"What makes Amherst look like Amherst?" Applying Class Activation Mappings on ResNet for City Feature Classification

Mehak Virender Nargotra
University of Massachusetts, Amherst
mnargotra@umass.edu

Preston Yee
University of Massachusetts, Amherst
pyee@umass.edu

GitHub: github.com/682_Computer_Vision_Amherst_Project

1. Introduction

Every city possesses some unique visual identities that define their character and set them apart from one another. In this research project, our objective is to investigate the unique visual characteristics that define Amherst's identity and set it apart from other cities. By utilizing modern computer vision and deep learning techniques, like Fully-Connected Neural Networks and Class Activation Mapping (CAM), our aim is to attain a thorough understanding of the distinct visual attributes that render Amherst unique when compared to other cities.

The motivation behind our research lies in the recognition that a city's visual identity holds significant implications for urban planning, cultural preservation, and marketing. Through an in-depth exploration of the visual components that shape Amherst's identity, alongside comparative analysis with other cities, our objective is to offer those valuable insights. These insights can serve as guidance for urban development decisions and contribute to the enhancement of Amherst's public image. This research bridges the fields of urban studies, computer vision, and deep learning, offering a multidisciplinary approach to understanding the visual character of Amherst.

2. Problem Statement

The objective of this research project is to undertake a comprehensive analysis of the distinct visual features that contribute to the unique visual identity of Amherst when juxtaposed with other urban environments. Our particular interest lies in the utilization of sophisticated computer vision and deep learning methodologies, specifically Fully-Connected ResNet Classification and various CAMs, to thoroughly explore and interpret these distinctive attributes. Through this study, we strive to gain a profound understanding of what sets Amherst apart in its visual representation compared to a diverse array of other cities. This research is motivated by the need to provide valuable insights into the city's unique visual identity, with potential implications for urban planning, cultural preservation, and effective marketing strategies.

As we advance through data preparation and model training, our anticipation is twofold. First, we expect the model to adeptly identify unique visual attributes that distinguish Amherst from other locales and precisely classify images based on their origin. Second, we apply CAM to the model and validation set of Amherst images to pinpoint and emphasize key visual features within Amherst's images, enhancing our understanding of its unique identity.

3. Related Works

In our quest to unveil the distinctive visual identity of Amherst, our project draws inspiration from studies such as 'What Makes Paris Look Like Paris' by Carl Doersch et al [1]. As we explore the unsupervised discovery of frequently occurring features in Amherst's visual landscape, we go beyond the iconic landmarks, into the city's distinct features. To initiate our journey, we focused on collecting the dataset for the city of Amherst, which was challenging to find for our specific purpose. When examining the paper 'Object Retrieval with Large Vocabularies and Fast Spatial Matching' [6], the main purpose behind gathering the 5062 images of Oxford Buildings was to search for specific Oxford landmarks on Flickr. However, a challenge with platforms like Flickr and other consumer photo-sharing websites for geographical tasks is the strong data bias towards famous landmarks.

To address this bias and provide a more uniform sampling of the geographical space, another approach, as followed by the 'What Makes Paris Look Like Paris' paper [1], involved collecting datasets using Google Street View—a vast database of street-level imagery captured as panoramas using specially designed vehicles. However, due to the limitations associated with using copyrighted data, making use of Google Street View may lead to facing legal obstacles. Its coverage may also prioritize more populated and tourist-heavy locations, introducing potential biases as well.

This led us to explore walking tour YouTube videos, offering a glimpse into the essence of the city. While these videos were used to capture the Non-Amherst dataset, comprising 19 cities (i.e., 5,000 images per city), we personally captured 5,000 images for the city of Amherst to minimize noise and extracted another 5,000 from YouTube in the same way as Non-Amherst.

Inspired by the approach outlined in Doersch et al's paper [1], the main idea focused on identifying rare and frequently occurring patterns in small image segments, specifically concentrating on the windows of buildings in Paris, resulting in the formation of clusters. They clustered with KNNs and employed Linear SVM to compare these clustered images with images from other geographical regions.

Traditional methods like KNN and Linear SVM work well with small, specifically-pointed datasets, but are inadequate for our larger, more busy datasets. It wasn't just a matter of dataset size; these methods also demanded manual feature engineering, which is exceedingly tedious. The results were highly dependent on the choice of hyperparameters and regularizations, prompting us to explore deep learning, employing fully connected neural networks, and techniques such as Class Activation Mapping (CAMs), Grad CAMs, etc. This comparison was aptly highlighted in the paper "ResNet: A Novel Method for More Accurate Segmentation and Classification of Remote Sensing Images Compared to KNN" [7], providing insights into the accuracies of both classifiers. The paper, "Building Instance Classification Using Street View Images" [4], uses a similar approach of using CNNs for the same purpose. The difference lies in that, instead of identifying the unique features of any city, they focus on a smaller portion of the city and classify buildings such as church, garage, house, office, etc., using a dataset of images from Flickr.

CAMs demonstrate remarkable localization ability for CNNs with a Global Average Pooling (GAP) layer, emphasizing interpretability and model trust assessment. This was evident in the research work on "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization" [8], where the idea of CAMs came into play, leading to a deeper exploration of the information retrieved using Grad-CAMs. Grad-CAMs further improved results, highlighting how we can enhance gradient-based sensitivity maps by averaging sensitivity maps from random samples in a neighborhood of an input [5].

Another example of CAMs for object localization is "Learning Deep Features for Discriminative Localization" [10], which distinguishes between multiple objects; each object is its own image. However, ours contain multiple objects and features for any given image.

4. Methodology/Technical Approach

Our approach to the problem can be divided into three sections; Dataset Searching and Collection, Code Implementation, and Evaluating Results.

4.1. Dataset Searching and Collection

Our research relies on datasets comprising of 10,000 images from Amherst and 5,000 images from at least 19 global cities, for a total of 20 cities. Amherst-specific datasets were to be collected by capturing video footage in Amherst and extracting relevant images. Our non-Amherst image datasets were sourced online from existing datasets. Additionally, we reached out to acquaintances outside Amherst to record video footage for us to extract images.

Searching online for existing datasets proved to be a challenge, as there was a scarcity of the datasets that met our criteria. We found only two valid datasets online that contained photos of outside cityscapes and area that we could use for training. As for video footage, we used a smartphone and drove around in a car around the area of Amherst, recording video footage as we go. However, even with the footage, we were only being able to extract about 5,000 images. To extract the obtain the remaining needed images for other cities, we turned to YouTube and download 1-hour long walking tours of various cities using YouTube-dl, and used FFmpeg commands to extract the frames of images from those videos.

We were able to collect about 5,000-10,000 images for 20 cities, which include: Amherst, MA, USA; Amsterdam, Netherlands; Pune, India; Singapore; Austin, TX, USA; Bali, Indonesia; Bergen, Norway; Boston, MA, USA; Los Angeles, CA, USA; Las Vegas, NV, USA; Lima, Peru; Manhattan, NYC, USA; Margao City, India; Paris, France; Sofia, Bulgaria; Tokyo, Japan; Venice, Italy; Würzburg, Germany; University of Salford Manchester, England; and Zürich, Switzerland (1). We used a Macbook M2 Max computer to store these large datasets, since it had 32GB of memory storage.

4.2. Code Implementation

For the coding section, we used Python and PyTorch packages to implement our solution. Our coding has been broken down into three components: Data Loaders, ResNet Classifier, and Class Activation Maps.

4.2.1 Data Loaders

We extracted the images stored in our files and splitted the cities by two types of classifications: binary and non-binary.

Binary classification was used to cluster all images from non-Amherst cities into one label called "Non-Amherst", while Amherst-related images are grouped under the label "Amherst". For non-binary classification, each city was assigned its own label by name. We did this to experiment between two different loss functions for training; binary cross entropy loss and cross entropy loss. The data was split into 80% training set and 20% validation set. We decided on not having a testing set, as we evaluated the program's performance and accuracy through qualitative analysis.

For loading the data, we referenced an article that utilizes Torch DataLoaders by Dilith Jayakody [3] as our starting point on how to best load and preprocess our dataset.

(a) Amherst, MA, USA

(b) Amsterdam, Netherlands

(c) Bergen, Norway

(d) Venice, Italy

(e) Würzburg, Germany

(f) Austin, TX, USA

Figure 1. Sample images of 6 out of the 20 cities collected for dataset

We used Torch DataLoaders and Torchvision transformers to load our dataset and apply data augmentation, respectively. Data augmentation includes resizing each image to 64x64 size, RandomHorizontalFlip to horizontally flip half the images, RandomRotation of 10, 20 and 40 degrees, and normalization to take into account various resolutions, angles of the images, as well as the coloration and shading (2). We then splitted the data into training and validation, and applied the Torchvision DataLoader class onto both training and validation partitions in batch sizes of 32, which we found to be large enough to train without having to run so many batches during training time.

```
transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomRotation(degrees=10),
    transforms.RandomRotation(degrees=20),
    transforms.RandomRotation(degrees=40),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

Figure 2. DataLoader for Preprocessing City Dataset

4.2.2 ResNet Classifier

For classification, we used a ResNet50 model with pretrained weights to train and adapt to our custom dataset. Subsequently, we trained the dataset using a 2-Layer Fully-Connected Neural Network. The neural network consists of a linear layer, ReLU, another linear layer, and a sigmoid function (3). When training these classifiers, we used BinaryCrossEntropyLoss (BCELoss) and CrossEntropyLoss as our loss functions. BCELoss wwas later applied for the case where our labels were binary ("Amherst" and "Non-Amherst"), while CrossEntropyLoss was for cases where our labels were non-binary ("Amherst", "Venice", "Austin", etc.).

```
class FCModel(torch.nn.Module):
    def __init__(self, input_dim, num_classes=2):
        super(FCModel, self).__init__()

        self.linear1 = torch.nn.Linear(input_dim, 256)
        self.relu = torch.nn.ReLU()
        self.linear2 = torch.nn.Linear(256, num_classes)
        self.softmax = torch.nn.Sigmoid()

    def forward(self, x):
        x = self.linear1(x)
        x = self.relu(x)
        x = self.linear2(x)
        x = self.softmax(x)
        return x
```

Figure 3. Fully Connected Model for ResNet Classifier

We trained over the course of three primary hyperparameters: two optimizers (Stochastic Gradient Descent (SGD) and Adam), two dataset partitions (Binary and Non-binary), and two maximum training epochs (5 epochs and 10 epochs). We set the learning rate to 0.001 since we found that it allows the classifier to train on the data quickly and converge the training and validation to an optimal loss and accuracy through multiple rounds of training and fine-tuning.

Once the training was complete, we saved each model as a .h5 file for later use in the Class Activation Mappings (CAMs). We also generated charts to visualize the training and validation loss and accuracy vs. epochs over the course of the training process for each model.

4.2.3 Class Activation Mappings

For class activation mapping, we referenced an article and a TorchCAM GitHub repository [2, 9] to help build our CAMs. We built three CAMs to run our trained modapplied it to 100 randomly sampled Amherst-specific images from the validation dataset, and applied a heatmap on top of those images to see what the model predicts or determines were the most distinctive Amherst-related features in a given validation image.

We extracted our saved models from training to then run on the CAM, along with a set of validation images to run our model. Images from validation dataset were preprocessed through a Torchvision Transformer for resizing and normalization to help CAMs generate best results, since the training dataset was also preprocessed through transformers before going into ResNet classifier.

```
input_tensor = test_transforms(img.type(torch.FloatTensor))
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
input_tensor = input_tensor.to(device)
if cam_type == 1:
    with SmoothGradCAMpp(self.model, 'layer4') as cam_extractor:
        out = self.model(input_tensor.unsqueeze(0))
        self.activation_map_list.append(cam_extractor(out.squeeze(0).argmax().item(), out))
elif cam_type == 2:
    with GradCAMpp(self.model, 'layer4') as cam_extractor:
        out = self.model(input_tensor.unsqueeze(0))
        self.activation_map_list.append(cam_extractor(out.squeeze(0).argmax().item(), out))
else:
    with CAM(self.model, 'layer4') as cam_extractor:
        out = self.model(input_tensor.unsqueeze(0))
        self.activation_map_list.append(cam_extractor(out.squeeze(0).argmax().item(), out))
```

Figure 4. Running CAM, GradCAM and SmoothGradCAM on dataset

After completing the implementation, we test ran through various combinations of optimizers, epochs and data partitions and fine-tuned our models. Validation images with their heatmaps from CAMs were saved into individual folders locally to allow us to evaluate.

4.3. Evaluating Results

For the evaluation phase, we conducted a qualitative analysis ourselves on the data to assess the accuracy of our program using the validation set. For each of the 100 heatmap-generated validation images from each CAM and hyperparameter combination, we determined that an image has "passed" the test if the heatmap covers an area or feature on the image that is distinctive to Amherst, such as buildings, landmarks, or unique designs (e.g.: unique pattern of a walkway or architecture of a building). An image was labeled as "failed" if it was unable to identify any unique features, or if it highlighted features that are non-distinctive to Amherst, such as green grass, white and gray clouds or blue skies.

5. Results

For the results, we trained a total of 8 models across three parameters: optimizers (Adam, SGD), number of epochs (5, 10), and data partitions between city datasets (Binary, Non-binary). With these eight models, we generated three types of CAM results each; CAM, GradCAM and SmoothGradCAM. For each of these CAM types for any given model, we randomly selected 100 Amherst-related images from the validation dataset and ran them through the CAM and trained model to produce heatmaps that would best highlight the key features in the image that make Amherst most distinct (8, 7). We also generated training and validation loss and accuracy graphs to determine the performance of our classifier (5, 6). In these figures, we showed an example of a CAM heatmap correctly applied to a validation image of Amherst based on these hyperparameters, followed by an accuracy of how many out of the 100 randomly selected validation images did the CAM correctly identify as unique features.

6. Evaluation

With our loss and accuracy graphs and CAM heatmap images generated, we began our evaluation. For the losses and accuracy, we evaluated the graphs based on whether the loss is converging towards zero (0) as possible with each epoch, and whether accuracy is converging towards one (1) with each epoch (5, 6). As for the heatmap results, we evaluated through qualitative analysis by viewing each of the 100 images per CAM for every model combination trained (8, 7).

6.1. Loss/Accuracy Graphs

From our loss graphs generated by our models (5), we found that all graphs were consistent in decreasing sharply after the first epoch, then continued to decrease in a more gradual rate as they continued down the remaining epochs. We saw that the training loss (green) and validation loss (blue) have converged towards each other as they approach an optimal loss. We also observed that for binary loss, the loss converged at around 0.01 for both SGD and Adam optimizers and 5 and 10 epochs. As for non-binary loss, the loss converged at around 2.1. However, for both binary and non-binary, we noticed that models that ran with Adam optimizer on 5 epochs models had slightly higher losses than other models with different combinations of optimizer and epochs, and seemed to yet reach an optimal loss (5c, 5g).

As for accuracy graphs (6), they too had been consistent in increasing in accuracy over the course of the epochs for all models. We also saw the training accuracy (green) is converging with the validation accuracy (blue). All accuracies are aggressively increasing at the first epoch, then continued to increase at a more gradual rates, and finally plateaued between 0.995 and 1.0 accuracy. Similar to the loss graph, models of both binary and non-binary partitions running Adam optimizer over 5 epochs had accuracy graphs that haven't reached their optimal loss (6c, 6g).

6.2. CAM Heatmaps

For our CAM heatmap results, we counted 75% (6/8) of the models showed that normal CAMs was able to correctly identify unique Amherst features better than GradCAM or SmoothGradCAM. For all of the models where normal CAM performed the best, with SmoothGradCAM following as seconds best, and finally GradCAM. From some of the example results shown, normal CAM generalized the best when it came to covering or identifying whole buildings or environments that make Amherst unique, such as the tall Du Bois library building and the campus layout, while not highlighting non-distinct areas like the blue sky.

SmoothGradCAM picked up on more nuanced and specific patterns in certain part of Amherst images, such as the triangular roof of one of the UMass buildings (8f), the positioning and orientation of the windows of a building (8l), and the structure of very unique architectures (8i). Of the 8 trained models, 75% (6/8) of the models had SmoothGradCAM as the second best performing CAM, and the remaining 25% (2/8) as the best, specifically, Non-binary data parition with SGD optimizer (8o, 7c).

For GradCAM, although not as strong in identifying unique features in Amherst images compared to the other two CAMs, under certain combinations of data partition and optimizer like Binary and Adam, or Non-binary and SGD, was able to also identify small unique details within certain objects in like the small windows of a building (8k), and the unique architecture of the Du Bois library in UMass Amherst (8h).

In correctly identifying features that are unique to Amherst for 100 validation images per model, we found that training a binary data partition (or Binary Cross Entropy Loss) with Adam optimizer training over 5 epochs, followed by running a normal CAM, has the highest accuracy of 68% (68/100).

6.3. Outliers

From our results, there were two models that showed unique results that is not as commonly reflected on other models; (Non-binary, Adam, 5 epochs), and (Non-binary, SGD, 5 epochs). For Non-binary, Adam, 5 epochs, normal CAM was able to highlight the general buildings and some streets that make Amherst unique (in this case, it was able to identify the garage building and one of the laboratories in University of Massachusetts, Amherst) (7d).

However, GradCAM and SmoothGradCAM were not able to identify any significant features about Amherst, even though it transformed the data the same way as other models and CAMs. A possible cause for this is because the model did not completely reached optimal loss and accuracy at the set epochs, according to the graphs of their training and validation for both 5 and 10 epochs (5g, 6g). For the case of 10 epochs (5h), we saw an improvement in terms of the CAM results. Unfortunately, the improvement in heatmap results did not improve the overall accuracy in our evaluation.

7. Limitations

In this project, there were a number of limitations that we had to face. These limits were:

7.1. Data Collection

Collecting images for cities was one of the most difficult challenges in the beginning of this project. Searching for any existing dataset of images of cities (specifically, buildings, streets, outside atmosphere) proved to be very difficult, since there was very little that existed in the first place. There were a couple cities we found online like the University of Salford Manchester and Zürich, but most datasets we found were either not enough (less than the required 5,000) or not useful for our purpose (e.g.: we found large datasets of Instagram photos of multiple cities, but they were not related to buildings or landscapes). The datasets we found had varying sizes and resolutions, which may have caused distortion between cities when resizing them in proprocessing and training.

We found the remaining of our needed datasets by extracting 1-hour walking tour videos of the cities from YouTube. However, we had to download the videos in a lower quality (480p) in order to fit them in our computer's limited space. With some of our videos recorded on phone with higher quality, there may be some inconsistency in the quality of images being compared between each other.

Finally, the downloaded videos were largely based around the city itself, which contained many buildings and densely-populated areas. Our dataset for Amherst (both on camera and from YouTube) contained more spacious and rural areas like farms, trees, open fields and university campus, which triggered our model to identify them as distinguishing features of Amherst from the other cities. Trees and open fields are not necessarily the most distinctive features that identify a city, unless they are something very uncommon or rare. It was logical and expected for our model to identify trees since there were very few trees in the images and videos for Non-Amherst cities. A possible solution to this would be to find and extract more rural areas around the United States or the world that are more closely resemble Amherst to help mitigate that.

7.2. Time Constraint

When training the datasets on the ResNet classifier, an issue we faced was how tedious and time consuming the training was, which caused us to have limitations on the number of epochs we tested on as hyperparameters, as well as batch size of the dataset in preprocessing to save time. We ran into complications when attempting to utilize the free GPU credit on Google Cloud from Google Colab in attempt to

speed up training time, and due to time constraint, were not able to use it to our advantage. Instead, we had to run 2-3 models at a time locally in order to best use our time for training, using CPU for training, which was incredibly time-consuming; each epoch in training took about 1-1.5 hours to train.

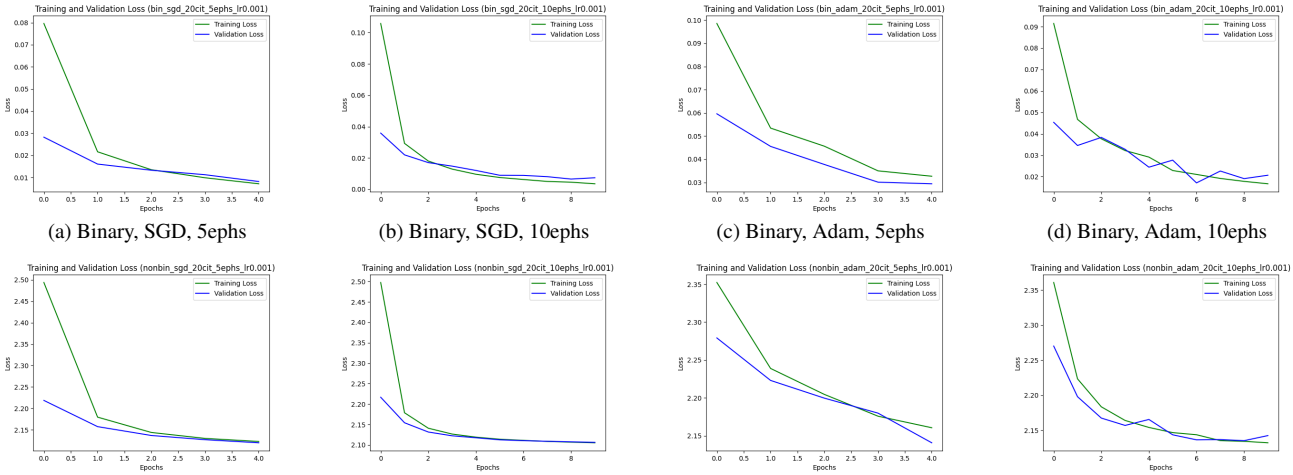
7.3. Human Error

Since our evaluation of the results involved human judgement, it was possible for our accuracy to be slightly skewed due to human error in deciding whether a given image should pass or fail the test. Some CAM images were not always clear or obvious as to whether the model was able to identify the features or not, causing ambiguity. For example, some images had the heatmap cover portions of both distinctive features and non-distinctive features. There were also cases where the generated image showed the heatmap over those distinctive images, but widely ranged in the coloration and shading. Some heatmaps were very clear in red, indicating high confidence in their identification, while others were more yellow and green hues, indicating lower confidence, but recognizing its existence in the image. Since we were limited to ourselves as two individuals evaluating, a possible remedy would be to have multiple participants judge the images to more robust, well-rounded and unbiased evaluation of the results.

8. Future Work

Potential endeavors we would like to explore in the future include using Vision Transformers (ViT) and Convolutional Neural Networks (CNN) for training on the dataset. We experimented and observed how the weighted ResNet classifier performed in identifying features that distinguish Amherst from other cities. Exploring more complex networks with pretrained weights, such as VGG16, will be interesting to see how much we can improve our results with Class Activation Maps. Another avenue with even more complexity would be ViTs, which have larger model sizes and higher memory requirements compared to CNNs, and would involve Natural Language Processing.

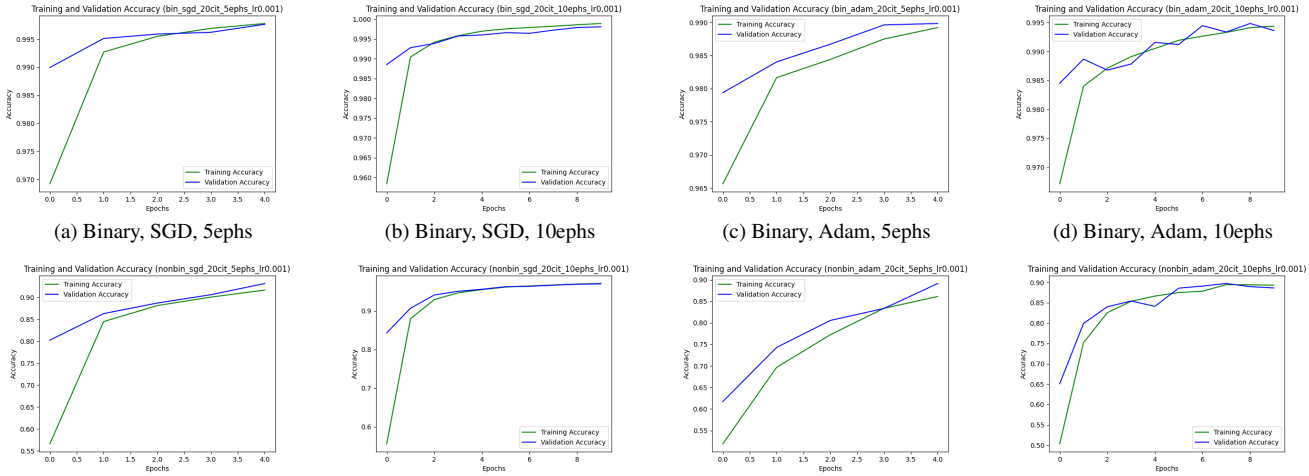
Another potential area for further exploration in our project is object detection and semantic segmentation. With the success in finding certain features in an image of Amherst that makes it unique, the next rational step would be to see if we can now identify and label specific objects or features in an image that make Amherst unique, such as identifying the feature to be a building, a specific type of tree, or random objects like a garbage can or lamp post. These potential works would be the next steps in seeing how much we can improve from where we have started, and would be interesting to see how much better can our image classification can get.



(a) Binary, SGD, 5ephs (b) Binary, SGD, 10ephs (c) Binary, Adam, 5ephs (d) Binary, Adam, 10ephs

(e) Non-binary, SGD, 5ephs (f) Non-binary, SGD, 10ephs (g) Non-binary, Adam, 5ephs (h) Non-binary, Adam, 10ephs

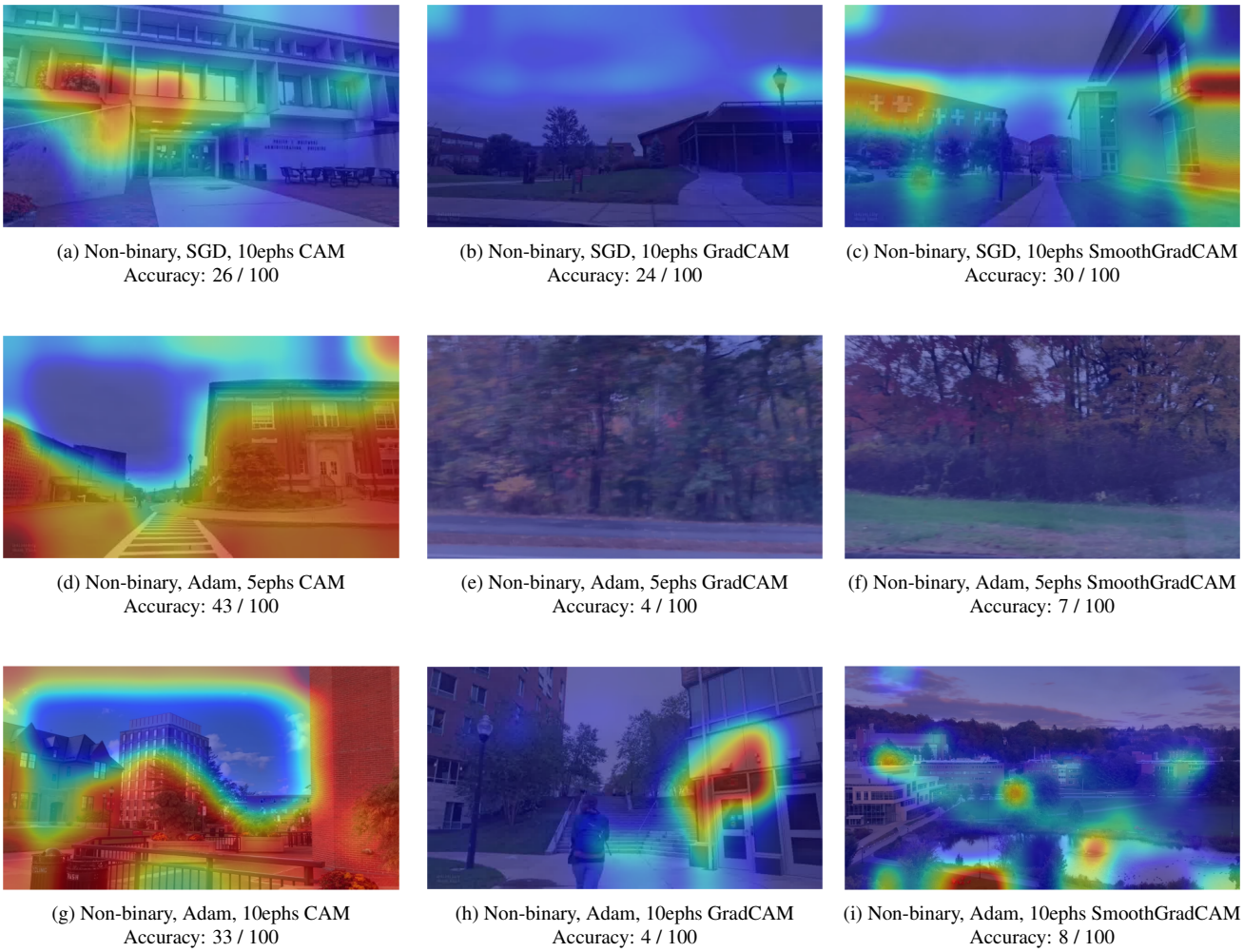
Figure 5. ResNet Classifier Training and Validation Loss based on Data Partition, Optimizer, Number of Epochs and CAM type



(a) Binary, SGD, 5ephs (b) Binary, SGD, 10ephs (c) Binary, Adam, 5ephs (d) Binary, Adam, 10ephs

(e) Non-binary, SGD, 5ephs (f) Non-binary, SGD, 10ephs (g) Non-binary, Adam, 5ephs (h) Non-binary, Adam, 10ephs

Figure 6. ResNet Classifier Training and Validation Accuracy based on Data Partition, Optimizer, Number of Epochs and CAM type



(a) Non-binary, SGD, 10ephs CAM
Accuracy: 26 / 100

(b) Non-binary, SGD, 10ephs GradCAM
Accuracy: 24 / 100

(c) Non-binary, SGD, 10ephs SmoothGradCAM
Accuracy: 30 / 100

(d) Non-binary, Adam, 5ephs CAM
Accuracy: 43 / 100

(e) Non-binary, Adam, 5ephs GradCAM
Accuracy: 4 / 100

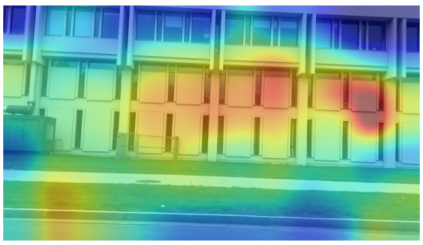
(f) Non-binary, Adam, 5ephs SmoothGradCAM
Accuracy: 7 / 100

(g) Non-binary, Adam, 10ephs CAM
Accuracy: 33 / 100

(h) Non-binary, Adam, 10ephs GradCAM
Accuracy: 4 / 100

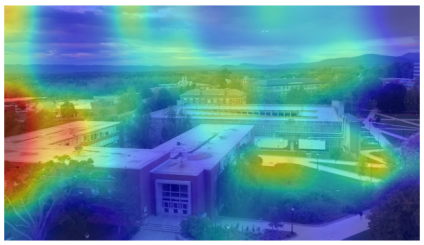
(i) Non-binary, Adam, 10ephs SmoothGradCAM
Accuracy: 8 / 100

Figure 7. Sample image of each CAM correctly highlighting Amherst's distinct features and its Accuracy based on Data Partition, Optimizer and Number of Epochs

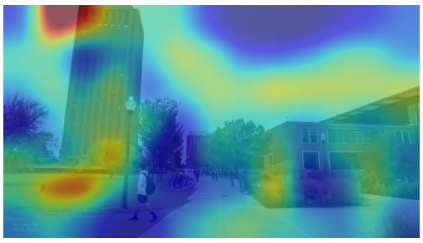


(a) Binary, SGD, 5ephs CAM
Accuracy: 39 / 100

(b) Binary, SGD, 5ephs GradCAM
Accuracy: 10 / 100



(c) Binary, SGD, 5ephs SmoothGradCAM
Accuracy: 14 / 100



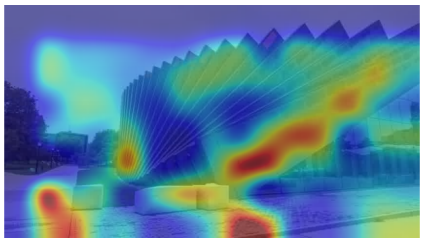
(d) Binary, SGD, 10ephs CAM
Accuracy: 49 / 100

(e) Binary, SGD, 10ephs GradCAM
Accuracy: 6 / 100

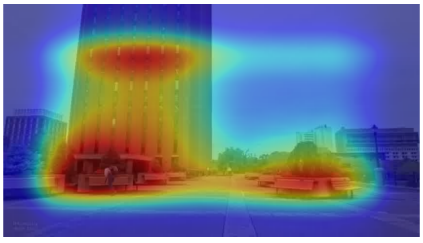
(f) Binary, SGD, 10ephs SmoothGradCAM
Accuracy: 26 / 100

(g) Binary, Adam, 5ephs CAM
Accuracy: 68 / 100

(h) Binary, Adam, 5ephs GradCAM
Accuracy: 16 / 100

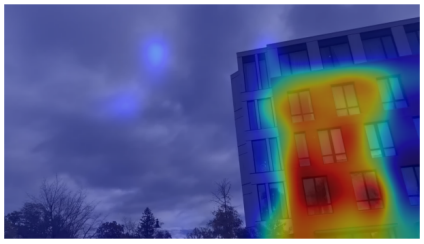


(i) Binary, Adam, 5ephs SmoothGradCAM
Accuracy: 36 / 100

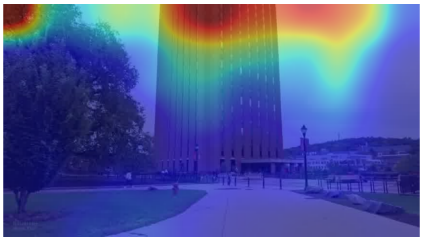


(j) Binary, Adam, 10ephs CAM
Accuracy: 45 / 100

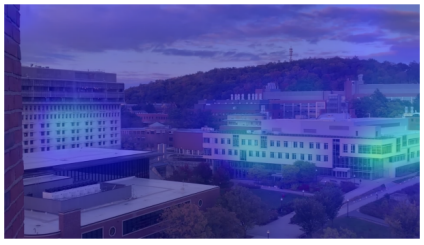
(k) Binary, Adam, 10ephs GradCAM
Accuracy: 10 / 100



(l) Binary, Adam, 10ephs SmoothGradCAM
Accuracy: 18 / 100



(m) Non-binary, SGD, 5ephs CAM
Accuracy: 9 / 100



(n) Non-binary, SGD, 5ephs GradCAM
Accuracy: 15 / 100

(o) Non-binary, SGD, 5ephs SmoothGradCAM
Accuracy: 30 / 100

Figure 8. Sample image of each CAM correctly highlighting Amherst's distinct features and its Accuracy based on Data Partition, Optimizer and Number of Epochs

References

[1] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A.A. Efros. What makes paris look like paris? *ACM Transactions on Graphics (SIGGRAPH)*, 31(4):101:1–101:9, 2012. 1, 2

[2] F.G. Fernandez, P. Mandke, and A. Stergiou. torch-cam. github.com/frgfm/torch-cam. 4

[3] D. Jayakody. Custom image classifier with pytorch – a step-by-step guide. dilithjay.com/blog/custom-image-classifier-with-pytorch/. 2

[4] J. Kang, M. Körner, Wang Y., H. Taubenböck, and X.X. Zhu. Building instance classification using street view images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018. 2

[5] D. Omeiza, S. Speakman, C. Cintas, and K. Weldemariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. *Intelligent Systems Conference 2019*, 2019. 2

[6] J. Philbin, R. Arandjelović, and A. Zisserman. The oxford buildings dataset. *Visual Geometry Group: Department of Engineering Science, University of Oxford*, 2023. 1

[7] K. Venkata Vinay Kumar Reddy and J.J. Thangaraj. Resnet is a novel method for more accurate segmentation and classification of remote sensing images as compared to knn. *Journal of Survey in Fisheries Sciences*, 2023. 2

[8] R.R. Selvaraju, M. Cogswell, Das A., R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 2019. 2

[9] L. Tang. Pytorch implementation of class activation map (cam). leslietj.github.io/2020/07/15/PyTorch-Implementation-of-Class-Activation-Map-CAM. 4

[10] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *Intelligent Systems Conference 2019*, 2019. 2