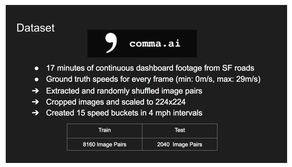
# vid2speed: Deep CNN for Vehicle Speed Estimation

**Benjamin Penchas ‘19**
**Marco Monteiro ‘18**
**Toby Bell ‘19**

## Problem Statement

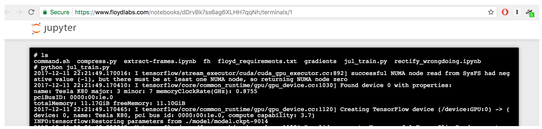
Given car dashboard video footage, we aimed to estimate the speed of a car using a deep neural network. We saw this problem as a small but important part of building an autonomous vehicle. The problem is by its nature underdetermined (since we have no absolute reference for scale/distance), so we treated it as a classification task where we bucketed speeds into 4 mph intervals.

Our dataset was a dashboard video of driving around the Bay Area, supplied and labeled by comma.ai. We separated this video into 20400 image frames, scaled each image to 224x224, and coupled every pair of frames. We then randomly shuffled these pairs--80% into our train set and 20% into our test set.

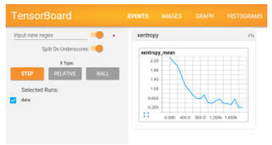



## Floydhub + Tensorflow

We used Tensorflow to implement our network and attempted to train locally. Doing so took on the order of seconds per minibatch (unacceptably slow). We solved this problem by using Floydhub, a cloud GPU platform. The Floydhub team gave us 100 hours of free premium GPU credits. On the Tesla K80, we were able to train for 15 epochs in around 30 minutes!




We also used TensorBoard to monitor our training and visualize our network. TensorBoard allows you to easily make a dashboard that displays intermediate results from the network as well as charts of the training loss / training accuracy.

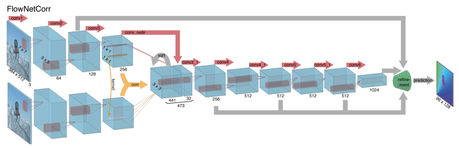



Using these tools allowed us to iterate much more quickly. As a takeaway on the tooling side, we learned to find one imaging library (probably OpenCV) and stick to it. Since each imaging library treats images slightly differently, bouncing between them led to small bugs that were difficult to catch.

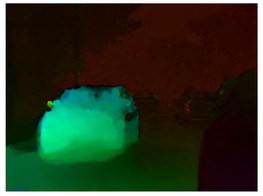
## Initial Experiments

**Learning from Optical Flow**

Optical flow is the problem of extracting the *derivative* of an image--given two subsequent frames, how much is each pixel moving. We thought investigating learned optical flow would be a good starting point for this project. Flownet is a state of the art optical flow network trained on the MPI Sintel dataset. Its architecture is as follows:




Flownet Architecture – Notice the two-stream feature extractor

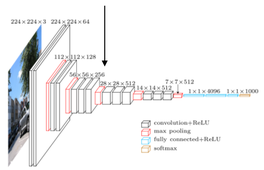


Optical Flow Output We Generated During Investigation

**Creating a Lightweight Optical Flow Model**

We saw the problem of dense optical flow as very similar to our video-to-speed problem, but not exactly the same. Rather than use a pretrained Flownet model, a very deep and complicated network, we wanted to design and train our own lightweight optical flow model specific to this problem.

We initially used VGG-19 as a dual stream feature extractor, mimicking the Flownet architecture and hoping to extract features like scale. We fed each image through a pre-trained VGG-19 model, up to the third block, and then concatenated the results. We then fed the resulting volume through our trainable custom CNN.

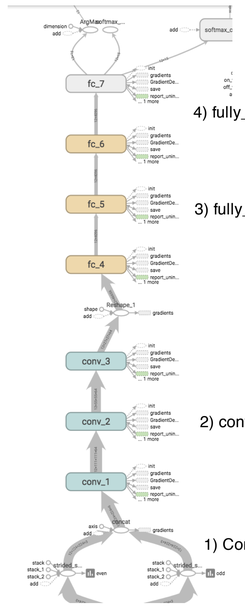



**Takeaways From Initial Experiments**

VGG-19 is an image classifier, and so it has been trained to extract semantic understanding. However, since subsequent frames are semantically very similar, the feature extraction with VGG-19 actually got rid of crucial spatial information. Using VGG-19 as a feature extractor led to a model that could not distinguish between images.

Additionally, VGG-19 has rapid down sampling with several max-pool layers. That architecture caused a loss of resolution in the images. For optical flow, and this video-to-speed problem, high spatial resolution is very important.

## Architecture




5) Apply softmax to get class probabilities across 15 classes (bucketed speeds)

4) fully_connected_layer x1: no activation, 15 neurons

3) fully_connected_layer x3: ReLU activation, 4096 neurons

2) conv_layer x3: ReLU activation, filter_size=3x3, stride=2, num_filters=64

1) Concat images and feed in volume

## Results

We trained our network on cloud GPUs for ~9000 minibatches, or roughly 15 epochs. We used the Adam optimization algorithm (learning rate 0.01) and no dropout. We plotted our training loss and accuracy:

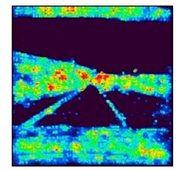
Training Loss


Training Accuracy


As expected with training on minibatches, our accuracy on each batch varies but tends upward. By the end of the 15 epochs, we were able to completely fit the train set. We then evaluated on our unseen test set. **Our trained model achieved 91% accuracy on the test set.** Given the varied nature of roads in the train/test sets, we feel confident the network would generalize further to unseen road environments. We did not get to try re-training with regularization (such as by using or dropout); in our next iteration we would like to try regularizing the network.

## Interpreting the Results

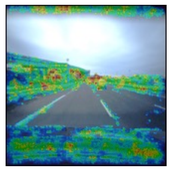
Raw image



Pixel-level saliency

**Peeking inside the “black box”**

To better understand our results, we took the gradient of the loss with respect to each pixel of an input image. We then visualized the gradients by coloring the pixels by the magnitude of the norm of their gradients (brighter here means a larger gradient). In other words, we are left with a “heat map” of which pixels the network focuses on to make its decision.



Overlayed saliency

As you can see, the network learned to ignore the road and the sky. It learned to focus on the white lines in the road (something we hypothesized it would). Surprisingly, it also focuses on the hood of the car / the top of the car. After investigating this phenomenon more, we believe the car itself shakes more at higher speeds and thus the network has learned to factor in the shaking of the car. How neat!

## References

E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In CVPR, 2017.

Fischer, Philipp, Dosovitskiy, Alexey, Ilg, Eddy, Hausser, Philip, Hazrba, Caner, Golkov, Vladimir, van der ¨ Smagt, Patrick, Cremers, Daniel, and Brox, Thomas. Flownet: Learning optical flow with convolutional neural networks. In ICCV, 2015.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

Websites:
https://github.com/pathak22/pyflow
https://comma.ai/