Machine Learning Engineer Nanodegree Capstone Project Report

Omar Talabay
stu251087@hotmail.com

March 9, 2018

1 Definition

1.1 Project Overview

Remote sensing (RS) image classification has attracted many researchers in the field of machine learning (ML) [2], [7], [9], [15]. RS images used to be low in resolution so that simple methods were used. In [16], it is stated that such simple methods don't perform well on images with complex structures. Most of those methods rely on hand-crafted features to feed different ML algorithms. This, in fact, requires domain knowledge expertise and takes a long time to capture complex structures in RS images.

Nowadays, RS images have increased in resolution and been publicly available, i.e. via Google Earth. With the increase of RS resolution, it is possible to recognize much of what is on the Earth's surface, i.e. ships, cars, airports, etc. The advent of high resolution RS images has opened eyes on how to utilize and exploit them efficiently since old techniques used early days are not satisfactory [1]. Thus, more advanced techniques are needed and, hence, deep learning (DL) techniques have come into place. There are many applications that make use of RS images, some are, but not limited to, natural hazard detection and geo-spatial object detection [3], [11].

To the best of our knowledge, DL and transfer DL have achieved the state-of-art performance in computer vision. Therefore, the focus of this project will be on these techniques. Specifically, the report provides evaluation using 3 different DL architectures as feature extractors without fine-tuning once and with fine-tuning. Those architectures are ResNet50 [8], Xception [4], and InceptionV3 [14].

1.2 Problem Statement

This project is an attempt of dealing with RS image classification problem using DL and transfer DL techniques. Authors in [2] provide a large scale data set and applies different classification techniques. The most accurate results are convolution neural networks (CNNs) based. To be more specific, the focus of this project is on CNNs networks transfer learning and fine tuning CNNs architectures other than the ones implemented in the paper.

1.3 Metrics

In classification problems, there are many evaluation metrics. Some of them are not suitable for use in the context of this project. Log-loss is an example which is used in cases where the output is in form of probability. Some are designed for binary classification like Receiver Operating Characteristic (ROC Curve). F-1 score is a good candidate specifically for imbalance data. In balanced data set, F-1 score is similar to the overall accuracy. For more detailed correct/incorrect classification with respect to each class, confusion matrix is used. In this project, overall accuracy and confusion matrix are adopted.

To be consistent in comparing the results with the ones in [2], two evaluation metrics are adopted. The first one is the overall accuracy (OA) which is defined as the ratio of correctly classified images regardless of which class they belong to, to the total number of images as depicted in equation (1). The second is confusion matrix. It is used to analyze the errors of each class and spots where the confusion a classifier makes.

$$Accuracy = \frac{\text{The number of correctly classified images}}{\text{the total number of images}} \tag{1}$$

2 Analysis

2.1 Data Exploration

The authors in [2] collected a data set called NWPU-RESISC45 from Google Earth. This data includes 45 classes of RS images. Each class has 700 images. An image in this data set is of 256 X 256 pixels in Red-Green-Blue color space. The classes included in the data set are airplane, airport, baseball diamond, basketball court, beach, bridge, chaparral, church, circular farmland, cloud, commercial area, dense residential, desert, forest, freeway, golf course, ground track field, harbor, industrial area, intersection, island, lake, meadow, medium residential, mobile home park, mountain, overpass, palace, parking lot, railway, railway station, rectangular farmland, river, roundabout, runway, seaice, ship, snowberg, sparse residential, stadium, storage tank, tennis court, terrace, thermal power station, and wetland. Figure 1 shows a preview of the data set.

According to [2], this data set has the following characteristics:

1. Large scale: as compared to the publicly available data sets, this is the larges one exists.
2. Rich Image Variation: images were carefully selected under different conditions, i.e. under all kinds of weathers, seasons, illumination conditions, imaging conditions, and scales. Therefore, this data set provide a better representative sample for each class.
3. High Within-Class Diversity and Between-Class Similarity.

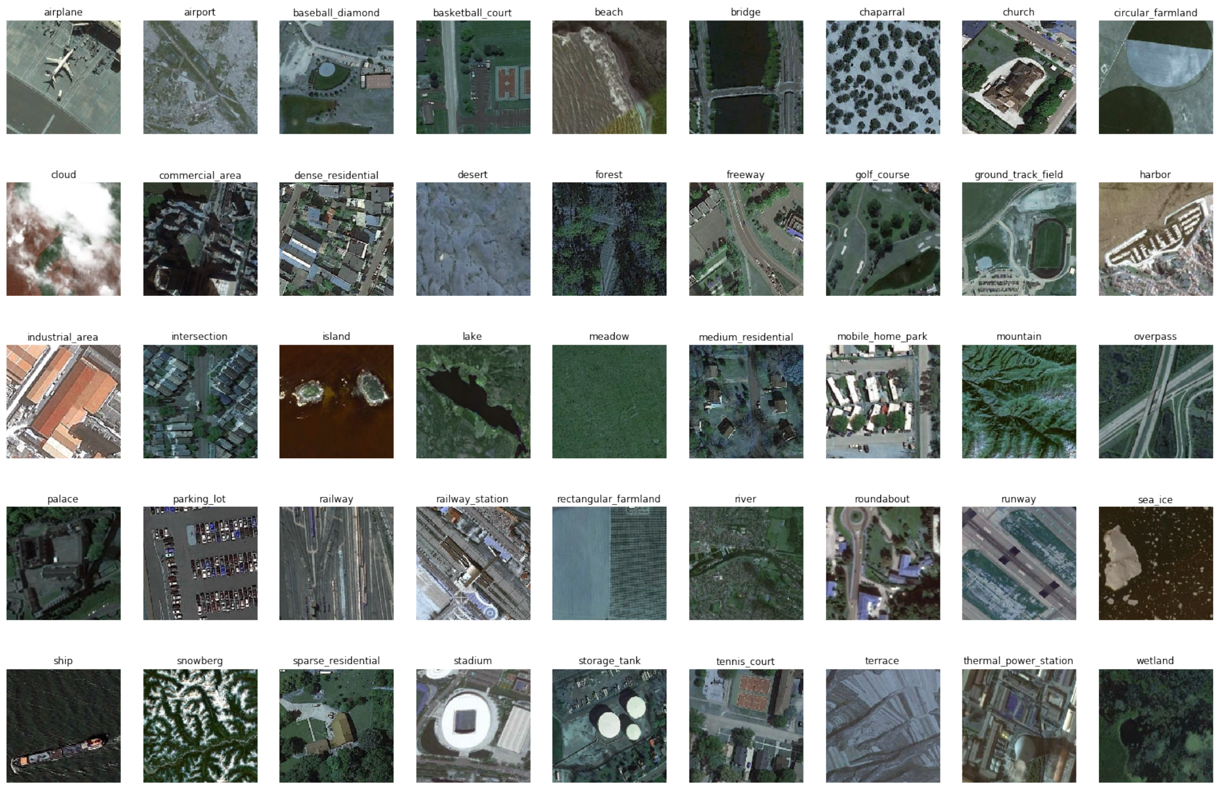
The experimental setup should follow the same pattern used in [2]. First, There are two settings for training and testing split. One is 10% for training and 90% for testing. The other is 20% for training and 80% for testing. Second, experimenting with three CNNs networks not used in the paper in two different settings. The goal of these steps is to be consistent with the experiments done and be able to compare the results.

2.2 Exploratory Visualization

Figure 2 shows classes distribution of the data set. As it can be seen, the data set is balanced with 700 images for each class. So, there is no need to use data balancing techniques in order to tackle the imbalance data.

From the figure 2, the data set looks challenging. For example, wetland and meadow share similar color characteristics. Also commercial area and dense residential classes look similar. This shows that this data set has higher inter-class similarity.

Figure 1: Some examples of NWPU-RESISC45 data set. An image from each class.



When exploring the means the images in gray scale color space, figure 3, it can be noticed that the distribution is nearly normally distributed with mean around 95.07 and 30.26 standard deviation. For quick outliers detection, it is reasonable to say that images means that 2 standard deviations away from the mean are considered outliers in terms of color intensity. Figure 4 shows the distribution of outliers. Desert, snowberg, and airplane classes are the majority of the outliers with 32%, 18%, and 11% respectively. Those classes are expected to be highly distinguishable compared to the other classes.

2.3 Algorithms and Techniques

CNN is a technique used in deep learning. CNN generally is a stack of layers, called architecture, and each layer passes its results to the next layer. Starting from the beginning, the input images, as in our case, get processed and passed to the next layer until reaching the final layer, which is the classifier. A typical CNN architecture contains the following components:

1. Convolutional layers: they are for extracting features.
2. Pooling layers: this is typically added between two convolutional layers to add some robustness to the model, i.e. image translation and rotation.
3. Dense layer: it is typically the last layer which makes classification decisions. Dropout layers may be added within dense layers to reduce the effect of over fitting.

Deep CNN used here are Xception, ResNet50, and InceptionV3 as feature extractors. This technique is called transfer learning. This concept means that a model is trained on a data set and the knowledge acquired there is applied to a different but related application. The

Figure 2: Classes distribution.

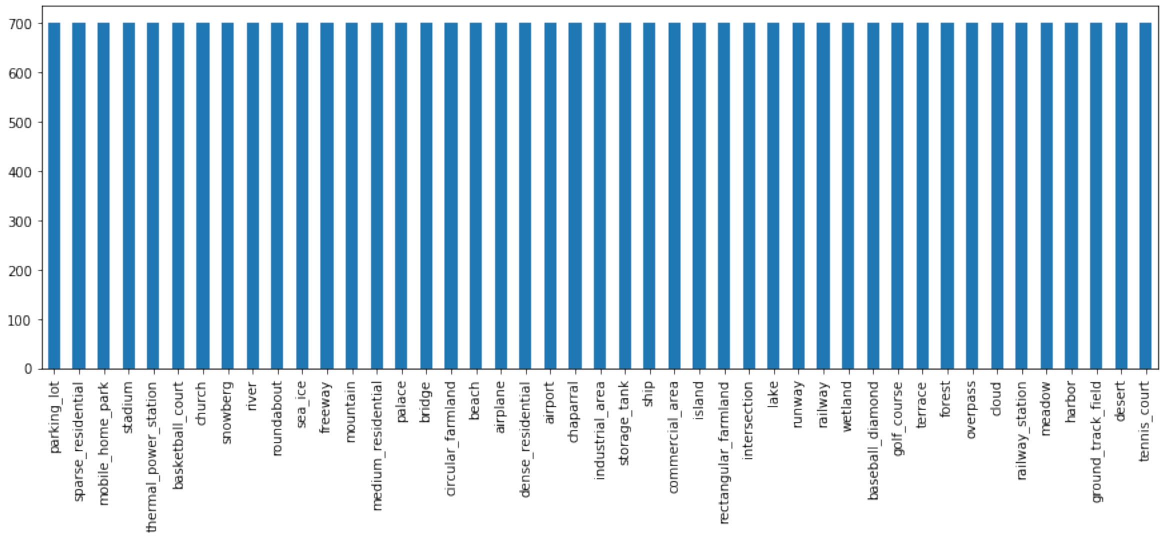


Figure 3: Images means distribution in gray scale color space.

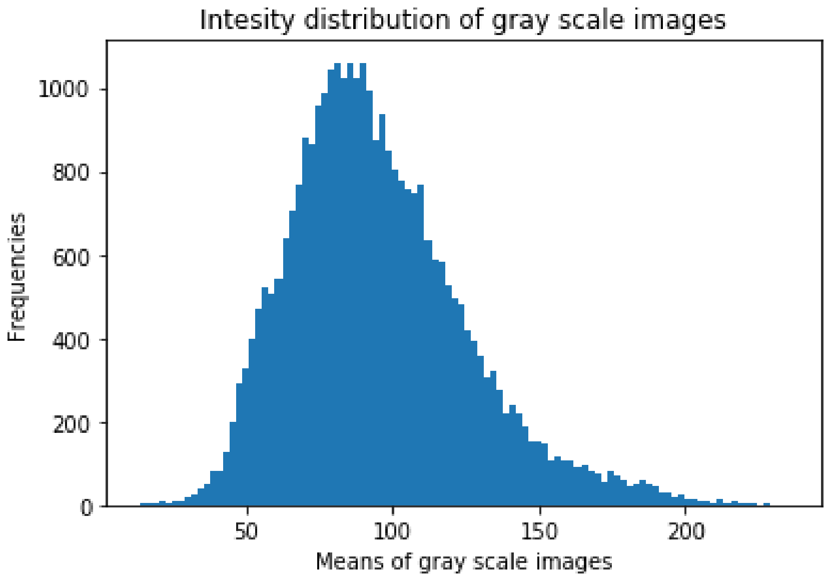
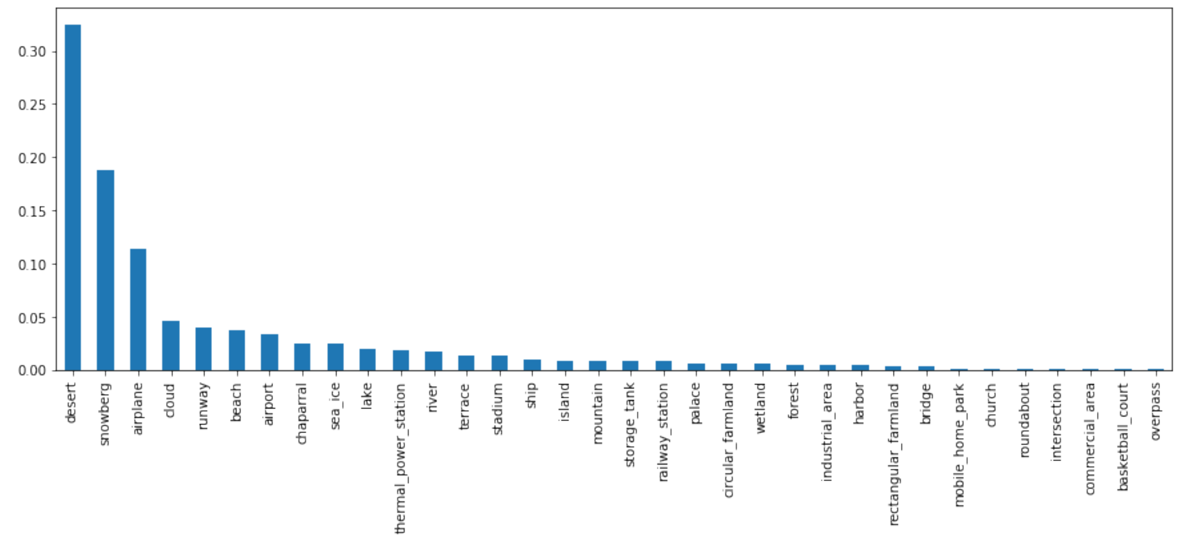


Figure 4: Images means distribution in gray scale color space.



models are chosen based on their performances on ImageNet dataset [6] and not used in the benchmark. The models used in the benchmark are less complex than the one used here specially in this large-scale data set. Therefore, it is interesting to see the effect of more complex models in RS images.Here is a brief overview for each one:

1. Xception: developed by [4] and achieved top-5 validation accuracy of 0.945 on ImageNet [6]. It was inspired by Inception and has the same number of parameters. It makes more use of model parameters, however.

2. ResNet50 [8]: The idea behind this architecture is that instead of designing deeper networks to learn more features, ResNet learns residuals. The goal is to circumvent the degradation of accuracy of complex deep networks.

3. InceptionV3 [14]: It is the third version of Inception network designed by Google. It is carefully designed deeper while keeping the computational cost constant.

Linear SVM classifier is used in all experiments with the default settings (C=1 which is a regularization parameter to control the amount of errors and the maximization of the margin) as provided by sklearn [12]. Three selected CNN architectures, (Xception, ResNet50, and InceptionV3), are used as feature extractors with two different modes. One without fine tuning and the other is with fine tuning.

Those models were pre-trained on ImageNet dataset [6]. This is a large scale data set with 3.2 million images. This makes them suitable for feature extraction since they learned to recognize complex features. In order to improve the generalization ability and enhance features quality, fine tuning these models is performed.

The idea behind fine tuning is as follow. First, a model with its pre-trained parameters are loaded and its last layer removed, and a new dense layer that is relevant, i.e 45 classes, is added. Then, the model is re-trained using stochastic gradient decent (SGD) or any other optimizer. After training, the dense layer (softmax layer) is removed and the model is used as a feature extractor.

Retraining the models have many parameters to be fine tuned. The ones experimented with in this project are:

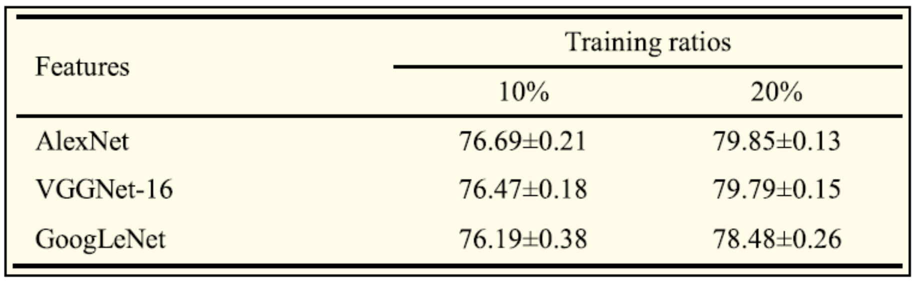
- Batch size: this denotes to the number of images taken for processing once a time. The larger this number the more accurate the estimate of the gradient is. The number used in this experiment is 32. It is relatively small compared with the benchmark settings. This is due the limit of computational power we have.
- The number of epochs: it refers to the number of iterations over the data set. For ResNet50 and InceptionV3, the number used is 3 since the models converge too quickly while 6-epoch is used for Xception. This is much lower Than what has been done in the bench mark due to the time limit constraint.
- Optimizer: SGD is used in all experiments with same settings. The parameter settings are similar to [2] and listed as follow:
  - Learning rate: it indicates how fast the learning is. In fine tuning, it should be small enough so that it doesn't distort the model's parameters learnt from ImageNet. The used learning rate in this project is 0.001.
  - Momentum: it takes the previous learning step under consideration when calculating a next step. This value is set to 0.9.
  - Weight decay: it set set to 0.0005. This will prevent weights from growing too large.

2.4 Benchmark

Keras is a high level neural network API for python. It includes many CNNs architectures with their pre-trained models. This in fact, will increase the deployment time [5]. In addition, Sklearn is a machine-learning python API. This API has many metrics for performance evaluation like overall accuracy (OA) and confusion matrix.

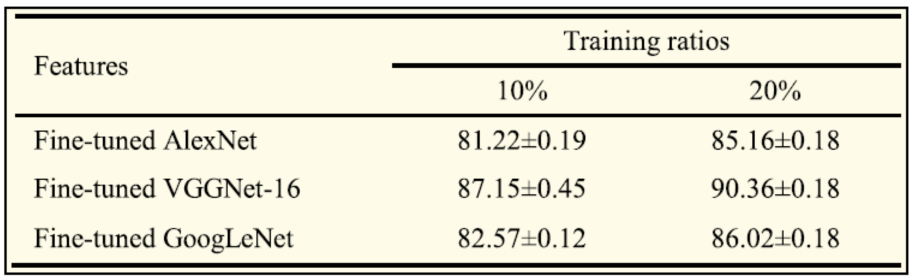
Authors in 6 provide the data set with extensive benchmark. The figures 5 and 6 show the best results achieved. So, our results will be compared against them.

Figure 5: OA under DL based CNN features with different training ratios



Features	Training ratios	
	10%	20%
AlexNet	76.69±0.21	79.85±0.13
VGGNet-16	76.47±0.18	79.79±0.15
GoogLeNet	76.19±0.38	78.48±0.26

Figure 6: OA of the three different find-tuned DL CNNs features under different training ratios



Features	Training ratios	
	10%	20%
Fine-tuned AlexNet	81.22±0.19	85.16±0.18
Fine-tuned VGGNet-16	87.15±0.45	90.36±0.18
Fine-tuned GoogLeNet	82.57±0.12	86.02±0.18

3 Methodology

3.1 Data Preprocessing

Each model performs a separate preprocessing step with a different type of scaling. This is to ensure that all feature values are standardized. Luckily, keras provides these functions for each model.

3.2 Implementation

Linear SVM classifier, with its default parameters, is used in all experiments. The difference is the use of features. The initial experiments uses the models Xception, ResNet50, and InceptionV3 as feature extractors with out fine tuning.

To extract features for a model, the model is loaded with pre-trained parameters and have its last layer (softmax layer) removed. Max pooling is used in all experiments for feature representation. The images are fed to the model and the output is the features which is then fed to SVM.

For SVM training and testing, 5-fold cross validation is used in all experiments so it gives a better assessment of the model results. There are two training-test ratios are considered for all experiments: 1)10 %-90%, 2) 20%-90%, as in [2]. The results achieved from deep CNN features are depicted in table 1 in terms of accuracy (1).

The specific steps for each model(without fine tuning) and for each training ratio are as follow:

1. Load all images with their corresponding labels.
2. Preprocess the images with its corresponding model scaling.
3. Load the desired model with its input configured to (256, 256, 3), its last layer (dense/softmax layer) removed, and its pre-trained weights loaded.
4. Replace the last layer, average pooling, with max pooling layer.
5. Extract image features by feeding them to the model.
6. For each fold in shuffled 5-fold cross validation of the extracted image features:
    a) Fit a linear SVM model, with its default parameters, on train set.
    b) Test the model on test set and compute the accuracy and confusion matrix.
    c) Add the accuracy and confusion matrix to accuracies and confusion matrices lists respectively.
7. Print the average accuracy and plot the normalized confusion matrices.

In regard to the coding process, there were not notable complications. Keras and Sklearn provide all necessary tools to get the job done.

3.3 Refinement

Even though the lowest accuracy seems to perform better than a change, the results are unsatisfactory. In order to improve the results, we adopted fine tuning technique. The idea of fine tuning is to use a pre-trained model weights as a starting point and re-train it on a new data set. The results show significant improvement in accuracy as compared to table 1. Table 2 shows the accuracies of the fine tuned features.

The steps taken to fine tune each model and for each training ratio is as follows:

Table 1: Deep learning CNN feature based results. 10% and 20% are the train size

Features	10 %	20%
Xception	71.5%	75.6%
ResNet50	32.3%	30%
InceptionV3	8.3%	10.7%

Table 2: Fine tuned Deep learning CNN feature based results. 10% and 20% are the train size

Features	10 %	20%
Xception	77.4%	85.3%
ResNet50	85.3%	89%
InceptionV3	79.4%	86.2%

1. Load all images with their corresponding labels.
2. Convert image labels to one onehot-encoded labels.
3. Preprocess the images with its corresponding model scaling.
4. Load a model with its input configured to (256, 256, 3), its last layer (dense/softmax layer) removed, and its pre-trained weights loaded. This is the base model in which each time to fine tune a model, its weights will be copied. Lets denote it by base_model.
5. Load a fully connected model with its input configured to (256, 256, 3), its dense layer included with class number of 45, and no weights loaded. This is the model to be fine tuned, lets call it fine_tuned_model.
6. Load a new model with its input configured to (256, 256, 3), its last layer (dense/softmax layer) removed, and the last average layer is replaced by max pooling. This will be used as feature extractor, lets call it feature_extractor_model.
7. Now, for each 5-fold cross validation step, train and test set:
   a) set the weights of fine_tuned_model with its corresponding layers of base_model. Randomly initialize the last layer of fine_tuned_model since this is the only layer that base_model doesn't have.
   b) After re-train ends, copy the weights of layers of fine_tuned_model to its corresponding layers to feature_extractor_model.
   c) Now, feature_extractor_model will be used as a feature extractor. In order to continue, the procedure is similar to steps 5 and 6 presented in the previous subsection 3.2. In summary, the process composes of feature extraction on train set and test set. SVM is fit and tested.
8. Print the average accuracy and plot the normalized confusion

4 Results

4.1 Model Evaluation and Validation

In this project, three models are used, Xception, ResNet50, and InceptionV3. The models are chosen based on two criteria. One is to use models not used by [2]. Second, choosing a-topthree models that are provided by keras [5].

Each experiment is performed 5 times in cross validation fashion. This in fact will increase confidence in the results. Also, the use of small train ratio, 10% and 20%, is a good indicative whether the models are able to generalize well or not. There is no data augmentation of any type since comparing the results with [2] is desired.

The results for deep CNN features are worse than the fine tuned deep CNN features. ResNet50 accuracies, for example, are 32.3% and 30% for train ratios 10% and 20% respectively while it jumps to 85.3% and 89% after fine tuning. This clearly shows that features with no fine tuning are not linearly separable. This is maybe because the domain where the model was trained on is different or the network architecture is not completely suitable. This analysis also applies to InceptionV3 which have 8.3% and 10.7% accuracies for train ratios 10% and 20% respectively.

Xception outperforms the previous models with accuracies of 71.5% and 75.6% respectively. When its features model is fine tuned, the accuracies jump to 77.4% and 85.3% with increase, approximately, of 7% and 10%. So, we can conclude that fine tuning provides significant improvement in classification accuracy. Tables 1 and 2 depict the achieved results for deep CNN features and fine tuned deep CNN features.

The parameters experimented with are batch size, the number of epochs, and the optimizer's parameters. All parameters are set the same in all experiments except the number of epochs. It is set to 3 for ResNet50 and InceptionV3 because it converges too fast. For Xception, it is set to 6 since the learning in this model is very slow. The optimizer used is stochastic gradient decent (SGD) with parameters learning rate, weight decay, and momentum set to 0.001, 0.0005, and 0.9 respectively. These parameters are briefly described in 2.3. The important point to put light on is learning rate. For fine tuning, this value should be much smaller than the one used in training the model from scratch to make sure the pre-trained weights don't get distorted.

4.2 Justification

The models used in the benchmark are AlexNet, VGGNet-16, and GoogleNet. According to [10], [5], and [13], those CNN networks achieved 80.3%, 90.1%, and 93.33% top-5 accuracies on ImageNet dataset [6], respectively. The models used in this project, (Xception, ResNet50, and InceptionV3), achieved 94.5%, 92.9%, and 94.4% top-5 accuracies, respectively, on ImageNet dataset [6]. This means that the models chosen here should extract better quality features than the benchmark's, and hence, the overall accuracies (OA) are expected to be better for this project.

Despite the assumption made earlier, benchmark models outperform the results achieved in this project in all settings. Deep CNN features results in the benchmark significantly higher than the one achieved here. This may be because the architectures used here are too deep so complex features, that are not linearly separable, are detected. In fact, the models depths used in the benchmark are much lower than the ones used here. The results can be seen at table 1 and figure 5 for complete comparison.

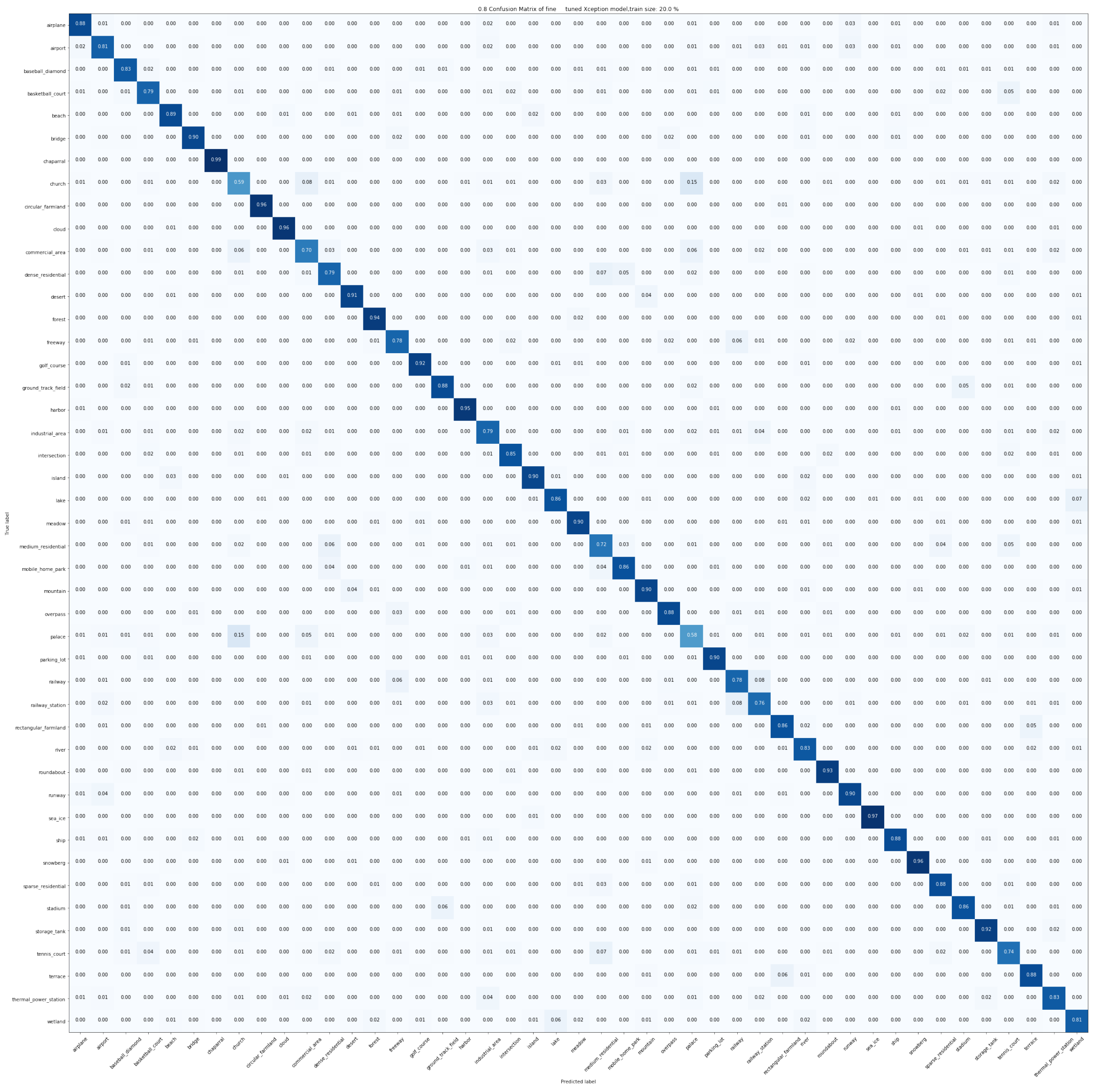
In comparing fine tuned deep CNN features, the benchmark models have slightly higher accuracies than the ones used here, as it can be seen from table 2 and figure 6. However, there are some settle differences here. First, the numbers of epochs used here are 3 and 6 while 15000 epochs used in the benchmark. Therefore, models used here probably didn't get enough iterations for fine tuning. Also, authors in [2] trained the final layer, dense layer, independently of the network and then re-trained the whole model. This also may play a significant role. These settings are not used in this project because of the time limit assigned to this course. I expect the results will be higher than the benchmark if the same settings were used.

5 Conclusion

5.1 Free-Form Visualization

On this section, the normalized confusion matrix of Xception features, figure 7, are discussed since it shows much stable results for all experiment settings. Most classes are correctly classified with at least 70% accuracy. There are two classes that are interesting to look at which have lower percentage than this. Church with 56% seems to get confused mostly with commercial area, medium residential, and palace classes. Palace class with 58% correctly classified is confused by church, commercial area, and industrial area classes. There is no wonder since if we take a look at figure 1, we find those classes share similar visual characteristics and it may even be hard for some human being to recognize.

Figure 7: Normalized confusion matrix of Xception with train set of 20%



5.2 Reflection

In this project, RS image classification is investigated using DL techniques. Specifically, two techniques were used. Pre-trained CNN models as feature extractors in which pre-processed images are fed to it are exploited. Those extracted features are then fed to a linear SVM.

The second technique is similar to the previous one except that before its used as feature extractor, it gets fine tuned in order to improve the generalization ability.

Due to the time constraint and hardware limitations, the experiments done for fine tuning are not typically similar to the benchmark. The number of epochs, for example, will take too long until it ends. For fair comparison, the same settings should be used. However, the results achieved by fine tuning are satisfactory.

5.3 Improvement

The results of the models can be improved in several ways. First, SVM can be tuned by using different kernel types and parameters. Also, the use of different classifiers techniques like boosting and bagging may be considered. In addition, fine tuning feature extractors play a significant role. In this experiment, low epoch and batch size were used due to the limit of computational power.

Different ways can be applied on data to improve accuracy. Actually, data augmentation by rotation, scaling, and adding noise shows better results on some applications. Therefore, doing so in the context of RS may improve the accuracy. Those techniques are not adopted here since we want to be consistent with the results in the benchmark.

Finally, deep feature cnn provides the state-of-the-art results on image classification. And it is actually a techniques of many. It is interesting to me to investigate those like sparse coding and auto-encoders in the near future.

References

[1] T. Blaschke. What's wrong with pixels? some recent developments interfacing remote sensing and gis. *GeoBIT/GIS*, 6:12–17, 2001.

[2] G. Cheng, J. Han, and X. Lu. Remote sensing image scene classification: benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[3] G. Cheng, J. Han, P. Zhou, and L. Guo. Multi-class geospatial object detection and geographic image classification based on collection of part detectors. *ISPRS Journal of Photogrammetry and Remote Sensing*, 98:119–132, 2014.

[4] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, 2016.

[5] F. Chollet et al. Keras. `https://github.com/keras-team/keras`, 2015.

[6] 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*, pages 248–255. IEEE, 2009.

[7] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren. Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning. *IEEE Transactions on Geoscience and Remote Sensing*, 53(6):3325–3337, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] 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*, 7(11):14680–14707, 2015.

[10] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[11] T. R. Martha, N. Kerle, C. J. van Westen, V. Jetten, and K. V. Kumar. Segment optimization and data-driven thresholding for knowledge-based landslide detection by object-based image analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 49(12):4928–4943, 2011.

[12] 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*, 12:2825–2830, 2011.

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[15] J. Wang, C. Luo, H. Huang, H. Zhao, and S. Wang. Transferring pre-trained deep cnns for remote scene classification with general features learned from linear pca network. *Remote Sensing*, 9(3):225, 2017.

[16] G.-S. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu. Aid: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3965–3981, 2017.