

Multiview Deep Learning for Land-Use Classification

F. P. S. Luus, B. P. Salmon, F. van den Bergh, and B. T. J. Maharaj

Abstract—A multiscale input strategy for multiview deep learning is proposed for supervised multispectral land-use classification, and it is validated on a well-known data set. The hypothesis that simultaneous multiscale views can improve composition-based inference of classes containing size-varying objects compared to single-scale multiview is investigated. The end-to-end learning system learns a hierarchical feature representation with the aid of convolutional layers to shift the burden of feature determination from hand-engineering to a deep convolutional neural network (DCNN). This allows the classifier to obtain problem-specific features that are optimal for minimizing the multinomial logistic regression objective, as opposed to user-defined features which trade optimality for generality. A heuristic approach to the optimization of the DCNN hyperparameters is used, based on empirical performance evidence. It is shown that a single DCNN can be trained simultaneously with multiscale views to improve prediction accuracy over multiple single-scale views. Competitive performance is achieved for the UC Merced data set, where the 93.48% accuracy of multiview deep learning outperforms the 85.37% accuracy of SIFT-based methods and the 90.26% accuracy of unsupervised feature learning.

Index Terms—Feature extraction, neural network applications, neural network architecture, remote sensing, urban areas.

I. INTRODUCTION

FEATURE design has been a mainstay in classifier applications, and much effort has been invested in hand-engineering specific features that are suitable only for select use-cases. The advent of GPU-accelerated computational resources made feasible the implementation of multilayer convolutional neural network (CNN) approaches for classification. Deep learning discovers optimal features for the given problem in order to minimize the log loss cost function during classification. It is important to investigate the performance benefits of using the optimal problem-specific features learned by deep learning instead of using user-defined features that trade problem-specific optimality for general applicability.

The features discovered by deep learning are optimal in the sense that they minimize the multinomial logistic regression

objective, and improved accuracy is expected compared to the use of more general user-defined features like SIFT and Gabor features. The objective of this research was to design a deep CNN (DCNN) for the UC Merced land-use data set [1], a data set compiled in 2010 and used as a benchmark in several land-use classification studies [1]–[7]. The challenge is to optimize classification accuracy by finding a proper selection of DCNN hyperparameters, which are defined as the DCNN settings, such as the architecture design, convolutional filter bank specifications, pooling layer specifications, and learning rate and momentum values, excluding the learned neuron weights and biases.

While the hyperparameter selection and the reduction of overfitting through data augmentation do have a significant impact on deep learning performance, an additional strategy is needed to achieve competitive performance. This requires moving beyond simple label-preserving transformations such as mirroring and rotation to augment the input data set while still adhering to the guiding principle of minimum intervention so that the majority of the feature learning burden can be delegated to the deep learning solution.

The approach contributed in this letter is a generalization of the multiview strategy used by Krizhevsky *et al.* [8] to admit multiple view scales used to extract partial input sample patches. Classes with size-varying objects, such as storage tanks, can then potentially be recognized more accurately if consensus of multiscale views is used, a hypothesis tested in this research.

Deep learning has been used previously in remote sensing for hierarchically extracting deep features with deep belief networks [9] or stacked autoencoders in combination with principal component analysis and logistic regression for hyperspectral data classification [10]. A hybrid DCNN was presented by Chen *et al.* [11] for improved vehicle detection in satellite images where variable-scale features are extracted through the use of multiple blocks of variable receptive field sizes or max-pool field sizes. Remote sensing image fusion with deep neural networks (DNN) has been done by Huang *et al.* [12] using stacked modified sparse denoising autoencoders for pretraining the hidden layers of the DNN to avoid the “diffusion of gradients” caused by random neuron initialization.

An overview of the UC Merced data set is given in Section II, and the methodology and design approach are discussed in Section III. The benchmark setup and description of the empirical investigation of different deep learning architectures are then given in Section IV. The results are also presented in Section IV, where class confusion, convergence, visualization of the inner workings of the network, and comparison to the results of other published methods are addressed before a conclusion is reached in Section V.

Manuscript received March 24, 2015; revised August 17, 2015; accepted September 21, 2015. Date of publication October 26, 2015; date of current version November 11, 2015. This work was supported by the National Research Foundation of South Africa.

F. P. S. Luus and B. T. J. Maharaj are with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa (e-mail: luus@ieee.org).

B. P. Salmon is with the School of Engineering and ICT, University of Tasmania, Hobart, Tas 7001, Australia.

F. van den Bergh is with the Remote Sensing Research Unit, Meraka Institute, Council for Scientific and Industrial Research, Pretoria 0001, South Africa.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/LGRS.2015.2483680

II. DATA SET

The UC Merced land-use data set [1] is investigated, which is a set of aerial orthoimagery with a 0.3048-m pixel resolution extracted from United States Geological Survey national maps. The UC Merced data set has been used as a benchmark for land-use classifier evaluation in numerous publications [1]–[7].

The data set consists of 21 land-use classes containing a variety of spatial patterns, some with texture and/or color homogeneity and others with heterogeneous presentation. The data set was compiled from a manual selection of 100 images per class, each RGB image being approximately 256×256 pixels. The 21 land-use types include 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 court classes.

III. METHODOLOGY

In this section, an overview of the training of a DCNN is first given, followed by a description of the important processing layers of a DCNN. The specific DCNN architecture instantiation developed for the UC Merced data set is then defined, and then methods of reducing training overfitting are given. A multiscale multiview input strategy is then described, which utilizes the defined DCNN.

A. Deep Learning

Deep learning is characterized as an end-to-end learning system typically consisting of more than five processing layers, which is usually supervised and produces a discriminative classification for a given input. The burden of feature determination is shifted to a DCNN, which learns the optimal features for the given problem in order to minimize a loss cost function. The features are learned in a hierarchical manner where higher level features are learned in deeper convolutional layers as combinations of lower level features determined in shallow layers.

An improved accuracy is expected by directly learning the features that minimize the multiclass log loss cost function $L = -(1/N) \sum_{i=1}^N \sum_{j=1}^K y_{i,j} \log(p_{i,j})$ for a given data set with N samples, compared to using predetermined features. The natural logarithm of the probability $p_{i,j}$ of sample i belonging to class j is counted by setting $y_{i,j} = 1$ only if i belongs to class j . Stochastic gradient descent can be used since the loss function is a sum of differentiable functions, and Nesterov's accelerated gradient, in particular, has been shown to be effective despite the use of noisy gradient estimates [13]. The update increment v_{t+1} and the updated network parameters w_{t+1} are calculated as follows, with momentum μ and learning rate ϵ :

$$v_{t+1} = \mu \cdot v_t - \epsilon \cdot \left\langle \frac{\partial L}{\partial w} \Big|_{w_i + \mu \cdot v_t} \right\rangle_{B_i} \quad (1)$$

$$w_{i+1} = w_i + v_{t+1}. \quad (2)$$

The loss gradient estimate $\partial L / \partial w$ is determined for the average loss over a smaller batch B_i of input samples for the DCNN parameters equal to $w_i + \mu \cdot v_t$.

B. Architecture Definitions

1) *Convolutional Layers*: A CNN consists of convolutional layers, each followed by optional subsampling and regularization layers, and ending in fully connected 1-D hidden layers. A convolutional layer receives a 3-D input and creates a 3-D output that measures the filter responses at each input location, calculated as the sum of the elementwise incidence product between the filter and image window. This convolutional response encodes the input in terms of learned templates to systematically reduce input dimensionality as a part of feature determination.

2) *Activation Functions*: Each filter response becomes the input to a nonlinear activation function, which should be non-saturating in order to accelerate learning. Rectified linear units (ReLU) ($f(x) = \max(0, x)$) are used in lieu of saturating nonlinearities after every convolutional and fully connected layer, except for the final dense layer which uses softmax activation ($f(x_j) = e^{x_j} / \sum_k e^{x_k}$) to maximize the multinomial logistic regression objective. Network implementation is simplified with the use of ReLU, as this activation function does not require input normalization to avoid saturation, although local normalization can promote improved generalization [8].

3) *Subsampling Layers*: Subsampling layers normally proceed convolutional layers to further reduce feature dimensionality but also to achieve translation invariance in the case of max-pool subsampling layers [8], e.g., a 2×2 max-pool layer divides the convolutional layer output into a set of nonoverlapping 2×2 cells and only records the maximum activated filter response in each cell, thereby halving the input dimensions and producing features that are increasingly invariant to image object translations.

C. Architecture Instantiation

The DCNN design given in this section was heuristically selected based on experimental investigation that adhered to the objective of layer dimension reduction, since it develops a strong hierarchical feature representation. The DCNN designed for the UC Merced data set accepts a $96 \times 96 \times 3$ input, which can be converted from an RGB to HSV (hue-saturation-value) color model. The HSV color model can more directly concentrate chromaticity to single filter layers, which can potentially simplify features and allow for the reduction of network complexity.

The input is converted to $45 \times 45 \times 64$ neurons with the first convolutional layer using 64 filters of $7 \times 7 \times 3$ operating at a stride of (2,2), before being subsampled with a 2×2 max-pool layer to obtain a $23 \times 23 \times 64$ output with 10% dropout. The second convolutional layer uses 192 filters of $3 \times 3 \times 64$ to produce a $21 \times 21 \times 192$ output, which is subsampled with a 2×2 max-pool to give an $11 \times 11 \times 192$ output with 20% dropout as shown in Fig. 1. A third convolutional layer with 192 filters of $3 \times 3 \times 192$ produces a $9 \times 9 \times 192$ output followed by a 2×2 max-pool layer which outputs $5 \times 5 \times 192$ neurons with 30% dropout. The final convolutional layer has 224 filters of $2 \times 2 \times 192$ and gives a $4 \times 4 \times 224$ output, which is max-pooled with 2×2 cells to render a $2 \times 2 \times 224$ output with 40% dropout. A fully connected dense layer with 256 hidden units is used with ReLU activation, and 50% dropout follows, after which another dense layer with 256 hidden units is used before resolving to 21 units in a softmax output layer.

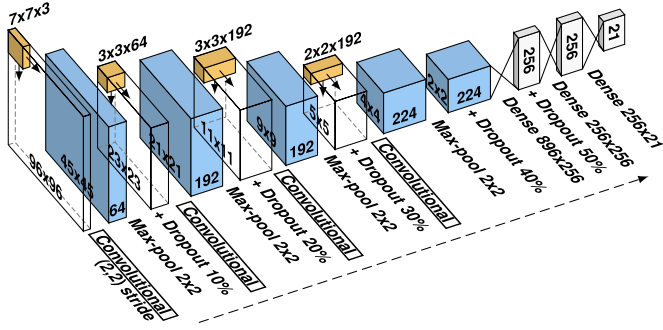


Fig. 1. CNN architecture with four convolutional layers accepting $96 \times 96 \times 3$ inputs and resolving to a 21-class softmax output layer.

All neuron biases are set to 0, and network weights are initialized randomly according to normalized initialization $U[-(\sqrt{6}/\sqrt{n_j + n_{j+1}}), (\sqrt{6}/\sqrt{n_j + n_{j+1}})]$ given by Glorot *et al.* [14], where n_j and n_{j+1} are the number of neurons in layers j and $j + 1$, respectively. The final DCNN weight and bias parameters are based on the epoch registering the minimum value for the log loss cost function on the training data.

D. Reducing Overfitting

1) *Dropout*: Convolutional and fully connected layers can be interconnected so that hidden neuron outputs are deactivated with probability p during training, with the remainder of the outputs multiplied by $1/(1 - p)$. This strategy reduces the coadaptation of neurons, since dropout forces neurons to provide more useful and robust contributions in combination with arbitrary active neuron combinations [8]. The set of dropped neurons changes randomly at every epoch, which changes the architecture and reduces overfitting at the cost of approximately $1/(1 - p)$ times the convergence period compared to training without dropout.

2) *Data Augmentation*: The original input data set can be expanded with label-preserving transformations such as horizontal and vertical flips and rotation. This presents the network with an enlarged set of inputs which may contain examples present in the test data set but not in the original training data set, thus improving classification accuracy. During training, all views are flipped horizontally or vertically with a probability of 0.5, but for testing, the model averaging only considers the untransformed views. The classifier is trained with transformed views so that any untransformed view can be recognized during testing.

E. Multiview Deep Learning

Another form of data augmentation involves the use of multiple partial views of a given input sample to train with and classifying test samples with the mean softmax output averaged over a predetermined set of classified patches or views, i.e., model averaging [8]. Some classes are distinguished by the presence of certain objects, such as airplanes and storage tanks, which only occupy a portion of a given sample. If these objects vary in size across different samples, then multiscale views can potentially produce stronger activations with higher probability than single-scale views.

The main contribution proposed is that a single DCNN can be trained with multiscale views to obtain improved classification accuracy compared to using multiple views at one particular

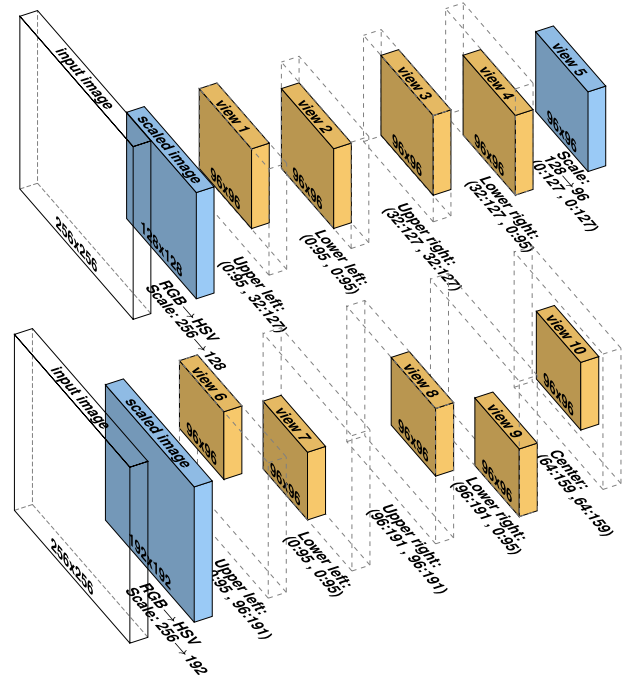


Fig. 2. Partial view selection specifications for composing a multiview input data set consisting of $10 \times 96 \times 96 \times 3$ inputs per sample.

TABLE I
FIVEFOLD CROSS-VALIDATION ACCURACY FOR VARIOUS DCNN ARCHITECTURES. ALL INSTANTIATIONS USE NESTEROV'S ACCELERATED GRADIENT (LINEAR MOMENTUM $\mu = 0.9 \rightarrow 0.999$, LINEAR LEARNING RATE DECREASES TO 0.0001, BATCH SIZES $|B_i| = 128$)

Parameter	Architecture						
	#1	#2	#3	#4	#5	#6	#7
Input size	80×80			96×96			128×128
Filter 1 size	7 × 7	7 × 7	7 × 7	7 × 7	7 × 7	9 × 9	9 × 9
Learning rate	0.005	0.005	0.005	0.005	0.005	0.005	0.01
Max epochs	1000	1000	300	300	300	1000	1000
Multiview	1	1	5	5	10	1	1
Multiscale	×	×	×	✓	✓	×	×
HSV: Acc. (μ)	86.76	88.00	90.53	91.18	92.34	87.10	83.29
$\pm\sigma$)	± 1.74	± 2.88	± 1.87	± 1.62	± 1.25	± 1.98	± 2.83
RGB: Acc. (μ)		87.14	91.10	92.76	93.48		
$\pm\sigma$)		± 3.77	± 0.80	± 1.46	± 0.82		

scale only. The UC Merced data set samples are downsampled from 256×256 to 96×96 based on empirical evaluation of the optimal input size, and ten multiscale views are extracted as follows. The first four augmenting views are acquired at the image corners at 75% input coverage, while the fifth view has 100% coverage. Views 6 to 10 are obtained at the corners and center at 50% input coverage, and all extracted views are scaled to the input size of 96×96 as shown in Fig. 2.

IV. RESULTS AND DISCUSSION

A. Experimental Setup

The standard benchmark conditions for the UC Merced data set first stipulated in [1] are followed to measure classification accuracy. Fivefold stratified cross-validation is used for all experiments, where four folds are used for training and model selection and the remaining unseen fold is classified to measure accuracy. Initial empirical evaluation indicated that the hyperparameters that most influence accuracy include the input

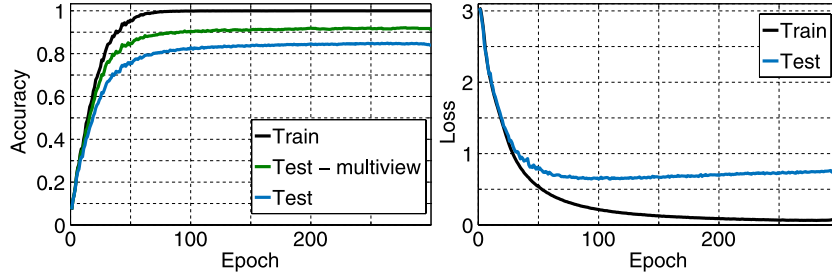


Fig. 3. Averaged fivefold cross-validation accuracy graphs for multiview architecture #5 (see Table I).

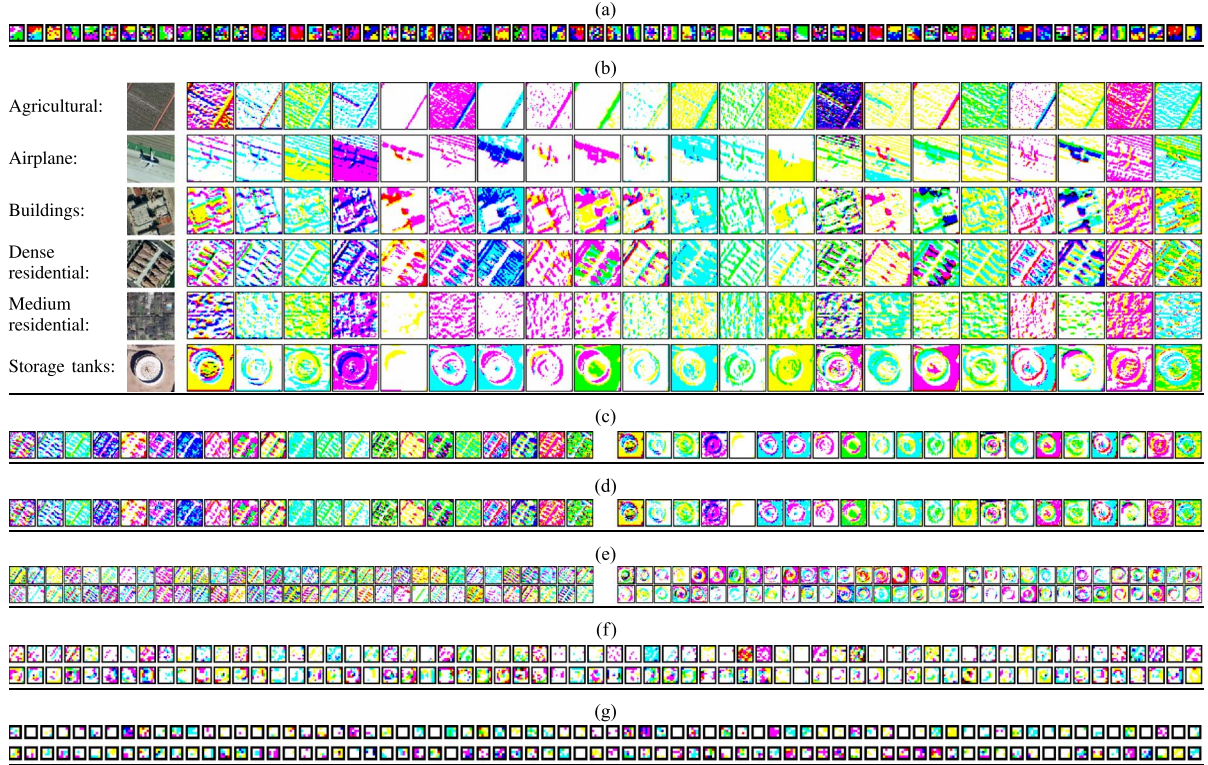


Fig. 4. Filters and CNN layer outputs for single-view architecture #2 (see Table I) and inputs from a selection of classes. Output visuals are mapped to full channel range and combined in some cases to occupy all RGB channels. (a) Filters - Convolution 1: Trained $7 \times 7 \times 3$ convolutional filters (64 filters). (b) Convolution 1: $45 \times 45 \times 64$ output from $96 \times 96 \times 3$ input convoluted with $7 \times 7 \times 3$ filters and (2, 2) stride. (c) Max-pool 1: $23 \times 23 \times 64$ output from $45 \times 45 \times 64$ input max-pooled with (2, 2). Outputs shown for dense residential and storage tanks. (d) Dropout 1: $23 \times 23 \times 64$ output from $45 \times 45 \times 64$ max-pooled input with 10% dropout. Outputs shown for dense residential and storage tanks. (e) Convolution 2: $21 \times 21 \times 192$ output from $23 \times 23 \times 64$ input convoluted with $3 \times 3 \times 64$ filters. Outputs for dense residential and storage tanks. (f) Convolution 3: $9 \times 9 \times 192$ output from $11 \times 11 \times 192$ input convoluted with $3 \times 3 \times 192$ filters. Dense residential (above) and storage tanks (below). (g) Convolution 4: $4 \times 4 \times 224$ output from $5 \times 5 \times 192$ input convoluted with $2 \times 2 \times 192$ filters. Dense residential (above) and storage tanks (below).

size, the first convolutional filter size and filter amount, and the network learning rate. Hyperparameter range selections are based around values that resulted in high classification accuracy during an initial evaluation. Various architecture instantiations are evaluated empirically to optimize the aforementioned hyperparameters.

B. Architecture Selection

Several architectures have been evaluated to obtain the best performing DCNN for the UC Merced data set, and the results are shown in Table I. The important design choices include the reduction in learning rate, using model averaging with an increasing number of multiple views, and finding the optimal input size of 96×96 .

The single-scale multiview input of Krizhevsky *et al.* [8] has been implemented in architecture #3, but its 91.1% accuracy is outperformed by the 92.75% of multiscale input (architecture #4). Using the first five views (architecture #4 in Table I) specified in Fig. 2 improved test accuracy from 87.14% to 92.76%, but using model averaging with all ten views (architecture #5 in Table I) resulted in an accuracy of 93.48% for RGB inputs.

The DCNN training convergence rate is illustrated for architecture #5 (see Table I) in Fig. 3, comparing the progression of training and testing accuracies in terms of training epochs. The single-view test accuracy is also shown, which performs poorer than with multiview model averaging.

Fig. 4 displays a visualization of the trained single-view DCNN architecture #2 (see Table I), showing the first convolutional filters and the convolutional responses for a selection

TABLE II
UC MERCED ACCURACY COMPARISON

Date	Method	Accuracy (%)
2010	SPM [1]	74.00
2010	SPCK++ [1]	76.05
2015	Saliency-UFL [2]	82.72±1.18
2014	Bag-of-SIFT [3]	85.37±1.56
	Single-view deep learning	88.00±2.88
2014	SAL-LDA [5]	88.33
2015	Pyramid of spatial relations [6]	89.1
2014	UFL [3]	90.26±1.51
	Multiview deep learning	93.48±0.82
2014	VLAT [7]	94.3

of UC Merced classes. The first max-pool and dropout outputs are also shown to illustrate their functions of subsampling and omission noise. The second, third, and fourth convolutional filter banks are too large to display in this letter and are not included. The convolutional filters are the core features that are learned by the DCNN, and it is seen that the network reduces convolutional response dimensions to a final single-dimensional response appropriate for the use of softmax activation.

C. Accuracy Comparison

A fivefold stratified cross-validation comparison of all of the important methods employed in the literature for the UC Merced data set is shown in Table II. The highest accuracies for the UC Merced data set have been achieved with unsupervised feature learning (UFL) [3] and the vector of locally aggregated tensors (VLAT) method [7], which is an extension of visual dictionary approaches like bag-of-words. Single-view DCNN is outperformed by these methods, but the 90.26% accuracy of UFL can be improved upon with a multiview DCNN which achieves 93.48%.

A confusion analysis was also performed for DCNN architecture #2 (see Table I), and the most notable class confusion was between *medium density residential* and *dense residential*, as well as between *buildings* and *storage tanks*. The classes with the least accurate predictions are the *storage tanks*, *buildings*, *medium density residential*, and *tennis court* classes.

The classes that benefited most from multiscale the five-view input were the *sparse residential*, *runway*, *dense residential*, *storage tanks*, *freeway*, *river*, and *overpass* classes, while the *agricultural* class performed the worst. This gives evidence for the hypothesis that object-based classes can benefit from multiscale views if the objects tend to vary in size, such as in the *storage tanks* and *sparse residential* classes.

D. Implementation Details

For the ten-view DCNN instantiation #5 (Table I), a running time of 36.6 s per epoch was attained on an Amazon Elastic Compute Cloud g2.2xlarge instance with a GRID K 520 GPU possessing 1536 CUDA cores and 4-GB video memory of which 1 GB was used. A Python implementation was used based on Theano and Lasagne [15], which provides a GPU-accelerated computational differentiation platform which automatically computes gradients for complex systems.

V. CONCLUSION

An end-to-end learning system with hierarchical feature representation has been designed in this letter for complex land-use classification of high-resolution multispectral aerial imagery. DCNN architectures were optimized in terms of cross-validation accuracy on the UC Merced land-use data set, and it was shown that multiscale views can be used to train a single network and increase classification accuracy compared to using single-view samples. A competitive performance was shown, where multiview DCNN outperformed both SIFT-based methods and UFL. Future research may investigate the performance benefits of a combination of DCNN cascaded with secondary neural networks and the use of only one view scale per network.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their astute observations and keen advice. The opinions expressed and the conclusions arrived at are those of the authors and are not necessarily to be attributed to the National Research Foundation of South Africa.

REFERENCES

- [1] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. ACM*, 2010, pp. 270–279.
- [2] F. Zhang, B. Du, and L. Zhang, "Saliency-guided unsupervised feature learning for scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 2175–2184, Apr. 2015.
- [3] F. Hu, G.-S. Xia, Z. Wang, L. Zhang, and H. Sun, "Unsupervised feature coding on local patch manifold for satellite image scene classification," in *Proc. IEEE IGARSS*, 2014, pp. 1273–1276.
- [4] V. Jovanovic and V. Risojevic, "Evaluation of bag-of-colors descriptor for land use classification," in *Proc. 22nd TELFOR*, Nov. 2014, pp. 889–892.
- [5] Q. Zhu, Y. Zhong, and L. Zhang, "Multi-feature probability topic scene classifier for high spatial resolution remote sensing imagery," in *Proc. IEEE IGARSS*, 2014, pp. 2854–2857.
- [6] S. Chen and Y. Tian, "Pyramid of spatial relations for scene-level land use classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 1947–1957, Apr. 2015.
- [7] R. Negrel, D. Picard, and P.-H. Gosselin, "Evaluation of second-order visual features for land-use classification," in *Proc. IEEE 12th Int. Workshop CBMI*, 2014, pp. 1–5.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
- [9] Y. Chen, X. Zhao, and X. Jia, "Spectral-spatial classification of hyperspectral data based on deep belief network," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 8, no. 6, pp. 2381–2392, Jun. 2015.
- [10] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE J. Sel. Topics Appl. Earth Obs. Remote Sens.*, vol. 7, no. 6, pp. 2094–2107, Jun. 2014.
- [11] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1797–1801, Oct. 2014.
- [12] W. Huang, L. Xiao, Z. Wei, H. Liu, and S. Tang, "A new pan-sharpening method with deep neural networks," *IEEE Geosci. Remote Sens. Lett.*, vol. 12, no. 5, pp. 1037–1041, May 2015.
- [13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. 30th ICML*, 2013, pp. 1139–1147.
- [14] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
- [15] F. Bastien *et al.*, "Theano: New features and speed improvements," in *Proc. Deep Learn. Unsupervised Feature Learn. NIPS Workshop*, 2012, pp. 1–9.