Determining Feature Extractors for Unsupervised Learning on Satellite Images

Behnam Hedayatnia*, Mehrdad Yazdani†‡, Mai Nguyen§, Jessica Block†, and Ilkay Altintas§
*Department of Electrical and Computer Engineering, UC San Diego, California, USA
†California Institute for Telecommunications and Information Technology, UC San Diego, California, USA
‡Open Medicine Institute, Mountain View, California, USA
§San Diego Supercomputer Center, UC San Diego, California, USA
bhedayat@eng.ucsd.edu, myazdani@ucsd.edu, mhnguyen@sdsc.edu, j.block@eng.ucsd.edu, altintas@sdsc.edu

***Abstract*— Advances in satellite imagery presents unprecedented opportunities for understanding natural and social phenomena at global and regional scales. Although the field of satellite remote sensing has evaluated imperative questions to human and environmental sustainability, scaling those techniques to very high spatial resolutions at regional scales remains a challenge. Satellite imagery is now more accessible with greater spatial, spectral and temporal resolution creating a data bottleneck in identifying the content of images. Because satellite images are unlabeled, unsupervised methods allow us to organize images into coherent groups or clusters. However, the performance of unsupervised methods, like all other machine learning methods, depends on features. Recent studies using features from pre-trained networks have shown promise for learning in new datasets. This suggests that features from pre-trained networks can be used for learning in temporally and spatially dynamic data sources such as satellite imagery. It is not clear, however, which features from which layer and network architecture should be used for learning new tasks. In this paper, we present an approach to evaluate the transferability of features from pre-trained Deep Convolutional Neural Networks for satellite imagery. We explore and evaluate different features and feature combinations extracted from various deep network architectures, and systematically evaluate over 2,000 network-layer combinations. In addition, we test the transferability of our engineered features and learned features from an unlabeled dataset to a different labeled dataset. Our feature engineering and learning are done on the unlabeled Draper Satellite Chronology dataset, and we test on the labeled UC Merced Land dataset to achieve near state-of-the-art classification results. These results suggest that even without any or minimal training, these networks can generalize well to other datasets. This method could be useful in the task of clustering unlabeled images and other unsupervised machine learning tasks.**

***Keywords*—** satellite imagery, remote sensing, deep learning, convolutional neural networks, transfer learning

I. Background

Satellite remote sensing research in the past decade has evaluated imperative issues to human and environmental sustainability including urban growth [1]–[3], urban quality of life [4]–[6] , identifying socioeconomic indicators and slums [7], [8], burn severity from wildfires [9], and forest cover change [10], to name a few. While satellite imagery is increasingly accessible, existing techniques for analyzing satellite imagery is increasingly compute intensive as well. Only in the last few years there have been research that analyzes imagery at national and global scales [10], [11], which are keeping up with the volume and resolution of imagery available. Their techniques require many hours of manual classifications, in the case of [11], to validate the analysis, and intensive supercomputing and classification in the case of [10]. Identifying the content of images efficiently remains a bottleneck. As the rate of change occurring on the Earth's surface increases (due to climate change, industrialization, massive migrations, among numerous other causes), being able to measure and monitor these changes are ever more critical, and labeling the content of satellite and aerial scene images is a prohibitively daunting task.

Automated methods and machine learning provide us the opportunity to track changes of scene imagery over time at scales that do not depend on human intervention. Recently deep learning methods for computer vision applications have brought impressive results [12]–[15]. A common theme with deep learning in computer vision is that architectures are based on Convolutional Neural Networks (CNNs) [12] consisting of many computation layers that allow for the learning of high level features.

In addition to architectural and algorithmic designs, the success of deep CNNs depend on large datasets with labels to achieve state-of-the-art supervised learning results. On the ImageNet [16] dataset that consists on the order of 1 million labeled images, He et al. [17] for example have designed and trained a network that surpasses human accuracy for the image classification task. At first glance, the volume of labeled images may appear to pose a challenge for using deep learning in new applications or new datasets. However, many studies [18]–[21] have shown that the features learned on the ImageNet dataset for classification tasks can be used for new image classification tasks and new datasets. This method of using features trained from one dataset for a completely different dataset with different categories is sometimes referred to as transfer learning. In other words, when learning a new task, instead of starting with a network initialized with random weights, we use the weights of the network learned from a different (and typically much larger) dataset for the new task. Sharif et al. [18] suggest that transfer learning should be used as an initial step for determining the baseline performance of a machine learning task. In this work, we investigate how transferable and generalizable are different networks and layers on satellite and aerial scene

978-1-4673-9005-7/16/$31.00 ©2016 IEEE

imagery.

The key contributions from this paper are on evaluating the transferability of features from a comprehensive set of networks and layers for the purpose of clustering and exploring the effect of fine-tuning with spatial data on these networks. We specifically:

1) Exhaustively evaluate over 2,000 CNN architecture and layer combinations for transferability from the ImageNet dataset to spatial satellite datasets.
2) Test the transferability of CNN features from unlabeled spatial data to labeled spatial data using pretrained networks and fine-tuned networks.

II. Related Work

We are not the first to look at the generalization of deep networks to adapt to other domains. Yosinski et al. [20] investigate exhaustively the transferability of an 8-layer CNN network based on the architecture of Krizhevsky et al. [12]. This is done by first training on an initial dataset, and then transferring the learned features to a second new network to be trained on a different dataset. Their results show that features transferred from different datasets are better than random weights when training on new tasks. By controlling for and fixing different layers, Yosinski et al. [20] also shows that higher layers are more specialized for the specific task, whereas lower layers are more general and less specific features. This suggests that higher layers are less transferable than lower layers as performance drops on high layer features. Our results, presented in this paper, further confirm this observation.

In remote sensing, labeling scene images is an especially challenging problem since the scene may have large variability and many aerial shots come without labels [21]–[23]. Castelluccio et al. [23] discusses using and comparing two specific networks (CaffeNet [24] and GoogLeNet [14]) trained on the ImageNet dataset for aerial image classification. Castelluccio et al. [23] specifically investigates: *(i)* sending images through the CNN, and extracting feature vectors from a penultimate layer used to train an off-line classifier; and *(ii)* using the training images to fine-tune the CNN for another task. In this paper, we focus on the former and look at more pre-trained networks than just CaffeNet and GoogLeNet, including more recent networks.

In addition, an important part of our paper is the use of combining features from multiple networks together to achieve better performance. Penatti et al. [22] explored this idea of feature fusion through the concatenation of the feature vectors computed by CaffeNet [24] and OverFeat [25]. In this paper, we significantly expand the number of networks and layers compared to previous studies by investigating 5 different networks and 11 different layers resulting in over 2,000 network-layer combinations. We also explore the idea of feature sub-selection techniques after concatenating features.

The rest of this paper discusses the methods (Section III), experimental results (Section IV) and scalable automation (Section V) of the contributions described above.

III. Methods

A. Datasets

We are using the Draper Satellite Image Chronology dataset [26] acquired from Kaggle which provides aerial shots of different locations. Images are grouped into sets of five where each image in a set was taken on a different day at a specific location, but not necessarily at the same time each day. The images for each set cover about the same area but are not perfectly aligned. Some locations show little evidence of changes from day to day while other locations will have subtle changes such as moving vehicles or changes in shadows. To see an example of this dataset, refer to row 1 of Figure 2.

TABLE I: Summary of Draper Dataset

Image Set	Number of Query Images	Number of Total Images
Original Set	70	350
Validation Set	50	250
Test Set	20	100

The Draper dataset contains 350 images total with dimensions 2329x3099. There are 70 different locations each with their own set of 5 images. These images are a priority dataset because they are labeled by their location and are grouped together as sets of 5 images which allows us to quantify the performance of a pre-trained network's layer as a feature extractor. The dataset can be found at [26]. As we see later, the dataset will be split into a validation set and testing set. The validation set contains a total of 250 images or 50 different locations and the test set contains a total of 100 images or 20 different locations. We denote the first image in a set of 5 images as the query image for that location. Table I summarizes how the data is split for validation and testing.

We also use a labeled dataset for additional testing and validation of our methodologies. We use the UC Merced dataset that is composed of 2,100 aerial scene images with 256 x 256 pixels divided into 21 land-use classes selected from the United States Geological Survey (USGS) National Map. The images are from diverse categories such as: agricultural, airplane, baseball diamond, beach, buildings, chaparral, dense residential, forest, freeway, golf course, harbor, intersection, medium density residential, mobile home park, overpass, parking lot, river, runway, sparse residential, storage tanks, and tennis courts. It is publicly available [27].

B. Overview of Network Architecture

We use the Caffe neural network library to utilize pre-trained models downloaded off Caffe Model Zoo [24]. A table of the networks used and from which layers were used to extract features is shown in Table II.

C. Feature Extraction

The method of utilizing pre-trained networks to apply to another learning task is known as transfer learning. Transfer learning allows us to apply the features extracted from

TABLE II: Pre-trained networks used using Caffe notation [24]

Network	Number of Layers	Layers Used	Our Notation
CaffeNet [24]	15	fc6,fc7,fc8	f_1,f_2,f_3
AlexNet [12]	15	fc6,fc7,fc8	f_4,f_5,f_6
VGGNet [13]	14	fc6,fc7,fc8	f_7,f_8,f_9
GoogLeNet [14]	121	loss3/classifier	f_{10}
ResNet-151 [15]	310	fc1000	f_{11}

large existing already-labeled datasets onto a new task. This approach is motivated by the fact that deep neural networks are supposed to find features in the representation of the task and that features that are learned early in the network are general enough that they can be applied successfully to another image dataset.

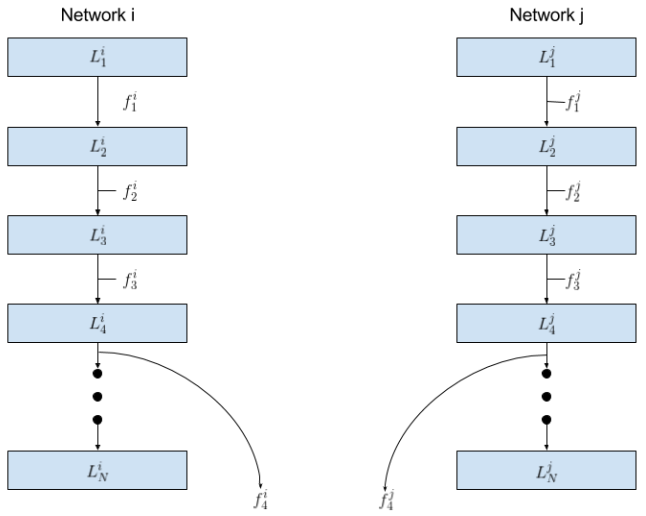

Fig. 1: Feature Extraction is done by sending a set of images through a network and extracting features at a certain layer L_k which results in a feature set f_k for a certain Network. E.g. we can extract a feature set f_4^i and f_4^j for two different networks

The process of transfer learning involves sending our own images through the network and extracting features at a certain layer. This process is illustrated by Figure 1. The features extracted are from a fully connected layer since the output parameters are more reasonable to work with. The output of a fully connected layer from an image is for example, a vector of size 1x1000 which is easier to manipulate as our feature vector rather than an output from a convolutional layer which may be a tensor of size 1x3x100x100.

This process is different from fine-tuning because we are not training our images and do not change the number of classes in the softmax layer of the network. Rather we use the parameters learned by a pre-trained model to see if it can be used for an unlabeled dataset where fine-tuning would not be possible.

A constraint of pre-trained models is that one may be restricted to the architecture of the network and may not be able to take out arbitrary convolutional or pooling feature layers from the pre-trained network. This constraint is especially apparent with GoogLeNet whose network architecture consists of inception layers. In this architecture, we extract features from the very last fully connected layer. This may have contributed to GoogLeNet's poor performance as feature extractors without the help of fine-tuning, which will be discussed later.

D. Feature Evaluation

To quantify performance on a newly extracted feature set, we use Nearest Neighbor Classification with Euclidean Distance. The process follows by fixing a query image from a set of 5 images of the same location, which we will denote by X_1. After applying Nearest Neighbors, the output will be X_2, X_3, X_4, X_5 which denote the 4 nearest neighbors to image X_1. Ideally, these 4 images should be the same location as the query image but taken at different times throughout the day with slight changes in camera angle. Nearest neighbor feature evaluation is illustrated by Figure 2. Once again, from our total set of 350 images there are 70 query images.

The metric we use to evaluate network accuracy is to count how many labeled images each nearest neighbor iteration gets correct and divide this by 4 for the total number of images it needs to get correct. This is done for each query image in the dataset. Note that this means the process is not repeated for two images in the same set of 5 images. The average is taken for the final accuracy of a certain layer for its respective network. A fallback of this metric is that it does not weigh images closer to the query image more than images farther away.

By utilizing the notion of location from the Draper dataset we can provide a more quantitative way of evaluating these networks across various layers, rather than using a heuristic evaluation from an unlabeled dataset. The best feature set from a network could then be used on an unlabeled dataset. Equation 1 calculates the accuracy for each query image, by finding how many images predicted for a query image match with the ground truth and dividing that by the number of nearest neighbors.

$$\text{Accuracy for one query image } Q = \frac{|\{X_Q^P\} \cap \{X_Q^T\}|}{N} \quad (1)$$

where N are the number of nearest neighbors (4 in our case), X_Q^P is the set of predicted nearest neighbor images to the query image Q, X_Q^T is the set of actual neighboring images. Note that we are not considering the ordering of the images that result from computing distances of the KNN algorithm hence we define both X_Q^P and X_Q^T as sets. In other words, Equation 1 computes the number of matches between the 4 nearest neighbors we find from a query image compared to ground truth (see Figure 2).

The dataset is split such that approximately 71% of the images we use to validate across all networks and layers

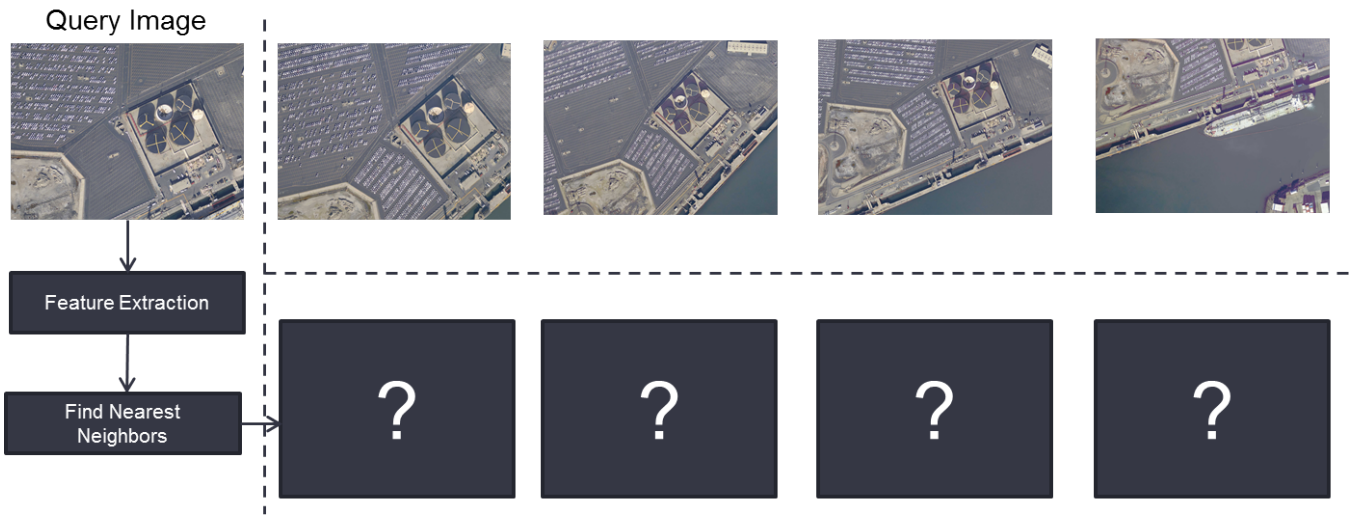

Fig. 2: The first row shows the Ground Truth for a query image. As seen, the images for each ground truth set cover approximately the same area but are not exactly aligned. The second row aims to find the 4 nearest neighbors to the query image using our feature set with the goal being to match those images to the ground truth. The accuracy for one query image is calculated using Equation 1 by finding how many images are in common with the ground truth and dividing that by the number of nearest neighbors.

and see which will give the highest accuracy. The remaining 29% of the dataset is used as a test to confirm the strength of the network. Splitting the dataset this way gives 250 images total or 50 query images to validate and 100 images or 20 query images to test.

E. Applying Feature Extraction to UC Merced Dataset

To quantify the performance of the feature combinations that we acquire, we evaluate on the labeled UC Merced dataset. That is, we determine the subset of feature combinations that yield the best results on the Draper dataset and now we want to test how this subset transfers to the labeled UC Merced dataset.

F. Fine-tuning

So far our feature extractors have been pre-trained networks which were originally trained on ImageNet, a general image dataset. Therefore, the networks did not get a chance to be trained on any spatial data or satellite imagery.

To introduce satellite imagery into the network we utilize the method of fine-tuning. Fine-tuning is a method to replace and retrain the softmax layer on top of the network on a new dataset and to fine-tune the weights of the pretrained network. We use the Kaggle Draper dataset from before and will use the locations as our labels. For each location the set of 5 images is split into 3 for training and 2 for validation and in total there are 344 locations or classes. Using this form of supervised learning is justified for our unsupervised problem because it is easy to obtain images from the same location and we are not giving it specific labels such as this is an image of a river or road.

Once trained we send the UC Merced dataset through the network and perform feature extraction. We test the performance of these features and run a Support Vector Classifier with 5-fold cross validation (as was done in [22], [23])

IV. Experiments and Results

A. Individual Pre-trained Networks as feature extractors

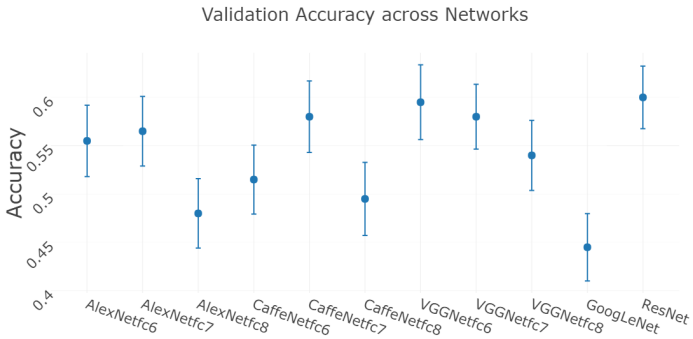

Fig. 3: Accuracies on validation set on Pretrained Networks using 50 query images. Higher accuracies for different networks and layers suggest better transferability for satellite imagery tasks.

Results are seen in Figure 3[1], note that a random classifier would achieve an accuracy of $\frac{1}{\binom{249}{4}}$ or 6.39e-7%. As seen from Figure 3, the layers that gave the best performance from CaffeNet and AlexNet were the fully connected 7th layer. The best layer from VGGNet was the fully connected 6th layer. The fully connected 8th layer, which is right before the softmax layer, had the lowest accuracy across all these three networks: CaffeNet, AlexNet and VGGNet. This result

[1] https://plot.ly/~bhedayat/6/validation-accuracy-across-networks.embed

confirms with Yosinski et al. [20]. that the last layer of these networks are too specific to the dataset that were originally used to train these pre-trained networks and has a high variance when used on a new dataset. The fully connected 6th layer for CaffeNet and AlexNet has too much bias or has under-fitted to use on a new dataset.

GoogLeNet had the worst performance, possibly due to the fact that its features were too specific to the dataset the network was trained on. The best network performance was ResNet fc1000 which was able to generalize well to the Draper satellite dataset.

It seems that residual networks are good for inference due to the identity layers which force the network to learn better features and understanding the image as a whole. Deeper networks also allow for a more high level feature extraction as well. This is important for spatial data which can be very cluttered and can have multiple objects within the image causing difficulties for the network to decide what is the picture supposed to be. ResNet was able to overcome this obstacle the best.

TABLE III: Tested on Networks using 20 query images

Network	Layer Used	Accuracy
ResNet	fc1000	66%
VGGNet	fc6	68.7%

VGGNet fully connected layer 6 and ResNet fc1000 were tested on the 20 query images to show good consistent performance in Table III.

B. Combining Pre-trained Networks as feature extractors

Next, feature sets across networks are combined to see if they could aid in performance. The total number of combinations was $\binom{N}{i}$ where i $\epsilon\{2,\ldots,N\}$ and N is number of feature sets. We have a total of 11 different feature sets. In total there were 2036 feature sets considered on the 50 validation query images.

An example of a feature set concatenated from a combination of $\binom{N}{2}$ using our notation would be $[f_2, f_3]$.

As shown in Figure 4[2], the highest performance achieved an accuracy of 68% and was a combination of the networks: ResNet fc1000, AlexNet 8th fully connected layer, VGGNet 7th fully connected layer and CaffeNet 7th fully connected layer. After seeing performance from single networks we see than even the worst performance from a single network can be improved through concatenation as the accuracy does not go below 50%. This was then tested on the 20 test query images which gave an accuracy of 72.5% with a standard error of 4.29%. The vector length from this concatenated feature set was 10,192 features long.

The feature set consisting of ResNet fc1000,AlexNet 8th fully connected layer, VGG 7th fully connected layer and Caffe Net 7th fully connected layer will be denoted as RAVC and is used for further analysis.

[2]https://plot.ly/~bhedayat/15/accuracy-across-concatenated-networks.embed

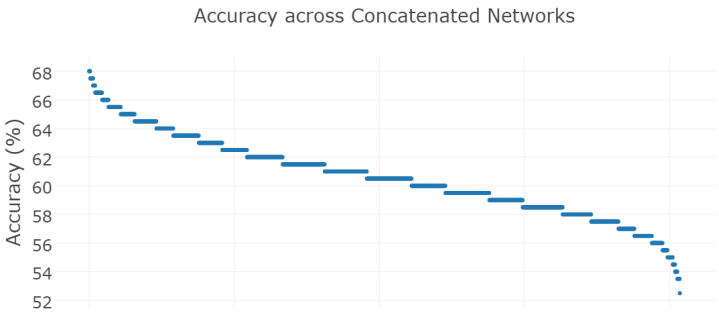

Fig. 4: Accuracies on validation set for Concatenating Pre-trained Networks together. The horizontal bars suggest there is a single dominant network feature set in the concatenated feature set resulting in same accuracies. Over 2000 Feature sets were considered.

C. PCA on concatenated Feature set

Next, we use principal component analysis (PCA) on the concatenated feature set, RAVC, to reduce dimensionality while retaining 99.2% of the variation to see if there is an improvement in performance. Using the 20 test query images the accuracy stays relatively the same at 71.25% with a standard error of 4.07%.

Since we are tackling this unsupervised problem with a combination of multiple pre-trained networks this result shows that we do not have worry about the curse of dimensionality since features from these networks are good discriminators.

D. Feature subsampling on concatenated Feature set

After PCA, we use the feature subsampling technique used in Random Forests to figure out which features are deemed important. This was done by training a Random Forest with 250 trees. Training was done by creating labels Ex: 1,1,1,1,1,2,2,2,2,2 etc. 5 repeats of a same label because there are 5 images for each location. The feature importance method from the scikit random forest class [28] was then used to get important features. The training was done with the RAVC feature set from the 50 query images or validation set.

Shown in Figure 5 is a log plot of the important features where the x axis goes from most important to the least important features. The most important features do not have a high value of importance and the feature importance slowly decreases throughout all the features. Also seen is that ResNet's features coincide more with the important features.

Figure 6 shows the trend in accuracy for Nearest Neighbor feature evaluation as more features are added. Subsampling features does not improve the network as the accuracy does not go above 68%.

For completeness, we randomly sub sampled 2,100 features from the RAVC feature set 100 times. The reason 2,100 features were sub sampled was because that showed a big jump in accuracy in Figure 6. The 2,100 features that was

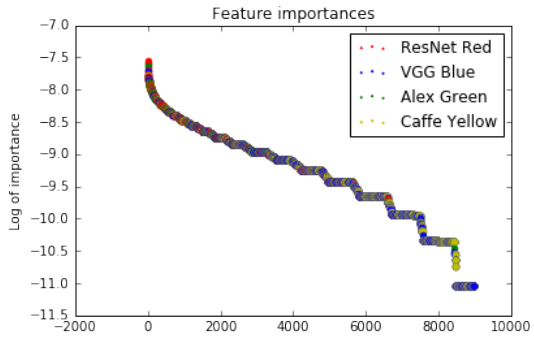

Fig. 5: Log Plot of Important Features

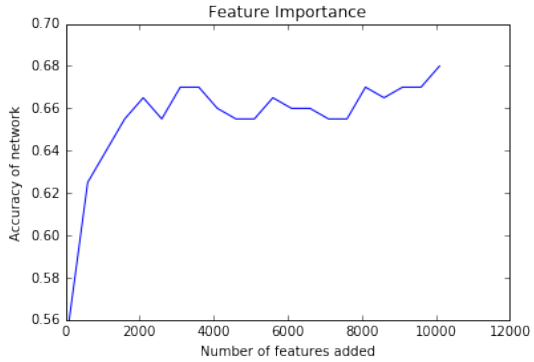

Fig. 6: Accuracy as features added

subsampled from the feature set achieved about the same accuracy of 66% as compared to random forest subsampling of 2,100 features. This provides us evidence that deep neural networks provide good discriminant features.

E. Visualizing the Nearest Neighbors

Figures 7 to 11 are query images and their corresponding nearest neighbors using our feature set. Image 0 is the query image and images 1 through 4 are the 4 closest neighbors with image 1 being the closest. This is to note that even though the network accuracy does not look statistically high, this method can still be used to cluster images that look alike. As seen in Figure 7, even though it got none of the ground truth images correct, the images predicted still look similar to the query image.

F. Applying Feature Extraction to UC Merced Dataset

To further test the feature and architecture combinations that we present in this paper, we evaluate on the labeled UC Merced dataset. That is, in previous sections we determined the subset of feature combinations that yield the best results on the Draper dataset and now we want to test how this subset transfers to the labeled UC Merced dataset.

To present a fair test of our methodology we do not rerun the full 2,036 concatenated feature set extraction, rather we extract features only from ResNet fc1000, VGGNet fc7,CaffeNet fc7 and AlexNet fc8. We run a Support Vector Classifier with 5-fold cross validation (as was done in [22], [23]) on 4 feature sets as shown in Table IV. A row

Fig. 7: Image with 0% correct Neighbors

Fig. 8: Image with 25% correct Neighbors

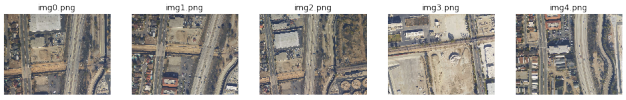

Fig. 9: Image with 50% correct Neighbors

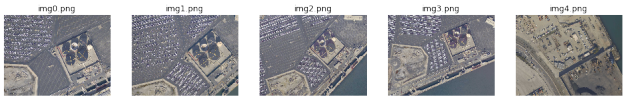

Fig. 10: Image with 75% correct Neighbors

Fig. 11: Image with 100% correct Neighbors

TABLE IV: Results for UC Merced Dataset

Feature Sets	Average accuracy
f_{11}	94.8%
f_8,f_{11}	96.6%
f_8,f_{11},f_2	96.9%
f_8,f_{11},f_2,f_6	**97.1**%

containing more than one feature set means those feature sets were concatenated together and we our notation to denote which feature sets we use. Our results show that feature combination as designed from one particular dataset using our method, transfers well to a new dataset. Using the feature set RAVC results in the best accuracy with 97.1% as opposed to just using ResNet which gave 94.8%.

We can see that the biggest increase in accuracy from Table IV is the jump of 2% from VGG's layer[3]. AlexNet and CaffeNet contribute only about 0.3% and 0.2% respectively. This suggests that both ResNet and VGG have design parameters that work well with spatial data without any sort of training.

G. Fine-tuning Network

We fine-tune AlexNet for 500 iterations with data augmentation such as random cropping and flipping the images. The training loss versus validation accuracy is shown in Figure 12.

Once trained we send the UC Merced dataset through the network and perform feature extraction. We extract features

[3] f_8 is a layer from VGG, f_{11} from ResNet-151, $f2$ from CaffeNet, and f_6 from AlexNet. See also Table II for our nomenclature.

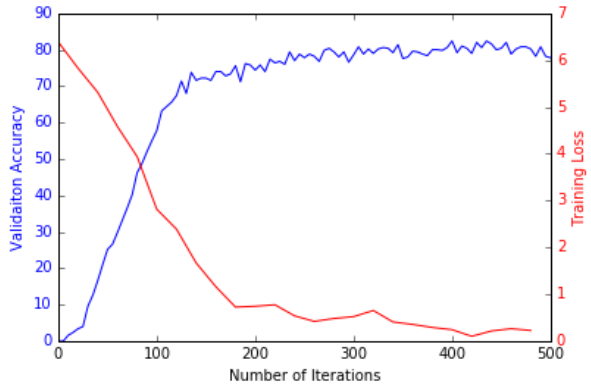

Fig. 12: Training Loss vs Validation Accuracy AlexNet

at different times of training. We test performance with a Support Vector Classifier with 5-fold cross validation. Zero iterations of training is equivalent to no fine-tuning and is used as our baseline. The results are shown in Table V

TABLE V: Results for Fine-tuning

Network	Number of Iterations	Average Accuracy
AlexNet	0	90.62%
AlexNet	100	91.52%
AlexNet	400	91.24%

As seen in Table V with fine-tuning we achieve a performance jump from 90.62% to 91.52% on AlexNet. This shows that by introducing spatial data into our network we can achieve higher performance when we extract features later on a different dataset. However we are also prone to overfitting as seen in Table IV where there is a drop in performance to 91.24%. We wish to continue to explore other architectures to see if fine-tuning them will show improvement.

V. Scaling Up

The feature extraction method that was described in Section III can be used to cluster images for applications such as image segmentation, image retrieval, and image labeling. All of these applications require the scalability of the developed methodology to big datasets. To support large-scale image processing, we are implementing an automation of the process of feature extraction followed by clustering using a workflow system along with a scalable clustering algorithm. Specifically, we are using the Kepler workflow system and the k-means clustering algorithm in Spark MLlib [29].

Spark [30] is a general engine for computation in a cluster environment. Spark provides mechanisms to support efficient distributed processing. Spark also provides scalable machine learning algorithms in its machine learning library, MLlib.

Kepler [31] is a scientific workflow system that provides a graphical user interface (GUI)-based approach to designing and executing the steps in a scientific process. Each step or operation that needs to be performed is represented by an

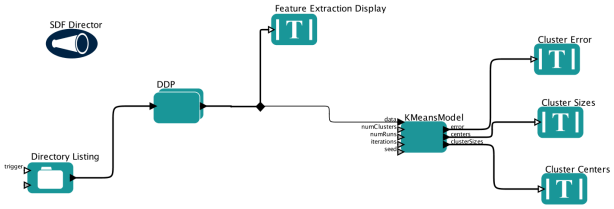

Fig. 13: Image processing workflow at the top level

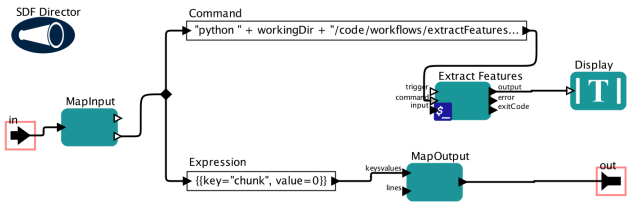

Fig. 14: Drilling down on the workflow. This shows operations on image slices that can be processed in parallel. Features can be extracted from multiple images at the same time

actor in Kepler. A director component is used to control the flow of execution in the workflow. Through its DDP (Distributed Data Parallel) components [32], Kepler allows for sub-workflows to be performed in parallel on top of big data platforms, e.g., Hadoop, Spark and Flink [33].

The top-level workflow in Kepler for our image processing process is shown in Figure 13. The DDP actor is where feature extraction of multiple images can be performed in parallel, using Map operation over the sub-workflow for feature extraction inside the DDP actor and gets executed in Spark. Since features can be extracted from each image slice separately and independently, multiple image slices can be processed in parallel without affecting the resulting features.

Figure 14 drills down and shows the operations that are performed for each image slice. Since the same operations are performed on all image slices to extract features, these operations can be executed in parallel, allowing several images to be processed at the same time. The amount of time to extract features from image slices can thus be significantly reduced.

To further increase scalability, we can use Spark MLlibs scalable implementation of the k-means algorithm, which speeds up processing of the k-means clustering algorithm. This is depicted as the KMeansModel actor in Figure 13.

The combination of Kepler and Spark provides a two-pronged approach to scale up our process to datasets with large images, while making the process repeatable at scale for multiple datasets. Using a workflow approach to execute operations in parallel together with a scalable clustering algorithm will maximize the benefits of a distributed environment.

VI. Conclusion

In this paper, we evaluate pre-trained ConvNets using basic transfer learning for satellite or aerial images. We quantify which layers from which networks provide the best features and combine various network's features together to improve performance. We also use feature subselection techniques such as the one utilized by random forests to see if we can pick out important discriminating features. We see that performance does not change significantly compared to randomly selected baseline set of features from a concatenated network. This suggests that most deep networks provide discriminant and robust features.

The use of pre-trained convolutional networks can be very beneficial for other tasks especially if the data is unlabeled. We also introduce spatial data into our pre-trained networks through the method of fine-tuning and see that performance can improve. For spatial data, labels can be very noisy due to the fact that there are many objects in a satellite image and deep networks such as ResNet are able to perform well with these cluttered images. We also discuss an approach to scale up our process using a workflow system and distributed processing to support large-scale image applications.

VII. Future Work

There are still many research directions regarding how pre-trained networks can be combined with other deep learning technologies to improve the process of unsupervised learning. Another approach is to fine-tune the deeper networks such as ResNet and then take the step of concatenating them which has been shown to have higher performance [23]. An important purpose of these techniques is to be able to identify features in a landscape and be able to map those features back on Earth. Future work will preserve the geospatial accuracy of the imagery to dynamically plot the resulting classifications back on a map. We will also experiment with using image analysis products such as the Normalized Difference Vegetation index (NDVI) and Normalized Difference Built-up Index (NDBI) as input features to refine results of the clustering in order to test the value of thermal spectral data in satellite imagery for classification using these methods.

VIII. Acknowledgments

The work presented in this paper was supported in part by NSF 1331615 under CI, Information Technology Research and SEES Hazards programs, NSF DBI-1062565 under CI Reuse and Advances in Bioinformatics programs, and the San Diego Supercomputer Center Workflows for Data Science (WorDS) Center of Excellence. We would also like to thank Kaggle and Draper Satellite for their dataset.

References

[1] R. B. Miller and C. Small, "Cities from space: potential applications of remote sensing in urban environmental research and policy," *Environmental Science & Policy*, vol. 6, no. 2, pp. 129–137, 2003.

[2] A. Belal and F. Moghanm, "Detecting urban growth using remote sensing and gis techniques in al gharbiya governorate, egypt," *The Egyptian Journal of Remote Sensing and Space Science*, vol. 14, no. 2, pp. 73–79, 2011.

[3] E. Besussi, N. Chin, M. Batty, and P. Longley, "The structure and form of urban settlements," in *Remote sensing of urban and suburban areas*. Springer, 2010, pp. 13–31.

[4] R. Jensen, J. Gatrell, J. Boulton, and B. Harper, "Using remote sensing and geographic information systems to study urban quality of life and urban forest amenities," *Ecology and Society*, vol. 9, no. 5, p. 5, 2004.

[5] B. Bhatta, S. Saraswati, and D. Bandyopadhyay, "Quantifying the degree-of-freedom, degree-of-sprawl, and degree-of-goodness of urban growth from remote sensing data," *Applied Geography*, vol. 30, no. 1, pp. 96–111, 2010.

[6] G. Li and Q. Weng, "Measuring the quality of life in city of indianapolis by integration of remote sensing and census data," *International Journal of Remote Sensing*, vol. 28, no. 2, pp. 249–267, 2007.

[7] P. Hofmann, J. Strobl, T. Blaschke, and H. Kux, "Detecting informal settlements from quickbird data in rio de janeiro using an object based approach," in *Object-based image analysis*. Springer, 2008, pp. 531–553.

[8] F. J. Tapiador, S. Avelar, C. Tavares-Corrêa, and R. Zah, "Deriving fine-scale socioeconomic information of urban areas using very high-resolution satellite imagery," *International journal of remote sensing*, vol. 32, no. 21, pp. 6437–6456, 2011.

[9] F. Mouillot, M. G. Schultz, C. Yue, P. Cadule, K. Tansey, P. Ciais, and E. Chuvieco, "Ten years of global burned area products from spaceborne remote sensinga review: Analysis of user needs and recommendations for future developments," *International Journal of Applied Earth Observation and Geoinformation*, vol. 26, pp. 64–79, 2014.

[10] M. C. Hansen, P. V. Potapov, R. Moore, M. Hancher, S. Turubanova, A. Tyukavina, D. Thau, S. Stehman, S. Goetz, T. Loveland *et al.*, "High-resolution global maps of 21st-century forest cover change," *Science*, vol. 342, no. 6160, pp. 850–853, 2013.

[11] R. Goldblatt, W. You, G. Hanson, and A. K. Khandelwal, "Detecting the boundaries of urban areas in india: A dataset for pixel-based image classification in google earth engine," *Remote Sensing*, vol. 8, no. 8, p. 634, 2016.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[18] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 806–813.

[19] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1717–1724.

[20] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[21] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sensing*, vol. 7, no. 11, pp. 14 680–14 707, 2015.

[22] O. A. Penatti, K. Nogueira, and J. A. dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 44–51.

[23] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, "Land use classification in remote sensing images by convolutional neural networks," *arXiv preprint arXiv:1508.00092*, 2015.

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[25] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[26] Draper, "Draper satellite image chronology," 2016, [Online; accessed 22-April-2016]. [Online]. Available: https://www.kaggle.com/c/draper-satellite-image-chronology

[27] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proceedings of the 18th SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2010, pp. 270–279.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[29] X. Meng, J. Bradley, B. Yuvaz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *JMLR*, vol. 17, no. 34, pp. 1–7, 2016.

[30] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets." *HotCloud*, vol. 10, pp. 10–10, 2010.

[31] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, "Scientific workflow management and the kepler system," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[32] J. Wang, D. Crawl, I. Altintas, and W. Li, "Big data applications using workflows for data parallel computing," *Computing in Science Engineering*, vol. 16, no. 4, pp. 11–21, July 2014.

[33] J. Wang, Y. Tang, M. Nguyen, and I. Altintas, "A scalable data science workflow approach for big data bayesian network learning," in *Proceedings of the 2014 IEEE/ACM International Symposium on Big Data Computing*. IEEE Computer Society, 2014, pp. 16–25.