Project Report - ECE 176

Matteo Persiani
A17368254

Carson Rae
A17558364

Abstract

This project uses various deep learning architectures to explore landmark classification, focusing on a few classes from the Google Landmark Dataset V2. We used a methodological approach to preprocess, augment, and classify landmark images, aiming to discover which network performs best in terms of accuracy and efficiency. Our exploration includes adapting and training several neural network models (ResNets, AlexNet, and VGG) to understand their performance and suitability for this task.

1 Introduction

Landmark recognition presents unique challenges due to global landmarks' vast diversity and similarity. We want to discover which CNN network performs best on these landmarks. Our motivation stems from the desire to do something related to San Diego. Hence, we included Geisel Library as one of our classes. We understand that different networks will lead have certain advantages and disadvantages for different problems. We propose a systematic approach to preprocessing and augmenting data, then training customized neural network models. Our initial results were not very promising but after adapting our networks and training using different optimizers, learning rates, etc., we achieved promising improvements over baseline models. We continued to tune our networks to achieve high-performing models.

2 Related Work

Deep learning for landmark recognition has been pretty extensively explored. Works such as *Google-Landmark Recognition with Deep Learning* [1], and *LANDMARK RECOGNITION USING CNN* [2]. Relations to our work are mentioned in the references section at the bottom of the file. Our work differentiates itself as we implemented all our models from scratch, and we are comparing multiple architectures to find the highest-performing model for the task.

3 Method

Our methodology encompasses three key components: data preprocessing, model architecture implementation and adaptation, and training optimization.

Data Preprocessing: We began by searching through the Google Landmarks V2 dataset for the landmark IDs that we wanted. Next, we wrote a Python script to search through all the landmarks and download only the landmark IDs we listed. Each landmark was resized to 128x128 and separated into its own folder in preparation for loading the dataset. After meeting with a TA, he suggested resizing our images to 256x256 instead to preserve more of the image detail. We updated the landmark filtering Python script and re-downloaded all the images. Additionally, we used a Google Chrome extension to download extra images for Geisel and the Leaning Tower of Pisa, which did not have many images. In the Python notebook for each model, we applied one of three different sets of augmentation to each training image. The data was then split 80/20 into a train/validation set (we discussed this split with the TA, and he approved it instead of splitting it 70/10/20).

Model Architecture:

1. **ResNet10:**
   (a) Convolutional Layer with a 7x7 kernel, stride 2, followed by Batch Normalization (BN) and ReLU activation.
   (b) Max Pooling Layer with a 3x3 kernel, stride 2.
   (c) Residual Blocks:
      i. A sequence of 1 residual block with 64 filters.
      ii. A sequence of 1 residual block with 128 filters, downsampled.
      iii. A sequence of 1 residual block with 256 filters, downsampled.
      iv. A sequence of 1 residual block with 512 filters, downsampled.
      v. Each residual block contains 2 convolutional layers with BN and ReLU, plus a shortcut connection.
   (d) Global Average Pooling to reduce spatial dimensions.
   (e) Fully Connected Layer for classification.
2. **ResNet34:**
   (a) Convolutional Layer with a 7x7 kernel, stride 2, followed by Batch Normalization (BN) and ReLU activation.
   (b) Max Pooling Layer with a 3x3 kernel, stride 2.
   (c) Residual Blocks:
      i. A sequence of 3 residual blocks with 64 filters.
      ii. A sequence of 4 residual blocks with 128 filters, downsampled.
      iii. A sequence of 6 residual blocks with 256 filters, downsampled.
      iv. A sequence of 3 residual blocks with 512 filters, downsampled.
      v. Each residual block contains 2 convolutional layers with BN and ReLU, plus a shortcut connection.
   (d) Global Average Pooling to reduce spatial dimensions.
   (e) Fully Connected Layer for classification.
3. **AlexNet:**
   (a) 5 Convolutional Layers:
      i. The first convolutional layer has 96 kernels of size 11x11, stride 4.
      ii. Max pooling and normalization layers.
      iii. The second convolutional layer has 256 kernels.
      iv. Max pooling and normalization.
      v. The next three convolutional layers have 384, then 384 again, and lastly 256 kernels, with some followed by max pooling.
   (b) 3 Fully Connected Layers: The first two have 1024 neurons each, and the last one has 12 neurons for classification, with dropout layers in between to reduce overfitting.
4. **VGG16:**
   (a) 13 Convolutional Layers: Arranged in 5 blocks, each with 2 or 3 convolutional layers.
   (b) The number of filters starts at 64 and is doubled after each max-pooling layer, going from 64 to 128, 256, and then 512.
   (c) Max Pooling follows each block to reduce dimensions.
   (d) 3 Fully Connected Layers: The first two have 4096 units each, and the third has 1000 units (for ImageNet classification), with ReLU activations.
   (e) Dropout is applied to the first two fully connected layers to combat overfitting.

Training Optimization: We found different optimizers and hyperparameters to perform better for each model.

1. **ResNet10:** SGD worked best for ResNet10 with hyperparameters of:
   (a) batch size = 64
   (b) learning rate = 0.002

(c) weight decay = 0.0001
(d) momentum = 0.9

2. **ResNet34:** SGD worked best for ResNet34 with hyperparameters of:
   (a) batch size = 64
   (b) learning rate = 0.005
   (c) weight decay = 0.001
3. **AlexNet:** SGD worked best for AlexNet with hyperparameters of:
   (a) batch size = 64
   (b) learning rate = 0.0001
   (c) weight decay = 0.005
   (d) momentum = 0.9
   (e) dropout = 0.5
4. **VGG16:** Adam worked best for VGG16 with hyperparameters of:
   (a) batch size = 64
   (b) learning rate = 0.00001
   (c) weight decay = 0.005
   (d) dropout = 0.6

4 Experiments

Dataset: Augmented Google Landmarks V2
We began our project with the intent to pick out some landmarks from the Google Landmarks Dataset and train a classifier on the data from this dataset. However, in attempting to accomplish this, we found that some of our desired landmarks did not have a large enough set of images. So, we used a Google plugin to download images from the internet. In the end, all of our classes have over 100 images.

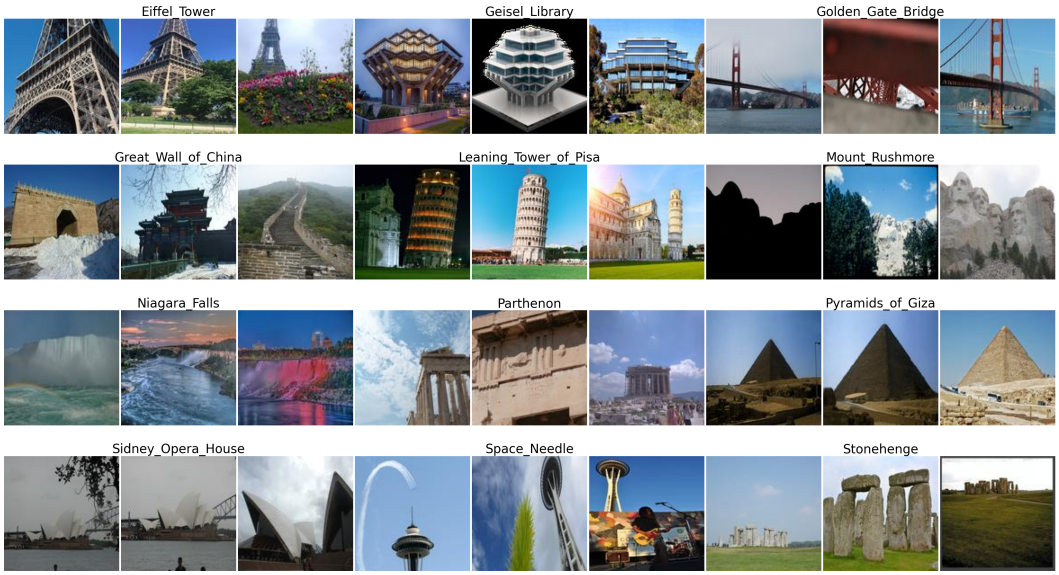


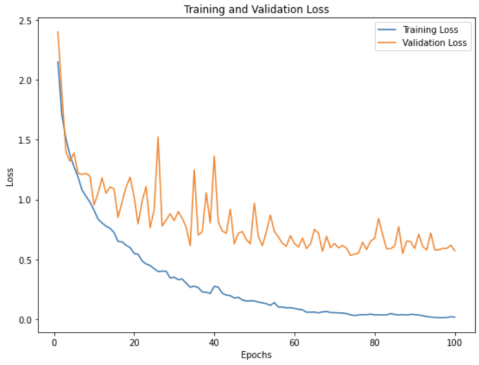
Figure 1: Examples of our data before augmentations.

Results: Our experiments reveal that ResNet10 outperformed all of the models, achieving an accuracy of 87%. ResNet34 and AlexNet both achieved accuracies of 80%. We believe that ResNet34 could have outperformed AlexNet, but issues with Datahub prevented us from being able to do extensive training on the model. VGG16 achieved an accuracy of 73%.

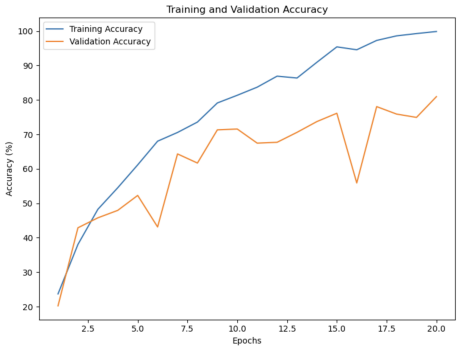
Ablation Study: Reducing dataset size by 50% led to a noticeable decrease in accuracy. Also, our models began performing better after we added about 500 more images to our training set, which showed us the importance of a large and diverse dataset.

Model Results:

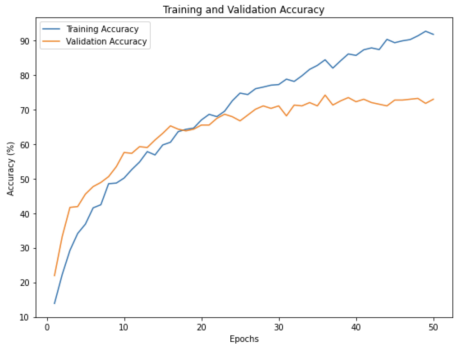
1. **ResNet10:**



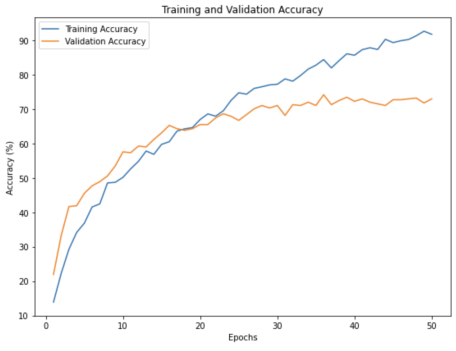
2. **ResNet34:**



3. **AlexNet:**



4. **VGG16:**



5 Supplementary Material

GUI: Additionally, we created an interactive GUI that allows the user to input an image from their own computer, and each model will display its individual result. VGG.pt was too large of a file to upload to GitHub, so the user needs to run VGG16.ipynb to save the model locally and then run the GUI after installing the necessary requirements. The GUI can also be used without VGG16.pt by commenting out the lines involving it.

Video: YouTube Link

References

[1] Chien-Yi Chang *Google-Landmark Recognition with Deep Learning*. Stanford University, 2021.

This landmark recognition model uses the same dataset as ours, but it is trained on the 20 most frequent landmark classes within the dataset. We were inspired by this project but wanted to put our spin on it and make it personal to us. This project was only used to help us find a dataset.

[2] Abhishek Gupta, Dr. Lokesh Kumar *LANDMARK RECOGNITION USING CNN*. International Research Journal of Modernization in Engineering Technology and Science, 2021.

This project manually created its dataset and does Landmark Recognition on 30 landmark classes. We used this project for inspiration on the types of models that would work well in a Landmark Recognition task.