Land-Use Image Classification

Mir Abir Hossain
Student Id: 012222006

A Project

For

Data Mining

CSE6011

The Department

of

Computer Science and Engineering

United International University

Dhaka, Bangladesh

August 2022

Table of Contents

Introduction

Image classification refers to a process in computer vision that can classify an image according to its visual content. For example, an image classification algorithm may be designed to tell if an image contains a human figure or not. While detecting an object is trivial for humans, robust image classification is still a challenge in computer vision applications. Image Classification is a vital task in various fields such as remote sensing, biometry, biomedical images, and robot navigation.

Image Classification includes following steps:
a. **Image Acquisition**: acquire the images for image processing.
b. **Image Pre-Processing:** In preprocessing image transformation, noise removal, at morphemically correction techniques are used.
c. **Feature Extraction**: Extracting the important characteristics of the image.
d. **Classification**: The images are classified based on the extracted characteristics into predefined categories by using suitable methods that evaluate the image pattern with images which inside the database.

In this project, we are using UC Merced Land Use Dataset[1]. This is a 21 class land use image dataset meant for research purposes.

There are 100 images for each of the following classes:

- agricultural
- airplane
- baseballdiamond
- beach
- buildings
- chaparral
- denseresidential
- forest
- freeway
- golfcourse
- harbor
- intersection
- mediumresidential
- mobilehomepark
- overpass
- parkinglot
- river
- runway
- sparseresidential
- storagetanks
- tenniscourt

Each image measures 256x256 pixels.

The images were manually extracted from large images from the USGS National Map Urban Area Imagery collection for various urban areas around the country. The pixel resolution of this public domain imagery is 1 foot. The dataset can be downloaded from here. [http://weegee.vision.ucmerced.edu/datasets/landuse.html]

Background

Image classification, which can be defined as the task of categorizing images into one of several predefined classes, is a fundamental problem in computer vision. It forms the basis for other computer vision tasks such as localization, detection, and segmentation (Karpathy, 2016). Although the task can be considered second nature for humans, it is much more challenging for an automated system. Some of the complications encountered include viewpoint-dependent object variability and the high in-class variability of having many object types (Ciresan, Meier, Masci, Gambardella, & Schmidhuber, 2011). Traditionally, a dual-stage approach was used to solve the classification problem. Handcrafted features were first extracted from images using feature descriptors, and these served as input to a trainable classifier. The major hindrance of this approach was that the accuracy of the classification task was profoundly dependent on the design of the feature extraction stage, and this usually proved to be a formidable task (LeCun, Bottou, Bengio, & Haffner, 1998). In recent years, deep learning models that exploit multiple layers of nonlinear information processing, for feature extraction and transformation as well as for pattern analysis and classification, have been shown to overcome these challenges. Among them, CNNs (LeCun, Boser, Denker, Henderson, Hubbard, & Jackel, 1989a, 1989b) have become the leading architecture for most image recognition, classification, and detection tasks (LeCun, Bengio, & Hinton, 2015). Despite some early successes (LeCun et al., 1989a, 1989b; LeCun et al. 1998; Simard, Steinkraus, & Platt 2003), deep CNNs (DCNNs) were brought into the limelight as a result of the deep learning renaissance (Hinton, Osindero, & Teh, 2006; Hinton & Salakhutdinov, 2006; Bengio, Lamblin, Popovici, & Larochelle, 2006), which was fueled by GPUs, larger data sets, and better algorithms (Krizhevsky, Sutskever, & Hinton, 2012; Deng & Yu, 2014; Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014). Several advances such as the first GPU implementation (Chellapilla, Puri, & Simard, 2006) and the first application of maximum pooling (max pooling) for DCNNs (Ranzato, Huang, Boureau, & LeCun, 2007) have all contributed to their recent popularity. The most significant advance, which has captured intense interest in DCNNs, especially for image classification tasks, was achieved in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 (Russakovsky et al., 2015), when the winning entry, by Krizhevsky et al. (2012), used a DCNN to classify approximately 1.2 million images into 1000 classes, with record-breaking results. Since then, DCNNs have dominated subsequent versions of the ILSVRC and, more specifically, its image classification component (Simonyan & Zisserman, 2014; Zeiler & Fergus, 2014; Szegedy, Liu, et al., 2015). In addition, selected representative examples of other improvement attempts related to the following different aspects of DCNNs—(1) network architecture (Lin, Chen, & Yan, 2013; Zeiler & Fergus, 2013; Gong, Wang, Guo, & Lazebnik, 2014; Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015); (2) nonlinear activation functions (He, Zhang, Ren, & Sun, 2015a; Xu, Wang, Chen, & Li, 2015); (3) supervision components (Tang, 2013; Zhao & Griffin, 2016); (4) regularization mechanisms (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012; Zeiler & Fergus, 2013); and (5) optimization techniques (Glorot & Bengio, 2010; Krizhevsky et al., 2012)—have also been implemented in recent years. Moreover, some of their open challenges, like their variance to geometric distortions (Gong, Wang, et al., 2014), the fact that their models are often large and slow to compute (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014), and the intriguing discovery of adversarial examples (Szegedy et al., 2014), have led to even more research focusing on image classification with DCNNs. Previously, several generic deep learning reviews (Bengio, 2009; Schmidhuber, 2015; Deng, 2014; LeCun et al., 2015), reviews that deal with deep 2354 W. Rawat and Z. Wang learning for visual understanding (Guo et al., 2016), reviews covering recent advances in CNNs (Gu et al., 2015), and a taxonomy of DCNNs for computer vision tasks (Srinivas et al., 2016) have been published. [2]

Objective

The task is to correctly predict the class of a given image.

Experimental Data

Our dataset consists of 2100 images of 21 class. Each class consists of 100 images. We divided our dataset into three parts. Training set, validation set and test set. Training set comprises 81% of the data, validation set comprises 9% of the data and test set comprises 10% of the data. We divided the dataset with *stratified=dataset.targets* condition on. Which means this proportion of datasets hold true for all the classes. That means, among 100 images of *beach* class, 81 of them made it to training dataset, 9 of them in validation dataset and 10 of them in testing dataset. And this is true for all other classes. In total, as we have 2100 images, training set holds 2100*81% = 1701 images, validation set holds 2100*9%=189 images and testing set holds 2100*10%=210 images.

Methodology

We have used two distinctive methods to solve the problem.

First, we try to solve the problem with traditional method where we first extract hand crafted features from the image and then apply machine learning model. We have used SVM classifier in our case, as suggested by different literatures mentioned in the background section.

Second method is the deep learning method. We have directly fed our images into deep convolutional neural network with the label and the network learns the parameters to correctly classify the images. We will discuss both the methodology in next sections.

Traditional Machine Learning Approach

Feature Extraction

RGB2GRAY

At first we converted our RGB images into gray scale image. We used the formula Y = 0.2125R + 0.7154G + 0.0721B which has been implemented in the skimage library. That converts our 256*256*3 image into 256*256*1 size.

HOG descriptor

Then we used histogram of oriented gradients, which is known as HOG descriptor. The idea behind hog is that an object's shape within an image can be inferred by edges, and a way to identify edges is by looking at the direction of intensity gradients (i.e. changes in luminescence). An image is divided in a grid fashion into cells, and for the pixels within each cell, a histogram of gradient directions is compiled. To improve invariance to highlights and shadows in an image, cells are block normalized, meaning an intensity value is calculated for a larger region of an image called a block and used to contrast normalize all cell-level histograms within each block. The HOG feature vector for the image is the concatenation of these cell-level histograms.

PCA

After applying HOG descriptor we flatten our image. At this point we have more than 0.2 million feature per image. (212484 to be exact). We can not run machine learning models with so many parameters. It will be computationally expensive and also the algorithms may not be well adapted to these many features.
That is why we use principle component analysis. It is a dimensionality reduction technique. PCA is a way of linearly transforming the data such that most of the information in the data is contained within a smaller number of features called components.
In our case we have applied PCA using sklearn library and we have found out 450 components which explains 90% of the variance in dataset.

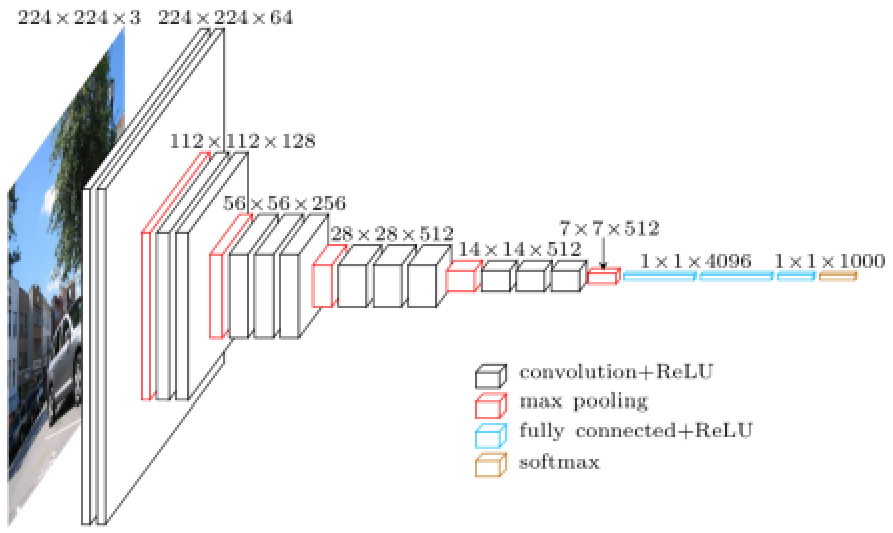
Training with SVM

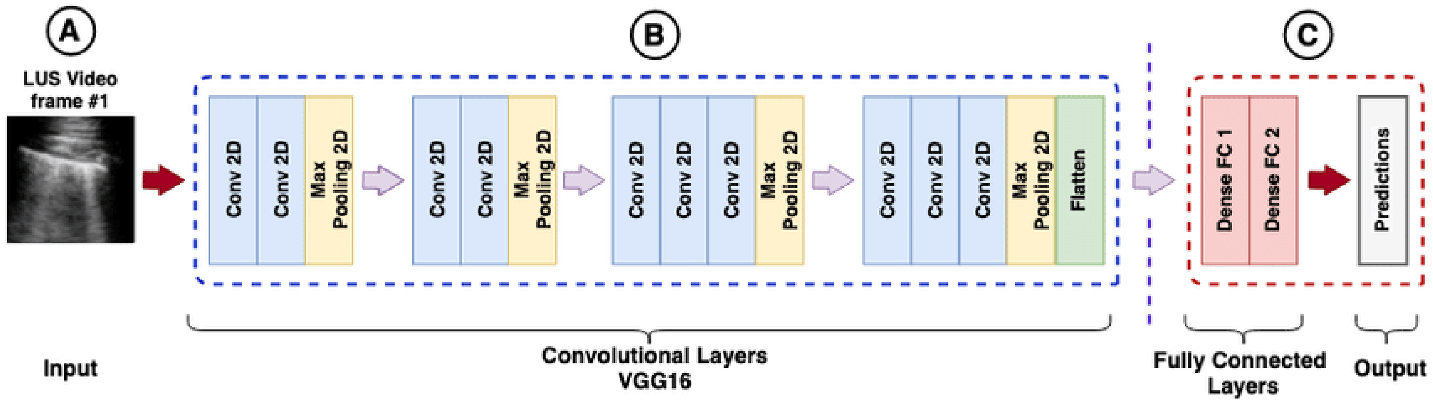
We have used a support vector machine (SVM)[3], a type of supervised machine learning model used for regression, classification, and outlier detection. "An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall."

In our case, we have used our training data to train the svm model and validation data to validate the model. Although we did not get a good accuracy which we have talked in the result section.

Deep Convolutional Neural Network

As we have mentioned in the background section, DCNN have been used with quite a success in recent years. We have tried to implement one DCNN model in this project. We have used VGG16 network [4]
The architecture diagram goes as follow:





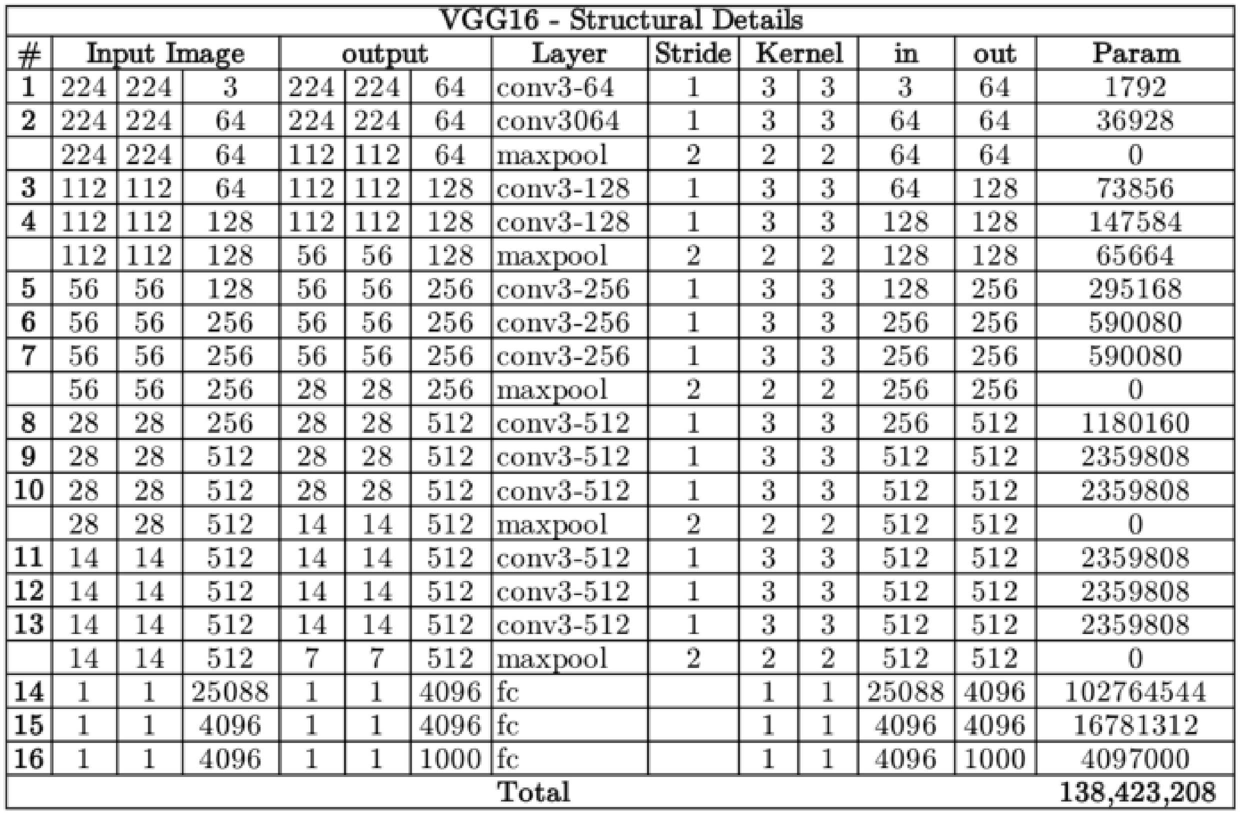
The VGG16 network was published in 2014 and it is a remarkable architecture for image data. It is called vgg16 as we can we there are total 16 convolutional layers. There are other variants of this architecture such vgg19 etc. In this project we have used the architecture from the original paper, vgg16.
VGG16 architecture uses only 3*3 filter throughout the network with same padding and maxpool layer of 2*2 filter of stride 2.

Instead of training a vgg16 network from scratch, we have used transfer learning method. Because training a vgg16 from scratch will be highly time consuming and computation consuming, overall inefficient. Instead, we have used the pretrained model that was already trained on ImageNet dataset [5] and implemented in Pytorch framework.

The original VGG16 network takes input image of size 224*224. But in the Pytorch implementation they have added an average pooling layer, which converts any input size into 224*224.

We have used pretrained model. As the model was pretrained on ImageNet dataset and it has 1000 classes, the last fully connected layer converts 4096 inputs to 1000 output. But in our case, we had to change it to 21 classes. Before putting the training data into the model, we froze all the parameters of all the layers except for the last layer.

Our VGG16 has in total 134 million parameter which is close to the original model's parameter count (138 million). Parameter count goes as follows:



VGG16 - Structural Details

#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
							Total						**138,423,208**

Experimental result

We have run both the SVM and VGG16 model in google colab [6].

For measuring the strength of the model, we have used accuracy score.
Accuracy score - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.
For a multiclass classification problem with more than two class, the accuracy score is the following

$$accuracy = \frac{no.of\ observations\ with\ correctly\ predicted\ class}{total\ observations}$$

Accuracy for SVM model: We have got training accuracy of only 36.75%. The accuracy was pretty low to move along with more exploration.

Accuracy for VGG16 pretrained model: We got a good accuracy from the VGG16 pretrained model. Accuracy for training data is 96.12%, accuracy for validation data is 96.30%, accuracy for test data is 96.19%.
As we can see, the pretrained VGG16 model has given us a very good accuracy and the accuracy value is consistent throughout training, validation and testing set.

Conclusion

Image classification task is a well explored areas compared to others. In early days, people need to use handcrafted features and then apply machine learning models. Getting better results with this approach was often difficult. Finding out the right set of feature extraction technique, combined with the right choice of machine learning model was a daunting task. But the rise of the deep learning techniques in past decade has brought a revolutionary change in the image classification task. Deep Convolutional Neural Networks is doing remarkably well in this field. It is no more necessary to handcraft features, cause DCNN can learn them by iterative process. Our outcome of the project also shows the same. While using hog descriptor feature extraction technique with SVM we got only 36.75% accuracy, using a pretrained VGG16 network, we got more than 96% accuracy.

Limitation

1. We have used pretrained model. As our dataset was small, it was practically impossible to gain such a high accuracy of 96% using such a small dataset with DCNN.
2. We have used well prepared dataset, if we have used more raw dataset, it would be difficult to gain good accuracy.
3. In machine learning approach we only used hog descriptor for feature extraction. There may exist other good feature extractors which could be useful with remote sensing data that we used. We could have used several extractor and combine them together for better accuracy.

Future Work

Current state of the model algorithms for image classification is already almost in an unexploitable situation. Every year in imagenet competition researchers around the globe came up with all highly sophisticated deep learning models to classify images correctly. With every passing month, new type of neural networks are being built with more parameters and with more robustness. Some of these new networks can be explored for the image classification tasks for better output.

References

[1]

Yang, Yi, and Shawn Newsam. "Bag-of-visual-words and spatial extensions for land-use classification." *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. 2010.

[2]

W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," in Neural Computation, vol. 29, no. 9, pp. 2352-2449, Sept. 2017, doi: 10.1162/neco_a_00990.

[3]

Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine learning* 20.3 (1995): 273-297.

[4]

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[5]

J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[6]

Bisong, E. (2019). Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_7