Deep Learning for Land Use Classification

Brian Chan {bchan17}, Advised by Patrick Cho

Abstract
Recent advancements in satellite technology continue to increase the affordability, accessibility, and accuracy of aerial satellite imaging. Insights from these images, particularly over time, is valuable for efforts ranging from urban planning and economic analysis [3] to military efforts, in both urban and rural as well as developed and developing regions. We utilize CNN architecture to develop an algorithm that, given a satellite image of a geographic region, classifies it based on the type of land use. Classes include scenes from urban, rural, and natural areas, predicting them to be one of 45 classes.

Dataset
- Image samples, clockwise from upper left: airport, sea ice, basketball court, and chaparral classes.

Dataset Features
- NWPU-RESISC45, a 2017 dataset presented by Gong Cheng, Junwei Han, and Xiaoliang Liu. It contains 30,500 256x256 RGB images split into 45 classes of 700 images each [2].
- More diverse in number of classes and samples than other classic land use datasets such as UC Merced Land Use [1], which has only 100 images for 12 classes and Sai-Whu Land Use datasets, which is largely urban China.
- Contains both urban and rural imaging, small and large features like the "church" class as well as the "harbor" class.
- Classes include meadow, island, freeway, cloud, etc.

Basic CNN
- Multi-layer CNN with ReLU activations and Maxpools.
- Built and trained from scratch.
- Tested with different number of layers, filter sizes, and FC nodes.
- Final layer is softmax for 45 classes.
- Utilizes ReLU activations.
- Top 3 accuracies.

Models
- CaffeNet Model
  - Derived from CaffeNet architecture, features 5 convolutional layers.
  - CaffeNet, along with GoogLeNet and AlexNet (from which CaffeNet is derived), are already popular algorithms for use in image classification.
  - Hyperparameter testing includes editing number of layers, nodes, and filter sizes.
  - Final layer is a softmax layer for 45 classes.

- Model Equations
  - Categorical Cross-entropy: per sample, loss function used to penalize less confident predictions when a sample belongs to a class.
    \[ -\sum_{c=1}^{M} y_{oc} \log(p_{oc}) \]
    \[ M = \text{total number of classes}, \quad y_{oc} = 1 \text{ if this observation } o \text{'s true class is } c, \quad p_{oc} = \text{predicted probability of the observation } o \text{ belonging to } c \]
  - RMS-new: updates weight parameters as a function of the gradient to spread up gradient descent
    \[ \nabla W_{t+1} = \nabla W_t - \gamma \nabla W_t \]
    \[ \nabla W_t = \nabla W_t / \sqrt{\frac{1}{t+\epsilon} + \lambda} \]
    \[ B_t = \text{exponential decay hyperparameter}, \quad \gamma = \text{exponentially weighted average of gradient at timescale } t \]

Data Processing and Hyperparameter Tuning
- Image Preprocessing
  - Image Resizing: All images are resized to 64x64.
  - RGB channels maintained.
- Data Split: Images are split, per class, into 80:10:10 Training:Validation:Testing. Results in 560:70:70 images per class.
- Images are also shuffled before distribution in case of underlying patterns in data collection.
- Keras generator allows for random horizontal flipping of images.

CNN
- Number of Layers: basic tuning of the number of layers in my CNN model.
- Learning rate: I experimented between .001 and .002 for the learning rate.

CaffeNet Model
- Filter Sizes and Strides: slight altering of the filter sizes and strides for early layers in the CaffeNet model.
- FC Nodes: tuning between 250, 1024, and 2048 nodes in the final FC layers of the network.
- FC Layers: also experimented with having 1 and 2 FC layers before softmax.

Discussion
- Overall, both algorithms were not very robust, and they could likely be improved upon by using transfer learning, which will be my next step before the final project report.
- Algorithms with better image segmentation abilities such as ResNet might achieve better results.
- Notably, while using the basic layered CNN, one class, snowbank, had 0 recall.

Future Work
- Final project report: work with pretrained models and transfer learning.
- Investigate other algorithms like GoogLeNet/ResNet.
- Collect/contribute additional classes to the dataset.

Results
- Model | Layers | FC Nodes | LR | Train Acc | Val Acc | Test F1
- Basic1 | 3 | 512 | .001 | 4802 | 4904 | 5393
- Basic2 | 4 | 512 | .001 | 4994 | 5489 | 5342
- Basic3 | 4 | 1024 | .002 | 5320 | 5345 | 5348
- Caffe1 | Many | 256 | .001 | 5222 | 4940 | 4948
- Caffe2 | Many | 1024 | .001 | 6248 | 5708 | 5846
- Caffe3 | Many | 2048 | .001 | 6196 | 5471 | 5628

Table 1: Parameters of CNNs

Figure 1. Validation set accuracy over epochs during training. Some models were trained up to 25 epochs.

References