School Safety Project

Ian Atol, Stephen Hung, Hanson Egbert,
Dmitriy Timokhin, Donnie Sanchez
Anu Uppuluri

Overview

The goal of this project is to develop automatic structural classification algorithms based on a standardized group of school building photos capturing key features of the Global Library of School Infrastructure (GLOSI) structural taxonomy. By doing so, the World Bank's manual classification process can be replaced and thus can save a significant amount of time spent labeling. Furthermore, a reliable national school baseline database can be developed more efficiently, in terms of reducing the cost and time of in-field data collection.

The intended users for this project are the engineers hired by the World Bank to survey the schools. By using this project, engineers require less manual work (i.e., detailed inspections on the ground) and can instead use photos collected by local students. Another group of possible users is the analysts at the World Bank. Given the structural classifications, analysts will be able to predict the damages that can occur in the event of a natural disaster (earthquake, hurricane, tornado, flooding, etc.). By doing so, they are able to provide information to local governments and guide intervention and investment plans.

Background and Related Work

Cal Poly conducted a similar project two years ago in Pantoja et al.'s paper "An Introduction to Deep Learning." (Pantoja, Behrouzi, and Fabris, 2018) Although this paper focuses on what makes up a neural network, and their different styles, the paper implements a type of Convolutional Neural Network, AlexNet, on classifying structural damage identification. Their data was provided by The Pacific Earthquake Engineering Research Center. It was labeled for classifications that include:

- Scene level (pixel, object, or structural)
- Damage condition (yes or no)
- Spalling condition (yes or no)
- Material type (steel or other)
- Collapse state (none, partial, or full)
- Component type (beam, column, wall, or other)

- Damage level (none, minor, moderate, or heavy)
- Damage type (none, flexural, shear, or combined)

We believe that our data will be labeled similarly to the data above. Pantoja's implementation takes the raw pixel values of a structure's image as input and outputs a score for each of the categories listed above. Their algorithm can draw a bounding box with an accuracy of 77%. Our implementation will not draw a box around the identified features, but we hope to identify them accurately through other means. Hopefully, we can achieve the same accuracy as Pantoja's team, or even better.

Difficulty

We predicted that the difficulty rating for this project would be 7/10. Communication posed a bit of a problem between us and the World Bank since it was an international partnership; however, they graciously agreed to video-conference with us about once a week even though it is not as effective as a face-to-face interchange of information. Communication within the team and with the DxHub was not a problem though, as all parties were on campus here at Cal Poly. We were originally planning on using more images for our training/validation/test data from the World Bank. However, it has been rather hard for the students and engineers on the field to take enough images of their respective schools and for the engineers to label them on time. We also had to consider the pros/cons about the best technologies/techniques to use for the project: cloud services like AWS (with DxHub providing us with credits), Google Cloud, IBM Watson, Microsoft Azure vs online notebooks like Jupyter Notebook, Google Colaboratory, Cocalc; Deep Learning frameworks like Tensorflow/Keras vs PyTorch. We finally stuck with running our Python training script (aided by other LINUX scripts) - we converted from the IPython format we tested in on Google Colab and JupyterLab - on powerful AWS EC2 instances like g4dn.8xlarge and p3.8xlarge. Future developers of this project may consider using Amazon SageMaker for a fully automated experience. Other considerations included whether we should incorporate transfer learning and build upon/customize pre-trained weights from pre-existing models offered by the various cloud services or whether we should start from scratch, borrowing just the vital elements from them that are of use to us. Furthermore, an important discussion was whether we should split up and compete for building/piecing-together the best models for our cause, or focus on perfecting just a single model. In the end, we believe our estimate of the difficulty was accurate.

Relevance

In this class, our main goal was to learn about the current developments in AI/Deep learning with a focus on multi-agent systems. Although this project does not include a multi-agent system, it does include a state of the art neural network. Since 'Intelligent Agents' is the broader topic of this course and seeking to understand/implement/better the intelligence/autonomous aspect of an agent is sometimes more crucial than working on its sensors and actuators, we deem that this project is apt for the course. Through this project, we learned a lot about building a production system that can be used to help engineers, apart from honing our machine learning and API-building skills with packages/platforms like Keras, Pandas, Tensorflow, Docker, TensorFlow Serving and even certain aspects of Python like array/list splices. Our goal was to be able to create a smooth transition of knowledge and implementation from us to the maintainers at the DxHub and World Bank.

Features

This project's main goal was to produce a model that can, given photos of the exterior walls of a school, accurately classify the building based on the World Bank's Global Library of School Infrastructure taxonomy. The positive effects of this model have been mentioned in our project overview but are copied below. As helper tools to accomplish this, we also created a web scraper for the World Bank's photo repository, as well as an API to provide convenient access to the model.

The intended users for this project are the engineers hired by the World Bank to survey the schools. By using our tool, engineers require less manual work (i.e., detailed inspections on the ground) and can instead use photos collected by local students. Another group of possible users is the analysts at the World Bank. Given our predicted structural classifications, analysts will be able to better model the damages that will occur in the event of a natural disaster (earthquake, hurricane, tornado, flooding, etc.). By doing so, they are able to provide information to local governments and guide intervention and investment plans.

Requirements

- Scraper to pull data from the World Bank website
    - Make sure this is safe
- ML model/algorithm that is able to classify photos of buildings as a specific building type with high accuracy
    - Code and descriptions of our thought process
    - Multiple models
- Ability to input new photos into the aforementioned model
- Easy access to the model (API)
    Be able to access the models predicted output by sending a get request

Evaluation Criteria

Due to the short term timeline of this project, the World Bank did not define any set of required metrics for success. Since this project is only the first phase of what is hopefully a long and fruitful project, any measurable success in classifying the World Bank's building photos will be useful for the World Bank as a preliminary MVP. We verified if our scraper retrieved all of the data by asking the World Bank to verify that the 17k photos are all of the labeled photos in their database.

In order to measure the accuracy of our model, we used training/testing/validation sets for our models. By creating a validation set at the beginning of the model building process we can remove the necessary bias of model fitting. In this validation set, we use a simple formula (predicted class == actual class)/number of classes. This gives us the percent of successful classifications and gives the end-users an overview of the performance of our model.

Once we have verified that our model is acceptably accurate, we use an API to give it previously unseen input photos. Our API has reasonable response times and good documentation. We then use the API and test the model by ensuring the input photos are properly and accurately classified.

	Feature	Requirement	Evaluation Criteria
Scraper	Web scraper for World Bank's photo repository	Encrypt image file to protect sensitive data	Visually inspect images

ML model/algorithm	A model that can, given photos of the exterior walls of a school, accurately classify the building according to the GLOSI taxonomy	Well organized and commented code for implementing the model	Accuracy rating from model
Input new photos	Ability to give the model new photos to classify	Get images outside of our training set, and test with our model	Ensuring the input photos are properly and accurately classified.
API	An interface with which to interact programmatically with the model	Be able to access the models predicted output by sending a get request	Reasonable response times, documentation

System Design and Architecture

Our project was to build an ML model to classify images of school buildings to specific typologies provided by GPSS. Therefore, in order to do this, our system design is less like an application and more of a description of how we will find the data and the usages for the model afterward. The project components are described in Figure 1 below.

The ML algorithm, on the other hand, is accessible via an API provided by TensorFlow Serving. This allows the UI team to directly take user input into the app and then send POST requests to the server and receive the corresponding predictions.

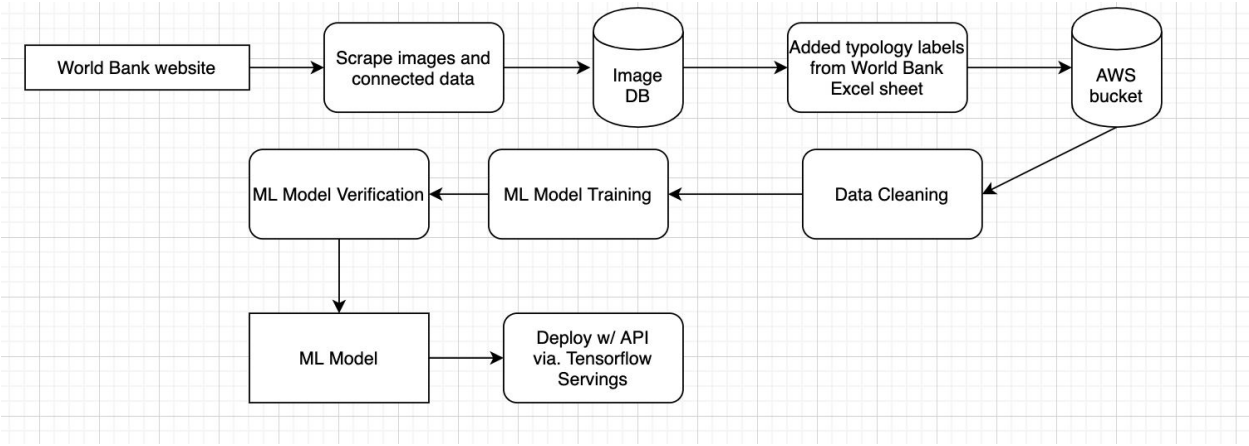

Figure 1: Block Diagram for Project Components

Implementation

To scrape the data from the World Bank website, we initially thought to use the Beautiful Soup Python library. Unfortunately, due to issues with the architecture of the website, Beautiful Soup did not meet our needs and we were forced to make a more manual Python script to scrape the data using the Requests and Urllib libraries. After scraping the images of the Nepalese schools and some of their labels from the website, we attached more labels that were provided by the World Bank to these images using another Python script. These scripts are available on the GitHub repository for reference. If access to the GitHub is needed, please contact the DxHub.

Adding the extraneous given labels to the scraped images was fairly simple, as both datasets had building ID columns and School ID columns. This way we can end up with building level and school level features which should all be useful.

After organizing and cleaning our combined dataset, we applied ML algorithms to the data and labels to try and predict the Main Structural System, Number of Stories, and Building Category (per the World Bank Global Library of School Architecture taxonomy) of a given, unlabelled image. The ML methods we applied are CNNs and Xception / InceptionResnetV2 model via Tensorflow Keras. Using these models, we use a method called transfer learning (Brownlee 2019) in order to reduce the time needed to train our model. Transfer learning is a method where a model that has already been trained on some data set is used to learn from a new data set rather than starting from zero knowledge. From this base model, we added-on new layers to be trained on our new data set.

Finally, for our API, we used TensorFlow Serving, hosted on a Docker instance inside of an AWS instance that can be accessed remotely. A request sent to the API

with a photo will respond with the results from a pre-trained model. The results from the API come in a JSON format with 3 main keys: “Building Category”, “Main Structural System”, and “Number of Stories”. Each of the values corresponding to those keys is a list of probabilities. For example, “Building Category” can return something like [0.989545524, 0.00306758168, 0.00122260209, 0.00616425881] where each element is a probability. These indices correspond to:

- 0 = LBM: Load-bearing masonry
- 1 = SF: Steel Frame
- 2 = TF: Timber Frame
- 3 = RC: Reinforced concrete

So in our example, the building category has a 98.95% chance of being Load-bearing Masonry. The other labels and their corresponding index are in our Glossary, Appendix 1.

Our main reasons for using TensorFlow Serving are as follows:

1. Our current ML model is using Keras which is TensorFlow’s API for building deep learning models. Therefore, exporting our model and using it in TensorFlow Serving was easy and supported with ample documentation.
2. Using TensorFlow Serving with Docker allows for some security in that our model will only be accessed through a docker image. Furthermore, Docker allows us to save ‘images’ and easily reproduce the server in case of problems or if anyone else wants to use our API pipeline.
3. Furthermore, setting up the TensorFlow Serving endpoints was fairly easy (“Serving a TensorFlow Model | TFX” n.d.) compared to other potential methods.

Ultimately, we found TensorFlow Servings to be fairly easy to use and setup, the docker container is set up with a simple terminal command and a POST request response is received within a couple of seconds.

Figure 2: Flow Diagram for ML Model

Validation and Evaluation

To validate and evaluate our results, we review our features list:

Our web scraper worked as intended and was able to grab the images with their labels from the provided World Bank site. However, as the project went on and expectations and goals changed, supplemental data was also provided by the World Bank themselves. These combined datasets would serve as the training data for our ML model.

The model itself ended up with 67%, 95%, and 81% accuracy on the respective categories of main structural system, height range, and building type. While the main structural system prediction rate is a bit lower than we would have liked, this is a very complex classification and relies as well on the accuracy of building type prediction. The ability to input new photos is successful, through the use of our API. The API works quickly and is well documented because of our choice of using TensorFlow Serving. Due to our use of Docker and AWS in conjunction to host this API, there is an additional benefit of security and low downtime for those that need reliable access to these predictions.

To account for the discrepancies between the categorical accuracies attained, we urge the teams that would follow up on/continue our work to consider that the amount of labeled data that we could use for training our CNN has been quite dissimilar for the three categories. This can be attributed to the difficulties in procuring and labeling said data by the structural engineers on the field, and school children, faculty and staff of the various Nepalese schools alike.

We initially experienced issues with overfitting and low performance due to one of the standard pre-trained Keras models (“Applications - Keras Documentation” n.d.) (“7 Best Models for Image Classification Using Keras” n.d.) we adopted for transfer learning at the time, namely, VGG19. As we started experimenting with these standard models, we came to know that our final model performed well when we used Xception or InceptionResnetV2. But, we hit a roadblock when the test accuracy wouldn’t improve in spite of training our models for upwards of 25 epochs. It turned out that we had turned off training/updating the weights of nearly all of the base model layers that we had borrowed for transfer learning (“Using Pre Trained VGG16 for Another Classification Task · Issue #4465 · Keras-Team/Keras” n.d.) when we were testing out our ideas on low-compute-power resources like Google Colab. But once we were given access to the mind-blowing compute power of AWS S3 and EC2, we could leverage the complete potential of transfer learning by making a third, then two-thirds and then finally almost all the layers - borrowed, novel alike - trainable.

The next and hopefully the worst (meaning that there would be little scope for improvement after fixing it) roadblock of them all that we hit was when our model wouldn’t go past the 85% mark for one of our top-performing categories - namely, height range/number of stories. We blasted past this one as well by some more hyperparameter tuning on our part - we tried monitoring the trends in accuracy across

all 30 epochs of our candidate models involving Xception and InceptionResnetV2, and found that there was a consistent increase in validation accuracy for up to 6 epochs of training, followed by fluctuations and then another peak at 26 epochs. But, we chose to stick with 6 epochs for generating our final “model.h5” file, just because the loss started fluctuating at 26 epochs, instead of the steady dip we experienced until the 6th. The real turn around for the performance of our model came when we started tweaking the learning rate hyperparameter. Perhaps the Stochastic Gradient Descent (SGD) optimizer could have given us different results, but we picked the Adam optimization algorithm after looking up popular reviews for classification problems similar to ours.

Finally, to increase the accuracy by almost 10% across all categories, we switched from the default learning rate of 0.001 to 0.0005, 0.0002 and then to 0.0001. Although none of the latter three could perform consistently well across all categories, yet we chose 0.0001 due to the highest overall decrease in loss it provided. Furthermore, although we were successful in bringing down the loss with the test dataset to less than 1 for some categories, yet the overall loss could not be brought down to lower than 1.9. There was almost a tie between Xception and InceptionResnetV2, which was broken by this Deep Neural Network benchmark paper: (Bianco et al. 2018). This official Keras documentation was one of our main references in this process: (“Applications - Keras Documentation” n.d.)

User Feedback

We had ongoing feedback throughout the timeline of our project because we were in regular contact with our clients. Due to the ever-changing nature of the specifics of scope and expectations for the project, an overall summary of user feedback is very difficult. Additionally, due to the outbreak of coronavirus near the end of the quarter, we were unable to present the project as it was finished to the World Bank team. We hope that we met their short-term expectations well for the project and that this paper will additionally help them be able to utilize the model in a more efficient way.

Conclusion

Overall, our project turned out to be successful given the evolving expectations of the parties involved, for example, with respect to what columns on the file titled

“final_main_columns.csv” (refer to S3 or GitHub) to use as the categories for image classification, which ones of the categories to classify on first, etc. Our model was able to classify with 67%, 95%, and 81% accuracy on the respective categories of main structural system, height range, and building type.

We ended up using a hierarchical architecture for the model. Since ‘Main Structural System’, is a subcategory of ‘Building Category’ with 12 more categories, it’s clear that the model would perform better on the 4 in ‘Building Category’, as shown in Figure 2. This led to a jump in accuracy since we used the prediction of ‘Building Category’ to predict ‘Main Structural System’ in our network. We have an API that is hosted inside a Docker container in our AWS instance with a good response time. From a technical perspective, we learned about containerization, implementing and improving a neural network model, and also about cleaning and processing large amounts of data. From a less technical perspective, we learned a lot about communication in a corporate setting. Future work in this project could include implementing the full GLOSI taxonomy to be classified, as well as classifying structural damage. The app team is working on building a front-end system to be able to access our model.

Glossary

Appendix 1. Index/Label for ML Model Results

Main Structural system(index, value):

- 0, UCM/URM2: Rubble (or field) stone in mud mortar
- 1, SFM1: Lightweight gravity steel frame with URM walls
- 2, TF: Timber frame
- 3, RC1: Reinforced concrete moment frame with/without in-fill walls that do not contribute to lateral stiffness
- 4, UCM/URM6: Dressed stone in cement mortar
- 5, UCM/URM7: Rectangular block in cement mortar
- 6, UCM/URM1: Dry rubble (or field) stone masonry
- 7, TFM: Lightweight gravity timber frame with URM walls
- 8, RC2: Reinforced concrete frame with in-fill walls as stiffening element
- 9, RC3: Reinforced concrete short column frame
- 10, UCM/URM4: Rectangular block (brick, concrete block) in mud mortar

- 11, A: Earthen Blocks or compressed stabilized soil blocks in mud mortar
- 12, SFM2: Lightweight gravity steel frame with RM, CM or precast walls
- 13, RM: Rectangular block in cement mortar with steel reinforcement
- 14, RC4: Reinforced concrete combined or dual system
- 15, CM: Rectangular block in cement mortar with RC confinement

Number of stories(index, num stories)

- 0, 1
- 1, 2
- 2, 3
- 3, 4
- 4, 5
- 5, 6

Building cat(index, category)

- 0, LBM: Load-bearing masonry
- 1, SF: Steel Frame
- 2, TF: Timber Frame
- 3, RC: Reinforced concrete

References

"7 Best Models for Image Classification Using Keras." n.d. Accessed March 17, 2020. https://www.it4nextgen.com/keras-image-classification-models/.

"Applications - Keras Documentation." n.d. Accessed March 17, 2020a. https://keras.io/applications/.

"Applications - Keras Documentation." ———. n.d. Accessed March 17, 2020b. https://keras.io/applications/#fine-tune-inceptionv3-on-a-new-set-of-classes.

Bianco, Simone, Remi Cadene, Luigi Celona, and Paolo Napoletano. 2018. "Benchmark Analysis of Representative Deep Neural Network Architectures." *IEEE Access* 6: 64270–77. https://doi.org/10.1109/ACCESS.2018.2877890.

Brownlee, Jason. 2019. "Transfer Learning in Keras with Computer Vision Models." *Machine Learning Mastery* (blog). May 14, 2019. https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/.

Pantoja, M., Behrouzi, A., & Fabris, D. (2018). An introduction to deep learning. Concrete International, 40(9), 35–41. https://digitalcommons.calpoly.edu/aen_fac/125

"Serving a TensorFlow Model | TFX." n.d. TensorFlow. Accessed March 17, 2020.

https://www.tensorflow.org/tfx/serving/serving_basic.
"Using Pre Trained VGG16 for Another Classification Task · Issue #4465 · Keras-Team/Keras." n.d. GitHub. Accessed March 17, 2020. https://github.com/keras-team/keras/issues/4465.