Project Recap

Estimated reading time: 5 minutes

This AI Capstone Project is an exciting hands-on journey into deep learning for image classification, specifically applied to satellite imagery. Imagine you're a data scientist for a fertilizer company, wanting to expand into new territory. The project enables you to build AI models that automatically classify images into agricultural and non-agricultural land, saving time and effort compared to manual classification and helping the company build its strategy efficiently.

In this course, the labs were structured across four modules, starting with simple data handling and moving to sophisticated neural networks. You worked with a dataset of satellite images stored in folders: one for "class_0_non_agri" (non-agricultural land) and another for "class_1_agri" (agricultural land). The goal was binary classification, as in, deciding if an image showed agricultural or non-agricultural land. You used Jupyter notebooks, which made it interactive, allowing you to run code, visualize images, train, and evaluate models.

Deep learning is like teaching a computer to recognize patterns through layers of math-inspired "neurons." You looked at the comparisons between frameworks like Keras (user-friendly, high-level) and PyTorch (more flexible, low-level). By the end, you integrated models and evaluated them, gaining skills applicable to real-world problems like climate monitoring or urban planning.

Let's take a recap of what you did across all modules.

Module 1: Data preparation and exploration

The foundation of any AI project is data, and Module 1 focused on loading and exploring satellite images efficiently. Think of data as the fuel for your AI engine; if it's messy, your model won't perform well. The labs emphasized memory efficiency, using lazy loading of images, one by one, instead of all at once, to avoid crashing your computer. You explored bulk loading trade-offs, where you read all images into a list of arrays (using NumPy) for faster access but higher memory use.

By the end of this module, you prepared balanced datasets (equal agri and non-agri samples) and learned preprocessing basics, like resizing images to a standard size (for example, 64x64 pixels) and applying augmentation techniques to increase the model training dataset.

Module 2: Building and comparing basic classifiers

Next, you created classifiers using Convolutional Neural Networks (CNNs), the workhorses of image AI.

First, you built a simple CNN in Keras (built on TensorFlow). Keras is beginner-friendly and has high-level commands. You defined layers: convolutional layers extract features such as edges or colors, pooling layers reduce size, and dense layers make decisions.

You compiled it with an optimizer (such as Adam) and loss function (binary cross-entropy), then trained on your image paths, splitting data into train/test sets (for example, 80/20). Evaluation uses metrics such as accuracy, which measures how often the model correctly classifies agri versus non-agri.

Next, you used PyTorch, which offers more control. You defined a class inheriting from nn.Module, with a forward method for data flow. It's like building with Lego bricks:

```
import torch.nn as nn
class SimpleCNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 32, 3)
        self.pool = nn.MaxPool2d(2,2)
        self.fc = nn.Linear(..., 1)  # Adjust based on shape
    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = x.flatten()
        return torch.sigmoid(self.fc(x))
```

Training involved data loaders (using torch.utils.data.DataLoader) for batching images, and a loop with optimizer steps. PyTorch was more "hands-on", allowing custom tweaks.

The comparative analysis lab had you compare both models on the same data. Keras might train faster, but PyTorch excels in debugging. You evaluated with confusion matrices (showing true positives and false negatives) and metrics such as precision/recall. Results showed >95% accuracy, depending on data quality. Key lesson: No single framework is best; choose based on expertise level and needs.

Module 3: Advanced models with Vision Transformers (ViTs)

Then, you were introduced to Vision Transformers (ViTs), a cutting-edge alternative to CNNs inspired by language models like GPT.

ViTs treat images as sequences of patches (such as chopping a photo into tiles), then use self-attention to focus on important parts. For beginners, attention is like the model "paying attention" to green patches for agri land. In Keras, you used pre-built layers or libraries such as tensorflow_addons and created:

```
from tensorflow.keras import layers
layers.LayerNormalization
layers.MultiHeadAttention
```

You designed and trained the CNN-ViT hybrid models for the satellite dataset, using transfer learning from a pre-trained CNN model to save time and computational resources. Training involved handling sequences, which were memory-intensive, so batch sizes were kept small. The CNN-ViT hybrid combines the strengths of CNNs and ViTs. The CNNs excel at extracting the local features, and ViTs look at the global correlations efficiently. Combining these two characteristics in a single CNN-ViT hybrid model provided you with the best of both worlds and created a model that could provide very accurate predictions.

ViTs often perform better on complex patterns (for example, distinguishing subtle vegetation) but need more data and computing. Accuracy might hit 90%+, highlighting why ViTs power modern AI such as DALL-E.

Module 4: Integration and final evaluation

The capstone wraps up by evaluating CNN-ViTs hybrid models. Evaluation includes cross-validation and real-world testing, and it discusses overfitting (the model memorizes training data) and solutions such as dropout.

Project conclusion and key learnings

This project transformed you from a beginner loading images to an expert builder of AI classifiers. You classified land types with >95% accuracy, learned Keras for simplicity and PyTorch for flexibility, and explored ViTs for advanced tasks. Applications extend from territory expansion in a fertilizer company to disaster response or farming tech. Challenges such as data imbalance teach real AI skills. Overall, it's a rewarding capstone, proving deep learning's power in solving real-world industry problems. If you're inspired, try extending it with your own datasets!