SAPIENZA
UNIVERSITÀ DI ROMA

Self-Driving Car 3D Model with Pattern Recognition via a Convolutional Neural Network

COMPUTER VISION PRESENTATION BY MATTEO PRATA & ANTONIO RICCIARDI - 11/01/2019

Introduction to the problem

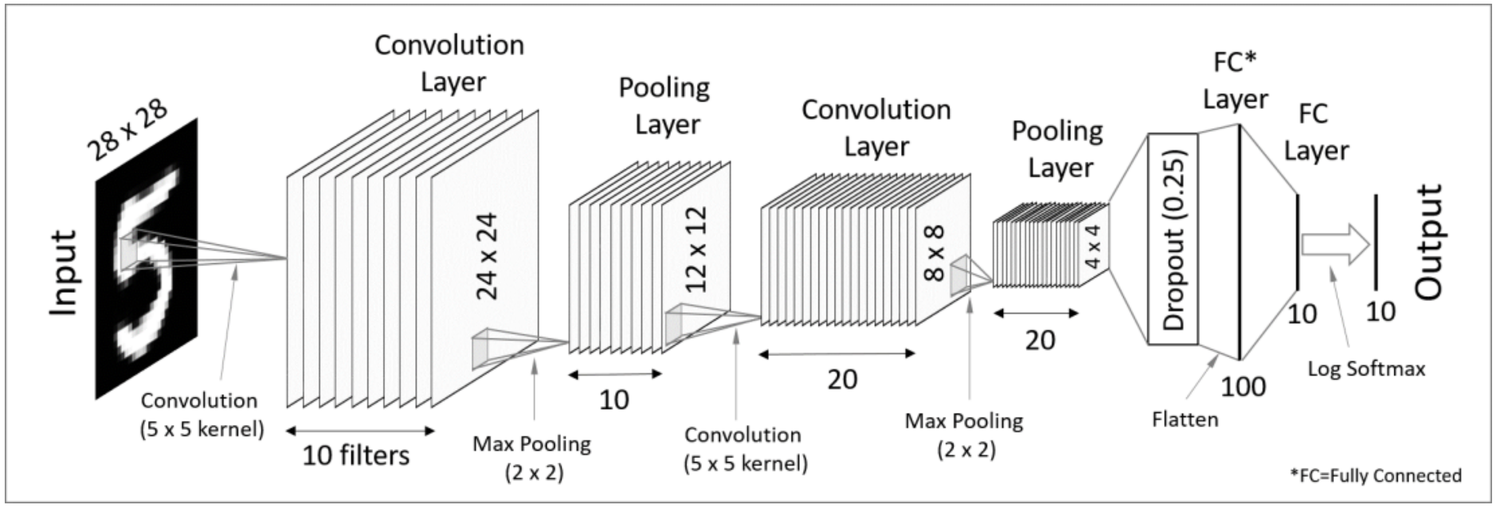
“End to End Learning for Self-Driving Cars”

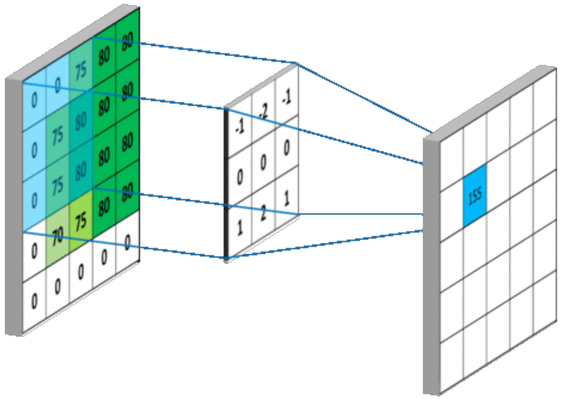
By Marius Bojark et al. (2016)

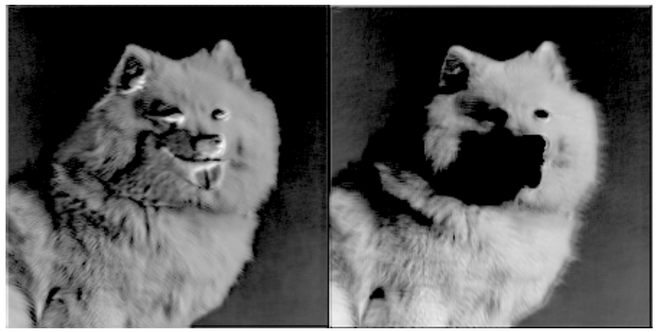
- Train a Convolutional Neural Network (CNN) to map raw pixels, from a front-facing camera mounted on a car, directly to the steering commands
- CNN goes beyond pattern recognition, it learns the entire processing pipeline to steer the car
- Empirical demonstration that CNNs can learn complex tasks of road following, only through vision and without any centralized set of manual constraints

The CNNs power

- CNNs are feed forward neural networks which have revolutionized pattern recognition
- Features are learned automatically from training examples
- Convolution captures the 2D nature of images

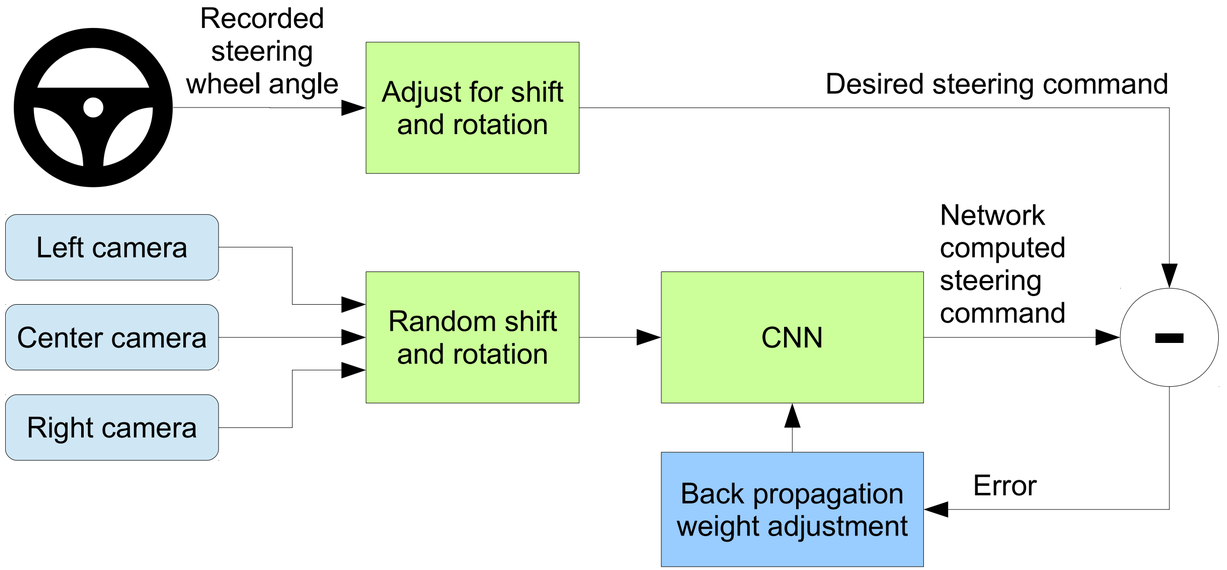




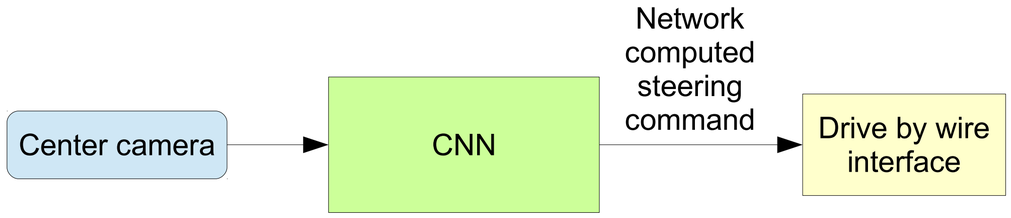


End to End Learning for Self-Driving Car

- **PHASE 1**
  Training phase

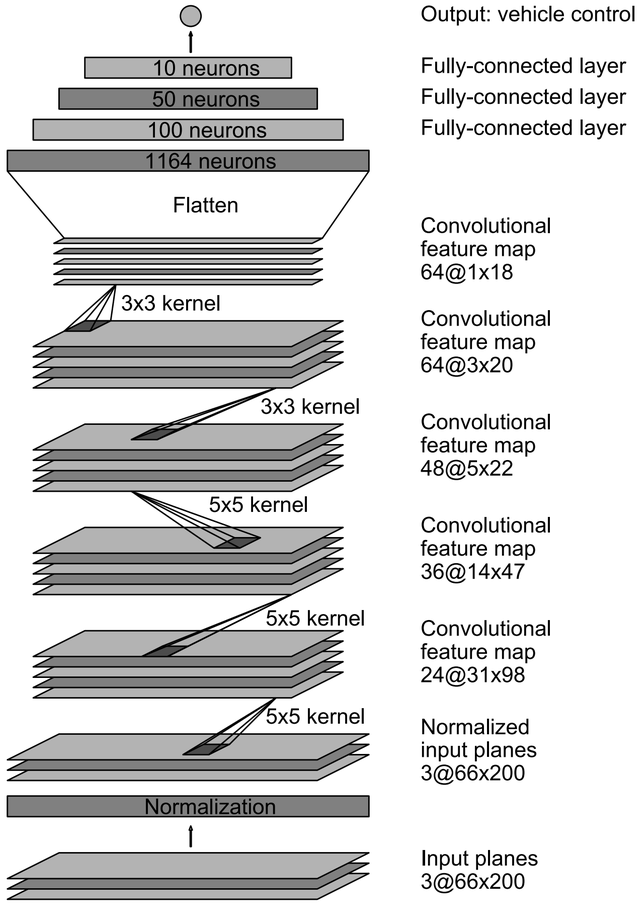


- **PHASE 2**
  Test phase



CNN architecture

- Input images captured from a wide variety of roads, diverse lights and wether conditions
- Input images are split into YUV planes and passed through the CNN
- **9 layers**: 1 normalization, 5 convolutional, 3 fully connected layers. 27 million connections and 250 thousand parameters.
- Train the weights of the FC layer so to minimize the mean squared error of predicted steering command and the correct one



Data selection and augmentation

- Sampled **10 FPS** from the videos recorded from the three cameras, avoid useless similar frames
- Sample only the frames where the driver was staying in a lane, discard the rest
- Added artificial **flips** and **shifts** to let the model learn how to recover from bad orientations, magnitude chosen from a normal distribution

Model evaluation

- The **autonomy** estimates what percentage of time the network could drive the car

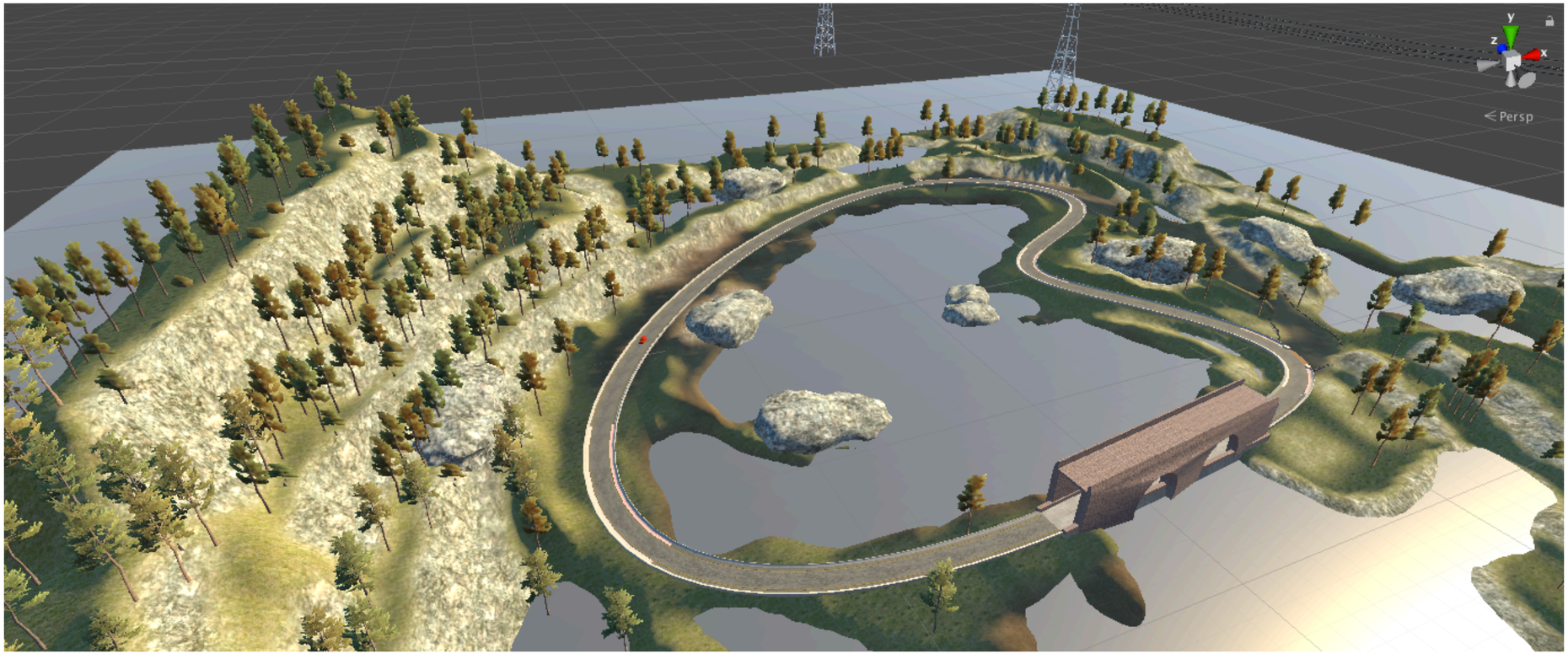
$$\text{autonomy} = (1 - \frac{(\text{number of interventions}) \cdot 6 \text{ seconds}}{\text{elapsed time [seconds]}}) \cdot 100$$

- Autonomy increases if the ration between human interventions and elapsed time decreases
- If we had 10 interventions in 10 minutes, we would have 90% of autonomy
- For a typical drive in Monmouth Country NJ, the car was autonomous approximately **98%** of the time. Ona multi-lane divided highway in Garden State Parkway, **100%**.

Demo on the road

Self-Driving 3D car driven by a CNN

- The 3D **environment** in Unity



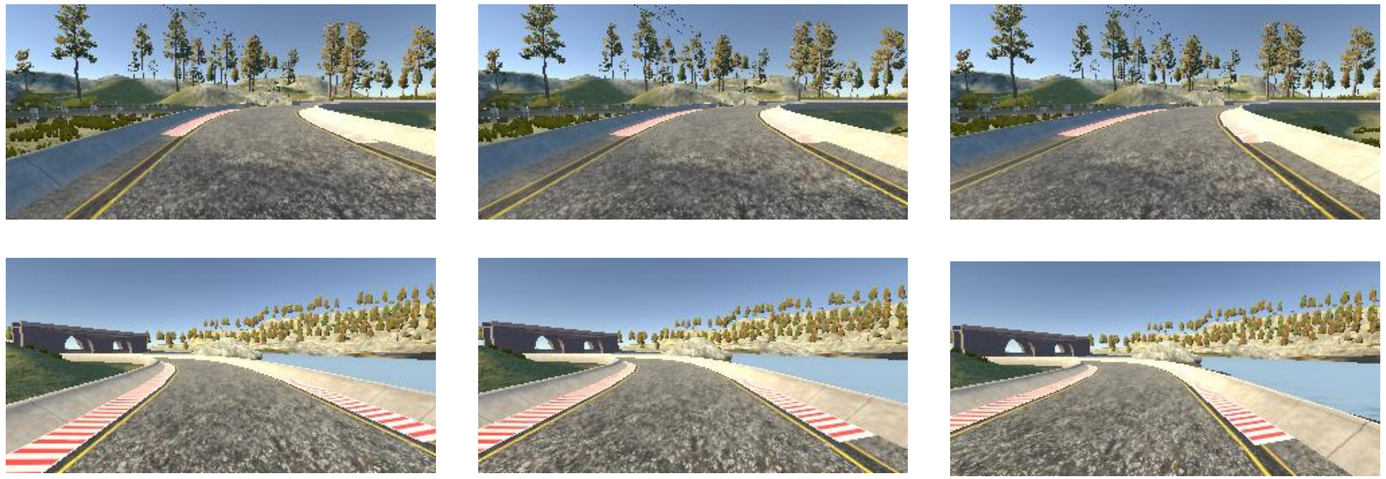
credits: https://assetstore.unity.com/packages/3d/environments/roadways/lake-race-track-55908

Self-Driving 3D car driven by a CNN

- The 3D **environment** in Unity

Dataset generation 1/2

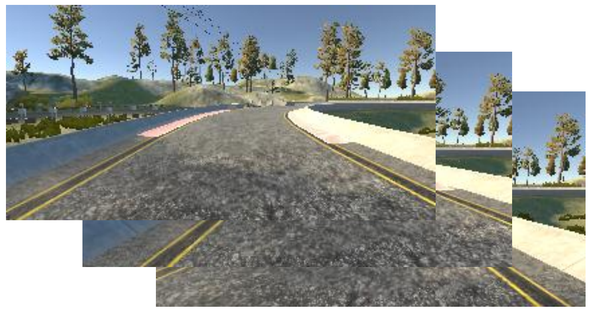
- Three **cameras** at constant distance on the *x* axis are bounded on the car
  - record the environment at 10 FPS
  - captured frames have size 320 x 160 x 3

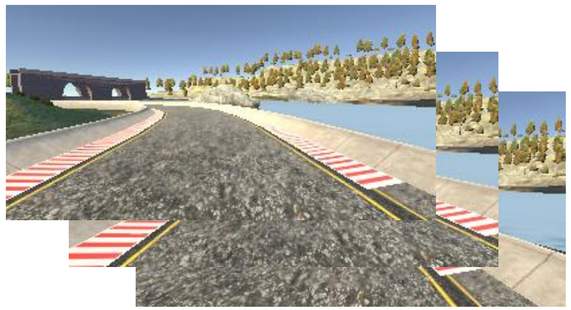


- We record the **steering angle** of the steering wheel
  - steering values in the non discrete range [-1, 1] where 1 = turn right , 0 = go straight, -1 = turn left

Dataset generation 2/2

- The final dataset has the following components





+

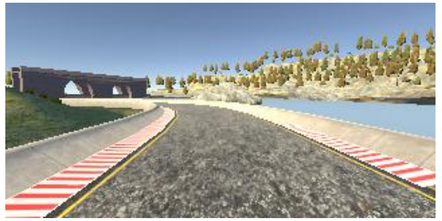

```
0,cam_mid_0.jpg,cam_lef_0.jpg,cam_rig_0.jpg,0.7393
1,cam_mid_1.jpg,cam_lef_1.jpg,cam_rig_1.jpg,0.7593
2,cam_mid_2.jpg,cam_lef_2.jpg,cam_rig_2.jpg,0.7793
3,cam_mid_3.jpg,cam_lef_3.jpg,cam_rig_3.jpg,0.7993
```

- Split dataset into **training set 80%** and **test set 20%**

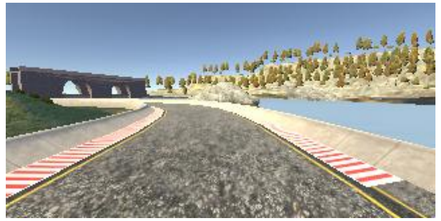
- Train the CNN with supervision by injecting it of **training images,** it produces a **steering angle**, prediction error is back propagated through the net

Dataset augmentation

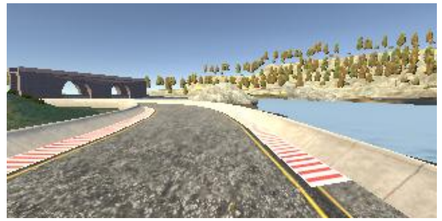
- **Augmentation** phase:
  - **choose one** only image between **left - mid - right** camera view + adjust steering angle



former s.a.: ***-0.5***
new s.a.: ***-0.2***



former s.a.: ***-0.5***
new s.a.: ***-0.5***



former s.a.: ***-0.5***
new s.a.: ***-0.8***

  - **randomly flip** and **slide** images + adjust steering angle

former s.a.: ***+0.2***

new s.a.: ***-0.2***

former s.a.: ***+0.2***

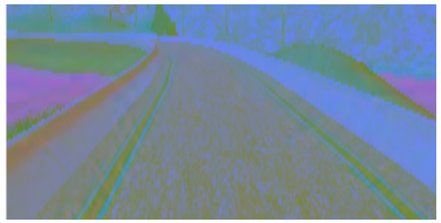
new s.a.: ***-0.1***

Dataset preprocessing

- **Preprocessing** phase:
  - **crop** and **interpolate** the input image in order to remove the sky and useless street information



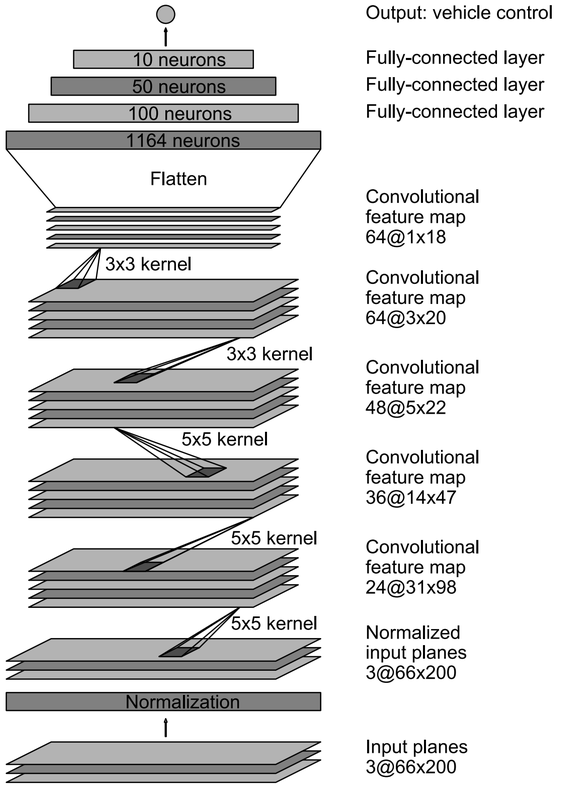
  - apply **YUV filter** to make the image simpler for learning but same feeling of human eye

Keras implementation of the model

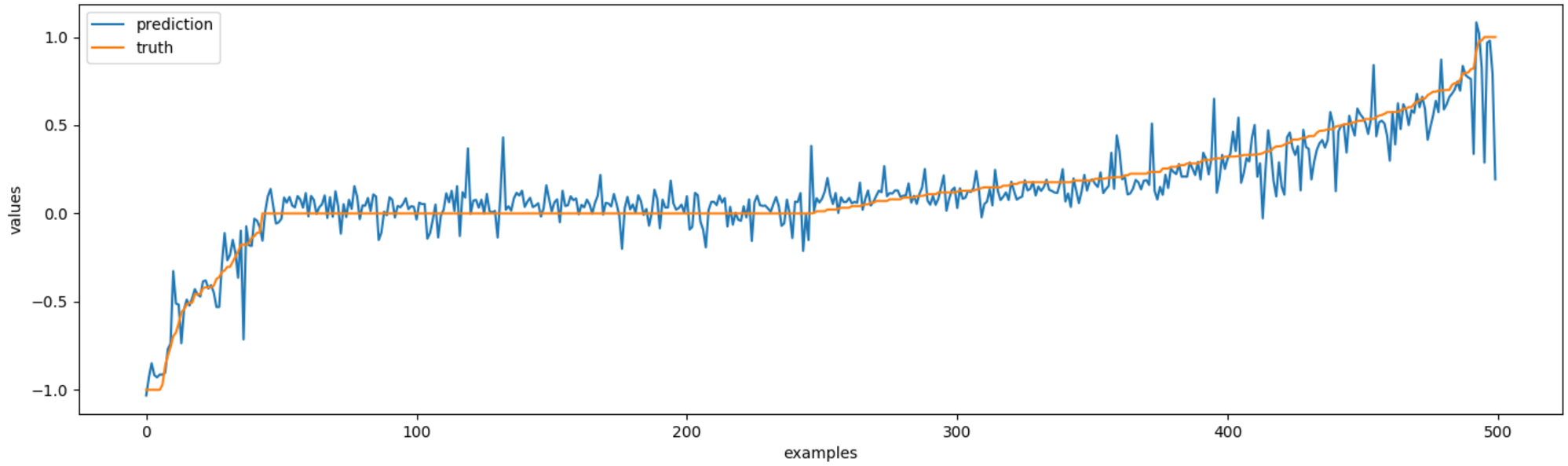
- We used **OpenCV** and **Keras** respectively for image preprocessing and neural implementation

```
def build_neural_model():
    """
    Build NVIDIA model:
    """

    INPUT_SHAPE = (IMAGE_HEIGHT, IMAGE_WIDTH, IMAGE_CHANNELS)

    model = Sequential()
    model.add(Lambda(lambda x: x / 127.5 - 1.0, input_shape=INPUT_SHAPE))
    model.add(Conv2D(24, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(36, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(48, 5, 5, activation='elu', subsample=(2, 2)))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Conv2D(64, 3, 3, activation='elu'))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(100, activation='elu'))
    model.add(Dense(50, activation='elu'))
    model.add(Dense(10, activation='elu'))
    model.add(Dense(1))
    model.summary()

    return model
```



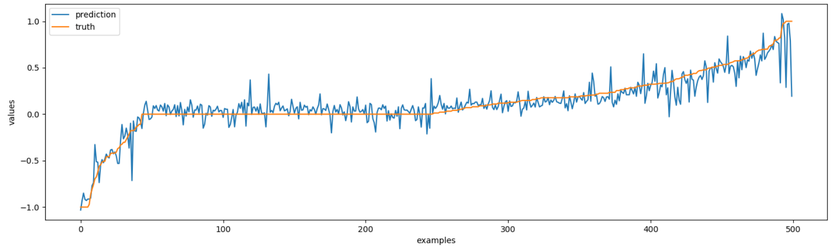
Evaluation

- Dataset generated by driving for **6 laps** using a **joypad**
- Training for **2 hours** (9000 instances, 300 epochs) with a GPU until loss reached its minimum
- Here's how well the CNN approximates the steering angle on 500 examples



Evaluation

- Dataset generated by driving for **6 laps** using a **joypad**
- Training for **2 hours** (9000 instances, 300 epochs) with a GPU until loss reached its minimum
- Here's how well the CNN approximates the steering angle on 500 examples



- Here's what the network could see in an intermediate convolutional layers

Demo

Conclusion

- Make parameters tuning with a **grid search** approach
- Train on **diverse roads** with different light and weather conditions so make the model more resilient
- Integration with **sensors,** avoid **obstacles** and much complicated **decision making**
- **Clone now from GitHub here: https://github.com/matteoprata/self-deiving-car-cnn**

Thank you!