Patch-based classification for building damage assessment using satellite imagery of natural disasters

DSE Capstone

Mentor: Professor Jie Wei
Student: Ivan Miller
Fall 2022

Problem Statement & Motivation:

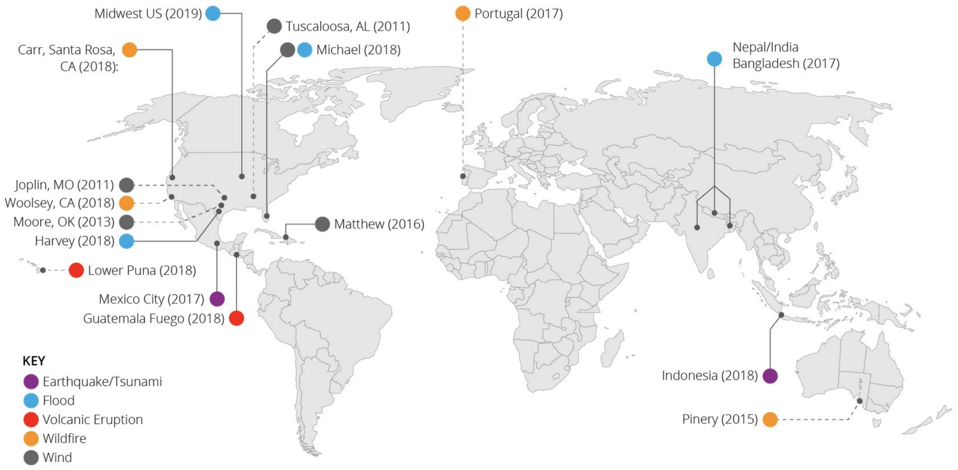
Damage Assessment with Known Coordinates

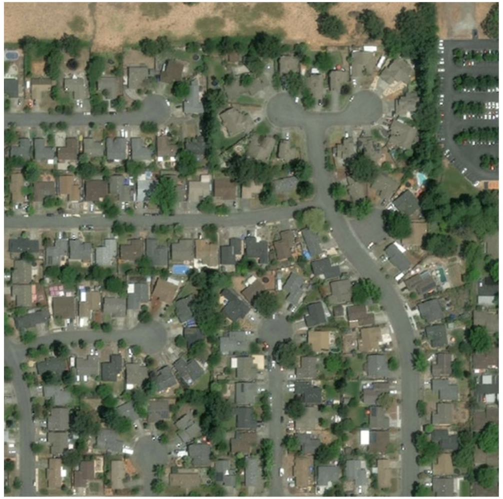
Create a machine learning classifier for building damage assessment using satellite images in order to:

- Reduce time necessary to produce a high quality result
- Increase accuracy of automated assessment above 80% benchmark
- Scale up - perform damage assessment at scale

Data Overview: xBD dataset

- xView2 competition by Defence Innovation Unit (2019)
- Images from Maxar's Open Data Program
- Total area of over 17,000 mi²
- 850,000 building polygons
- 19 natural disasters of 6 different types





EDA and Data Preprocessing

- Image conversion
- Metadata tracking for mapping of patches to original images:
  - Disasters
  - Damage types
  - Disaster types
- Pre & Post patches
  - Tested 150x, 100x, 64px, and **50px**
- “Quality control” for Dark / Bright images:
  - Satellite shifts
  - Clouds

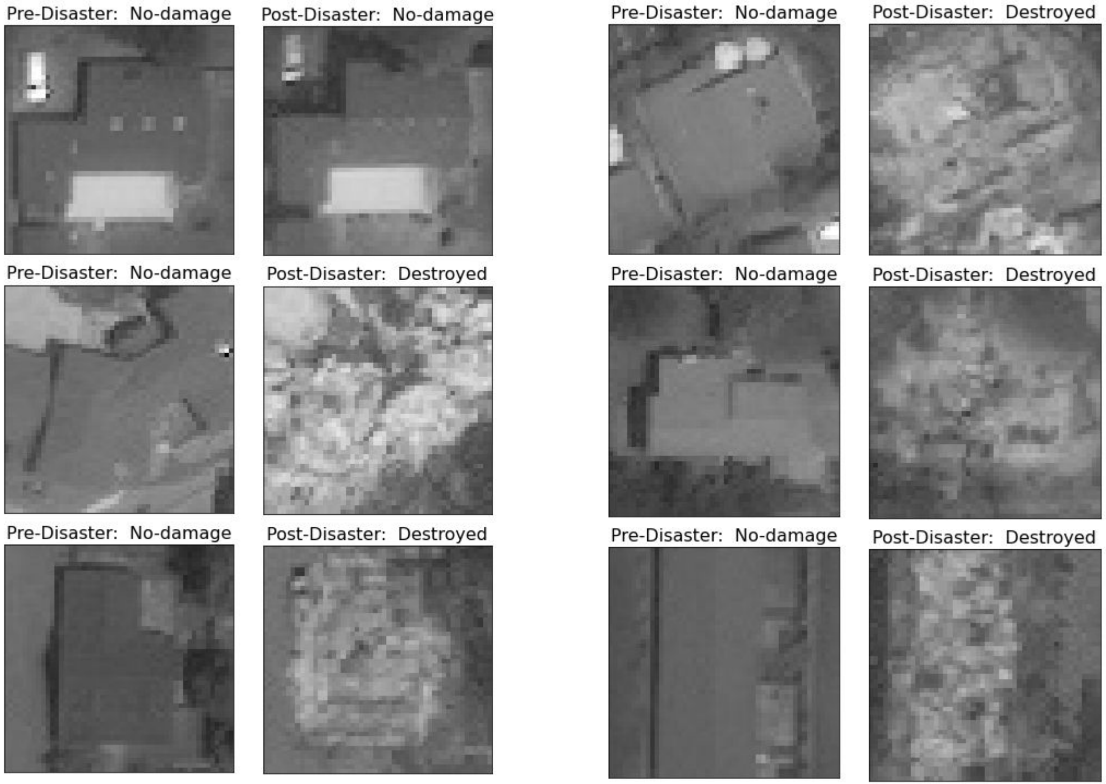
PRE	POST
42%	58%

Damaged	No Damage
44%	56%

Over 220k total 50x50px patches were created and split into three parts to create train, test, and validation datasets.

Example Pre-Post Disaster Patches from xBD Dataset

50x50 patches from Santa Rosa wildfire, 2017



Final version of a 50 x 50 px mage patch from the original satellite images.

Note a drastic difference in visual representation between pre and post-event images that led to the highest validation accuracy of 95.97% among all disaster types.
Over 220k total patches created.

Methods: Deep Learning with PyTorch

Results of fine -tuning of 3 deep learning models with different architecture on patches from xBD dataset:

Model	F1 Score	Precision	Recall	Validation Accuracy	Training Time*	Number of Epochs	Model Size	Inference Time
MobileNet_v3_40	80.41%	79.02%	81.85%	89.35%	427m 21s	40	22.1Mb	5m 33s
ResNet50_40	75.94%	79.24%	72.90%	87.70%	426m 31s	40	94.4Mb	4m 44s
ResNet50_10	72.87%	78.79%	67.78%	86.56%	60m 27s	10	94.4Mb	4m 36s
MobileNet_v3_10	72.45%	78.23%	67.47%	86.62%	58m 59s	10	22.1Mb	5m 26s
ViT l 32_10	64.90%	61.00%	69.34%	74.31%	101m 26s	10	1.23Gb	13m 21s

* All models were trained on NVIDIA A100 Tensor Core GPU (200GiB RAM, 30 vCPUs)

Augmentation with PyTorch and Albumentations library:

RGBShift: Randomly shift values for each channel of the RGB input

RandomBrightnessContrast: Change brightness and contrast of the input image.

MultiplicativeNoise: Multiply image to random number or array of numbers

HueSaturationValue: Randomly change hue, saturation and value of the input image

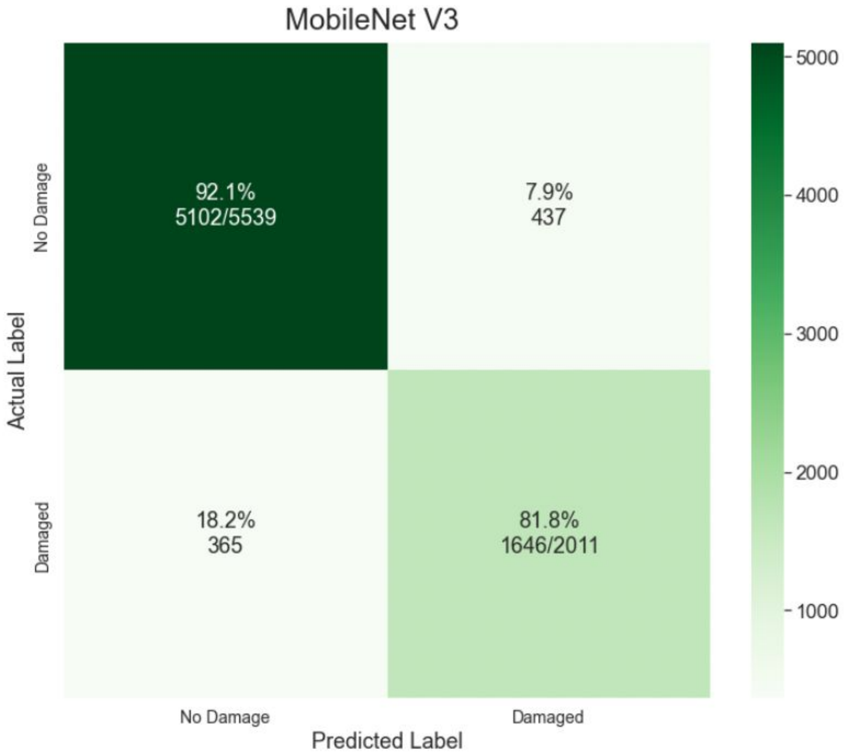
Methods: Deep Learning with PyTorch

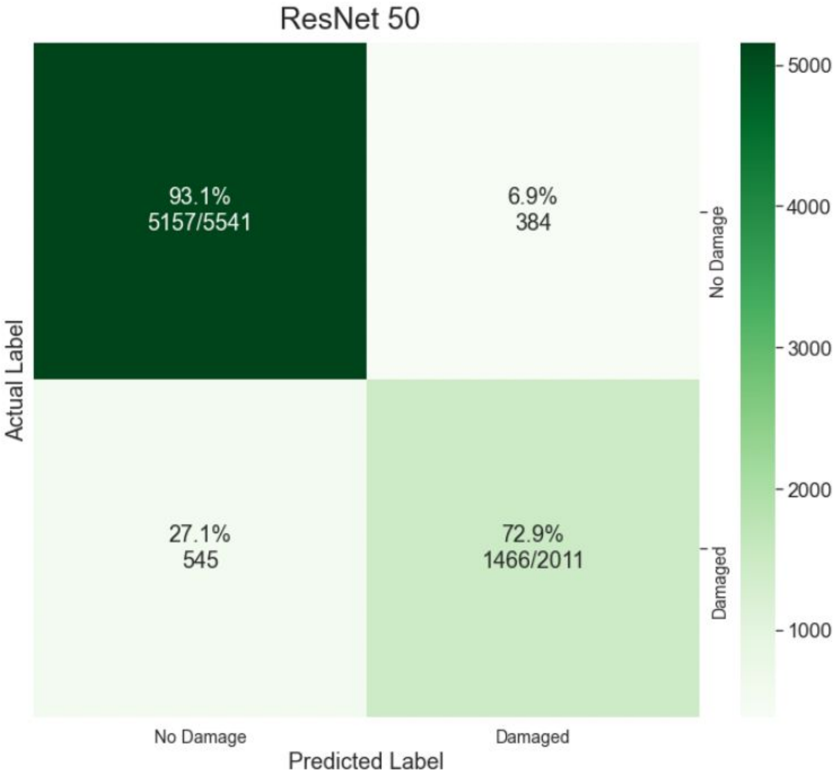
Results of fine -tuning of 3 deep learning models with different architecture on patches from xBD dataset:

Model	F1 Score	Precision	Recall	Validation Accuracy	Training Time*
MobileNet_v3_40	80.41%	79.02%	81.85%	89.35%	427m 21s
ResNet50_40	75.94%	79.24%	72.90%	87.70%	426m 31s
ResNet50_10	72.87%	78.79%	67.78%	86.56%	60m 27s
MobileNet_v3_10	72.45%	78.23%	67.47%	86.62%	58m 59s
ViT l 32_10	64.90%	61.00%	69.34%	74.31%	101m 26s

* All models were trained on NVIDIA A100 Tensor Core GPU (200GiB RAM, 30 vCPUs)

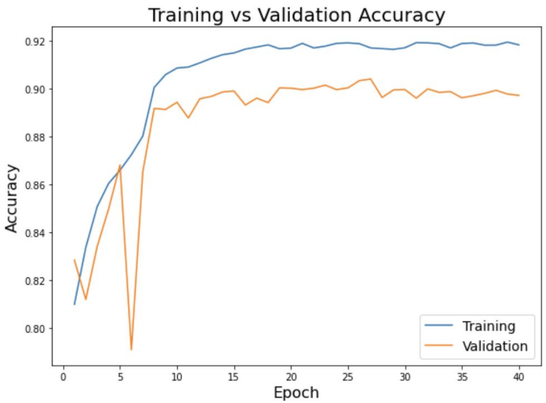
Confusion Matrices for MobileNet vs ResNet 50

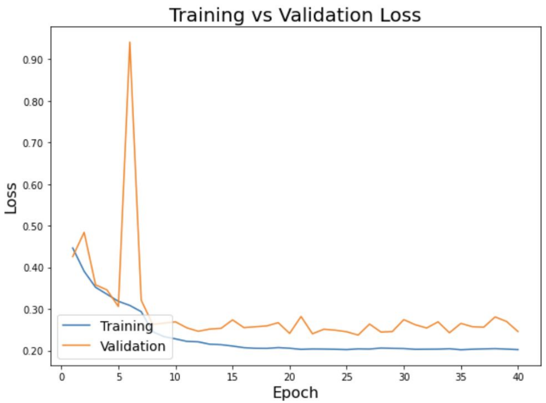




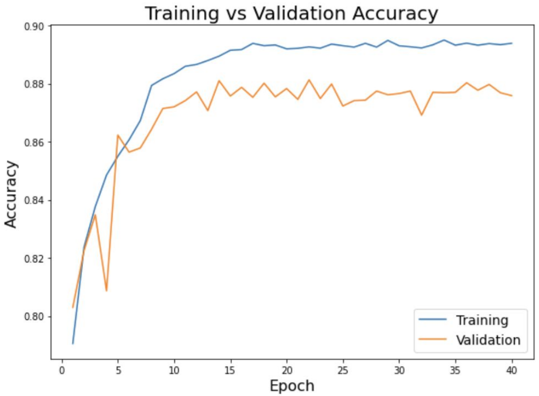
Accuracy vs Loss after training for 40 epochs

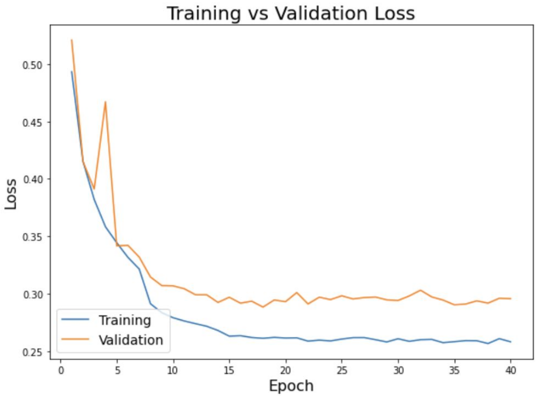
MobileNet V3





ResNet50

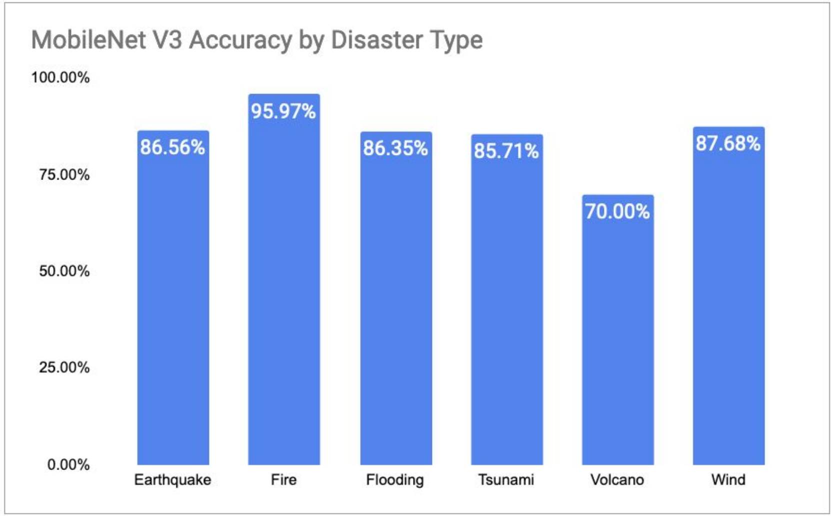


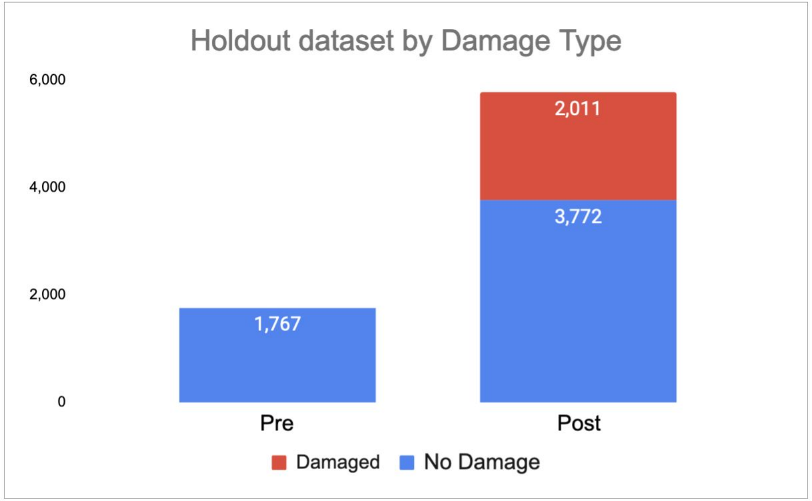


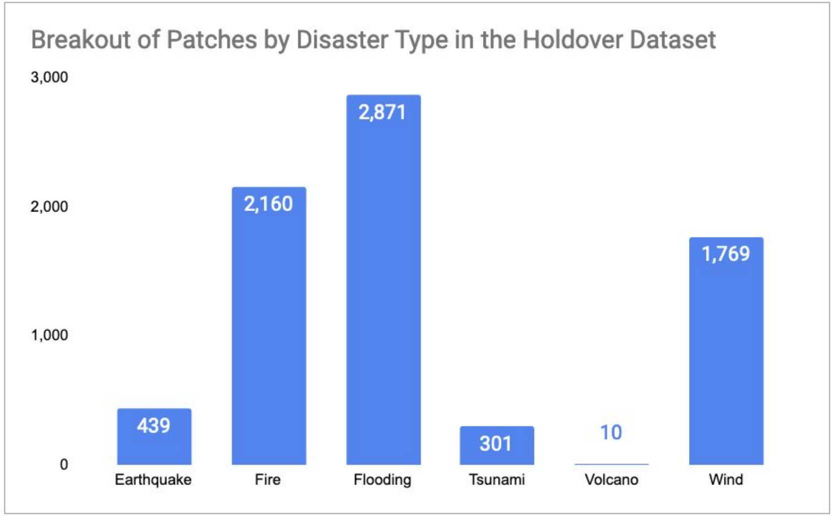
	MobileNet	ResNet50
F1 Score	80.41%	75.94%
Precision	79.02%	79.24%
Recall	81.85%	72.90%

Analysis of Misclassified Patches

When looking at at a breakout of the classification results by the type of disaster, we see that, despite imbalance of the holdover dataset, the model still performs quite well (outside of extreme examples, see volcano disaster type).

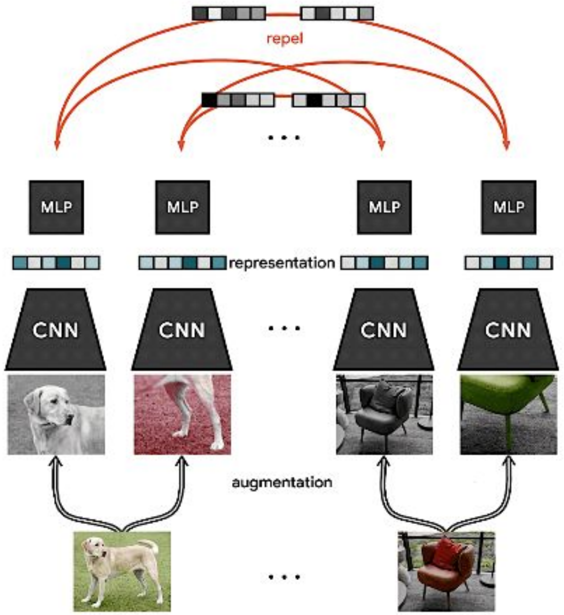


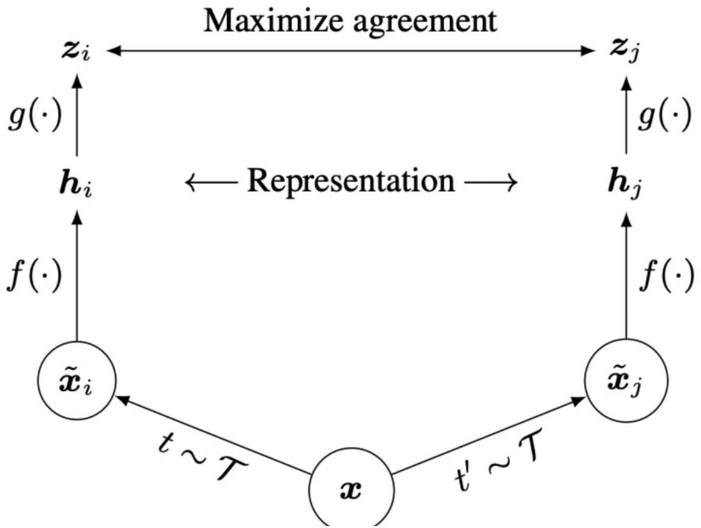




Other Attempted Techniques

Contrastive Learning of Visual Representations using SimCLRv2 from Google Research team





Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \tau$ *and* $t' \sim \tau$) and applied to each data example.

!! Requires larger batch sizes (128 vs 4)!!

A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss function.

Future Work

1. Application for Inference on unseen images [in progress]

   Objectives:

   - Automatically create a patch of the uploaded image
   - Run the patch through trained MobileNet V3 model
   - Display the result

2. Finalize SimCLRv2 implementation with PyTorch using a more powerful server to account for larger batch sizes.
3. Create a hybrid model with trained CNN and Non-Deep Learning Methods working in tandem to improve prediction. Potential options include:
   - K-Means Clustering
   - XGBoost

Related Work

Generalization gap and ways of improving out-of-domain performance

Benson, V.; Ecker, A. "**Assessing out-of-domain generalization for robust building damage detection**". Published at NeurIPS 2020 Workshop on Artificial Intelligence for Humanitarian Assistance and Disaster Response (AI+HADR 2020). https://arxiv.org/pdf/2011.10328

Overview of loss functions

S. Jadon, “**A survey of loss functions for semantic segmentation**”. 2020 IEEE International Conference on Computational Intelligence in Bioinformatics and Computational Biology.https://doi.org/10.48550/arXiv.2006.14822

Overview of Vision Transformer

A. Dosovitskiy et al **“An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”** https://doi.org/10.48550/arXiv.2010.11929

Contrastive Learning

T.Chen et al **“Big Self-Supervised Models are Strong Semi-Supervised Learners”** https://arxiv.org/abs/2006.10029

Design and Plan

Before 09/01/2022	• Meet Professor Wei to finalize the topic and go over the requirements and scope of the project. • Get familiar with the topic and read recently published papers. • Start learning key concepts of PyTorch.	10/13 - 10/26/2022	• Determined the combination of model parameters that yielded best results to date
09/01 - 09/14/2022	• Download xBD dataset and focus on preprocessing the data. • Continue learning PyTorch. Implement a standard model for classification task. • Explore different areas of the project to finalize the roadmap.	10/27 - 11/09/2022	• **Fine-tuning** of the “best” model, focus on augmentation to improve generalization • **Inference:** ○ xBD patches • **Contrastive Learning**
09/15 - 09/28/2022	• Continue learning PyTorch. Experiment with ResNet and MobileNet (V2 and V3) models. • Use pre-trained model for image classification.	11/10 - 11/23/2022	• Combine all modalities of the data and continue fine-tuning the model. • Tested the model on unseen data - publicly available images
09/29 - 10/12/2022	• Start training the model on xBD dataset. • Establish a baseline for performance and experiment with optimizers and loss functions.	11/24 - 12/07/2022	• Collected all results, created presentation, and finalized the report.

Thank you!