CIS 6930 Topics in Computing for Data Science
Week 3: Convolutional Neural Networks

9/16/2021

Yoshihiko (Yoshi) Suhara

2pm-3:20pm & 3:30pm-4:50pm

Week 3!

- ~~Week 1: Deep Learning Basics (Thu 9/9)~~
- ~~Week 2: AutoEncoder (Tue 9/14)~~
- **Week 3: Convolutional Neural Networks (Thu 9/16)**
- Week 4: GAN (Tue 9/21)
- Week 5: Word embeddings: Word2vec, GloVe (Thu 9/23)
- Week 6: Recurrent Neural Networks (Tue 9/28, Thu 9/30)
- Week 7: Review/Project pitch & Mid-term (Tue 10/5, Thu 10/7)
- Fall Break
- ...

Why You Should Remember So Many Techniques As Data Scientists/Software Engineers/Researchers

- To have more options (more is better, with a note)
- To tell the stakeholder why your choice should work better than other alternative solutions
  - Professionals should consider possible solutions to discuss risk & return

- Why do we need CNNs? Why are CNNs a better option than MLP?

Hint: Understand What and Why (When)?

- What is the technique?
- Why do I need this technique and when should I use it?
  - What are alternatives? Why do these alternatives not work for the problem?

- Why do we need CNNs? Why are CNNs a better option than MLP?

Checklist

- Solution / Technique:
  - Key Idea
- Alternatives / Existing Solutions:
  - Limitations

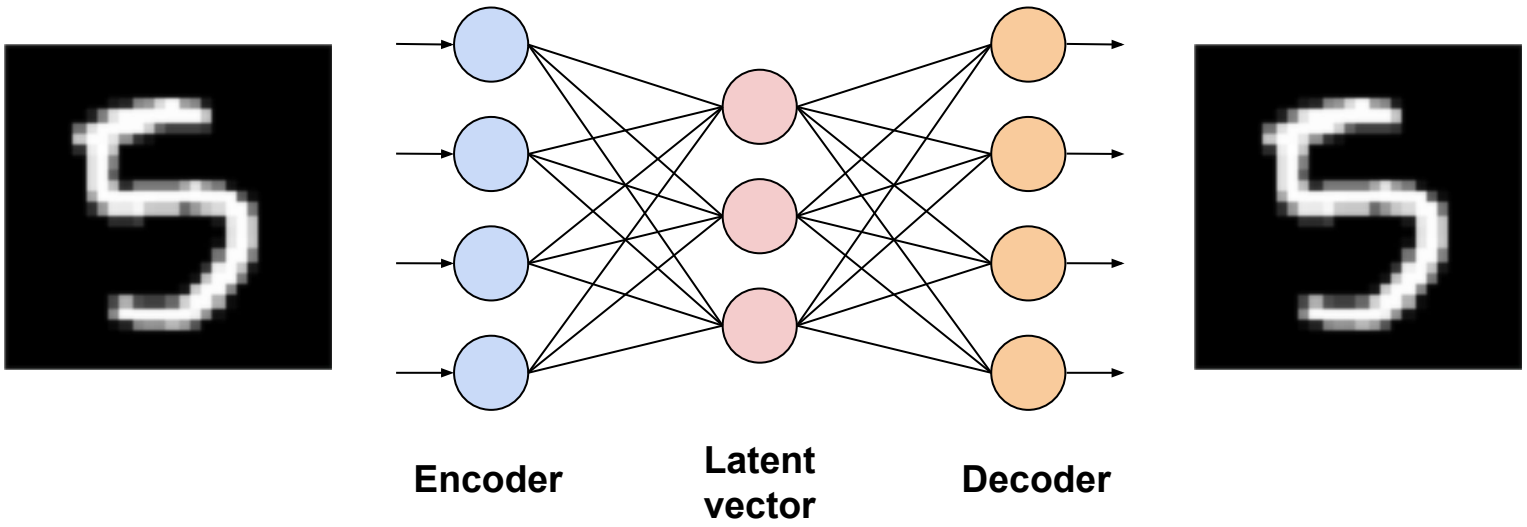
- It is preferable that you can logically explain those points in plain language
  - Be accountable! Don't just say this achieves higher accuracy!

- + If you obtain empirical results, that can be a research paper!
  - Of course, you need to justify the novelty of the technique, though

Recap

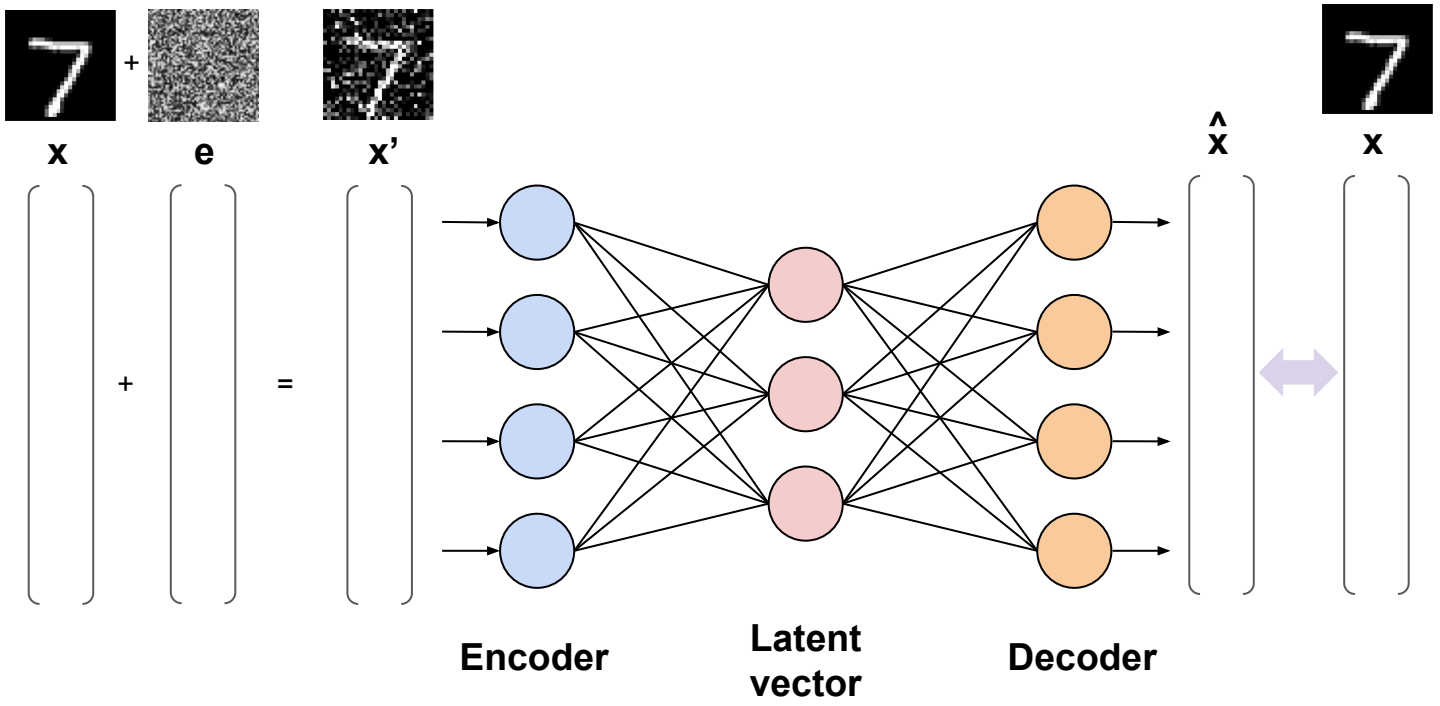
Recap: Autoencoders

- **Encoder-decoder models** that **learn to reconstruct the original data**



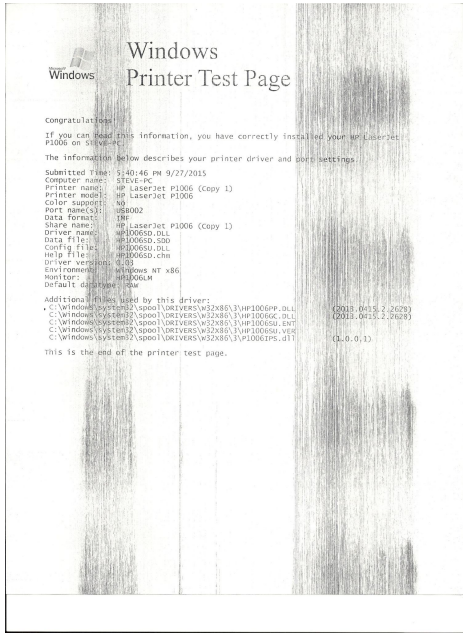
Recap: Denoising Autoencoder

- Learn to reconstruct the original data from **corrupted input**



Recap: Autoencoder Applications

- Image restoration problems in general



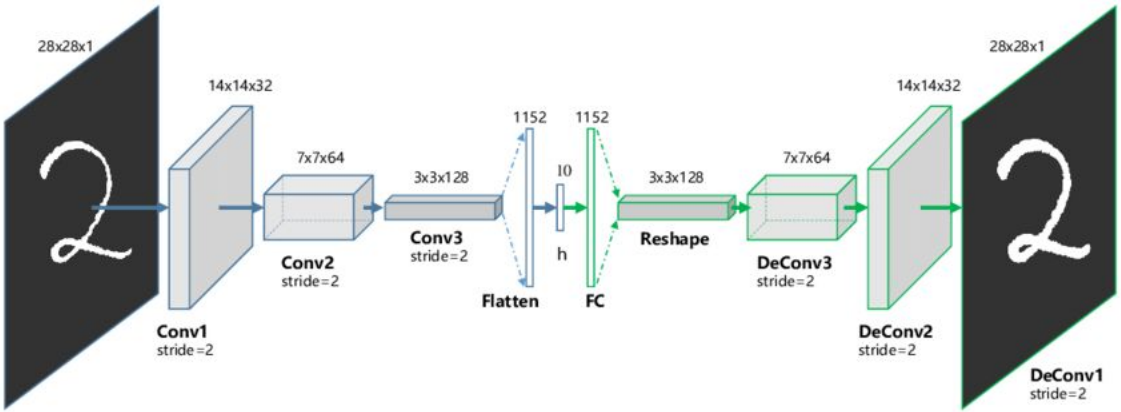
How do you train Autoencoder models for the problems?

Another Application: Image Super-Resolution

- 

[New!] Convolutional Autoencoder

- We will discuss it later. Stay tuned!

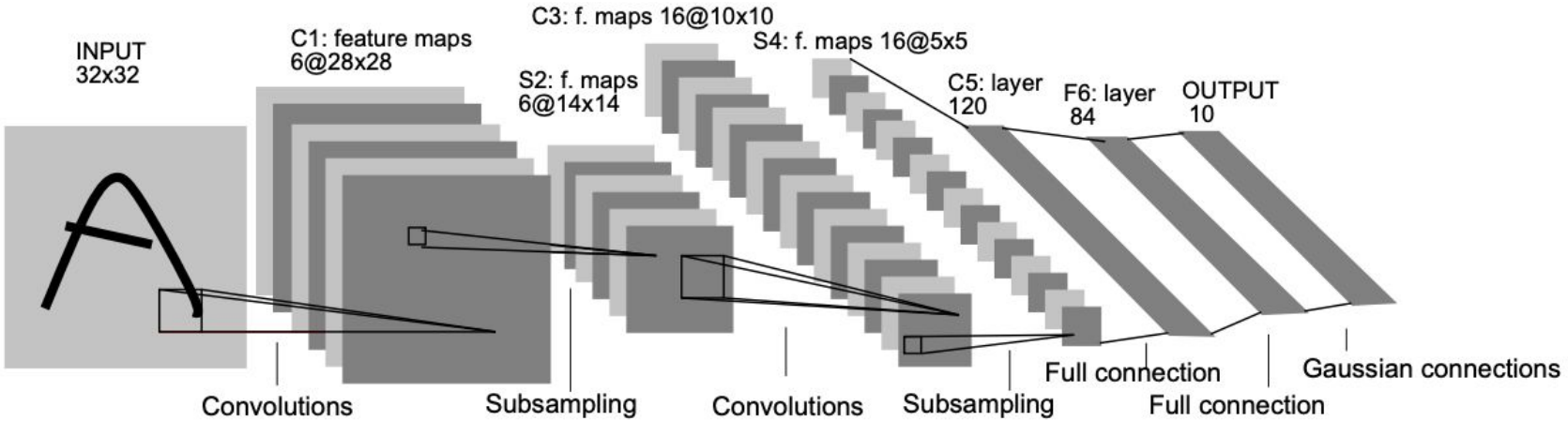


Today’s Goal (1)

- Learn Key Concepts of Convolutional Neural Networks
  - Convolution filters
  - Max pooling, Mean pooling
  - Fully-connected layers

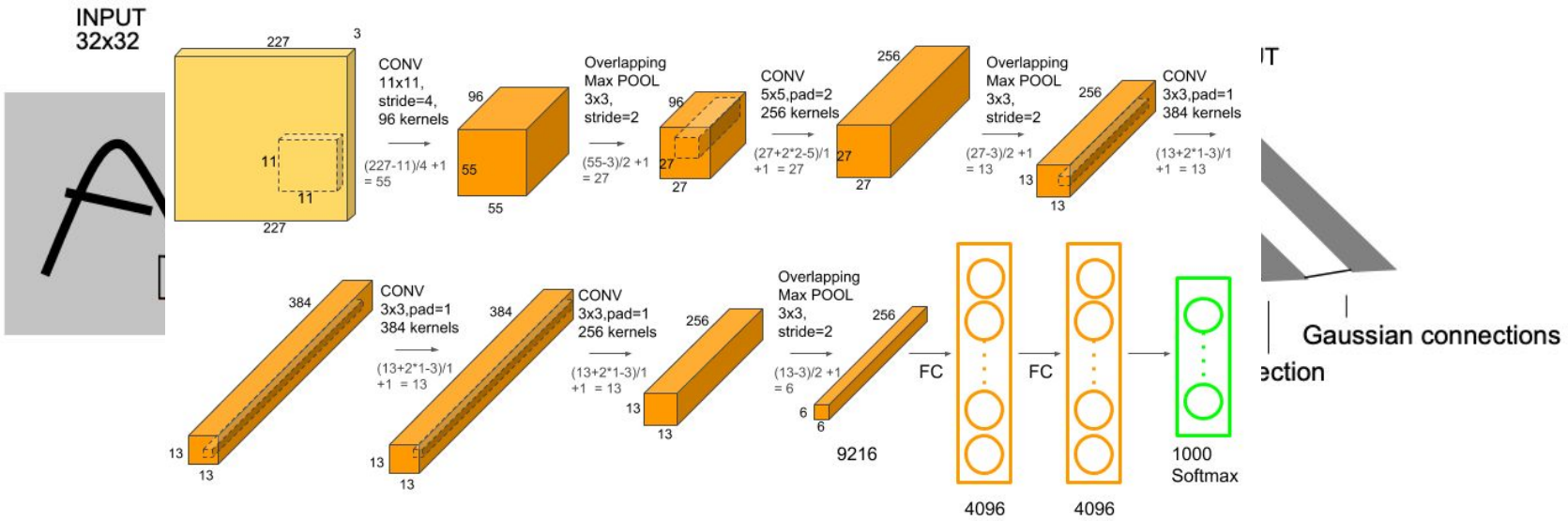
Today's Goal (2)

- You will be able to implement this (↓) using PyTorch on your own



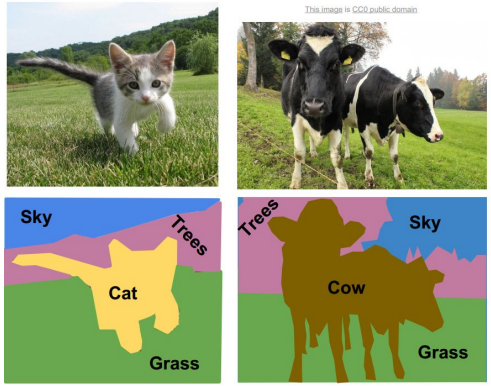
Today's Goal (2)

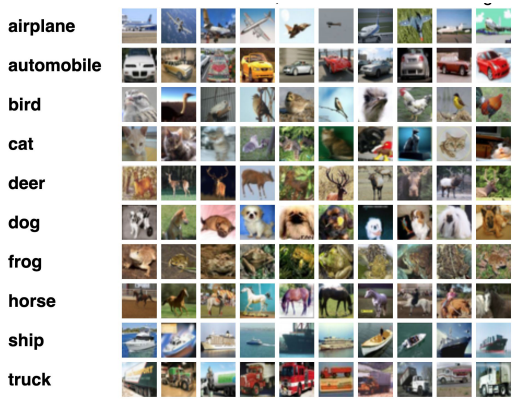
- You will be able to implement this (↓) using PyTorch on your own. Even this

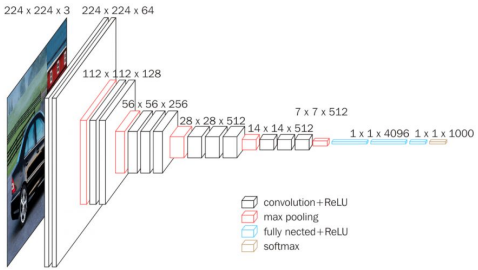


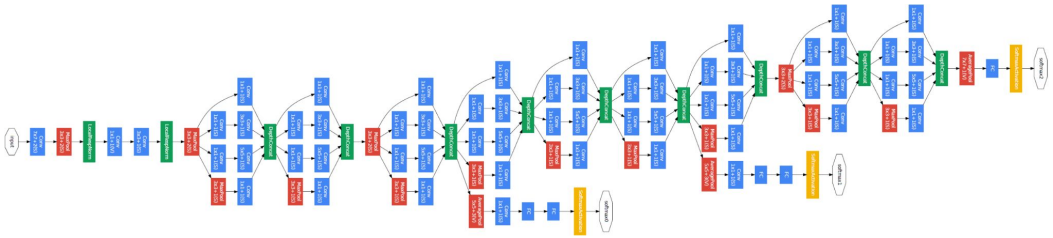
And...

- We will also discuss **advanced CNN models** and **applications**





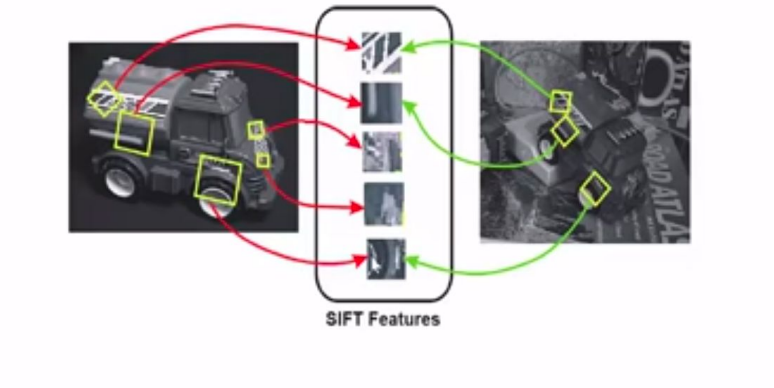


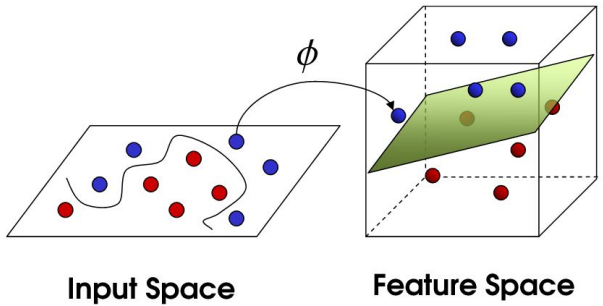


Why Convolutional Neural Networks?

Pre-Deep Learning Era

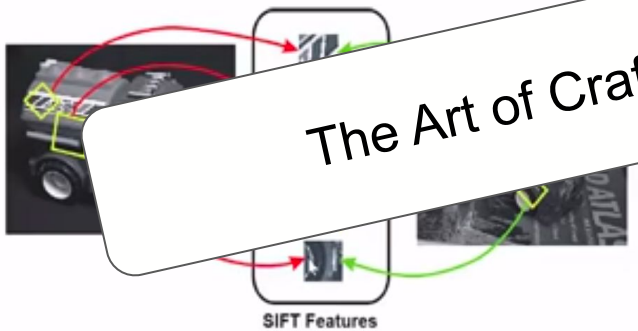
- Feature extraction + Nonlinear model (e.g., SVM + Kernel trick)
  - Scale-invariant feature transform (SIFT)
  - Bag-of-Features





Pre-Deep Learning Era

- Feature extraction + Nonlinear model (e.g., SVM + Kernel trick)
  - Scale-invariant feature transform (SIFT)
  - Bag-of-Features

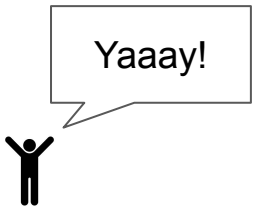


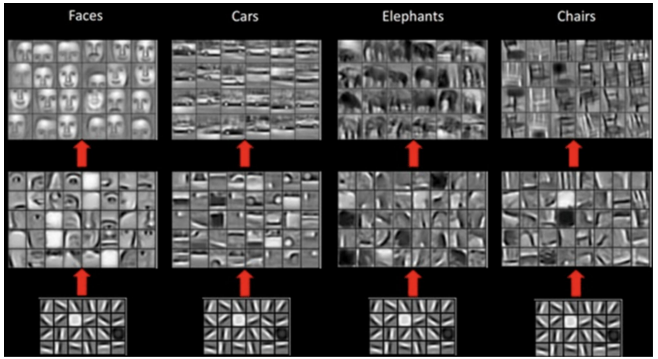
The Art of Craftsmanship!

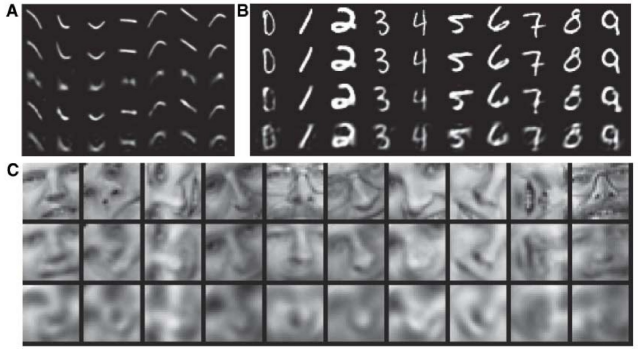
Return of Neural Networks: Deep Learning (2010s-)

- Maturity of Neural Networks techniques → Representation Learning
  - **Auto-Encoders (2006)**
  - Advances in Convolutional Neural Networks (originally 1998)

- The age of Big Data
  - Computational resource ↑
  - Data size ↑

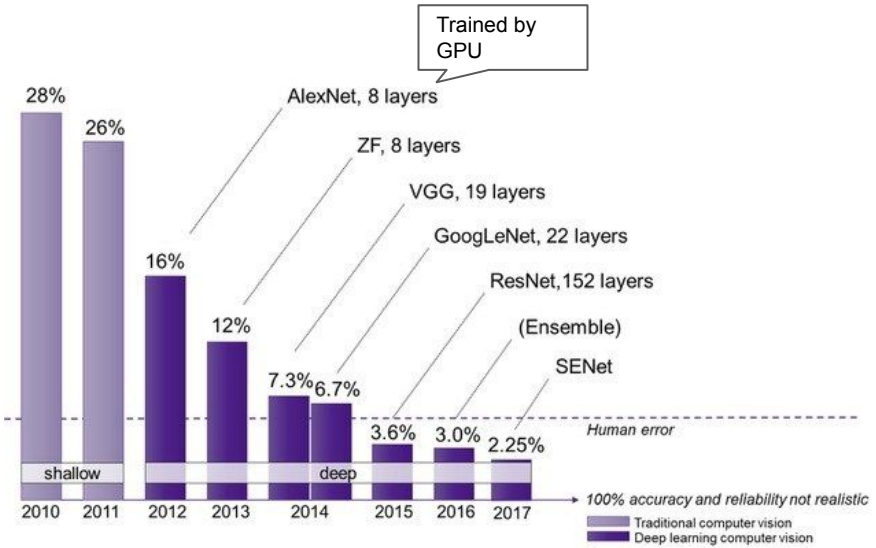






ImageNet Large Scale Visual Recognition Challenge (LSVRC)

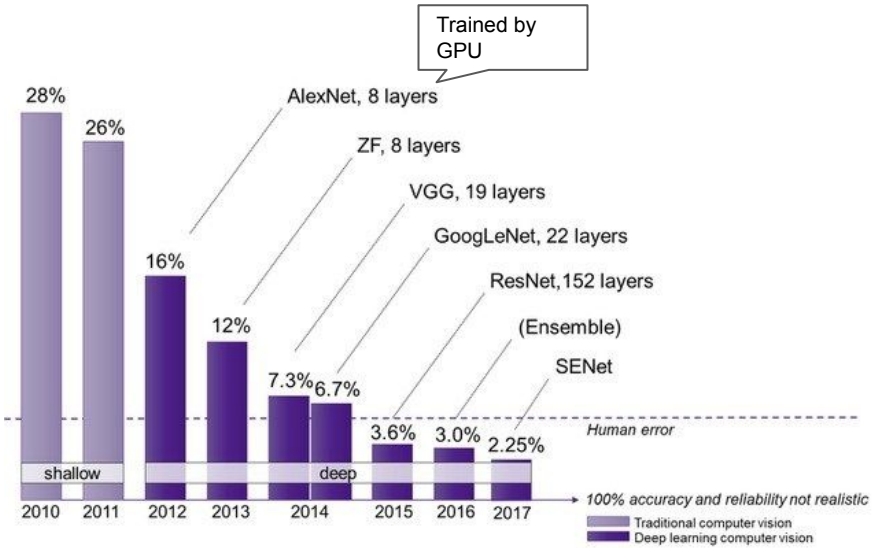
Since 2012, Deep Learning models have **dominated the competition** (and models have been getting deeper and deeper)



https://semiengineering.com/new-vision-technologies-for-real-world-applications/

ImageNet Large Scale Visual Recognition Challenge (LSVRC)

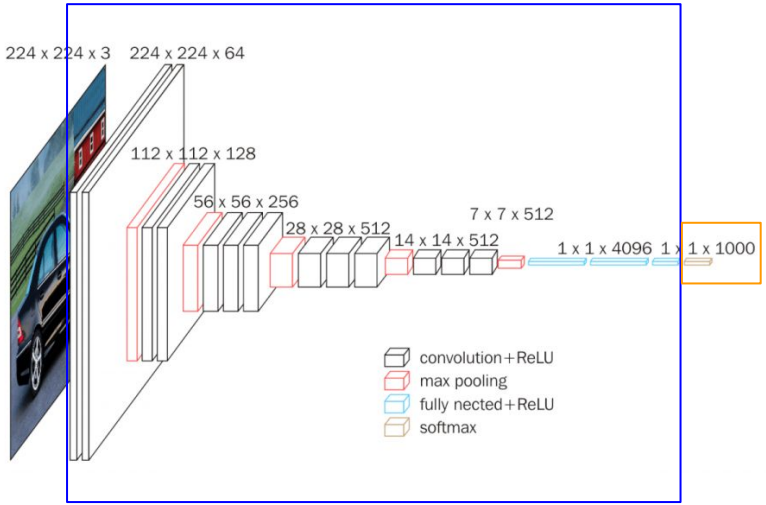
Since 2012, Deep Learning models have **dominated the competition** (and models have been getting deeper and deeper)



https://semiengineering.com/new-vision-technologies-for-real-world-applications/

What Makes Convolutional Neural Networks So Powerful?

- The representation learning capability =~ “Feature” extraction ability
  - boosted by large-scale data + computational resources



VGG 16

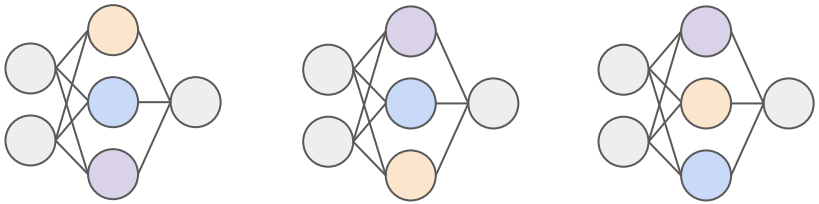
CNN? ConvNet?

- Convolutional Neural Networks are often abbreviated as **CNN** or **ConvNet**
- In this lecture, I mostly use CNN

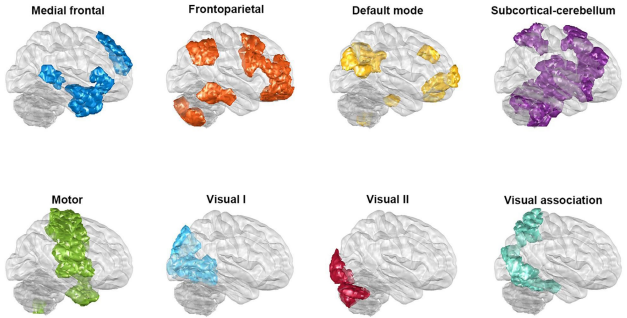
CNN Principles

Redundancy had been the Main Obstacle

- Multi-layer NN models are (too) versatile
- Neural Networks are (intrinsically) redundant
  - This made NN training difficult
  - → not really anymore, thanks to the advances in optimization techniques :)

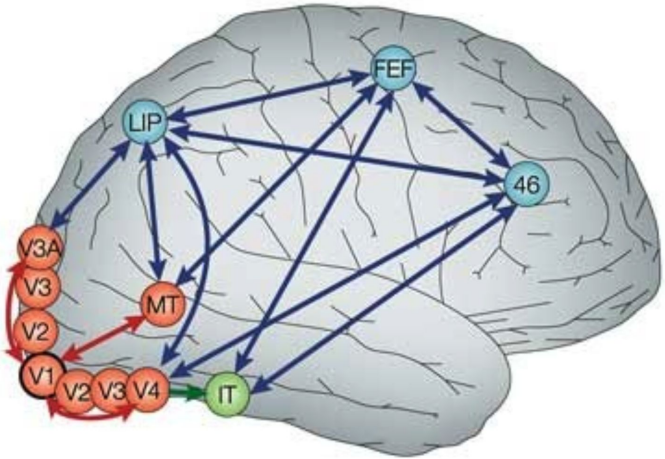


e.g., Parameter interchangeability

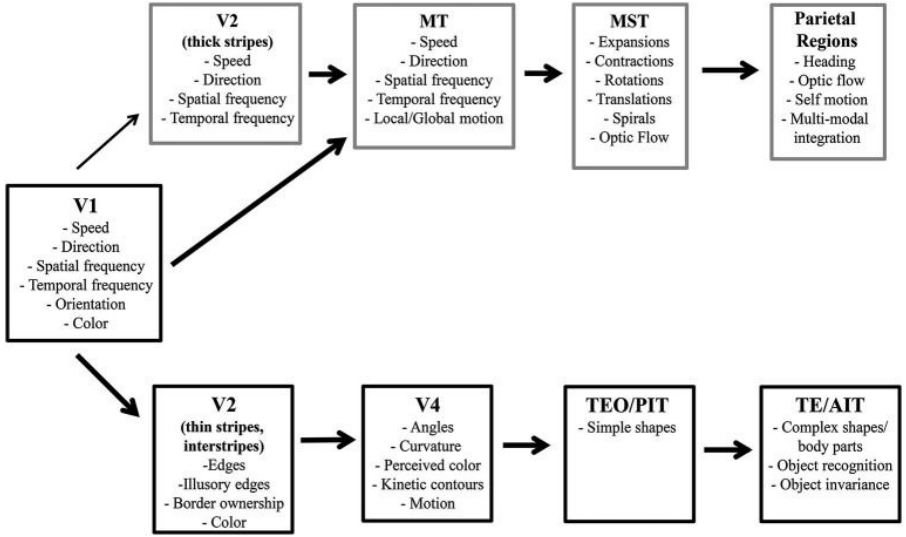


How Vision Works

- Visual elements are processed in a multi-stage manner



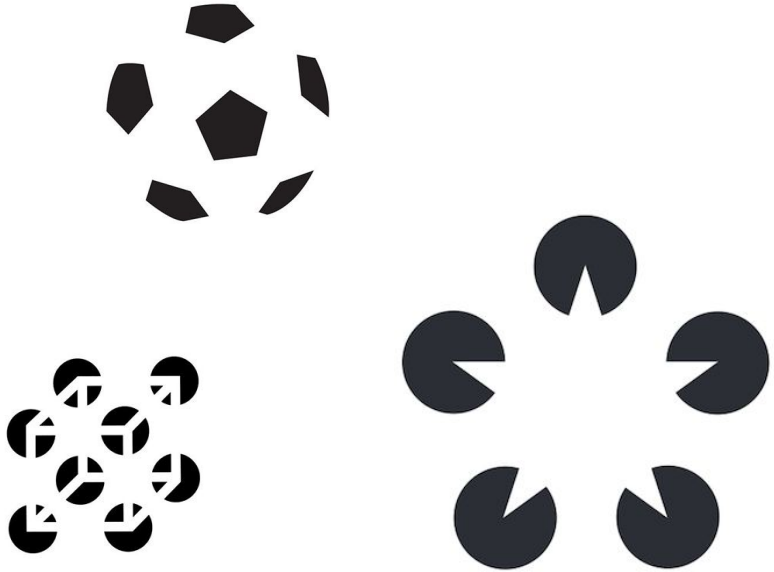
https://www.nature.com/articles/nrn1055



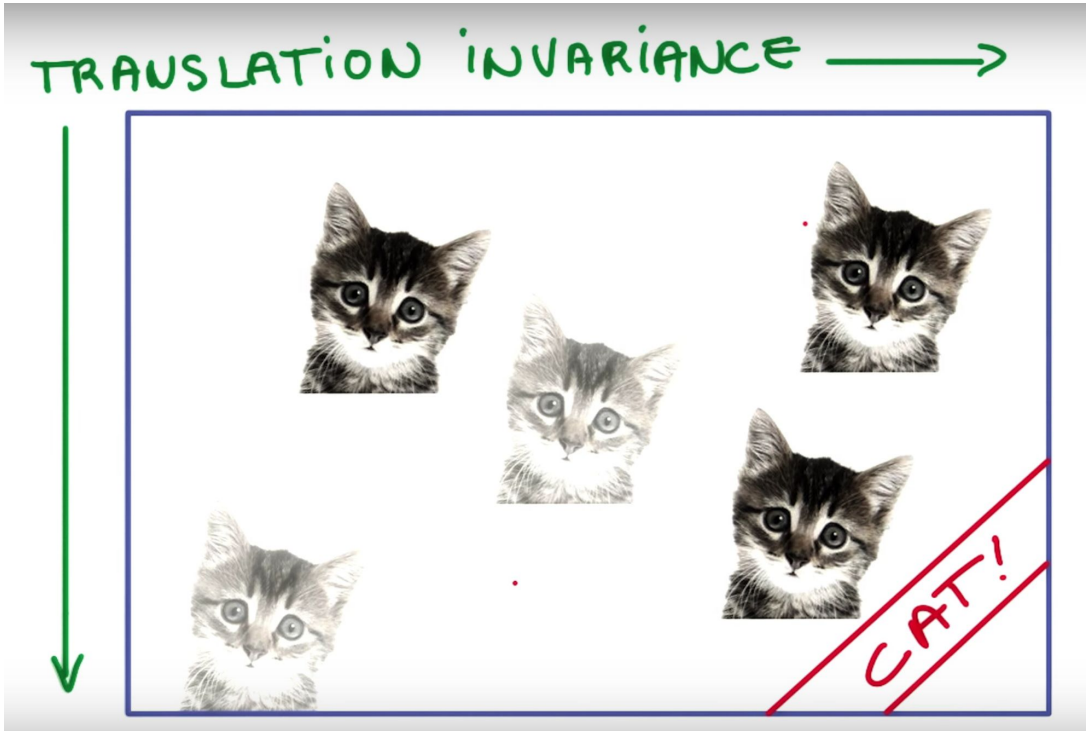
https://europepmc.org/article/med/25140147

Gestalt Principles

- Our brains tend to organize visual elements into groups



Key Property: Translation Invariance
→ Local Features + Combinations



https://www.cc.gatech.edu/~san37/post/dlhc-cnn/

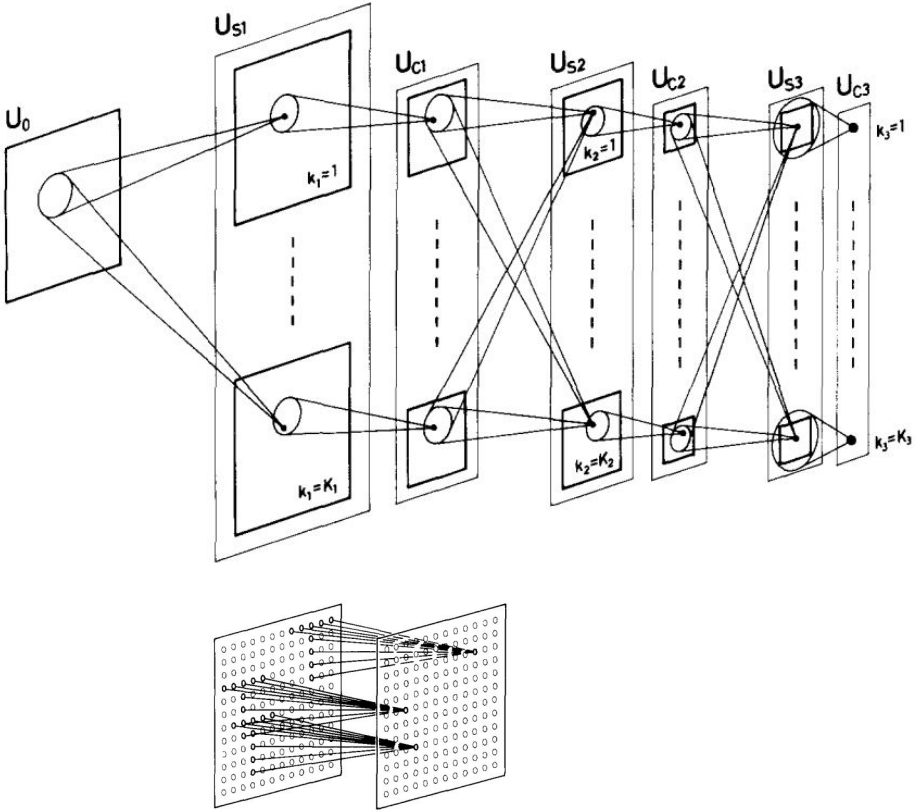
Neocognitron [Fukushima 1980]

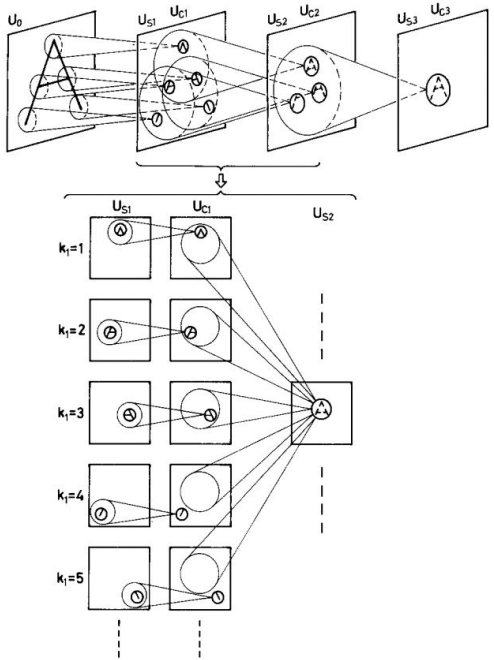
Biol. Cybernetics 36, 193–202 (1980)

Biological Cybernetics

Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position

Kunihiko Fukushima
NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan



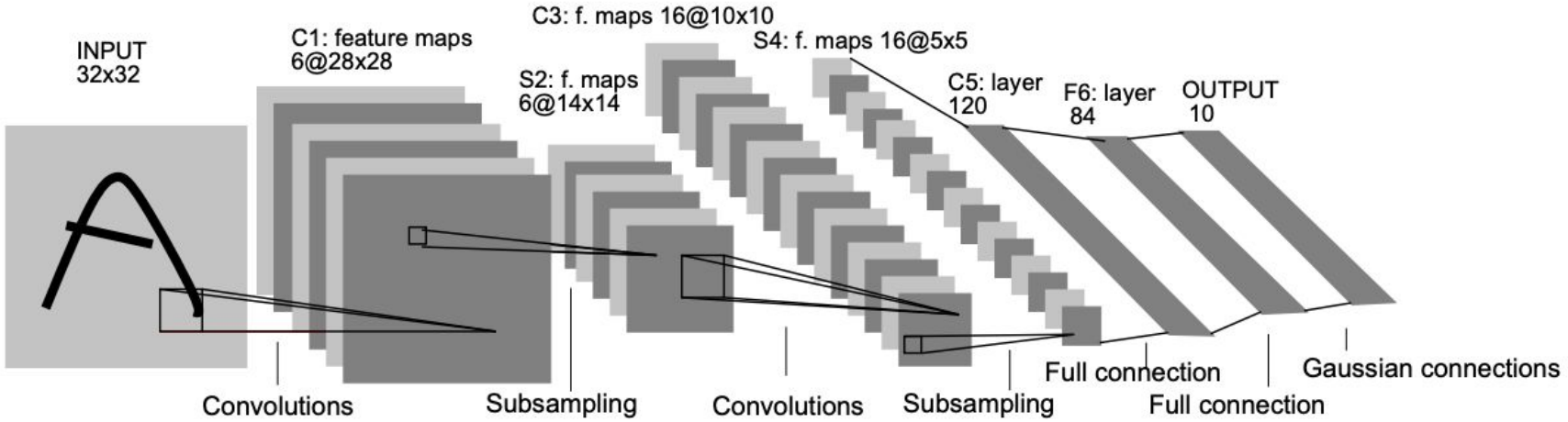


LeNet-5 [LeCun et al. 1998]

PROC. OF THE IEEE, NOVEMBER 1998

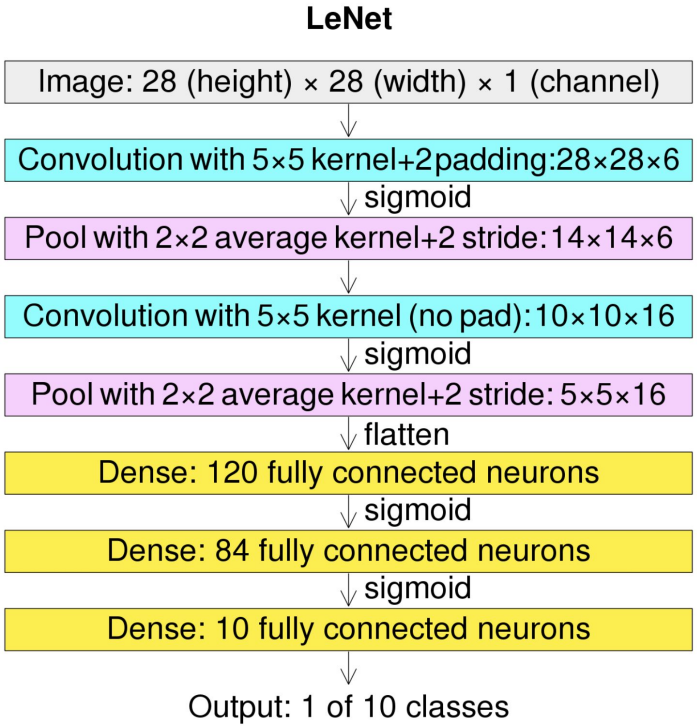
Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner



LeNet-5 is among a few "deep" neural networks in the pre-Deep Learning age

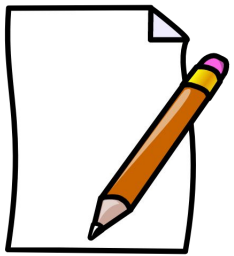
Model Architecture of LeNet



Key Techniques: Convolution & Pooling

Key Concepts in CNN

- Convolution
- Pooling
- Fully-connected Layer



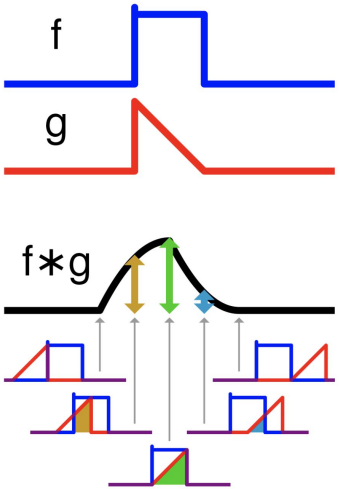
Key Concepts in CNN

- **Convolution**
- Pooling
- Fully-connected Layer

What is Convolution?

- Mathematically, the integral of the product of the two functions after one is reversed and shifted

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau$$



- For Deep Learning, we consider convolution in discrete space

Convolution for Edge Detection

- Extracting edge patterns by applying convolution filters (aka kernels) to the target image

Convolution: Terminology

$x_{1,1}$	$x_{1,2}$	…	$x_{1,M}$
…			
…			
$x_{N,1}$	…		$x_{N,M}$

Input image /Feature map

*

w_1	w_2
w_3	w_4

Convolutional filter (aka kernel)

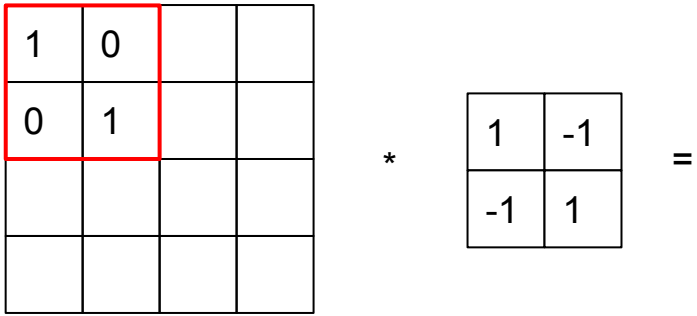
=

Feature map

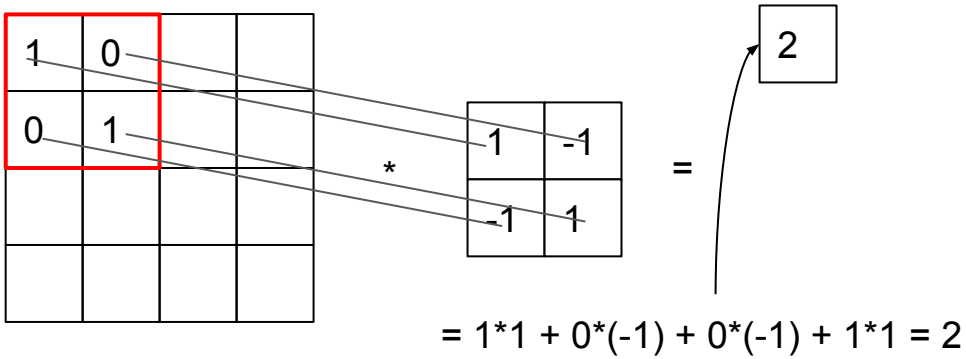
Convolutional Filter Examples

Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

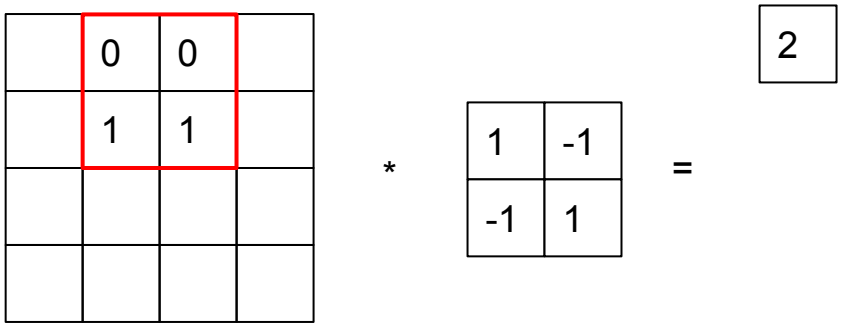
Convolution: Toy Example



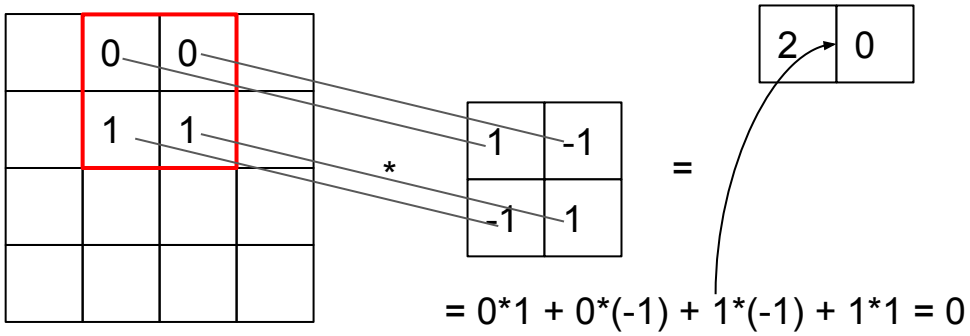
Convolution: Toy Example



Convolution: Toy Example



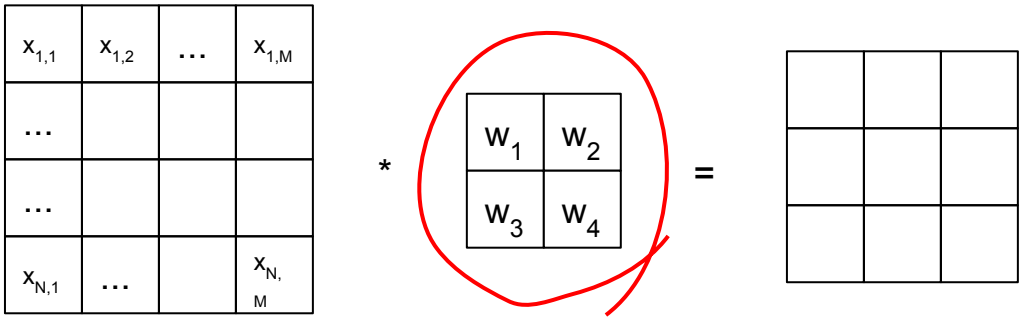
Convolution: Toy Example

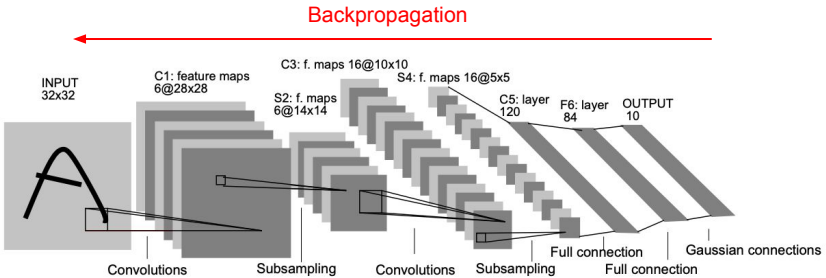


In this example, we use a **stride size of 1**

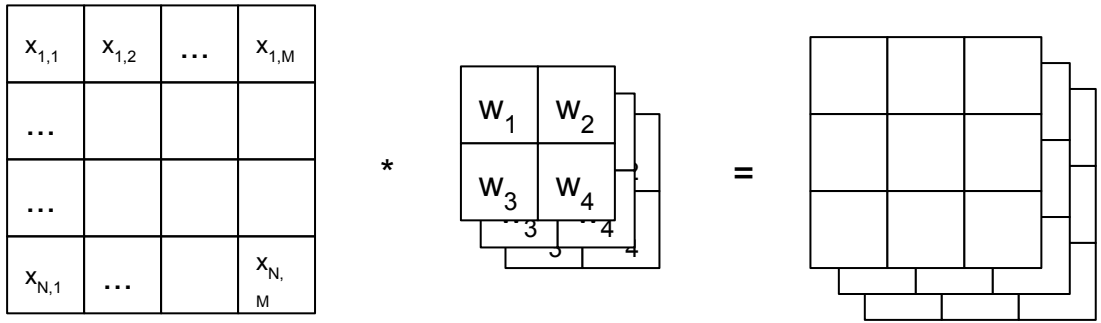
Convolutional Filters as NN Blocks

- We can “learn” convolutional filters using backpropagation!





A Convolutional Layer Can Have More Than One Filter



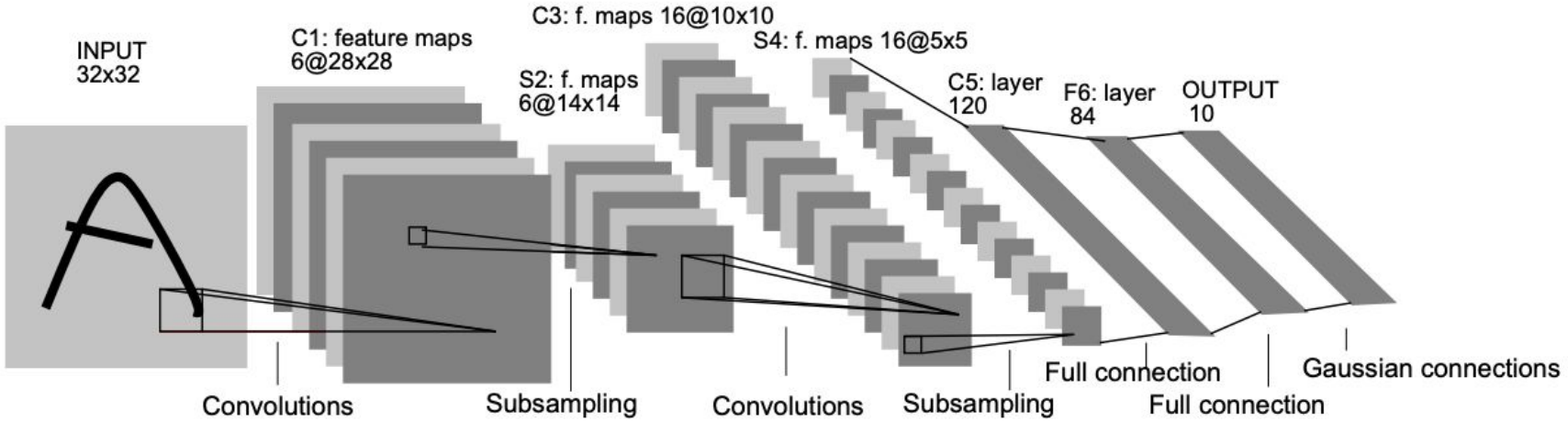
in_channel = 1 **out_channel = 3**

```
nn.Conv2d(in_channels=1, out_channels=3,
          kernel_size=2, stride=1, padding=0)
```

CONV2D

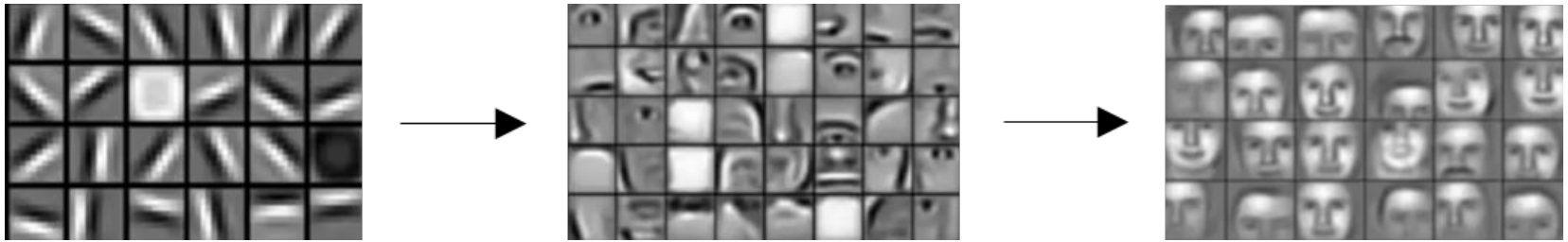
CLASS `torch.nn.Conv2d(`*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None*`)` [SOURCE]

LeNet-5 Has Multiple Convolutional Layers



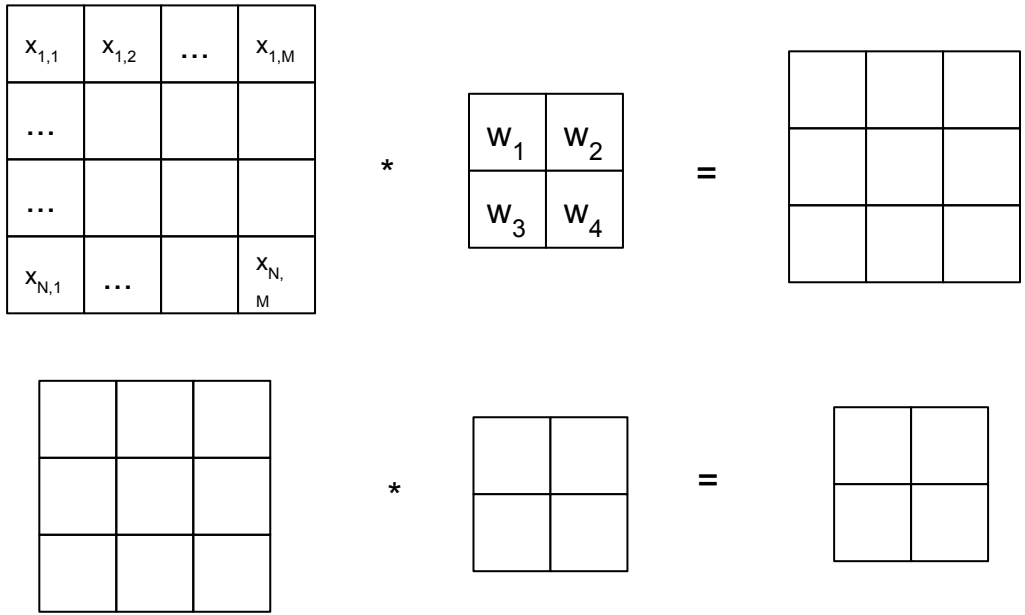
Why Need Deep Layers?

- (Shallow) Edge detection → Texture patterns → Object patterns (Deep)



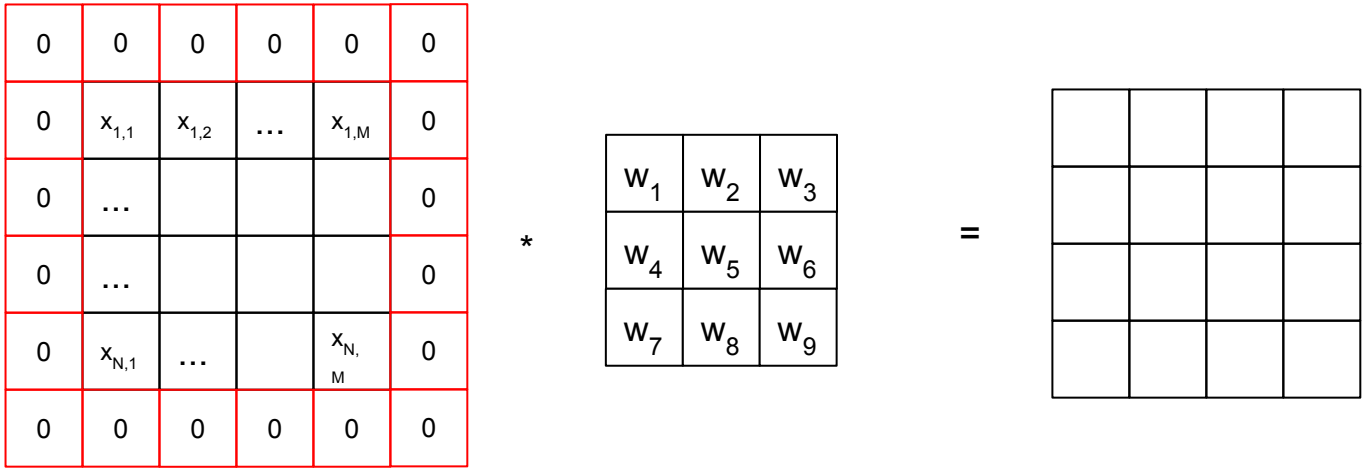
Feature Maps Keep Shrinking!

- Feature maps are getting smaller as we apply more convolution filters
- Any solutions?



Feature Maps Keep Shrinking! → Padding

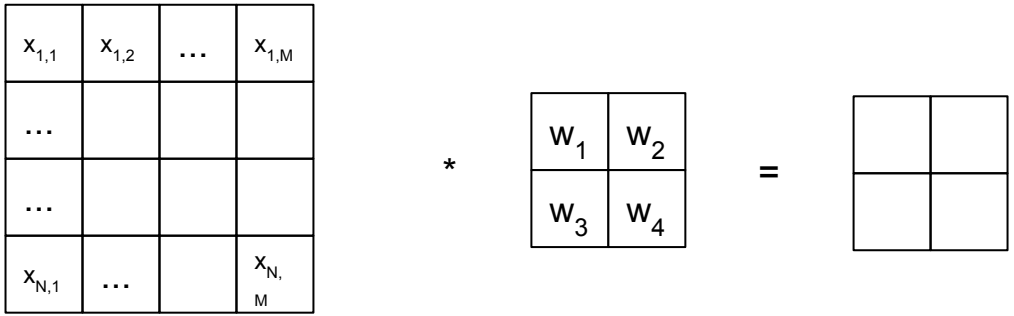
- Feature maps are getting smaller as we apply more convolution filters
- Any solutions? → Padding!



4x4 + 1 padding | 3x3 (stride=1) | 4x4

Quiz 1

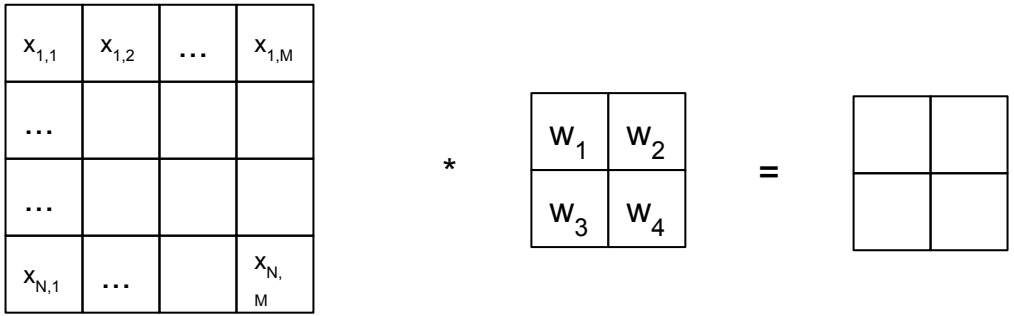
- What's the shape of the output feature map?



4x4 2x2 (stride=2) 2x2

Quiz 1

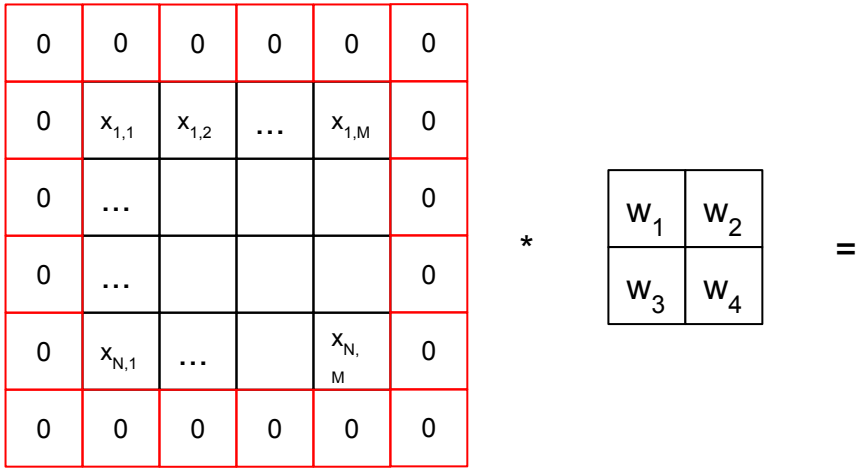
- What's the shape of the output feature map?



4x4 2x2 (stride=2) 2x2

Quiz 2

- What's the shape of the output feature map?

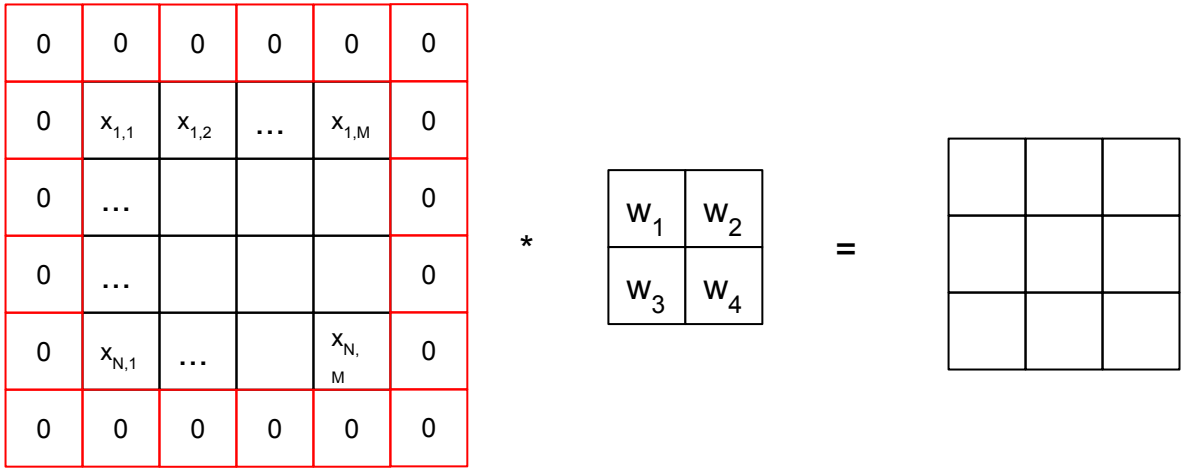


4x4 + **1 padding**

2x2 (**stride=2**)

Quiz 2

- What's the shape of the output feature map?

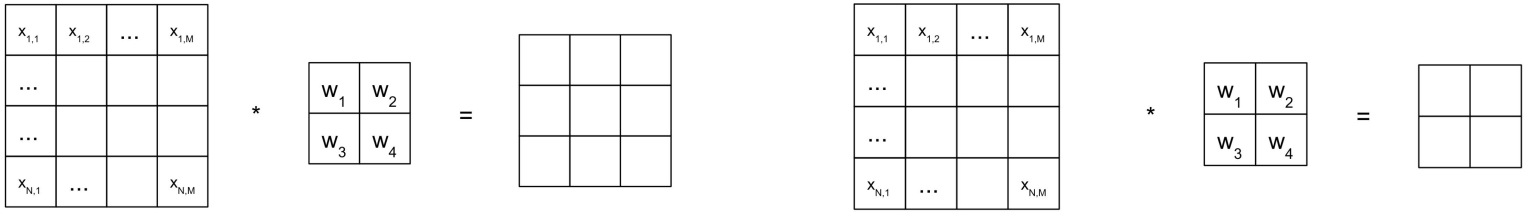


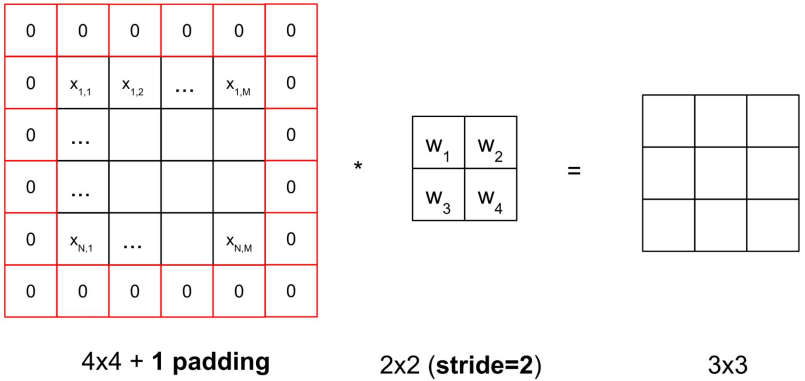
4x4 + **1 padding**

2x2 (**stride=2**)

3x3

Quiz: Can you derive the general form?





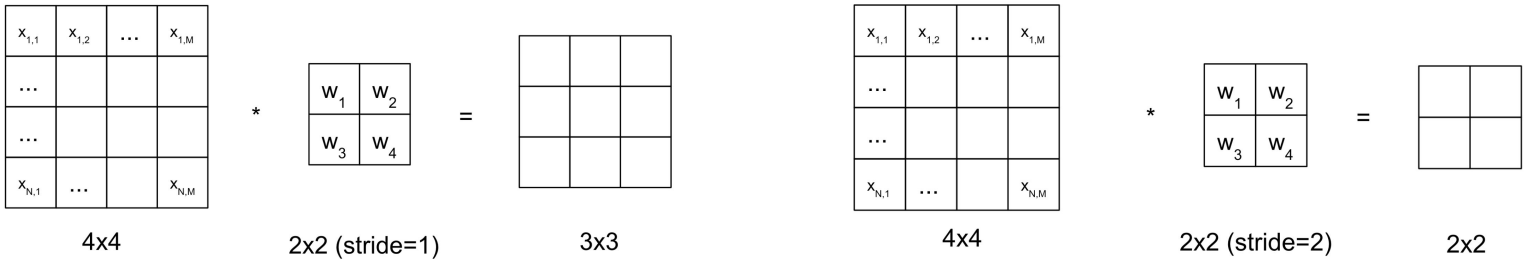
N x N
(+p)

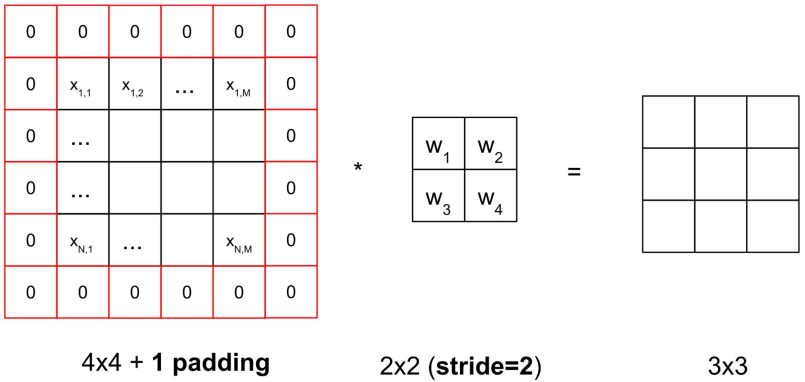
*

k x k
(stride=s)

= ? x ?

Quiz: Can you derive the general form?





N x N (+p) * k x k (stride=s) = ? x ?

$$? = \left\lfloor \frac{N + 2p - k}{s} \right\rfloor + 1$$

Key Concepts in CNN

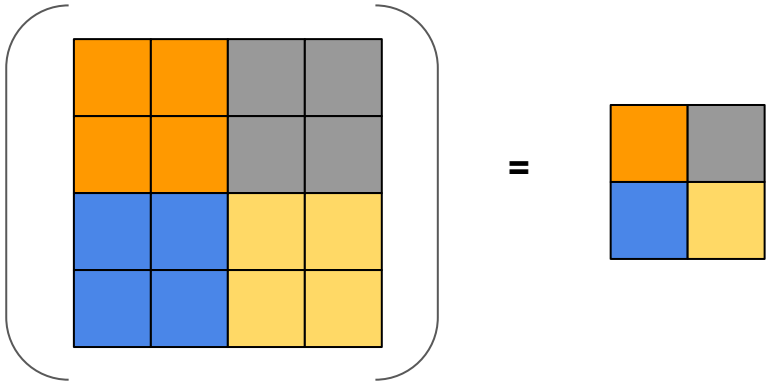
- Convolution
- **Pooling**
- Fully-connected Layer

What is Pooling? Why Do We Need This?

- Subsampling process to **adjust the resolution**
- Typical pooling choices: Mean or Max pooling

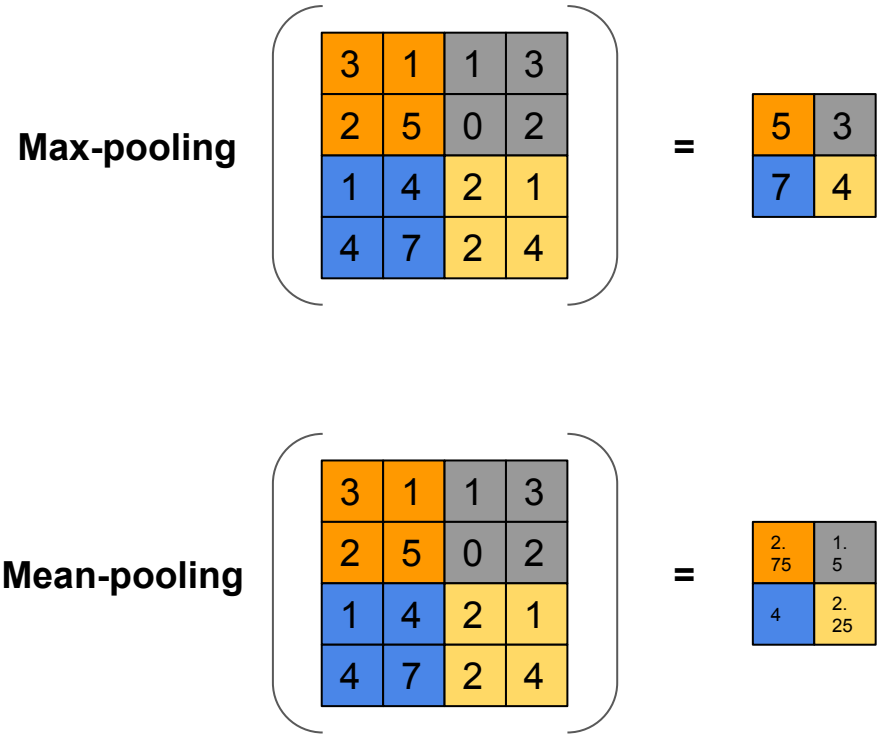
Pooling

Pooling size: 2x2

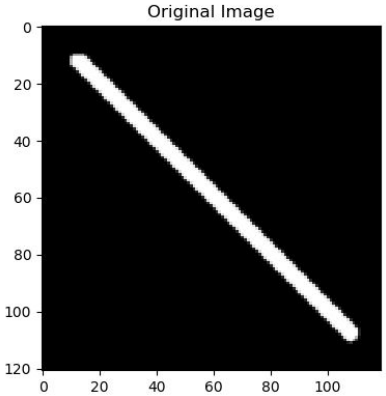


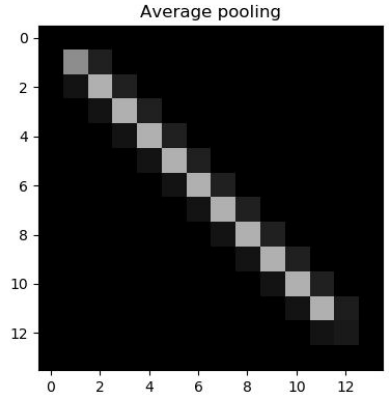
=

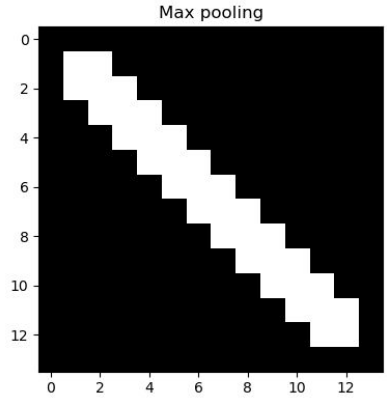
Mean and Max Pooling

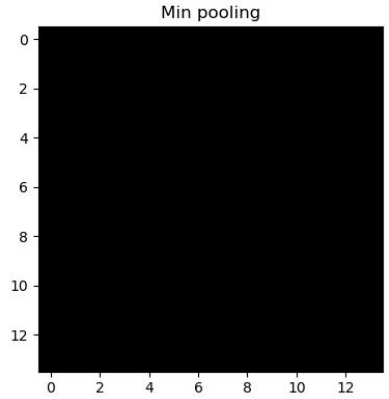


Pooling examples









Convolution & Pooling in PyTorch

CONV2D

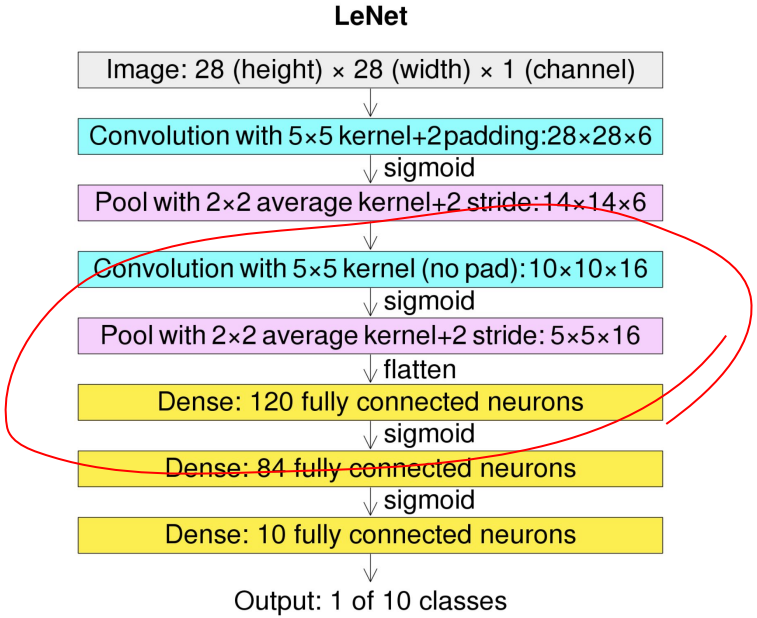
CLASS `torch.nn.Conv2d(`*in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None*`)` [SOURCE]

AVGPOOL2D

CLASS `torch.nn.AvgPool2d(`*kernel_size, stride=None, padding=0, ceil_mode=False, count_include_pad=True, divisor_override=None*`)` [SOURCE]

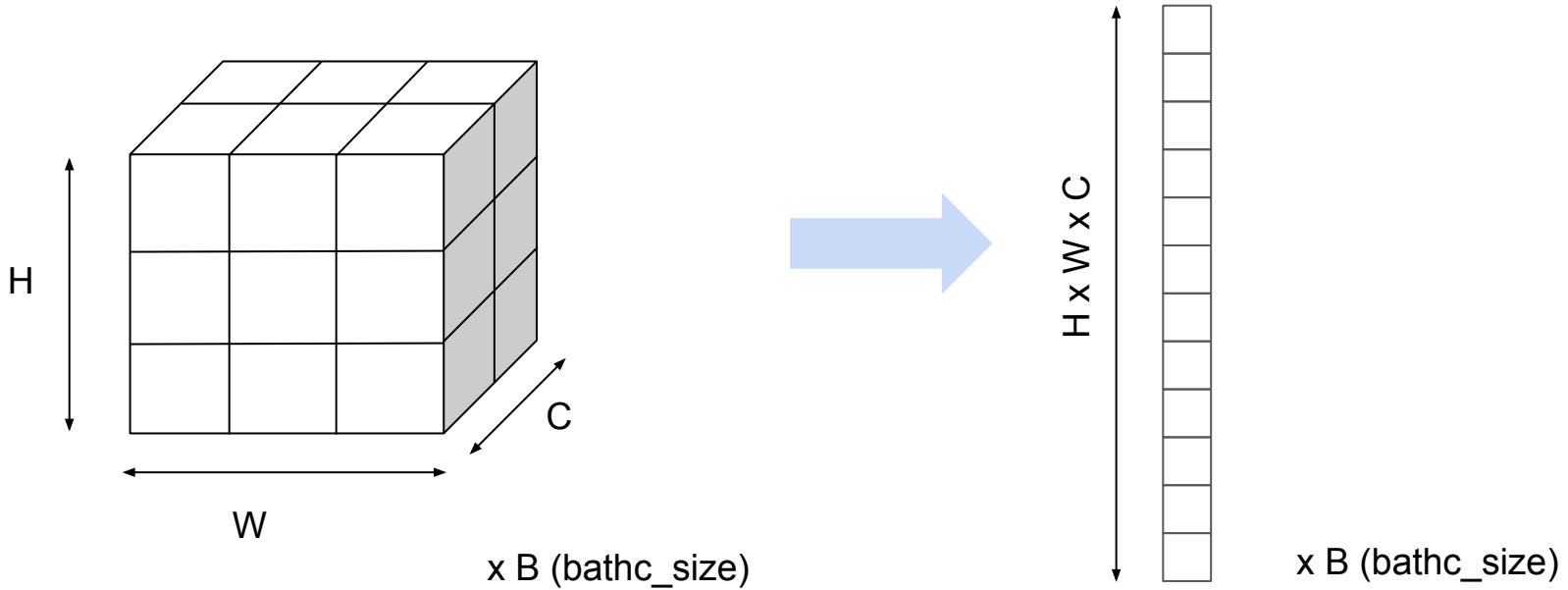
Key Concepts in CNN

- Convolution
- Pooling
- Fully-connected Layer



Flattening Feature Maps into Vectors

- Feature maps are not directly compatible with `Linear` layers
  - (batch_size, #channel, height, width) =! (batch_size, dim)



Flattening Tensor with PyTorch

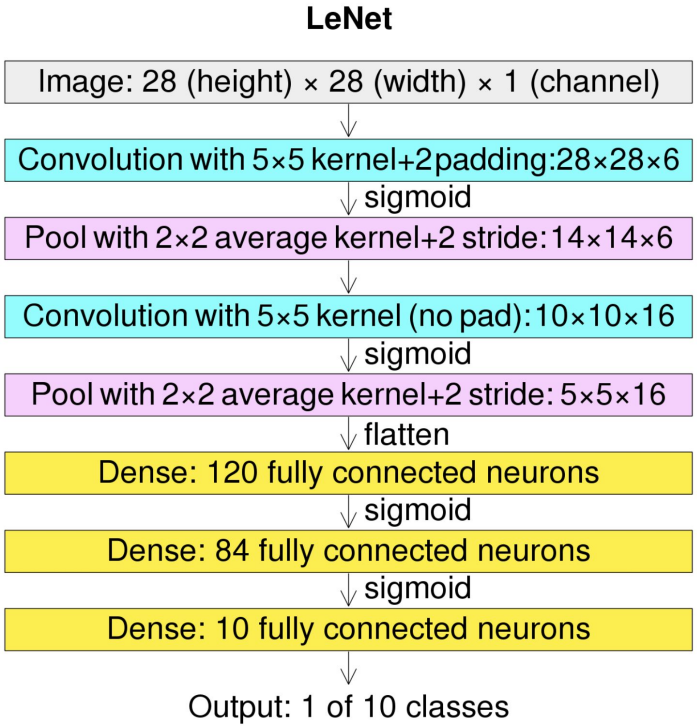
```
>>> torch.flatten(torch.Tensor, start_dim)

>>> out.shape

(8, 5, 28, 28)

>>> torch.flatten(out, 1)

???

>>> torch.flatten(out, 2)

???
```

Flattening Tensor with PyTorch

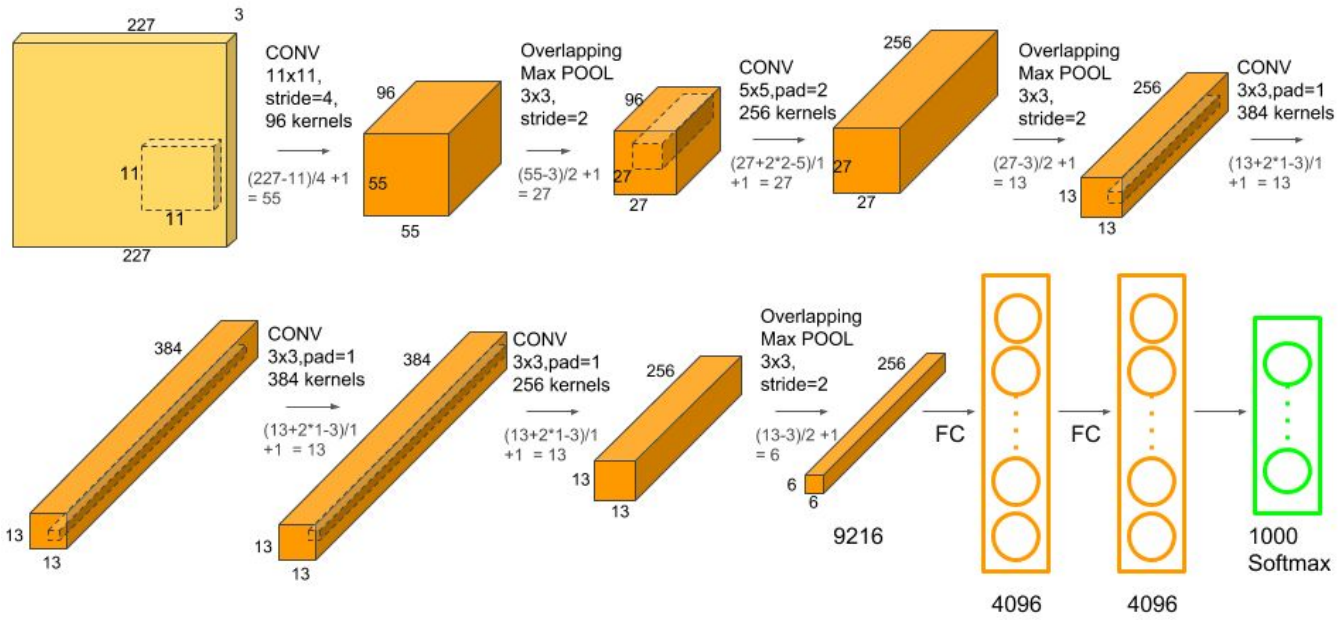
```
>>> torch.flatten(torch.Tensor, start_dim)

>>> out.shape

(8, 5, 28, 28)

>>> torch.flatten(out, 1)

(8, 3920)

>>> torch.flatten(out, 2)

(8, 5, 784)
```

Any questions?

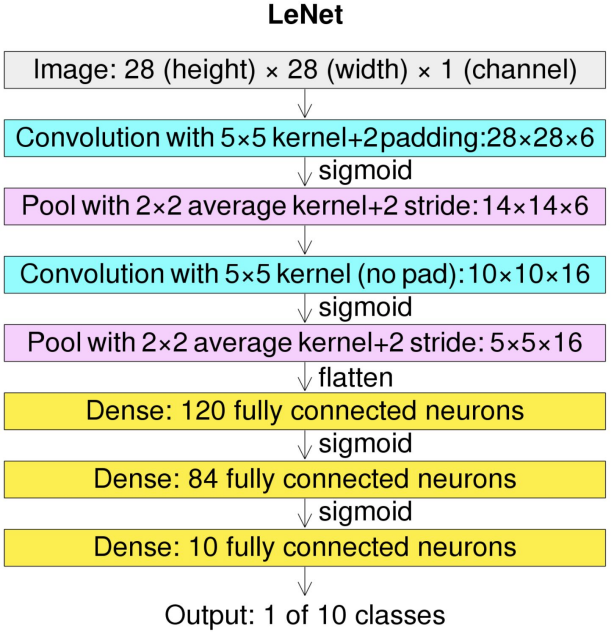
Model Architecture of LeNet

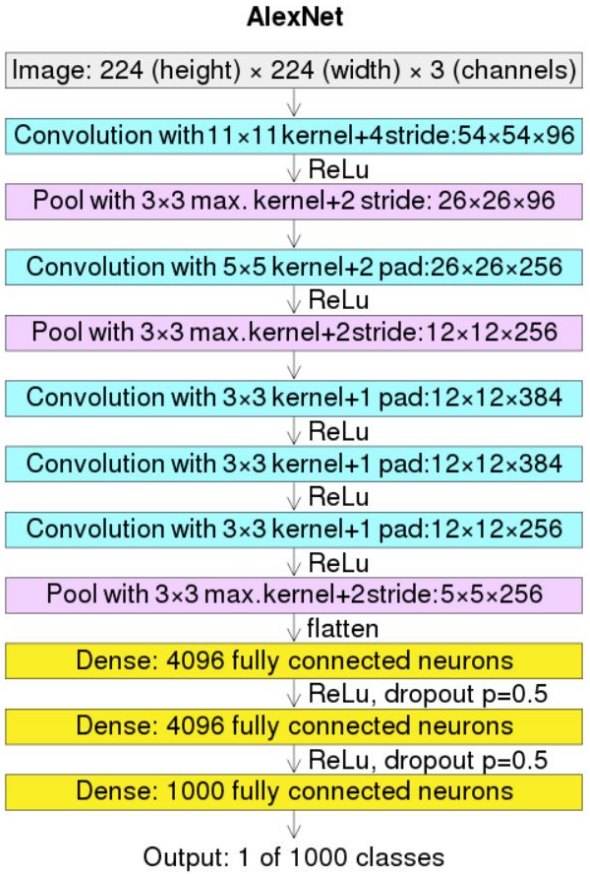


Does this figure make sense?



Model Design → Blueprint → PyTorch code





Hands-on Session: Playing with Convolution/Pooling

- https://colab.research.google.com/github/suhara/cis6930-fall2021/blob/main/notebooks/cis6930_week3a_convolutional_neural_networks.ipynb

Summary

- Translation Invariance & Local features
- Key Concepts in Convolutional Neural Networks
  - Convolutional layers
  - Pooling layers
  - Fully-connected layers (+ flattening)
- Designing CNN models =~ Designing feature map processing

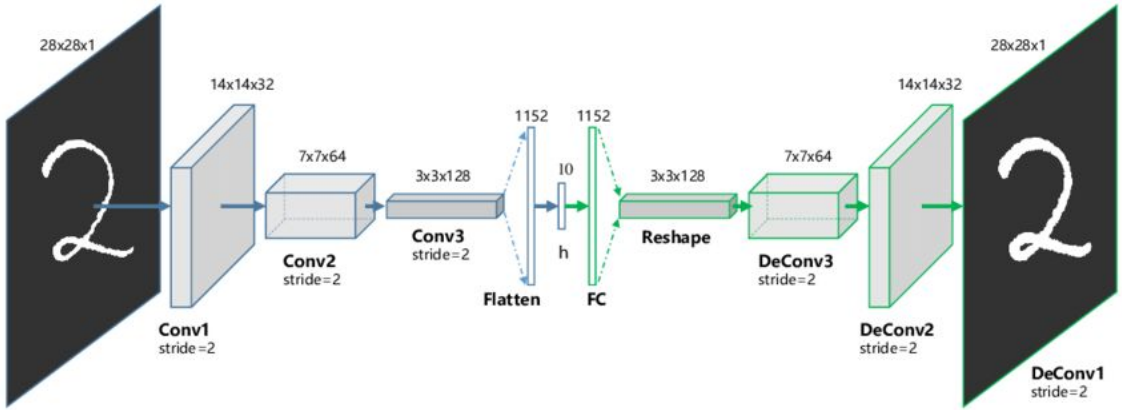
10min Break

Convolutional Filters + Autoencoder
= Convolutional Autoencoder!

10min

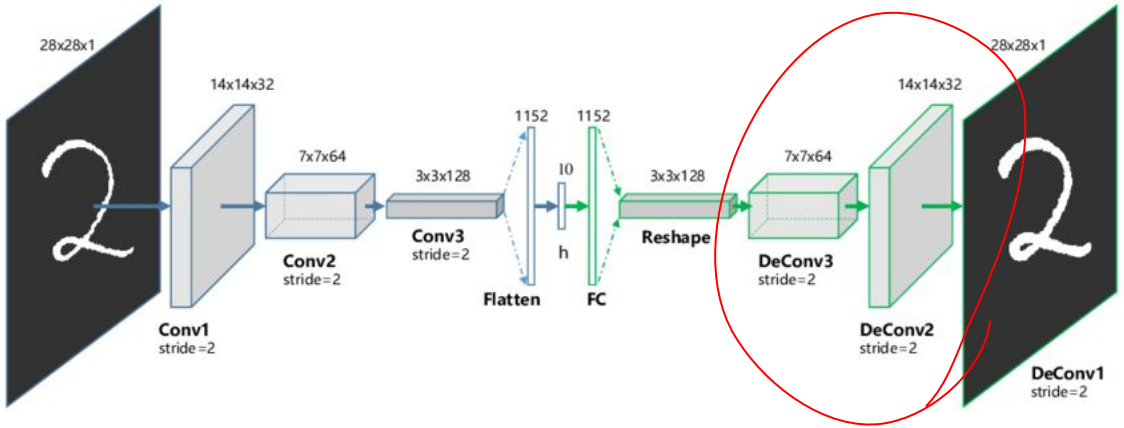
Convolutional Autoencoder

- Convolutional layers keep or shrink feature maps (i.e., **downsampling**)
- How can we expand feature maps? (i.e., **upsampling**)

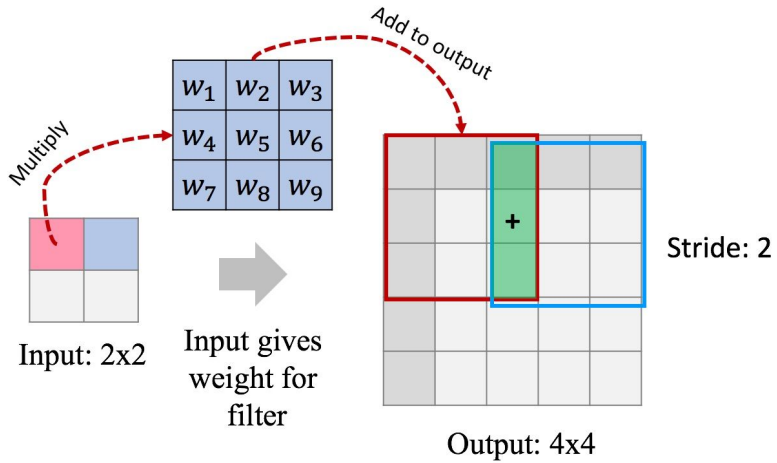


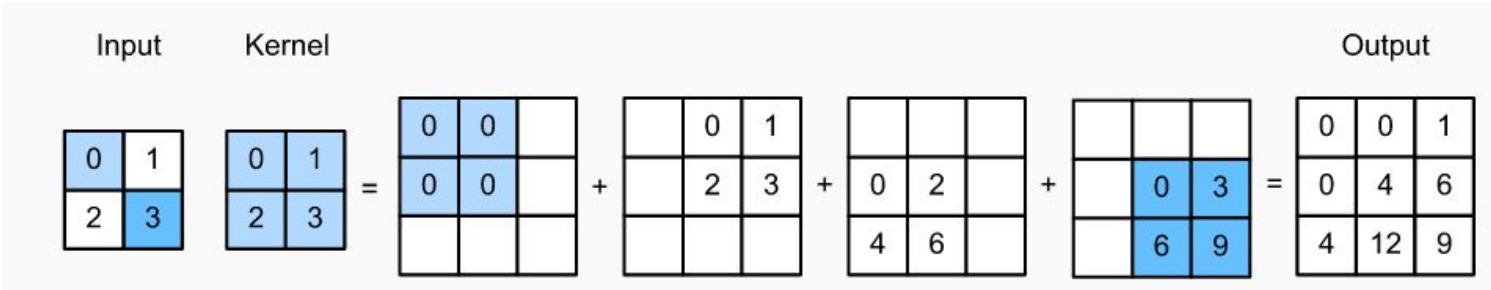
Convolutional Autoencoder

- Convolutional layers keep or shrink feature maps (i.e., **downsampling**)
- How can we expand feature maps? (i.e., **upsampling**)



Transposed Convolution





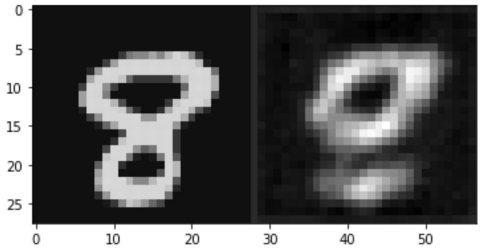
https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba

Demo: Convolutional Autoencoder

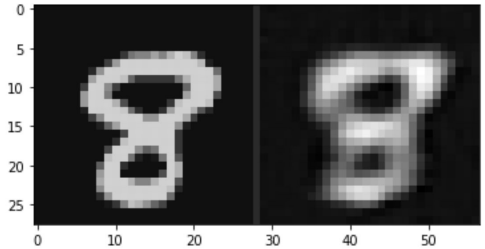
- https://colab.research.google.com/github/suhara/cis6930-fall2021/blob/main/notebooks/cis6930_week3a_convolutional_neural_networks.ipynb

Convolutional Autoencoder

Epoch 0
Training loss: 0.4371
Validation loss: 0.2830

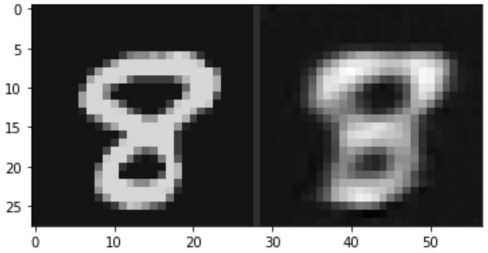


Epoch 4
Training loss: 0.2227
Validation loss: 0.2192



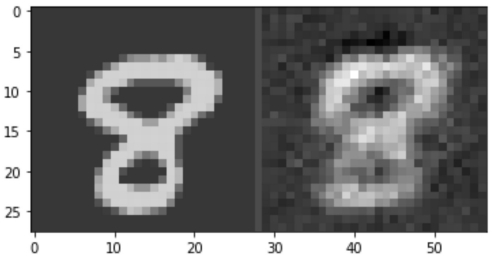
Elapsed time: 37.0 [sec]

Epoch 9
Training loss: 0.1994
Validation loss: 0.1985

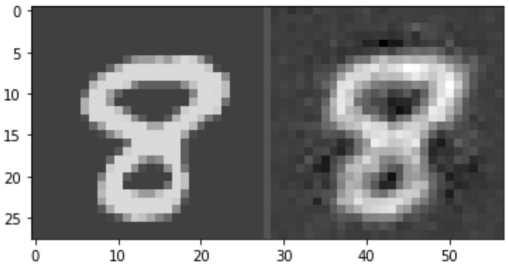


Single-layer Autoencoder

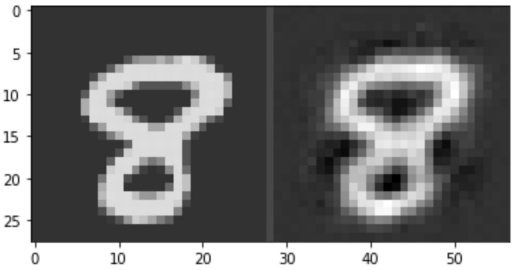
Epoch 0
Training loss: 15.2723
Validation loss: 9.5237 /16



Epoch 4
Training loss: 5.0810
Validation loss: 4.9002 /16

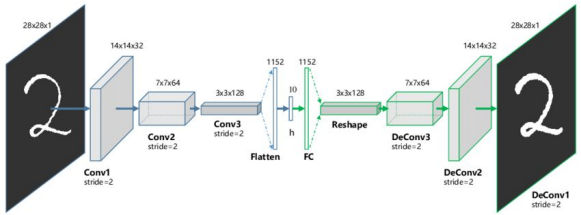


Epoch 9
Training loss: 3.9312
Validation loss: 3.8820 /16



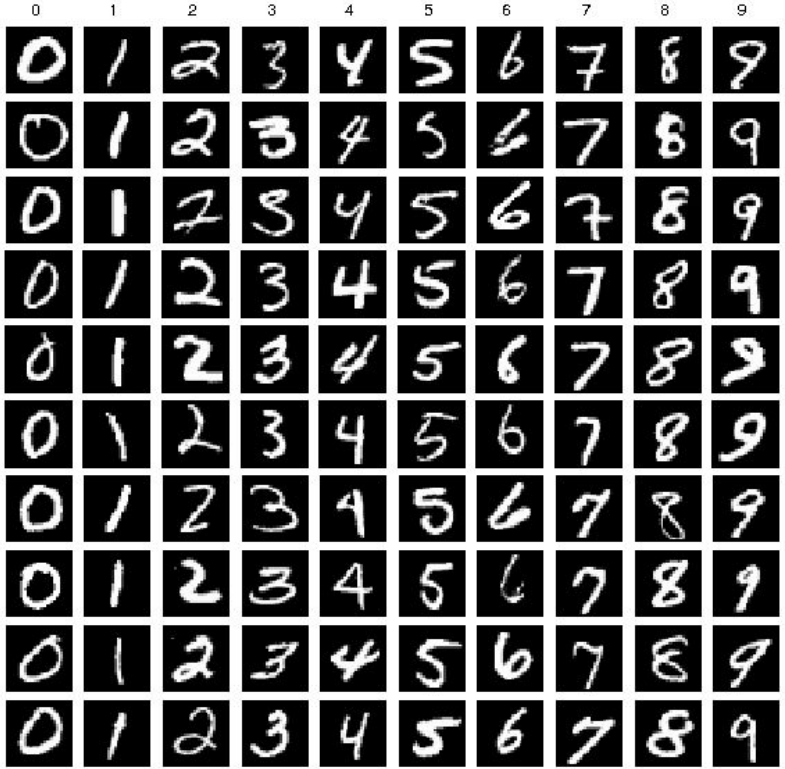
Tips: Keep the tensor shape information in comments

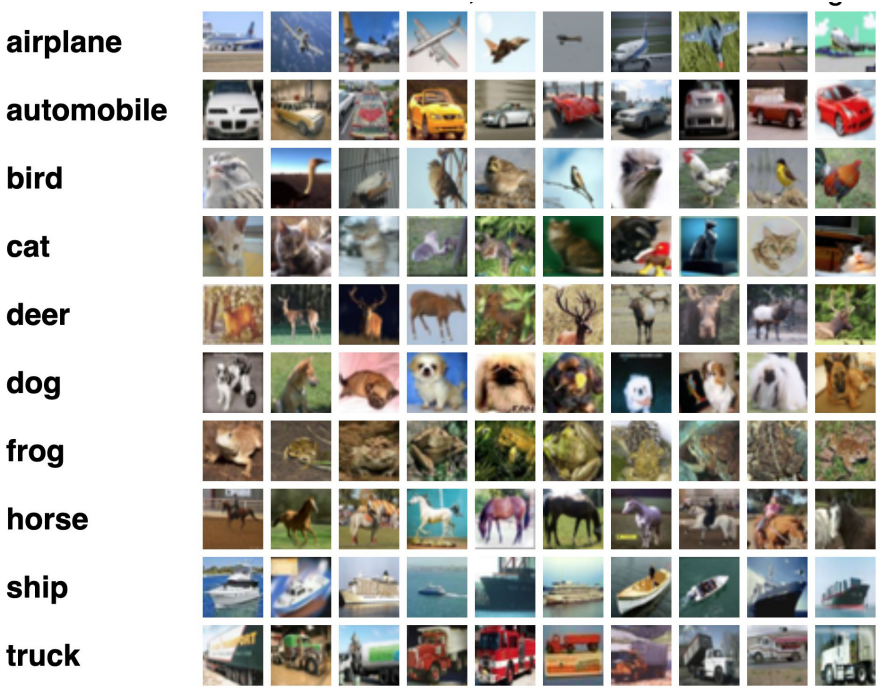
```
class ConvAutoEncoder(nn.Module):
    def __init__(self,
                 hidden_dim: int = 16):
        super().__init__()
        self.encoder = nn.Sequential(nn.Conv2d(in_channels=1,
                                               out_channels=8,
                                               kernel_size=3,
                                               stride=1,
                                               padding=1), #-> (_, 8, 28, 28)
                                     nn.ReLU(),
                                     nn.Conv2d(in_channels=8,
                                               out_channels=4,
                                               kernel_size=3,
                                               stride=1,
                                               padding=1), #-> (_, 4, 28, 28)
                                     nn.MaxPool2d(kernel_size=2, stride=2)) #-> (_, 1, 28, 28)
        self.flatten = nn.Flatten(start_dim=1) #-> (-, 1*28*28)
        self.hidden1 = nn.Linear(1*28*28, hidden_dim)  # -> 16
        self.hidden2 = nn.Linear(hidden_dim, 1*28*28)  # -> 4*28*28
        self.unflatten = nn.Unflatten(1, (1, 28, 28))  # -> (1, 28, 28)
        self.decoder = nn.Sequential(nn.ConvTranspose2d(in_channels=1,
                                                        out_channels=4,
                                                        kernel_size=3,
                                                        stride=1,
                                                        padding=1),  #->(_, 4, 28, 28)
                                     nn.ReLU(),
                                     nn.ConvTranspose2d(in_channels=4,  #->(_, 1, 28, 28)
                                                        out_channels=1,
                                                        kernel_size=3,
                                                        stride=1,
```



Grayscale → RGB

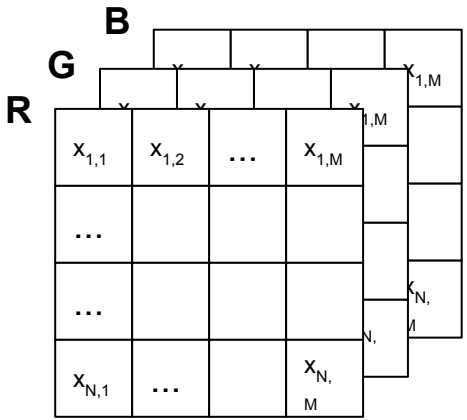
What about Color Images?





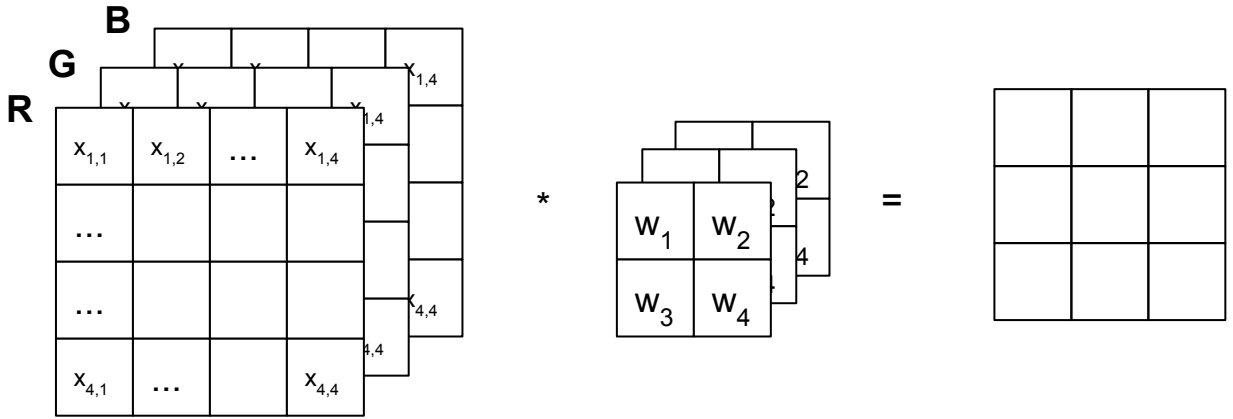
Let’s think about color images. Now what?

- Height x Width x **Channel**



Nothing new anymore! We already discussed feature maps with multiple channels :)

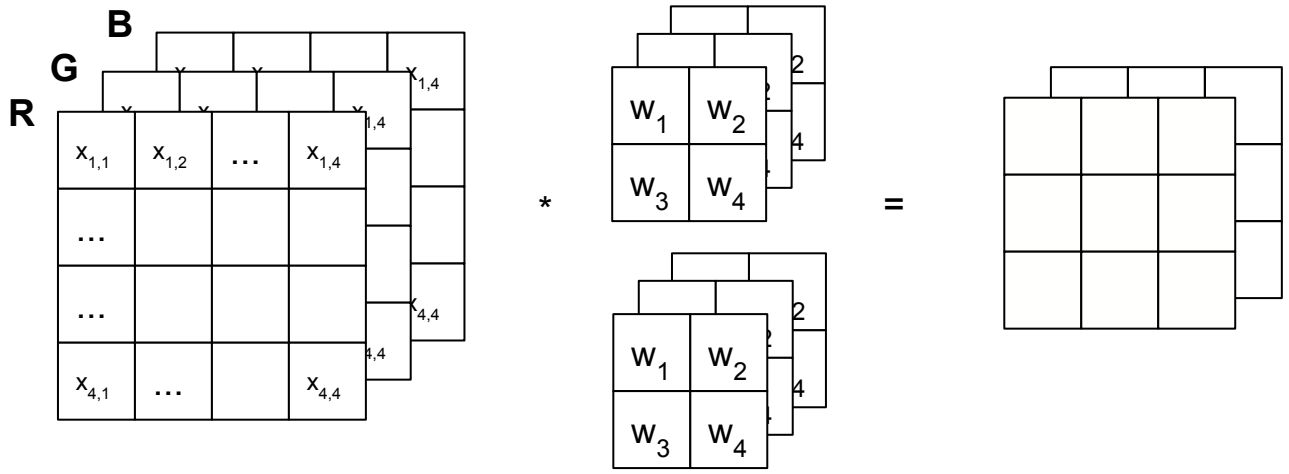
Convolutions on RGB image



(3, 4, 4) (3, 2, 2) (1, 3, 3)

Convolutions on RGB image

```
Conv2d(in_channels=3, out_channels=2, kernel_size=2, stride=1, padding=0)
```



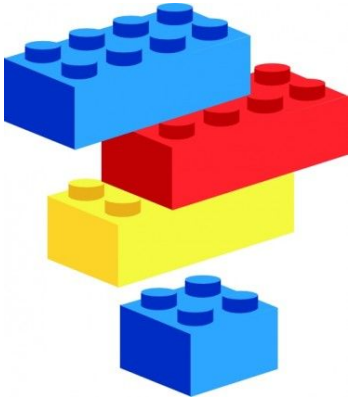
(1, 3, 4, 4) (2, 3, 2, 2) (1, **2**, 3, 3)

(Always) Check the Shape using Python Interactive Shell

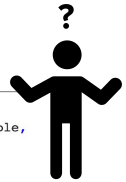
0s

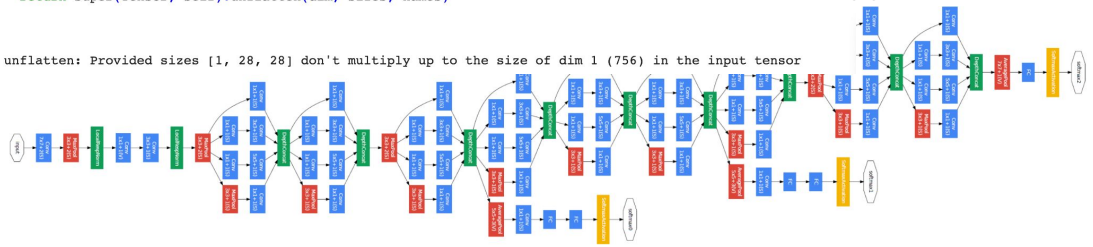
```
filter1 = nn.Conv2d(in_channels=3, out_channels=2,
                    kernel_size=2, stride=1, padding=0)
filter1.weight.shape
```

```
torch.Size([2, 3, 2, 2])
```



```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-25-15a70062d1b8> in <module>()
      5           "n_epochs": 10}
      6 model = ConvAutoEncoder()      # lr = 0.0001
----> 7 output = train(model, train_dataset, valid_dataset, config)

                                  6 frames
/usr/local/lib/python3.7/dist-packages/torch/_tensor.py in unflatten(self, dim, sizes)
    865         if isinstance(sizes, OrderedDict) or (isinstance(sizes, (tuple, list)) and isinstance(sizes[0], (tuple,
list))):
    866             names, sizes = unzip_namedshape(sizes)
--> 867         return super(Tensor, self).unflatten(dim, sizes, names)
    868
    869

RuntimeError: unflatten: Provided sizes [1, 28, 28] don't multiply up to the size of dim 1 (756) in the input tensor
```





Hands-on Session: Get Familiar with 4d Tensor

- (B, C, H, W) or (N, C, H, W)
- https://colab.research.google.com/github/suhara/cis6930-fall2021/blob/main/notebooks/cis6930_week3a_convolutional_neural_networks.ipynb

10min

Keywords

- Convolution/Pooling filter
  - Kernel size
  - Stride
  - Padding

- Tensor shape (BCHW)
  - **B**atch size
  - **C**hannel
  - **H**eight
  - **W**idth

Conv2d

- `Conv2d(in_channels, out_channels, kernel_size, stride, padding)`

CONV2D

CLASS `torch.nn.Conv2d(`*`in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros', device=None, dtype=None`*`)` [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

where $\star$ is the valid 2D cross-correlation operator, N is a batch size, C denotes a number of channels, H is a height of input planes in pixels, and W is width in pixels.

Batch Normalization

- Deeper models → Gradient explosion problem (compound effect!)
  - (gradient > 1)$^{\#\text{ of layers}}$
  - Training process becomes unstable :(

- Batch Normalization
  - Standardization inside each batch

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

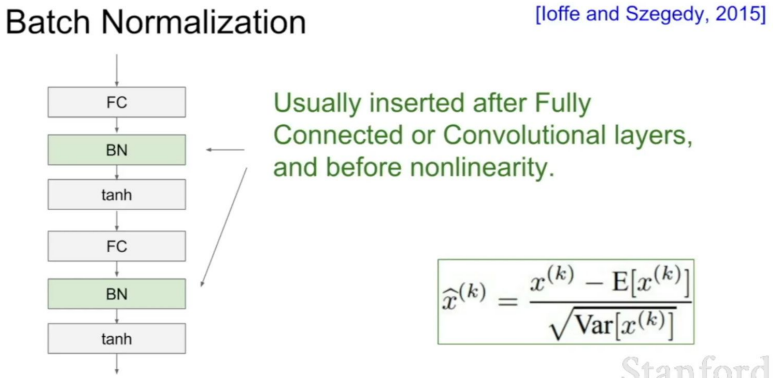
Docs > torch.nn > BatchNorm2d

BATCHNORM2D

CLASS `torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True, device=None, dtype=None)` [SOURCE]

Where Should I Insert Batch Normalization Step?

- No consensus!



When i get back to the code used for vgg16bn.py
i found that the fcblock defines the batch normalization after the nonlinearity

```
    def FCBlock(self):
        model = self.model
        model.add(Dense(4096, activation='relu'))
        model.add(BatchNormalization())
        model.add(Dropout(0.5))
```

https://forums.fast.ai/t/lesson-2-using-batch-normalization-after-non-linearity-or-before-non-linearity/4817

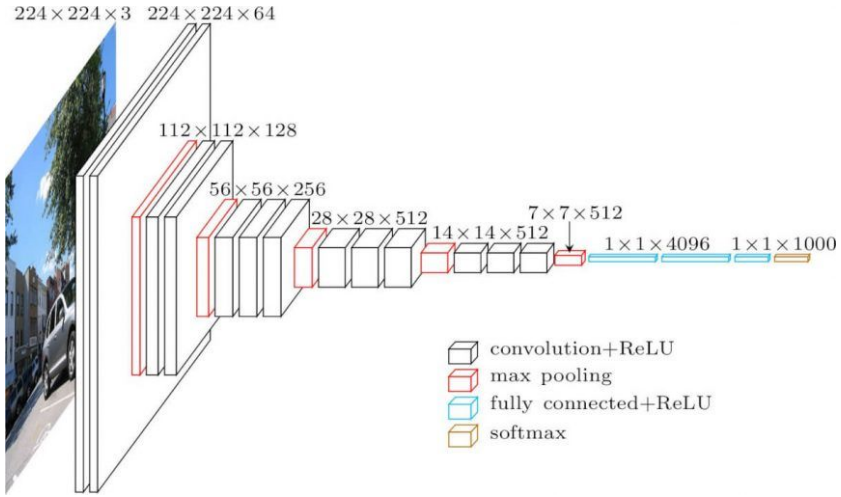
Advanced CNN Models

15min

Major CNN Models

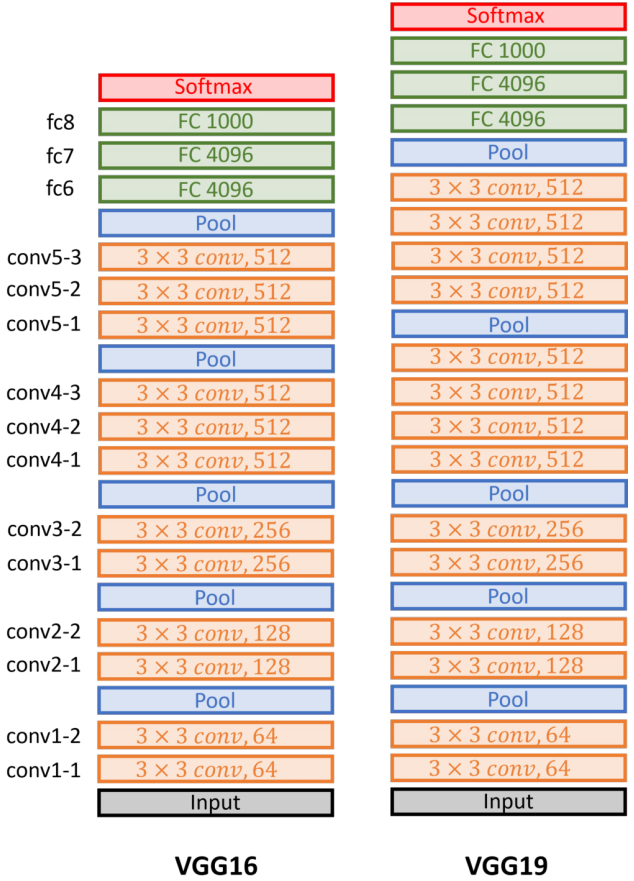
- ✅ LeNet-5 (1998)
- ✅ AlexNet (2012)
- VGG-16/19 (2014)
- GoogLeNet (2014)
- ResNet (2015)

VGG-16/19 (2014)



VGG-16

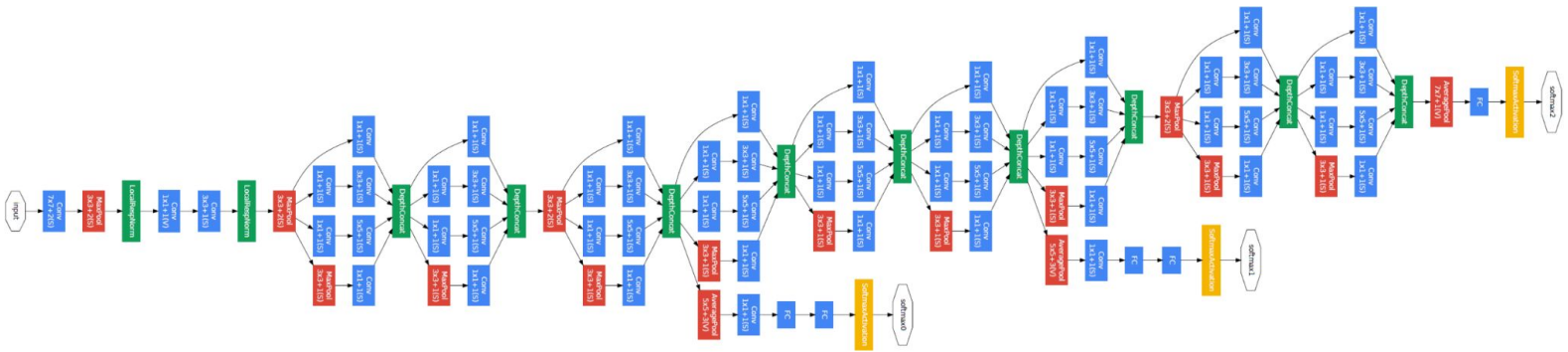
Visual **G**eometry **G**roup at Oxford
(the team's name at ILSVRC 14)

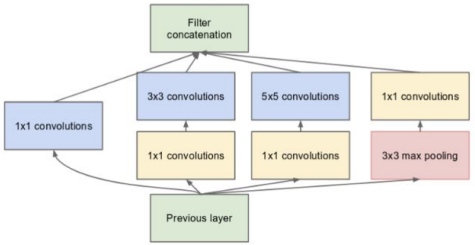


http://datahacker.rs/deep-learning-vgg-16-vs-vgg-19/

GoogLeNet (2014)

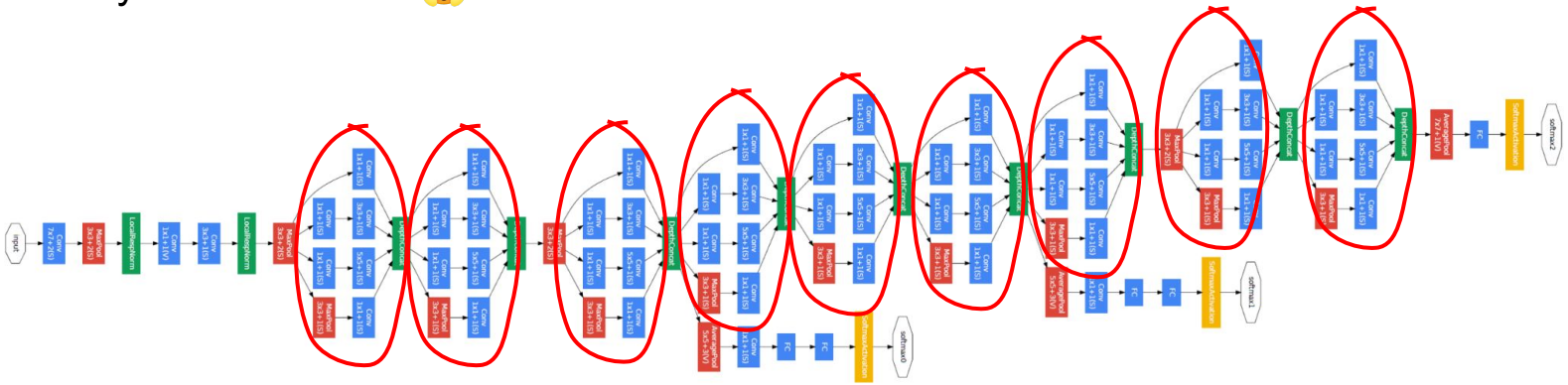
- 22-layer CNN model 😱





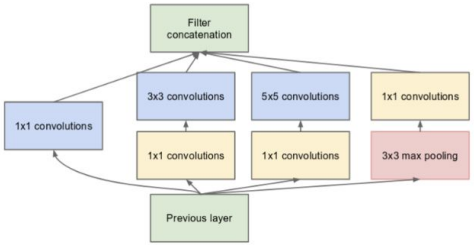
GoogLeNet's Key Idea (1): Inception V1

- 22-layer CNN model 😱



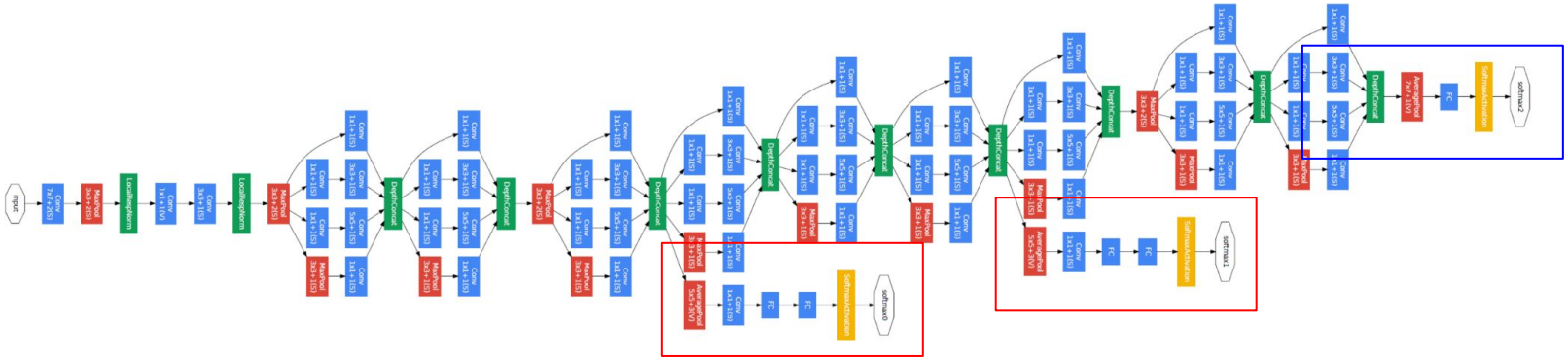
Inception v1 ("Network in Network" [Lin et al. 2013])

The use of 1x1 Conv filter to reduce the complexity

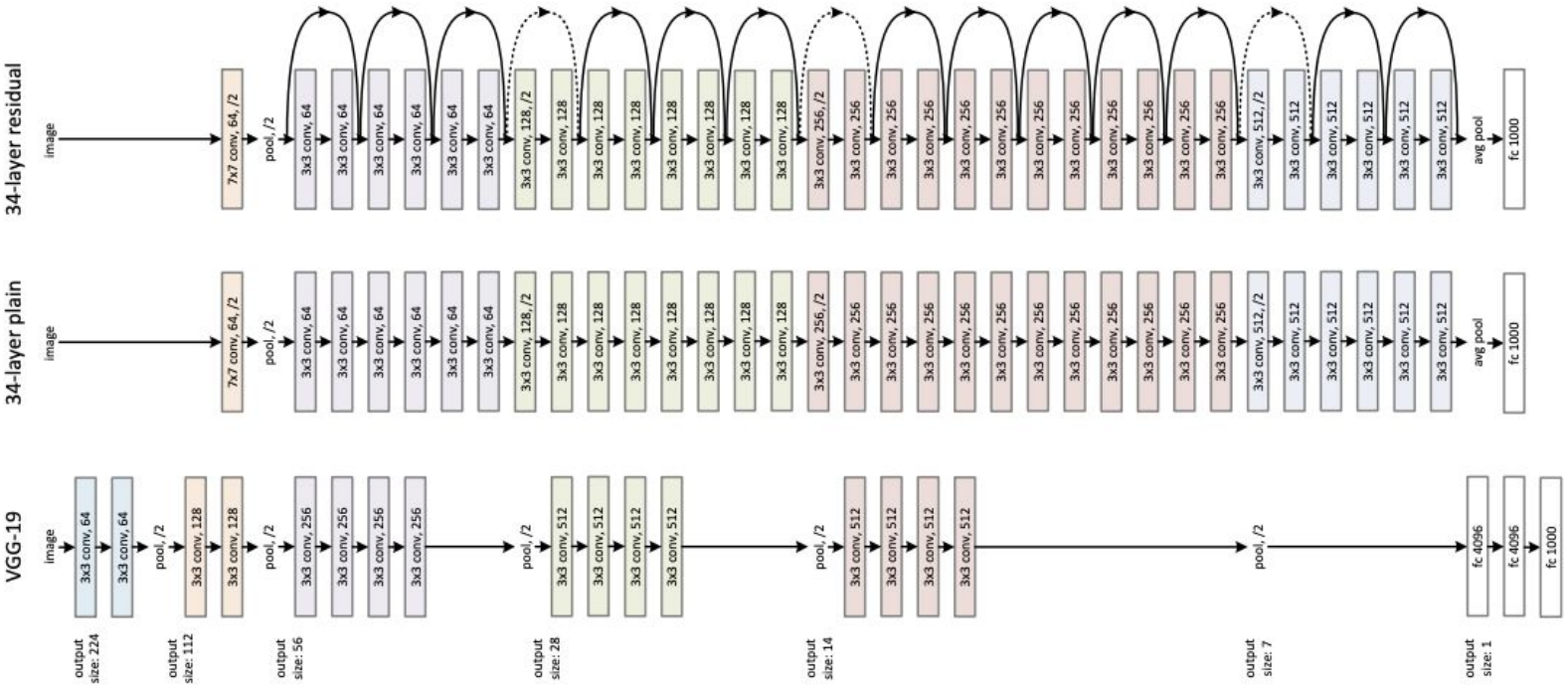


GoogLeNet's Key Idea (2): Auxiliary Loss

- 22-layer CNN model 😱

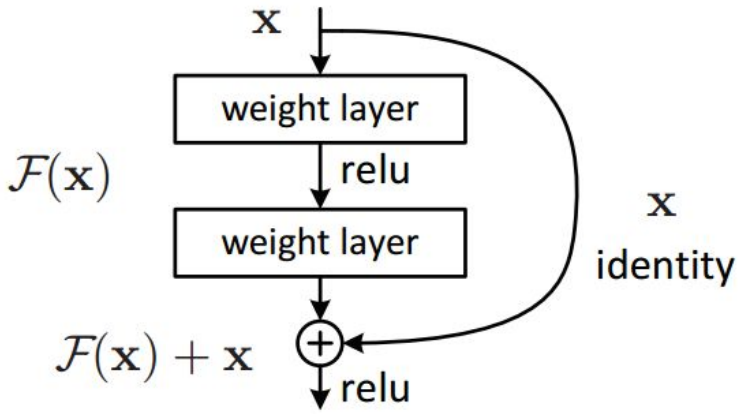


ResNet (2015)


VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
avg pool
fc 1000

ResNet's Key Idea: Residual Blocks

- Improving the gradient propagation effectiveness by adding connections between distant blocks (inspired by Highway Networks)



https://arxiv.org/abs/1512.03385

Residual Connections Help When CNNs Get Deeper

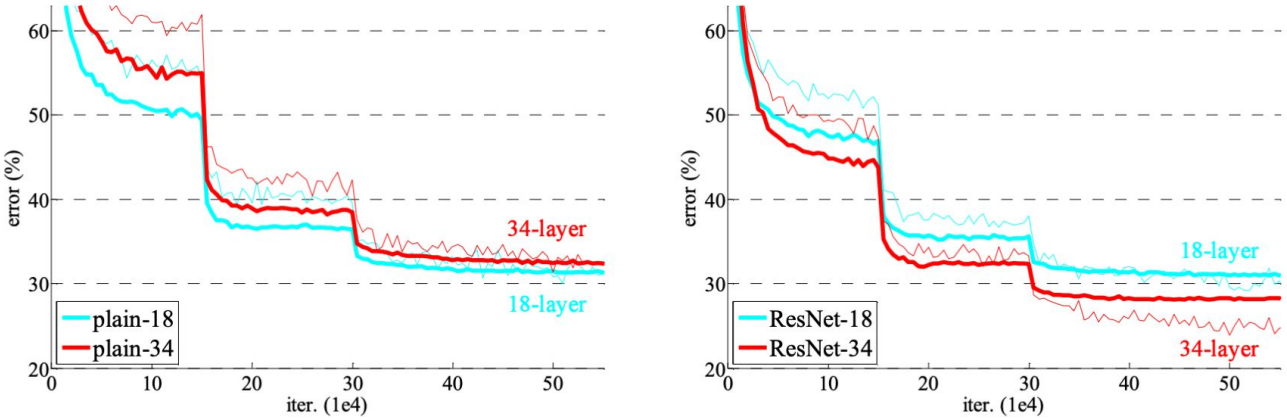


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

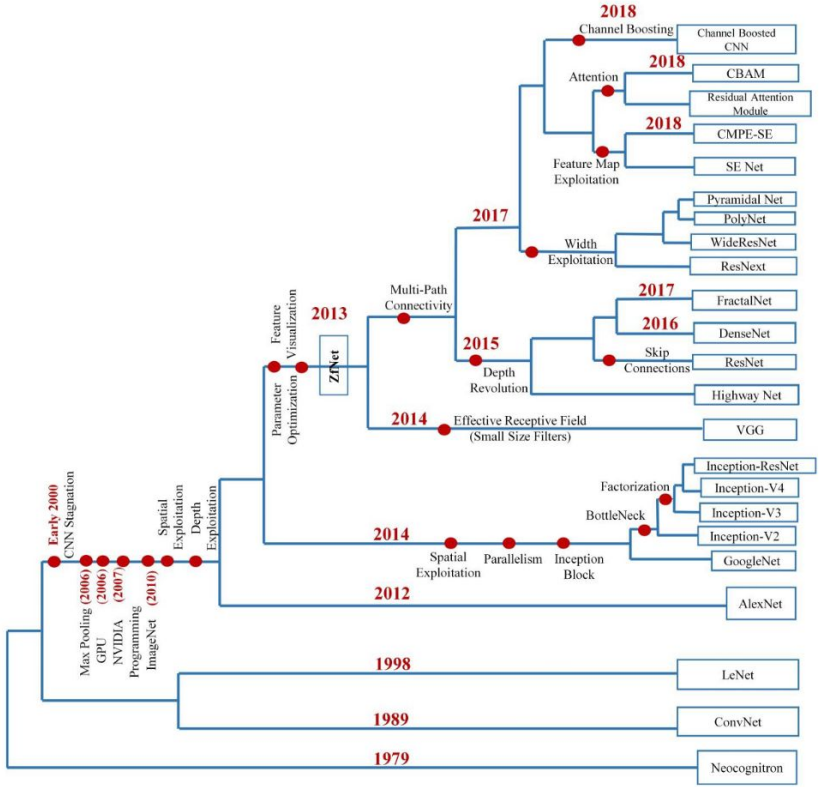
	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Table 2. Top-1 error (%, 10-crop testing) on ImageNet validation. Here the ResNets have no extra parameter compared to their plain counterparts. Fig. 4 shows the training procedures.

Major CNN Models

- ✅ LeNet-5 (1998)
- ✅ AlexNet (2012)
- ✅ VGG-16/19 (2014)
- ✅ GoogLeNet (2014)
- ✅ ResNet (2015)

CNN Model Tree

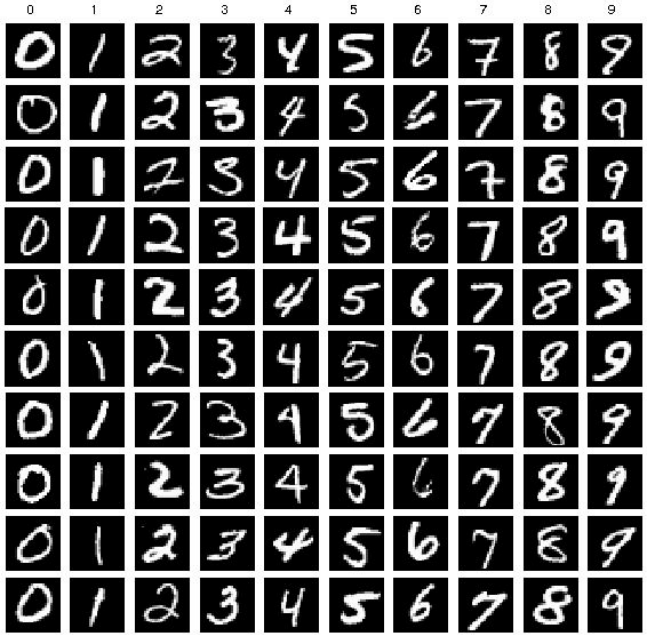


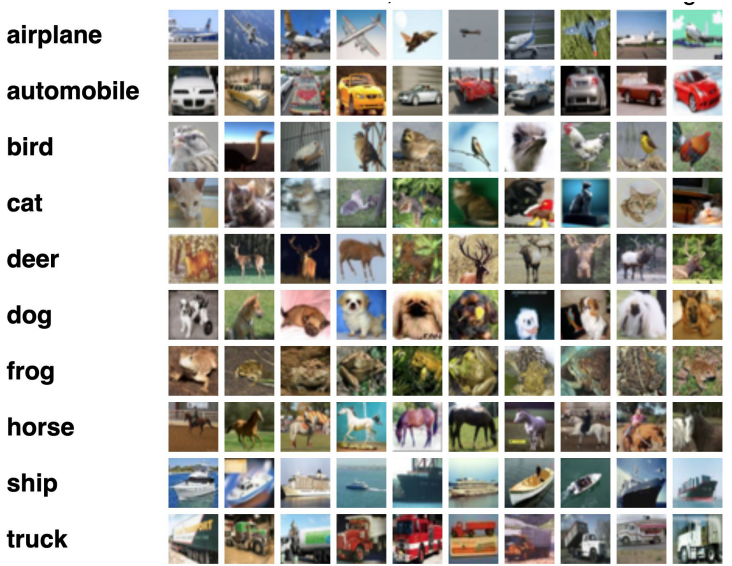
https://towardsdatascience.com/top-10-cnn-architectures-every-machine-learning-engineer-should-know-68e2b0e07201

CNN Applications

15min

Image Classification





Facial Recognition

- Binary classification: Predicting if the target person or not
- Multi-class classification: Detecting who the person is

Emotion Detection

- Multi-class classification

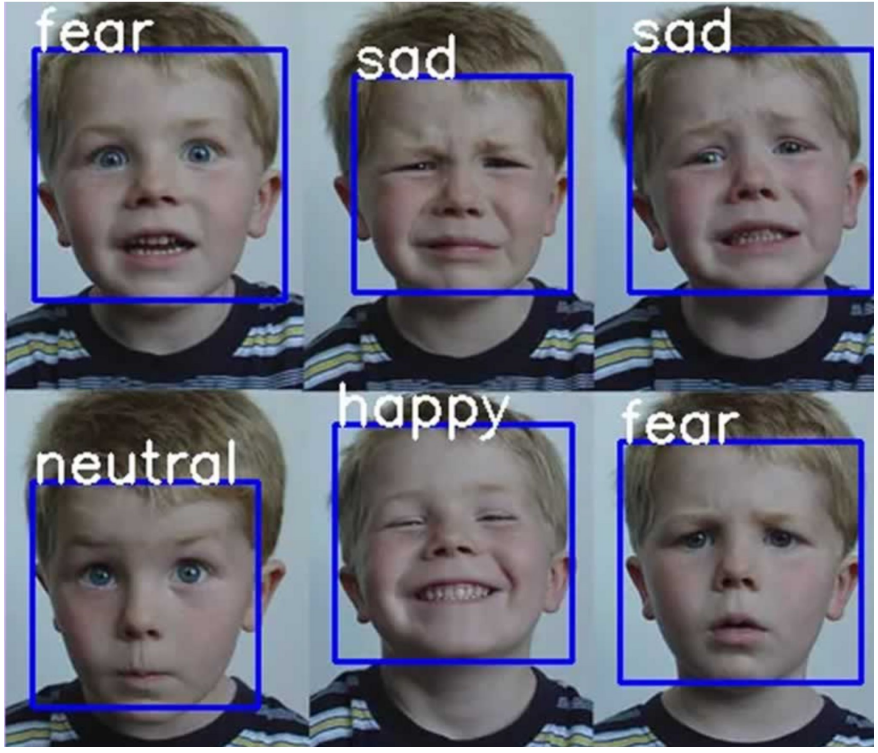
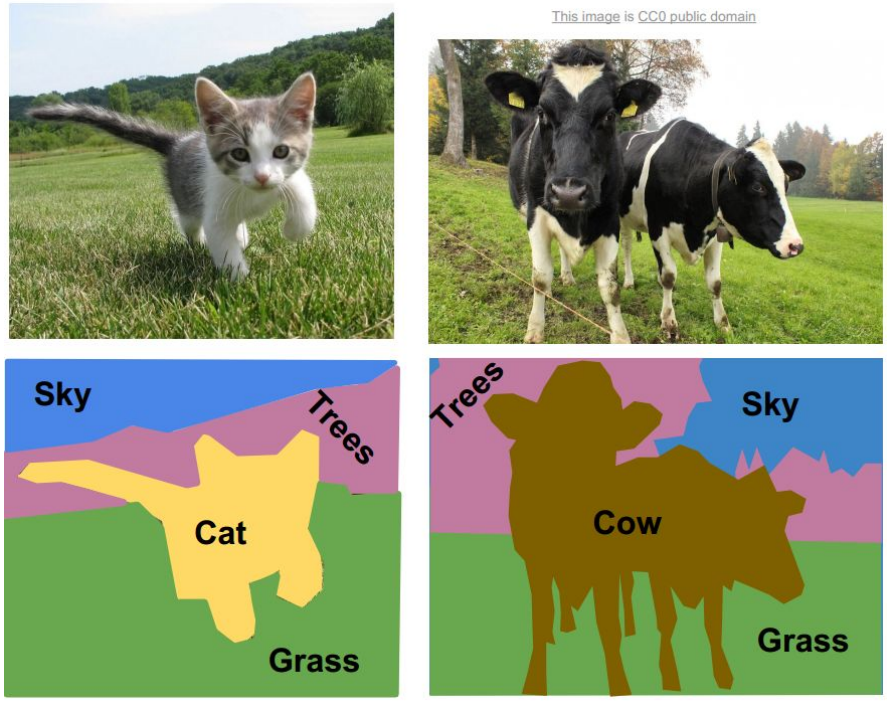


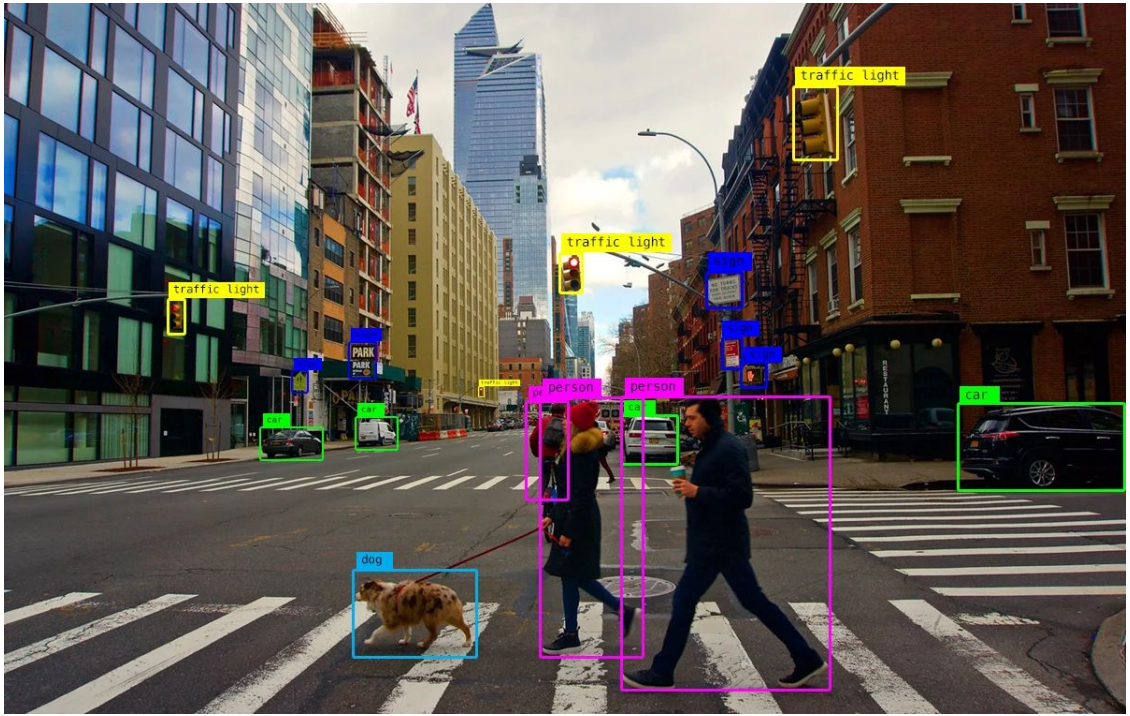
Image Segmentation

- Output layer: N x M pixels



Object Detection

- How can we model bounding box? Hint: ((x1, y1), (x2, y2)) + label



Summary

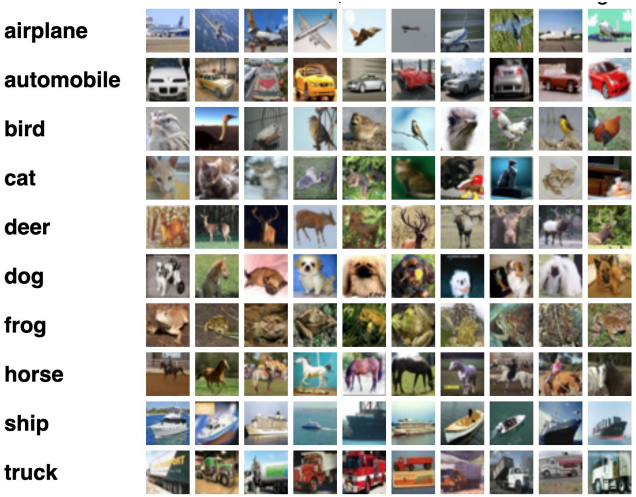
- Translation Invariance & Local features
- Key Concepts in Convolutional Neural Networks
  - Convolutional layers
  - Pooling layers
  - Fully-connected layers (+ flattening)
- Designing CNN models =~ Designing feature map processing
- Convolutional Autoencoder
- A new weapon
  - Batch Normalization
- CNN models
  - LeNet-5, AlexNet
  - VGG-16/19, GoogLeNet, ResNet
- CNN applications

Assignment:

https://colab.research.google.com/github/suhara/cis6930-fall2021/blob/main/notebooks/cis6930_week3b_convolutional_neural_networks_cifar10.ipynb

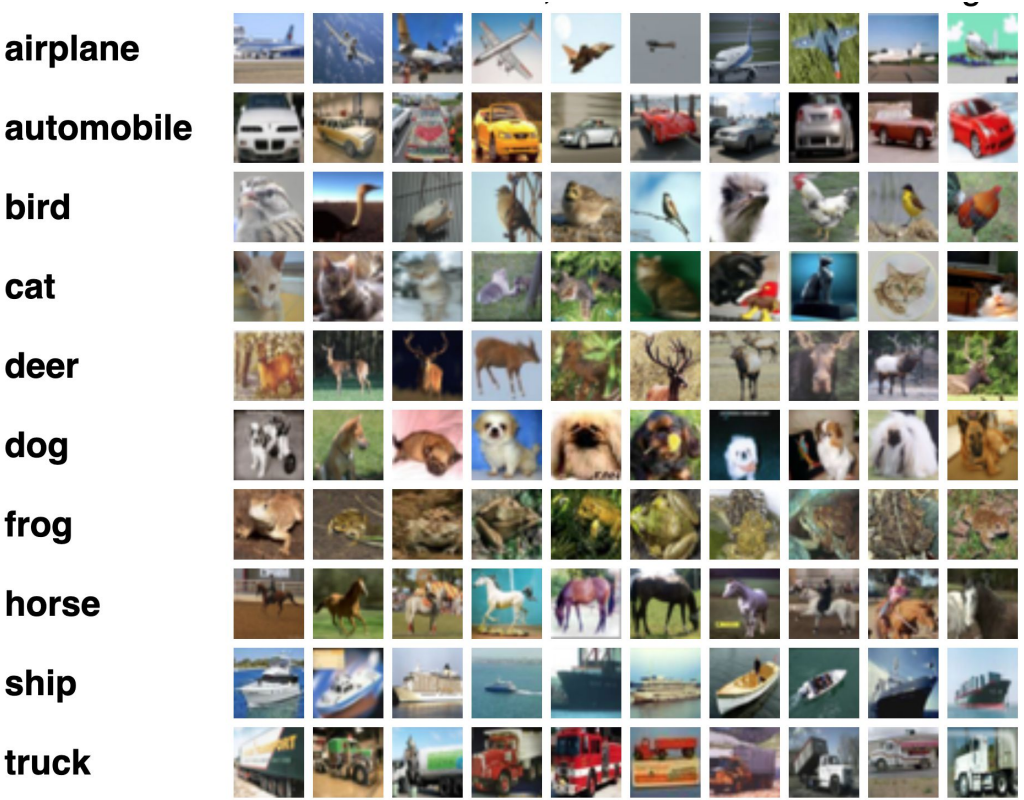
10min

Design/Develop your own CNN: <Your name here>Net … for CIFAR-10!



Coming soon!

CIFAR-10 Dataset

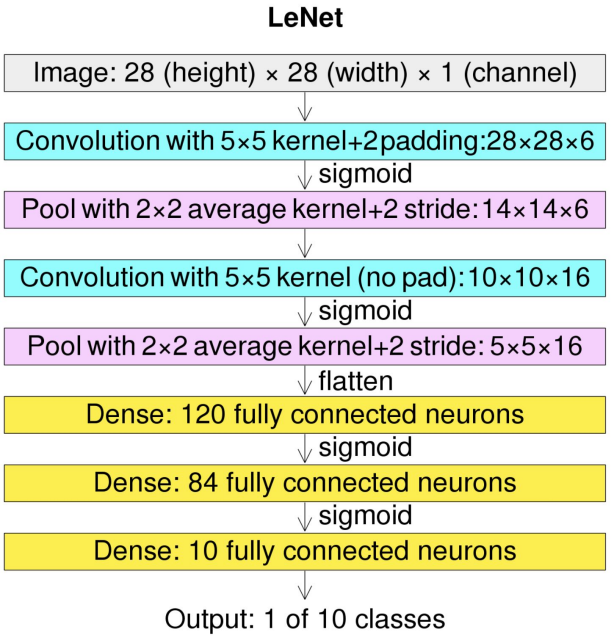


Assignment (due Fri 9/22*) (* No late penalty until Mon 9/25)

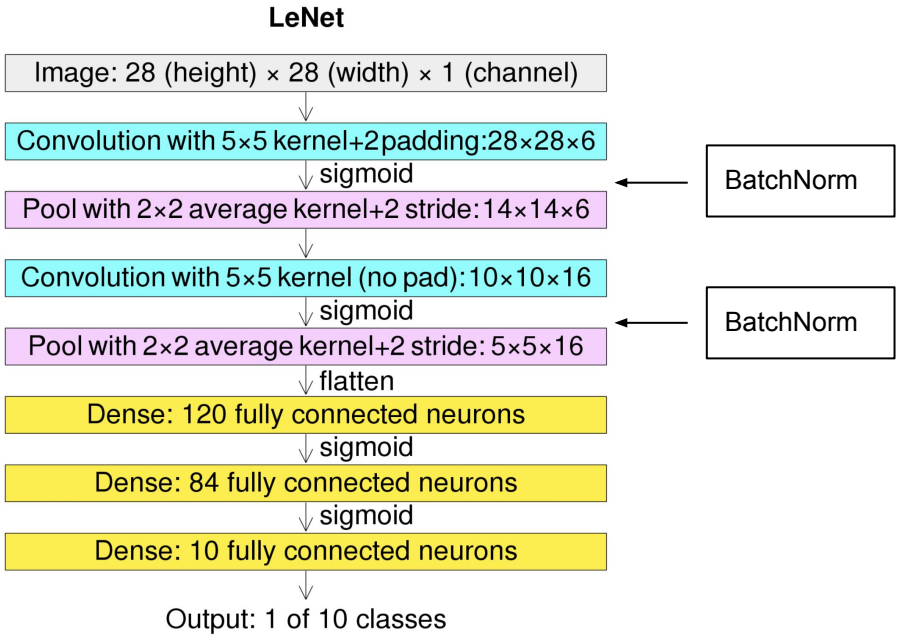
- 1. Design and develop two CNN models using PyTorch
  - **1 baseline model**
    - LeNet-5 might be a good starting point (which does not have to be an existing model)
  - **1 proposed model (Assign a name to the model!)**
    - Try to use/combine techniques that you learned from the course
- 2. Train and evaluate **two models** on the CIFAR-10 dataset
  - Use the same optimization configuration for the two models
  - Report training/validation loss and validation/test accuracy
- 3. Discuss the results
  - Which model performs better?
  - Why do you think the one is better than the other?

- Hint: What are the differences b/w the two models? Can you attribute the performance difference to the architecture difference?
- [Optional] Choose the model that performs the best on the validation data as the final model instead of using the model after training for all epochs.

Hint: Another way to design a baseline
For Comparative Study (aka Ablation Study)



75% Test accuracy



85% Test accuracy

Assignment Checklist

- ☐ Does the code has two CNN model classes?
  - ☐ Does the proposed model have a name? :)
- ☐ Are the two models trained and evaluated? Are both **training/validation loss** and **validation/test accuracy values** reported?
- ☐ Does it have a discussion about why you consider one model performs better than the other