deep learning architectures & software

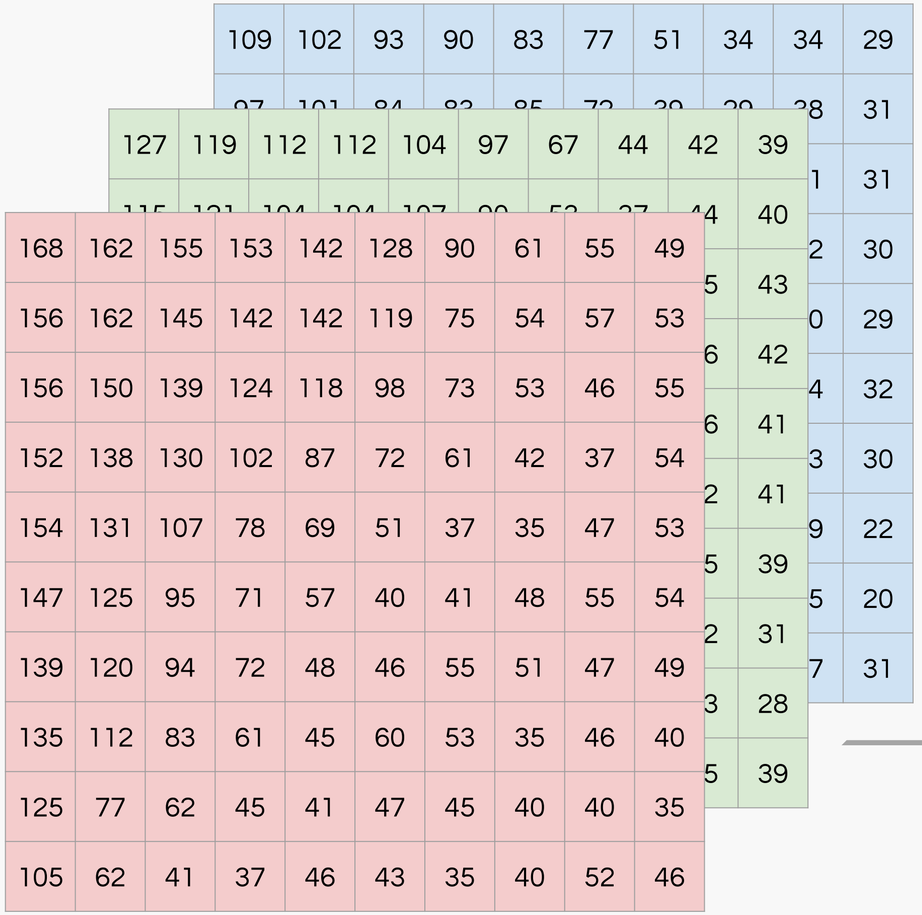
learning to **see**

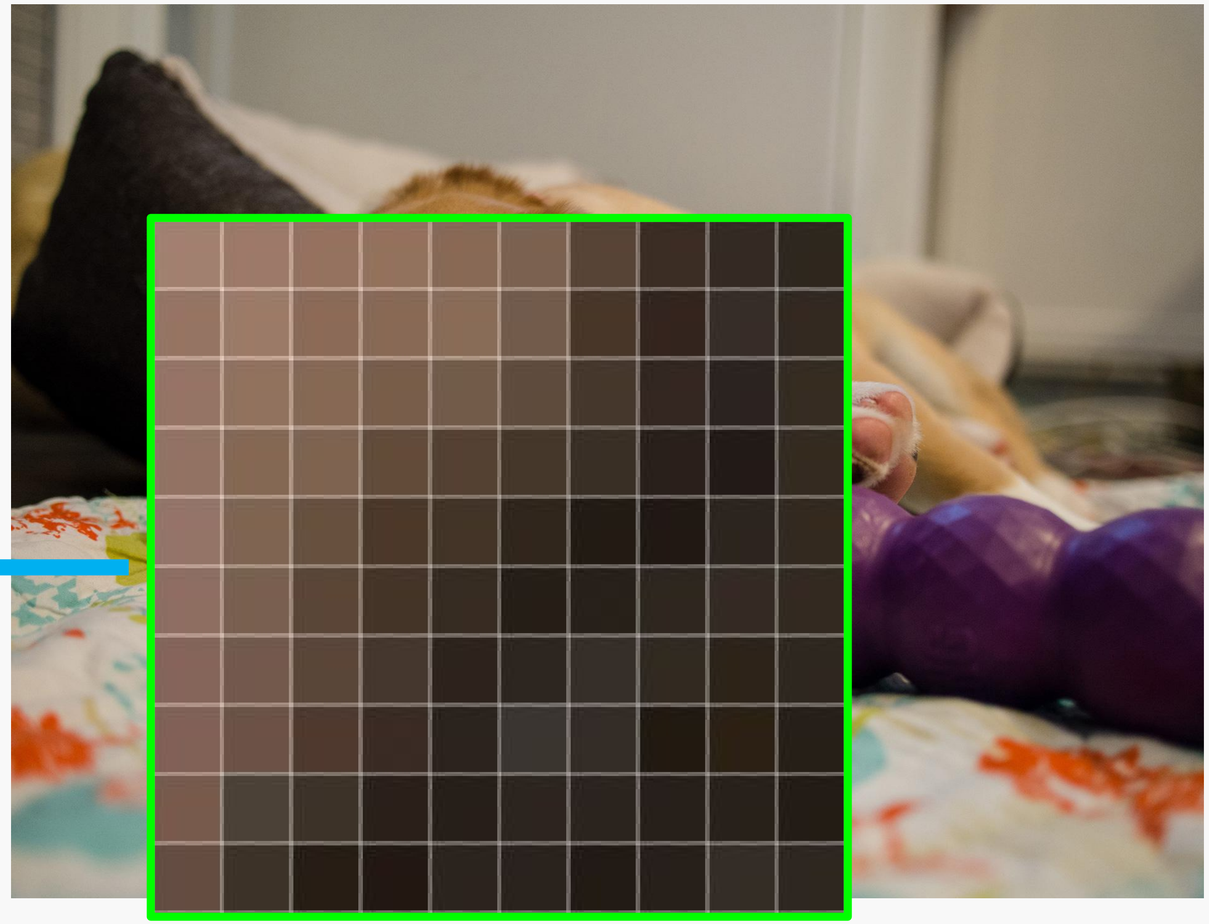
computer vision

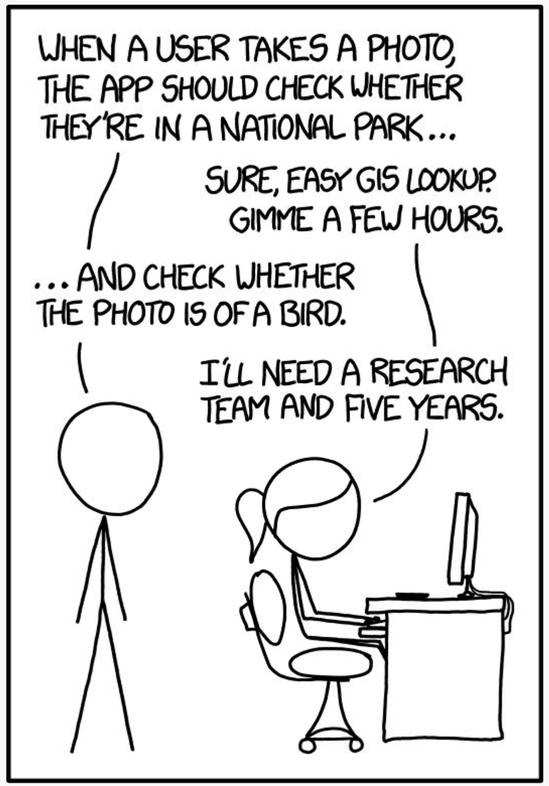
Why is this so **difficult** !?

What a computer “sees”

What we see







IN CS, IT CAN BE HARD TO EXPLAIN THE DIFFERENCE BETWEEN THE EASY AND THE VIRTUALLY IMPOSSIBLE.

8,294,400 pixel values

241	180	147	221	72	77	23	83	213	183	52	94	214	196	138	105	201	17	57	58	165	32	23	251	220	64	30	187	40	136	59	246	123	183	198	246	78	138
72	226	245	255	23	224	45	148	81	27	91	67	40	253	148	33	119	72	116	235	253	194	147	30	221	125	254	37	28	156	179	58	210	7	15	156	110	22
205	89	66	59	240	8	136	217	230	137	143	231	48	5	84	39	191	4	215	18	117	247	136	57	91	13	238	42	121	137	79	243	244	12	73	148	176	164
166	99	103	55	78	67	100	113	55	210	165	5	178	137	181	90	160	91	176	189	133	220	1	68	16	162	236	78	7	70	91	229	96	187	101	29	147	244
141	162	42	237	251	136	18	12	244	248	195	173	212	49	241	18	88	176	255	178	201	2	10	3	148	163	72	106	193	223	57	27	152	174	101	243	180	160
241	110	64	173	84	226	145	211	133	107	82	227	100	193	19	91	131	50	244	208	197	136	249	97	158	178	243	95	57	84	195	150	33	2	205	166	199	72
139	220	33	171	29	48	141	34	104	3	45	96	147	252	29	179	52	39	11	157	234	143	237	174	142	93	6	44	131	242	143	87	80	137	62	172	202	83
202	95	31	222	75	243	234	205	234	237	197	199	171	221	195	191	248	226	144	30	211	9	35	41	214	160	82	19	186	207	26	198	209	167	108	211	176	35
219	26	130	170	217	167	178	127	112	118	207	12	48	190	213	100	239	186	77	207	90	221	227	96	168	106	173	48	72	205	93	186	70	194	211	144	183	98
216	208	28	84	172	191	63	120	95	254	69	182	140	139	19	250	199	232	123	76	253	53	197	104	93	49	234	103	105	221	202	8	139	208	32	231	191	230
227	41	12	212	39	186	75	55	118	203	95	97	200	111	219	47	59	172	79	75	118	126	54	127	238	42	134	129	7	61	16	173	253	185	215	240	25	150
133	124	128	24	187	154	2	51	72	35	185	0	170	167	106	185	154	189	192	238	133	199	207	30	56	76	162	233	87	146	73	16	152	49	180	121	232	1
39	93	21	91	77	60	141	52	42	98	223	232	132	242	238	16	114	221	31	228	181	30	238	57	112	251	238	13	100	96	200	208	208	120	148	245	101	55
89	173	164	61	124	241	142	232	115	17	172	97	53	116	187	47	148	79	23	94	141	207	195	106	211	155	43	112	249	197	184	53	26	81	191	144	47	19
212	153	246	18	158	210	149	68	10	208	48	230	104	40	253	249	18	252	176	64	91	132	137	83	9	236	16	139	133	12	81	21	208	145	43	68	229	219
94	228	28	92	162	54	90	254	135	5	223	240	253	33	39	78	224	210	254	104	163	75	251	9	234	164	151	98	148	216	98	85	110	251	142	92	190	158
98	76	23	203	91	180	230	128	44	117	58	243	75	151	187	115	208	105	57	69	184	159	41	231	18	53	38	255	96	153	129	92	231	84	147	231	159	160
27	187	0	149	113	9	73	244	253	205	47	164	7	135	193	43	181	190	237	51	107	47	132	54	141	42	112	218	64	179	73	254	44	134	228	253	252	61
18	125	83	47	220	228	35	88	25	13	120	43	40	200	83	166	44	77	190	209	24	165	138	158	157	42	122	30	164	101	53	102	84	159	171	64	85	7
59	90	18	193	111	249	114	48	221	95	18	246	186	122	0	59	164	163	194	222	241	177	50	207	220	182	99	224	17	158	213	120	69	48	47	117	57	198
93	8	21	70	158	69	197	220	137	166	14	45	34	226	36	118	173	178	29	44	27	210	153	193	209	19	30	150	224	191	48	200	146	6	44	170	9	220
234	7	102	157	16	240	203	37	8	135	107	218	7	170	49	89	0	94	104	26	233	31	243	149	5	220	3	73	222	197	65	92	61	203	86	199	143	184
189	6	181	196	140	57	42	39	22	231	190	58	196	136	17	5	135	80	216	64	145	73	162	249	10	177	194	40	190	164	64	119	79	142	4	19	141	111
6	83	203	153	214	65	159	114	221	193	75	123	55	192	241	112	35	152	237	45	22	40	29	70	32	171	124	193	52	248	166	128	191	175	134	233	235	181
2	156	89	151	19	27	10	92	89	72	62	131	133	240	146	150	230	91	198	213	166	23	187	54	107	208	37	245	40	111	171	207	10	244	29	106	190	187
138	83	155	25	107	33	211	176	247	179	175	24	156	227	194	255	149	48	162	30	64	13	92	94	213	253	1	170	156	74	37	238	103	30	181	33	51	169
84	236	235	144	74	182	184	59	71	118	38	211	164	177	235	136	143	128	78	201	149	69	247	199	18	40	140	55	64	110	163	86	234	172	164	18	160	46
96	96	164	129	71	108	128	173	39	124	80	156	203	118	208	208	159	212	135	4	156	182	86	209	144	196	21	167	207	125	254	219	199	241	49	107	239	130
35	31	137	0	131	40	206	247	178	221	171	49	169	242	21	148	189	54	192	77	88	134	134	166	178	237	66	91	56	205	22	132	247	229	243	176	229	40
48	246	203	107	167	50	131	152	225	211	201	163	109	85	69	152	63	51	136	211	213	195	119	242	118	217	59	135	15	94	146	72	222	189	186	208	123	219
117	243	183	45	167	78	216	18	241	112	248	77	127	178	137	254	101	8	137	196	44	148	6	91	140	85	238	42	34	112	31	142	94	21	148	17	2	179
116	27	95	102	0	130	15	63	96	228	1	201	81	152	163	198	17	221	173	131	229	252	65	217	224	116	179	230	100	174	178	75	1	203	234	189	158	47
68	186	36	14	229	218	116	185	125	131	60	85	193	109	12	95	106	60	155	19	75	29	90	242	5	106	122	57	30	166	162	84	137	32	128	75	188	172
50	217	253	101	150	240	221	222	121	180	157	23	145	213	87	9	231	198	19	122	51	77	196	176	126	172	62	245	93	87	32	167	216	55	115	82	22	255
212	187	89	156	227	72	212	177	177	170	61	99	117	98	144	207	112	121	198	240	165	192	5	89	28	255	166	69	109	84	100	234	252	196	109	80	47	38
229	160	104	41	7	172	218	184	217	126	2	22	98	134	191	91	53	40	154	80	93	76	19	127	235	70	3	89	211	65	142	144	199	119	254	10	255	161

1920

1440

convolutional neural networks

Take advantage of **structure** or **spatial** characteristics inherent in your **data**

```
I hear and I forget;
I see and I remember;
I do and I understand
```

TRANSLATION INVARIANCE

dog

classification example

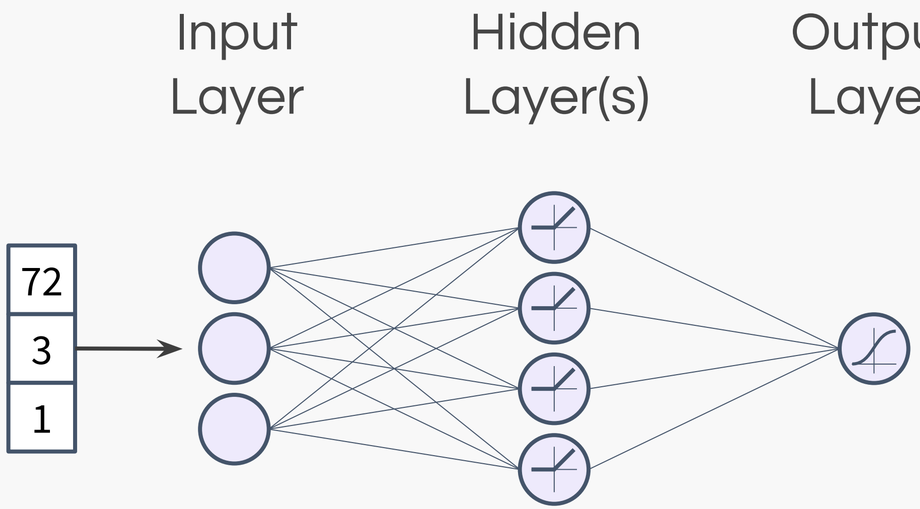
Translation Invariance

Rotation/Viewpoint Invariance

Size Invariance

Illumination Invariance

ConvNets can **vastly** reduce the **number** of **parameters** in the network



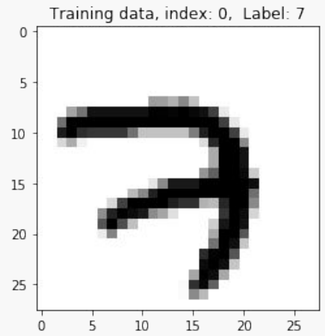
12 Weights + **4** Weights = **16** Weights

784
Input
Nodes

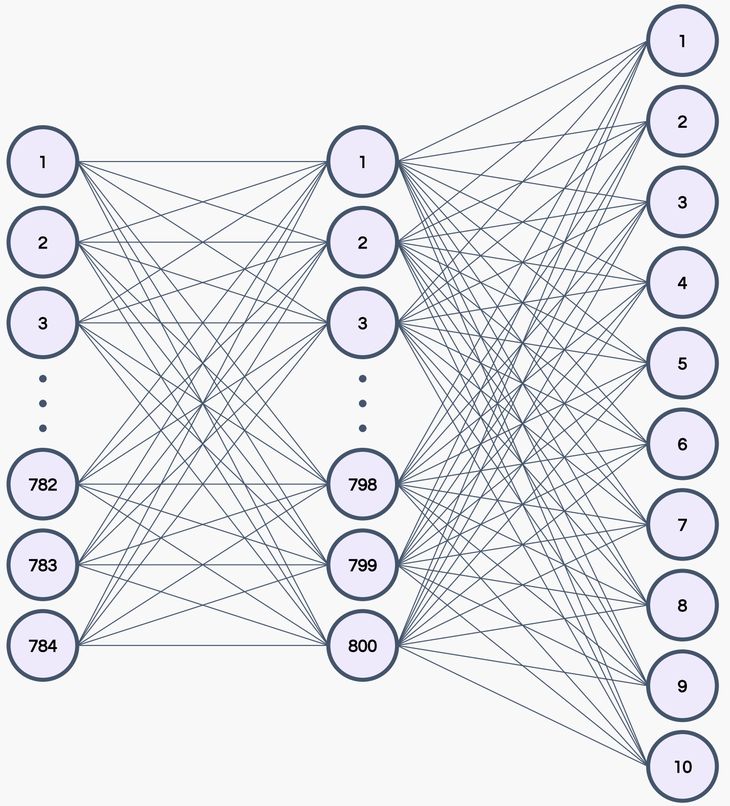
800
Hidden
Nodes

10
Output
Nodes

MNIST Dataset



28 x 28 x 1



627,200 Weights + **8,000** Weights = **635,200** Weights

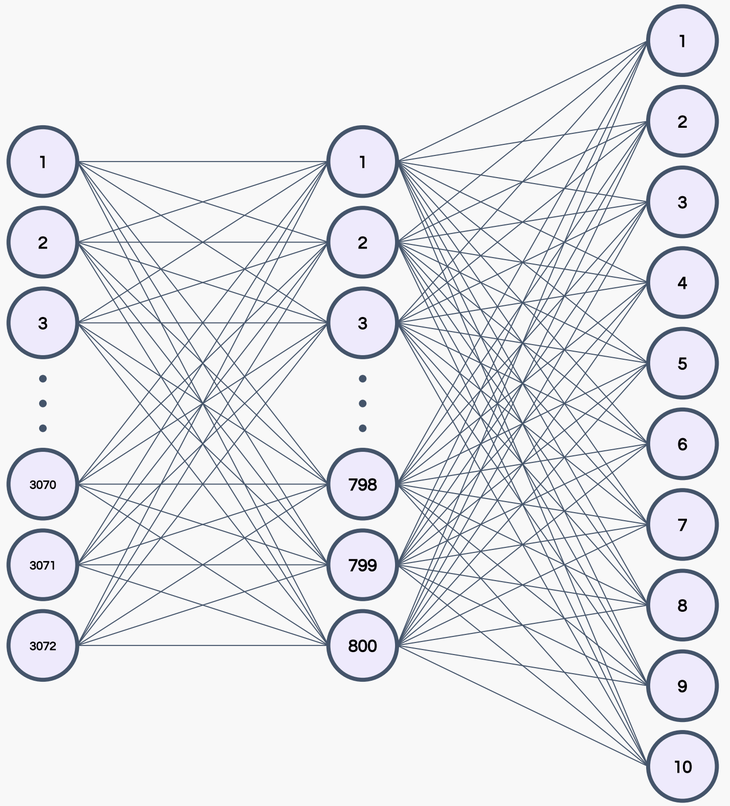
3,072
Input
Nodes

800
Hidden
Nodes

10
Output
Nodes

CIFAR-10 Dataset

32 x 32 x 3



2,457,600 Weights + **8,000** Weights = **2,465,600** Weights

Driving Dataset

455 x 256 x 3

349,440 Input Nodes

40 Hidden Nodes

1 Output Node

13,977,640

Weights

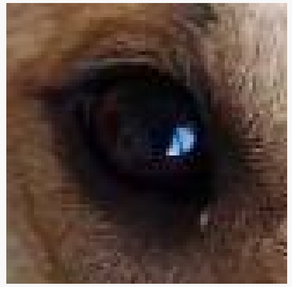
filters & weight sharing

inputs contain structure

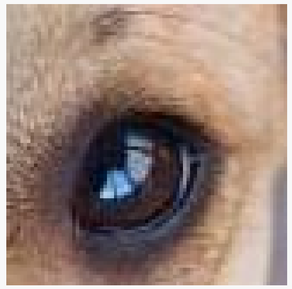
relative proximity matters

order matters

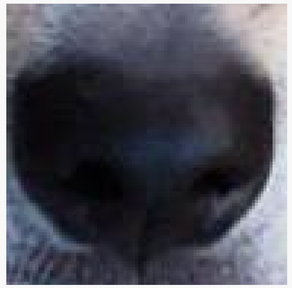
Classify this Image:



eye



eye



nose

fur

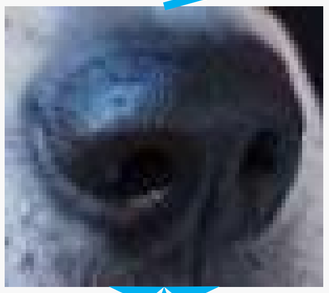
Going One Step Further

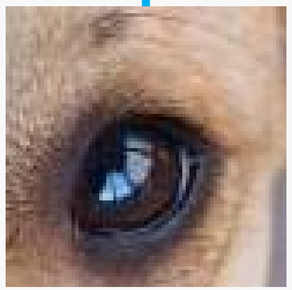
How do we know
what a nose is?

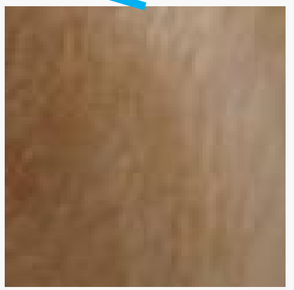
Final Layer

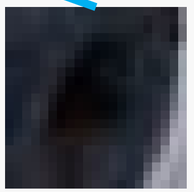
Layer 2

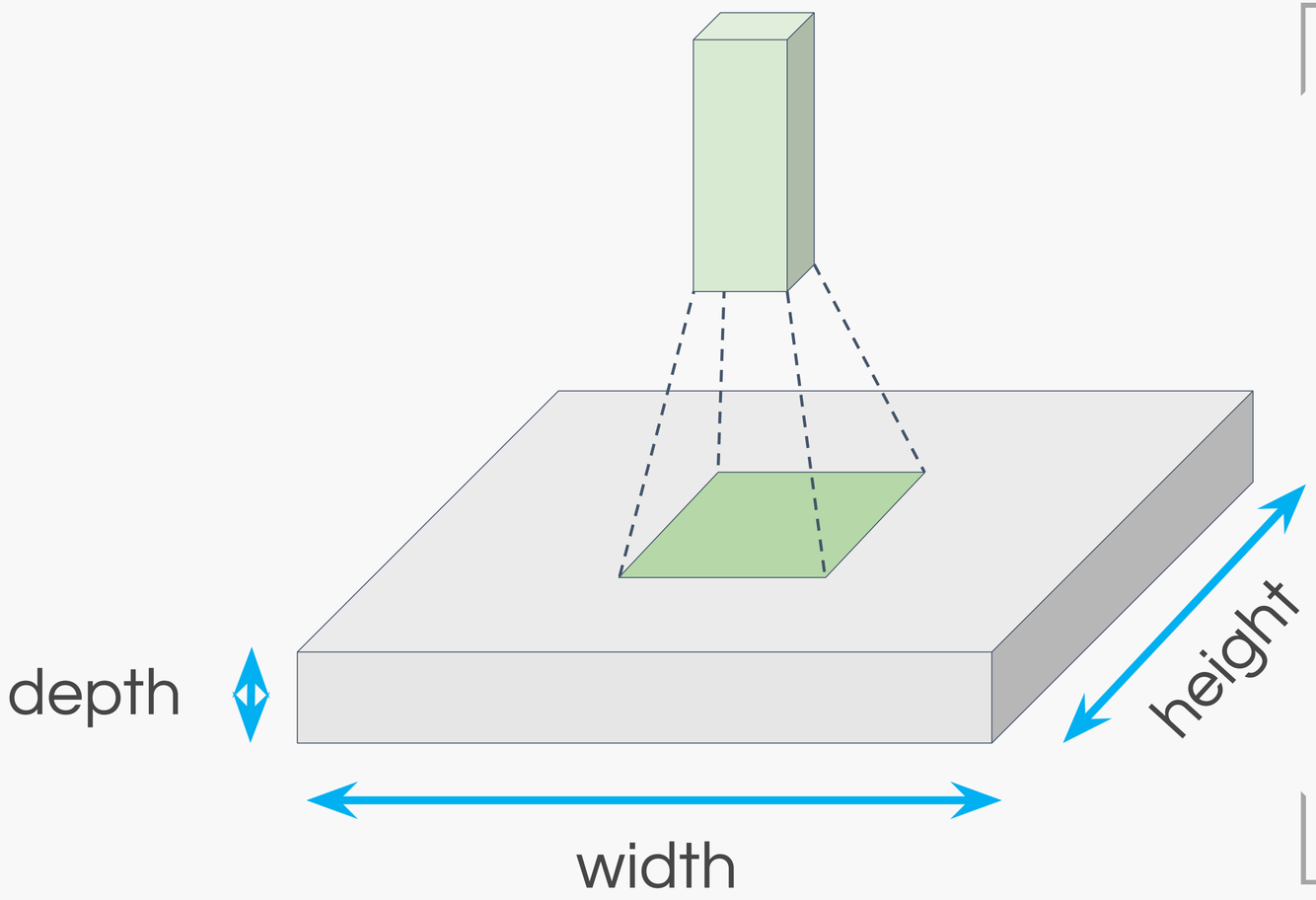
Layer 1



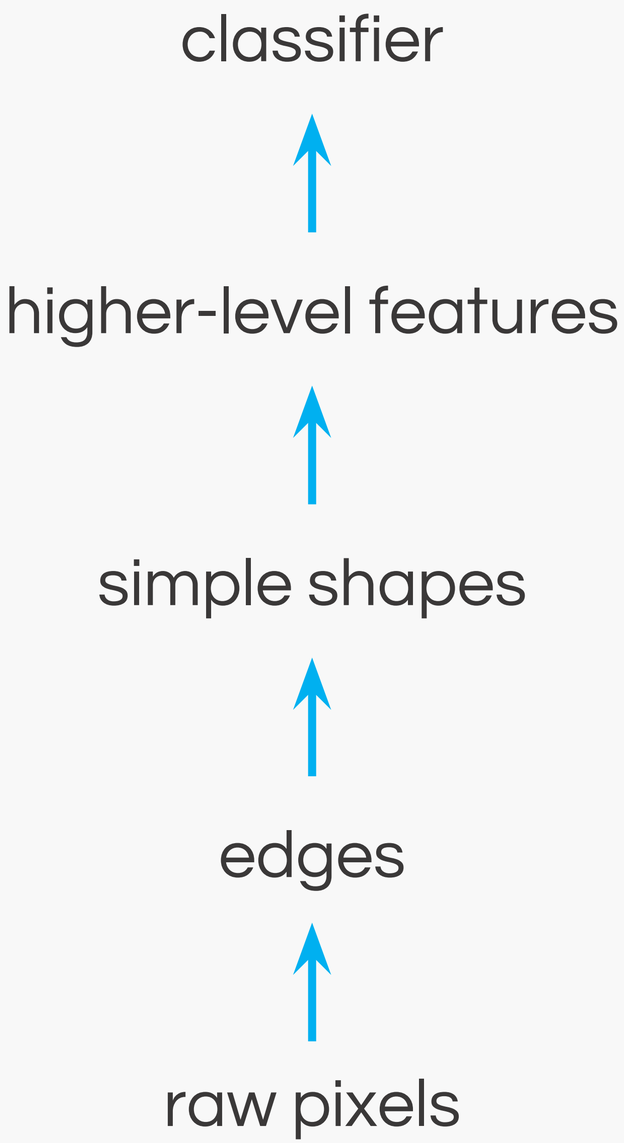






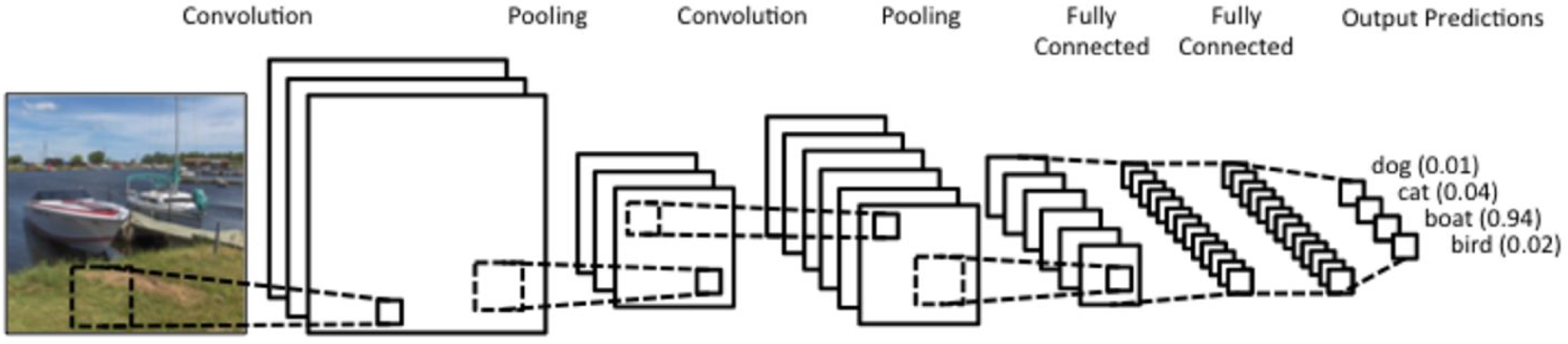


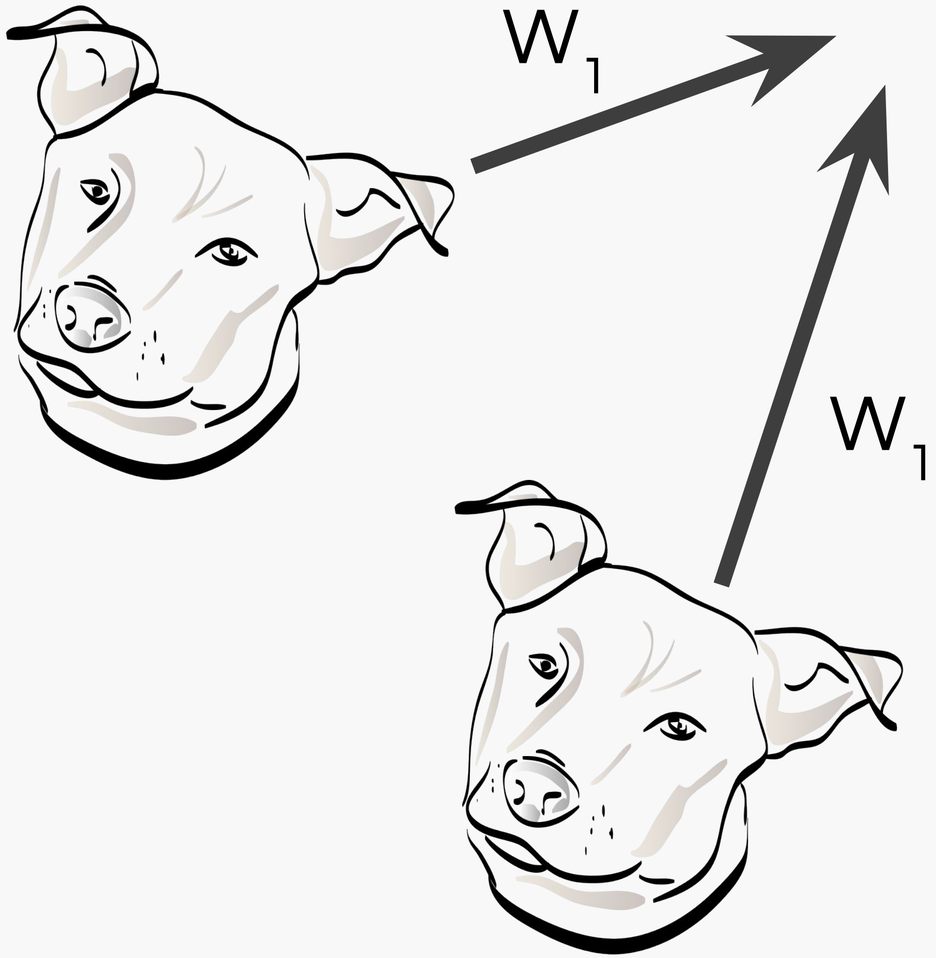
filters


classifier
higher-level features
simple shapes
edges
raw pixels

filters

Example CNN


Convolution
Pooling
Convolution
Pooling
Fully Connected
Fully Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)



weight sharing

https://github.com/frankhinek/Meetup-ADL-17-Apr-10

README.md

Autopilot

Overview

In this project we have a modified implementation of the NVIDIA End-to-End Deep Learning for Self-Driving Cars self-driving car model and this paper. Our implementation is based on the python TensorFlow implementation by Sully Chen.

The project consists of a python-based training script and an inference implementation. A convolutional neural network (CNN) is trained to map raw pixels to steering commands. The image frames were captured by a dash-mounted video camera. At the end of the training, the model is saved. This saved model is then loaded by the inference implementation to evaluate what the model predicts for the steering commands.

Dependencies

Python Packages

- OpenCV 3.1.0
- Pillow 4.0.0
- SciPy 0.19.0
- TensorFlow 1.0.1

The commands below will be useful for those using Anaconda/Miniconda and pip to manage Python packages. If you use another method you will have to adapt the steps.

```
conda create -n autopilot python=3.5
source activate autopilot
export TF_BINARY_URL=https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.0.1-py3-none-any.whl
pip install --ignore-installed --upgrade $TF_BINARY_URL
pip install pillow==4.0.0
conda install -c menpo opencv3=3.1.0
conda install -c anaconda scipy=0.19.0
```

Note: Only tested with macOS 10.12.3 and Ubuntu 16.04.2 LTS.

Dataset

The driving_dataset.zip file contains 45,568 JPEG images of video frames that are used to train the model and the observed steering wheel angle readings. It is 2.2 GB compressed, so you'll want to be on fairly fast internet connection before starting the download.

- Download the driving dataset and unzip it into the repository folder.
- The path should be something like `/path/to/repo/Autopilot-TensorFlow/driving_dataset/`.

Usage

Train the Model Using the Prerecorded Training Dataset

The model needs to be trained with the prerecorded driving data. Training the model should create a `save` folder that will contain the saved model and checkpoint files. We will use the saved model for inference.

- `python train.py` to train the model
- To visualize the training performance using Tensorboard use `tensorboard --logdir logs`, then open http://0.0.0.0:6006/ in your web browser.

Evaluate the Model Using the Prerecorded Training Dataset

The `save` directory contains a model that was already trained using the JPEG and steering wheel readings dataset that you downloaded earlier. When run two windows will be launched displaying the video frames captured during the drive and the steering wheel angle predicted by the pre-trained model.

- `python run_dataset.py` to run the model on the dataset

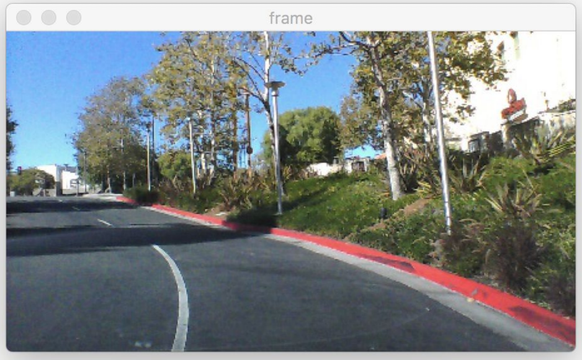
Evaluate the Model Using a Live Webcam Feed

If you have a dash-mounted video camera that is connected to your computer you can use `run.py` to evaluate the trained model using the captured frames to see what steering wheel positions it predicts. I have not tested this code that was forked from Sully Chen's repository, so you may need to make some modifications.

- `python run.py` to run the model on a live webcam feed

Credits

This repository was forked from SullyChen/Autopilot-TensorFlow, modified to support TensorFlow 1.0, and expanded with additional documentation. Full credit for the original code goes to Sully Chen. Additional code and comments from Tomi Malla's repository.



Code Example

Modified implementation of the NVIDIA self-driving car paper based on a TensorFlow implementation by Sully Chen.

Includes a pre-trained model you can run on a system without requiring GPUs to accelerate the training process.

Note: Only tested on Mac and Linux.

Deepen Your Understanding

Stanford CS 231 Notes

http://cs231n.github.io/convolutional-networks/

Brandon Rohrer Video

https://www.youtube.com/watch?v=FmpDIaiMIeA

Christopher Olah Writings

http://colah.github.io/posts/2014-07-Conv-Nets-Modular/

Sources & Credits

Slide No(s).	Source	Link
5	xkcd	http://bit.ly/xkcd-ml
7	LoggaWigler	http://bit.ly/2nAVQmP
8, 21	Leunert	http://bit.ly/2oUHmOV
9	Matt Krause	http://bit.ly/2p1lq4l
11	LeCun, Cortes, Burges	http://bit.ly/2o3x8YH
12	Krizhevsky, Nair, Hinton	http://bit.ly/2onUNFw
13, 22	Sully Chen	http://bit.ly/2p1mNjJ
20	Karpathy, Johnson	http://bit.ly/2oo4VhN
3-5, 15-17	Frank Hinek	http://bit.ly/linkedin-hinek