Global Program for Safer Schools (GPSS)

Zachary Cipolla, Evan Diaz, Nicholas Hansen, Laura McGann, Melissa Nardone, Finian Rawson

Cal Poly Digital Transformation Hub, World Bank
CSC 480 - Dr. Kurfess

Cal Poly San Luis Obispo, Computer Science and Software Engineering Department
March 2022

Contents

Abstract

The Safer Schools project is a collaboration between the Cal Poly Digital Transformation Hub (DxHub), Cal Poly students, and the World Bank's Global Program for Safer Schools (GPSS). The purpose of the project is to easily identify schools at risk for structural damage from natural disasters in developing countries. The GPSS is an ongoing project, and the result will be a mobile application that will be accessible to school administration. The mobile application will take in several images of a school's structure and classify its vulnerability, and depending on the risk, notify a person of authority or engineer so that resources can be allocated to provide support.

The objective of the Global Program for Safer School project for the current quarter was to develop a binary classifier for structural images of school buildings. The model for the binary classifier was required to be developed in an ArcGIS notebook using TensorFlow and Keras and was trained on a labeled dataset from OneDrive. The input for the model is four structural images from one school building and the output of the model is the probability that the structure represents "wall" or "frame". The model was developed using transfer learning and used the Keras pretrained model Densenet121 for the base model. Next steps for the project will be the integration of the trained model into Survey123.

I. Introduction, Background, Related Work

Overview

The Safer Schools project is a collaboration between the Cal Poly Digital Transformation Hub (DxHub), Cal Poly students, and the World Bank's Global Program for Safer Schools (GPSS). The purpose of the project is to simplify the identification of school infrastructure vulnerability on a large scale.

The result will be integration into a mobile app that can be used by school administrators, architectural engineers, and other community members and personnel involved in school evaluation. The application will guide the users through photographing a school's structure. The images will then be analyzed using artificial intelligence, and the results will be provided to engineers and officials who can now make an informed decision whether a particular school requires infrastructure investments and funding.

Background

Natural disasters put more than 1,000,000 school buildings in low- and middle-income countries at risk of collapsing, putting an estimated 875 million children and teachers at risk of injury. Monitoring every school and its structure is a huge financial and time investment. Using artificial intelligence, the school structure analysis can be automated, and investments can be prioritized for schools with high structural vulnerability. School building structure was classified using the 12 structural taxonomy classifications used by the GPSS.

The Global Program for Safer Schools project was initially started in Winter 2020 by students in CSC 486, Human-Computer Interaction, and was continued by students in CSC 480 and CSC 580, Artificial Intelligence. The work completed this quarter includes migration from the existing classification model to an ArcGIS notebook environment. The model now needs to be retrained and tuned with a more balanced data set and implemented and tested in Survey123, the GPSS's desired mobile application.

Difficulty

Considering the quantity and breadth of information available about this project, a large amount of time was spent researching existing work and relevant topics.

Some prerequisite knowledge on the AI topics involved, such as neural networks and computer vision related operations, were required to effectively participate in this project. Since project team members had varying degrees of proficiency with these technologies, a large time investment was made into knowledge transfer and collaboration.

Previous groups laid the groundwork for our goals this quarter, so a considerable amount of time was spent looking and learning from the work that they already did. Most of our focus was improving upon the model previously built with newly added restrictions and requirements. The initial few weeks were spent adjusting to unfamiliar technologies being used, such as services like ArcGIS, Keras, and TensorFlow. Therefore, the degree of difficulty can be assessed at 9/10.

Related Work

Students from both Cal Poly's artificial intelligence courses (CSC 480 and CSC 580) have previously worked on the Safer Schools project with Cal Poly's DxHub and the World Bank. The project utilizes course materials to solve real world problems and provides an opportunity to develop and train neural networks for structural classification.

The main technical component of this project included developing/improving a neural network classifier which is a common instance of machine learning, a sector of artificial intelligence. We also gained real-world experience applying such technology to develop a system for a customer, working within the constraints of available data and required formats and tools we needed to use.

II. System and Design Architecture

Overview

The objective of the Global Program for Safer Schools project for the current quarter was to develop a binary classifier for structural images of school buildings. The model for the binary classifier was developed in an ArcGIS notebook using TensorFlow and Keras and was trained on a labeled dataset from OneDrive. The input for the model is four structural images and the output of the model is two probabilities, one each for the "wall" and "frame" structure class labels. An end-user survey was also created in the format of Survey123, the mobile application that will eventually be used by school administration for live building evaluation. Once the trained model and survey are connected, the new input images – taken in the Survey123 survey with SmartCamera integration – will be fed to the model which will then output a class prediction.

Diagram

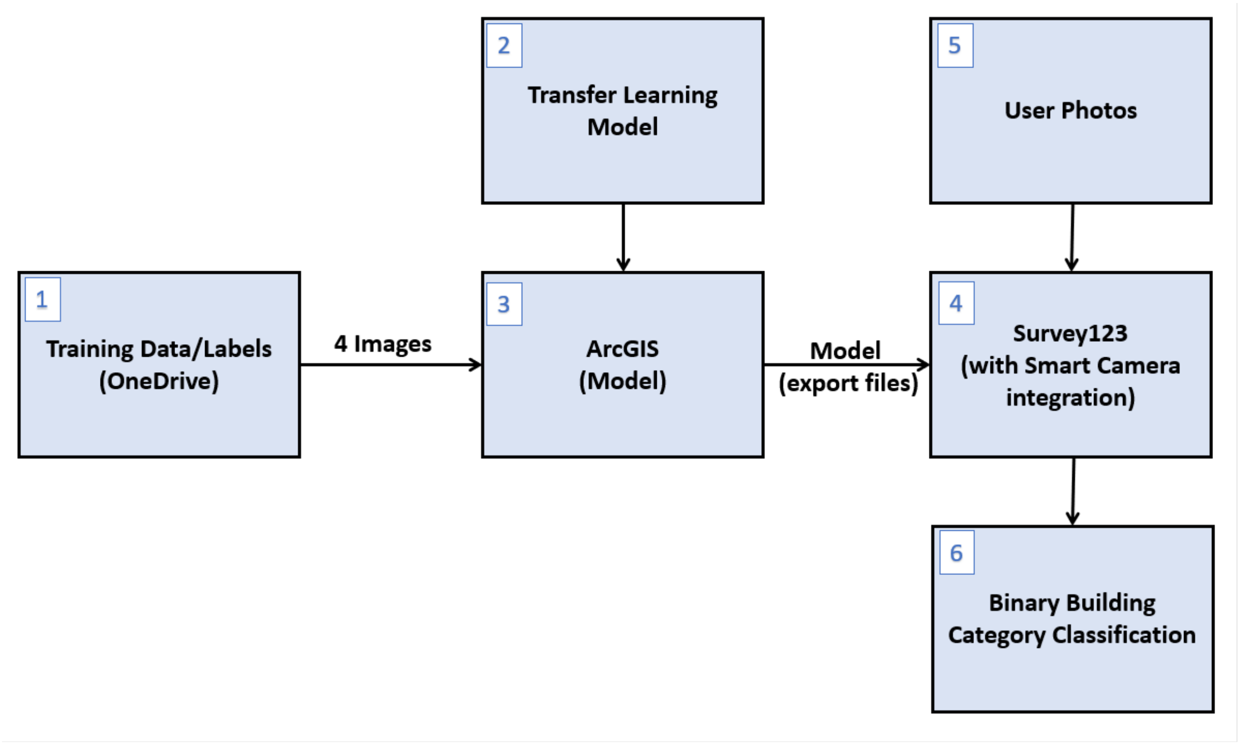

1
Training Data/Labels (OneDrive)
4 Images
2
Transfer Learning Model
3
ArcGIS (Model)
Model (export files)
5
User Photos
4
Survey123 (with Smart Camera integration)
6
Binary Building Category Classification

Figure 2.1. System design.

Functional Components

1) OneDrive (Dataset storage)

On OneDrive, there is a 15 GB file with a large set of folders, each associated with a specific building. Each folder is labeled with a building code that is unique to the building. Within a specific building's folder, there are photos of the building – relevant to structural condition evaluation – from different perspectives: inside (diaphragm) and outside (building side) the building. As of now, we have a CSV file containing the correct wall/frame classification of around 1000 of those buildings, noted by their codes.

Each row contains such a building code – a reference key to the building image folders – and the corresponding label.

2) Acquired base image classification models for transfer learning

We built our model using transfer learning, experimenting with 5 different base models (existing image classifiers). Our choices include Densenet121, Xception, VGG16, InceptionV3, and MobileNetV2. The rest of our model (much fewer layers, much less project-specific building image training data) was trained independently on top of these base models.

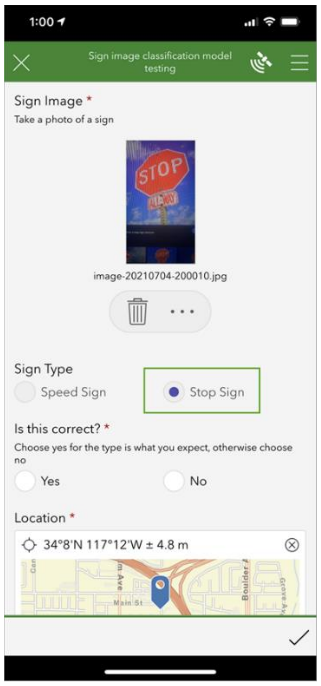
3) ArcGIS (Model creation and training)

We implemented our building image classification model in an ArcGIS notebook. Since our task was to begin experiments on new transfer learning base models and rectify previous overfitting issues, we simplified the scope of the problem. Namely, the initial model input decreased from 8 photos to only 4 photos (2 building side (external), 2 diaphragm (internal)) and the output is only a binary building category classification (described further in Section 6 below). Future work will be required to export our model into a format compatible with Survey123 (described in further detail in Section 4 below).

4) Survey123 (Mobile application)

Survey123 is the mobile application that our fully trained model will be integrated into so school administrators can take pictures with their phones and the model will immediately classify the images on-device.

The photo below shows an example of the Survey123 app in use, specifically with a model for classifying speed vs stop signs. In this example, the user took a new photo of a stop sign, and the trained model integrated into the app automatically predicted the stop sign classification. The user then has the option to (in)validate the model's prediction by selecting "yes" or "no".


1:00

Sign image classification model testing

Sign Image *

Take a photo of a sign

image-20210704-200010.jpg

Sign Type

Speed Sign [✓ Stop Sign]

Is this correct? *

Choose yes for the type is what you expect, otherwise choose no

Yes No

Location *

34°8'N 117°12'W ± 4.8 m

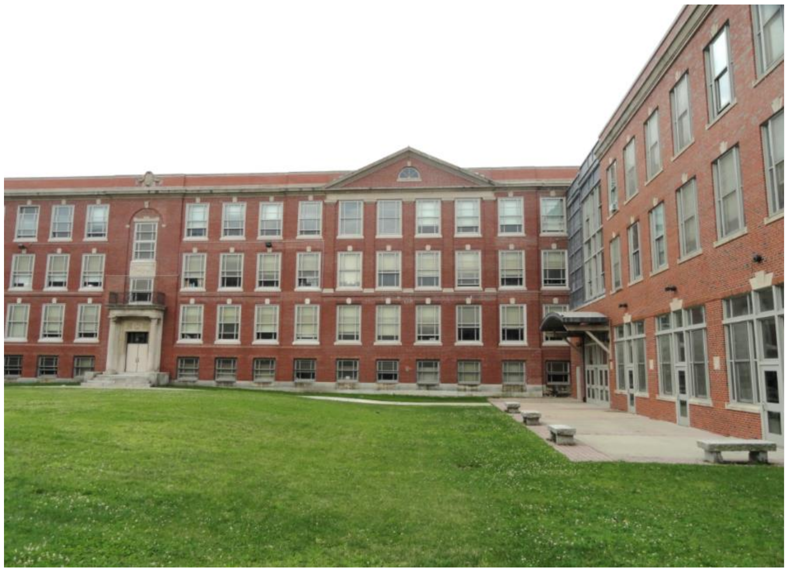
(https://learn.arcgis.com/en/projects/train-a-model-to-identify-street-signs/#train-a-model)

Figure 2.2. Example survey in Survey123.

Survey123 integrates well with ArcGIS in that a model built from ArcGIS's own FeatureClassifier can be exported directly into a format Survey123 accepts. Even though we implemented our models in an ArcGIS notebook, we did not use the FeatureClassifier, but rather a raw TensorFlow/Keras model. Thus, Future Work details what must be done to export the model in these formats, but once done, those files should be easily connected to the Survey123 + SmartCamera survey, and school administrators should be able to take multiple images for classification.

5) User photo (new input for trained model)

After the trained model is integrated into Survey123, users can utilize the SmartCamera feature to take different photos of a school and see the resultant binary classification. The model accepts a total of four user photos – split into 2 different categories (diaphragm and side images) – to make its building category prediction. Photo examples are shown below.



(https://upload.wikimedia.org/wikipedia/commons/5/58/Side_view_-_Boston_Latin_School_-_DSC09810.JPG)

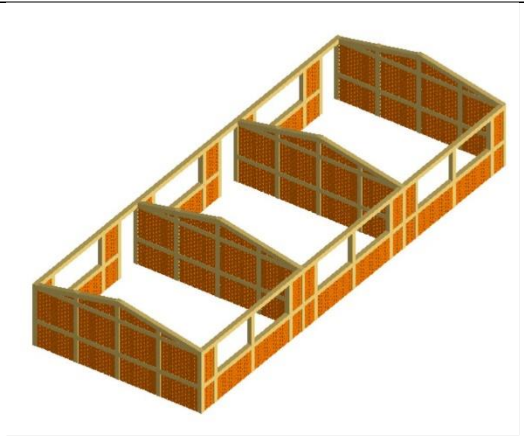
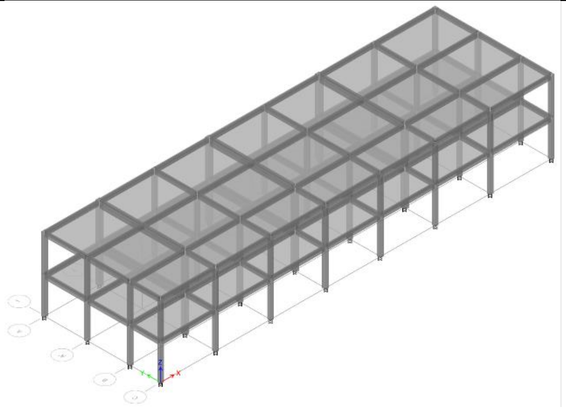
Figure 2.3. Example External Building Side Photo.

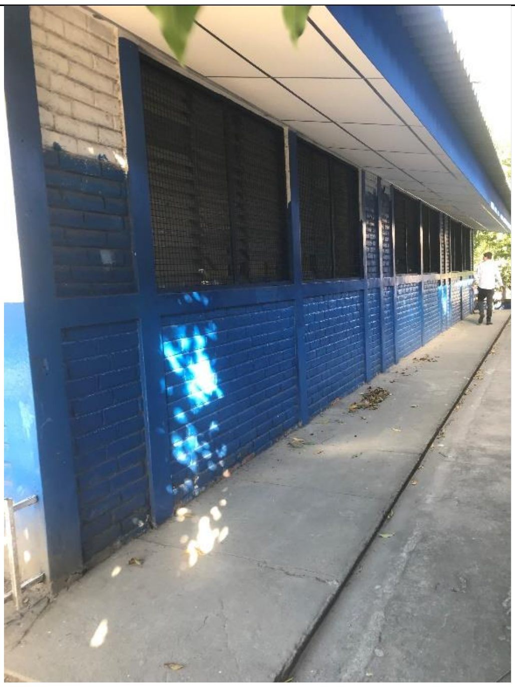
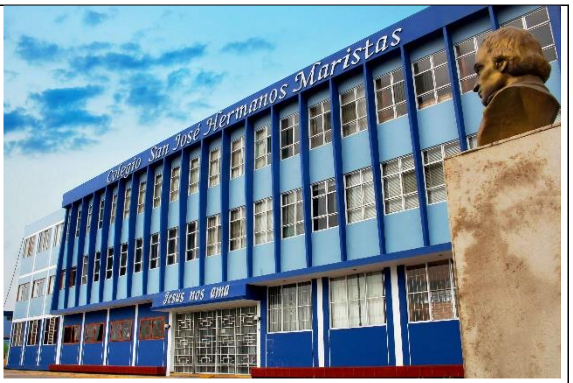
(https://live.staticflickr.com/7869/32031661877_566f606ca1_h.jpg)

Figure 2.4. Example Internal Diaphragm Photo.

6) Building category classification (wall/frame)

Our goal was to simply predict a binary building category classification: wall or frame. The reason for this small scope was due to the nature of the project's goal: conduct *initial* experiments on various base models and try to avoid overfitting the model.

	Wall	**Frame**
Model		

Real Image		

(https://gpss.worldbank.org/en/glosi/building-catalogue-types?f%5B0%5D=construction_type_block%3A31&f%5B1%5D=construction_type_block%3A36)

Figure 2.5. Example image classification.

III. Prototypes and Implementation

Overview

The implementation of this project is dictated largely by the end goals: implementation of an image classification model, specifically in ArcGIS so it can be exported and compatible with the mobile application Survey123. Figure 3.1 depicts a high-level, structural overview of tools used and their relation to each other.

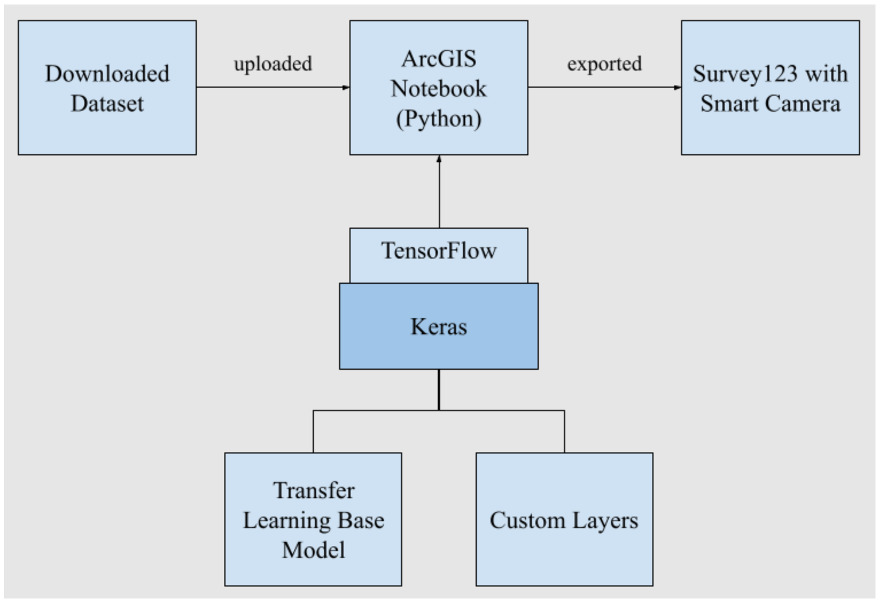


Figure 3.1. High-level structural implementation overview: tools and components used.

Beginning on the top left in Figure 3.1 above, our dataset was sourced and stored elsewhere, so we developed scripts to scrape and download a subset of the full online dataset. This data was then uploaded to ArcGIS, an online python notebook environment (explained in more detail in future sections below) in which we developed our building image classification model. We used Keras, an open-source neural network library built on top of TensorFlow, the more general and complex (also open-source) machine learning library. Keras supplies numerous pre-trained image classification models. We tested multiple of these models as base models for transfer learning and then added our own custom neural network layers on top of them to "learn" our specific classification problem. This is all implemented in Python. Once trained and tuned, our model must be exported to formats compatible with the mobile app, Survey123, for final deployment in the field.

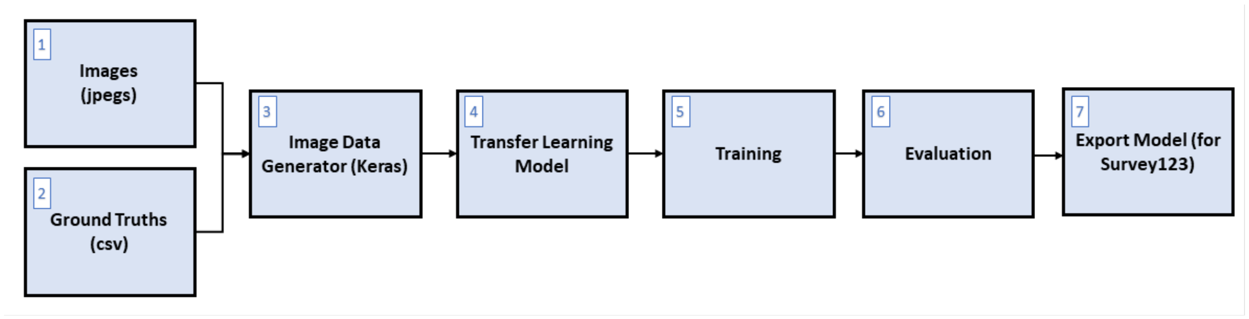


Figure 3.2. Classification model functional implementation overview: steps in the construction, training, and deployment process.

Figure 3.2 represents the architectural flow within ArcGIS. Once the training images along with the ground truths are uploaded into an ArcGIS notebook, they are split into the training and validation sets using an image data generator. The binary classification model is then developed using transfer learning and the layers are modified to meet the project requirements. The model is retrained using selected training parameters, and the results are analyzed. Based on the model validation accuracy and loss, the training parameters are tuned. Finally, once the model has reached a desired accuracy, it must be exported in a compatible format and linked to the Survey123+SmartCamera survey.

Technologies, Tools, Languages, Development Environment

ArcGIS

An ArcGIS notebook was the main development environment for the neural network. For our purposes, it functions nearly identically to a Jupyter notebook or Google Collab, but we used ArcGIS instead of these other development environments to better prepare for exporting the model in the formats needed for Survey123.

Survey123

Survey123 is a mobile application – closely related to ArcGIS – meant for crowd-sourcing images (e.g. for a survey). While the app can be used both for initial dataset collection (i.e. with users taking and labeling new photos), we will only be using it on the downstream end with SmartCamera integration (i.e. after model development and training). In this use, the trained model is “uploaded” to the app, and instead of users manually labeling new photos, the integrated model outputs a predicted classification on its own.

To “upload” the model, it was developed in an ArcGIS notebook and subsequently exported into a TensorFlow Lite model (.tflite) and an Esri model definition file (.emd). It was initially thought that to make the model compatible with Survey123, the model must be developed in ArcGIS. After further consultation with Survery123 representatives it was noted that as long as the model could be exported in a TensorFlow Lite model and an Esri model definition file, the model could be integrated into Survey123. This means that it would be possible for future teams to utilize different notebook environments. Once this model is integrated into the Survey123 app, any on-site personnel can take pictures of new buildings and immediately receive a predicted classification. The few engineers on staff can work remotely to verify model predictions.

TensorFlow/Keras

The deep learning libraries used for the project include TensorFlow and Keras. Keras is a high-level neural network library with a much more accessible API (purely Python) compared to TensorFlow, another, more general open-source machine learning library on which Keras is intended to be deployed. Keras' high-level methods internally call TensorFlow's functions, delegating the more complex, mathematical operations to TensorFlow's optimized C++ implementation.

Transfer learning is applied by using the pre-trained models defined by Keras and modifying them by adding additional layers and retraining the neural network. The input layer is modified to accept four images of size 450 by 600 of the RGB color space, and the output layer is modified to provide a binary classification. The pre-trained model with the additional layers is retrained and then ready to be exported. For the project, Keras is also used for splitting the dataset for training and validation and developing an image data generator for training the model. The image data generator loads the images by batch and formats/resizes them by the specified parameters during the model training.

Transfer Learning Models

Keras has many pre-trained base models which we tested for our classification implementation. The base models tested include Densenet121, Xception, VGG16, InceptionV3, and MobileNetV2. The model that incorporated Densenet121 produced the best performance metrics, and, therefore, was selected as the final model ready for export.

Prototype Functionality

The model is trained and developed in an ArcGIS notebook using the provided dataset. The model architecture and structure can be modified in the notebook along with the training parameters. Modifications to these values impact the model performance. The model can be exported in the needed formats directly from the notebook. The TensorFlow Lite exporting process has been included in the current implementation, but the process for Esri model definition (.emd) file creation still needs to be determined. Once exported, the model can be connected to the Survey123 survey. School faculty will take new images via the Survey123+SmartCamera survey and images will be classified as either walls or frames. Eventually, alternative classification models will be developed and included in the application to rank school overall structural stability.

Obstacles and Implementation Issues

Before our team started work with the DxHub and World Bank, many other students and groups contributed to the Safer Schools project. This meant that a lot of work needed to be done reviewing previous work before the development and implementation process could be started. The team's initial understanding of deep learning and transfer learning – agnostic of the specific image classification problem domain – was also limited. Because of this initial setback, more time needed to be devoted to background research.

During the project development process, several issues occurred. One major issue was with data acquisition. Originally, the 15 GB image dataset was stored in OneDrive, and when the attempt was made to download the images, the team was unable to obtain a zip file that was not corrupt due to overall dataset size. A scraper script was developed to attempt to download the images to resolve the issue. This process took several attempts due to OneDrive throttling. Once the images were finally uploaded into ArcGIS, it was noted that 22 jpeg images were corrupt. Because of this issue, those images had to be removed from the training set.

During the initial processes of training our models, we quickly reached the credit limit that our ArcGIS accounts had been given initially. This was a temporary problem, as Cal Poly is granted a "functionally unlimited" number of credits for ArcGIS, and our accounts were fully funded again. In addition, we also encountered issues with memory usage caps in ArcGIS. This was not a problem we could resolve with our accounts, as these limits are nonadjustable. Instead, we were forced to make some adjustments to the original model parameters set by previous groups, such as image input size, when training in the ArcGIS environment to reduce memory usage.

Next, finding which base model worked best for us was easier said than done. Not only did we have to try and compare results from using different pre-trained models on ImageNet, but tweaking the number of epochs, batch sizes, and resolution of images resulted in varying outcomes and training durations. Transfer learning with these configurations ended up taking a few hours minimum.

Finally, when training our models, we had to use four images of the same building as input for a single classification. Exploring methodologies for implementing this functionality most effectively is an ongoing process. For now, we are leveraging the strategy of previous groups – concatenating the image data into a single input layer – but there is a possibility one can achieve better results by leveraging other methods.

IV. Validation and Evaluation

Evaluation Plan

We evaluated our progress based on the following milestones and metrics:

1. Affirm we can extract and organize data into model-accessible formats and locations.
2. Affirm we can export the best model for compatibility and use with the mobile application Survey123 + SmartCamera.
3. Affirm we can build a model that accepts 4 building images and outputs a single binary building structure classification.
    a. Compare different model architectures using performance indicators including accuracy, precision, recall, f1-measure, and confusion matrix.
    b. Modify the training parameters of the best performing model until performance is increased to a maximal (or diminishingly improved) amount.
    c. Compare the same metrics of running the selected best model but with different types (side vs diaphragm) and number of input images.

Mappings of Features to Requirements

The table below contains the preliminary requirements and features that were intended for the project. The features and requirements were later revised and updated in the Conclusion portion of this documentation.

Index	Features	Requirements
1	Engineers and school administrators can view the predicted classification of new school building photos.	Model can be exported in the 3 formats needed for Survey123 compatibility. Survey123 must accept four images as an input for the classification model including two side images and two diaphragm images.
2	Via a seamless experience, engineers and school administrators receive trusted-enough predictions to minimize manual effort and cost of analyzing school building structural integrity.	Data is easily accessible to the model and easily selectable (i.e. to use only a balanced subset). Model can be exported into the required format for Survey123. Performance measured using f1 score, recall, precision, accuracy, and analyzed using a confusion matrix.
3	Engineers and school administrators are informed of the scope/limits of the model, so they know when to trust it and when to rely on human analysis instead, or when to wait for future development.	Experimental model implementations output predictions for one GLOSI taxonomy class and performance (execution time, precision, recall) can be compared. Performance indicators must be above acceptable thresholds before use from the public/school administrators.

Table 4.1. Mappings of features to requirements.

Usage of Evaluation Criteria

The first evaluation criterion is a binary yes/no checkbox: are we able to acquire the data in a usable format for model training? While we experienced the most difficulties with this step, we were able to download our dataset of images, upload it to an ArcGIS notebook, reorganize the images into the needed directory structure, split the data into train and validate groups, and train the model. Out of 1049 buildings, 976 were successfully loaded to train the model. This criterion holds as a standard for evaluation because it checks if we have completed a necessary step. This satisfies our first requirement (R1).

The second evaluation criterion is also a binary yes/no: can we export the model in the three necessary formats (.tflite, .dlpk, .emd) to upload to Survey123 without issues and conduct a survey using SmartCamera? While we initially thought our decision to build the model in an ArcGIS notebook would ensure our ability to export in the three formats, discoveries late in the quarter showed otherwise. Easy export from ArcGIS relied upon using ArcGIS's own FeatureClassifier and its built-in export functions. However, the FeatureClassifier can only accept a single image as input. Thus, at the termination of this project, we found ourselves in a dichotomy: the raw TensorFlow model we built that accepts 4 images but cannot be immediately exported, vs. the unused FeatureClassifier that could be easily exported but could not accept all 4 images at once. This problem is discussed more in Future Work, but we were able to approach this criterion from the opposite direction as well: Survey123. We were able to construct a survey that can capture 4 new building images (2 of each kind) with the SmartCamera integration and then – given a connected model – predict the building's structure class. The few caveats here are that while the survey can capture 4 photos, still only one can be input *at a time* into *any* connected model, and there is no model actually connected yet, thus no actual testing occurred. This component of the problem is also talked about in the Future Work section. While this criterion does satisfy our second requirement (R2), we ran out of time to actually meet the criterion, although we discovered (and discuss in this document) much information with regards to how to do so in future.

The third evaluation criterion, rather than a binary yes/no, is more of a set of standard machine learning model evaluation metrics: accuracy, recall, precision, f1-measure, confusion matrix, and training/processing time. After further discussions with our external partner, as well as experimentation, we determined precision, recall, and f1-measure are the most valued metrics, with accuracy as an initial indicator. The Table 4.2 below shows the values of these metrics for each model tested, and the following Table 4.3 shows further experimentation with the hyperparameters of the model found as "best": Densenet121. (This criterion satisfies requirements 3 and 4.)

Model	**Accuracy**	**Precision**	**Recall**	**F1-Measure**
Densenet121	0.722	0.662	0.815	0.697
Xception	0.708	0.618	0.745	0.652
VGG16	0.639	0.565	0.421	0.457
InceptionV3	0.708	0.669	0.657	0.637
MobileNetV2	0.701	0.701	0.773	0.690

Table 4.2. Model performance testing results.

Selected Model: *Densenet121*

Image Size	**Epochs**	**Steps per Epoch**	**Batch Size**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
(450, 600, 3)	20	100	4	0.711	0.673	0.82	0.739
(450, 600, 3)	50	150	4	0.755	0.714	0.664	0.771

(480, 640, 3)	10	150	4	0.810	0.776	0.871	0.820

Table 4.3. Selected model training parameters performance tests.

Overall Assessment

Our evaluation criteria proved to be relevant measures of assessing feature development and requirement satisfaction. Only the first and third criteria were satisfied in the final scope of our project – we successfully developed a binary classification model in ArcGIS trained on given data – but much of our findings and starter work for criterion 2 are discussed below in Conclusions and Future Work.

V. Conclusions and Future Work

Overview

Over the course of the quarter, the team successfully migrated the existing transfer learning framework into an ArcGIS notebook. The model was then modified to classify school building images as either containing frames or walls. The input of the existing transfer learning framework was decreased from eight to four images to simplify a user's interaction with the classification. Several transfer learning models were researched and tested along with experimentation of their training parameters. Keras's Densenet121 pre-trained model resulted in the best performance and was chosen as the base model. Metrics including recall, precision, and f1-score were added to the notebook to analyze performance and compare methods and model implementations. Finally, the model was exported as a TensorFlow Lite file, and a boilerplate, compatible Survey123 survey was created. Immediate next steps include additionally exporting the model to an Esri model definition file (.emd), finishing image handling and model use in the survey itself, and exploring new model-survey relationships to provide the best outcomes. Future work should be able to use these existing pieces and notes on further direction to finish connecting a model to an active survey for use by school administrations around the globe.

Features to Requirements

Due to the short timeline available to the team for the project development, the scope of team's implementation had to be decreased and not all desired features were implemented in this iteration of the classification project. No frontend features were included in the current version of project. However, the supporting classification model was developed and trained in ArcGIS. Future students will need to focus on the frontend features for school administration and the integration of the model in Survey123.

Table 5.1 includes the updated features of the project along with their corresponding requirements. More features can be added to the design with new teams of students.

Index	Features	Requirements
1	The project will include a wall and frame binary classifier.	The classifier is to be implemented in an ArcGIS notebook. The classifier model was developed using transfer learning, takes an input of four images for each school building (two side images and two diaphragm images), and produces a binary classification of either wall or frame for the set of images for a building.
2	The trained model must be compatible with Survey123.	For the model to be compatible with Survey123, the trained model should be exported to a TensorFlow Lite file and Esri model definition file. Edits to the survey itself must also be made to "pass" all 4 images to the model.
3	The performance of the classification model should be easily observable.	Performance metrics should be incorporated into the neural network model. The metrics include accuracy, precision, recall, and the f1-score. The model performance should be measured against validation and testing datasets.

Table 5.1. Features and requirements.

System Design and Implementation

The final system design of the project can be found in Figure 5.1. For the CSC 480 winter quarter, the team was able to implement blocks 1, 2, and 3. While block 4 is partially developed, it and blocks 5 and 6 need to be further developed and integrated by future teams. Our team's implementation consisted of four main components: dataset extraction from OneDrive and integration into ArcGIS, transfer learning binary classification model development and training, exporting a model into a TensorFlow Lite format, and providing the basis of an eventual end user survey.

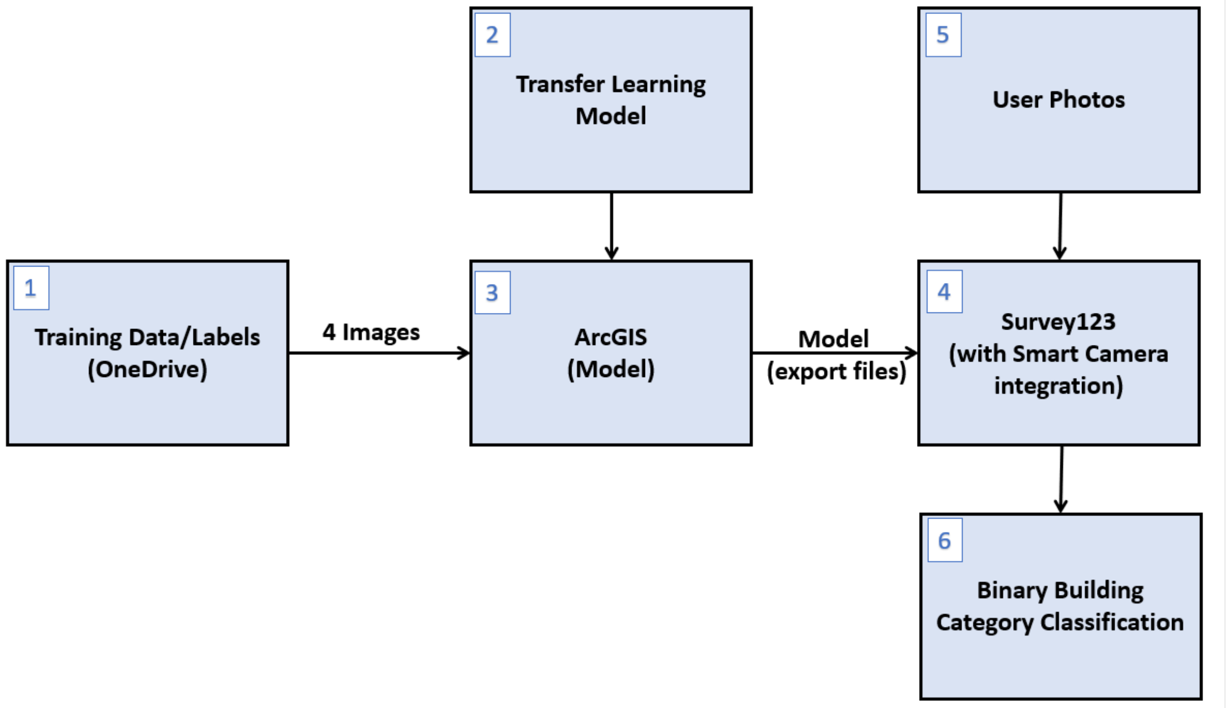

2
Transfer Learning Model
5
User Photos
1
Training Data/Labels (OneDrive)
4 Images
3
ArcGIS (Model)
Model (export files)
4
Survey123 (with Smart Camera integration)
6
Binary Building Category Classification

Figure 5.2. System design.

Several Keras pre-trained base models were tested including Xception, VGG16, ResNet50, EfficientNetB0, EfficientNetB4, and Densenet121. Once the best performing base model was selected, many different training hyperparameters were tested. The optimal training parameters implemented in the design can be found below in Table 5.3.

Model	**Image Shape**	**Epochs**	**Batch Size**	**Steps per Epoch**	**Dropout Rate**
Densenet121	(480, 640, 3)	10	4	150	0.3

Table 5.3. Final model training parameters.

Relevance for Class Topic

Students from both Cal Poly's artificial intelligence courses (CSC 480 and CSC 580) have previously worked on the Safer Schools project with Cal Poly's DxHub and the World Bank. While this course served as a conceptual basis for artificial intelligence, our project allowed us to gain insights in the implementation of machine learning solving real world problems and provides an opportunity to develop and train neural networks for structural classification.

The main technical component of this project is developing/improving a neural network classifier which is a common instance of machine learning, a sector of artificial intelligence. We will also gain real-world experience applying such technology to develop a system for a customer, working within the constraints of available data and required formats and tools we need to use.

Lessons Learned

Before our team started work with the DxHub and World Bank, many other students and groups contributed to the Safer Schools project. This meant that a lot of work needed to be done reviewing previous work before the development and implementation process could be started. The team's initial understanding of deep learning and transfer learning when applied to image classification was also limited. To overcome these initial setbacks, we rapidly gained new knowledge about TensorFlow and Keras, as well as features and parameters specific to the problem of image classification.

During the project development process, we learned from the several issues that occurred. Our first major problem with data acquisition taught us not to underestimate the difficulty of and time needed for initial data preprocessing. We also more specifically learned about OneDrive throttling and how to overcome it via a scraper script. Once we had the data, we discovered that even then, parts of it were corrupt, reinforcing the importance of understanding the contents of the data you work with.

Our next, more minor issues were with maxed out credit limits on ArcGIS and capped memory usage. To solve the former, we gained more experience communicating with third parties and working in the framework of a larger organization. To work around the latter, we exercised our creative problem-solving skills and learned some more nitty gritty details about TensorFlow/Keras models and the ArcGIS environment.

Finally, to successfully train and tune multiple models, we had to build on our rapidly acquired, fundamental knowledge of TensorFlow, Keras, transfer learning, and image classification and apply it to our specific problem. We discovered the finer distinctions between epochs, batch sizes, and image resolutions, as well as the varying effects these changes – and different base models – had on ultimate results, including metrics and longer-than-expected training durations.

Overall, this project provided a valuable opportunity to experience working on a real-world project with both a real client and internal, supporting parties.

Future Work

Immediate future work centers around completing the connection between the model and Survey123. This involves both exploring different model structures, as well as building out the survey itself.

When training our models, we must use four images of the same building as input for a single classification. Exploring methodologies for implementing this functionality most effectively is an ongoing process. For now, we are leveraging the strategy of previous groups by concatenating the image data into a single input layer, but there is a possibility one can achieve better results by leveraging other methods.

Currently, we have a transfer learning model that performs relatively well on the provided dataset for the classification of wall/framed buildings. However, due to time constraints, we were not able to complete the task of integrating this model into the Survey123 + SmartCamera application. This is due to the fundamental difference between the problem statement at hand and the capabilities of the third-party mobile application we are attempting to deploy on. We were tasked with training a model that could accept multiple images of the same building (from various viewpoints), inputting *all* these images into a single deep learning model and subsequently outputting a single classification for the combined features of these images. However, after consultation with software representatives from Esri, it was discovered that the mobile application (Survey123) does not currently support multi-image input into a single model. This poses a challenge, as to maintain the integrity of our building structure feature extraction, it is preferred that simultaneous multi-image input with single output classification be performed during prediction.

There are various ways future teams can approach this challenge and various suggestions we received from Esri representatives to attempt to integrate this. While Survey123 does not support the type of input we would prefer, it does support the injection of custom data preprocessing and postprocessing JavaScript. It is possible that this functionality could be leveraged in the future to mimic the originally intended model architecture, namely, to join the results – in some to-be-determined manner – of 4 independent outcomes, one for each input image. In addition, Survey123 does support the usage of multiple different deep learning models, meaning when building a survey, you can select which model to "pass" each image to. Therefore, it could be possible to segment the functionality of the original model architecture into multiple models. One possibility is depicted below in Figure 5.4.

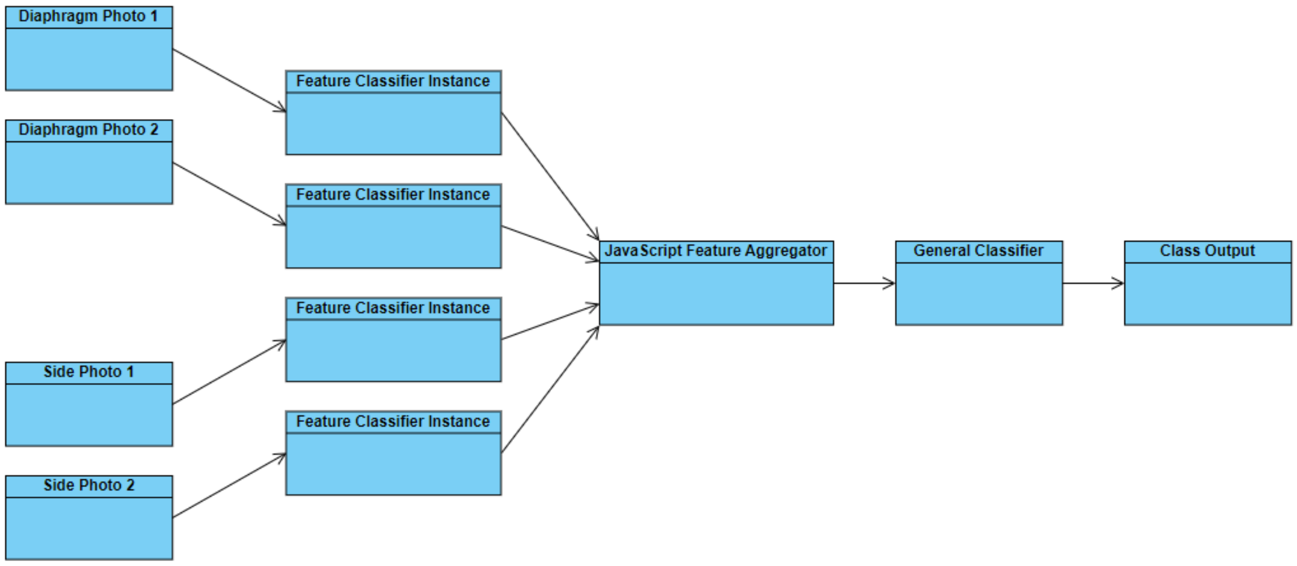

Diaphragm Photo 1
Diaphragm Photo 2
Side Photo 1
Side Photo 2
Feature Classifier Instance
Feature Classifier Instance
Feature Classifier Instance
Feature Classifier Instance
JavaScript Feature Aggregator
General Classifier
Class Output

Figure 5.4. Potential Alternative Architecture.

This alternative architecture in Figure 5.4 adheres to the single image input limitations, but rather than selecting or combining inputs, all four images are still utilized by applying the model to each image individually to extract significant building features. The model could be the same for all four images, or two models could be trained, one for diaphragm images and one for side images. In the "dual model" case, both models would still output a respective set of identified features. The model(s) could then be implemented directly using ArcGIS's FeatureClassifier, or in a raw TensorFlow/Keras neural network. In any case, these output features/classes would then be aggregated in a post-processing JavaScript (which can be integrated into a survey) and fed to another general classifier that would interpret the entire set of features for final classification. The general classifier could take the form of either a secondary deep learning model that is trained on these features, or – if possible – a simplistic mathematical or logical model mapping these features to an overall building classification. This architecture would rely on the existence of training images labeled with distinct features, not just overarching classifications. If that data does not exist or is too hard to acquire, a similar architecture (single-image-input model applied to each image individually with aggregated results) could still be developed. The difference would be that each application of the model would directly output an overall class prediction, and then those predictions would be aggregated into the final class prediction without the need for the “general classifier.”

One entirely different alternative to the on-device classification in Survey123 is to offload the classification work. Instead of trying to build and export the model in a format compatible for Survey123, the mobile app could simply be used to capture new images. These images are then available online via ArcGIS’s services, and from there they can be classified offline via whichever model is needed.

All these directions can be explored from the work we have done here. Even beyond, teams could attempt to make more complex models that output multiple building structure taxonomy classes, all in an effort to make schools safer around the globe, and to make doing so a more seamless process.

VI. References

"Basic Classification: Classify Images of Clothing: Tensorflow Core." *TensorFlow*, https://www.tensorflow.org/tutorials/keras/classification.

Chatterjee, Poulomi, and Sharath Kumar Nair. "A Complete Understanding of Dense Layers in Neural Networks." *Analytics India Magazine*, 18 Sept. 2021, https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/#:~:text=Units%20are%20one%20of%20the,dimensionality%20of%20the%20output%20vector.

"Keras Models - Sequential and Functional Model of Keras." *TechVidvan*, 5 July 2021, https://techvidvan.com/tutorials/keras-models/#:~:text=Keras%20provides%20two%20types%20of,an%20easy%20to%20use%20model.

MLK. "Activation Function." *MLK - Machine Learning Knowledge*, 2 Nov. 2019, https://machinelearningknowledge.ai/glossary/activation-function/.

"The Sequential Model: Tensorflow Core." *TensorFlow*, https://www.tensorflow.org/guide/keras/sequential_model.

Team, Keras. "Keras Documentation: Dense Layer." *Keras*, https://keras.io/api/layers/core_layers/dense/.

Team, Keras. "Keras Documentation: Transfer Learning & Fine-Tuning." *Keras*, https://keras.io/guides/transfer_learning/.

Wolfe, Mike. "Tensorflow vs Keras: A Comparison." *Medium*, Towards Data Science, 26 Aug. 2021, https://towardsdatascience.com/tensorflow-vs-keras-d51f2d68fdfc.

Hebbar, Nachiketa. "Transfer Learning with Keras(Resnet-50)." *Chronicles of AI*, Chronicles of AI, 16 July 2021, https://chroniclesofai.com/transfer-learning-with-keras-resnet-50/.

Verma, Shiva. "A Simple Guide to Using Keras Pretrained Models." *Medium*, Towards Data Science, 5 Oct. 2021, https://towardsdatascience.com/step-by-step-guide-to-using-pretrained-models-in-keras-c9097b647b29.