Optimizing Neural Networks for Cloud Detection

Harrison Engoren

August 8, 2017

[1]

Abstract

Neural networks are used extensively for image classification, and consist of an input layer several hidden layers, and an output layer. The network works iteratively, progressing through each hidden layer, transforming its input, and outputting the input for the next layer. When the network reaches the final output layer, it makes a predictive classification: it then compares its prediction to the expected classification and 'learns' by altering the weights of the network accordingly. We used neural networks' ability to classify images to identify cloud and no-cloud regions in a satellite image. We tested three networks – LeNET, SegNET, and AlexNET – over seven different input image sizes – 60x60, 120x120, 180x180, 210x210, 227x227, 240x240, and 300x300 pixels – to determine their average classification accuracy. While the mean for SegNET and AlexNET practically remained constant across all sizes, we found that LeNET's average classification accuracy increased nearly 20% from 60x60 to 300x300 inputs.

1 Introduction

Neural networks are extremely effective in accomplishing image classification. A neural network comprises a data layer, a variable number of hidden layers, and a classification layer. These hidden layers are usually either a convolutional layer, a fully connected layer, or a pooling layer. It is standard for an input layer to consist of a value for each pixel. Since we are classifying Red, Green, Blue (RGB) images, we will have a value for each color channel for each pixel.

A convolutional layer uses filters to convolve, or slide, through the input image and create an activation map that indicates the response of the filter at each position. Each of these filters serves to help the neural network learn a specific feature of the image. For example, a neural network attempting to classify different vehicles may learn to recognize a wheel-like structure when it sees a car or truck. Fully connected layers work in a similar manner, but convolutional layers are related to a specific region in the input image, while fully connected layers transfer information from the entire image.

Pooling layers work in tandem with convolutional layers to continue to downsize the image. A pooling layer functions as a summary of the convolutional layer: it takes the maximum response from a region of the convolutional layer (often 2x2) and creates a smaller feature map. We can think of the network as simply identifying whether or not a specific feature has been located in that region of the image without the exact position. This reduces the amount of information the network needs to store and makes it easier for the network to iterate through future hidden layers.

Finally, the output layer, also known as the classification layer, uses of a softmax function that serves to output a probability distribution. This probability distribution allows us to measure the probability that an image is a cloud. [4]

We will tile a large satellite image into coinciding squares and use our trained neural networks to classify each square with the probability it is a cloud. Using these classifications, we can generate a map of where the network believes there are clouds. In turn, this method will allow us to determine the location of clouds throughout the image. In similar applications, a tile size of 227x227 pixels

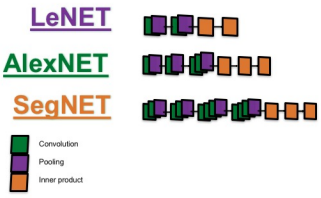

Figure 1: Network Architectures

has been used, but we will measure the classification ability of six other tile sizes: 60x60, 120x120, 180x180, 210x210, 240x240, and 300x300.

2 Experiment

We will use the Caffe framework to train and test three different networks, LeNET, AlexNET, and SegNET, for each tile size. LeNET is a relatively shallow network, with only two convolutional, two pooling, and two fully connected layers. [3] AlexNET is slightly deeper, with two extra convolutions and a pooling layer in addition to a third fully connected layer at the end. [2] SegNET is the deepest network we trained and tested, with 13 hidden layers. It begins with two convolutional layers followed by a pooling layer. This pattern is repeated before it progresses to three convolutional layers followed by a pooling layer, which also occurs twice. Finally, the SegNET architecture concludes with three fully connected layers. [1] These architectures can be visualized in Figure 1.

Each network will be trained long enough so that it can reach a maximum accuracy, but not so long that the network overfits to the training set. If a network is trained too long on the same data set, the network can learn to recognize features specific to the training data as opposed to general properties of the images. This overfitting is comparable to cramming the night before a test - rather than learning and understanding the general principles, you end up memorizing the details of the material you studied. However, if the test is dissimilar to your study materials, you will perform poorly on the test and will not be able to apply your knowledge in the future. Likewise, the training data and testing data fed to the network are very different, so if the network memorizes the training set, it will perform poorly on the test set and other input data.

From each run, we recorded the maximum test accuracy and calculated the statistical properties of the results, such as the mean and standard deviation. We can see this distribution of maximum accuracies for each network for selected tile sizes in Figure 2. Despite the mean accuracy for AlexNET and SegNET remaining virtually constant, we see that LeNET's mean accuracy increases with tile size. A full graph comparing depicting the relationship between the mean accuracy for each network and tile size can be seen in Figure 3. Figure 3 is a holistic comparison of all the networks we trained and tested.

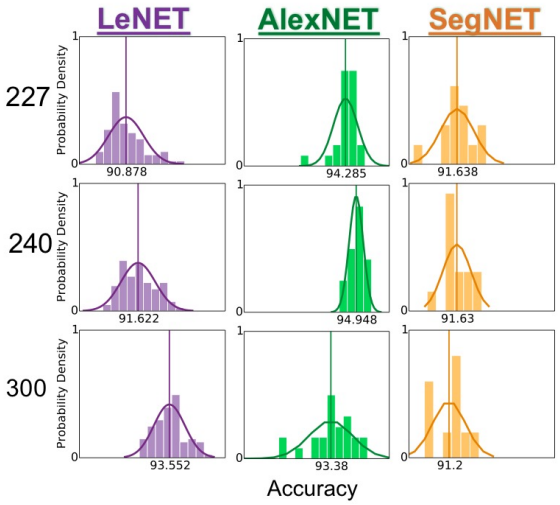

Figure 2: In the left column, we see a clear increase in mean accuracy as tile size increases. However, in the middle and right columns, we see the mean accuracy remain approximately constant as tile size increases.

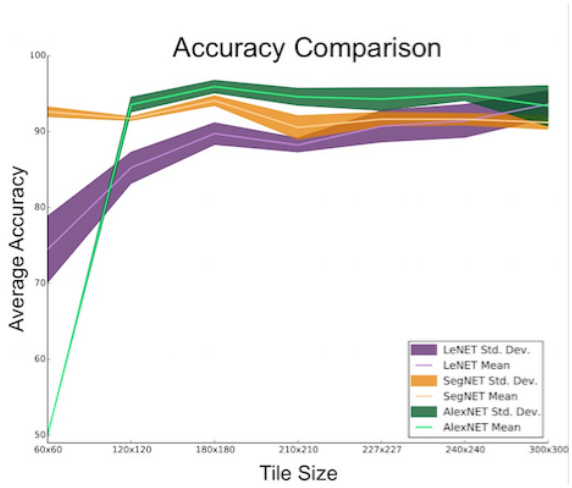

Figure 3: Graph plotting average accuracy for each network against tile size

3 Results

In order to determine the best tile size to label cloud and no-cloud regions, we have to segment the overhead image into tiles and deploy the neural network. First, we tile the image into x by x squares with a predetermined stride length less than x. For our segmentations, we used a stride length of 20. A smaller stride length should result in a more accurate prediction of each pixel's probability of being a cloud. A smaller stride length will also increase the amount of time it takes for the network to deploy, so there should be caution in setting a very small stride length. Once we have tiled the image, we use the weights determined by the Caffe model to deploy the network to classify each tile. Each pixel in a given tile will be assigned a score - the probability that the tile is a cloud. Each pixel will then accumulate a total score and we can divide the total score by the number of times the pixel has been classified to obtain the average score. The average score for each pixel will represent the probability that it is part of a cloud region. Finally, we can use the Python Image Library (PIL) to create an image from the matrix of average scores that corresponds to where the network identifies cloud regions and its confidence level. The results of this methodology can be viewed in Figure 4.

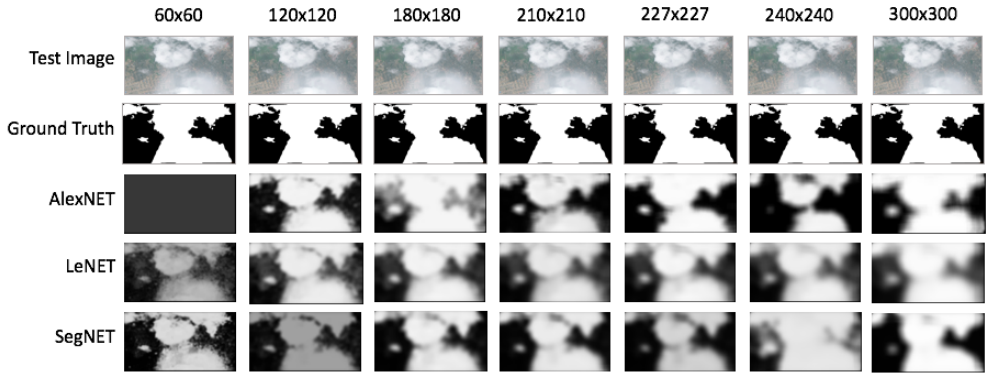

Figure 4: Results on a sample test image.

4 Discussion

Overall, this study shows that optimal tile size depends on the neural network. It was clear that LeNET's average classification accuracy increased with size. From the smallest tile size, 60x60, to the largest tile size, 300x300, there was almost a 20% increase in accuracy. Although AlexNET performed poorly and was unable to learn for 60x60 tiles, its difference in accuracy across the other tile sizes was negligible. SegNET also only had minor fluctuations in its' average accuracy; the mean accuracy was stable.

Obviously, we cannot feed the full overhead image to the neural network and locate cloud and no-cloud regions, so we would like to experiment with 500x500, 700x700, and 900x900 tiles to try to find an upper bound on classification accuracy. Caffe networks also include the option to randomly crop the input image. For example, the network can be trained and tested on a randomly selected 227x227 portion of a 300x300 tile. It would be interesting to see how the classification accuracy performs when a random crop is implemented. There are various ways to implement the random crop; the image could be cropped down to any smaller tile size. A rigorous experiment would involve exploring all possible combinations of a random crop for all the tile sizes. A random crop may improve segmentation results because the neural network could focus more on the main part of the tile instead of the outlying regions.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[4] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.