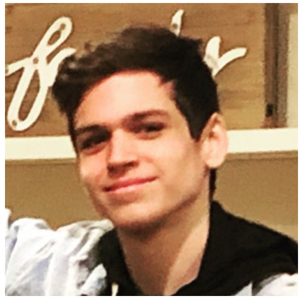
ITCS 4152/5152: Speed Limit Sign Detection

Group 11: Thomas Carr, Kyle Ward, Payne Miller, Jay Yadav, Makalia Vang

UNIVERSITY OF NORTH CAROLINA
CHARLOTTE

Introductions

- Payne Miller
  - Preparation of Data & Research
  - Assisted with Backend Demo
  - Presentation Creation



- Kyle Ward
  - Model Creation & Research
  - Presentation Creation

- Jay Yadav
  - Preparation of Data & Research
  - Assisted with Backend & Frontend Demo
  - Presentation Creation

Introductions

- Thomas Carr
  - Preparation of Data & Research
  - Assisted with Backend & Frontend Demo
  - Model Creation & Research
- Makaila Vang
  - Presentation Creation
  - Preparation of Data & Research

Problem and Motivation

- Speed Limit Detection
    a. Sometimes the speed limit can switch suddenly and traditional GPS systems don't always have the right information stored (outdated.)
    b. Speeding Tickets are expensive.
- There are two main customers for our project:
    a. Autonomous vehicles
        - Need to know what speed it is allowed to do without relying on external data.
    b. Map/GPS companies.
        - Can use Street View images in order to get the speed limit of roads.

Our Dataset

- We used 1,000 images of various quality, lighting, and speed limit values within our model.
- Obtained images via the Open Source platform Kaggle.
- Labels follow the format: speedLimit{Value}_{id}.avi_image{imageNumber}.png
- Train/Test/Validation Split:
    a. Train (70%): 767 images
    b. Test (10%): 106 images
    c. Validation(20%): 216 images

Our Dataset (cont.)

Our Methodology

- Our model is a CNN built using Tensorflow 2
  a. Utilizes max pooling, average pooling, and upsampling layers for feature extraction
  b. Adam optimizer with Categorical Cross-entropy loss function
  c. Output is a probability distribution of possible classes (25,30,35,40,45,50mph)
- Outside-In approach:
  - Assumptions: Speed limit sign is square, numbers will always be contained within this square. Number will exist solely of lines and curves
  - First Conv-Pooling layer uses a max-pool pair for extracting the parts that define the lines of the rectangle and parts of the numbers
  - Second Conv-Pooling layer pair uses an average-pool to extract the contours and curvatures of the numbers
  - These are upsampled before feeding into the MLP part of the network
- We transformed the images into 64 x 64 pixels and applied a Grayscale filter.

Our Methodology - Second Model

- 3 Convolutions Before Any Pooling
  - In hopes of finding a new parameters missed by a simpler model
- Reshaped Images to 224x224x3
  - Used RGB instead of BW
- Added More Data Augmentation
  - Gaussian Blur
  - Rotation (30°)
  - X/Y Shifting (30px)
- Regularization
  - Batch Normalization
  - Dropout
  - 80/20 train/test split

Layer (type)	Output Shape	Param #
conv2d_44 (Conv2D)	(None, 224, 224, 16)	784
conv2d_45 (Conv2D)	(None, 224, 224, 16)	4112
conv2d_46 (Conv2D)	(None, 224, 224, 16)	4112
batch_normalization_18	(None, 224, 224, 16)	64
average_pooling2d_8	(None, 112, 112, 16)	0
conv2d_47 (Conv2D)	(None, 112, 112, 32)	4640
conv2d_48 (Conv2D)	(None, 112, 112, 32)	9248
conv2d_49 (Conv2D)	(None, 112, 112, 32)	9248
batch_normalization_19	(None, 112, 112, 32)	128
average_pooling2d_9	(None, 56, 56, 32)	0
conv2d_50 (Conv2D)	(None, 56, 56, 64)	18496
conv2d_51 (Conv2D)	(None, 56, 56, 64)	36928
conv2d_52 (Conv2D)	(None, 56, 56, 64)	36928
batch_normalization_20	(None, 56, 56, 64)	256
max_pooling2d_12	(None, 28, 28, 64)	0
conv2d_53 (Conv2D)	(None, 28, 28, 128)	73856
conv2d_54 (Conv2D)	(None, 28, 28, 128)	147584
conv2d_55 (Conv2D)	(None, 28, 28, 128)	147584
batch_normalization_21	(None, 28, 28, 128)	512
max_pooling2d_13	(None, 14, 14, 128)	0
flatten_5 (Flatten)	(None, 25088)	0
dense_25 (Dense)	(None, 256)	6422784
dropout_17 (Dropout)	(None, 256)	0
dense_26 (Dense)	(None, 128)	32896
dropout_18 (Dropout)	(None, 128)	0
dense_27 (Dense)	(None, 32)	4128
dropout_19 (Dropout)	(None, 32)	0
dense_28 (Dense)	(None, 16)	528
dense_29 (Dense)	(None, 7)	119

Total params: 6,954,935
Trainable params: 6,954,455
Non-trainable params: 480

Results

- Our best model achieved **90% accuracy**.
- From a development perspective:
    a. Started with a base model of 85% accuracy
    b. From there we iterated to 86% accuracy
    c. Ultimately optimized up to 90% accuracy.
- We used the simple formula for Accuracy:
    a. Accuracy = number of correct reads / total number of reads.
- Accuracy was categorical cross entropy.

Results (cont.)

Conclusions

- Our best model achieved **90% accuracy**.
- Faced some challenges
    a. Scheduling between full-time students and full-time employees/part-time students.
    b. We were pre-processing the images on our own, but then figured out a CNN would do it better.
    c. Picking a topic of interest
    d. Finding a large, diverse, dataset was a huge boon.

Live Demo

- Hosted our backend server on a personal server machine.
- Wanted to hit a stretch goal of a live system demonstration.
- speadle.web.app

Q & A

- Any questions?

Thank you!

- Thank you Professor Archit and TAs.