Deep Learning Standalone

–

for Chemistry

1. ML Basic
2. Pytorch Basic
3. MLP with Fingerprint Representation
4. CNN with SMILES Representation
5. GNN with Graph Representation
6. Experiment Management and Hyperparameter Tuning with Tensorboard
7. Practical Tips

Topics to learn in this session

1. Convolutional Neural Network (CNN)

Basic of CNN

2. What is SMILES?

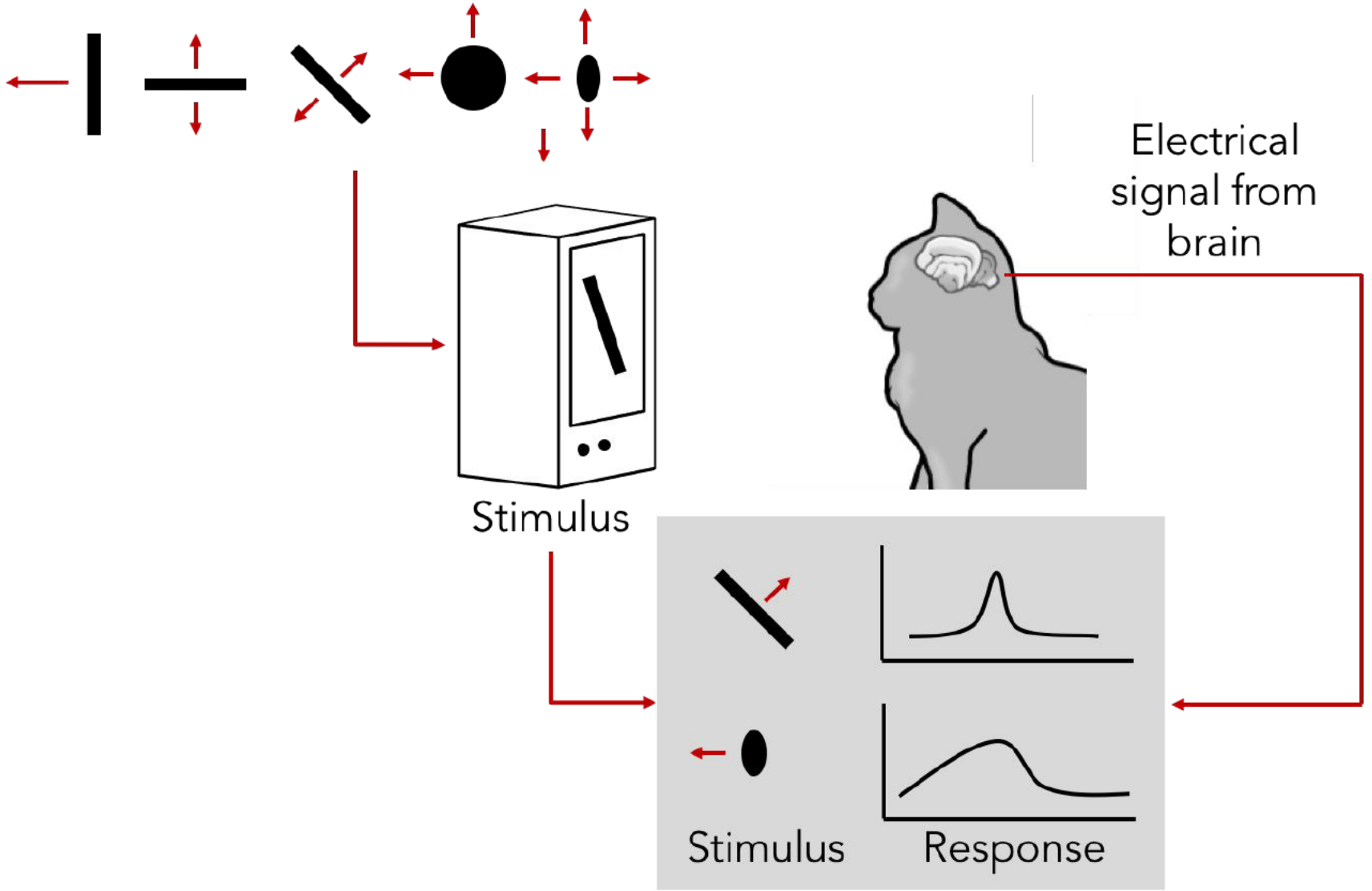
Molecular Representation with Linear String

3. Implementing CNN with pytorch

Predicting lipophilicity value of molecules

What is Convolutional Neural Network?

How human recognize an image?



Hierarchical organization

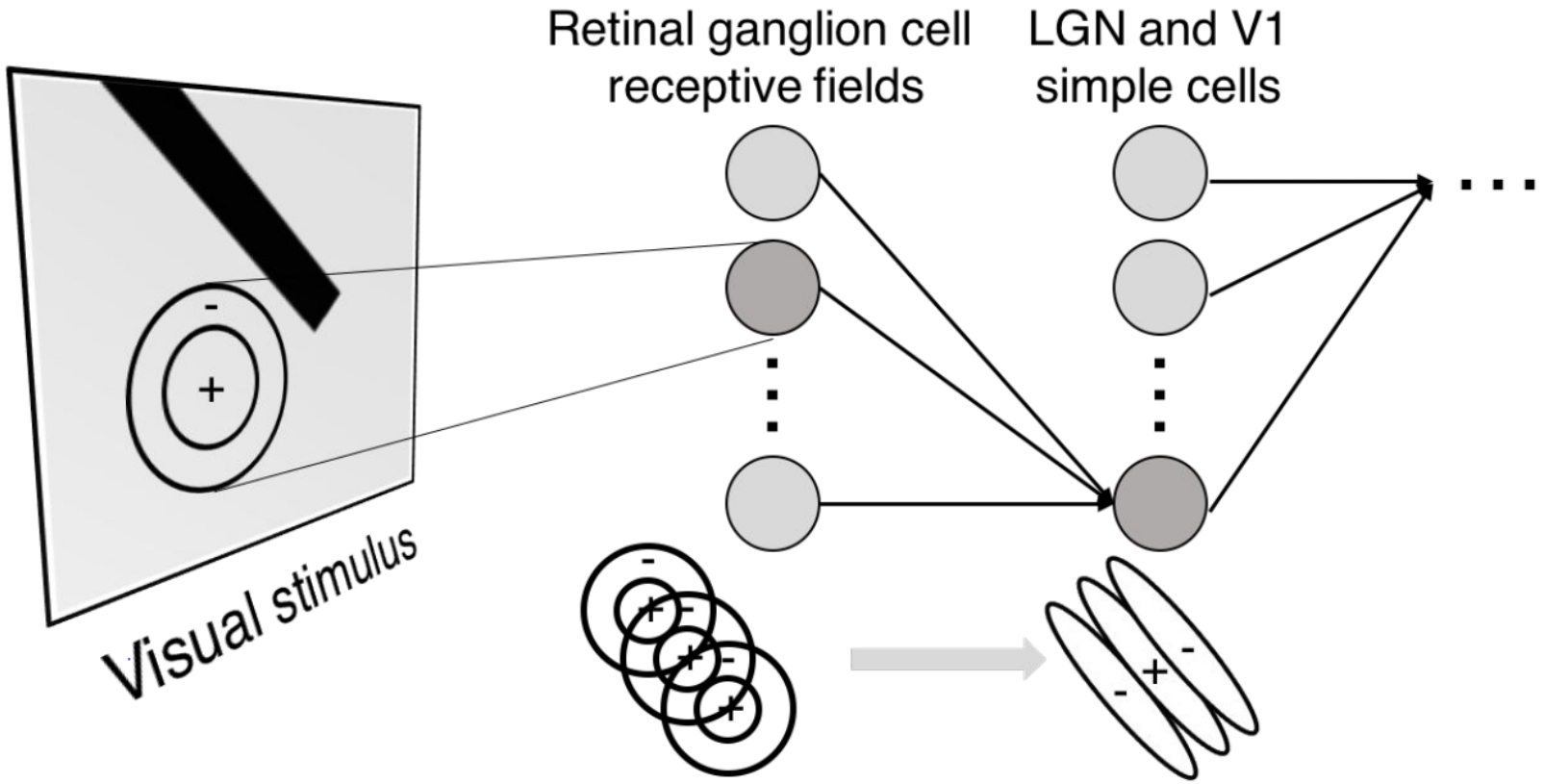
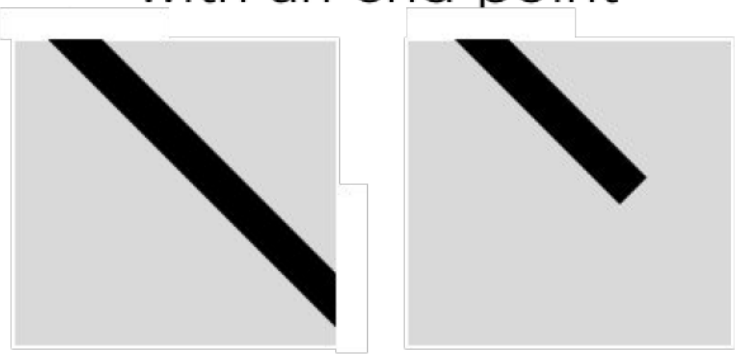


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point

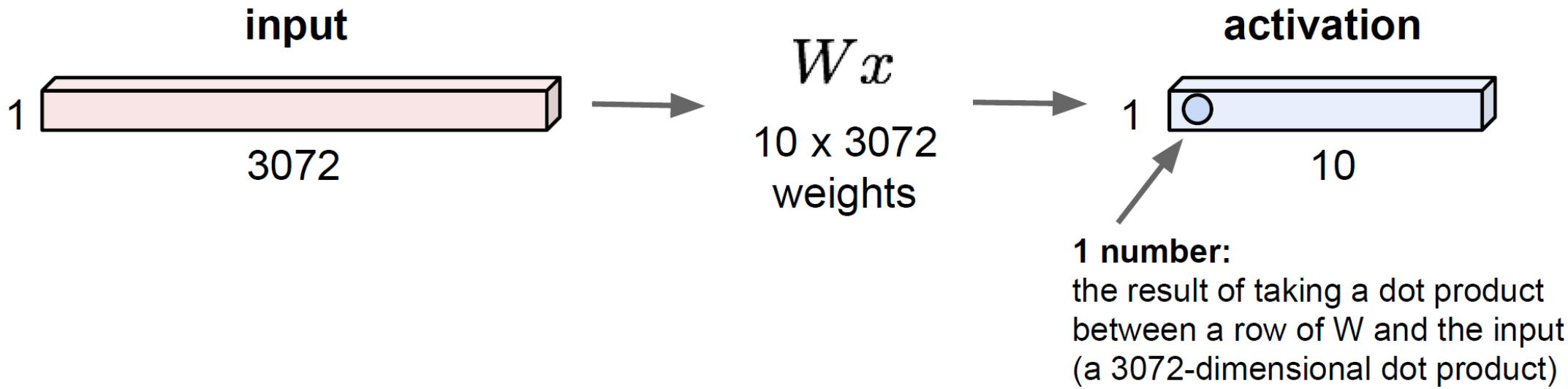


No response

Response (end point)

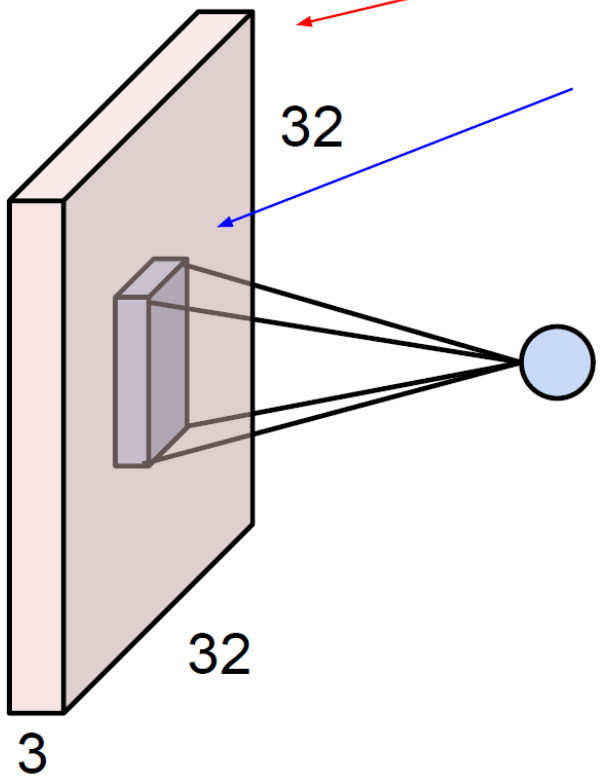
MLP / Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



Convolution Layer

: Preserve the spatial structure



32x32x3 image

5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

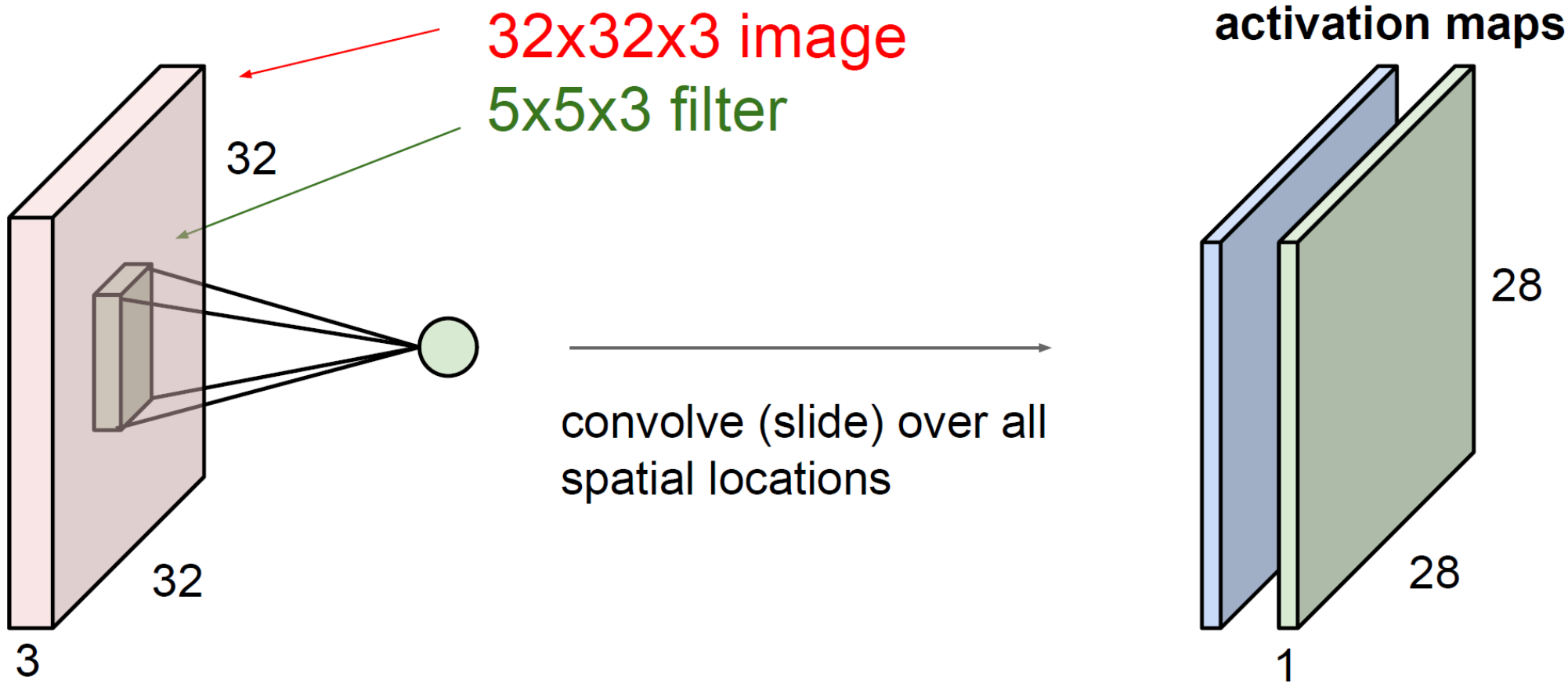
$$w^T x + b$$

Convolution Layer

: Preserve the spatial structure

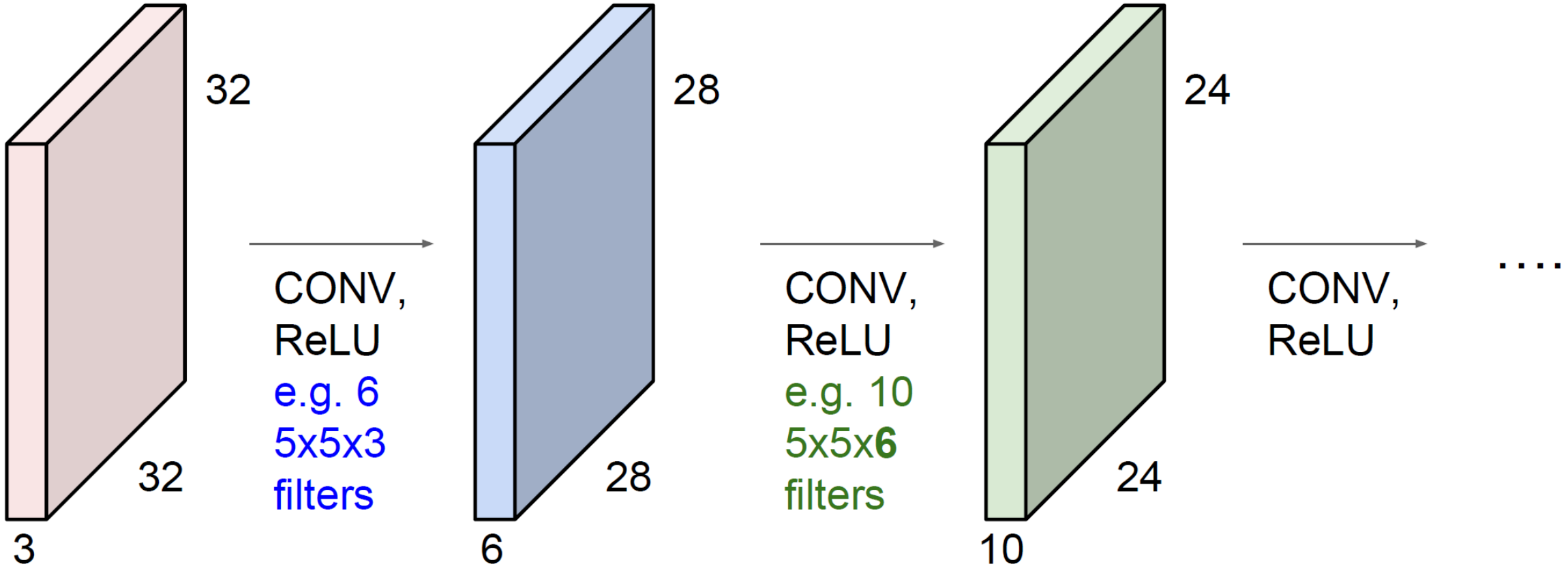
Convolution Layer

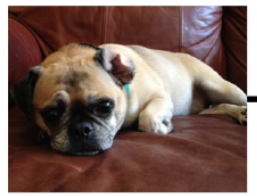
: Preserve the spatial structure



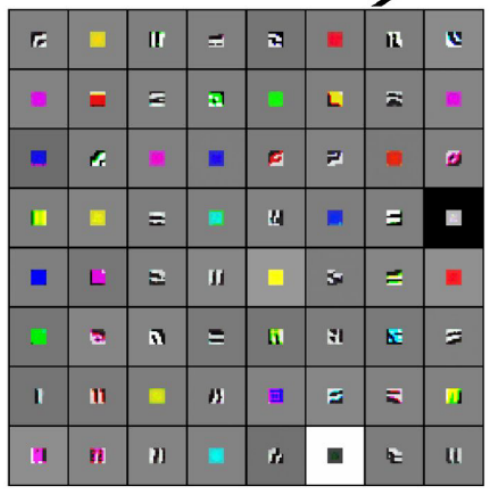
Convolutional Net

: Sequence of Convolutional Layers, interspersed with activation functions

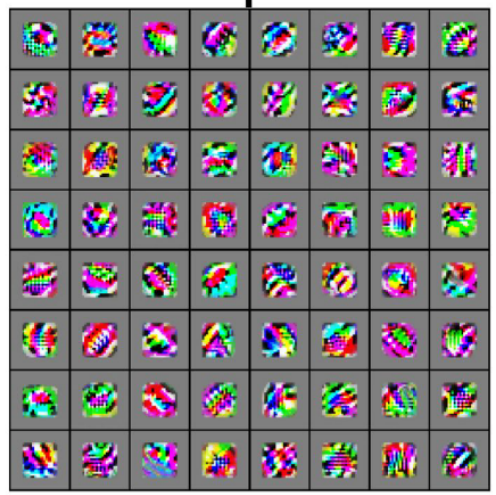




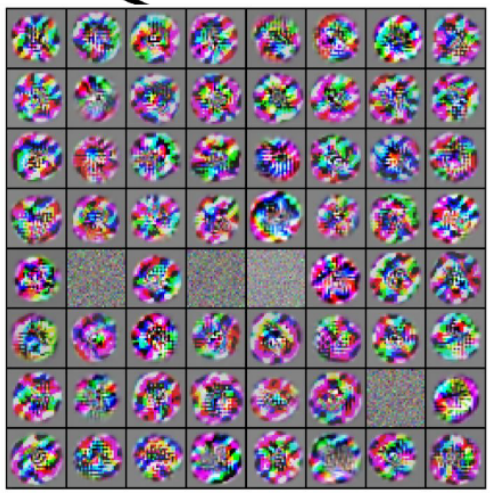
Low-level features → Mid-level features → High-level features → Linearly separable classifier



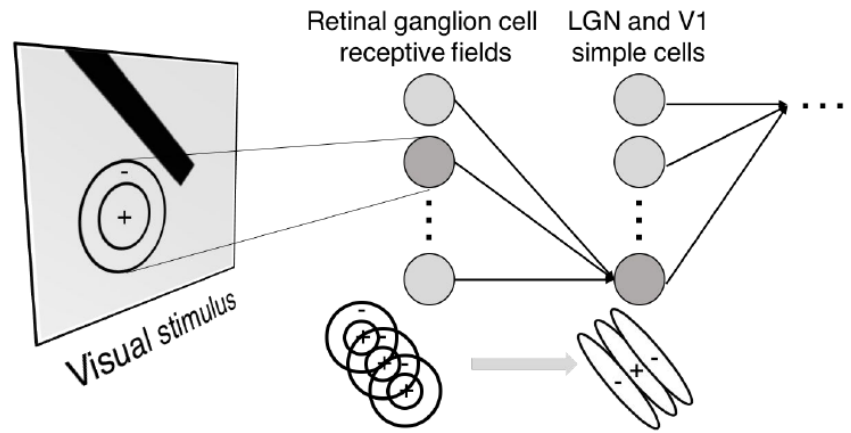
VGG-16 Conv1_1



VGG-16 Conv3_2



VGG-16 Conv5_3



Complex cells:
Response to light orientation and movement

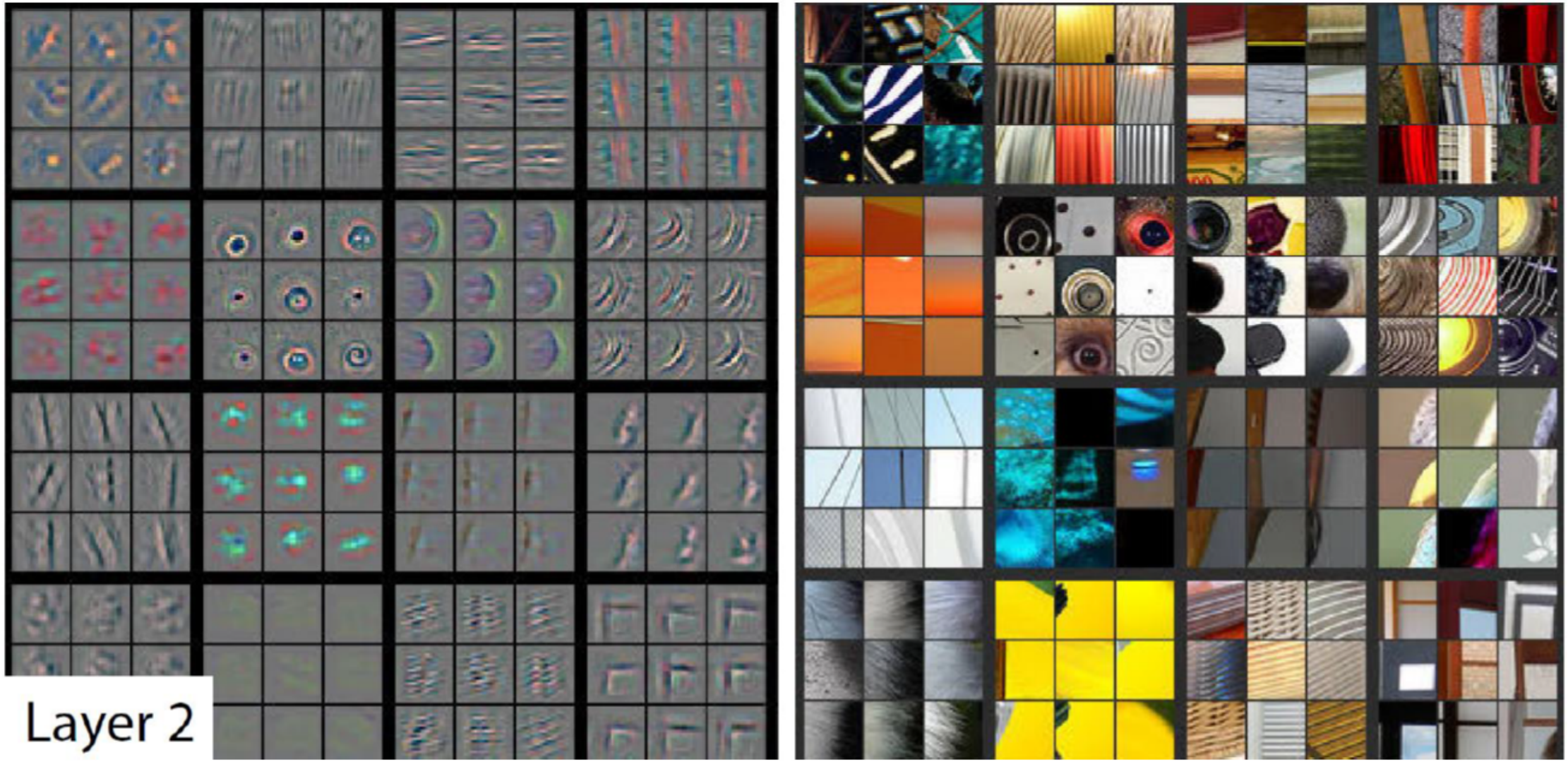
Hypercomplex cells:
response to movement with an end point

Stanford, cs231n 2017

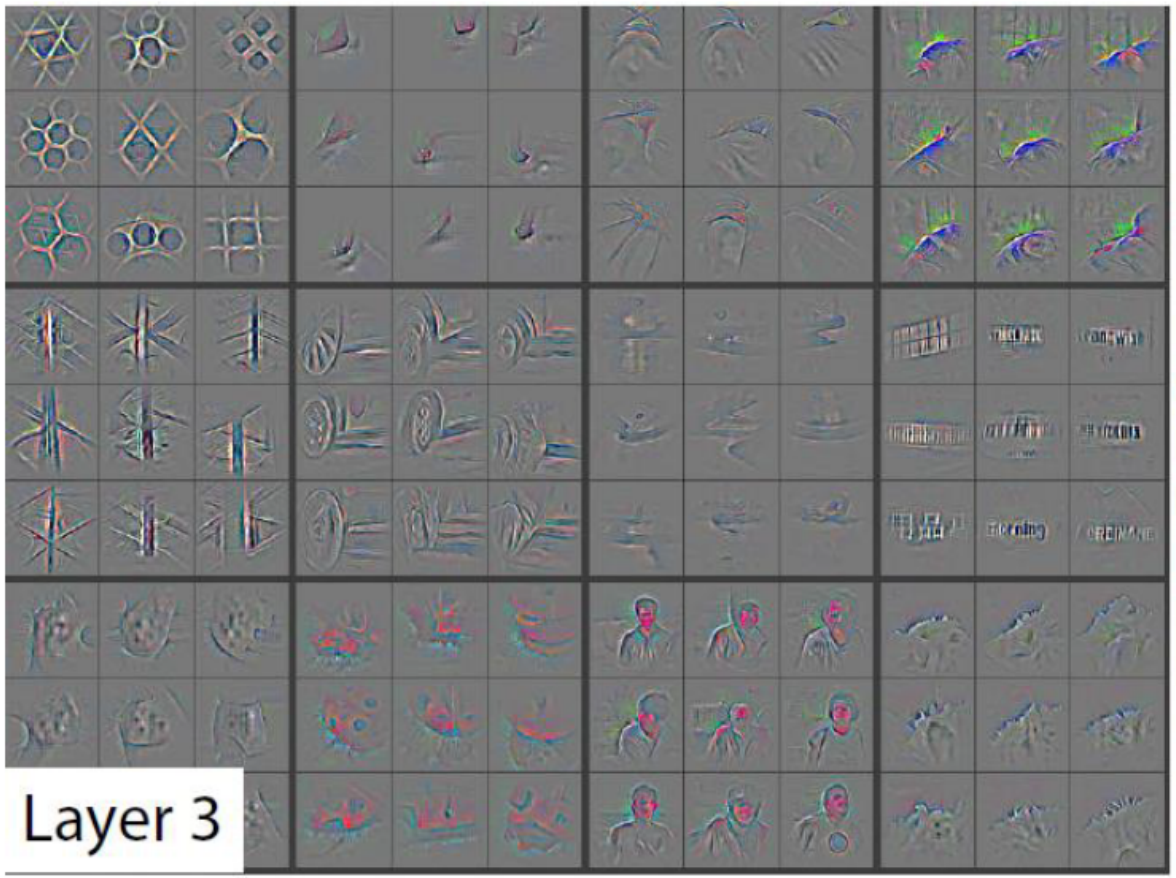
Layer 1

Layer 1

Layer 2

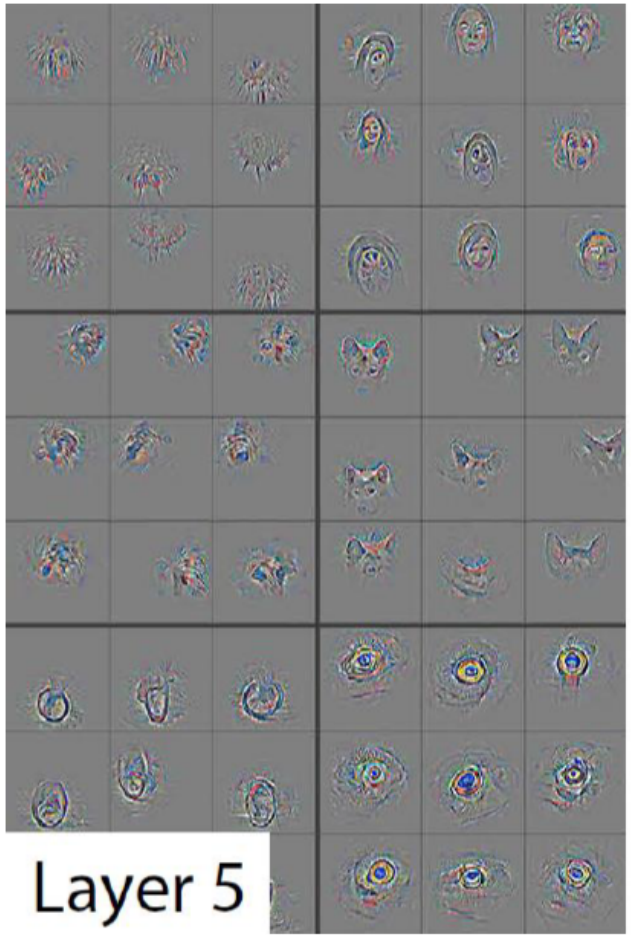


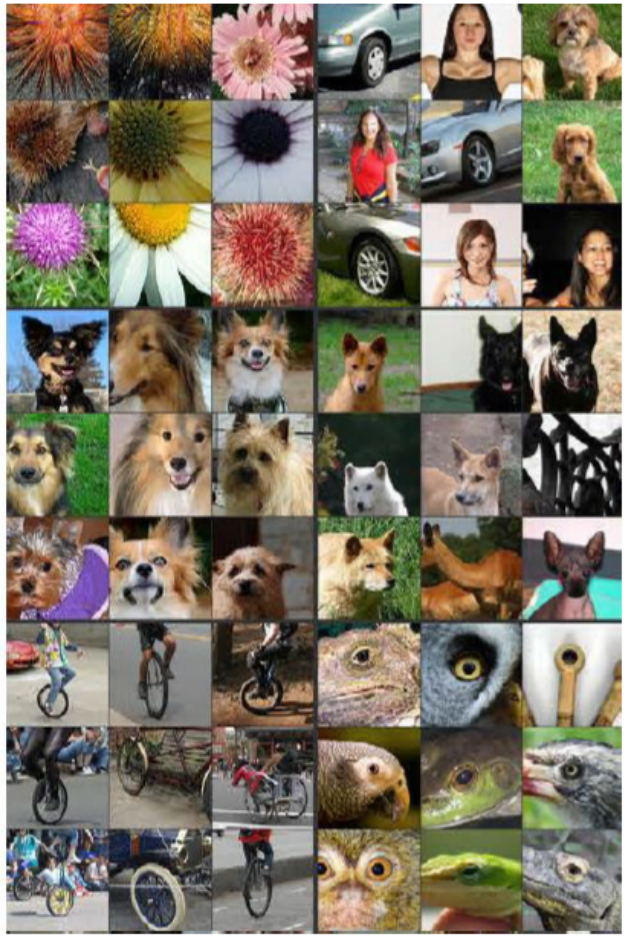
Layer 3

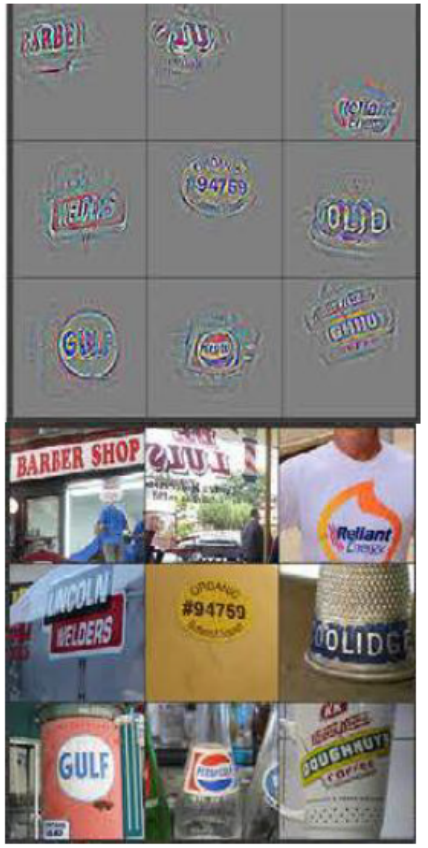


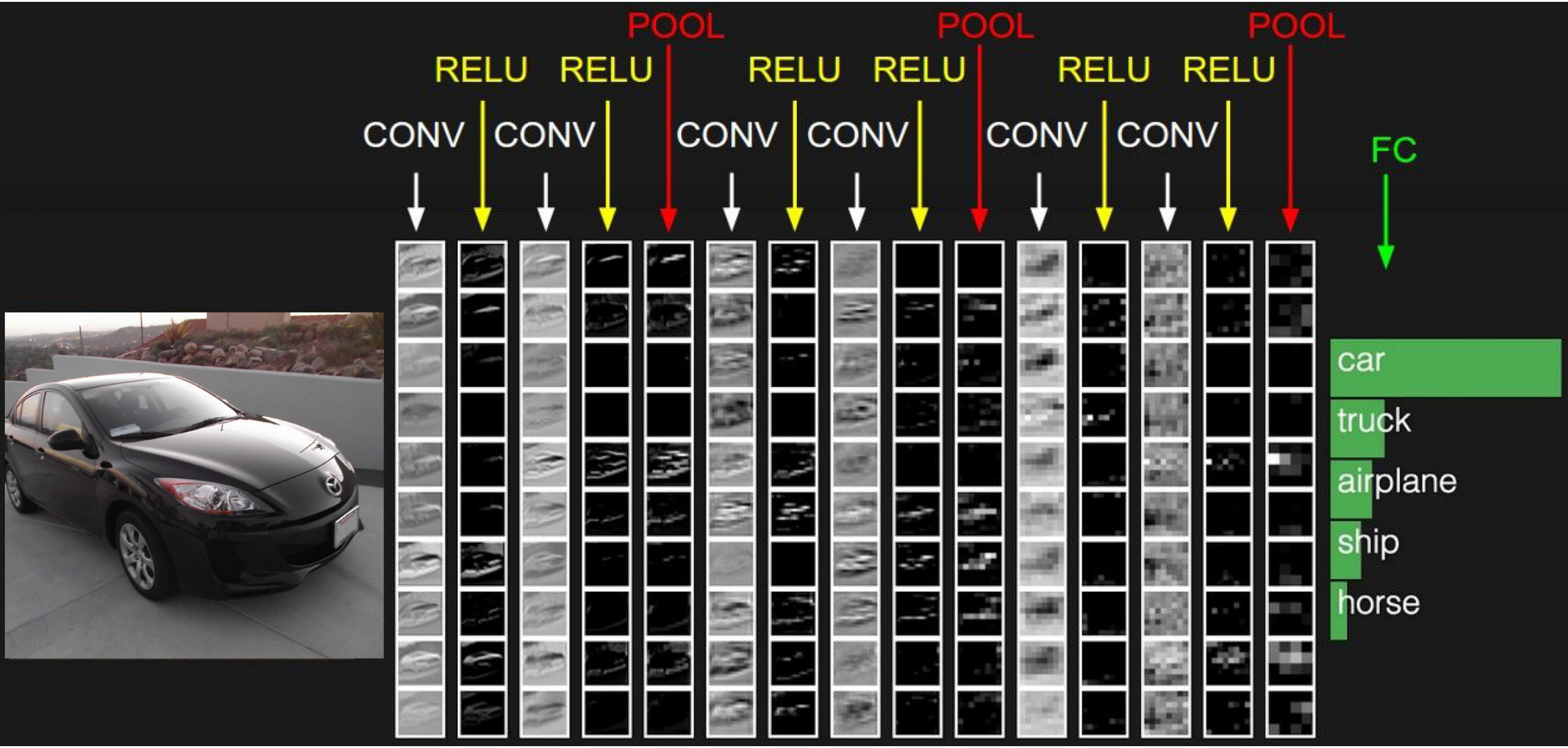
Neuron view of Convolutional Layer

Layer 5

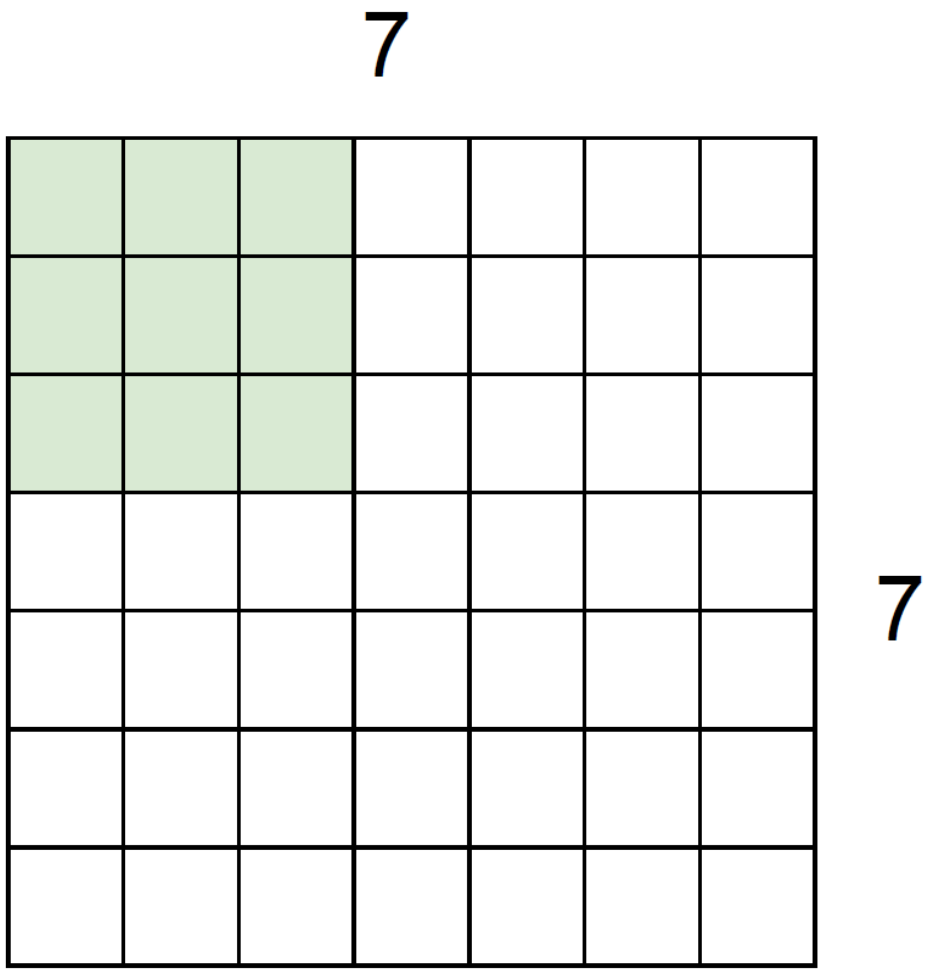






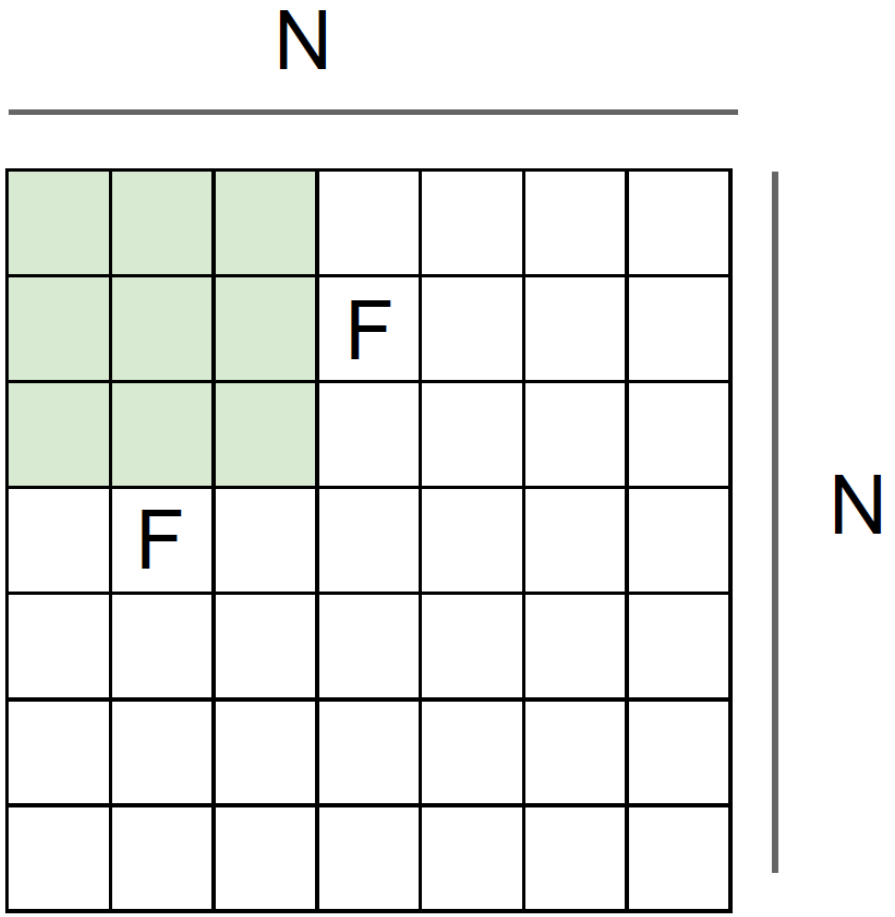

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Calculating spatial dimension



7x7 input (spatially)
assume 3x3 filter

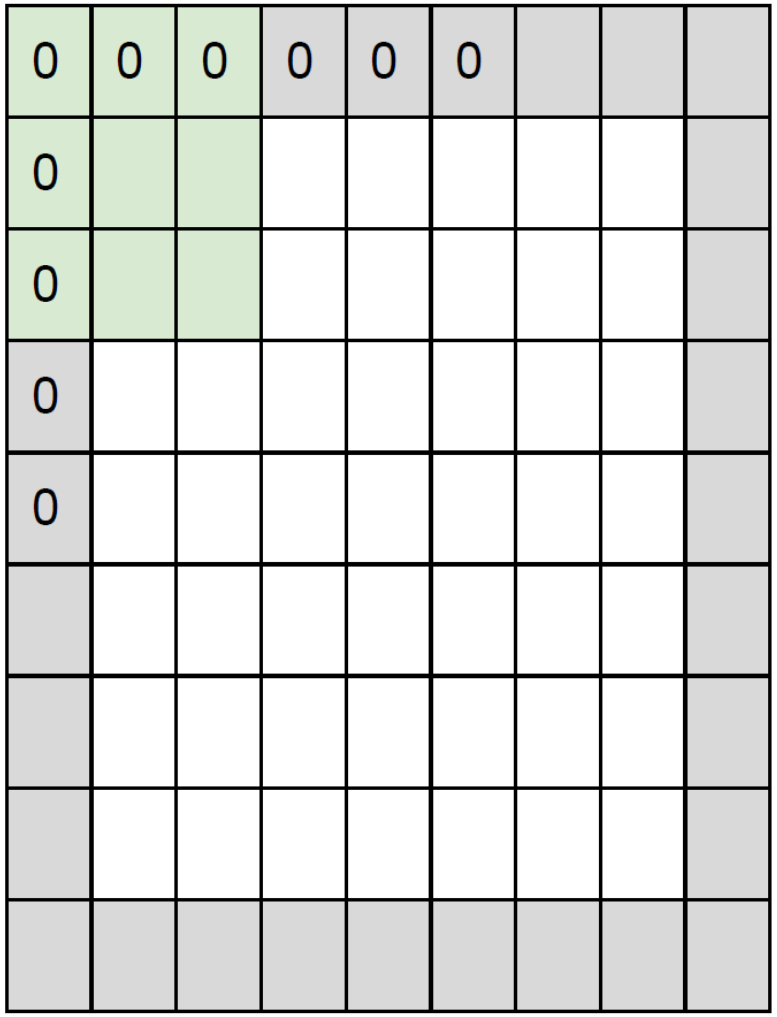
Calculating spatial dimension



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33

Zero padding



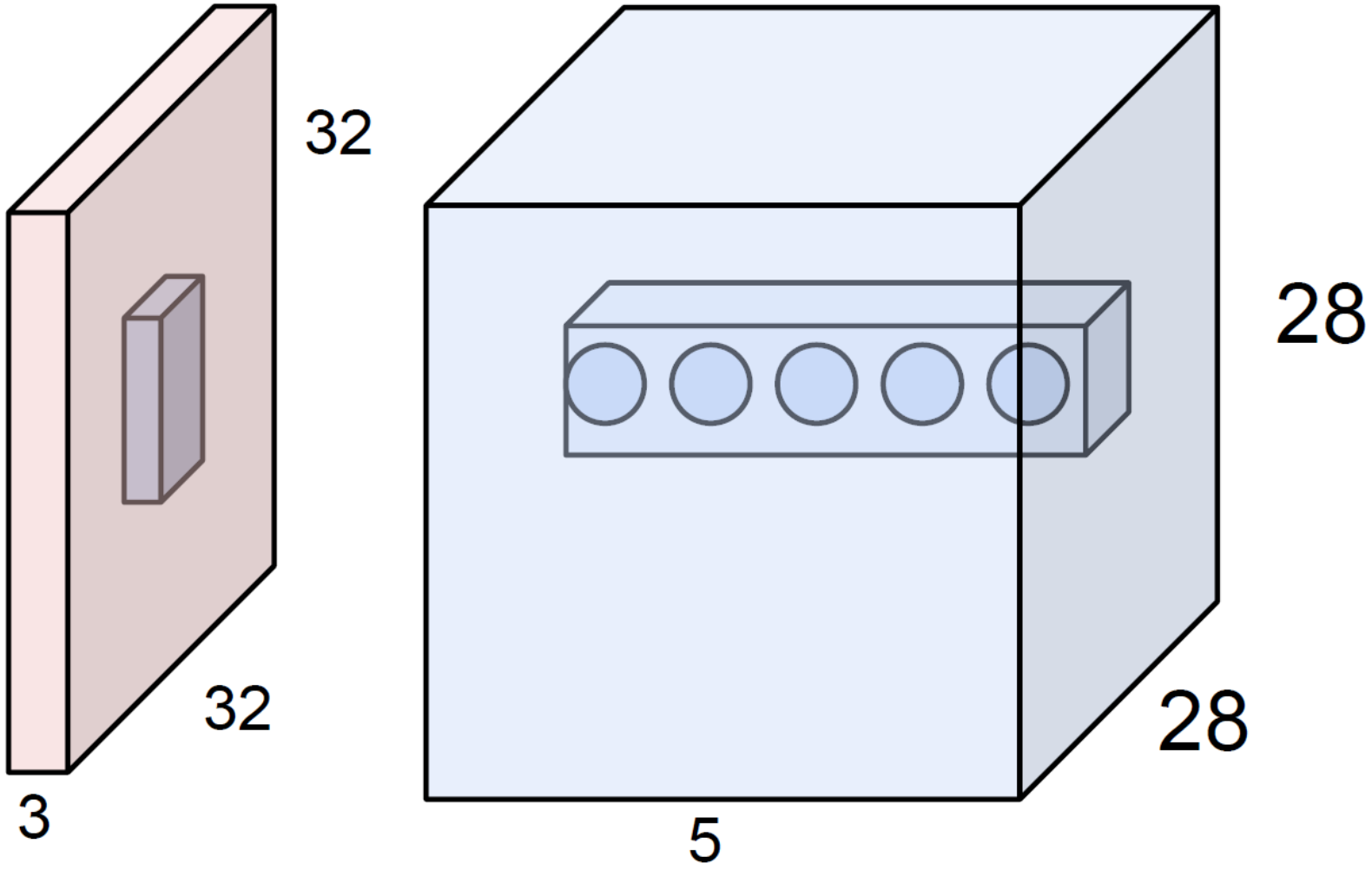
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

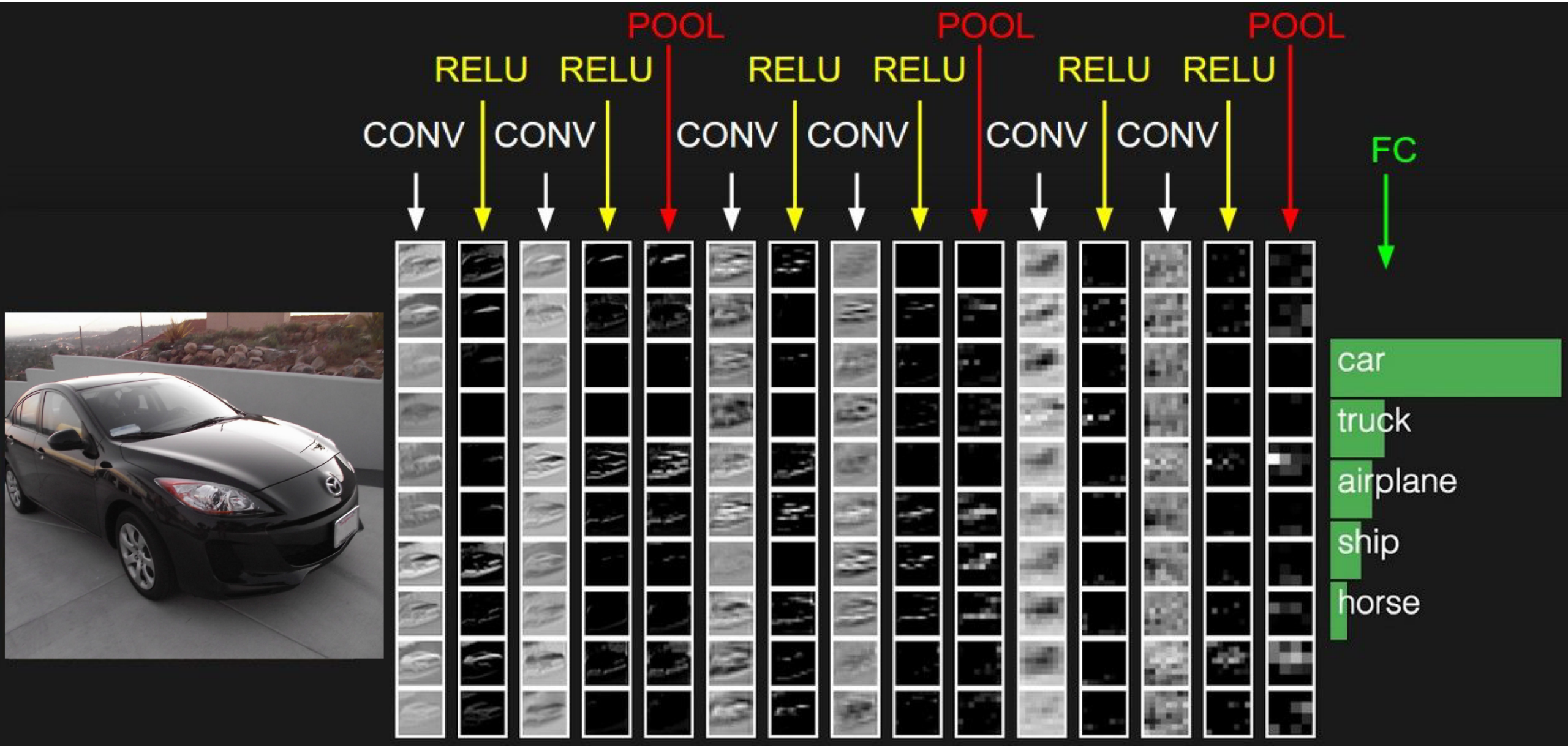
Dimension Formula

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$

Neuron view of Convolutional Layer

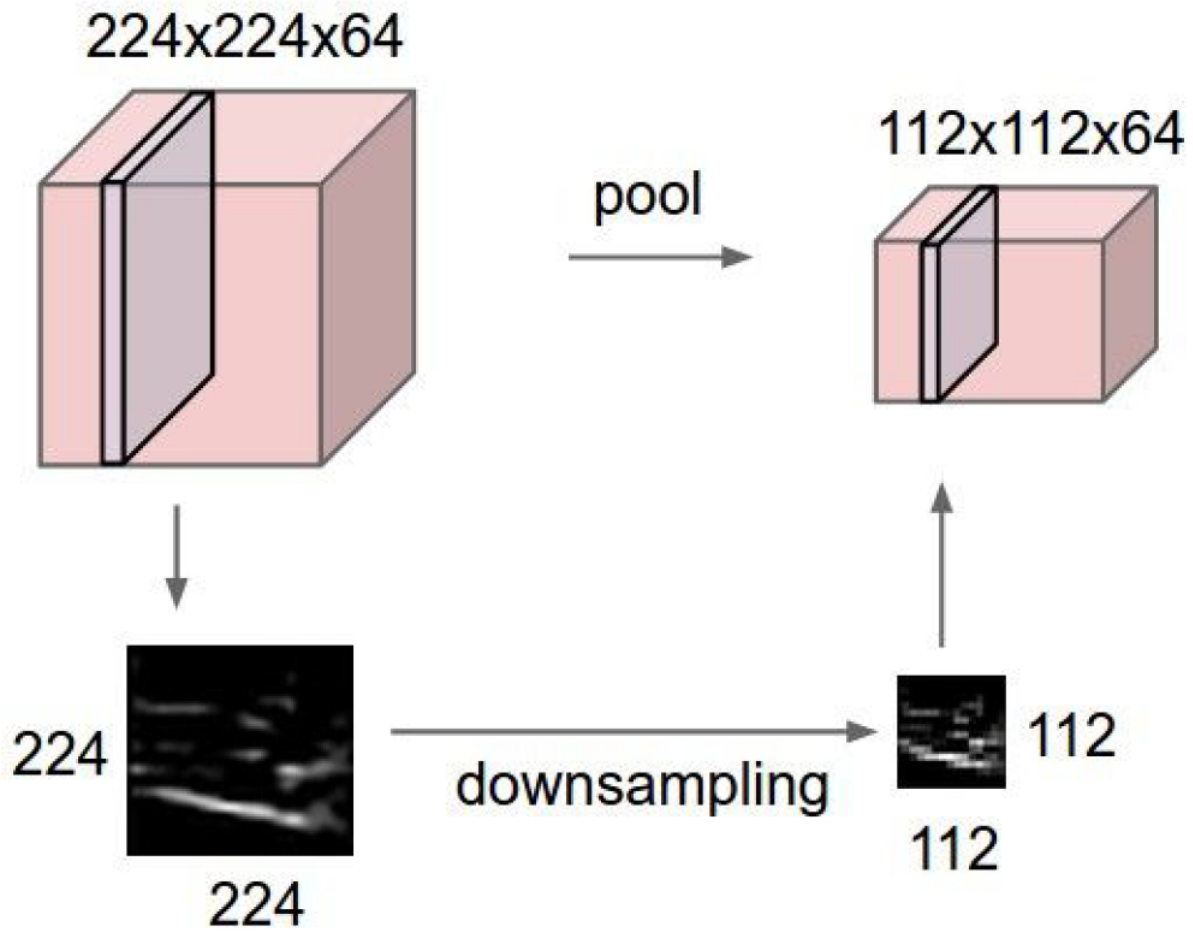


Pooling and FC Layer

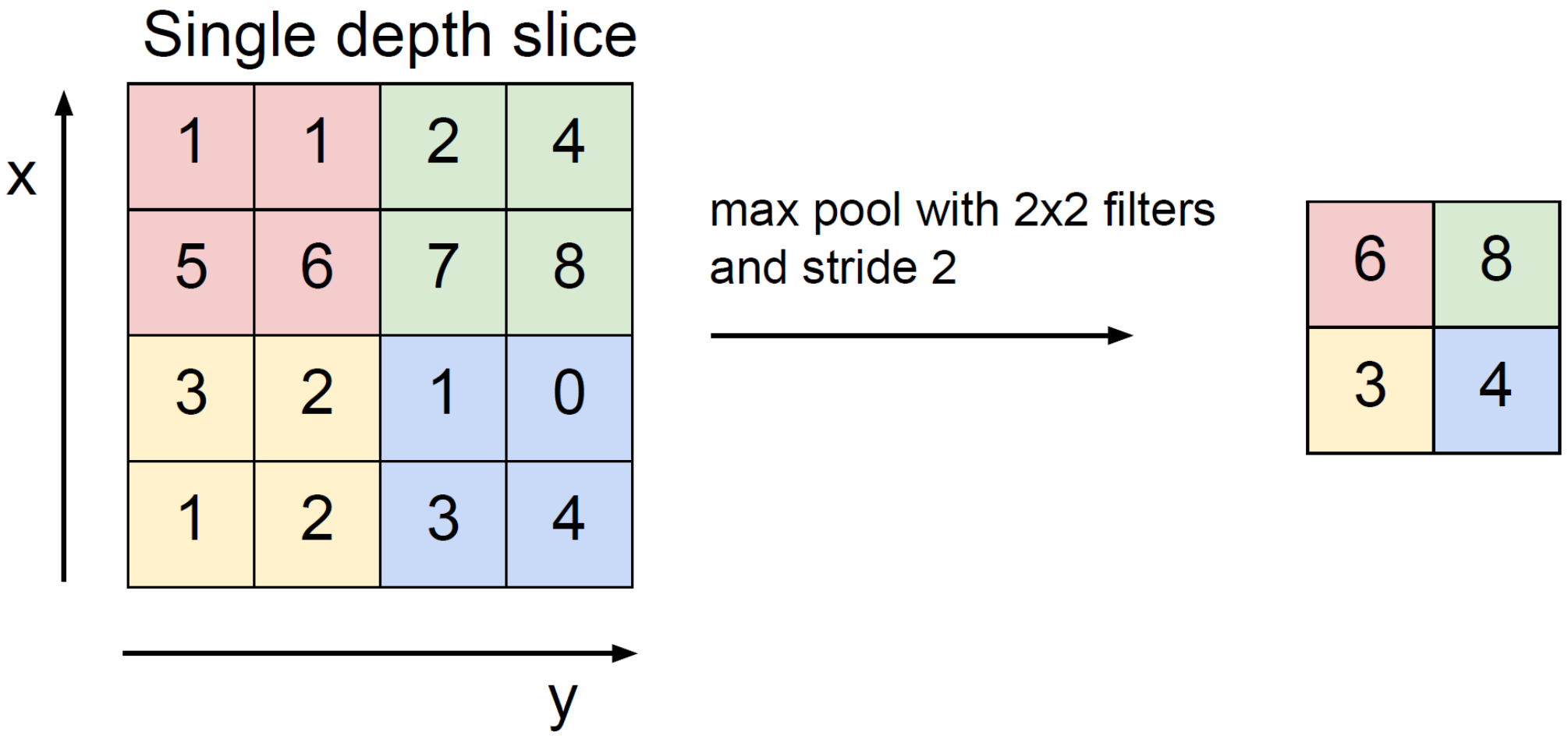

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Pooling layer

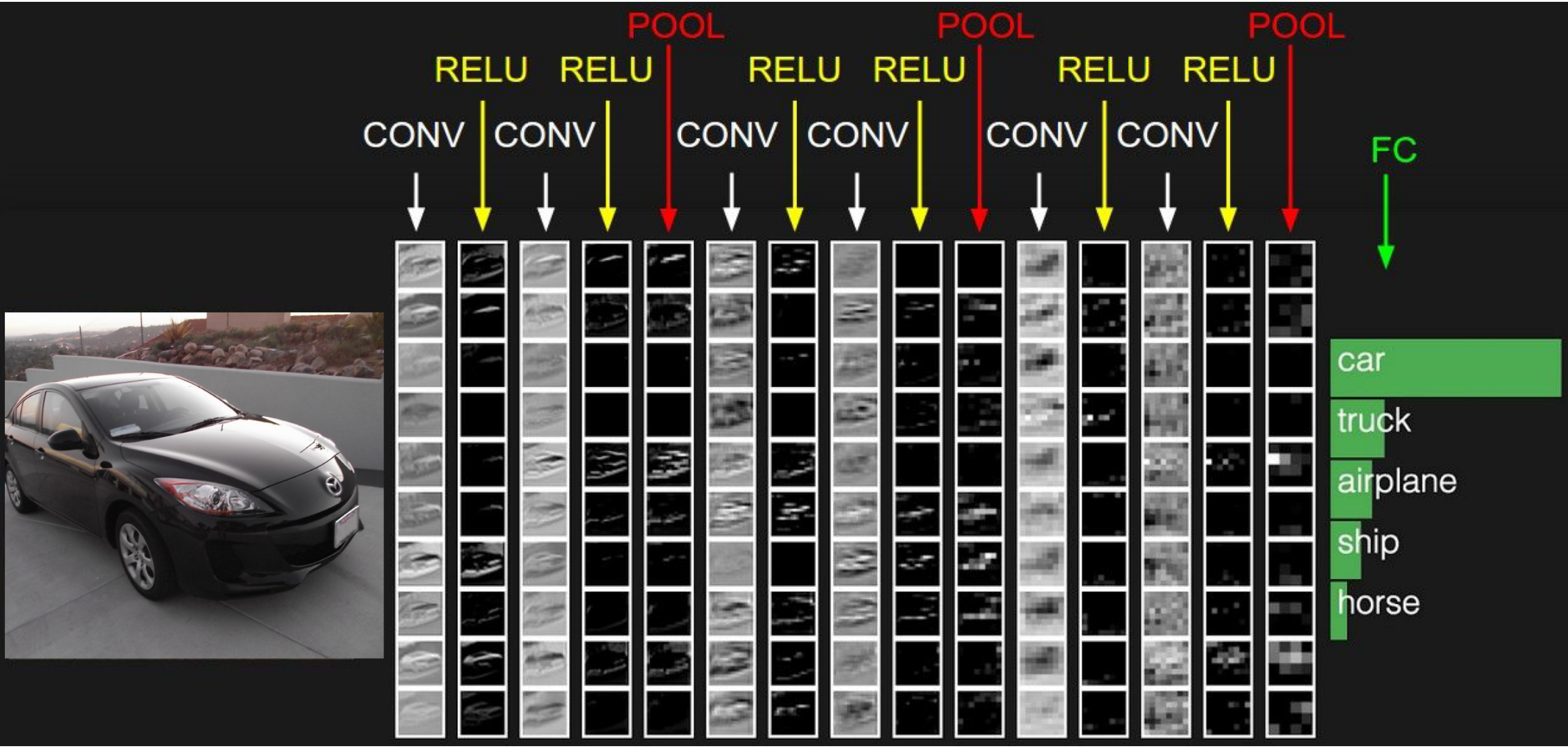
- makes the representations smaller and more manageable
- operates over each activation map independently:



MAX POOLING

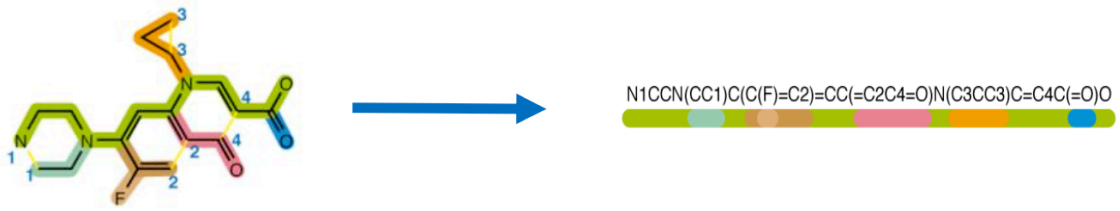

Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Fully Connected Layer


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

What is SMILES?

Data Representation – SMILES



*"The **simplified molecular-input line-entry system** (**SMILES**) is a specification in form of a line notation for describing the structure of chemical species using short ASCII strings. SMILES strings can be imported by most molecule editors for conversion back into two-dimensional drawings or three-dimensional models of the molecules."*

https://en.wikipedia.org/wiki/Simplified_molecular-input_line-entry_system

Data Representation – SMILES

	CMPD_CHEMBLID	exp	smiles
0	CHEMBL596271	3.54	Cn1c(CN2CCN(CC2)c3ccc(Cl)cc3)nc4ccccc14
1	CHEMBL1951080	-1.18	COc1cc(OC)c(cc1NC(=O)CSCC(=O)O)S(=O)(=O)N2C(C)...
2	CHEMBL1771	3.69	COC(=O)[C@@H](N1CCc2sccc2C1)c3ccccc3Cl
3	CHEMBL234951	3.37	OC[C@H](O)CN1C(=O)C(Cc2ccccc12)NC(=O)c3cc4cc(C...
4	CHEMBL565079	3.10	Cc1cccc(C[C@H](NC(=O)c2cc(nn2C)C(C)(C)C)C(=O)N...
...	...	...	...
4195	CHEMBL496929	3.85	OCCc1ccc(NC(=O)c2cc3cc(Cl)ccc3[nH]2)cc1
4196	CHEMBL199147	3.21	CCN(C1CCN(CCC(c2ccc(F)cc2)c3ccc(F)cc3)CC1)C(=O...
4197	CHEMBL15932	2.10	COc1cccc2[nH]ncc12
4198	CHEMBL558748	2.65	Clc1ccc2ncccc2c1C(=O)NCC3CCCCC3
4199	CHEMBL237889	2.70	CN1C(=O)C=C(CCc2ccc3ccccc3c2)N=C1N

Examples of SMILES in Lipophilicity Dataset

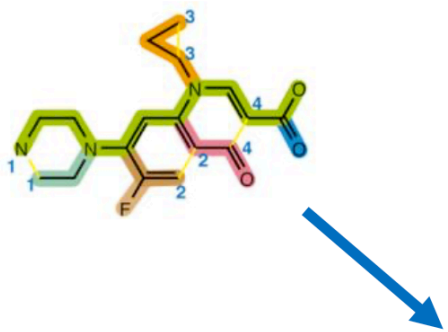
Contain upper/lower character with special character like (,),=,@,[,]

Arbitrary Length Character Sequence

How can we represent those sequences for CNN?

Data Representation – SMILES

How can we represent those sequences for CNN? Make a sequence as a 2D image!



Split SMILES string and convert it array of each character

N	1	C	C	N	(	C	C	1	)	C	(	C	(	F	)	=	C	2	)	=	C	C	(	=	C	2	C	4	=	O	)	N	(	C	3	C	C	3	)	C	=	C	4	C	(	=	O	)	O

Shape = [#batch, #characters]

Data Representation – SMILES

How can we represent those sequences for CNN? Make a sequence as a 2D image!

N	1	C	C	N	(	C	C	1	)	C	(	C	(	F	)	=	C	2	)	=	C	C	(	=	C	2	C	4	=	O	)	N	(	C	3	C	C	3	)	C	=	C	4	C	(	=	O	)	O

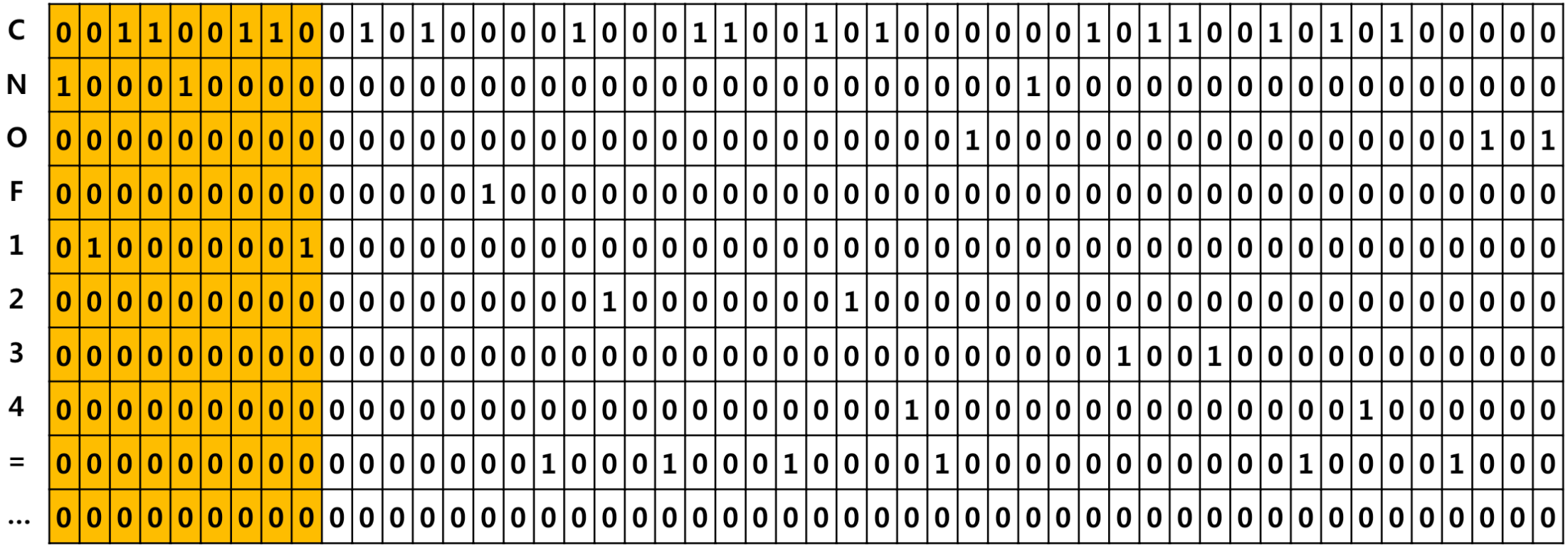
One-hot encoding

C	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0
N	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
F	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	
...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Shape = [#batch, 1, #characters, #vocabulary size]

Data Representation – SMILES

How can we represent those sequences for CNN? Make a sequence as a 2D image!



2d convolution, kernel size=9x(vocab_size), stride=1, filter=64

Shape = [#batch, 1, #characters, #filters]

Data Representation – SMILES

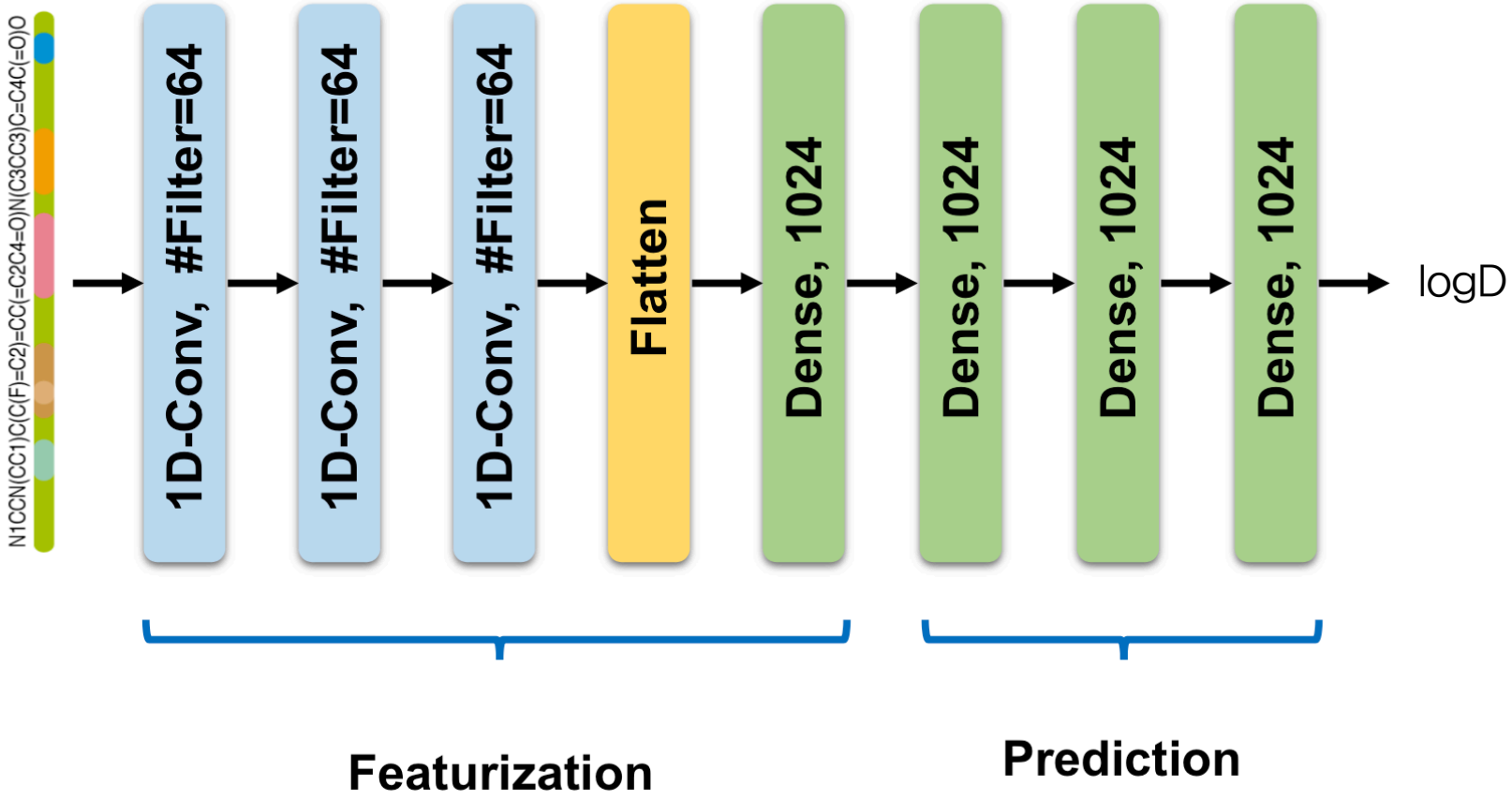
How can we represent those sequences for CNN?   Make a sequence as a 2D image!

C	0	0	1	1	0	0	1	1	0	0	1	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	0
N	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2d convolution, kernel size=9x(vocab_size), stride=1, filter=64

Shape = [#batch, 1, #characters, #filters]

Overall Architecture of the CNN model



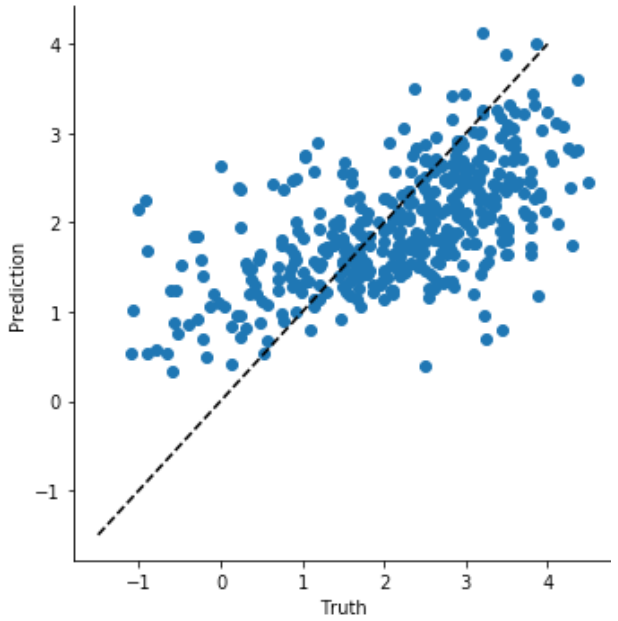
Implementing Vanilla GCN

1. Dataset and DataLoader
→ Return matrix X with one-hot-encoding and lipophilicity y

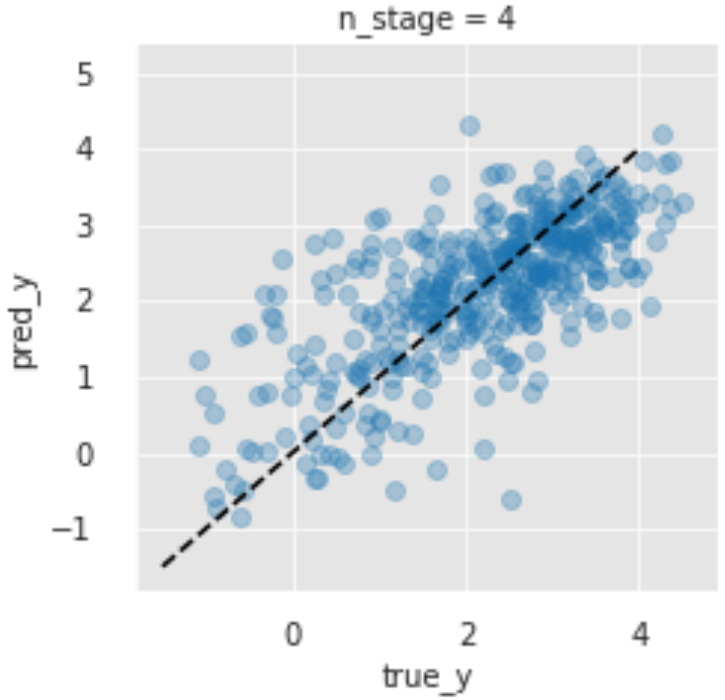
2. CNN architecture
→ convolution layer
→ 1D Batch Normalization for GCN
→ Flatten Layer

3. Hyperparameter Tuning
→ Tweak hyperparameter to maximize the performance

Results



2 hidden Layer MLP
(0.5 dropout rate)
MAE : 0.774



4 Layer CNN
(0.3 dropout rate)
MAE : 0.643

CNN shows improved results!

Summary

- CNN is excellent for handling 1D, 2D or 3D spatial data
- SMILES is linear string representation use widely
- SMILES string can be converted to the 2D image with one-hot-encoding
- Watch out for the dimension change through multiple CNN layers
- CNN model was better than MLP model with fingerprint