IST-718

Image Classification Project

Brenon Tate, Benjamin Dieck

Submitted: Mar XX, 2024

Table of Contents

[1 Scope 5](#_Toc160698888)

[2 Data cleansing/Preprocessing 6](#_Toc160698889)

[3 Data/Model Analysis 7](#_Toc160698890)

[4 Conclusion 8](#_Toc160698891)

List of Tables

[Table 1 Model Comparison 11](#_Toc160431891)

List of Figures

[Figure 1 Sentinel-2 Satellite 5](#_Toc160699764)

[Figure 2 Land Category Distribution 7](#_Toc160699765)

# Scope

Remote sensing plays a crucial role in many industries, to include national defense, natural disaster response and crop management. The ability to accurately classify sensor data enables informed decision-making, resource allocation, and strategic planning, ultimately contributing to enhanced efficiency, safety, and sustainability across diverse sectors.

Our aim was to train several models using the Landsat image set and evaluate their performance to identify the most accurate one. We sought to strike a balance between training speed and computational resource usage. The Landsat image set consists of 27,000 labeled images from the Sentinel-2 satellite, spanning ten different categories to include forests, annual crops, pastures, and several others.

The images were collected from the Sentinel-2 satellite and are available to the public via the [earth explorer government webpage](https://earthexplorer.usgs.gov/). The Sentinel-2 was launched on June 23rd, 2015, and travels in a sun-synchronous orbit with a repeat cycle of ten days, and period of about 100 minutes (about 1 and a half hours). The satellite carries a Multispectral Sensor (MSS) that captures imagery in multiple bands including visible, near-infrared (NIR) and short-wave infrared (SWIR).

Given their widespread adoption in image classification tasks, Convolutional Neural Networks (CNNs) emerged as our initial preference. Additionally, we sought to assess the performance of a pre-trained InceptionV3 model to gauge the compatibility of its preset weights with our dataset. To ensure comprehensive comparison, we opted to explore less conventional algorithms during model training as well.

A satellite in space above earth

Description automatically generated

Figure 1 Sentinel-2 Satellite

# Data cleansing/Preprocessing

The Landsat image set was labeled and required very little initial preprocessing. The top-level folder contained sub folders for each of the ten classes and included .csv files with image labels already separated into training, validation, and test sets. These .csv files allow the sub-folders to be easily separated using the file paths along with the ImageDataGenerator function.

Image preprocessing steps such as flattening or resizing was model specific and will be included in section 3 during the model setup and definition.

# Strong Models

After splitting the data using the ImageDataGenerator function, a bar chart was created to display the distribution of the images. As shown in Figure 2, five of the classes contain 3000 images each, four of them contain 2500 images, and one class contains 2000.

A graph of different colored rectangular shapes

Description automatically generated

Figure 2 Land Category Distribution

# Straw Man Models

Anyone that spent time working with their father at a young age will probably have been drilled into them the idea that there is a proper tool for every job. And so likewise within data science there are dozens of different ways to model a problem, but not every model is appropriate for every job. Convoluted Neural Networks is an algorithm that was specifically designed to assist machines in viewing the world in a way that human beings do (Saha). It combines the speed freak capabilities of machine learning, and neural networks and applies it to a particularly complex array of computational tasks including image, video, and signal recognition. Other engines and models have begun to build up around similar architectures including the visual image transformer which utilizes methodologies akin to the generalized pretrained transformers of the GPT models.

CNN implementations have become exponentially better and faster since 2004 replacing many other algorithms. One of the points of our study became to quantify why that was. SVM modeling and other tasks are computationally cheap, and perform quickly at many tasks with SVM being one of the original models utilized for image tasks of facial recognition. So how it is that they have become so antiquated? Part of this lies in their strengths.

Looking at algorithms previously utilized for different classification tasks they excel with data structured as binary classified information, and with non-sparse datasets. In image classification you cannot help but work with sparse information, however with some additional pre-processing we can give the models a fighting chance to punch up into this more difficult classification task.

Utilizing the MinMaxScalar() function of the sklearn.preprocessing library we can transform our array data into values that stay within the 0 to 1 range. This doesn’t make it a perfect binary classifier, but it should move it into the wheelhouse of the SVM algorithm, and more akin to something that could be processed by KNN and perhaps Random Forest. It is worth noting that this pre-processing step gave a 3 to 11% increase in model accuracy, providing the greatest improvement on the Random Forest model.

Both the K Nearest Neighbors Model, and the Random Forest Model did exactly what they are billed to do. They took a massive amount of data, generated their classification rules for the set, and applied the model in a relatively quick manner with the Random Forest performing fastest, and returning a model within about 2 minutes. K nearest neighbors still performed pretty quickly, relatively speaking, returning a model and a prediction in just under 8 minutes. However both of these models struggled to provide any kind of reliable predictive power.

At 49.3% Accuracy, the random forest model had extreme difficulties in determining images that were classified as AnnuaCrop types. It likewise failed to identify Permenant crops, at all, and trended to classify many images as residential. It also wasn’t able to identify what a highway was mistaking them for AnnualCrop or Residential images.

A diagram of a confused matrix

Description automatically generated

Figure 3 Random Forest C-Matrix

The K Nearest Neighbor Model uses Euclidean distance between plotted points to establish similarity between different groups of data. That proximity has a tendency to have overlap between different groups if the classifiers get too complicated, and that is exactly what happened in this case. At a 446 second runtime, and with an abysmal 35.7% accuracy KNN visibly cannot identify images. The one exception to this was actually somewhat surprising as it was able to pick out Forests with some degree of accuracy, even if it had a tendency to mistake them for SeaLakes. But it cannot be reliably used to classify any image it seems. It had strong accuracy in predicting SeaLakes, but that is likely because it called almost everything in the dataset a SeaLake.

A diagram of a confused matrix

Description automatically generated

Figure 4 KNN C-Matrix

We utilized the Support Vector Classification function of the sklearn library to develop our SVM model. Within this we selected the Radial Basis Function (rbf) kernal rather than the linear or polynomial kernal as we are dealing with non-linear data across many variables. Additionally the rbf kernal has a tendency to run closer to a Gaussian distribution as we might see in nature. It really is only able to deal with values between 0 and 1 which is why it was so important to do the additional preprocessing step with the MinMaxScalar function prior to running the model even if it added another 5 minutes to our computing time. Put together this was a marked improvement in our accuracy, as it delivered 66.5% accuracy in predicting labels across the dataset. This isn’t surprising as this is has been used as an image classifier in the past, with facial recognition (Phillips).

What jumps out though is how computationally expensive and intensive utilizing the SVM algorithm is when applied to this task. It took this algorithm over 16 hours to run the initial model, let alone the prediction which took an additional 17.8 hours. The points brought up by this paper are performed through significant trial and error, as the first models made for the SVM were done utilizing no processing and using a linear kernel. So days of processing went into not only generating the model, but attempting to optimize it.

One additional measure was attempted during this study and that was applying hypertuning to the parameters within the rbf kernel. This involved creating a list of 3 values for two different parameters of the rbf kernel, Gamma and C. Coding was generated to conduct a grid search between combinations of the two parameters to find the best one. The experiment was terminated after running for 5 days.

A graph showing different types of plants

Description automatically generated

Figure 5 SVM C-Matrix

From the confusion matrix we can see that the model performed far better than the other two more traditional classification models attempted. While the 66.5% accuracy might not be enough to make a money decision behind, it is certainly better than random guessing across 10 categories. The SVM model was particularly good at predicting Forest, Industrial, Pasture, Residential, and SeaLake images. It trended towards overclassifying images as PermanentCrop and HerbaceousVegetation.

Table 1 Model Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Elapsed training time (s) | Prediction Time (s) | Accuracy (%) |
| Keras Sequential CNN |  |  |  |
| InceptionV3 |  |  |  |
| SVC w/ rbf | 56280 | 64200 | 66.5% |
| KNN | 446 | - | 35.7% |
| Random Forest | 72 | - | 49.3% |

# Conclusion

XX

Table 1 Model Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Elapsed training time (s) | Prediction Time (s) | Accuracy (%) |
| Keras Sequential CNN |  |  |  |
| InceptionV3 |  |  |  |
| SVC w/ rbf | 56280 | 64200 | 66.5% |
| KNN | 446 | - | 35.7% |
| Random Forest | 72 | - | 49.3% |

# References

Phillips, P. J. (1998). Support Vector Machines Applied to Face Recognition. Retrieved from https://proceedings.neurips.cc/paper\_files/paper/1998/file/a2cc63e065705fe938a4dda49092966f-Paper.pdf

Saha, Sumit, **A Comprehensive Guide to Convolutional Neural Networks – the ELI5 Way**, Towards Data Science, 15 December, 2018, <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>, accessed 9 March, 2024.