Mapping disaster risk from aerial imagery-improving accuracy using snapshot ensemble

Dev Shah
Department of Computer Science
Rutgers University
ds1560@scarletmail.rutgers.edu

Kush Aswani
Department of Computer Science
Rutgers University
kaa189@scarletmail.rutgers.edu

Tejasva Tomar
Department of Computer Science
Rutgers University
tt425@scarletmail.rutgers.edu

Abstract

In areas like the Caribbean which face a considerable risk from natural hazards like earthquakes, hurricanes, and floods, there is a devastating effect on human life because of this. This is especially true where houses and buildings are not up to modern construction standards. The traditional method for identifying high-risk buildings involves going door to door by foot, taking weeks if not months, and costing millions of dollars. Our challenge is to help automate this process by predicting the roof type of each building from drone imagery. Our goal is to correctly classify each of the building footprints with the roof material type. We used convolutional neural nets to help solve this multi-class image classification problem. Ensembling different models would have helped improve the overall performance but training CNNs is computationally very expensive. Hence, we used snapshot ensembling to create ensembles of different models using minimal computational resources. The original paper naively takes an average of the M models, but we furthermore investigate different ensemble methods. In experiments, we evaluate our approach compared to the original based on the log loss scores.

1 Introduction

In countries like St. Lucia, Guatemala, and Colombia, which are in the Caribbean region, face considerable risk from natural hazards like earthquakes, hurricanes, and floods, these forces have a devastating effect on human life. This is especially true where houses and buildings are not up to modern construction standards, often in poor and informal settlements. The traditional method for identifying high-risk buildings involves going door to door by foot, taking weeks if not months, and costing millions of dollars. One particularly relevant characteristic is roof construction material. Roof material is one of the main risk factors for earthquakes and hurricanes and a predictor of other risk factors, like building material which is not readily identifiable from the air. Our goal is to use provided aerial imagery to classify the roof material of identified buildings in the three countries. Models that can accurately map disaster risk from drone imagery can help drive faster, cheaper prioritization of building inspections and target resources for disaster preparation where they expect the most impact.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

1.1 Dataset

This problem is an ongoing data science challenge hosted by drivendata.org. The dataset contains aerial drone imagery for buildings prepared by the joint work of WeRobotics and the World Bank Global Program for Resilient Housing. The images are annotated with characteristics that matter the most to building inspectors. The images provided are seven high-resolution GeoTIFF images of different areas. Using the geoJson library in python, we segmented the whole image into individual roof images. In total, we have five different classes. The total number of images in training set in each class is as follows concrete cement(1377), healthy metal(7381), incomplete(668), irregular metal(5241), others(193). However, there exist a few images in our data set which have not been verified, causing an underlying ambiguity in the labels for these images. We have ignored these images while training our models. We further split this data into training (80%), validation sets(10%) and test set(10%). The total number of test images is 7326 images for which the organizers don't provide the labels. Our final aim is predict the labels for these images using our trained models.

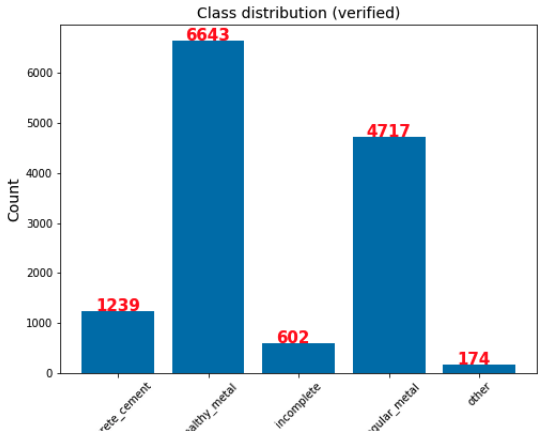

Figure 1: Dataset Class Imbalance

To tackle this problem of class imbalance we performed different data augmentation techniques like rotation, random crops, adding noise and sharpening.

2 Related Work

Recently in the field of neural networks, there has been a lot of talk about ensembles of different models as the approach to developing high-performance image classification systems. Typically these outputs are combined using voting rules, statistical techniques, or other integration schemes. Despite its distinct advantages, the use of ensembling for deep neural networks is not nearly as wide-spread as it is for different algorithms. One primary reason for the lack of adaption is the cost of learning multiple neural networks. Training deep networks can last up to weeks. As the training cost for ensemble methods increases linearly, it is uneconomical for most students/researchers without access to computational power. Snapshot Ensembles: Train 1, get M for free method focuses on learning an ensemble of multiple neural networks without incurring any additional training costs. This paper achieves the goal by training a simple and straight forward neural network. Their approach leverages the non-convex nature of neural networks. While training, every time it converges to different local minima along its optimization path, they save the model parameters. To obtain repeated rapid convergence, this paper leverage recent work on cyclic learning rate schedules. After finding M models, the paper suggests averaging all the outputs from M models naively. Many people have also tried to improve their performance by taking a weighted average of the output.

3 Our contribution

The three significant components of snapshot ensemble methods are neural network, a learning rate scheduler, snapshot selection strategy, and a model ensemble algorithm. The authors of snapshot

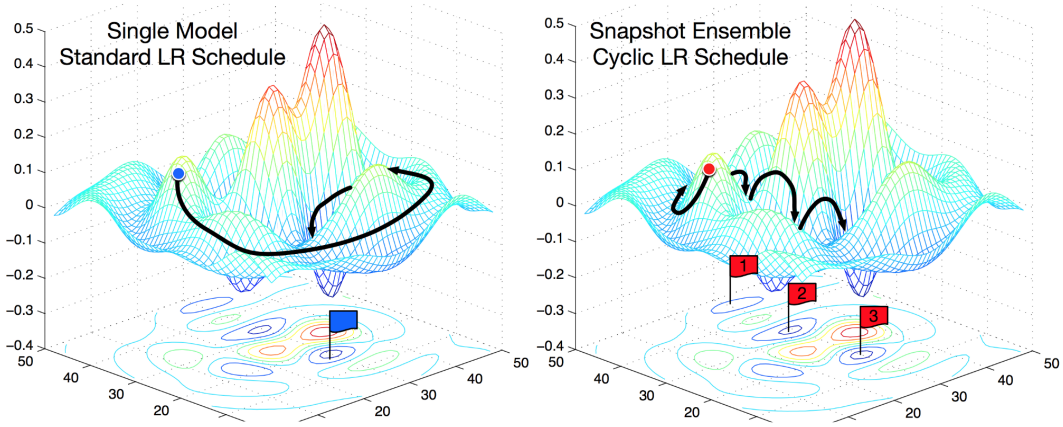

Figure 2: Snapshot Ensembling

ensembling suggest using cosine annealing cycles of learning rate scheduler in which the learning rate is reduced during each cycle, and reset to an enormous value when a new cycle starts to avoid similar snapshots. As a result, during each cycle, the learning algorithm reaches close to some local minima and escapes the local minima when the next cycle starts. This schedule helps to find less correlated snapshot models. And finally ensemble the model snapshots together, by merely taking average model outputs.
The original work does not check the effects of using different cyclic learning schedulers, different optimization techniques, as well as the efficiency of using different model ensemble methods. We first try different cyclic schedulers (linear and cosine) along with different optimization techniques (Stochastic Gradient Descent and Adam) and see what particular combination produces the best result based on the log loss of the test set. Various researchers have introduced different ensemble methods like a weighted average in which higher weights are assigned to the snapshot with higher accuracy. Many researchers make a mistake of using the training set itself to optimise the weights. However, the correct method is to find the weights using a separate validation set and then report the loss using predictions made on the test set.

$$\hat{y}_{\text{ensemble}} = \frac{1}{M}\sum_{i=1}^{M}\hat{y}_i$$

Figure 3: Normal Sum

$$\hat{y}_{\text{ensemble}} = \frac{\sum_{i=1}^{M} w_i \hat{y}_i}{\sum_{i=1}^{M} w_i}$$

Figure 4: Weighted Sum

Having tried simple model averaging as well as weighted model averaging, we then implement a locally weighted average in which a similarity matrix is created in the following way:
For any given sample from the test set, we first use a Gaussian similarity kernel to calculate the similarity between the M predictions for the test sample and the M output predictions for each of the training samples. For two examples a and b, the similarity for the prediction outputs from the M snapshot models is given by:

Then for each test sample, go through all training samples, and take the weighted average of training labels:

$$\text{sim}(a,b) = e^{-\sum_{i=1}^{M}||\hat{y}_a - \hat{y}_b||_2^2}$$

Figure 5: Calculating similarity

$$\hat{y}_{\text{ensemble}}(a) = \frac{\sum_{i=1}^{m_{train}} sim(a, train_i) y_i}{\sum_{i=1}^{m_{train}} sim(a, train_i)}$$

Figure 6: Locally weighted Ensembling output

We also propose another method to find the most probable model out of the 4 models for a given test sample using a Dirichlet distribution. For any given model, we find all the samples in the validation set where it gives the correct predictions. We then find the Dirichlet distribution parameters using

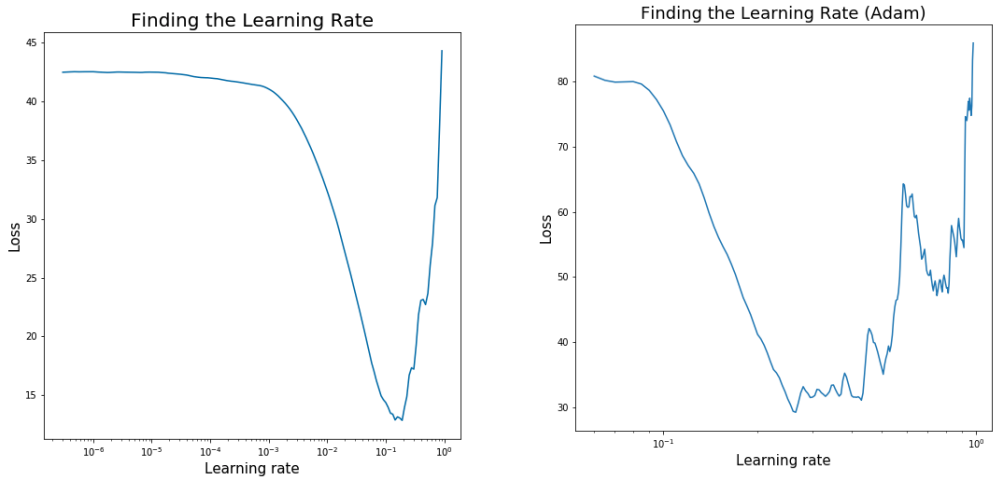

Figure 7: Finding learning rate range sgd and adam

MLE to fit the predictions for these correctly predicted samples. We do the above procedure for all the models. Then for any given test sample, we find the predictions for each of the models. For each model, we then check how likely were the predictions it made to come from the Dirichlet distribution for that model. We then select the model whose predictions are most likely to come from its corresponding Dirichlet distribution. We finally take the predictions made using this model. We have used the Dirichlet distribution as we are modeling the probabilities of five classes.

4 Experiments

4.1 Finding learning rate range

To find the upper and lower bound for the cyclic learning rate, we use the learning rate range test for both SGD and Adam. In this, the learning rate is gradually increased from a minimum to a maximum, and the loss is plotted. The range is then decided by looking at the curve where the drop in the loss is maximum. We find the optimal learning rate to be 0.1 to 0.001 for SGD and 0.1 to 0.0001 for Adam.

4.2 Cyclical Learning rates

Now, we use two different ways to cycle the learning rate for snapshot ensembling: linear and cosine. The formula for cosine annealing is given as follows:

$$\alpha(t) = \frac{\alpha_0}{2}\left(\cos\left(\frac{\pi \mathrm{mod}(t-1, \lceil T/M \rceil)}{\lceil T/M \rceil}\right) + 1\right)$$

M – Number of cycles
T – Total number of epochs
t – Epoch Number
α_0 - Starting learning rate (max value)

Figure 8: Cosine annealing formula

The figures(910) below tell us how the learning rate varies between the upper and lower bound. We have used a cycle length of 75 epochs and a total of 300 epochs.

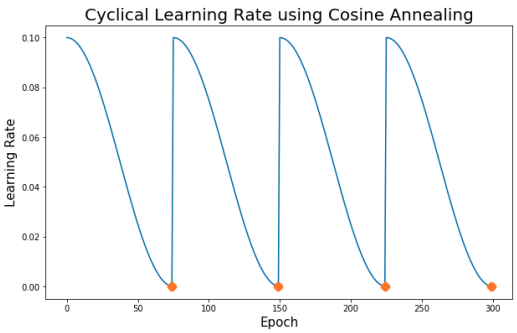

Figure 9: Cosine annealing Cycles

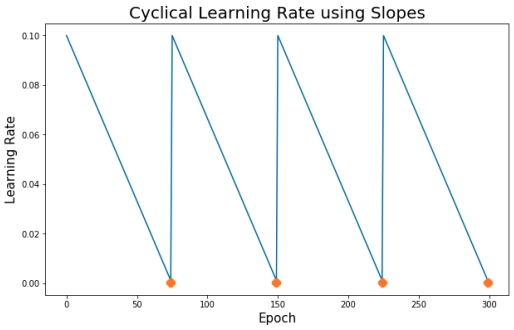

Figure 10: Linear Cycles

4.2.1 Snapshot Ensembling

Before proceeding it is important to note that we tried both the cyclical learning rates for both SGD and Adam; however, the figures below are only shown for SGD, which performed better than Adam. The figures for Adam can be found in the appendix. The figure below tells us the comparison between using a standard LR scheduler in which the learning rate is gradually decreased every till it reaches a constant.

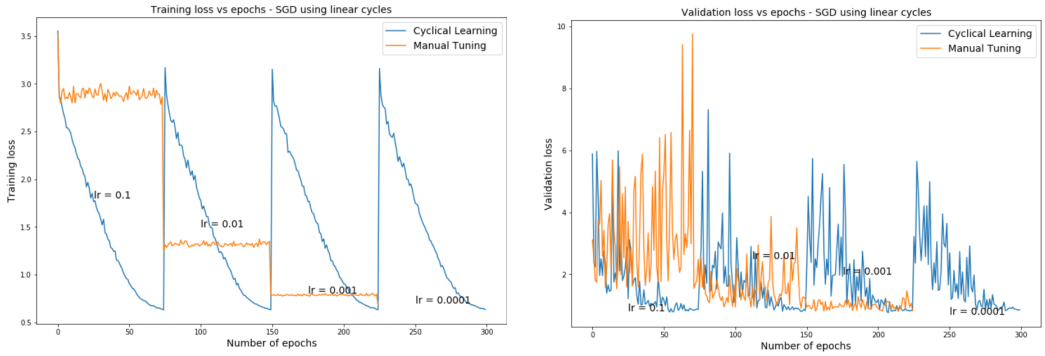

Figure 11: Linear Cycles SGD

Note that the paper for snapshot ensembling proposes that we use the model at the end of every cycle. However, we found out that instead of selecting the last model for every cycle as the snapshot, we select the best model in every cycle by looking at the validation loss, and we see that it performs better as summarized in the table below. The baseline model for our challenge is using a pretrained resnet50 model. We then proceed to use different lr scheduler as well as different optimisers and different instances of snapshoting as described above. Finally, we perform different ways to ensemble our snapshots, the results of which are shown below.

Model	Optimizer	Ensembling methods			
		Simple Average	Weighted Average	Similarity (Gaussian Kernel)	Most probable model using Dirichlet distribution
ResNet50 with linear cycles	Adam	0.7821	0.7750	0.9622	0.8167
	SGD	0.7636	**0.7590**	0.96	0.7802
ResNet50 with cosine annealing	Adam	0.7912	0.7804	0.9787	0.7843
	SGD	0.7779	0.7716	0.9619	0.7734

Figure 12: Using the best model in every cycle as the snapshot looking at the validation loss

5 Conclusion

From the above results, we see that the best performing model was the weighted average snapshot ensembling model using linear cycles(SGD optimizer), which gives a value of 0.7590 on the test set. We then submitted our model to the competition hidden test set where we are currently ranked 63rd among 1303 teams, which was a great achievement for us since this is our first-ever hands-on project using convolutional neural networks. Also, we believe that the Gaussian similarity model does not perform as well as the others due to us finding the similarity between the train and test predictions, and since our model has been trained on the training set, there exists a risk of overfitting. An improvement for this particular model would be to use a separate validation set (on which the model has not been trained), make the predictions of the M models on this validation set, and then calculate the similarity between the predictions of the test and validation set. Finally, we calculate the weighted average using the true labels of the validation set and the similarities as the weights. We also notice that our proposed method of finding the most probable model for any given test samples using the Dirichlet distribution performs better than the model which uses the Gaussian Kernel similarity and almost as well as the weighted average

References

[1] Gao Huang , Yixuan Li , Geoff Pleiss, Zhuang Liu, John E. Hopcroft, Kilian Q. Weinberger. SNAPSHOT ENSEMBLES: TRAIN 1, GET M FOR FREE. In ICLR 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016

[3] Ilya Loshchilov and Frank Hutter. Sgdr:Stochastic gradient descent with restarts. arXiv preprint arXiv:1608.03983, 2016.

[4] Tommaso Furlanello,Zachary C. Lipton,Michael Tschannen,Laurent Itti,Anima Anandkumar. Born-Again Neural Networks. [5]William H. Beluch, Tim Genewein, Andreas Nurnberger,Jan M. Kohler. The power of ensembles for active learning in image classification. In CVPR2

Contributions

Our project was divided into three major sections: the first was to extract the roof top images from the .tiff files provided to us. The next task was pre-process the data and get the training and validation sets ready for the network to be trained. Finally, we trained various architectures and used various ensembling methods. Kush Aswani trained various architectures (focusing on snapshot enesembling selecting the last model for every cycle) using his Nvidia 2060 GPU while Tejasva Tomar preprocessed the data and implemented snapshot ensembling using the best models from every cycle. Dev managed the cloud services for this project on Amazon Sage maker while implementing Dirichlet distribution ensembling process and trained the models using the standard learning rate scheduler method. Finally, the results from the various architectures implemented were summarised in this paper.

Appendix

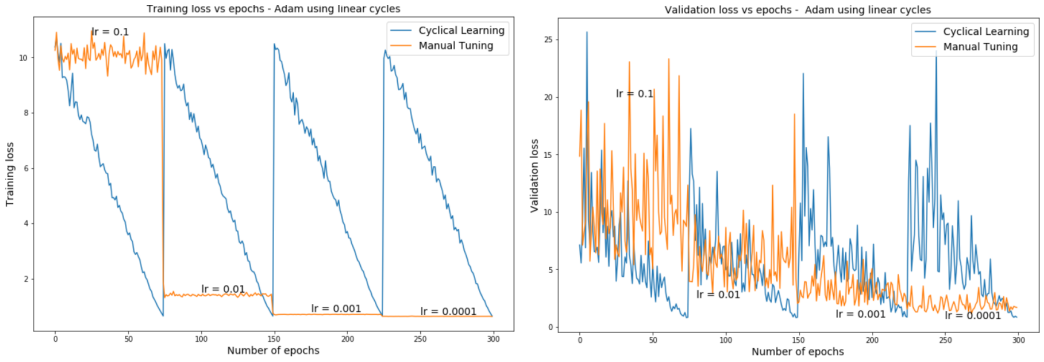

Figure 13: Linear Cycles Adam

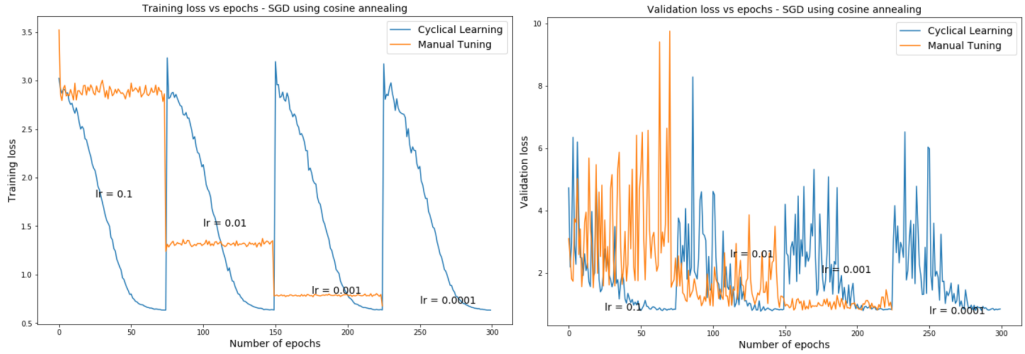

Figure 14: Cosine Cycles Sgd

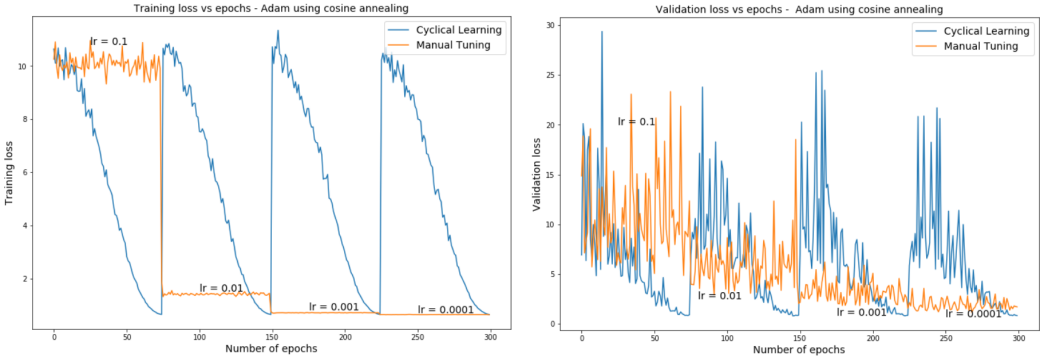

Figure 15: Cosine Cycles Adam

Model	Optimizer	Ensembling methods			
		Simple Average	Weighted Average	Similarity (Gaussian Kernel)	Most probable model using Dirichlet distribution
ResNet50 with linear cycles	Adam	0.8355	0.8329	1.0562	0.8334
	SGD	0.8476	0.8416	1.0236	0.8195
ResNet50 with cosine annealing	Adam	0.8618	0.8550	1.0256	0.8256
	SGD	0.8439	0.8365	1.0374	0.8105

Figure 16: Using the last model as the snapshot for every cycle looking at the validation loss

Figure 17: Data Augmentation