School Safety Project Document

Alberto Rodriguez, Theo Watkins, Lemar Popal, Joshua Frederick, Caleb Watts, Donald Sanchez, Justin Evans, Weston Montgomery, Sherry Lin, Maxim Korolkov, Nick Schnorr, Jack Pietrok

Computer Science Department, Cal Poly San Luis Obispo

CSC 570-02: Deep Learning and Knowledge Graphs

Professor Franz Kurfess, PhD

March 16, 2021

Abstract

In rural areas of Nepal and surrounding countries, earthquakes are common and often deadly. The World Bank has been collecting data over the past decade on school buildings in these areas in order to evaluate their structural integrity and determine the risk earthquakes pose to students in school. Originally the World Bank manually classified the data by studying photographs of the buildings and reporting structural features and overall integrity. Recently they have been attempting to replace this system with an automatic classifier which leverages computer vision techniques. This project attempts to improve the state of the World Bank's current automatic classifier by gathering additional training data from previously unused World Bank sources. This data is used to create a binary classifier that will work to classify the buildings that are input to the Global Library of School Infrastructure (GLOSI). In order to validate the results from this new binary classifier the School Safety team has created scripts that print out the accuracy, precision and F score of the model. This along with the GradCam platform will allow for better training in future scenarios.

Introduction, Background, and Related Work

Introduction

The goal of this project is to capture key features of the Global Library of School Infrastructure (GLOSI) structural typologies [1], by expanding a deep learning classification system that integrates school building photographs. By doing so, we hope to improve upon The World Bank's current manual classification method by supplementing or replacing it with an automatic system. Use of this system can reduce the cost and time of in-field data collection, and it will help promote creation of a reliable school baseline database.

To complete this task, we use deep learning to create a more robust image classification system that can use images taken in the field and provide classification of the buildings based on previous classifications. Along with this improvement to the GLOSI, we will also be writing up a best practices document of the type of images and number of images that would be optimal to submit to the platform. This would allow more basic users rather than just trained structural engineers to upload and submit data to the GLOSI platform. The World Bank and governments would then be able to get a better understanding of the risks to their school infrastructure.

Background & Related Work

This project has been worked on by multiple Cal Poly groups in previous quarters, in collaboration with the World Bank and Cal Poly DxHub. Previous teams have built a baseline ML model, built on InceptionResnetV2, which was used to predict three building criteria: building category, main structural system, and building height. They obtained accuracies of 81%, 67%, and 95% in the respective categories [2]. In addition, some work has been done to improve the training data pipeline, as well as improve the ML model using data augmentation and multiple

input capability. Other work in Image classification has been conducted in many other fields such as medical research and product classification which can be altered to look at building safety and architecture [3]. Also the use of Knowledge graphs has been used to further refine image classification providing better accuracy [4].

System Design

The overall project is comprised of the following components: data preprocessing, data pipeline, logic tree with binary classifications (model experiment 1), model performance evaluator, heat map (model experiment 2), data augmentation improvement (model experiment 3), and model fine tuning (model experiment 4). Our team is composed of two sub-teams that will work distributedly across all seven components of the project. The ultimate goal of this project is to create an ML system which can classify school site data (building images), so the World Bank and local governments can make meaningful policy decisions based on quantitative evidence.

Functional Components

Data Preprocessing

Previous teams have already scraped a World Bank website for the front image of every building in Kyrgyz and Nepal. The sponsor requested our teams to scrape the remaining images of each Nepal building. There are roughly 8000 buildings with more than eight images per building. This system interfaces with the OneDrive as the images must be uploaded here.

Data Pipeline

Once all the required images were uploaded to OneDrive and the full dataset was assembled there, we set up a data pipeline to access that dataset for training and testing of our ML models. This involved us using an API to pull data to AWS where our training and testing was performed. The World Bank has not specified if they need this to be accessible for the other services, so this was only used internally for training.

Logic Tree with Binary Classifications

The previous classifier model takes in a group of eight photos at a time, and it outputs the result to a concatenated string describing the structural or functional features of the building. Our goal was to adjust the classification approach and simplify the previous one to a binary decision tree model. We used separate models to classify binary aspects of a building, such as material type and building height. Then, these properties were used to determine the overall building category, based on a predefined logic table.

Model Performance Evaluator

This component measures the performance of the classification models by calculating accuracy, precision, recall, F1 score, and area under the curve. It takes in the classifications on the testing data made with the model and compares them to the classifications entered manually. Some of these measurements are built into frameworks like Keras, but we also built a script that used

scikit learn metrics to evaluate these parameters..

Heat Map

Researchers from Georgia Tech have created a program called “Grad-CAM” which uses deep networks and gradient-based localization to visually represent distinguishable features using “heat maps”. Our team referenced the research and modified the publicly available Grad-CAM code in order to generate a “heat map” which can distinguish different aspects of the building. The original Grad-CAM software is able to distinguish cats and dogs in the same image; for our purposes, we needed to distinguish components such as building stories, columns, and windows. This technology allows us to determine if our binary classifiers are converging on aspects of a building that are relevant to the question.

Improve Data Augmentation

We primarily used Keras for data augmentation. During the model training stage, we specified various data augmentations as images are passed to the model. Some useful augmentations included zoom, rotate, brightness adjustment, horizontal flip, and shear. Some other common augmentations are not useful in the context of images of school buildings, like vertical flips, because images aren’t taken upside down. Since the domain was user taken images of schools, some augmentations that represent user error or normal user variation, such as noise injection could prove beneficial. These augmentations significantly increase the diversity of images available for training models, without actually collecting new images, and make class proportions more balanced. Furthermore, the model will be able to generalize more (i.e. not overfit) and should perform better on test data.

Fine Tuning Model

Fine tuning the model could easily be done with AWS Sagemaker. Sagemaker iteratively tunes the hyperparameters of a neural network to improve performance. Additionally, Keras will allow us to alter parameters like the learning rate, loss function, batch size, and number of epochs. Choosing the correct values or functions for these parameters can significantly impact the accuracy of the system. Through research and experimentation we hope to find the ideal values for these parameters with our newly augmented model and data.

Figure 1: High-level System UML Diagram

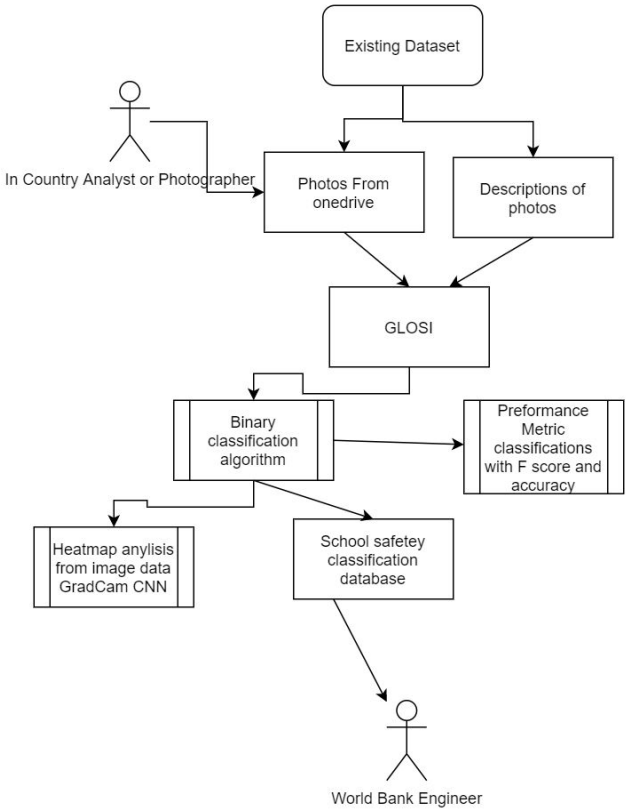
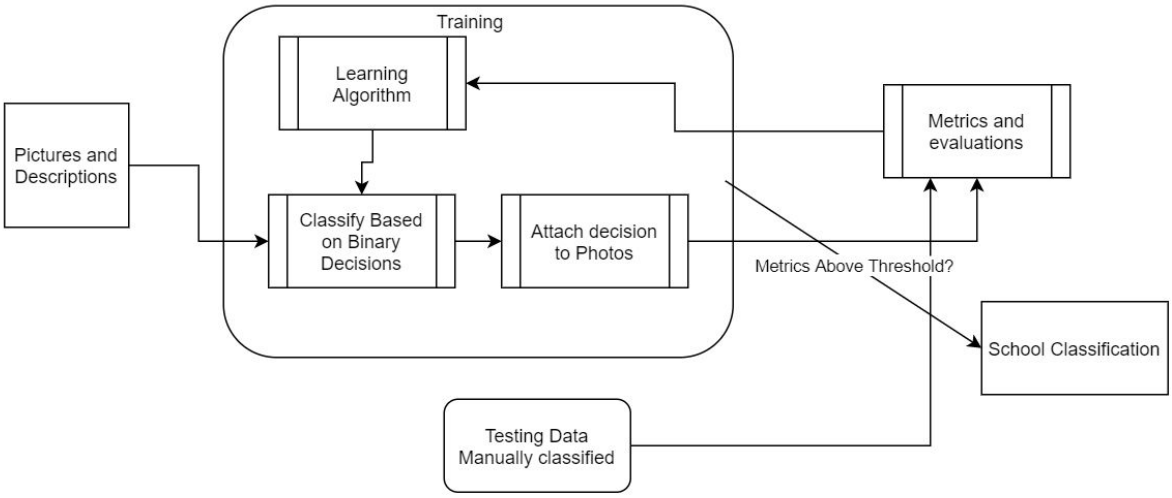


Figure 2: Decision Tree Training Model



APIs and libraries

Data Pipeline

OneDrive developer tools: https://github.com/OneDrive/onedrive-sdk-python
This API was used to automate downloading images into AWS for training and testing purposes.
Libraries used: Urllib, pandas

Binary classification model

Libraries used: Tensorflow, Keras, pandas, sklearn, numpy
These are the most common libraries for any machine learning related tasks. Tensorflow and Keras were used for model architecture, training and testing. Pandas was used for accessing and formatting the provided labels and classes. Sklearn is used for preparing data for training and testing, for example performing data splits. Numpy is used for some general data processing.

Heat Map

Libraries used: OpenCV
Same ones used for binary classification models with the addition of opencv for image formatting.
Github Repository: https://github.com/ramprs/grad-cam/
Grad-CAM Demo: https://www.youtube.com/watch?v=COjUB9Izk6E&feature=youtu.be

Performance metrics

All the required metrics can be accessed from Tensorflow or sklearn and these libraries allow for the definition of new metrics if need be.

Improve Data Augmentation

We used Keras, an open-source library that provides a Python interface for artificial neural networks and makes it easy to create, train, and evaluate models. It is built on top of the Tensorflow library.

Fine Tuning Model

Our team suggests using AWS SageMaker to perform the final model tuning. Given that our project is deployed in the AWS ecosystem, SageMaker is a natural decision as it will integrate seamlessly with the other components of the system. We never used SageMaker, but suggest its use in the future.

Knowledge representation issues

The main knowledge representation issue that is apparent to us right now is how the performance of the binary classification model on the multiclass based dataset. Using multiple binary classifiers may be less accurate at categorizing building-types overall, and could introduce unwanted complexity. We monitor model accuracy using the various evaluation metrics described to make sure this issue is addressed.

Implementation Details

Our system consists of various main components working independently, namely, we have binary classification models, an algorithm for highlighting important features, and a web scraper that gathers the data used for our algorithms. Below we describe the technologies that were used to create these components as well as some of the challenges we have faced while building them.

Technologies, Tools, Languages, Development Environments

All of the components in our system were hosted on AWS services. Each component utilized different technologies within the AWS suite, each of which are noted below.

Binary Classifier
Our binary classifier was adapted from the previous group's model and uses Python3's tensorflow packages. In particular, models are defined using the keras frontend for tensorflow. The model is trained from a keras data generator pipeline that performs the necessary preprocessing and augmentation of image data.

Grad-CAM
Another component that our system includes Grad-CAM which is an algorithm that is able to produce "visual explanations" for Convolutional Neural Networks (CNN) [5]. The original Grad-CAM system is written in the Lua programming language and uses Caffe deep learning framework as a way of inputting the CNN [6]. However, since our classification models are written in Keras we are using a Keras implementation of the Grad-CAM [7].

Data Augmentation
We used Python's Keras library for data augmentation. Keras was a great choice for data augmentation for three reasons. (1) The convolutional neural network model was already written using Keras, so we did not have to learn/include a new library to the code, (2) it provided the features we wanted natively, and (3) the images were randomly (with some constraints)

augmented in memory before being passed to the model, eliminating the need to save augmented images to disk, which would have made the size of the dataset much larger.

Web Scraping
For web scraping our toolset consisted of Python 3+ and Python libraries Selenium and Requests. Requests was used in order to log in to the main sida webpage and send POST and GET requests to retrieve building page links. Selenium was used to load each building web page in order to scrape image links and masonry data.

Prototype Functionality

Binary Classifier
Currently, our prototype for the binary classifier is in the form of multiple separate models. These models are based on the general architecture that the previous group implemented in terms of the input and output handling and model setup. We created multiple models that separately evaluate the buildings and print out the classifier dependent on what is expected. This made the training process easier and faster, while still meeting the requirements of the World Bank team.

Grad-CAM
The original Grad-CAM prototype can be observed in a demonstrative video the creators published [8] or in the figure below. The visual explanation produced for the inputted CNN is a heat-map of the features that contributed the most to the CNN's output. We adjusted the implementation of Grad-CAM to take in multiple inputs. This was used to understand our models to make training adjustments easier.

Grad-CAM applied to a cat-dog image classifier.

Data Augmentation
As images are passed to the model for training we were able to specify various data augmentations. We decided to use the following augmentations: zoom (± 20%), rotate (± 20%), brightness adjustment (± 40%), horizontal flip, and shear (± 20%). These adjustment ranges were selected somewhat arbitrarily. In the next week, we would like to investigate if the ranges can be optimized as hyperparameters. We hope that this assists in achieving maximum model accuracy.

Web Scraping

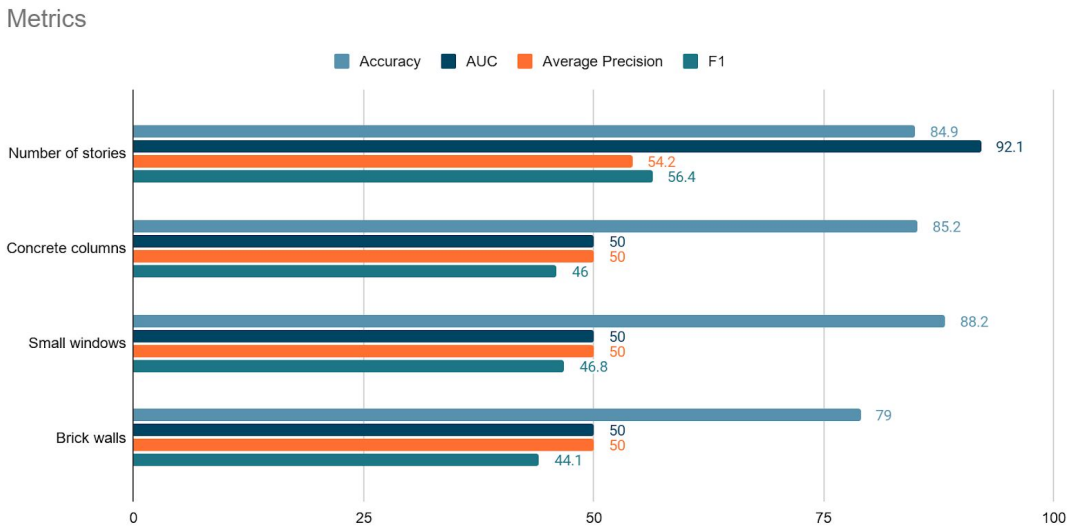
The Python scraper is broken into two main parts. The first piece of the scraper uses Python's request library and takes advantage of a filter, POST, and GET requests found on sidas website in order to pull links to all 17,000+ buildings that sida has images and other information about. After the links or all buildings are retrieved the second piece of the scraper is used.

The second piece of the scraper uses the selenium Python library. This library enables the scraper to be able to load javascript that is responsible for rendering images and other web page elements that are not present in the static webpage html. This part of the scraper will iterate over all of the previously retrieved building links and load each of their webpages in order to pull any found image links and masonry data pertaining to the given building.

Evaluation

Model Performance Evaluator

This component of the system measures the performance of the classification models by calculating accuracy, precision, recall, F1 score, and area under the curve. It takes in the classifications on the testing data made with the model and compares them to the classifications entered manually. Some of these measurements are built into frameworks like Keras or sklearn, and they will be more useful for evaluating binary classifications. As an additional matrix we have a script to get the Cohen Kappa score that measures the inter-rater reliability. This runs off of Sklearn reading in the labels from both the manual and model generated images. Currently the models training results in a score of less than .2 meaning they have a poor agreement.



Due to time constraints, we were only able to fully implement a few binary classifier models, with limited training. Our current data is not as accurate as the multiclass model from previous

groups, which could indicate that the binary method is inadequate for the task of building classification. However, our results demonstrated adequate model functionality as a baseline, and when supported by heatmap analysis and further training we expected the models would produce improved performance results. The datasets we trained with were very biased towards specific categories for some of the questions, and this can be resolved by using data augmentation and balancing the inputs to form a more even distribution of true/false labels. Future groups may be able to fully utilize our models using the additional Nepal data that we set up this quarter, or by incorporating a hyperparameter optimization tool like Sagemaker, though this could require extensive data reorganization on behalf of the World Bank.

Grad-CAM Evaluation

The evaluation of heatmap activations via the Grad-CAM algorithm is mostly performed observationally for each network used. When tested with single-input sample networks, the algorithm has produced reasonable output heatmaps, in-line with model accuracy. When used with the binary classifiers, heatmaps can be verified on a per-model basis by observing how the individual model accuracy corresponds to focus regions of the generated heatmaps. With high accuracy models that we have trained both the single and multi-image versions are able to produce reliable heat maps based on the training model.

Conclusion

The goal of our project was to assist the World Bank in analyzing and classifying school buildings, to improve their ability to determine building condition. We were provided building photos from the Kyrgyz Republic and Nepal, and were tasked with implementing various building classification models using tools provided by AWS and previous groups. In addition, we explored options for improving validation methods and increasing model explainability, including methods for generating activation heatmaps. While we were unable to create models for every classification category requested due to time constraints, we were able to implement much of the underlying functionality for these models, and improve the software structure for future groups. In the following sections, we outline the key aspects of our final implementation, its relevance to the field of AI, and some of the challenges we faced during this project.

Requirements

The requirements that we were given for the project were to create and refine a binary classification system for judging the structure and safety of school buildings in Nepal and Kyrgyz. Other tasks included the preparation of different data sources and setting up evaluation metrics for the models that we created. This involved scraping data from multiple sites in the form of both photos and the accompanying annotations for them in some cases.

Specifically, the requirements laid out were:

1. Prepare a set of training data using photos taken in Kyrgyz and Nepal.

2. Improve the data pipeline for OneDrive and implement a pipeline for ArcGIS that is capable of scraping new information, i.e. more photos and masonry information.
3. Separate the classifier models into a set of binary classifiers with a simplified logic tree. The binary classifiers requested were as follows, in descending order of importance.
    a. Number of stories
    b. Presence of concrete pillars
    c. Relatively small windows
    d. Brick walls
    e. PC wall panels
    f. Major portion of the walls are of earthen material
4. Set up model performance evaluations.
5. Improve data augmentation and fine tune the models as needed.

Additionally, towards the end of the project, we introduced the following requirements after discussion with the World Bank's team:

6. Improve the existing code base's structure, moving code out of single files as appropriate into a more modular design.
7. Add documentation for the existing code base and indicate what future work should focus on.

Out of the initial requirements, we fulfilled #1, #2, and #4 exactly as discussed. For #3, in the interest of time, we focused our attention on only a few of the models, which were approved by the World Bank team in advance. For #5, we performed some fine tuning of the models, but the existing data augmentation implementation was broken in the code base that we were provided, so only simple data augmentation was performed. For tasks #6 and #7, we reorganized previous teams' project structure, and provided additional documentation for all relevant scripts.

System Design and Implementation

For our project we had 4 main parts: the binary classifier, Grad-CAM, Data Augmentation and Web Scraping. We inherited a code base used by a project team from Winter 2020 that used Keras to perform the training for the model. In this section we will discuss the technical features and implementation issues we ran into with each section.

Binary Classifier

The binary classifier adapted from the previous sections model used tensor flow packages with a keras front end. Since the previous model was so vastly different from the binary classification system and documentation was lacking this ended up being a major hurdle for development. While different models were created most of them achieve very low accuracy. This problem was compounded by the issue of not being able to transfer files directly from the S3 buckets. This meant that other teams that wanted to work and train with the models were met with this barrier.

The classification team also had issues with the data provided and instances of AWS they were

allowed to use. The issue with the data was not having access to a well organized dataset, with very few of the needed photos being ready on the provided onedrive. The AWS issue meant that large training instances could not be spun up and any training that was done would have to be evaluated after a very long training period. This led to slow development times and overall lack of progress.

Grad-CAM

The Grad-CAM implementation originally only allowed for one input. Since our model would always take in multiple inputs it had to be redesigned to take multiple pictures in as input. To achieve this the Grad-CAM team were able to load all the photos and model in but then make it run through each of the photos individually to show the heatmaps. It would still be able to feed all the pictures into the model as required but only provided the heatmap for the one picture being addressed at a time. Since the models from the binary classifier were of very low accuracy this did not prove to be very helpful but tests on other models showed that it could eventually be used to provide better feedback to trainers.

Data Augmentation

Data augmentation is done through simple keras augmentations defined for a data generator. The data augmentation in place when the project began already contained the preprocessing that would be needed for the data used to train the model. Because of this, the data augmentation step should have largely been taken care of by using this provided pipeline; however, we found issues with correctness of the augmentation that was provided. Thus, we would have needed to restart how the augmentation was set up instead of simply tweaking it. We decided to only perform simple augmentation for the sake of time.

Web Scraping

This team was tasked with getting the data from the Kyrgyz dataset provided by the World Bank. This turned out to be a lot more labor intensive task than first thought taking most of the quarter to scrape all of the images. Each of the images had to be loaded into their respective URLs then have the different annotations attached to them. After all this was done a separate scraping script went though and downloaded each of the images. This data has all been collected now but was not able to be put to use for training the model this quarter.

Relevance for AI

This project used a deep learning classification system to capture key features of school buildings from photographs. Using Keras to train models, the project accurately labeled a set of binary classification questions for each photo. This allows for automating the process of the user manually entering answers to each binary question. The intent of the binary classification method was to simplify the existing multi-class system, in an attempt to reduce complexity and improve model performance. Our results demonstrate that this was not the case, but this could

be due to other factors, such as those listed in the next section.

Lessons Learned and Future Work

Our team struggled to adequately train the data. The documentation and code structure from the last team made it difficult to effectively adapt their work to meet the needs of our sponsor. We communicated this issue with our sponsor, and we made an effort to better document and organize all work that has been completed up to this point. A key takeaway from this experience is the importance of quality code structure, file organization, and comments when working on projects which will eventually be referenced or passed on to future teams.

Now that we have scraped the Nepal building image data, created a binary classification system, and adapted GradCAM to work with future trained models, the next step is to train models which can classify high priority features such as concrete columns and brick walls in image data. In addition, models which can classify lower priority features such as PVC panels and earthing material in walls should also be trained. We advise the World Bank and the DxHub to turn these deliverables into a year-long senior project or master's thesis. At this point of the project, it would be impractical to try and accomplish these deliverables in a single quarter because it takes a lot of time just to become familiar with AWS, the datasets, and the previous teams' work.

References

[1] "OVERVIEW," *Global Program for Safer Schools (GPSS)*. [Online]. Available: https://gpss.worldbank.org/en/glosi/overview. [Accessed: 19-Jan-2021].

[2] M. Jiang et al., "Making Schools Safer and Resilient at Scale in Kyrgyzstan." https://drive.google.com/file/d/1fLE8G4m7tjpMRoAIs32pz0MmTpqIdJAW/view

[3] Zhineng Chen, Shanshan Ai, and Caiyan Jia. 2019. Structure-Aware Deep Learning for Product Image Classification. ACM Trans. Multimedia Comput. Commun. Appl. 15, 1s, Article 4 (February 2019), 20 pages. DOI:https://doi.org/10.1145/3231742

[4] D. Zhang *et al*., "Knowledge Graph-Based Image Classification Refinement," in *IEEE Access*, vol. 7, pp. 57678-57690, 2019, doi: 10.1109/ACCESS.2019.2912627.

[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," Int J Comput Vis, vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: 10.1007/s11263-019-01228-7.

[6] "Caffe | Deep Learning Framework." https://caffe.berkeleyvision.org/ (accessed Feb. 21, 2021).

[7] "keras-team/keras-io," GitHub. https://github.com/keras-team/keras-io (accessed Feb. 21, 2021).

[8] “Grad-CAM - YouTube.” https://www.youtube.com/watch?v=COjUB9Izk6E&feature=youtu.be (accessed Feb. 21, 2021).