Structural Damage Classification for Post-Earthquake Recovery (Computer Vision)

Wen-Yi Yen[1] and Mengfan Zhang[1]

***Abstract*— Recent studies in the field of structural engineering raise eyes on the importance of post-earthquake recovery in urban areas. With the modern remote sensing technology, building and structural component images become accessible via aerial drones or related devices. In this article, the state-of-the-art deep learning technology for a civil engineering application is implemented, namely recognition of structural damage from images. In order to avoid overfitting, transfer learning is introduced and applied. Eight structural damage classification problem are tackled. Several convolutional neural network models are combined together and yield a promising recognition result.**

I. Introduction

Structural health monitoring and rapid damage assessment after natural hazards and disasters have become an important focus in civil engineering [1], [2], [3]. Meanwhile, artificial intelligence technologies are developing rapidly, especially in applications of deep learning. The big potential in deep learning application in structural engineering prompts the Kaggle Competition - "PEER Hub ImageNet Challenge" held by University of California at Berkeley [4]. Totally 8 tasks are raised:

- Task 1: Scene level identification. See Fig. 1.
- Task 2: Damage state check. See Fig. 2.
- Task 3: Spalling condition check. See Fig. 3.
- Task 4: Material type identification. See Fig. 4.
- Task 5: Collapse check. See Fig. 5.
- Task 6: Component type identification. See Fig. 6.
- Task 7: Damage level detection. See Fig. 7.
- Task 8: Damage type detection. See Fig. 8.

Fig. 1. Classification of: pixel / object / structural levels.

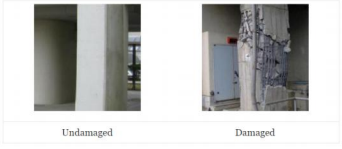

Fig. 2. Classification of: damaged / non-damaged.

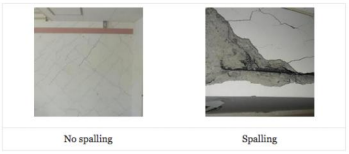

Fig. 3. Classification of: no spalling / spalling.

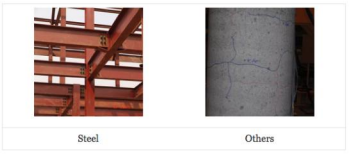

Fig. 4. Classification of: steel / others.

Fig. 5. Collapse check: no / partial collapse / collapse.

Fig. 6. Classification of: beam / column / wall / else.

Fig. 7. Damage level detection: no / minor / moderate / heavy damage.

Fig. 8. Classification of: no / flexural / shear / combined damage.

[1]W.-Y. Yen - Student in CEE, SUNetID: wyen
[1]M. Zhang - Student in CCRMA, SUNetID: zhangmf

Convolutional neural network (CNN) has been at the heart of spectacular recent advances in deep learning. Compared with traditional computer vision and machine learning approaches, CNN no longer needs hand-designed low-level features or so-called feature engineering where the millions of parameters inside a typical network are capable of learning amounts of mid- to high-level image representations with input data obtained from a pixel matrix (tensor). Another unique characteristic of deep CNN is its depth of architecture. Many well-designed CNN architectures, such as VGGNet [5], GoogleNet [6] and Deep Residual Net [7] (just to name some) demonstrated the great performance improvement with substantially increasing the depth.

In this article, the pretrained ImageNet models and transfer learning are employed in our problems. The pretrained models have powerful generalization abilities for it can performs well in different data sets. The major advantage of transfer learning is that it can relax the requirement that training a deep CNN needs a large number of data, through tuning part of the parameters from pretrained model in source domain with few labeled data in target domain, which might lead to a good performance for the target data set. In transfer learning, all parameters in the net before the fully connected layers are frozen, features are extracted from the last layer, then the fully connected layers are added to train the extracted features. After that, parts of the networks are frozen and the remaining parameters are retrained with gradient descent and back propagation.

II. Related work

In recent studies by Gao [1], some similar tasks such as damage classifications and component recognitions are presented. The main method of tackling the probelms in the paper is transfer learning using the pretrained VGG16 model. Initially, we approach the competition tasks following the outlines of the paper. However, our work diverges with it in the final phase, which includes the combination of several ImageNet models and the data augmentation using the fancy principal component analysis (fancy PCA).

III. Dataset and features

The dataset presented in this paper is directly provided by the PEER PHI Competition. The data amount and training / validation split is shown in Table I. Each task has different challenges, such as highly imbalanced data is a bug issue to overcome. In some cases that focus on concrete cracks and spalling, the implementation of fancy PCA could help increase accuracy.

TABLE I

The count of images in each tasks

	class 0		class 1		class 2		class 3	
	Train	Validation	Train	Validation	Train	Validation	Train	Validation
Task 1	5370	500	5210	500	5330	500	-	-
Task 2	2680	500	2220	500	-	-	-	-
Task 3	1500	100	1600	90	-	-	-	-
Task 4	1200	310	2500	320	-	-	-	-
Task 5	200	40	180	40	36	16	-	-
Task 6	200	40	750	50	1280	40	200	50
Task 7	1450	50	280	40	420	40	300	40
Task 8	1450	50	150	50	350	50	470	50

IV. Methods

A. Initial Phase - Transfer Learning Using VGG16

In our tasks, only few data is presented. Transfer learning is a very effective tool to address the lack of data. In the CNN, parameters in shallower layers represent low-level features, such as color, texture and edges, while parameters in deeper layers attempt to capture more complicated and abstract high-level features. Therefore, the major objective of transfer learning in CNN is to make use of parameters in a well-trained model from the data set in source domain to help with training the data set in the target domain. The original training data set for a pretrained VGG-16 Model is ImageNet, which includes thousands of images related to buildings, bridges, pillars, walls, etc., belonging to the civil engineering field. Therefore, for our classification tasks in the domain of structural engineering, source and target domains are assumed to be related. Moreover, from the perspective of our task objectives, the component type classification is a similar but easier task than that of the ImageNet classification problem, since it reduces the 1,000 classification to a binary one.

Configuration and procedure for normal training of the whole CNN, feature extraction, and fine-tuning are illustrated in Fig. 9. Moreover, since we focus on a small data set, while implementing fine-tuning, data augmentation tricks are used to avoid overfitting, such as zoom in/out, translation, reflection, rotation, and color changes. To implement the

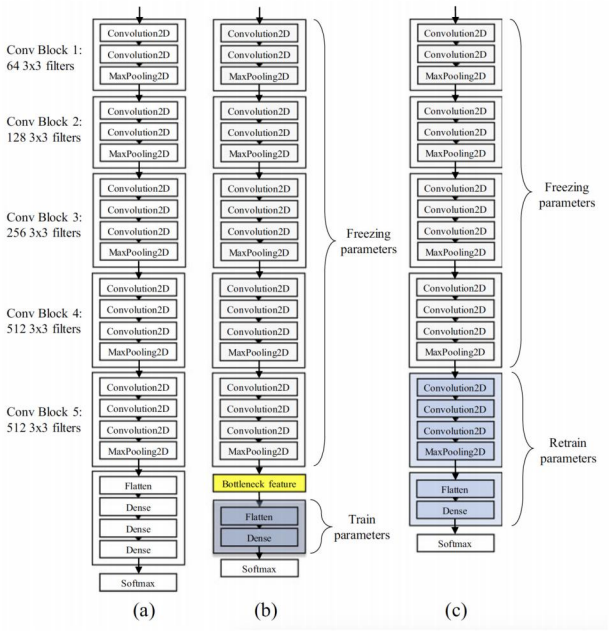

Fig. 9. Configuration and procedure of transfer learning.

transfer learning in deep CNN, we first extract the bottleneck feature for CNN. Convolution operations are only performed once in the feed forward procedure. The outputs of the last layer are taken as the bottleneck feature of the training data. Then the fully connected layers are added to fix the classification problem. Extracting the bottleneck features greatly decreases the training time and number of epochs, because it only trains the bottleneck features using a shallow fully connected neural network, instead of training from top to bottom through large number of conv layers for multiple times. After that, we use fine-tuning to retrain some parts of the CNN. As mentioned before, conv blocks in the beginning represent low-level features, which might be similar for both source and target domains, and the objective of tuning the last several conv blocks is to adjust the mid- to high-level features to the target domain.

B. Final Phase - Combination of Imagenet models, Fancy PCA and CAM

In the final phase of the competition, our team combined several ImageNet pretrained models and average their results at the softmax layers. This is an experiment inspired by what happened in the past Netflix Competition - the rank second to last groups averaged their results and turned out to predict as good as the first-ranking group. Thinking ahead, the networks "deeper" than the VGG16 model would presumably improve the performance of classifying high-level information, such as task 5: no / partial collapse / collapse.

Fancy PCA is a scheme that performs PCA on sets of RGB pixels throughout the training set by adding multiples of principles components to the training images. It is believed to be effective in "texture-wise" classification problems. Among the tasks we worked on, task 2: damaged / non-damaged and task 3: spalling / no spalling are examples that texture changes significantly between classes. In order to visualize the effectiveness of this particular data augmentation method, the class activation maps are used. Class activation maps (CAM) are a simple technique to show which regions in the image are relevant to the class. This approach helps us understand whether fancy PCA intensifies the characteristics of the classes or misleads the classifications.

V. Experiments, Results and Discussion

A. Initial Phase - Transfer Learning Using VGG16

In this phase, our team do parameter tuning on: the number of Conv blocks to freeze, the batch sizes, the dropout rates, L2 regularization variables, data augmentations, learning rates and decaying variables, and Adam and SGD algorithm parameters. To visualize parts of the experiment, please see Fig. 10 and Fig. 11. It is found that different tasks yields very different optimal variables. In this case only task 7 is shown, but the major point is to save the model that maximizes F1-score during runtime. With that saved model, we then make predictions for initial submissions for the competition.

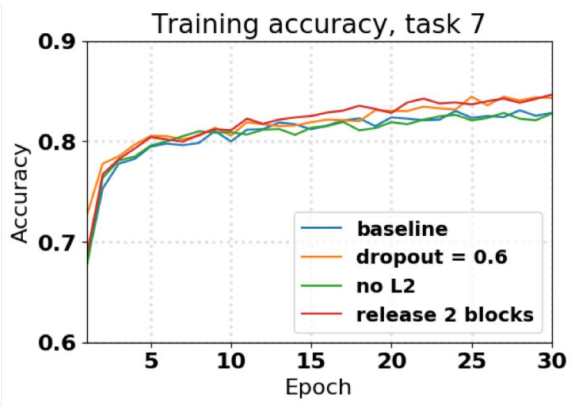

Fig. 10. An example result of hyperparameter tuning - task 7 training accuracy.

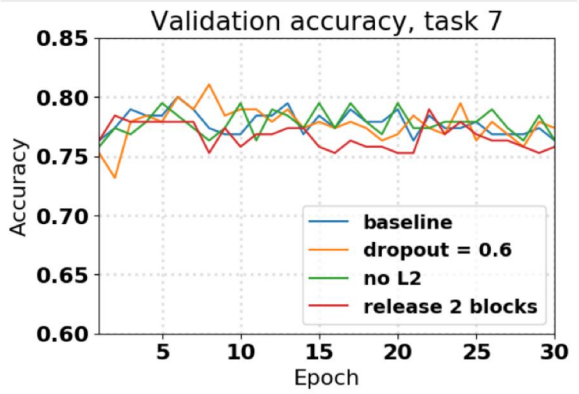

Fig. 11. An example result of hyperparameter tuning - task 7 validation accuracy.

B. Final Phase - Combination of Imagenet models, Fancy PCA and CAM

In the final phase of the competition, we experiment on 13 pretrained models in ImageNet, including: ResNet50, InceptionV3, InceptionResNetV2, Xception, VGG16, VGG19, DenseNet121, DenseNet169, DenseNet201, MobileNet, MobileNetV2, NASNetMobile, and NASNetLarge. It is found that by averaging all the results on the softmax layers, the performance can improve up to 4% as compared to our initial results experimented on VGG16. One interesting thing to note is the "DenseNets" systematically perform worse than other models. We think this is due to the fact that the architectures are deeper and harder to train on small datasets, as shown in Fig. 12. With this in mind, we excluded the DenseNets and proceed. Another interesting thing to note is - for task 5, VGG16 and VGG19 have worse performance compared to the other pretrained models. The explanation for this is VGGs are shallower among the models we implemented.

One important data issue to tackle is the highly imbalanced characteristic. For instance, the task 5 data in 3 classes has image counts: (240, 220, 52). This imbalancement caused predictions to lean over only to output the first two classes. We cope with this problem by oversampling the data in the

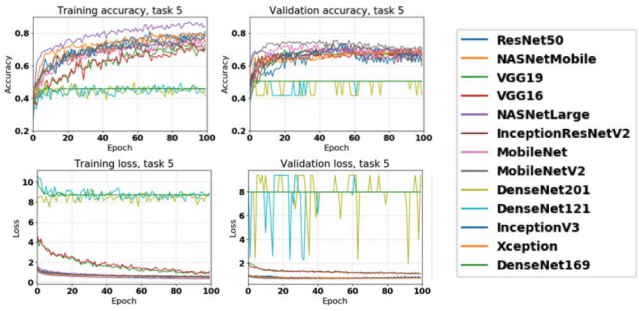

Fig. 12. Task 5 multi-model training, DenseNets are harder to train.

third class and also include more intensive data augmentations. The results show about 1% improvement in the test prediction, although the validation accuracy drops a little. This suggests that the distribution of the test images is slightly different from the data we received. The experimental results are shown in Fig. 13.

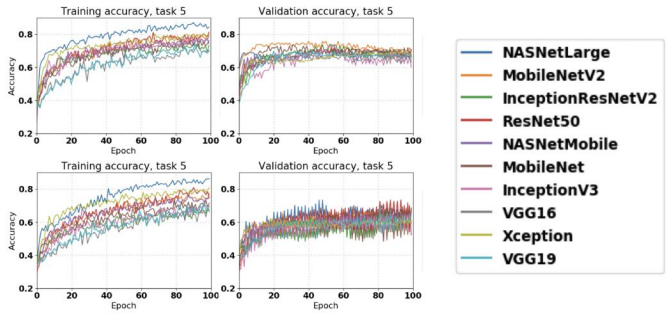

Fig. 13. Task 5 multi-model training, the top images represent the original training, and the bottom images are those with oversampled data.

The particular data augmentation approach - fancy PCA is implemented in attempt to improve further on the tasks with sufficient data. With bear eye, it is hard to tell the difference between the original images from those adjusted using fancy PCA. Thus, we implement class activation maps to visualize the improvement or shortcoming of the fancy PCA approach. It is found that "texture-wise" tasks, such as task 2: (damaged / non-damaged) and task 4: (spalling / no spalling) yield up to 1% improvement using fancy PCA. Other tasks with high-level characteristics, such as task 5: (no / partial collapse / collapse) and task 8: (no / flexural / shear / combined damage) are sometimes misleaded by the change in pixel colors. This observation is presented and described in Fig. 14.

Finally, the best predictiction of all 8 tasks are shown from Fig. 15 to Fig. 22. Overall, the predictions stands at about 15% to 20% among all the competitors.

VI. Conclusions

In a practical aspect, the authors have implemented VGG16 for transfer learning in the initial phase of the competition for 8 tasks. Then, multi-model learning is carried out in the later phase with 13 ImageNet pretrained models.

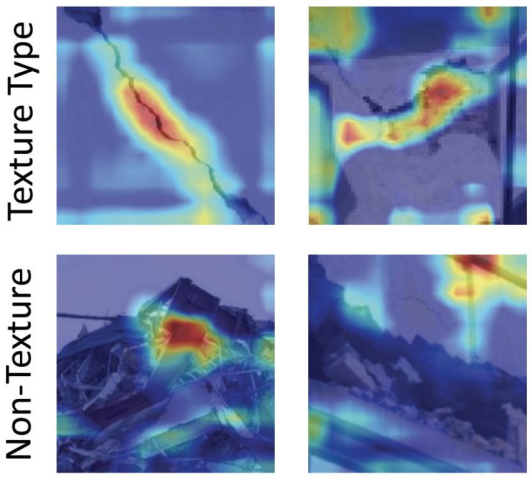

Fig. 14. Visualizing the effect of implementing fancy PCA. The top images show the approach intensify the important regions in "texture-wise" classifications; whereas the bottom two images show the misleading results in images with high-level information.

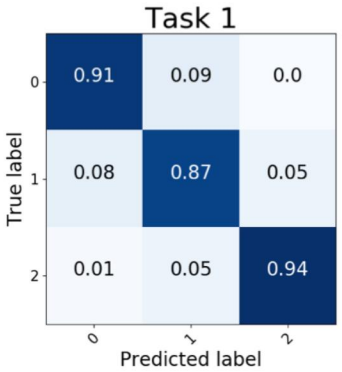

Fig. 15. The best result for task 1. Accuracy: 90.7%

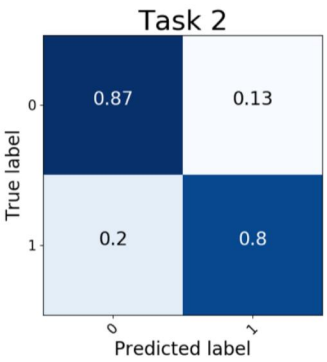

Fig. 16. The best result for task 2. Accuracy: 83.5%

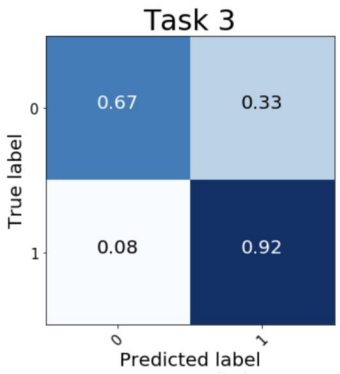

Fig. 17. The best result for task 3. Accuracy: 82.7%

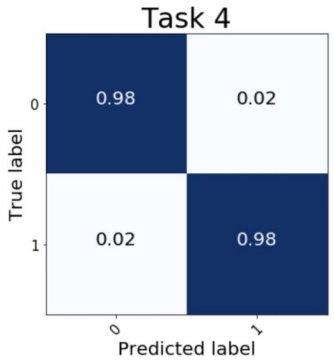

Fig. 18. The best result for task 4. Accuracy: 98.0%

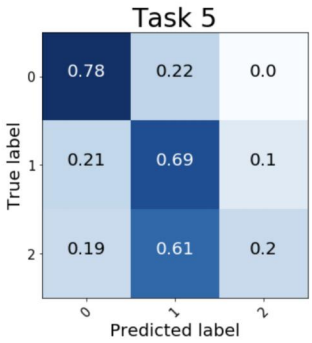

Fig. 19. The best result for task 5. Accuracy: 67.9%

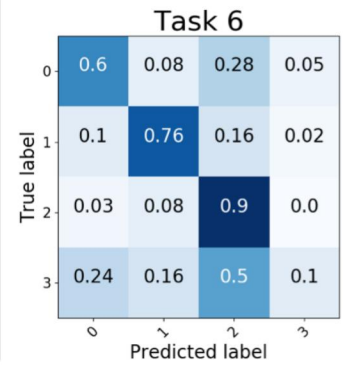

Fig. 20. The best result for task 6. Accuracy: 75.3%

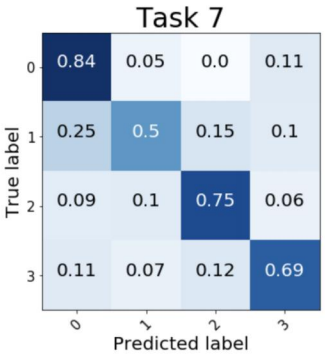

Fig. 21. The best result for task 7. Accuracy: 73.6%

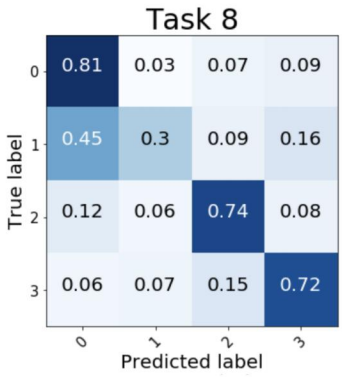

Fig. 22. The best result for task 8. Accuracy: 74.1%

The performances of the different architectures are briefly discussed. In order to cope with the imbalanced data in certain tasks, the oversampling data approach is applied and discovered some improvement. The fancy PCA approach is implemented as a data augmentation method. By visualizing with CAM, the authors found that the approach improves the texture-wise tasks, but could be misleading for the tasks with high-level information. Overall, the final predictions stands at about 15% to 20% among all the competitors.

VII. Codes

Github code link: https://github.com/wenyiyen/CS230-Final-Project.git,

Google Drive: https://drive.google.com/drive/folders/18dr2PusDLyAbr2Hs0-Cia3SSIimYVEcP?usp=sharing

VIII. Contribution

The code development and report write-up are done cooperatively. Both of the team members discussed the framework and did web searches in the early phase. Then, Wen-Yi worked on the initial code development, which is then handed to Mengfan for debugging and adding more features. The report is written based on discussions and mutual-editting.

References

[1] Gao, Y. and Mosalam, K. (2018). Deep Transfer Learning for Image-Based Structural Damage Recognition. Computer-Aided Civil and Infrastructure Engineering, 33(9), pp.748-768.

[2] Yeum, C. M., Dyke, S. J., Ramirez, J., Benes, B. (2016). Big visual data analytics for damage classification in civil engineering.
[3] Cha, Y. J., Choi, W., Bykztrk, O. (2017). Deep learningbased crack damage detection using convolutional neural networks. ComputerAided Civil and Infrastructure Engineering, 32(5), 361-378.
[4] Apps.peer.berkeley.edu. (2018). PEER Hub ImageNet Challenge — PEER Hub ImageNet Challenge. [online] Available at: https://apps.peer.berkeley.edu/phichallenge/ [Accessed 15 Oct. 2018].
[5] Simonyan, K. Zisserman, A. (2014), Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556.
[6] Szegedy, C., Liu,W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. Rabinovich, A. (2015), Going deeper with convolutions, in Proceedings of the IEEE International Conference Computer Vision Pattern Recognition (CVPR), Boston, MA, 19.
[7] He, K., Zhang, X., Ren, S. Sun, J. (2016), Deep residual learning for image recognition, in Proceedings of the IEEE International Conference on Computer Vision Pattern Recognition (CVPR), Las Vegas, NV, 77078.