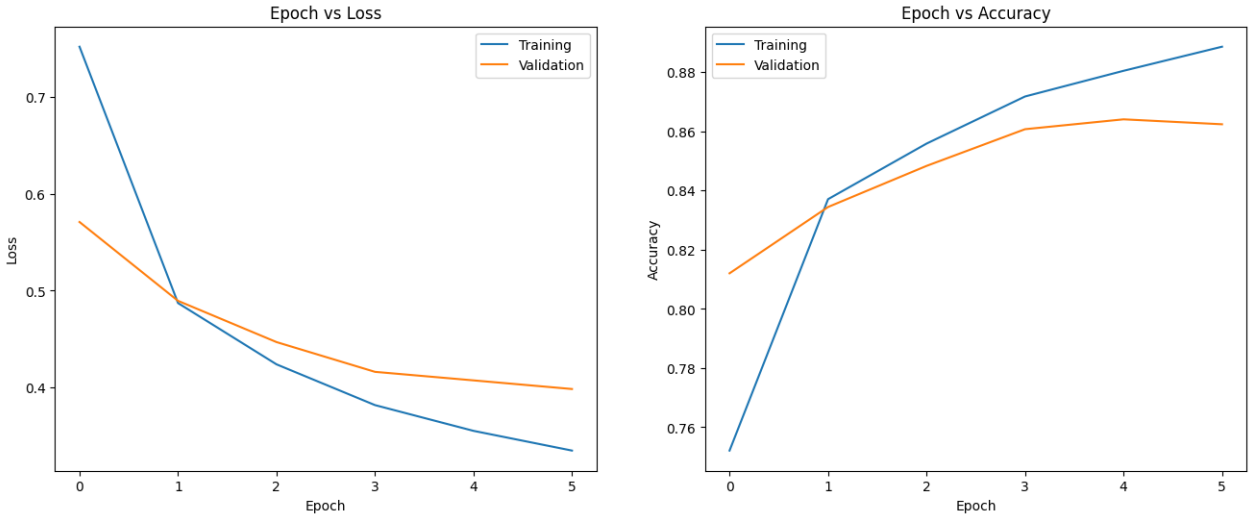

Figure 2a. Final Results Figure (a) - Training and Validation Loss and Accuracy per epoch

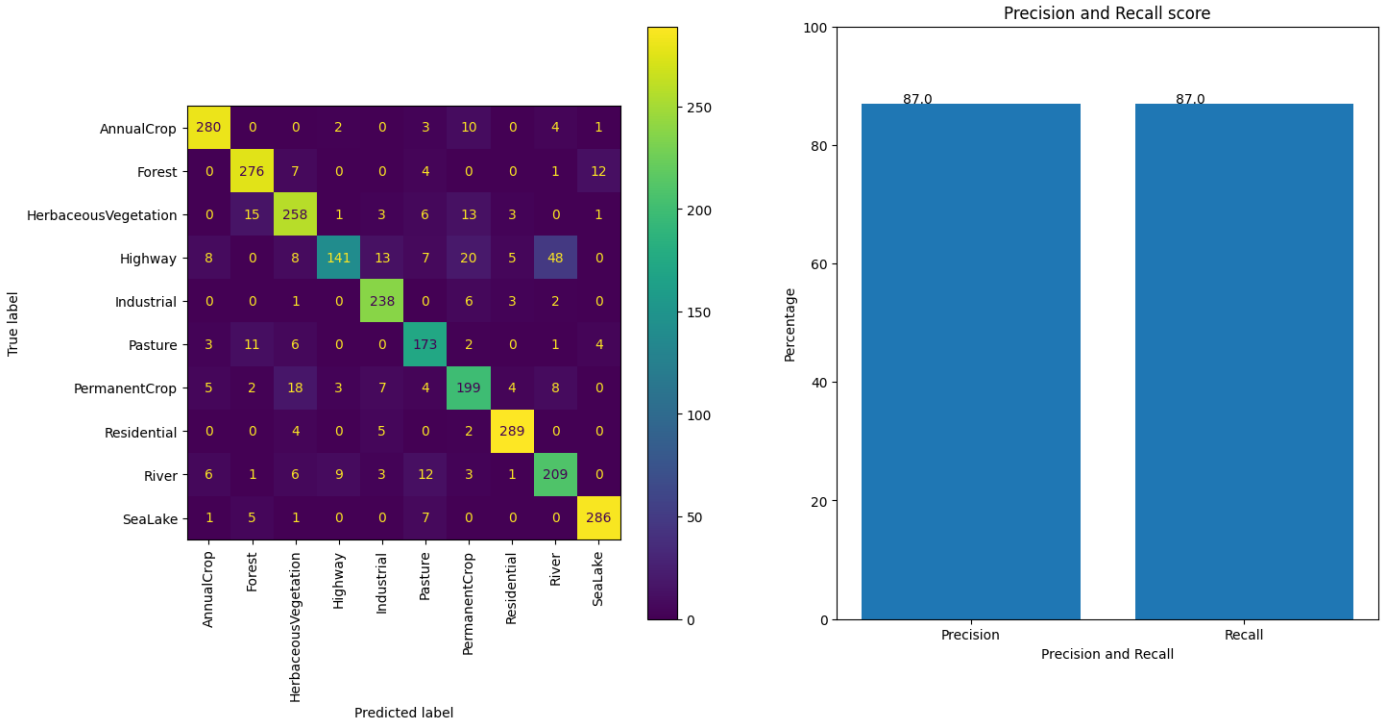

Figure 2b. Final Results Figure (b) - Confusion matrix, precision and recall of prediction on the test set

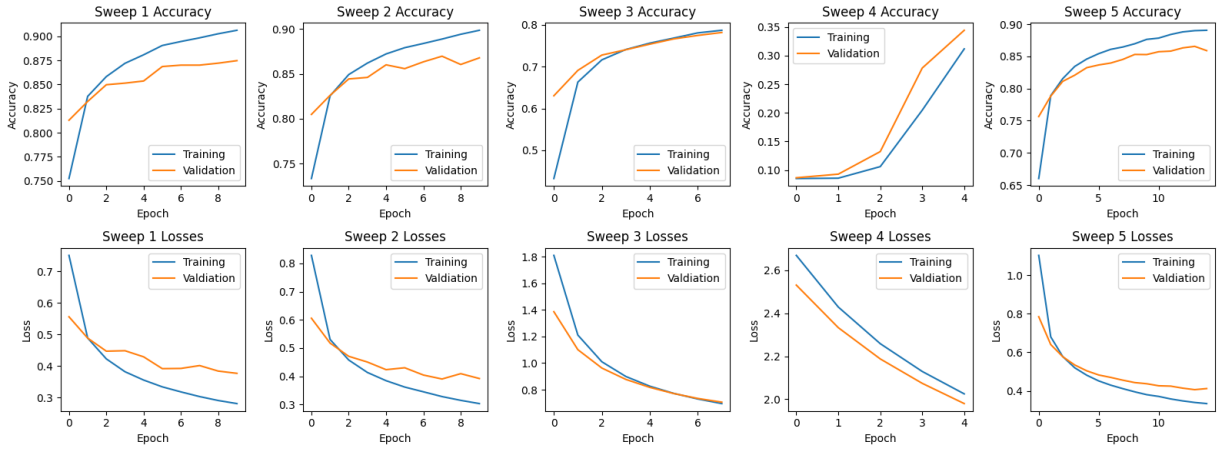

Figure 3a. Additional Results Figure (a) - Accuracy and Loss graphs per sweep

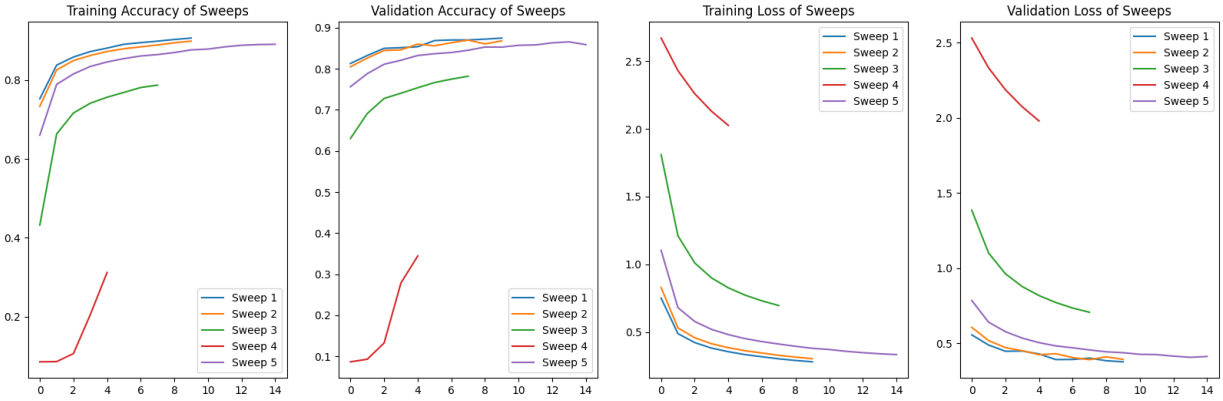

Figure 3b. Additional Results Figure (b) - Accuracy and Loss graph, aggregated sweeps