Landmark Recognition

Deepak Nagaraj, Mike Stackhouse and Roland Lim

Objective

Recognize well-known landmarks based on images with inference on the edge: Image Classification

Why?

✓ Organize personal photo collections

✓ Visual search

✓ Computer vision, autonomous driving

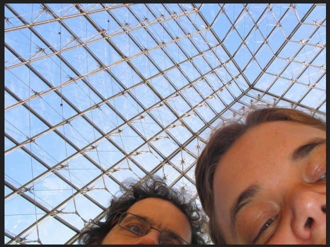
✓ Many other commercial applications
e.g. gaming, education, tourism, $$

Dataset

- The Paris Dataset
- 6412 images collected from Flickr
- 12 Paris Landmarks
    - La Defense Paris
    - Eiffel Tower Paris
    - Hotel des Invalides Paris
    - Louvre Paris
    - Moulin Rouge Paris
    - Musee d'Orsay Paris
    - Notre Dame Paris
    - Pantheon Paris
    - Pompidou Paris
    - Sacre Coeur Paris
    - Arc de Triomphe Paris
    - Paris

Data preparation - Gathering

- Poor quality images removed
  - Dataset contained quality rating of each image - kept images where >25% of object is visible

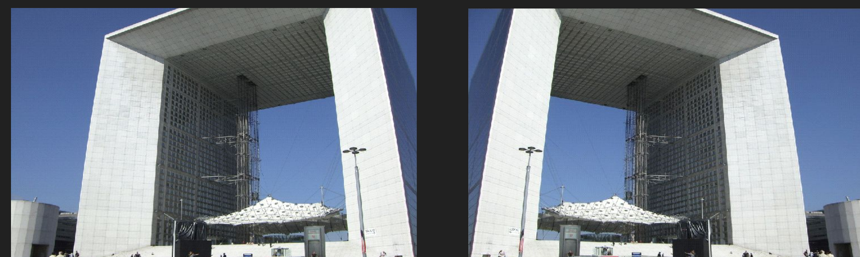



- Supplement lacking classes
  - Eiffel Tower, Musée d'Orsay, and The Centre Pompidou had <100 images each
  - Used [Google Images Download] to add ~75 additional images each

- Correct distractor images
  - Images in the “general” class were matched other classes

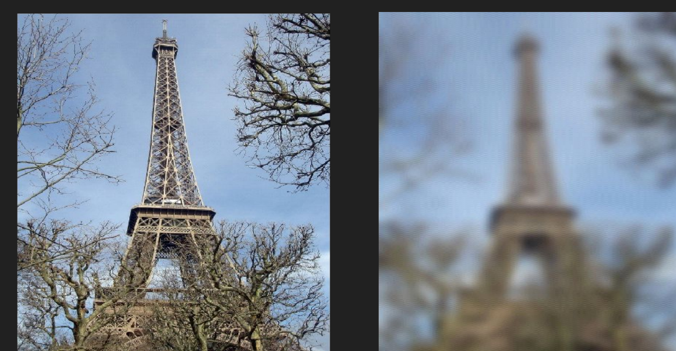
Data preparation - Augmentation

- Applied augmentation techniques using the [PIL / Python Image Library]



Horizontal Flip

Grayscale



Blur

Model Development

- Leverage existing models by applying transfer learning

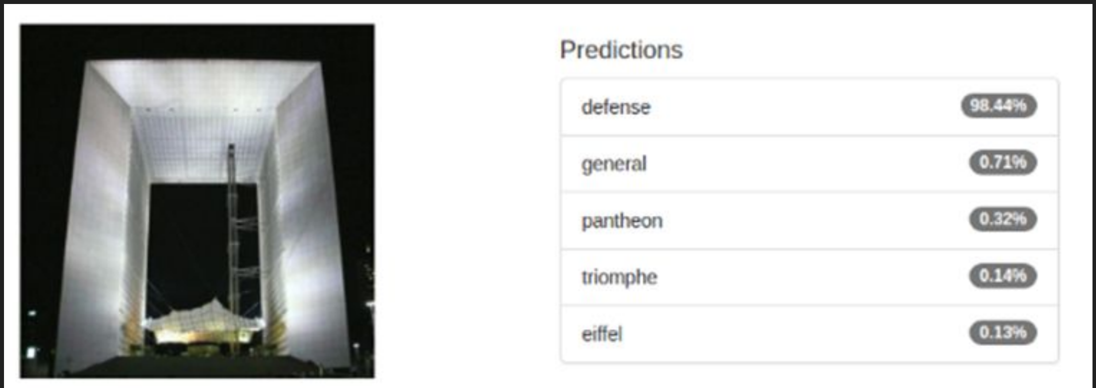
- Took two approaches:

  - NVIDIA DIGITS
    - AlexNet
    - VGG16

  - Tensorflow for Poets
    - MobileNet 0.50
    - Inception V3

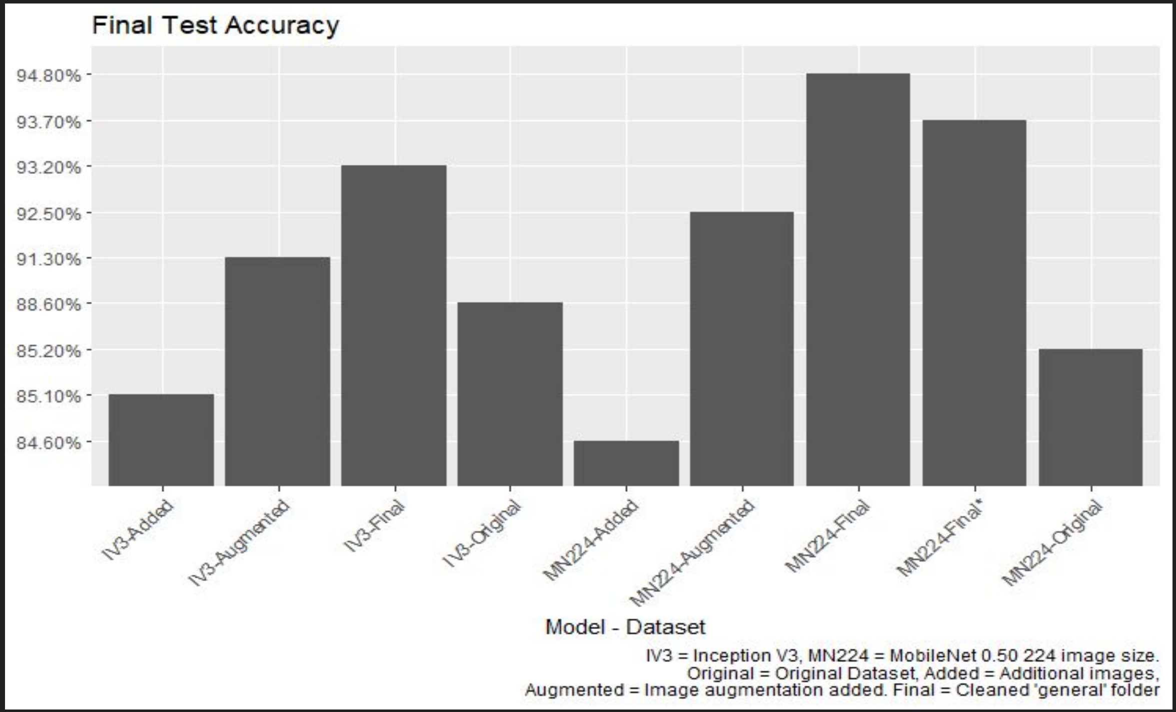
DIGITS

- DIGITS does not have bottlenecking built in but allows you to download a pretrained model
- Trained on TX2 for 3 epochs:
  - AlexNet performance was fairly poor
  - Achieved up to 93% accuracy with VGG16
- VGG16 achieved 97% trained on cloud P100 with 30 epochs
  - Largest concern is that the model is very heavy



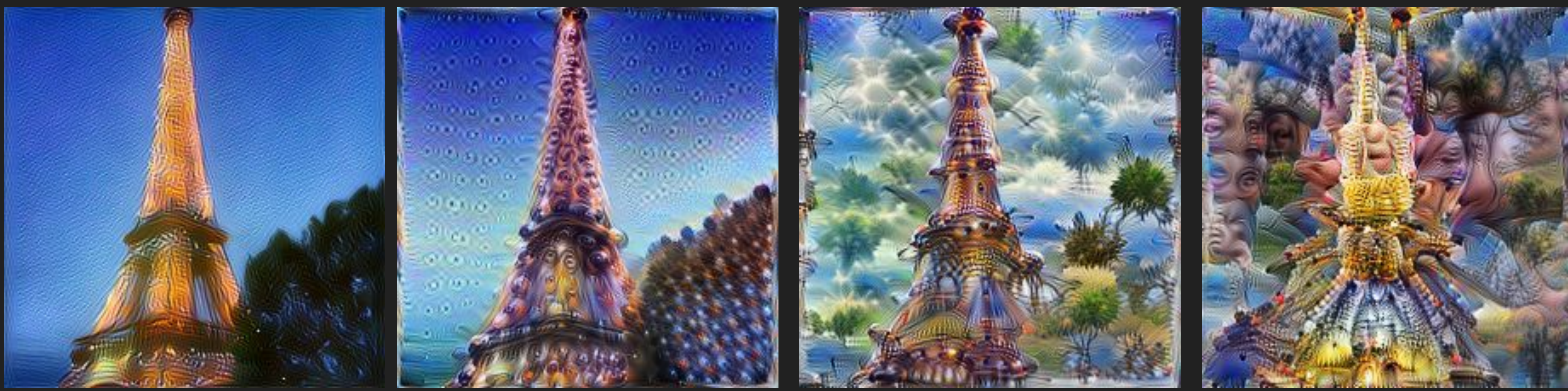
Tensorflow for Poets

- Very efficient training using bottleknecking (<20 minutes)
- Inception V3
  - Top accuracy 93.2%
- MobileNet 0.50
  - Top accuracy 94.8%



Neural Net Visualization

- Looking inside a neural net to see how image morphs over the layers
- Eiffel Tower over Inception v1:



Why was that classified wrong?

- Example: Les Invalides misclassified as Sacre Coeur

Conclusion and Next Steps

- MobileNet is our ultimate architecture of choice, despite higher performance from VGG16 trained on a cloud P100
  - MobileNet architecture is much smaller and more practical for edge computation, yet still achieved 94.8% accuracy
- Next steps would be to expand the number of landmarks classified
  - Need a larger dataset (i.e. Google Landmark Recognition Challenge)
  - Image generalization could become problematic, so possibly link location data
- Could apply model pruning leveraging NVIDIA transfer learning toolkit
  - Cuts model size down further, faster computation

References

- http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/
- https://www.kaggle.com/c/landmark-recognition-challenge
- https://landmarksworkshop.github.io/CVPRW2018/
- https://www.blippar.com/blog/2018/02/16/landmark-recognition-api-never-confuse-landmark-again
- https://cloud.google.com/blog/products/gcp/around-the-world-landmark-detection-with-the-cloud-vision-api
- https://medium.com/@abhinaya08/google-landmark-recognition-274aab3c71ae
- https://towardsdatascience.com/google-landmark-recognition-using-transfer-learning-dde35cc760e1
- https://medium.com/@abhinaya08/google-landmark-recognition-274aab3c71ae
- Lucid Visualization Library: https://colab.research.google.com/github/tensorflow/lucid/blob/master/notebooks/misc/feature_inversion_caricatures.ipynb