Landmark Recognition

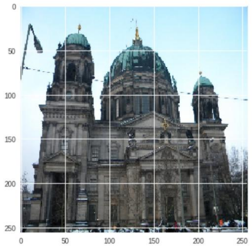
Introduction

Our project consisted of a modified version of Google's Landmark Recognition challenge. We trained four Keras models on a reduced dataset, in order to determine which was best to utilise for a larger dataset. Employing various visualisation techniques for quantitative analysis revealed that our model correctly identifies the architectural components to a landmark, such as the domes and spires of the Berlin Cathedral. We were able to use a VGG16 model on our large dataset to achieve a validation accuracy of above 65%.

Data/Features

We used Google's labelled Landmark training set of coloured images in order to construct two datasets, one consisting of 10 categories and ~1000 images, and another consisting of 50 categories and ~25000 images. We then split these into training, validation, and test sets. All images were reduced to 256X256 size.. We used zoom, horizontal flips, and rotations for image augmentation. We then trained on these images and the augmented images.

Example data image of the Berlin Cathedral



Models

Separable Convolutional 2D

The first model was composed of Separable Convolutional 2D layers, max pooling layers, batch normalization layers, dropout layers to prevent overfitting, and finally a softmax predictive layer. Separable convolutions is first a depthwise spatial convolution (acting on each input channel separately) followed by a pointwise convolution which mixes together the resulting output channels.

Residual Neural Network

We used a prebuilt model of a residual neural network from Keras. We initialised the weights on ImageNet, an made one convolutional block trainable We then added a 128-unit dense layer and a Softmax output layer.

Xception

We used a prebuilt Xception model, and made the last convolutional block trainable.

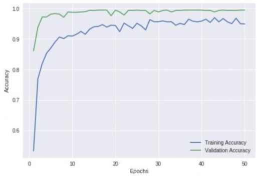
Model Accuracies (10 Classes)

Model	Training Accuracy	Test Accuracy
Separable Convolution 2D	0.95	0.79
VGG 16	0.93	0.99
ResNet 50	0.99	0.98
Xception	0.96	0.90

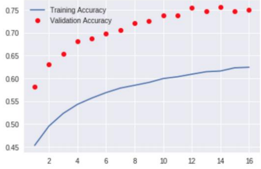
VGG16

The VGG16 consists of a 16 layer model using max-pooling and 3x3 convolutions. We initialised the weights on ImageNet, but then for the larger dataset we retrained the weights. We added a dropout layer, and a softmax predictive layer. For the larger dataset, we were only able to train over 15 epochs due to time restraints

Accuracy Plot (10 classes)



Accuracy Plot (50 classes)



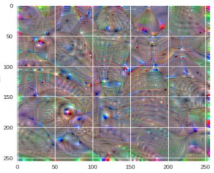
Results and Visualisations

The maximum activation image[1] is composed of the image that would maximize the filter output activations of each filter layer in the model:

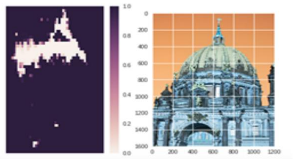
$$\frac{\partial Activation\ Loss}{\partial input}$$

Implementing this for our model produced images that maximized the model's filters to predict the current class. For the Berlin Cathedral class, dome like structures appear to maximize activations.

Maximum Activation Image

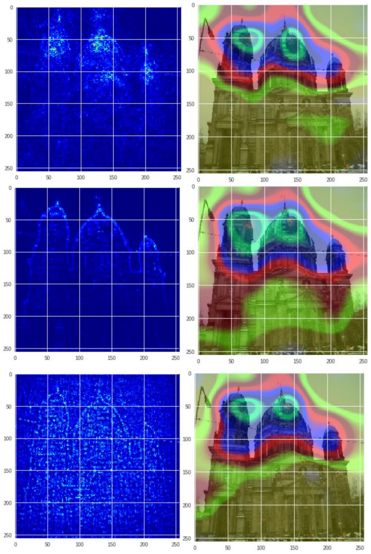


Occlusion Sensitivity



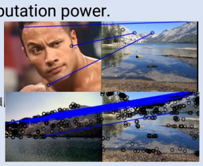
Blocking out key features of the structure in an implementation of occlusion sensitivity[4], such as the dome of the Berlin Cathedral, drastically reduced the accuracy of the model, showing that our model is looking for such features. The saliency maps[2] visualize which regions of the image would cause the most change to the output, if they were changed. The class activation maps visualize the attention over the penultimate convolutional layer in respect to the input. Those areas which are most important are represented by a higher "heat map" designation.

Saliency Maps and Class Activation Maps
Top to Bottom: No backprop modifiers, Guided, Relu,



Discussion and Future Work

With additional computation resources, the VGG-16 model can be trained with more epochs and a dataset of more classes to gauge how practically applicable our model is. While attempts at training ResNet-50 on 50 or more classes resulted in overfitting, additional computation power could also allow flexibility in tuning hyperparameters. To classify non-landmark images, DELF[3] can be used since inliers vary for correctly and incorrectly classified images. DELF can extract features in an image which can be compared to the features of an image of the class that the non-landmark image was classified to. Comparing the number of matched between images of the class and those not in it can create a threshold for whether or not an image classified by our model is actually in the class. This problem requires greater computation power.



Using DELF to classify non-landmarks
Top: DELF features inliers between incorrectly classified image
Bottom: Inliers between images of same class

1. Erhan, Dumitru, Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Visualizing higher-layer features of a deep network." University of Montreal 1341, no. 3 (2009): 1.
2. Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: visualising image classification models and saliency maps (2014)." arXiv preprint arXiv:1312.6034 (2013).
3. Noh, Hyeonwoo, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. "Large-Scale Image Retrieval with Attentive Deep Local Features." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3456-3465. 2017.
4. Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." In European conference on computer vision, pp. 818-833. Springer, Cham, 2014.

Adil Nygaard (adiln@stanford.edu)   Aamnah Khalid (aamnah@stanford.edu)   Uzair Navid Iftikhar (unavid@stanford.edu)