with tensorflow

구현을 위한 딥러닝

- 고려대학교 물리학과 한승희
- mod96@naver.com

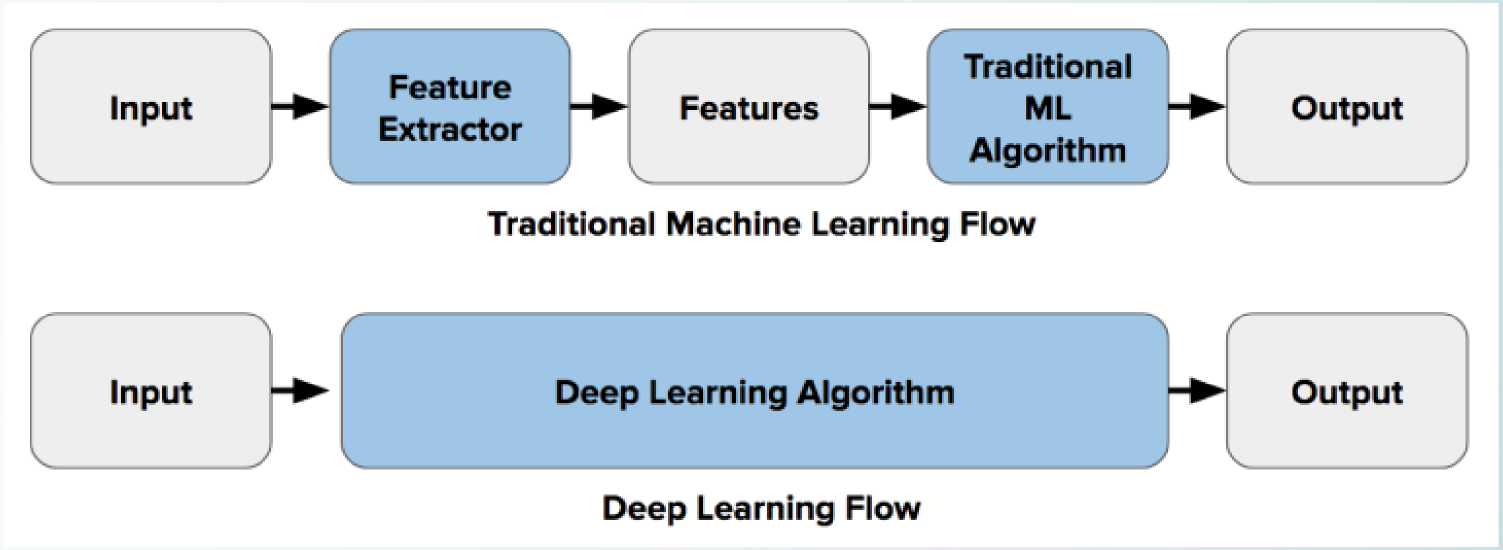
Contents

1 Traditional CV
2 Convolutional Layer
3 Why CNN?
4 Bias - Variance Tradeoff
5 Useful Things
6 Implementation

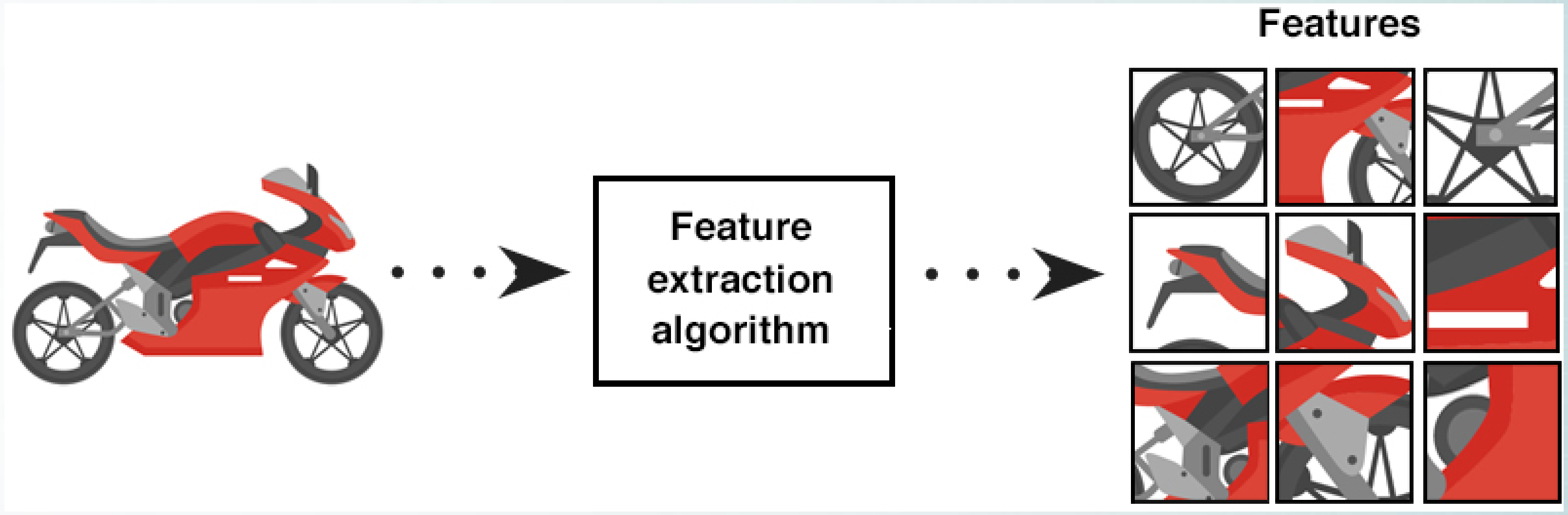
Contents

1 **Traditional CV**
2 Convolutional Layer
3 Why CNN?
4 Bias - Variance Tradeoff
5 Useful Things
6 Implementation

Traditional vs DL in CV



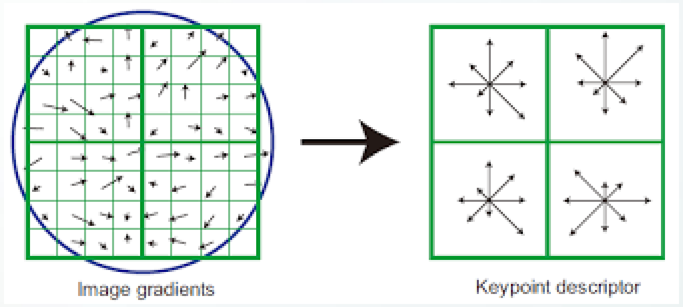
Feature?

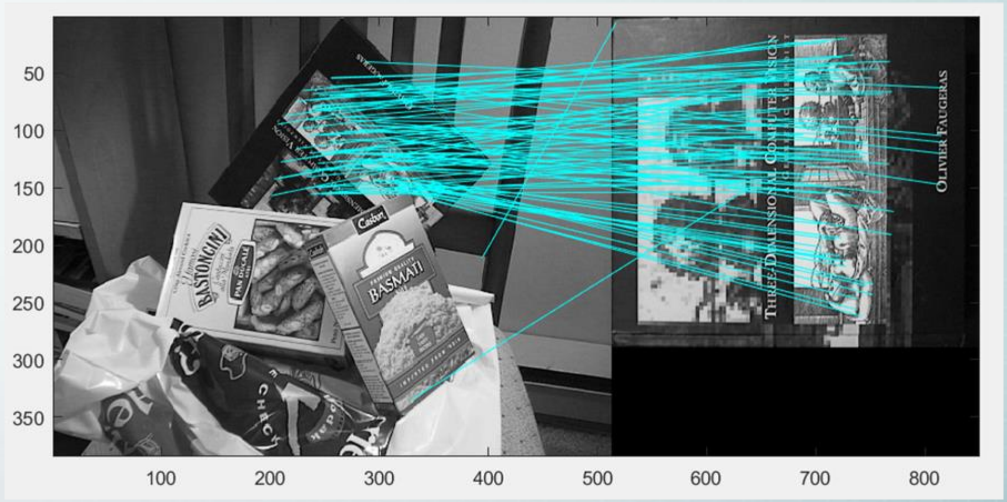


❖ EX> Color, Global Shape(PCA space), Local shape(shape context), Texture(Filter banks)

Feature?

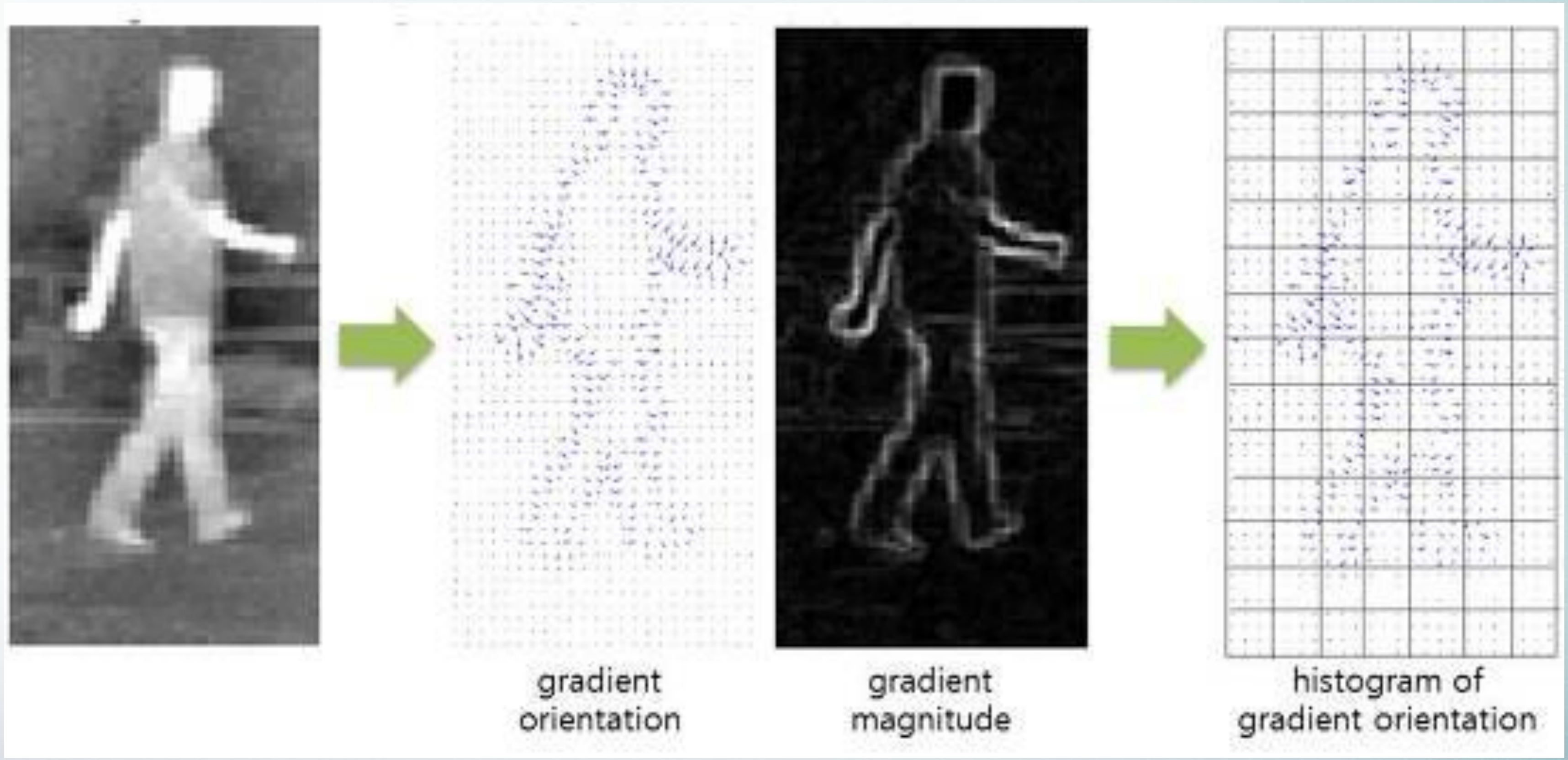
- SIFT, SURF, BRIEF, ORB, …





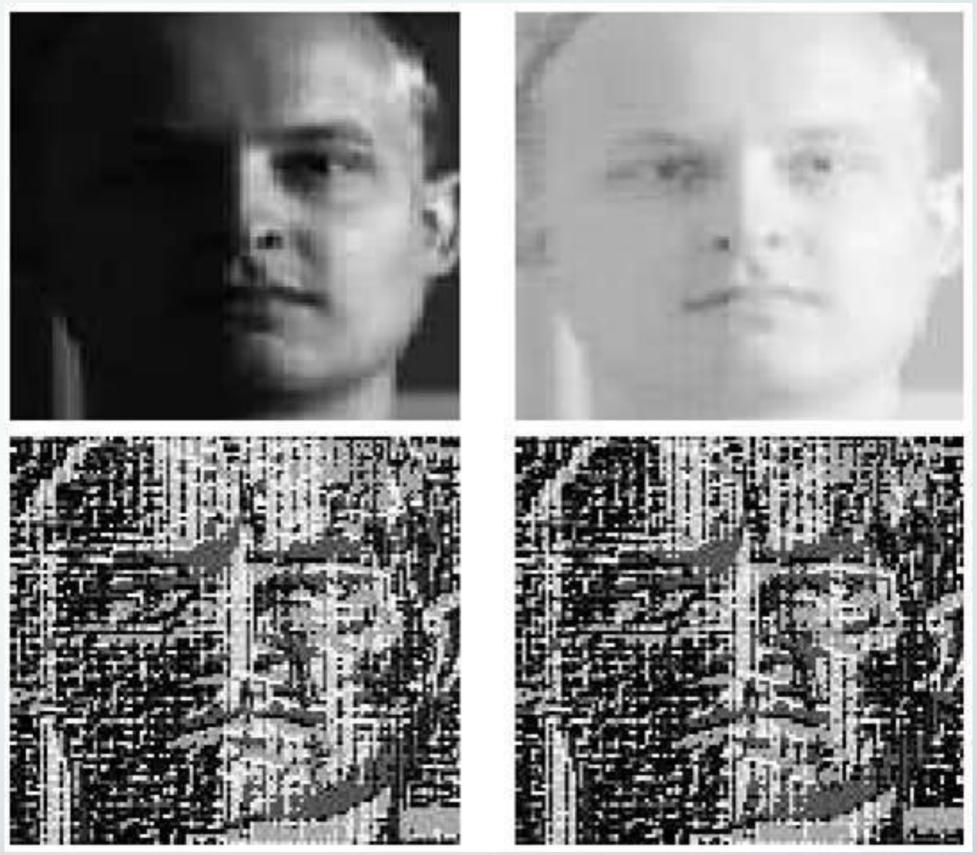
Feature?

- HOG, …


gradient orientation
gradient magnitude
histogram of gradient orientation

Feature?

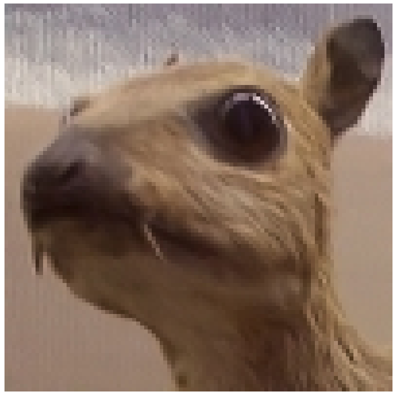
- MCT, Haar feature, LBP, …



Feature?

- Convolution

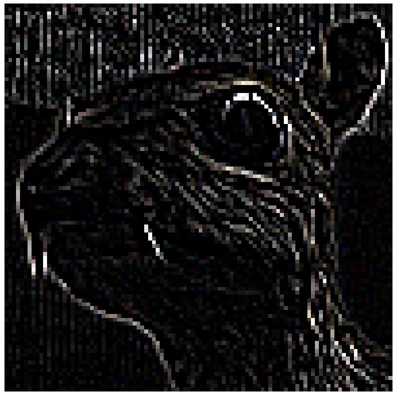
Input image



Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map



- https://en.wikipedia.org/wiki/Kernel_(image_processing)

Contents

1 Traditional CV

2 **Convolutional Layer**

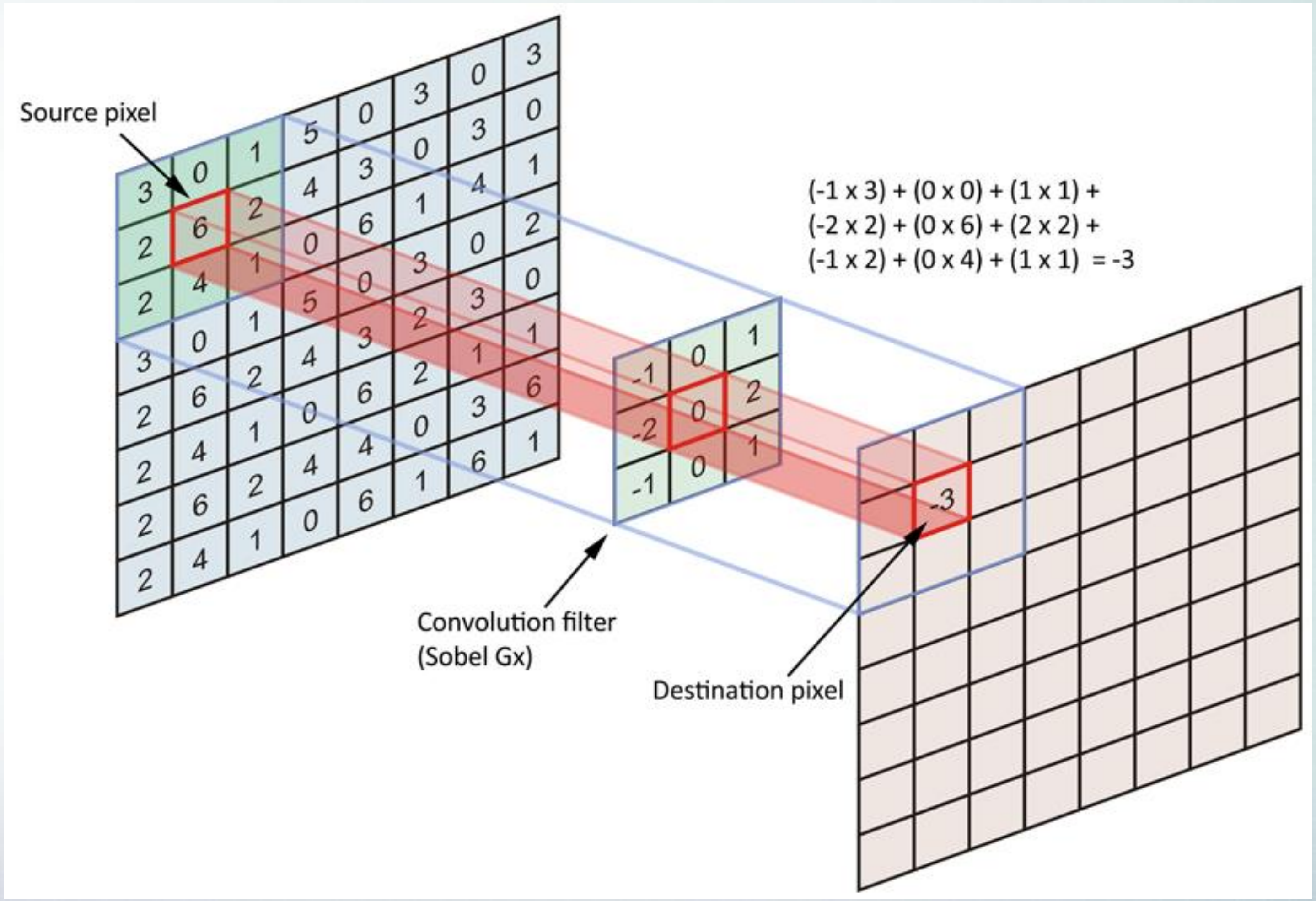
3 Why CNN?

4 Bias - Variance Tradeoff

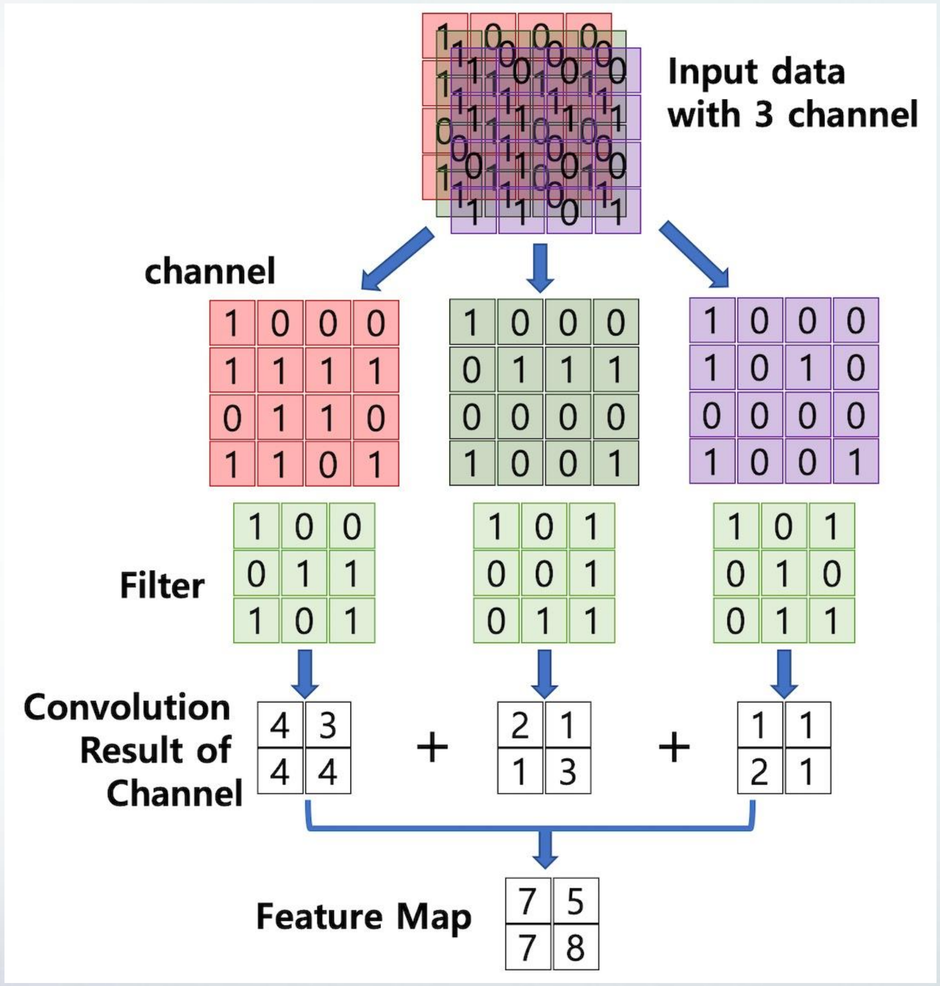
5 Useful Things

6 Implementation

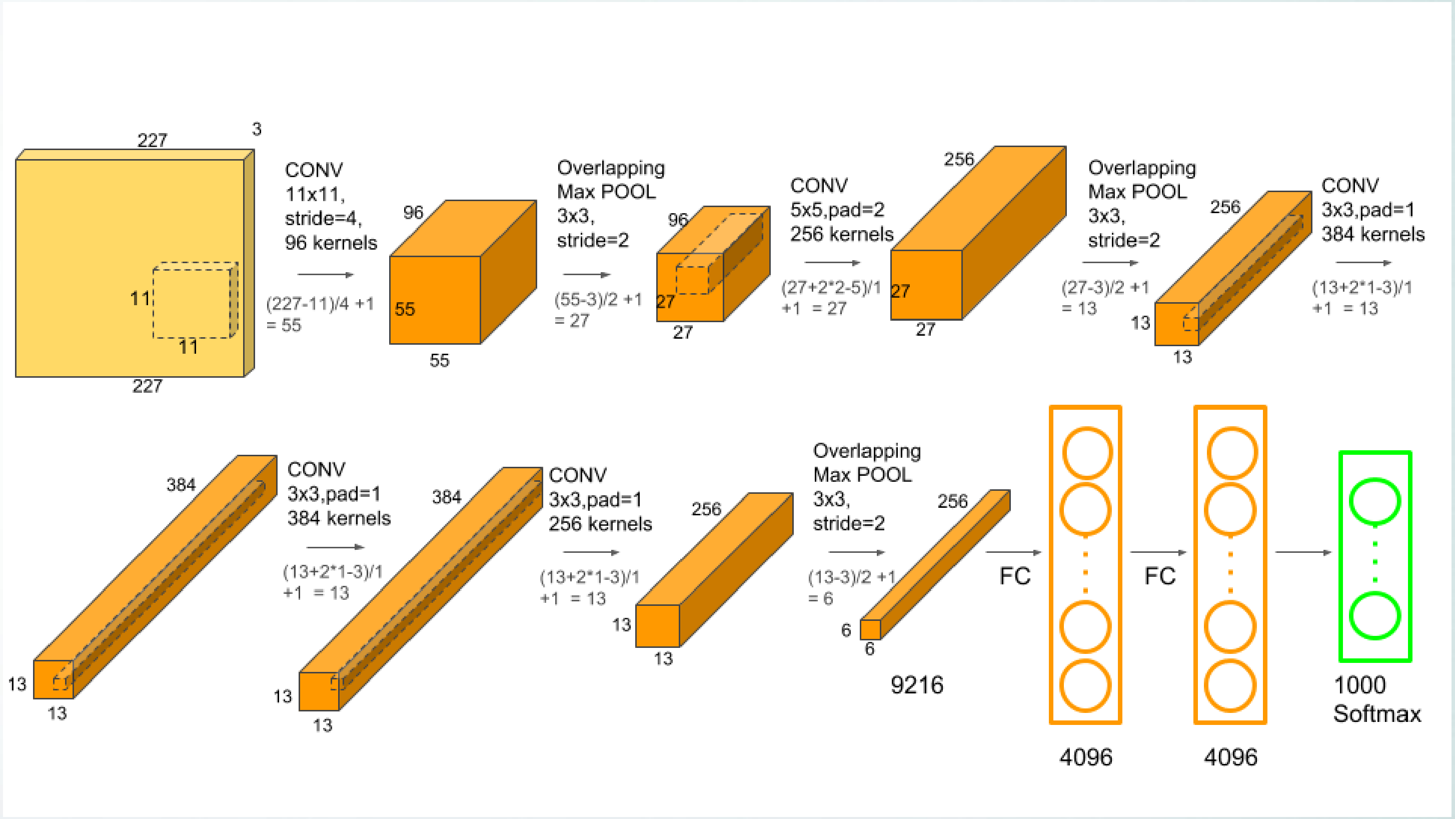
Kernel (filter), Stride, Padding


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Kernel (filter)



Example



227
3
11
11
227
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

Company Logo

Contents

1 Traditional CV

2 Convolutional Layer

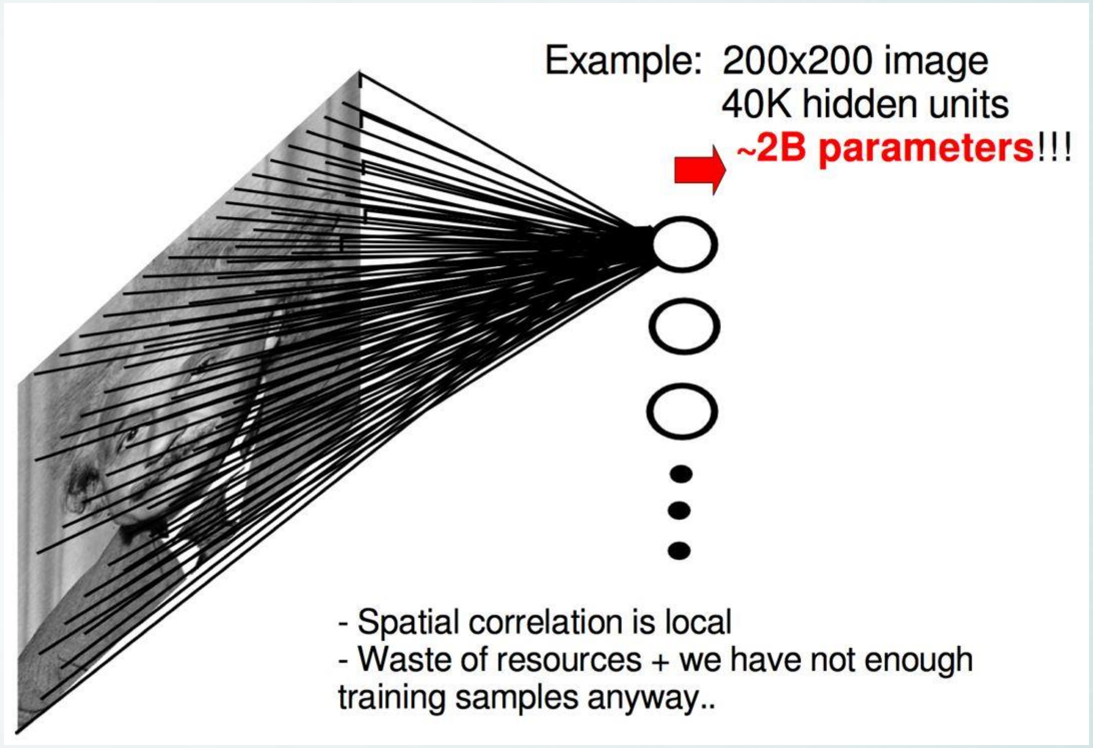
3 **Why CNN?**

4 Bias - Variance Tradeoff

5 Useful Things

6 Implementation

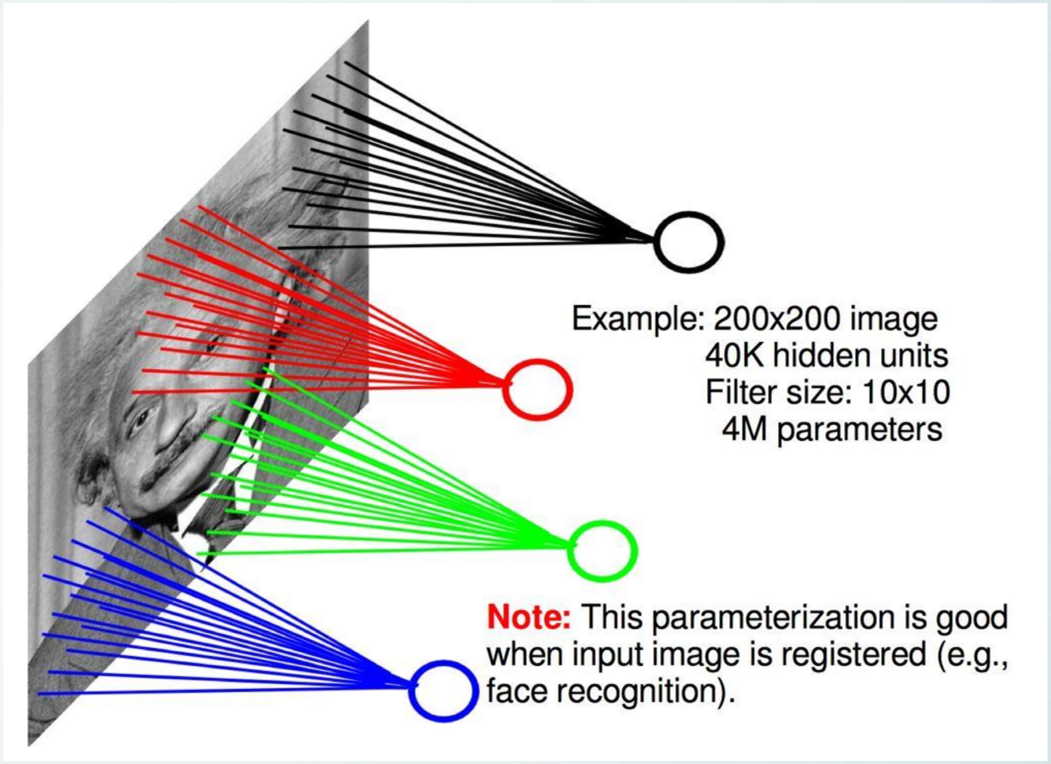
If Fully Connected(Dense)...



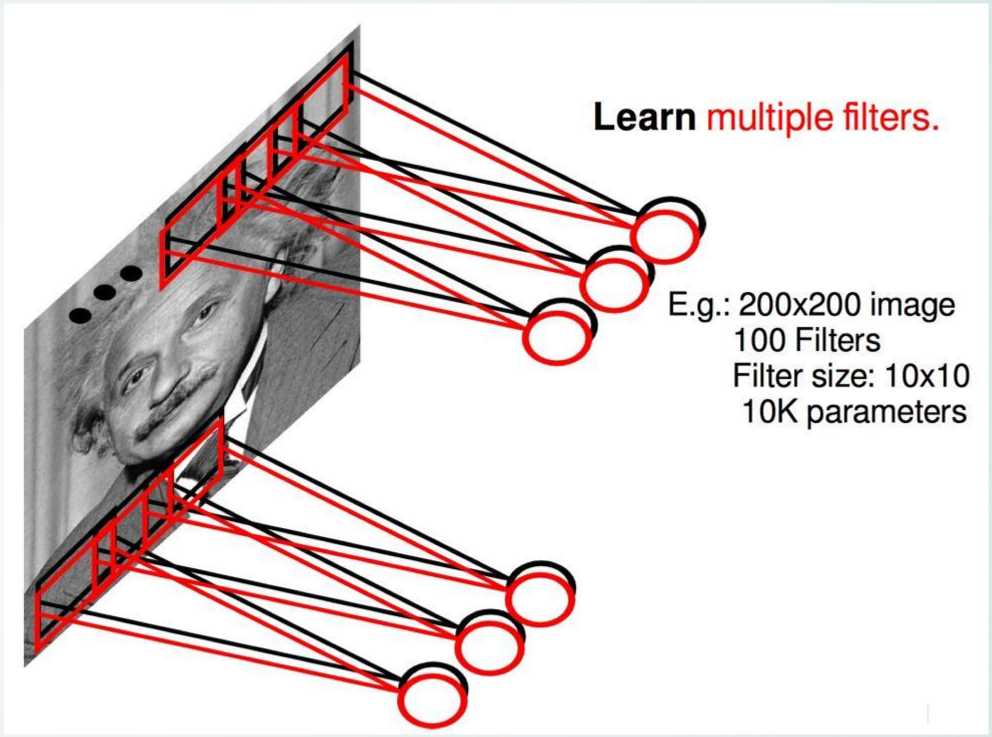
Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..

If Locally Connected


Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters
Note: This parameterization is good when input image is registered (e.g., face recognition).

CNN


Learn multiple filters.
E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Company Logo

Contents

1 Traditional CV

2 Convolutional Layer

3 Why CNN?

4 Bias - Variance Tradeoff

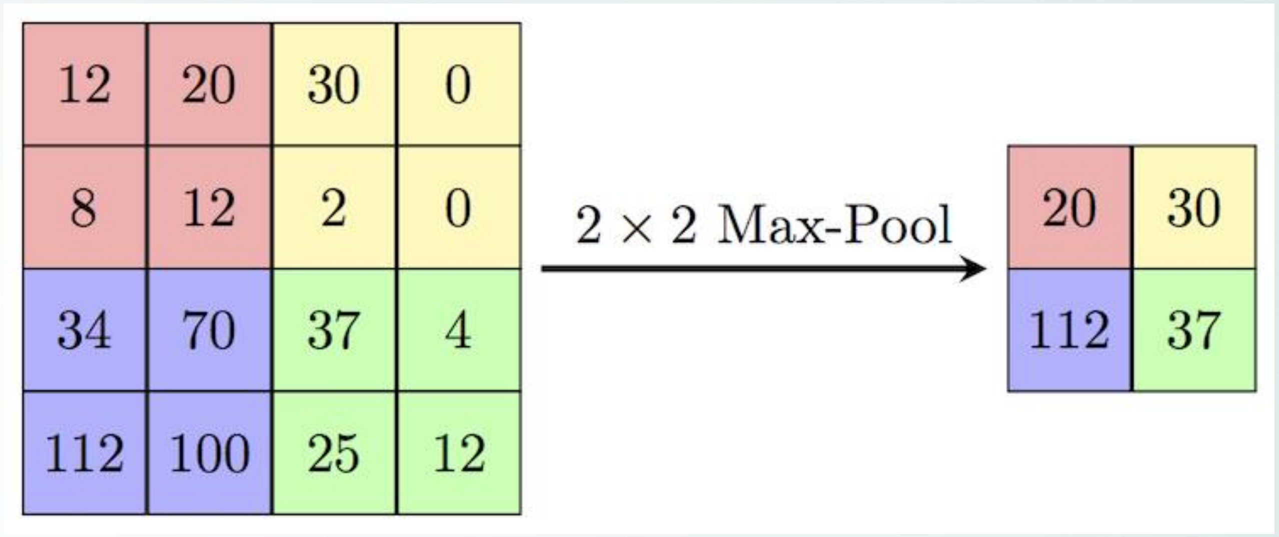
5 Useful Things

6 Implementation

As before..

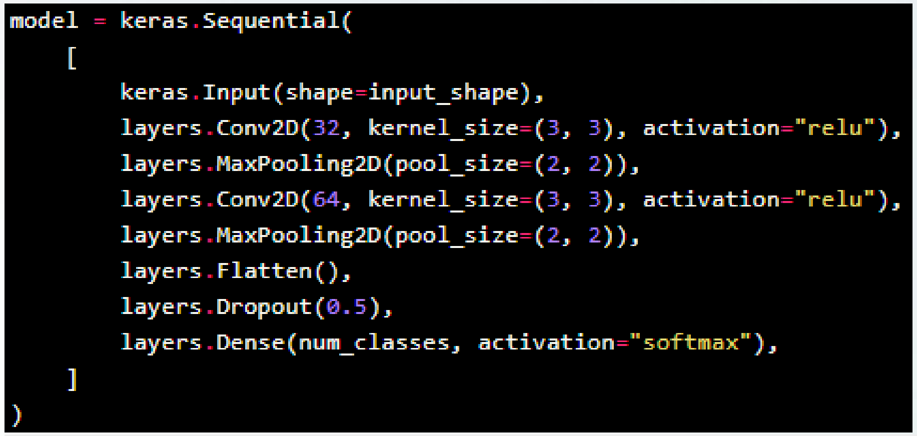
- Batch Normalization
- Dropout
- Weight Initialization methods
- Regularization
- …

Pooling

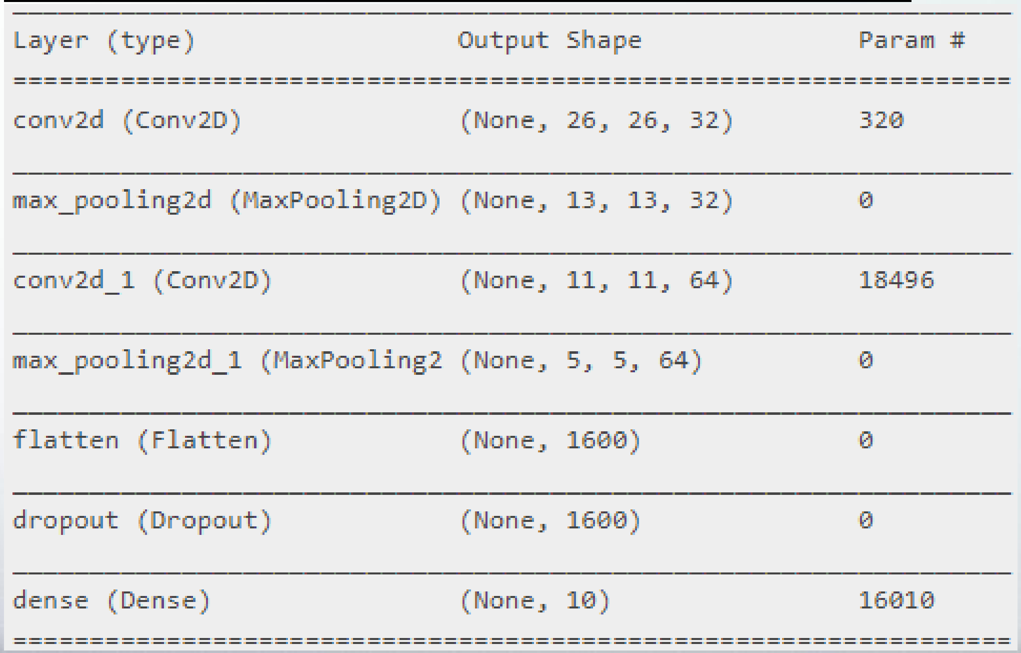


- MaxPool, AveragePooling
- GlobalMaxPool, GlobalAveragePooling

Sample Calculation



```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320

_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0

_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496

_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0

_________________________________________________________________
flatten (Flatten)            (None, 1600)              0

_________________________________________________________________
dropout (Dropout)            (None, 1600)              0

_________________________________________________________________
dense (Dense)                (None, 10)                16010
=================================================================
```

Company Logo

Contents

1 Traditional CV

2 Convolutional Layer

3 Why CNN?

4 Bias - Variance Tradeoff

5 **Useful Things**

6 Implementation

tf.keras.applications

- https://www.tensorflow.org/api_docs/python/tf/keras/applications

OpenCV

cv2.dnn

ImageNet

https://paperswithcode.com/sota/image-classification-on-imagenet

and more…

tf.keras.preprocessing.image

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

Company Logo

Contents

1 Traditional CV

2 Convolutional Layer

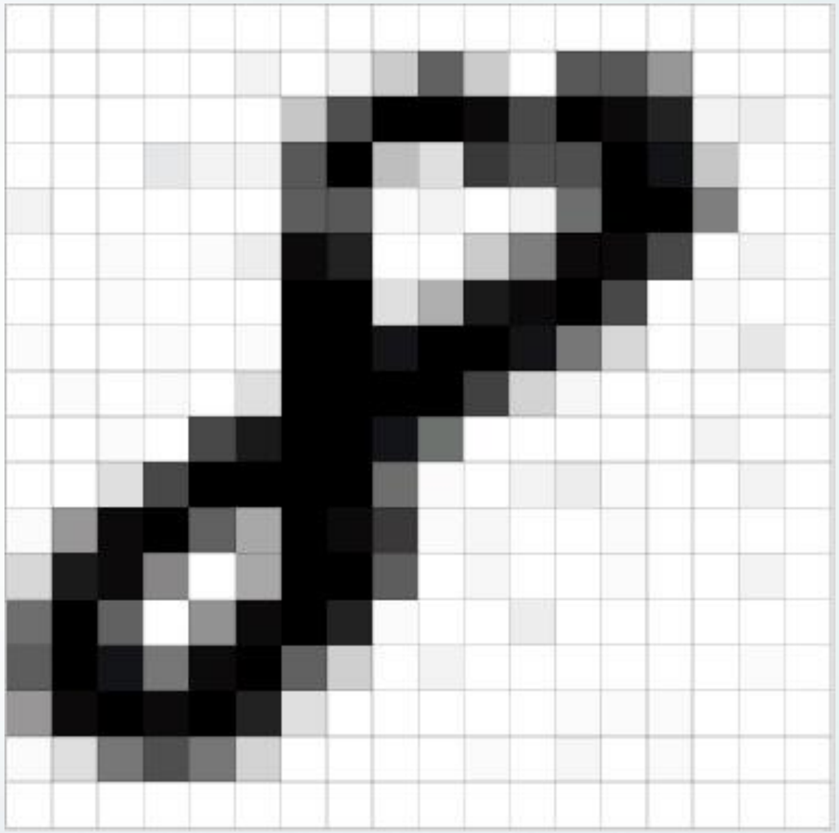
3 Why CNN?

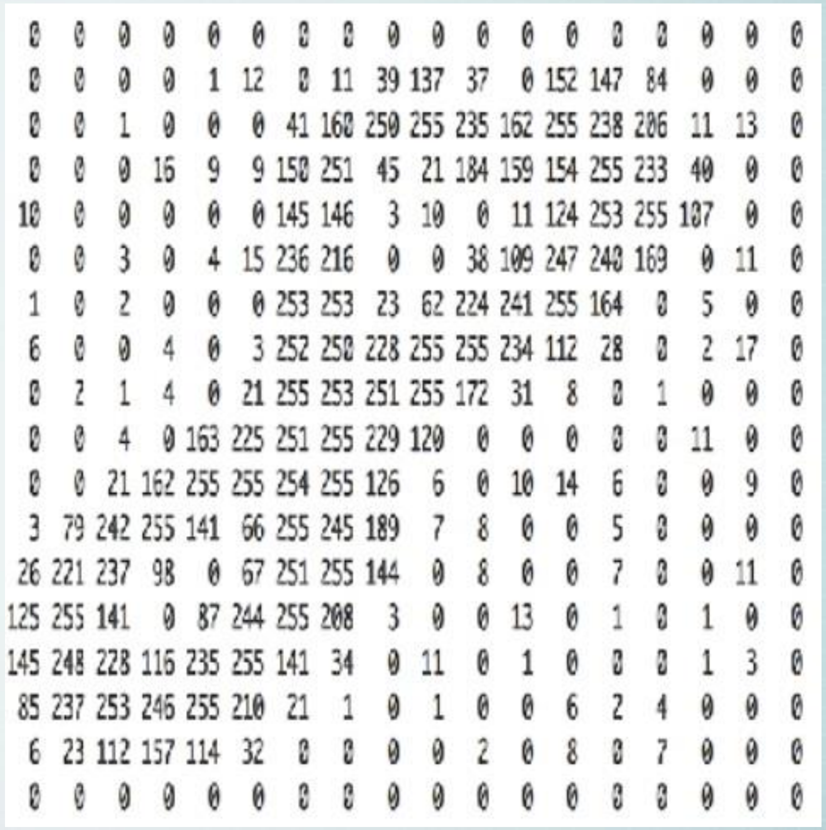
4 Bias - Variance Tradeoff

5 Useful Things

6 **Implementation**

MNIST





0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CIFAR-10