Transfer Learning for High Resolution Aerial Image Classification

Yilong Liang*, Sildomar T. Monteiro*†, Eli S. Saber*†

*†Department of Electrical and Microelectronic Engineering
Rochester Institute of Technology, Rochester, NY, 14623
Email: {yxl7245, sildomar.monteiro, esseee}@rit.edu

Abstract—With rapid developments in satellite and sensor technologies, increasing amount of high spatial resolution aerial images have become available. Classification of these images are important for many remote sensing image understanding tasks, such as image retrieval and object detection. Meanwhile, image classification in the computer vision field is revolutionized with recent popularity of the convolutional neural networks (CNN), based on which the state-of-the-art classification results are achieved. Therefore, the idea of applying the CNN for high resolution aerial image classification is straightforward. However, it is not trivial mainly because the amount of labeled images in remote sensing for training a deep neural network is limited. As a result, transfer learning techniques were adopted for this problem, where the CNN used for the classification problem is pre-trained on a larger dataset beforehand. In this paper, we propose a specific fine-tuning strategy that results in better CNN models for aerial image classification. Extensive experiments were carried out using the proposed approach with different CNN architectures. Our proposed method shows competitive results compared to the existing approaches, indicating the superiority of the proposed fine-tuning algorithm.

I. INTRODUCTION

Recently, the remote sensing images have been obtained with increasing spatial and spectral resolutions. The expansion of the spectral domain advances the spectral-related applications, such as material identification. Meanwhile, the increase in spatial resolution makes the spatial details and textures of the scenes more discernible. Recently, the WorldView-3 satellite sensor provides the images with panchromatic resolution of 0.31m, under which the territorial structures of the scene are accurately captured in the image. Classification of high spatial resolution remote sensing images refers to assigning semantic labels to recognize the images. The problem has become an ongoing research topic in the past decades due to the significant role it plays in facilitating many applications such as vegetation monitoring in agriculture, airport security and aviation safety operation in surveillance, land cover change detection in environmental monitoring, and so on.

Within the literature, a large amount of research efforts have been devoted to the image classification problem, focusing on the design of better feature representations of the images. These image representations ranges from low-level ones, such as color autocorrelogram (ACC) [1] and local color histogram (LCH) [2], to mid-level variants, such as bag of visual words (BoW), and to high-level features that are automatically learned from the image dataset [3]. Recently, deep learning techniques, especially the convolutional neural networks (CNNs), have been applied for image classification in computer vision areas and astounding classification results were achieved. One of the major advantages of applying the CNN approach in computer vision applications is that a large dataset containing millions of labeled images is available for training a robust deep CNN architecture. However, much fewer labeled high resolution aerial images are available. As a result, different transfer learning techniques were adopted in order to apply the CNN approach to classify these images. Transfer learning refers to the technique that pre-trains a deep CNN model on a large but different dataset and then adapts the trained model to specific problems where much smaller image datasets are available.

In this paper, we explore the transfer learning techniques for recognizing the high resolution aerial imagery. First, we extract image features using different pre-trained CNN architectures trained using the ImageNet challenge dataset [4] and learn new classifiers for recognizing our images. Next, we fine-tune the pre-trained CNNs to make the neural network better adapt the aerial imagery. The proposed method was demonstrated to classify the images from the land use land cover (LULC) image dataset [5]. Our contributions include: (1) extract different features using various state-of-the-art CNN architectures for aerial image classification, (2) propose a better fine-tuning framework for remote sensing aerial imagery with small datasets, and (3) perform a comparative study on different transfer learning techniques to better understand the CNN based image features.

II. RELATED WORK

Image classification has been thoroughly studied in the computer vision field and many algorithms have been proposed to solve this problem. Compared to computer vision applications, classification of the aerial images are more challenging given different appearances of the objects randomly rotated within the scene and increasing complexities of the background textures. Initially, the problem was addressed with the bag of word (BoW) model, which extracts image features using the histograms of visual words derived from low level features in the images [5]. However, the BoW model does
features that aggregate the outputs from mid-level layers and from mid-level layers of the CNN, Hu et. al. [17] proposed better image features. In order to make use of the outputs the convolution layers of the pre-trained CNNs to achieve Castelluccio et. al. [16] proposed to fine-tune the weights of CNN models were used for image representations. Later on, and the activation maps form the second last layers of the demonstrated in [3], where the pre-trained neural networks of extract features. The first attempt to apply the pre-trained CNN tion, one possible approach is to apply the pre-trained CNN methods are proposed to learn feature extractors from the image scenes [8], [9]. On the other hand, tremendous progresses have been made on supervised feature learning owing to the development of convolutional neural networks. The groundbreaking work of AlexNet proposed by Krizhevsky et al. [10] learns the image features with a eight-level deep neural network, which consists of five convolution layers and three fully connected layers, using ImageNet [11] consisting of 1000 image categories. The success of the AlexNet lies in several tricks introduced during training, including the application of rectified linear layer, the addition of the “dropout” layer, and the deployment of multiple GPUs. Ever since then, there is an ongoing interest to learn deep CNN architectures for image recognition. Better performances were achieved using other CNN architectures, such as VGG [12], GoogleNet [13], and ResNet [14]. Results from these CNN based algorithms indicate that the more layers the neural network contains, the better results it obtains. However, issues regarding to designing these deep neural network arise as well, leading to many simplification methods of filters to make sure the CNN architectures work properly.

In order to adapt the CNNs for the aerial image classification, one possible approach is to apply the pre-trained CNN models from the large image dataset, such as ImageNet, to extract features. The first attempt to apply the pre-trained CNN architectures on classifying high resolution aerial images was demonstrated in [3], where the pre-trained neural networks of AlexNet and Overfeat [15] were adopted as feature extractors, and the activation maps form the second last layers of the CNN models were used for image representations. Later on, Castelluccio et. al. [16] proposed to fine-tune the weights of the convolution layers of the pre-trained CNNs to achieve better image features. In order to make use of the outputs from mid-level layers of the CNN, Hu et. al. [17] proposed features that aggregate the outputs from mid-level layers and last fully connected layers of the CNN structure with feature coding algorithms, showing improvement on the classification results.

III. METHODS

A. Convolutional Neural Networks

The convolutional neural networks (CNNs) provide end-to-end solutions for many image related tasks, such as image segmentation, recognition, and object detection. Usually, the architecture of a CNN model refers to the structure that contains a sequence of different layers concatenated one next to the other. Each layer is characterized with neurons that perform different differentiable mathematical operations, so as to make the neurons learn-able through back-propagation. In general, typical layers include the convolutional (CONV) layer, the pooling (POOL) layer, the rectified linear layer (ReLU), the fully connected (FC) layer, and the softmax (SOFTMAX) layer. The CONV layer contains a bank of three dimensional filters that convolve over the whole input data and outputs a data cube that concatenates different filter outputs together. The POOL layer aims at reducing the spatial size of input feature map with different down-sampling techniques, such as averaging and taking the maximum of local regions of the input. In the FC layer, each neuron is connected to all the output neurons of the previous layer. Therefore, it can be recognized as a special case of convolutional layer whose receptive field covers the entire spatial region of its input. Finally, the SOFTMAX layer is connected to the last FC layer to generate the classification score. In practise, the deep CNN contains many layers and, therefore, the number of learning parameters grows significantly, which leads to the possibility of long training time and data over-fitting. As a result, multiple approaches were proposed to mitigate these problems. For example, the drop out layer (DROP) is used to randomly drop a pre-defined portion of the parameters in the filters and the batch normalization (NORM) is introduced to normalize each channel of the feature map by averaging over spatial locations with a number of instances.

B. Existing CNN architectures for image classification

Based on various combinations of CNN layers with different parameters, a variety of CNN architectures were derived. In this section, four CNN architectures are briefly discussed: AlexNet [10], VGG-F [12], GoogleNet [13], and ResNet [14]. These pre-trained CNN architectures were trained on the ImageNet [11] dataset that contains over hundreds of thousands of images with 1000 classes.

The input image sizes required for AlexNet and VGG-F are $227 \times 227 \times 3$ and $224 \times 224 \times 3$ respectively. The number of filters and their corresponding spatial receptive field are illustrated in Fig. 1. For example, the CONV1 layer in AlexNet contains 96 convolutional filters with receptive filed of $11 \times 11$, indicating that each filter is convolved with $11 \times 11$ spatial regions across the input image. The pooling layers are inserted after the layers of CONV1, CONV2 and CONV5 to reduce the spatial dimensions of the output.
Fig. 1: CNN architectures for AlexNet and VGG-F with the filter specification indicated as: “numer of filters \( \times \) horizontal receptive field size \( \times \) vertical receptive field size”

![AlexNet architecture](image1)

![VGG-F architecture](image2)

Being the groundbreaking work for image classification, AlexNet contains five CONV layers and three FC layers. After the pooling operation, the spatial dimension of the CONV5 output becomes \( 6 \times 6 \) and thus the spatial receptive field of the FC6 layer is configured as \( 6 \times 6 \). Finally, two more FC layers (FC7 and FC8) are concatenated afterwards to generate the classification result. In order to reduce the number of parameters in the AlexNet, the architecture of VGG-F is proposed. The architecture is very similar to that of the AlexNet, except that the number of convolution filters is significantly reduced (from 96 to 64, and 384 to 256) for three CONV layers (CONV1, CONV3, CONV4).

A straightforward way to improve the performance of the neural network is to increase the depth and width of the neural network. However, the complexity of the network increased tremendously as the number of parameters grows aggressively. As a result, the GoogleNet is proposed to address this challenge. The intuition behind the architecture is based on the observation that the correlation within the image pixels tend to be local. Therefore, it is possible to take the local correlations into account to reduce the number of learning parameters. The GoogleNet solves this issue by introducing the inception modules when designing the CNN architecture, as shown in Fig. 2. Each inception module includes a pooling layer and three convolution filters with spatial sizes of \( 1 \times 1 \), \( 3 \times 3 \), and \( 5 \times 5 \), in order to cover larger receptive field of each cluster. Responses from these filters are then concatenated together as the module output. The \( 1 \times 1 \) convolution layers before the \( 3 \times 3 \) and \( 5 \times 5 \) convolutions are utilized to reduce the dimensions the corresponding filter output. The pooling operation is also utilized since it has shown good performance in previous neural networks, such as AlexNet and VGG-F. It is worth mentioning that the spatial size of these filters are small, being either \( 1 \times 1 \), \( 3 \times 3 \), or \( 5 \times 5 \), resulting in significantly reduced number of learning parameters.

The last pre-trained CNN considered in our paper is ResNet [14], which proposed to add residual learning blocks to learn the residual of non-linear function instead of the function itself. The neural network consists of stacks of residual learning blocks, illustrated in Fig. 3. The basic residual learning block simply contains two \( 3 \times 3 \) CONV layers and learns the difference \( F(x) \) between the output \( F(x) + x \) and input \( x \) of the block. The “bottleneck” version of the residual learning block is also introduced to reduce the computational complexity through interpolation of \( 1 \times 1 \) convolution layers. The two \( 1 \times 1 \) CONV layers are inserted to reduce and increase the dimensions of the data, which is very helpful for designing deeper ResNet. Due to the simplicity that each learning block is composed of mostly \( 3 \times 3 \) CONV layers, the architecture of ResNet usually becomes very deep. For example, the ResNet-152 [14] stacks multiple “bottleneck” building blocks to generate the deep learning architecture with 152 layers.

![ResNet architecture](image3)

### C. Proposed Transfer Learning Schemes for Aerial Image Classification

In our paper, two transfer learning schemes are proposed for aerial image classification, as shown in Fig. 4. The first method feeds the image to the pre-trained CNN models and extract image features using the outputs from different layers. Classification algorithms, such as support vector machines or simple linear classifier, are then employed to classify these features. The second method modifies the classification layer of the existing architectures and train the filter weights of the neural network. The parameters of the filters in different layers are initialized from the pre-trained CNN, except for the last classification layer, whose parameters are initialized with random numbers from Gaussian distribution. During fine-tuning, we proposed to train the filter weights of last few layers only and fix the parameters of the filters from other layers. The reasons behind this learning adaptation are twofold. First,
A. Aerial image classification using pre-trained CNN

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCMerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

A. Aerial image classification using pre-trained CNN

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.

IV. EXPERIMENTAL RESULTS

In this section, we validate the proposed algorithm with experiments on high resolution aerial images from the UCmerced Land Use Land Cover (LULC) [5] dataset, where images of 21 categories are present and each category consists of 100 images instances with spatial resolution of 30cm. We adopt five-fold cross validation through our experiment and the dataset is divided into 5 non-overlapping subsets. During the experiment on each fold, images from 4 subsets are utilized for either training or testing the proposed CNN, and the rest subset is used for testing purpose and generating results. At last, we aggregate the testing results from all the folds by simple averaging operation. Note that we perform data augmentation on the training dataset in both fine-tuning and feature extraction processes. Specifically, the data augmentation procedure crops out a sub-image with required dimensions from the original image and randomly flips the image horizontally.
TABLE I: Classification performance using features from pre-trained CNNs (without fine-tuning).

<table>
<thead>
<tr>
<th>Pre-trained CNN</th>
<th>Feature from 3rd last layer</th>
<th>Feature from 2nd last layer</th>
<th>Feature from last layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>93.62 ± 1.16</td>
<td>94.05 ± 1.33</td>
<td>93.38 ± 1.22</td>
</tr>
<tr>
<td>VGG-F</td>
<td>95.47 ± 1.80</td>
<td>95.24 ± 1.72</td>
<td>93.33 ± 2.51</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>-</td>
<td>94.05 ± 1.64</td>
<td>93.14 ± 1.36</td>
</tr>
<tr>
<td>ResNet</td>
<td>-</td>
<td>96.24 ± 0.80</td>
<td>96.07 ± 1.26</td>
</tr>
</tbody>
</table>

TABLE II: Classification performance using features extracted from fine-tuned CNNs.

<table>
<thead>
<tr>
<th>Fine-tuned layers</th>
<th>All layers</th>
<th>Last few layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>94.37 ± 1.34</td>
<td>95.00 ± 1.74</td>
</tr>
<tr>
<td>VGG-F</td>
<td>95.02 ± 1.27</td>
<td>95.76 ± 1.70</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>93.17 ± 2.25</td>
<td>94.81 ± 1.41</td>
</tr>
<tr>
<td>ResNet</td>
<td>96.05 ± 0.27</td>
<td>97.19 ± 0.57</td>
</tr>
</tbody>
</table>

with proposed fine-tuning algorithm. When comparing the fine-tuning results in Table II to the results from the best feature performance in Table I, it is observed that features from fine-tuning the whole CNN architectures improve with very small amount over the best pre-trained features, however, sometimes perform worse. One of the reasons is that fine-tuning the whole neural network is prone to fall into the over-fitting problem with the limited dataset, since there are too many parameters being updated in this strategy. On the other hand, the proposed fine-tuning approach avoids this problem and consistently improves the pre-trained features. Therefore, we conclude that proposed fine-tuning method better adapts the new dataset by keeping weights from low levels of the pre-trained CNNs and fine-tuning the high-level features.

Even though there are only small differences between two fine-tuning strategies, we argue that the proposed method is more valid in this case. The reasons for this small improvement lies in the following factors: (1) Low level layers of the CNN architectures provide robust image features across different dataset. Therefore it is not necessary to fine-tune the whole neural network, especially the layers corresponding to low-level image features, such as edges and corners. (2) Given the small image dataset, fine-tuning the whole neural network is more likely to over-fit the training data, leading to inferior performance. We proposed to address this issue by fine-tuning on much fewer filter parameters. (3) We argue that fine-tuning the whole network is more likely to lead to over-fitting, although it is still likely to get good performance, given a well performed pre-trained neural network and small learning rates during the fine-tuning process.

In order to qualitatively evaluate the performances of different features (pre-trained model, pre-trained model with fine-tuning on all layers, pre-trained model with proposed fine-tuning method), we adopt the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm proposed in [18] to visualize the feature distributions for different classes. The algorithm performs dimensionality reduction on high-dimensional data and, at the same time, preserves the significant structures of high-dimensional data. With all the CNN models available, we extract image features for the original image dataset with 2100 images and apply t-SNE to reduce these image features to two dimensions. The scatter plots of these images in the corresponding reduced two-dimensional feature space are illustrated in Fig. 5. Each point in the plot represents an image in the dataset and different classes are indicated with different colors. From these plots, it is obvious to see that image features extracted from the proposed fine-tuning method tend to cluster better in the reduced two dimensional feature space than the features extracted using other two schemes, therefore it is easier to differentiate them using a designed classifier.

The other way to compare the performances of different CNN models is to visualize the filter weights and its corresponding activation outputs of various layers. We apply the toolbox provided by Yosinski et al. [19] for this purpose. In order to demonstrate the idea, an “airplane” image from the dataset is fed into the modified pre-trained AlexNet model, the fine-tuned AlexNet model using the traditional fine-tune method, and the fine-tuned AlexNet model with the proposed approach. For each layer of the network, the toolbox plots the learned weights and activation maps of the filters from different layers. Outputs from the low-level filters respond to edges and corners in the image while high-level filters from CONV5 create meaningful objects. The last layer of AlexNet contains 21 neurons, each one corresponds to the probability of the image being classified as a specific class. The “CONV5” response using the proposed fine-tuned model is shown in Fig. 6. We also display the classification outputs from different CNN models in the same figure. According to these results, only the model from our proposed fine-tuning scheme correctly classify the image as an airplane (with probability of 0.88), while the other two methods classify the image as “intersection” with probability of 0.5 and 0.41 respectively, and consider it as an airplane with probability of 0.3. To better understand what what filters in “CONV5” layer learn from our dataset, we manually feed all the “airplane” image scenes from the UCMerced dataset [5] to the proposed fine-tuned AlexNet model and visualize the filter responses from “CONV5” layer. We observe that one of the neurons from “CONV5” layer obtains high response for image scenes with single blue plane, as indicated in Fig. 7. These responses also indicate that the neuron is capable to detect planes with different orientations.

V. DISCUSSION

One of the major difficulties regrading to training a deep neural network is to make sure that the trained neural network is not over-fitted on the training dataset. As a result, the drop out (DROP) layer was proposed in AlexNet and VGG-F to reduce the number of training parameters when learning CNN architectures. On the other hand, the GoogleNet and ResNet
reduce the number of learning parameters by designing filters with smaller spatial receptive fields while at the same time extend the depth of the neural network. In the fine-tuning problem, data augmentation that generates more training images is preferred for training a deep neural network. However, in this paper, we propose to reduce the number of training parameters by fine-tuning few layers of the CNN architecture, instead of the whole network. Through experiments on aerial image classification using the pre-trained CNN architectures, we argue that low levels of the pre-trained CNN architectures generalize well to other classification tasks. Therefore, we argue that only higher levels of the architectures are needed for fine-tuning. In order to determine which layers are needed for fine-tuning, we carry out image classification experiments based on image features directly extracted from pre-trained CNN architectures. One possible reason for getting better results with less number of layers for fine-tuning is that a large number of training parameters are reduced, which is desirable given the small number of images in the dataset.
techniques that modify a small number of layers only, 2) Investigate other data augmentation techniques for generating more images, in the hope of getting more layers involved during the fine-tuning process, 3) Visualize the differences between different fine-tuning methods and analyze the responses of different neurons for recognizing different scenes/objects.

ACKNOWLEDGMENT

The authors would like to thank Yangqing Jia et. al. for making the Caffe toolbox available online.

REFERENCES

Fig. 7: “CONV5” layer activation visualization of airplane scenes that contain a single blue airplane, filters indicated with red “bounding-box” have high responses on different orientations of the airplane.