CS230

A ResNet-50-based Transfer Learning and Ensemble Method for Structural Health Monitoring

Langston Nashold
Stanford University
lnashold@stanford.edu

Salil Deshpande
Stanford University
salil512@stanford.edu

Chun-Liang Wu
Stanford University
wu0818@stanford.edu

Jaewon Saw
Stanford University
jaewons@stanford.edu

Abstract

In Structural Health Monitoring (SHM), engineers and inspectors analyze images of damaged buildings, and assess whether said buildings are structurally sound. In this report, we present deep learning solutions to three visual classification tasks that are performed in SHM: Scene-Level classification, Collapse-Mode classification, and Damage-Level classification. Our algorithms are trained on data from the PEER Hub Image Net (Φ-Net), which had curated datasets for the three tasks we worked on. For all three tasks, we show how by starting with a ResNet-50 architecture and then applying transfer learning and ensembling, we are able to match, and in one case outperform, the state of the art. Furthermore, we present an analysis of sources of error in our model, and demonstrate that the most common sources of error are mislabeled data or data with ambiguous labels on the boundary between classes.

1 Introduction

After a large earthquake event, structural engineers and inspectors will visit affected areas to perform post-earthquake reconnaissance, and assess the nature and extent of the damage done to buildings, bridges, and other infrastructure. In an approach known as vision-based Structural Health Monitoring (SHM), engineers and inspectors will take images of a damaged building and use them to assess whether said building is safe for continued occupancy and operation [3].

Applications of deep learning to vision-based SHM has gained some attraction in recent years, but has yet to be fully utilized in the field. In this project, we apply deep learning to classify the type and extent of damage seen in an image, in order to ultimately assess how safe a building is for use. Specifically, we seek to answer three specific questions: (1) What part of the building is in the image, (2) What is the extent of collapse in a building, and (3) What is the extent of damage seen in a structural object.

We trained deep learning models for these three tasks, using the Peer Hub Image Net Dataset. We use a ResNet-50 architecture, using transfer learning to initalize our model. Each model takes as images of buildings as inputs, and outputs one of several classes as a label.

Code: `https://github.com/lgnashold/Structural-Image-Recognition`

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

Literature search shows that CNNs have been widely used to extract abstract features that are able to discriminate various aspects of interest in SHM. Wang et al. used AlexNet and GoogleNet to classify damage in masonry structures into four categories - intact, spall, crack, and efflorescence [10]. Feng et al. used Inception-v3 as the basis network for hydro-junction infrastructure, with the final layer's number of fully connected neurons modified to five, as applicable to the five labels of crack, seepage, intact, and rebar exposure [4]. Similarly, Liang applied VGG-16 for bridge inspection, where the last layer was modified to two, same as the number of classes considered - major failure and no failure [9]. Gao and Mosalam performed performed image classification for the same tasks considered in our project; they used VGG-16, VGG-19, and ResNet-50 architectures, and concluded that ResNet-50 produced the best performance overall [6]. Gao and Mosalam also concluded that data augmentation had limited efficacy in improving their accuracy, but transfer learning was very effective.

3 Dataset and Features

We used the PEER Hub Image Net, or Φ-Net, datasets for structural image recognition [5]. There are eight Φ-Net datasets that pertain to each of eight structural image recognition classification tasks. We worked on three of these tasks - (1) identifying what part of the building is in an image, (2) identifying the extent of collapse in an image, and (3) identifying the extent of damage seen in an image - **referred to by Φ-Net and in the rest of this paper as Task 1, Task 5, and Task 7, respectively**.

Each of our datasets contains 3-channel images that are 224x224 pixels. Each image has been preprocessed so that each color channel is centered; that is, each color channel of each image has independently had its mean pixel intensity subtracted from it. Each dataset came split into train and test sets. There are 24,309 train images and 2,997 test images in Task 1, 1,226 train images and 146 test images in Task 5 and 4,138 train images and 498 test images in Task 7.

We did not perform any additional preprocessing to the data as this image format (224x224 pixels and channel centered) is the preprocessing approach used in the original ResNet-50 paper [8]. Because the datasets for Tasks 5 and 7 were especially small, we applied data augmentation to those datasets by rotating pictures by up to 30 degrees and performing horizontal flipping. After augmentation, the size of the train datasets of Task 5 and 7 became 2,226 and 7,138, respectively. After performing this augmentation, we randomly selected 10% of the train images to be set aside as validation datasets.

For Task 1, each image can have one of three labels - Pixel (an image of building material), Object (an image of a structural component), or Structural (an image of the whole building). For Task 5, each image can have one of three labels - None (no collapse visible), Partial (partial collapse visible), or Global (the building is fully collapsed). For Task 7, each image can have one of four labels - None (no damage visible), Minor (minor damage visible), Moderate (moderate damage visible), Heavy (heavy damage visible). These labels were one-hot encoded.

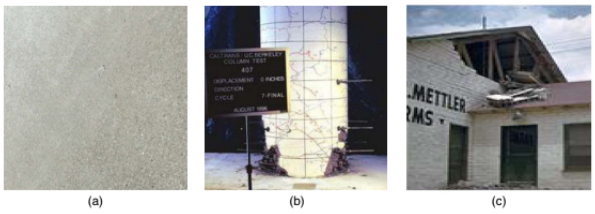

Figure 1: Pixel, Object, and Structural Level Images from PHI-Net

4 Methods

We performed each of these image classification tasks using Convolutional Neural Networks (CNNs). CNNs are the standard class of neural networks that are used in visual recognition tasks. In particular, we based our model on the ResNet-50 architecture, which uses residual blocks and skip connections to allow for larger, more sophisticated models to be built without running into the issues of over-fitting

or vanishing gradient. Our implementation was done using the widely-used deep learning framework Keras [1].

Our ResNet-50 models were pretrained on ImageNet before being fine-tuned on our dataset. Fine-tuning was done by training only the last several model layers on our dataset. Transfer learning works well because the earlier layers of a CNN typically learn visual structures common to all images, such as edges, shapes, and colors. By using this transfer learning approach, we were able to successfully train a sophisticated models despite having small datasets.

Our final solution for each of the three tasks was to build an ensemble of ResNets to make predictions. Ensembles are a composite of multiple neural networks, and enjoy better generalization and stability than single network architectures [7]. We built each ensemble by stacking the five best ResNets (among tens that we tested during hyperparameter search) for each task. The output of the five layers are concatenated before being passed through a dense layer of 10 neurons and a final softmax output layer. We trained these last two layers to learn how to best combine the predictions of the five constituent models.

Training was done using the categorical cross-entropy loss function. The equations below show the loss and cost functions, respectively, that we used. Note that C refers to the total number of classes we performed classification over.

$$\mathcal{L}(y^{(i)}, \hat{y}^{(i)}) = -\sum_{j}^{C} y^{(i)} log(\hat{y}^{(i)}) \quad L = \sum_{i=1}^{m} \mathcal{L}(y^{(i)}, \hat{y}^{(i)})$$

We used the Adam optimization algorithm during training. Adam combines gradient descent with momentum, which uses an exponentially weighted moving average of gradients to compute an update, and RMS Prop, which normalizes the update size taken based on past gradients. Since we had balanced classes across all our tasks, accuracy served as a good metric for success in classification. We also recorded the AUC score as an alternative measure of our performance.

5 Experiments and Results

5.1 Models Tested

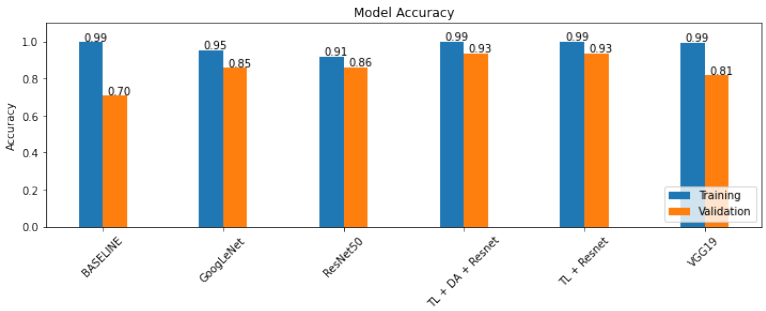

Figure 2: Accuracy of Models Tested on Task One

We began our work by comparing five standard CNN architectures in terms of performance in order to identify an architecture that we could build off of. We looked at a baseline model (2 convolutional layers, 1 dense layer, and 1 output layer), VGG-19, GoogLeNet, ResNet-50, and a ResNet-50 pretrained on ImageNet [2]. Each model was trained and tested on Task 1 data, and the same hyperparameters were used in all of them. Figure 2 above shows the results. The pretrained ResNet-50 had the best validation accuracy, as well as the smallest gap between train and validation accuracy. The small gap between train and validation accuracy reflects that the ResNet did not overfit, which we is due to the skip connections built into the ResNet architecture. From this experiment, we selected the ResNet-50 with transfer learning as our baseline model across all three tasks, and moved on to hyperparmaeter tuning.

5.2 Hyperparameter Tuning

Due to computational constraints, we only tuned our models over three parameters - the learning rate, the learning rate decay, and number of frozen layers in our transfer learning model. We tuned these using random search, sampling the learning rate over $[10^{-4}, 10^{-1}]$, learning rate decay over $[0.7, 1]$ (higher value meaning less decay), and the number of frozen layers over $[115, 176]$. After receiving preliminary results, we reduced the search range. The best results are listed in a table in Appendix 1. For all three tasks, we found that the combination of a learning rate in $[10^{-3}, 5 \times 10^{-3}]$, a decay rate of $[0.75, 0.85]$, and a choice of frozen layers between $[130, 150]$ worked best, with some slight variation based on task.

5.3 Ensemble Model

The result of each ensemble model for each task is shown as Table 1. The validation accuracy of the three tasks slightly compared to the validation accuracy of the 5 best models in each task (Table 2). The result shows that the ensemble model can produce the better predictive performance than any single ResNet-50.

Table 1: Performance of Ensemble Model for each task

Task	Val Accuracy	Test AUC	Test Accuracy	Test Accuracy from [6]
Task 1	0.9412	0.9671	0.9343	0.934
Task 5	0.8341	0.8674	0.7603	0.781
Task 7	0.8235	0.8881	0.7430	0.745

5.4 Error Analysis and Relabeling

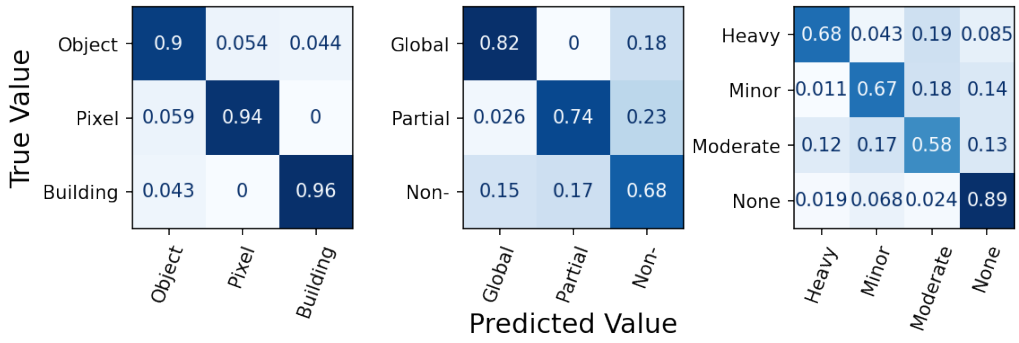

Figure 3: Confusion Matrices for Three Tasks

The confusion matrix for task one shows that most mistakes were essentially "one class adjacent", e.g., buildings might be misclassified as objects, but never as pixel-level. For task 7, this pattern also held: most confusion was between damage types, and most often between directly adjacent classes. It was also easier to recognize nondamaged buildings than any other class. In task 5, most mislabelings happened between whether there was collapse or not. Interestingly, global and partial collapse were almost never confused.

We then looked at the images our model got wrong. Anecdotally, we found that a large number of the images we got incorrect were mislabeled. Therefore, we decided to quantitatively analyze how many images were mislabeled for each dataset. For Task 1, roughly two thirds of the images we got wrong in our validation set were mislabeled. After relabeling our validation set, our accuracy improved from 93.2% to 95.5%. For Task 5, 40% of the images we got wrong in our validation set were also mislabeled. The accuracy of validation set can be improved from 83.4% to 90.3% if the labels are correct. For Task 7, 25% of the images we got wrong in our validation set were also mislabeled. The accuracy of validation set can be improved from 83.4% to 87.6% if the labels are correct. This suggests one promising avenue for better performance is by changing the dataset, rather than improving the model.

6 Discussion

With the ensemble models, we were able to slightly pass the state of the art for Task 1 , and come close to the state of the art for Tasks 5 and 7, as shown in Table 1. Although we tested more architectures than prior works, we still we found that ResNet-50 and Transfer Learning produced the best results.

We also found that the models that worked the best on each task were largely similar, with similar optimal hyperparameters. This suggests that a common strategy for training models on a dataset with multiple tasks would be to find one model that works well, and then fine tune it on each task.

Furthermore, the error analysis conducted provides insight into the dataset: the fundamental problem is that the boundaries between classes are not clear-cut. This occurred in all the tasks; each task classifies the "degree" or "extent" of some quality: image scale, building collapse, or object damage. Although some images our model got wrong were blatantly mislabeled, many more seemed to be on the edge between two categories and were hard for even humans to classify. You can see an example of this in Figure 4: it is difficult to determine whether this should be classified as a building or an object. This also matches the results from our confusion matrices: images were misidentified as the classes nearest to them.

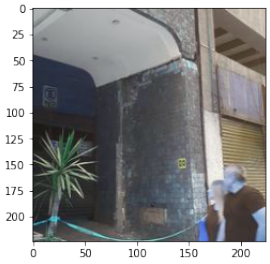

Figure 4: Borderline Image from Task One

Fundamentally, when we map these sliding scales to arbitrary buckets, it's hard for even a human to make a decision when the image is on the boundary. This suggests that human-level error on these tasks might actually be relatively high. Since human error can be a proxy for Bayes Error, this could mean that the current state-of-the-art might be approaching Bayes Error.

7 Conclusion and Future Work

The report aims to classify the structural images via Convolutional Neural Network (CNN). The report utilized ResNet-50 as our baseline model and coupled it with transfer learning. We also built ensemble models by stacking the five best ResNet-50 for each task to further improve and stabilize the performance of the classification, the results of which are in Table 1. It was also found that the majority of error (66% for Task 1, 40% for Task 5 and 25% for Task 7) came from mislabeled data, and that the boundaries between classes were not clearly defined.

Since we found a large portion of error was due to either mislabeled data, or, more prominently, data near classification boundaries, a promising avenue of future work would be revisiting the dataset structure and labeling process. Although such dataset design is outside the scope of our research, removing the ambiguity between classes could help lower human and Bayesian error and improve overall performance.

The report also recommends the future work can work on the multi-label classification for structural images. In this report, the classification is divided into three different tasks. However, one structural image always has multiple attributes. For instance, one pixel-level image can be classified as non-collapse and minor damage. Therefore, if the future research can build one single multi-label classification model, the model can be more practical for post-earthquake reconnaissance.

8 Team Member Contributions

Langston Nashold: Created the initial transfer learning model, did a large portion of the hyperparameter tuning, did part of the image augmentation, and created the confusion matrices. Also helped with lots of miscellaneous code, like loading data and saving models.

Chun-Liang Wu: Test different models for classification, data augmentation for Task 5 and 7, hyperparameter tuning for Task 5 and 7, build ensemble model, error analysis for Task 5 and 7.

Salil Deshpande: Hyperparameter tuning, error analysis and relabeling.

Jaewon Saw: Obtained and organized the data sets, reviewed literature to examine existing approaches, helped with preprocessing, and relabeled incorrect images.

References

[1] Francois Chollet et al. Keras, 2015.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Charles R Farrar and Keith Worden. An introduction to structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851):303–315, 2007.

[4] Chuncheng Feng, Hua Zhang, Shuang Wang, Yonglong Li, Haoran Wang, and Fei Yan. Structural damage detection using deep convolutional neural network and transfer learning. *KSCE Journal of Civil Engineering*, 23, 08 2019.

[5] Yuqing Gao and Khalid M. Mosalam. Deep transfer learning for image-based structural damage recognition. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):748–768, 2018.

[6] Yuqing Gao and Khalid M. Mosalam. Peer hub imagenet: A large-scale multiattribute benchmark data set of structural images. *Journal of Structural Engineering*, 146(10):04020198, 2020.

[7] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. pages 770–778, 2016.

[9] Xiao Liang. 06 2019.

[10] Niannian Wang, Qingan Zhao, Shengyuan Li, Xuefeng Zhao, and Peng Zhao. Damage classification for masonry historic structures using convolutional neural networks based on still images. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1073–1089, 2018.

9 Appendix: Hyperparameter Tuning

Table 2: Hyperparameter Tuning

	Initial LR	LR Decay	Frozen Layers	Val Accuracy
Task 1				
29	0.00278	0.79656	134	0.93789
33	0.00170	0.82071	128	0.93583
20	0.00177	0.73389	121	0.93501
8	0.00181	0.85374	130	0.93459
12	0.00312	0.72173	130	0.93418
Task 5				
1	0.001582	0.769320	130	0.852018
2	0.001007	0.743979	118	0.847534
3	0.001308	0.739929	127	0.843049
4	0.002698	0.643310	130	0.838565
5	0.003679	0.680110	130	0.834081
Task 7				
1	0.036720	0.831557	148	0.830532
2	0.003082	0.867293	129	0.817927
3	0.002851	0.778810	148	0.816527
4	0.018452	0.809755	144	0.815126
5	0.016607	0.799959	145	0.812325