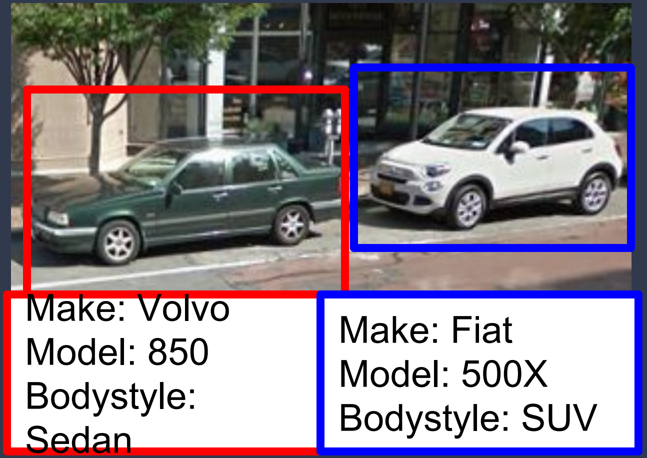
Automotive Image Classification with Convolutional Neural Networks

Joshua Wilding
General Assembly Data Science Immersive Meet and Hire
11/7/2018

GENERAL ASSEMBLY

Overall Project Goal



- Build a model that can classify the make/model/bodystyle/year of a car based on an image
- Data: Stanford Cars Dataset
- Also could use car shopping websites
- Ideally, model will account for different points of view (front/side/rear view etc)
- Stretch goal: User can submit image and model will identify all vehicles and classify them

Data

- Sourced from [Stanford Cars Dataset]
- Contains 16,185 images of cars
- 196 unique vehicle classes
  - 9 body styles
  - 48 makes
  - 157 models
  - 16 years

- Lincoln Town Car Sedan 2011
  - Body Style: **Sedan**
  - Make: **Lincoln**
  - Model: **Town Car**
  - Year: **2011**

- Land Rover Range Rover SUV 2012
  - Body Style: **SUV**
  - Make: **Land Rover**
  - Model: **Range Rover**
  - Year: **2012**

Model Structure

- Create a decision tree of neural networks
- Start by creating a neural network (NN) that classifies **body style** (SUV, Sedan, Coupe ….)
- Within each body style category, create NN that classifies **make** (Honda, BMW, Ford….)
- Same logic for **model** and **year**

- Advantage: Breaks classification problem into more manageable pieces
- Disadvantage: Requires training multiple neural networks

1. Input image:

2. Input NN → Body Style = **Convertible**
3. Convertible NN → Make = **Chrysler**
4. Chrysler Convertible NN → Model = **Sebring**
5. Chrysler Sebring NN → Year = **2012**
6. Output = **Chrysler Sebring Convertible 2012**

Image Pre-Processing

- Randomly alter images used to train neural network
  - Horizontal flips
  - Rotations
  - Image shears
  - Vertical and horizontal shifts
  - Zooms
  - Convert to grayscale
  - Standardize image size
- Removes unnecessary information
- Ensures all images can be compared
- Deters model from overfitting training data
- Maintains image details and features

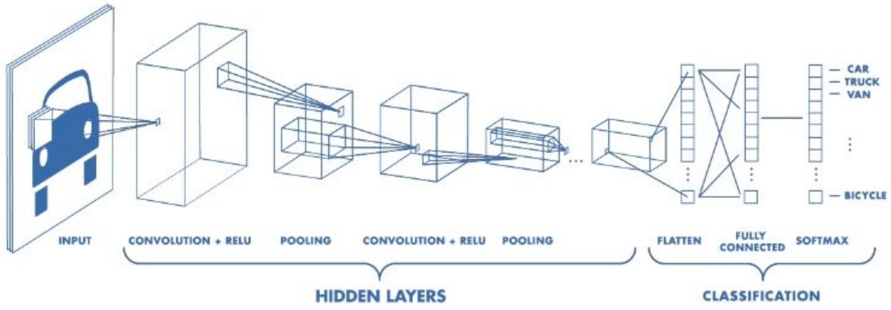
Original Image

Image for Neural Network Input

Modeling with Convolutional Neural Networks

- Contain several layers used to classify image
- Input layer
  - Raw image data
- Convolutional layers
  - 2D filters that pass over all parts of the image
  - Identify details of the image
  - Shape of filter (square, vertical, horizontal) determines detail identification
- Pooling layers
  - Reduce size of image
  - Maintain strongest features (pixel value)
- Fully connected layers
  - Perform logical reasoning used for classification
- Output layer
  - Probability of image belonging to each class

- Dropout
  - Randomly avoids a proportion of hidden nodes
  - Used to combat overfitting to training data



Running the Neural Network

- Started by trying to classify body style
  - 9 categories
- Trial and error: required constant changes to network structure and parameters
  - Make a change to my model
  - Run several epochs while keeping a close eye on training and test data accuracy
  - Save model for future use if improved
  - Make minor changes to model and re-train

- Problem: My laptop’s CPU takes too long to train neural network
- Solution: Amazon Web Services
  - Connected to a virtual GPU on the cloud
  - Got extra computing power on demand when needed
- Why a GPU?
  - Designed to perform large calculations in parallel that require lots of memory
  - Well suited for matrix multiplication used in neural networks
- Model training was about 10x faster

Neural Network Results

- Results were not what I was expecting
- Maximum accuracy was 30% on training and test data for body style classification
- Unable to classify make, model, or year due to time constraints and lack of predictive power

- Why?
  - Was the problem too big for this approach?
  - Was more training needed?
  - Is this as good as it gets?

Input image (Sedan)

Model Predictions:

- 62% Convertible
- 38% Van

Reflections on Capstone Project

What I learned

- Neural networks are extremely sensitive to their parameters
- Achieving the right balance between over and underfitting is tricky
- Image classification is difficult with multiple classes
- GPUs are significantly better for machine learning than CPUs

Things I would do differently

- Start with a smaller problem
    - I initially wanted to classify all 196 classes at once
- Use AWS earlier
    - I wasted many hours waiting for model to train on my laptop’s CPU

Moving Forward

If I had more time

- Get more accurate predictions of bodystyle
- Predict make, model, and year
- Bring in outside data
    - Scrape car databases and auto sales websites for more images to train model
- Implement object detection
    - Identify vehicle from an image containing other objects
- Design a user friendly application for model predictions
    - User submits photo and model identifies and classifies cars

Potential Applications

- Law Enforcement
    - Identify suspect vehicles in security camera footage
    - Use in conjunction with license plate readers to ensure plate is on proper vehicle
- Online Auto Sales
    - Allow buyers to search based on images
    - Allow sellers to auto-populate information
- Just for fun
    - Help curious car enthusiasts identify interesting vehicles

Thank You!

Sources

- Jonathan Krause, Michael Stark, Jia Deng, LiFei-Fei: **3D Object Representations for Fine-Grained Categorization** 2013
- https://cdn-images-1.medium.com/max/1600/1*NQQiyYqJJj4PSYAeWvxutg.png

About Me

I am an intellectually curious Data Scientist who recently graduated from General Assembly's Data Science Immersive program.

Contact Information

(201) 562-5351
josh@wildinghome.com
linkedin.com/joshuawilding
joshuawilding.github.io