Lecture 13

Introduction to Convolutional Neural Networks

STAT 479: Deep Learning, Spring 2019

Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching/stat479-ss2019/

CNNs for Image Classification

Image Source:
twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

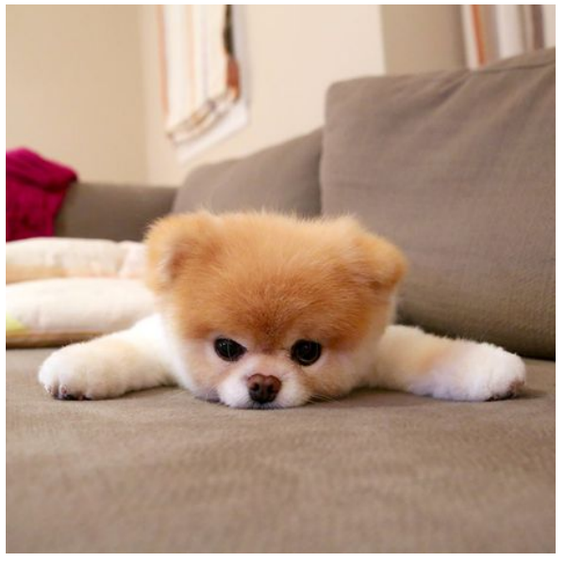
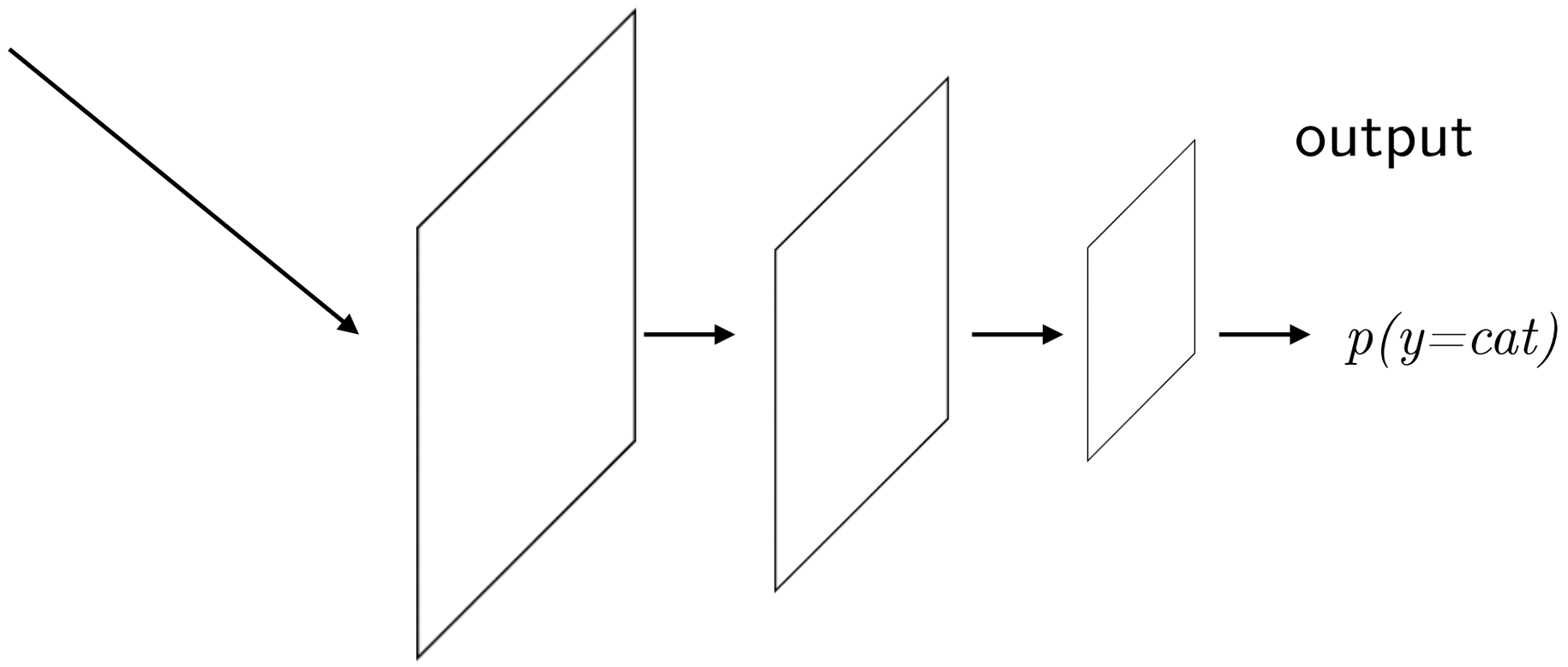


Image Source: https://www.pinterest.com/pin/244742560974520446



Some Applications of CNNs

Predicting people's age

(160,000 images from the RenRen Social Network)

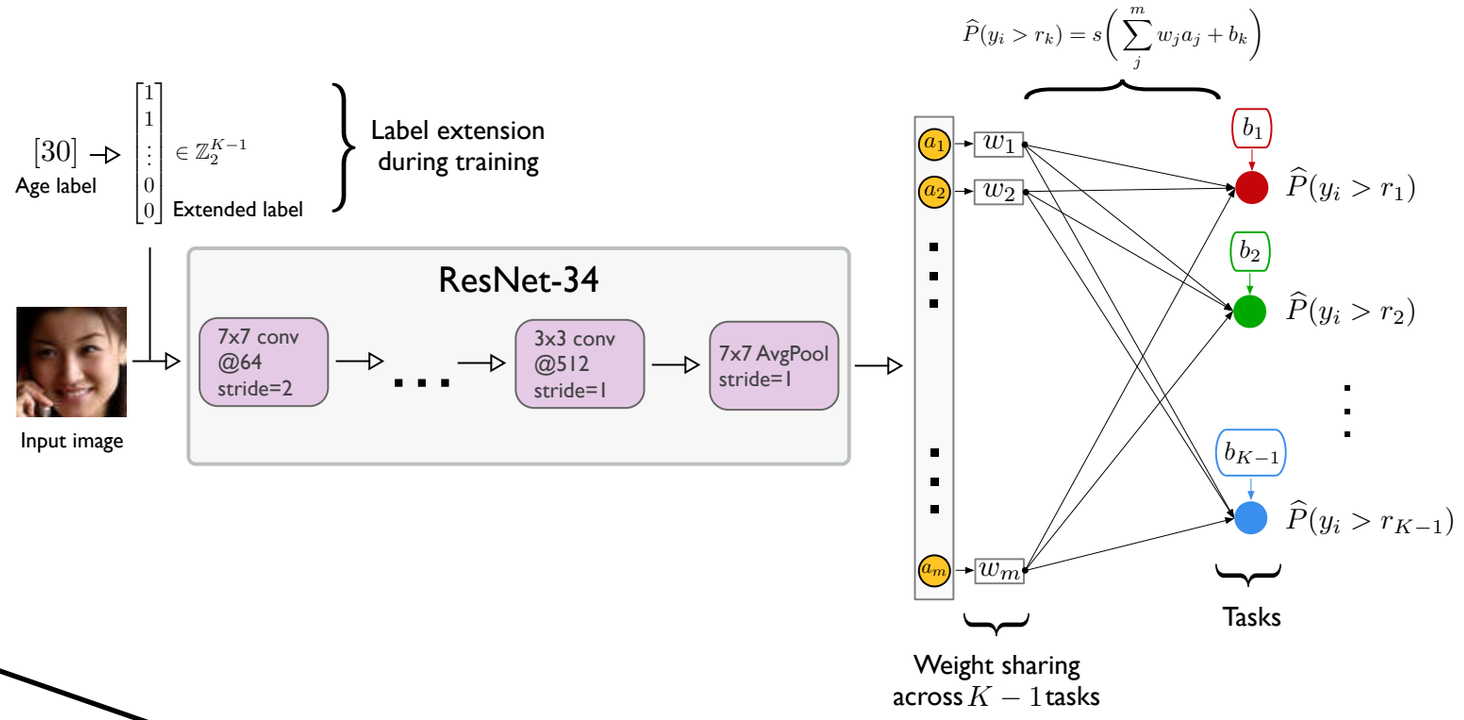


Figure 2. Illustration of the Consistent Rank Logits CNN (CORAL-CNN) used for age prediction. From the estimated probability values, the binary labels are obtained via Eq. (5) and converted to the age label via Eq. (1).

Table 1. Age prediction errors on the test sets *without* task importance weighting.

Method	Random Seed	MORPH-2		AFAD		UTKFace		CACD	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
CE-CNN	0	3.40	4.88	3.98	5.55	6.57	9.16	6.18	8.86
	1	3.39	4.87	4.00	5.57	6.24	8.69	6.10	8.79
	2	3.37	4.87	3.96	5.50	6.29	8.78	6.13	8.87
	AVG ± SD	3.39 ± 0.02	4.89 ± 0.01	3.98 ± 0.02	5.54 ± 0.04	6.37 ± 0.18	8.88 ± 0.25	6.14 ± 0.04	8.84 ± 0.04
OR-CNN (Niu et al., 2016)	0	2.98	4.26	3.66	5.10	5.71	8.11	5.53	7.91
	1	2.98	4.26	3.69	5.13	5.80	8.12	5.53	7.98
	2	2.96	4.20	3.68	5.14	5.71	8.11	5.49	7.89
	AVG ± SD	2.97 ± 0.01	4.24 ± 0.03	3.68 ± 0.02	5.13 ± 0.02	5.74 ± 0.05	8.08 ± 0.06	5.52 ± 0.02	7.93 ± 0.05
CORAL-CNN (ours)	0	2.68	3.75	3.49	4.82	5.46	7.61	5.56	7.80
	1	2.63	3.66	3.46	4.83	5.46	7.63	5.37	7.64
	2	2.61	3.64	3.52	4.91	5.48	7.63	5.25	7.53
	AVG ± SD	**2.64 ± 0.04**	**3.68 ± 0.06**	**3.49 ± 0.03**	**4.85 ± 0.05**	**5.47 ± 0.01**	**7.62 ± 0.01**	**5.39 ± 0.16**	**7.66 ± 0.14**

Cao, Wenzhi, Vahid Mirjalili, and Sebastian Raschka. "Consistent Rank Logits for Ordinal Regression with Convolutional Neural Networks." *arXiv preprint arXiv:1901.07884* (2019).

Some Applications of CNNs

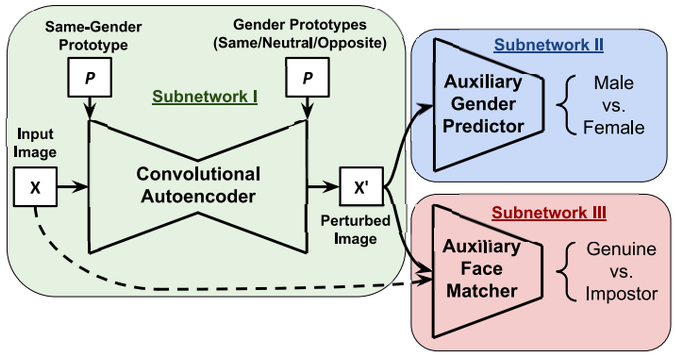
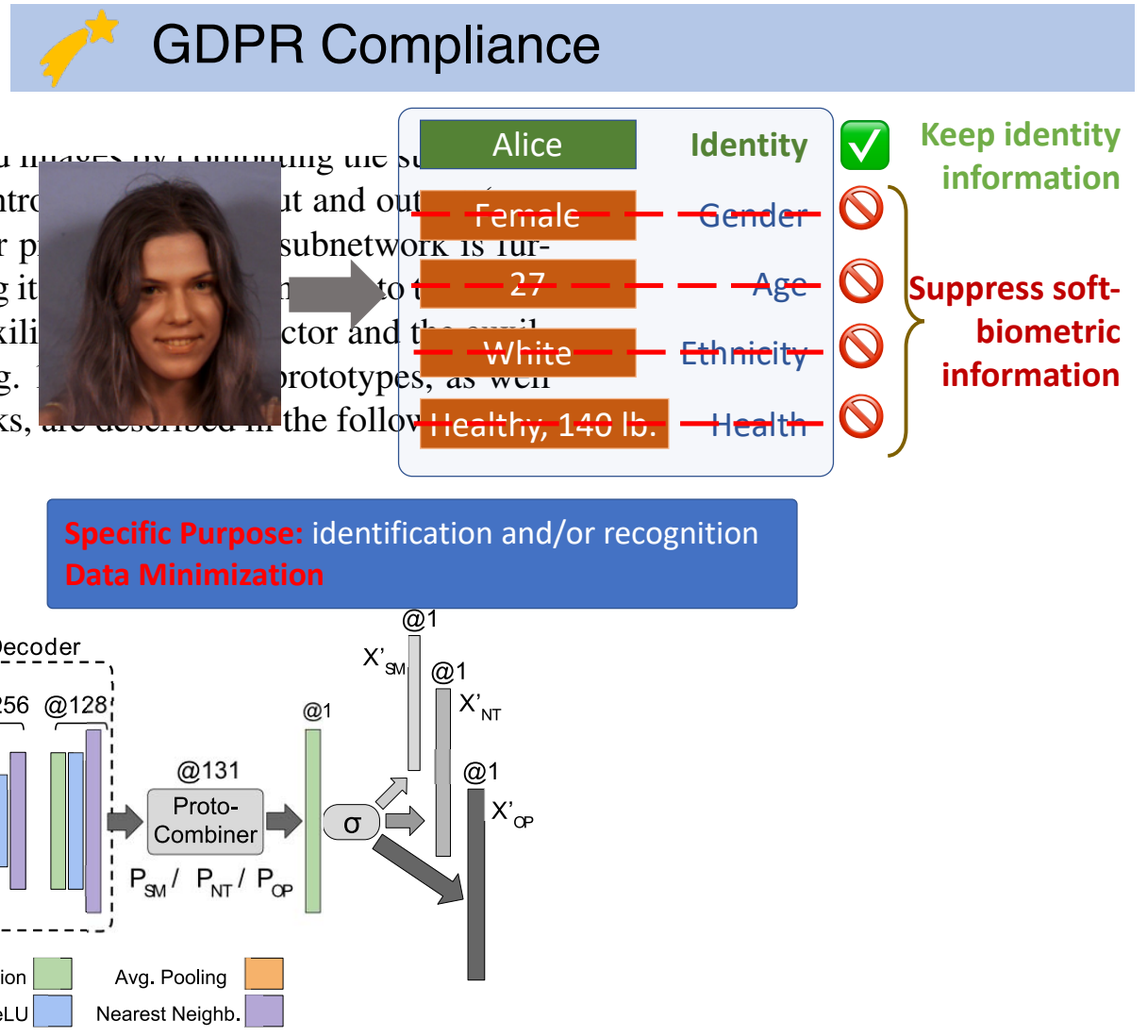


Figure 1. Schematic representation of the semi-adversarial neural network architecture designed to derive perturbations that are able to confound gender classifiers while still allowing biometric matchers to perform well. The overall network consists of three sub-components: a convolutional autoencoder (subnetwork I), an auxiliary gender classifier (subnetwork II), and an auxiliary matcher (subnetwork III).

2.2.1 Convolutional autoencoder

The architecture of the convolutional autoencoder sub-



Vahid Mirjalili, Sebastian Raschka, Anoop Namboodiri, and Arun Ross (2018) *Semi-adversarial networks: Convolutional autoencoders for imparting privacy to face images.* Proc. of 11th IAPR International Conference on Biometrics (ICB 2018), Gold Coast, Australia.

Vahid Mirjalili, Sebastian Raschka, and Arun Ross (2018) Gender Privacy: An Ensemble of Semi Adversarial Networks for Confounding Arbitrary Gender Classifiers. 9th IEEE International Conference on Biometrics: Theory, Applications, and Systems

Some Applications of CNNs

Dermatologist-level classification of skin cancer

An artificial intelligence trained to classify images of skin lesions as benign lesions or malignant skin cancers achieves the accuracy of board-certified dermatologists.

In this work, we pretrain a deep neural network at general object recognition, then fine-tune it on a dataset of ~130,000 skin lesion images comprised of over 2000 diseases.

FULL NATURE ARTICLE >

OPEN-ACCESS PDF >

Esteva, Andre, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. "Dermatologist-level classification of skin cancer with deep neural networks." *Nature* 542, no. 7639 (2017): 115.

Why Image Classification is Hard

Different lighting, contrast, viewpoints, etc.

Image Source:
twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

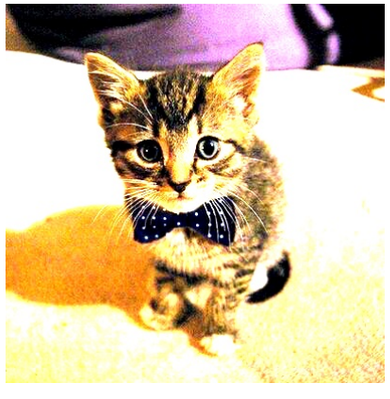
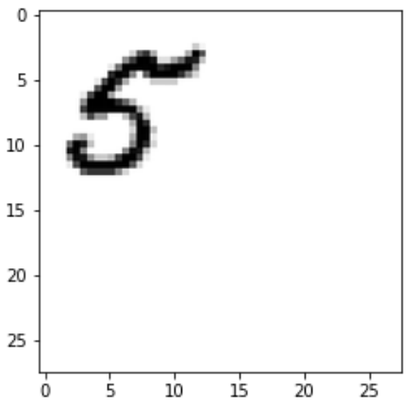
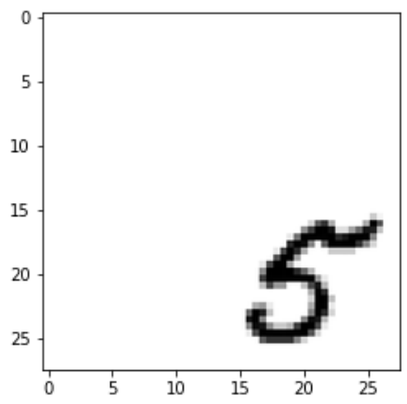


Image Source: https://www.123rf.com/photo_76714328_side-view-of-tabby-cat-face-over-white.html

Or even simple translation

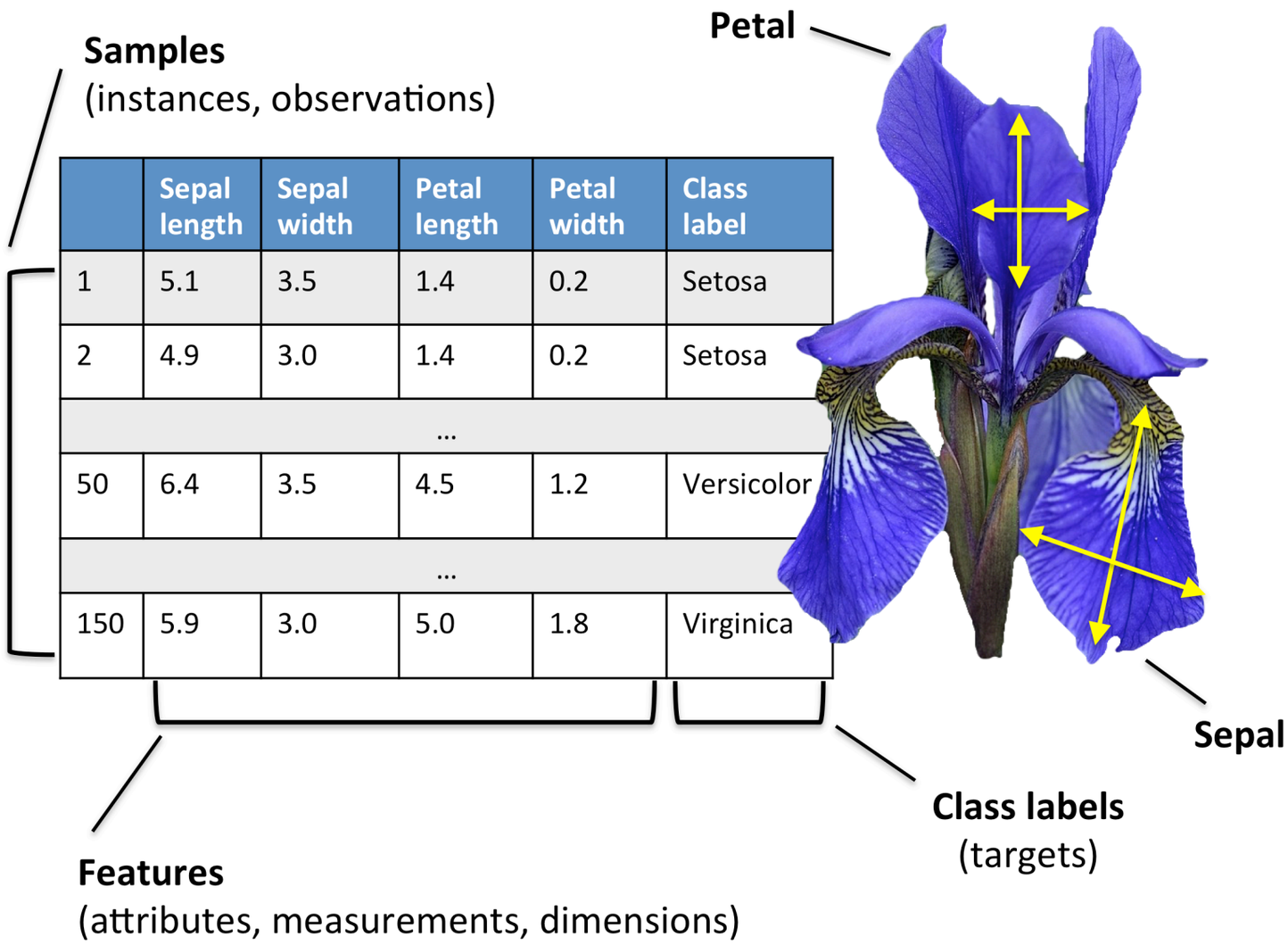




This is hard for traditional methods like multi-layer perceptrons, because the prediction is basically based on a sum of pixel intensities

Traditional Approaches

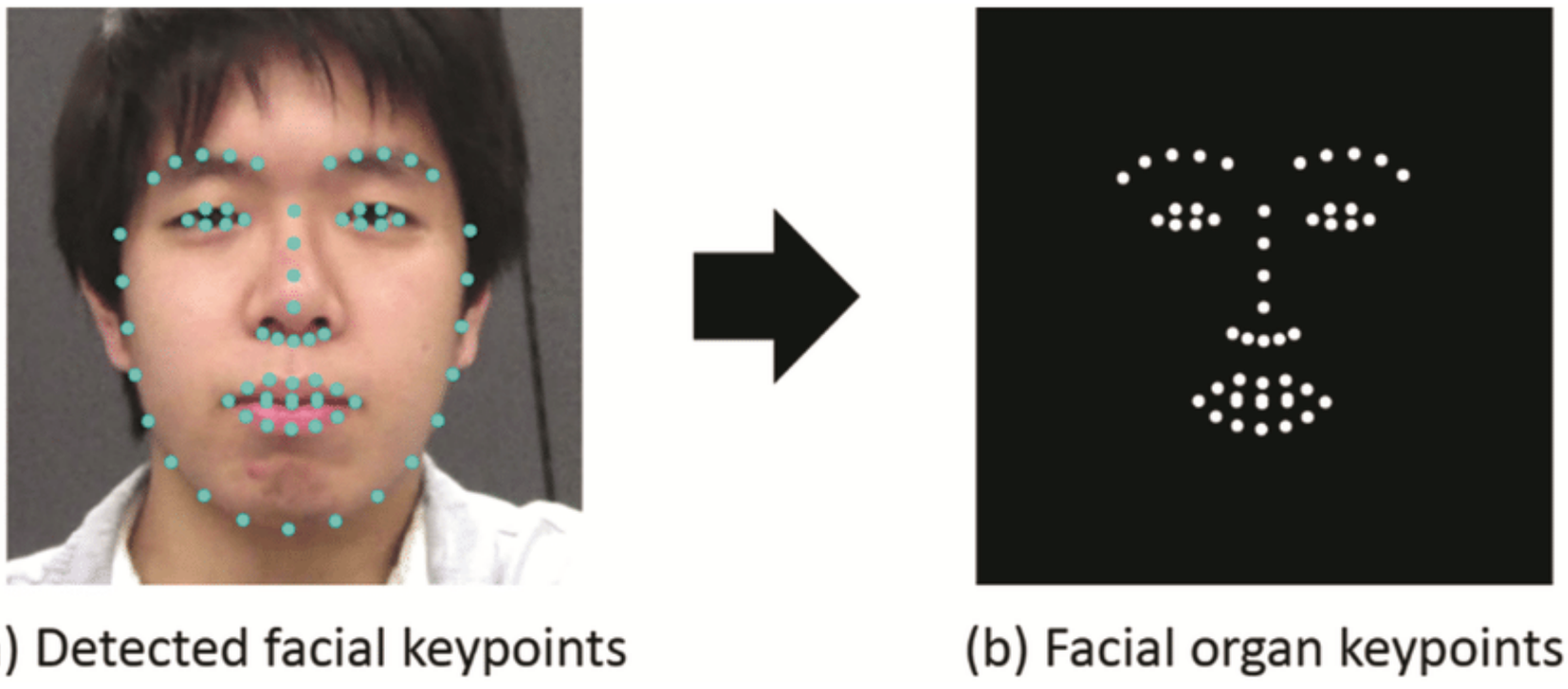
a) Use hand-engineered features



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Traditional Approaches

a) Use hand-engineered features



(a) Detected facial keypoints (b) Facial organ keypoints

Sasaki, K., Hashimoto, M., & Nagata, N. (2016). Person Invariant Classification of Subtle Facial Expressions Using Coded Movement Direction of Keypoints. In *Video Analytics. Face and Facial Expression Recognition and Audience Measurement* (pp. 61-72). Springer, Cham.

Traditional Approaches

b) Preprocess images (centering, cropping, etc.)

Image Source: https://www.tokkoro.com/2827328-cat-animals-nature-feline-park-green-trees-grass.html

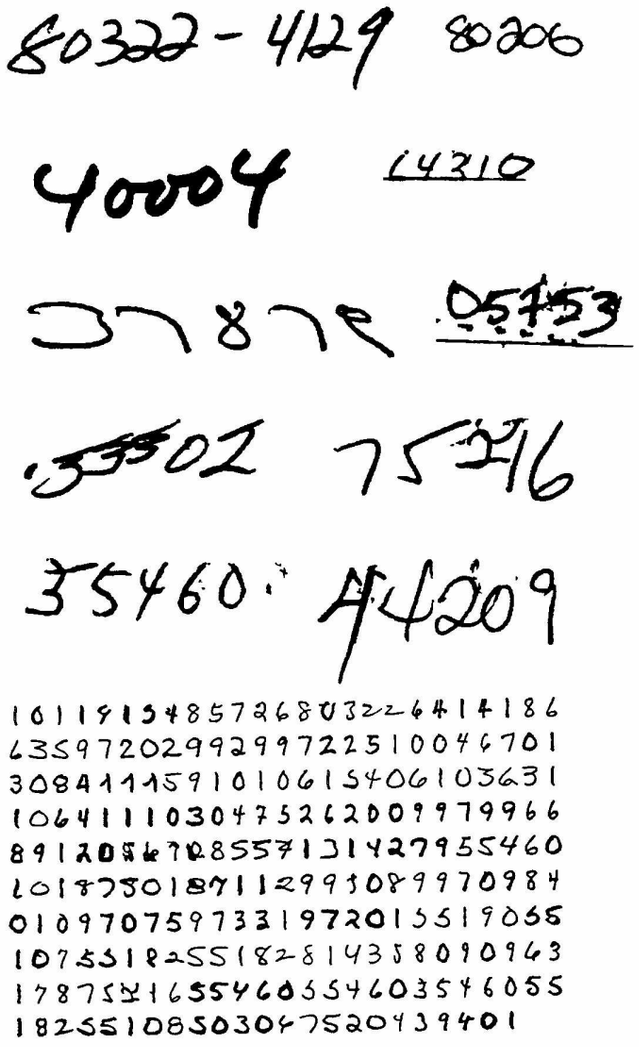
Main Concepts Behind Convolutional Neural Networks

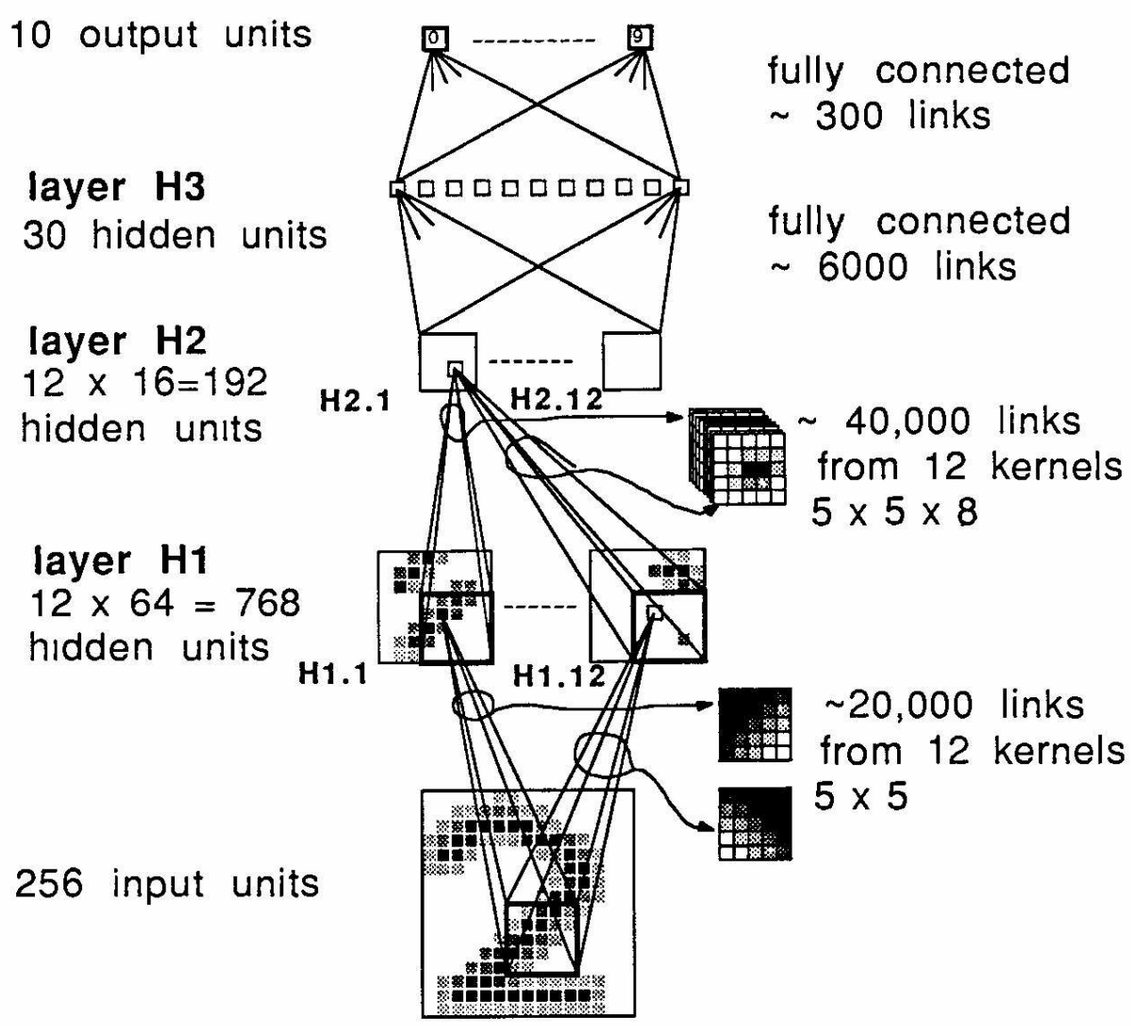
- **Sparse-connectivity**: A single element in the feature map is connected to only a small patch of pixels. (This is very different from connecting to the whole input image, in the case of multi-layer perceptrons.)

- **Parameter-sharing**: The same weights are used for different patches of the input image.

Image Source: https://www.tokkoro.com/2827328-cat-animals-nature-feline-park-green-trees-grass.html

Convolutional Neural Networks





Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

Convolutional Neural Networks

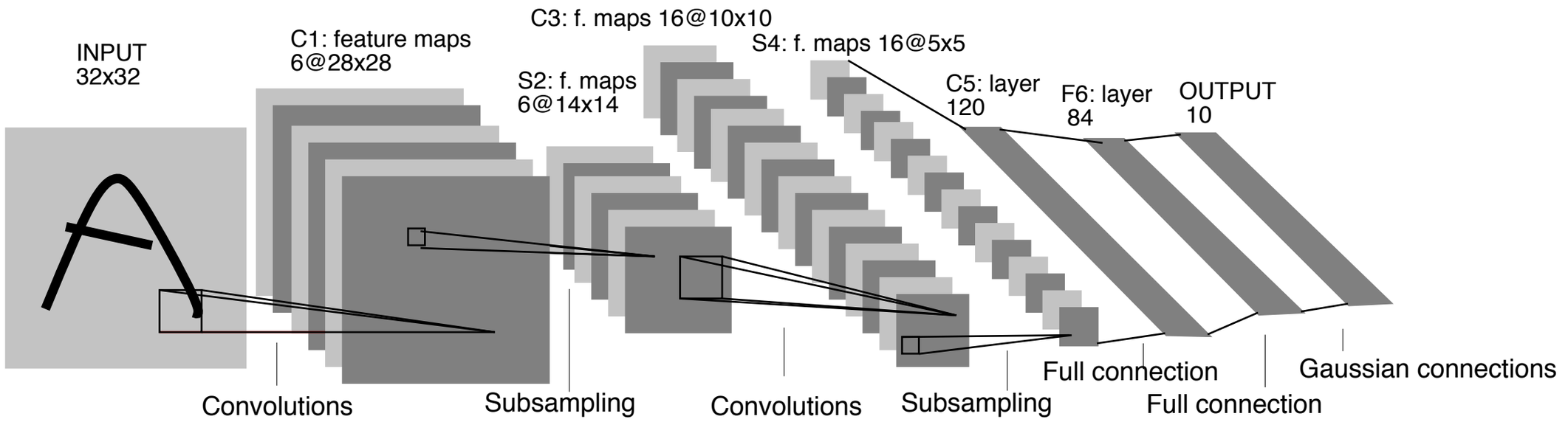


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):2278–2324, 1998.

Convolutional Neural Networks

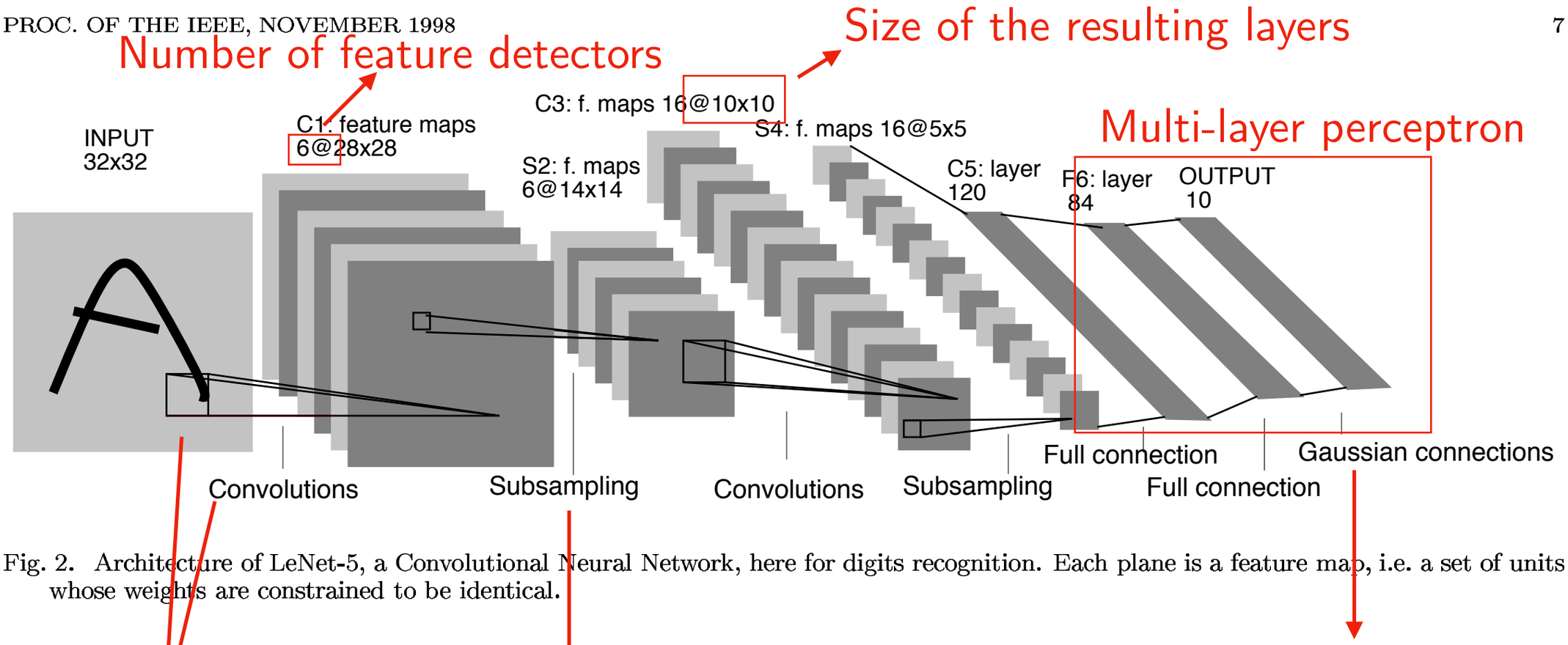


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

nowadays called "pooling"

basically a fully-connected layer + MSE loss
(nowadays better to use
fc-layer + softmax
+ cross entropy

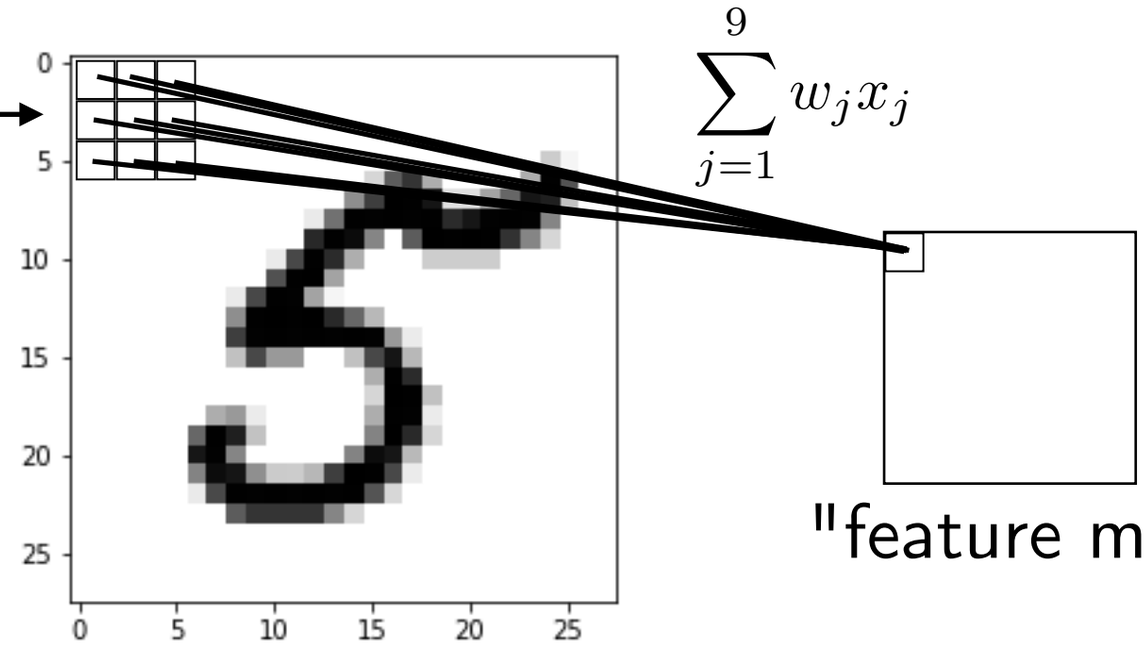
"Feature detectors" (weight matrices)
that are being reused ("weight sharing")
=> also called "kernel" or "filter"

Yann LeCun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):2278–2324, 1998.

Weight Sharing

A "feature detector" (kernel) slides over the inputs to generate a feature map

The pixels are referred to as "receptive field"

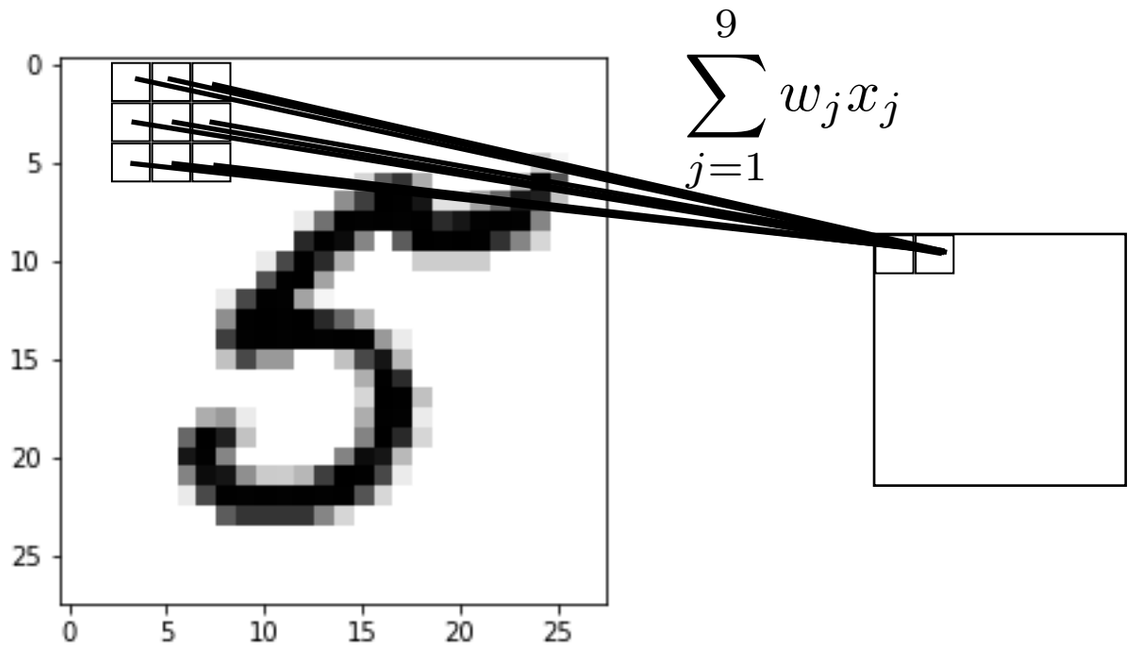


Rationale: A feature detector that works well in one region may also work well in another region

Plus, it is a nice reduction in parameters to fit

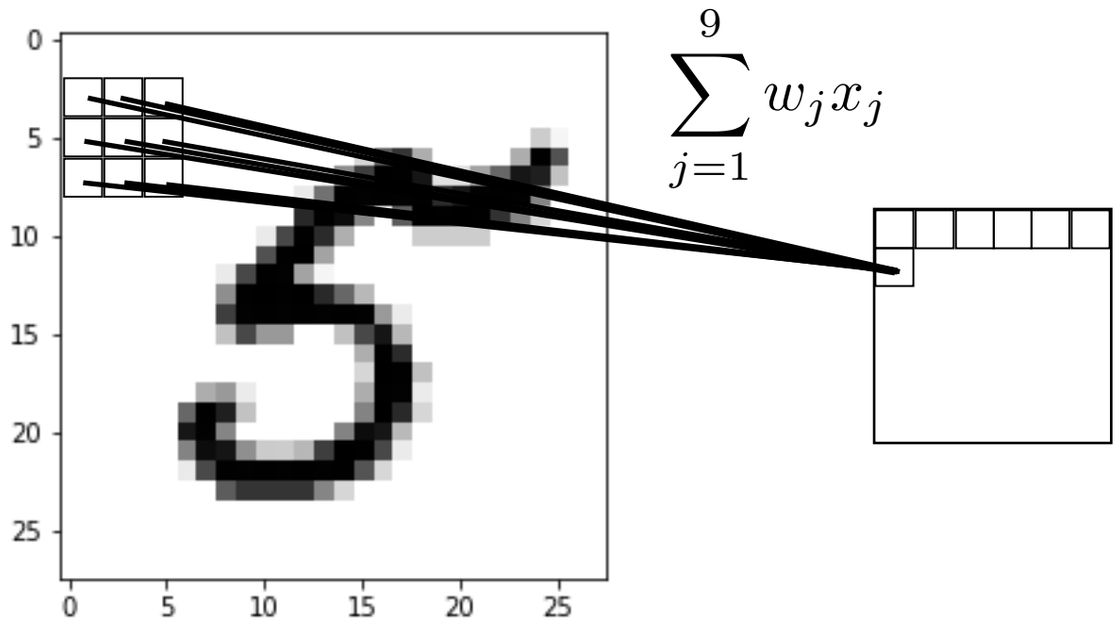
Weight Sharing

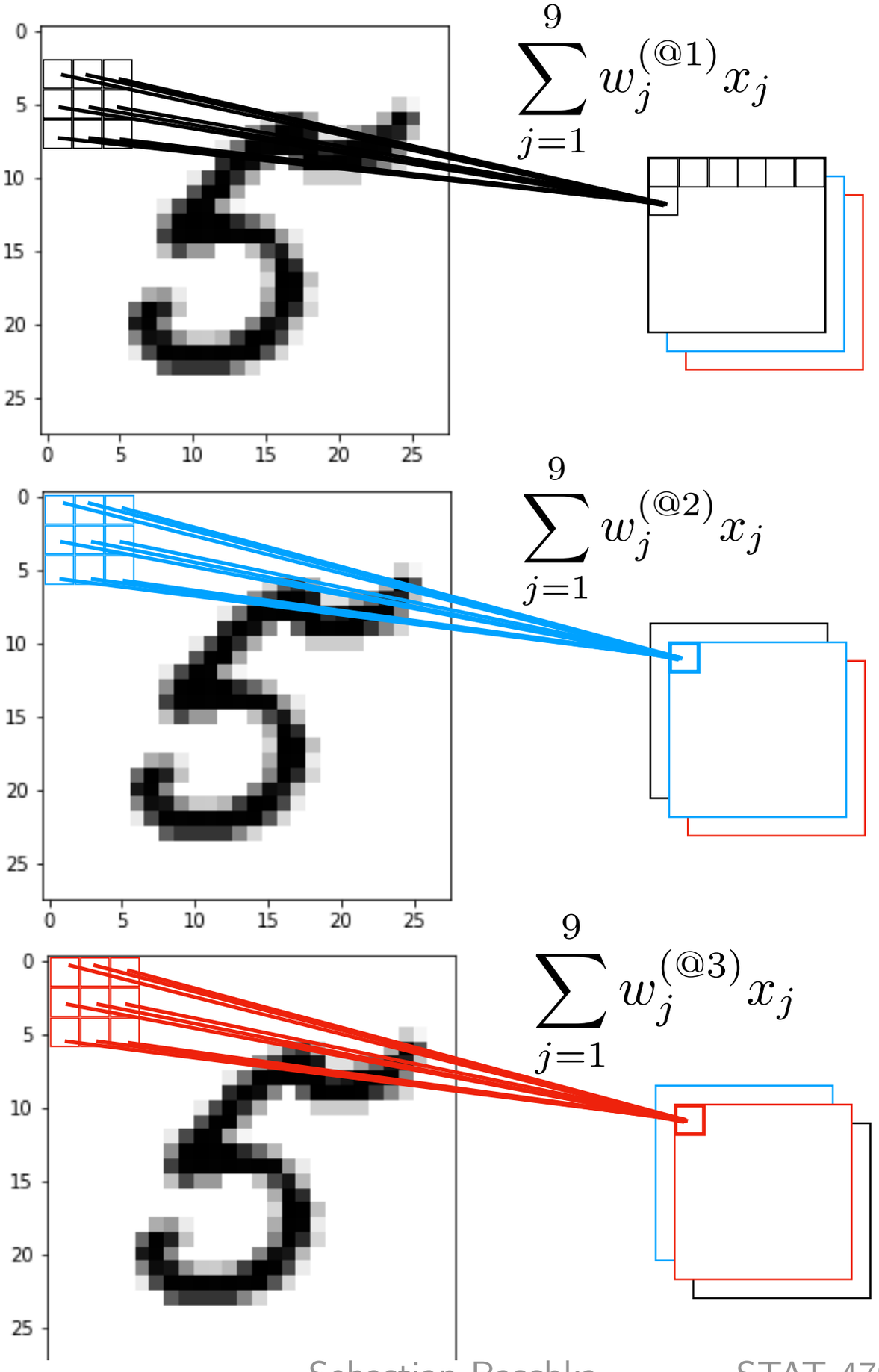
A "feature detector" (kernel) slides over the inputs to generate a feature map



Weight Sharing

A "feature detector" (kernel) slides over the inputs to generate a feature map





Multiple "feature detectors" (kernels) are used to create multiple feature maps

Hidden Layers

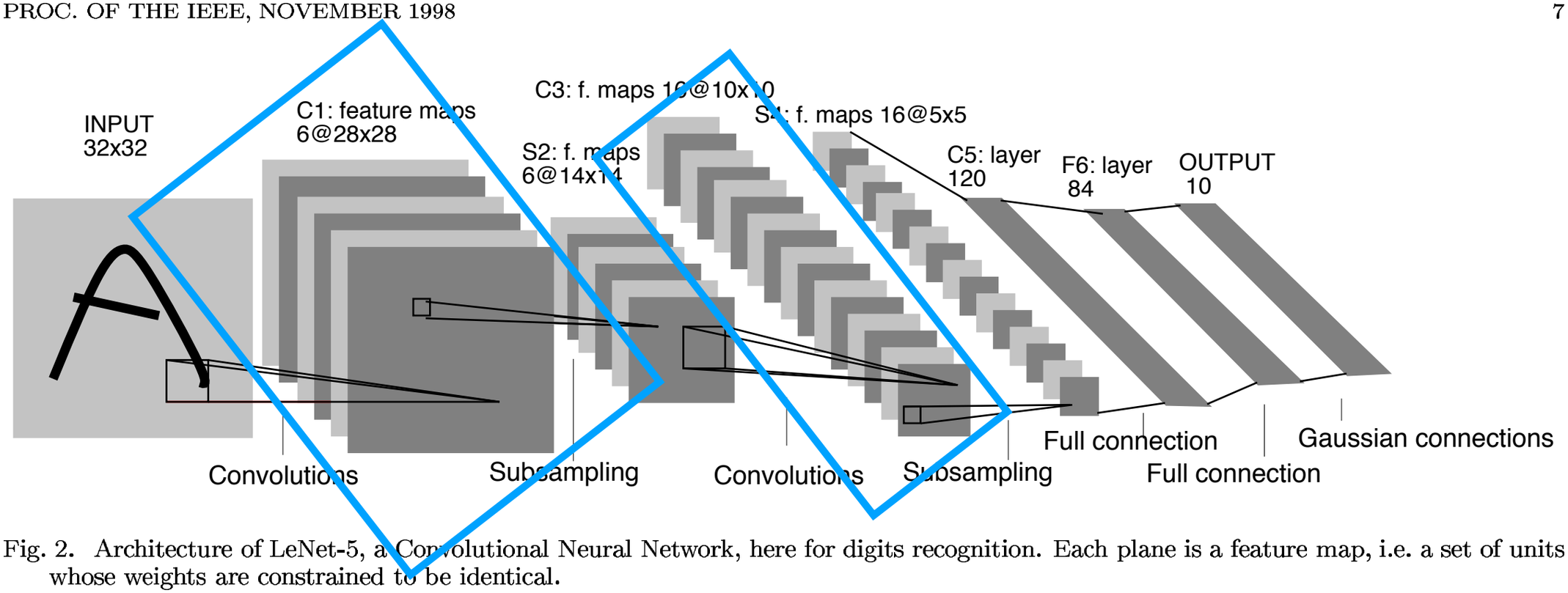
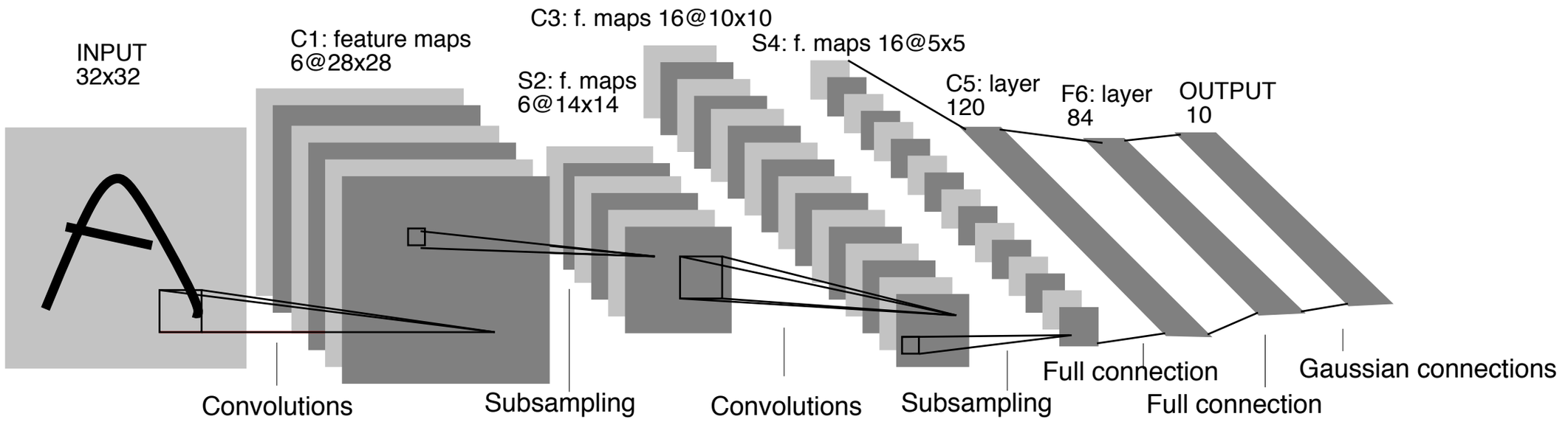


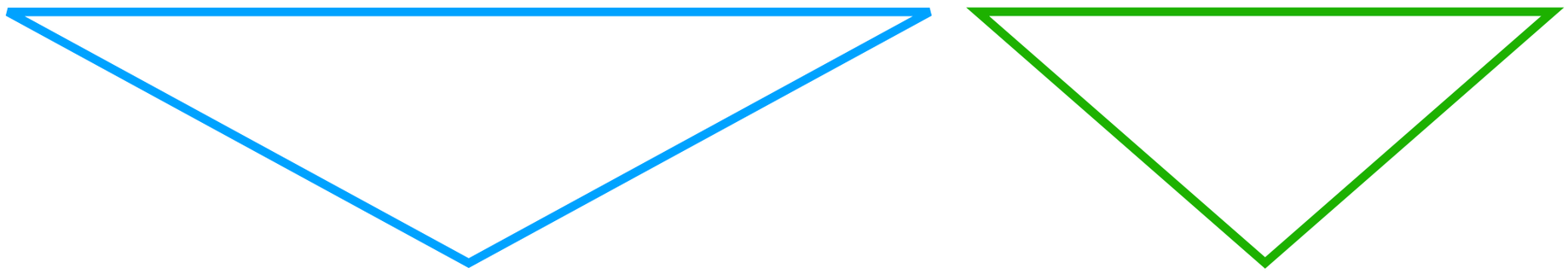
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Each "bunch" of feature maps represents one hidden layer in the neural network.

Counting the FC layers, this network has 5 layers

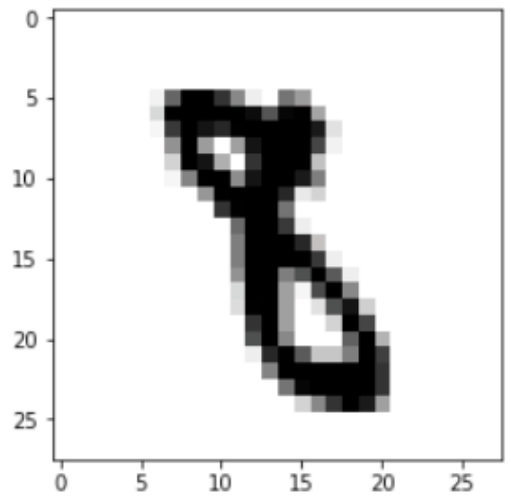
Hidden Layers





"Automatic feature extractor" "Regular classifier"

Kernel Dimensions



```
a.shape
```

```
(1, 28, 28)
```

```
import torch
```

```
conv = torch.nn.Conv2d(in_channels=1,
                       out_channels=8,
                       kernel_size=(5, 5),
                       stride=(1, 1))
```

```
conv.weight.size()
```

```
torch.Size([8, 1, 5, 5])
```

```
conv.bias.size()
```

```
torch.Size([8])
```

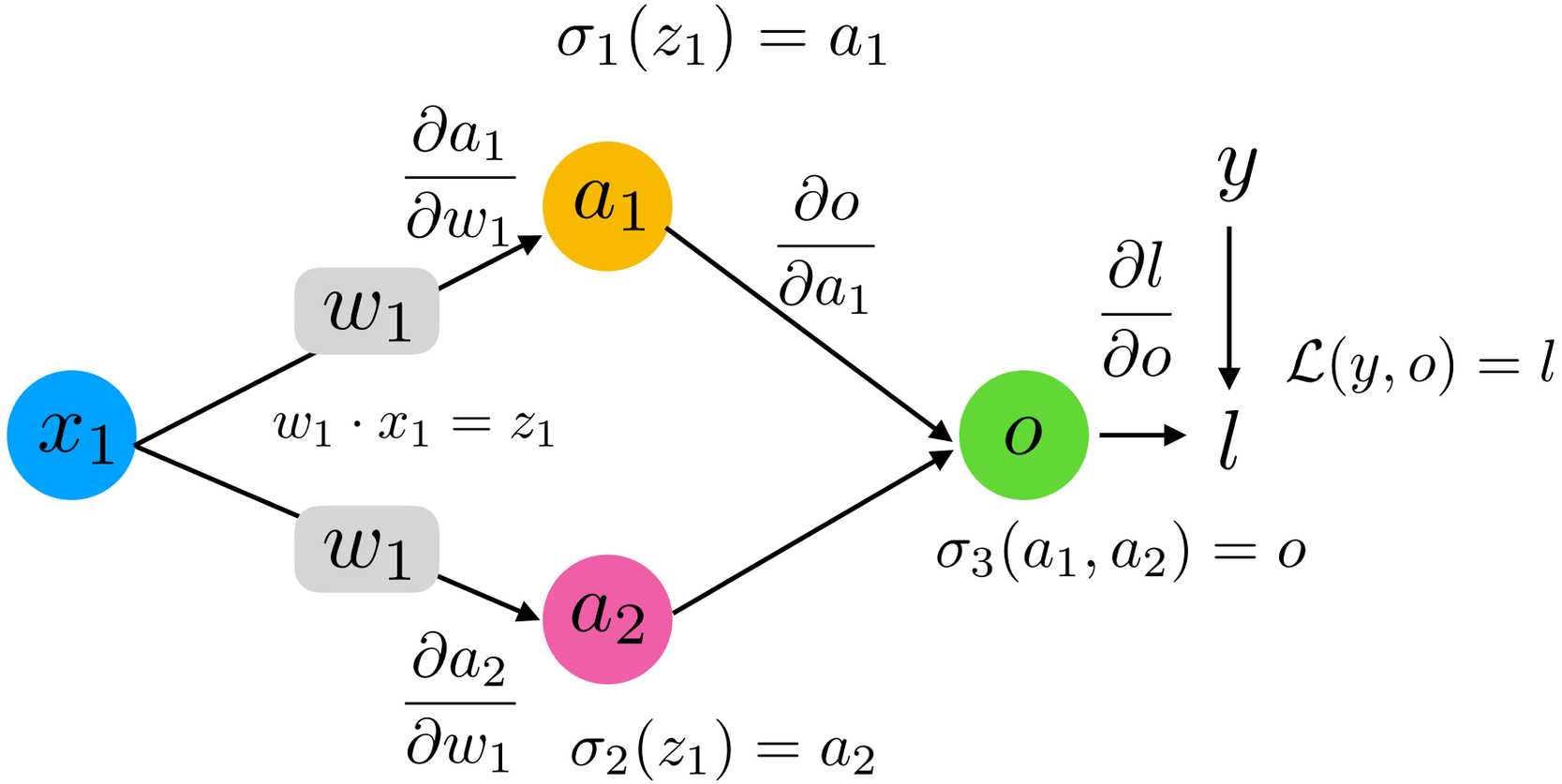
For a grayscale image with a 5x5 feature detector (kernel), we have the following dimensions (number of parameters to learn)

What do you think is the output size for this 28x28 image?

Backpropagation in CNNs

Same overall concept as before: Multivariable chain rule, but now with an additional weight sharing constraint

Remember Lecture 6? Graph with Weight Sharing



Upper path

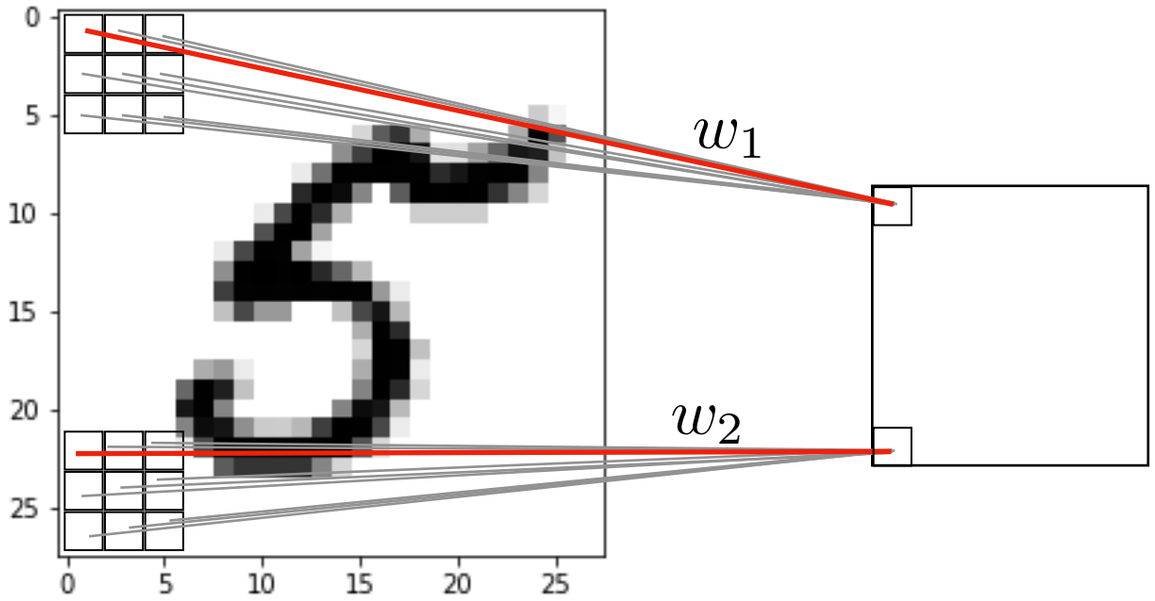
$$\frac{\partial l}{\partial w_1} = \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_1} \cdot \frac{\partial a_1}{\partial w_1} + \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_2} \cdot \frac{\partial a_2}{\partial w_1}$$ (multivariable chain rule)

Lower path

Backpropagation in CNNs

Same overall concept as before: Multivariable chain rule, but now with an additional weight sharing constraint

Due to weight sharing: $w_1 = w_2$



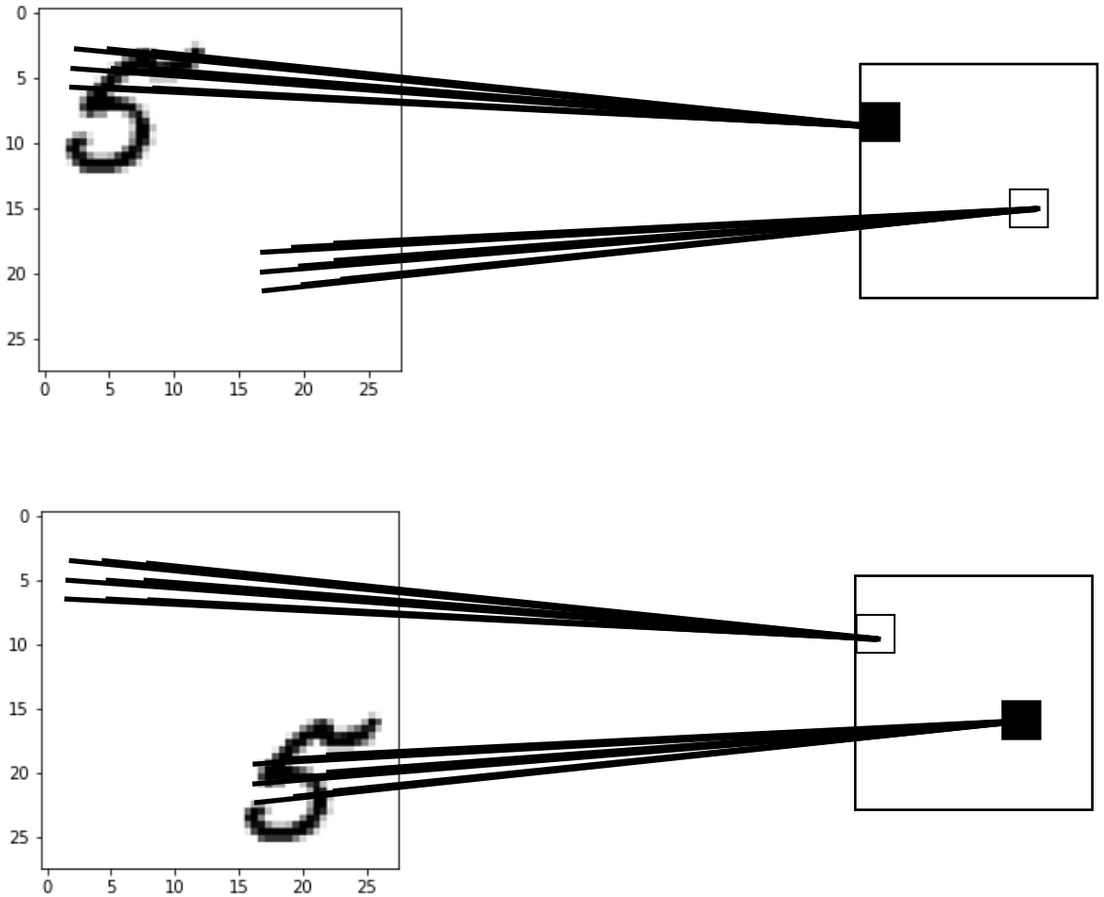
Optional averaging

weight update:

$$w_1 := w_2 := w_1 - \eta \cdot \frac{1}{2}\left(\frac{\partial \mathcal{L}}{\partial w_1} + \frac{\partial \mathcal{L}}{\partial w_2}\right)$$

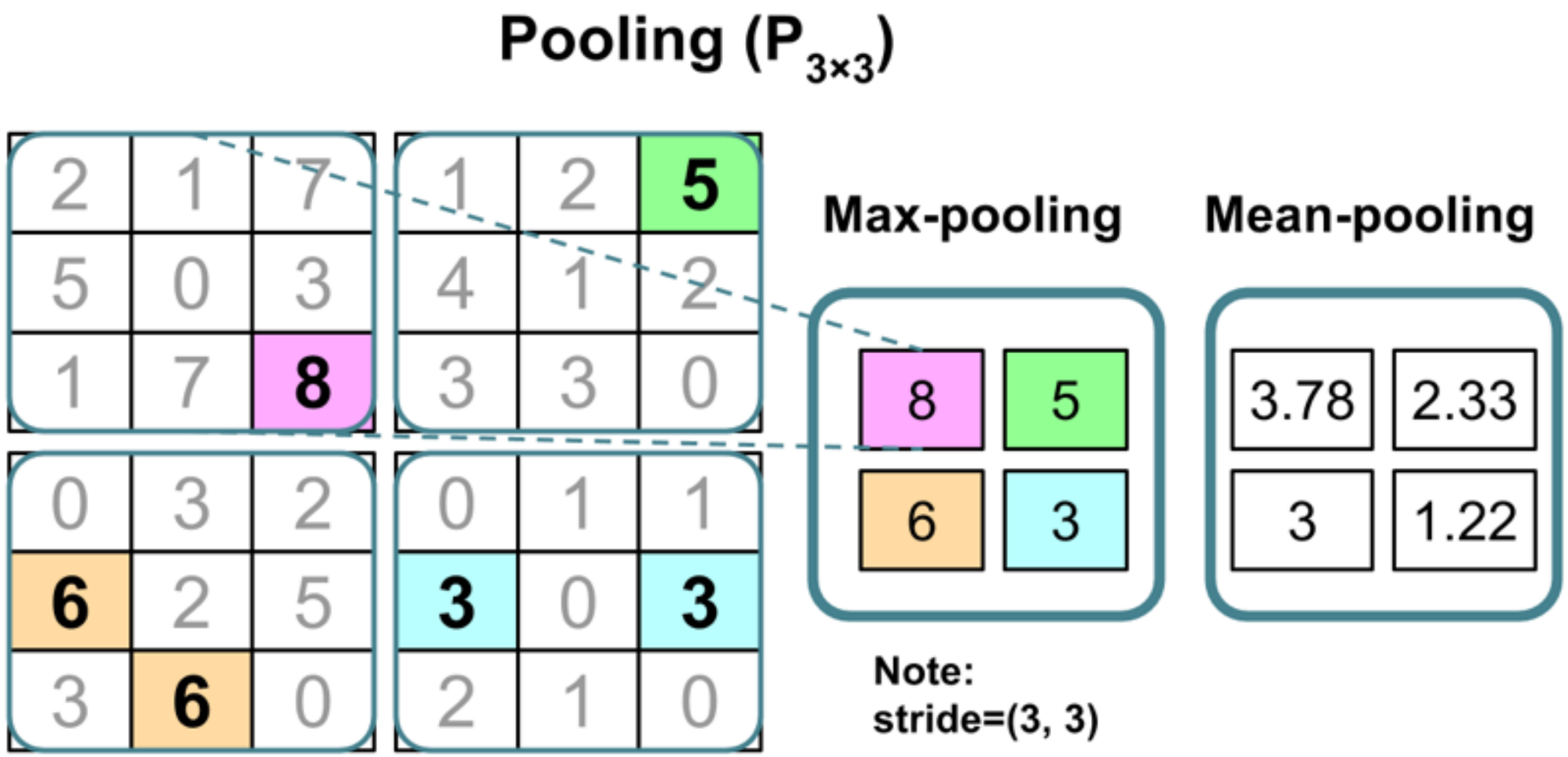
CNNs and Translation/Rotation/Scale Invariance

Note that CNNs are not really invariant to scale, rotation, translation, etc.



The activations are still dependent on the location, etc.

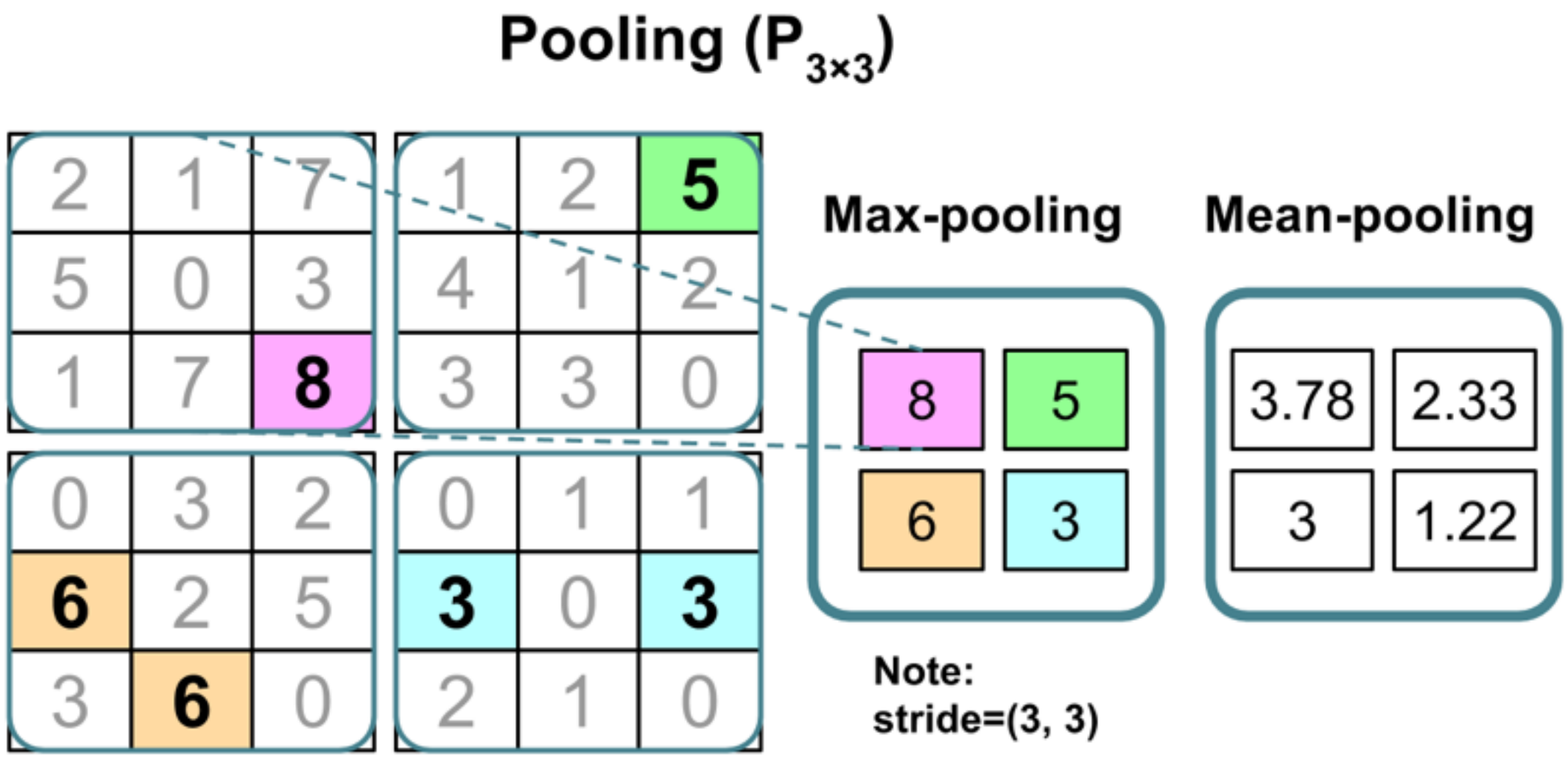
Pooling Layers Can Help With Local Invariance



Downside: Information is lost.
May not matter for classification, but applications where relative position is important (like face recognition)

In practice for CNNs: some image preprocessing still recommended

Pooling Layers Can Help With Local Invariance



Note that typical pooling layers do not have any learnable parameters

Downside: Information is lost.

May not matter for classification, but applications where relative position is important (like face recognition)

In practice for CNNs: some image preprocessing still recommended

Main Breakthrough for CNNs: AlexNet & ImageNet

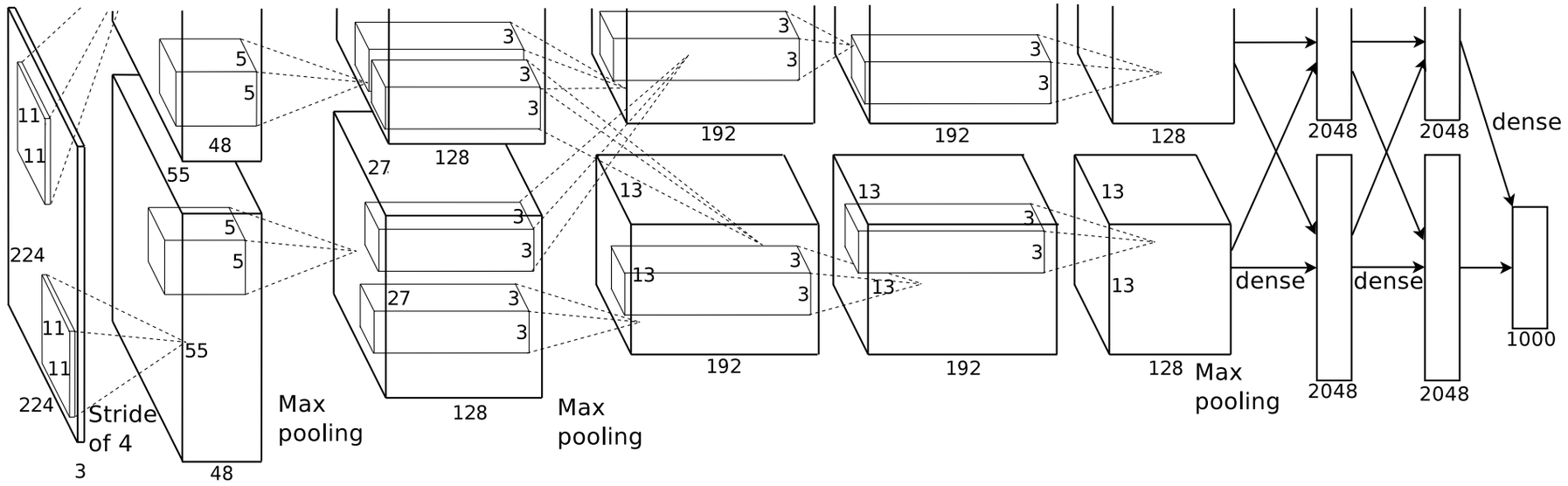


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Main Breakthrough for CNNs: AlexNet & ImageNet

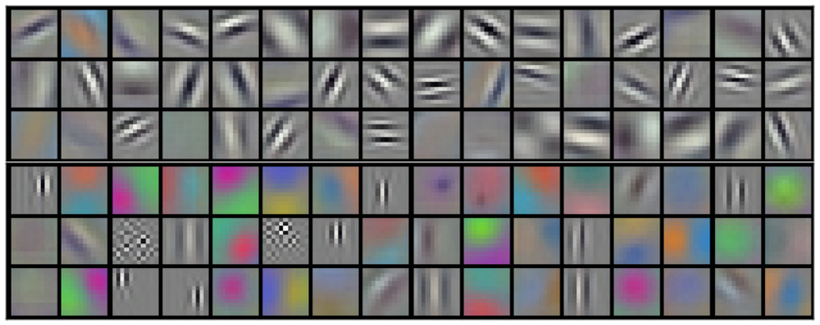


Figure 3: 96 convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2. See Section 6.1 for details.

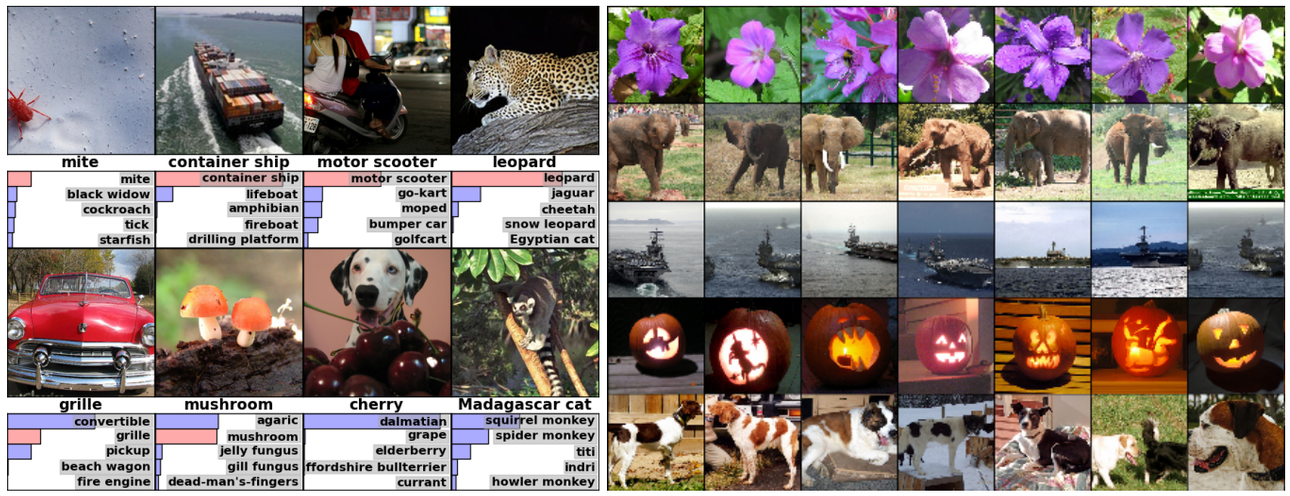
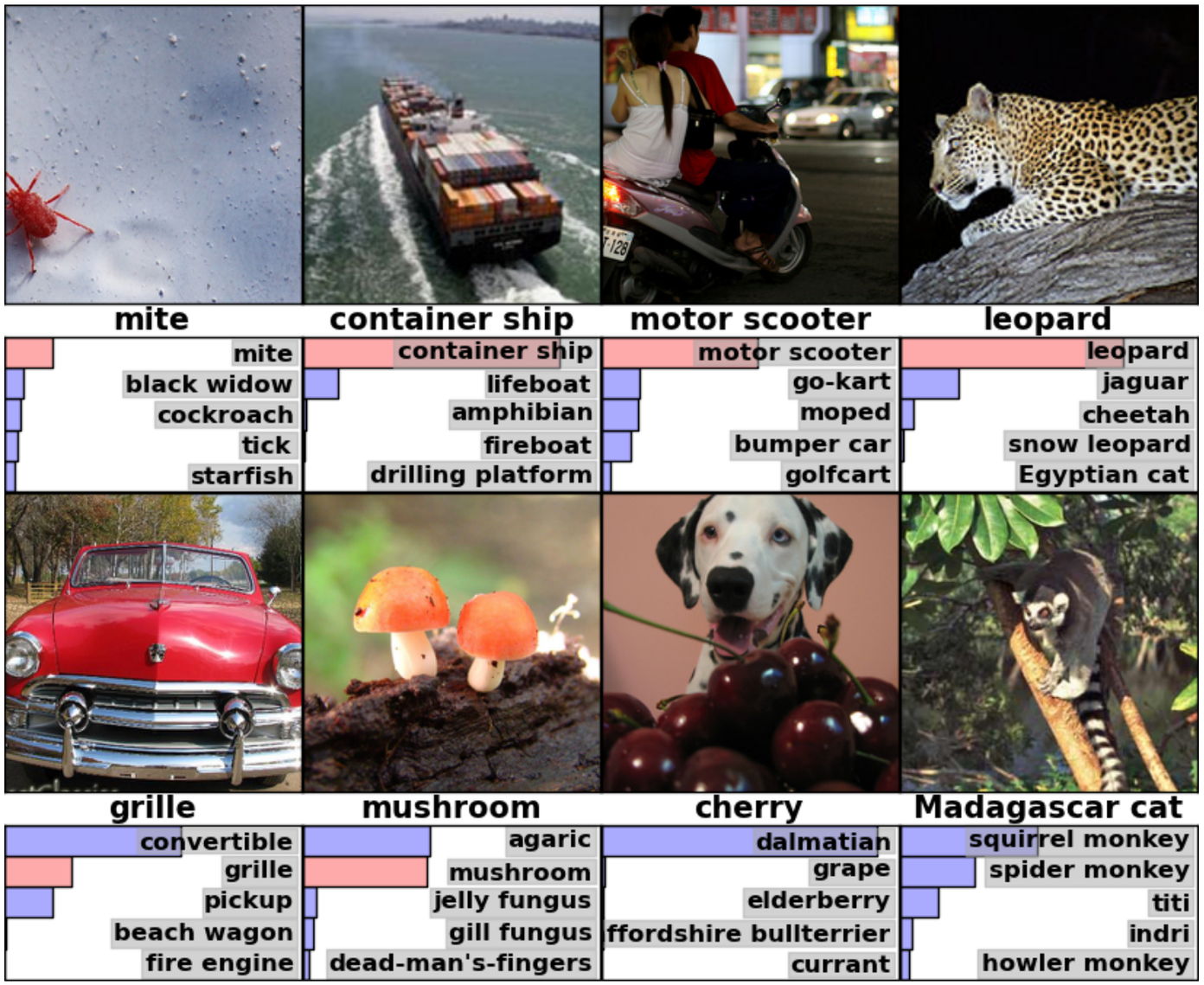


Figure 4: **(Left)** Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). **(Right)** Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Main Breakthrough for CNNs: AlexNet & ImageNet

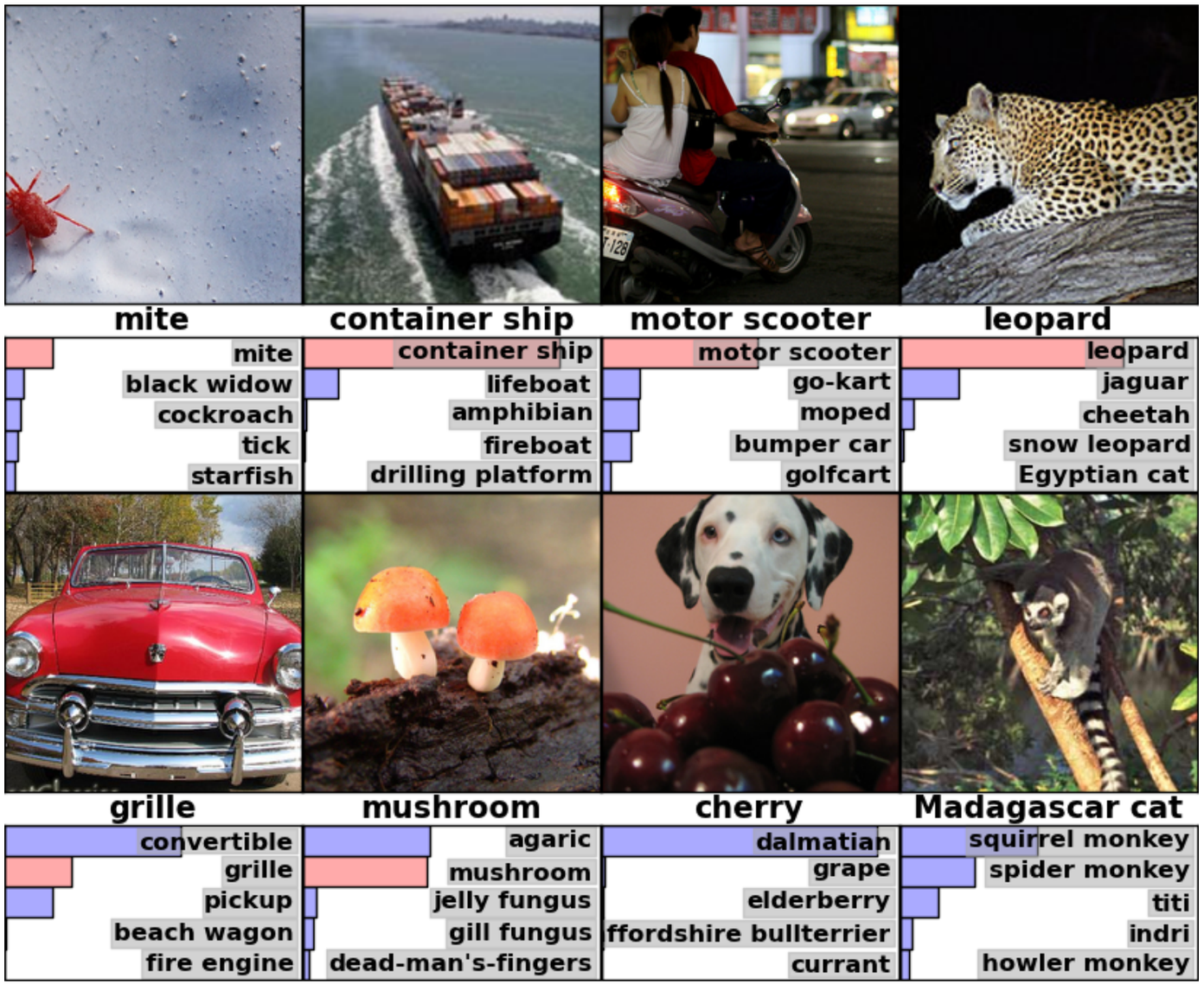


The ImageNet set that was used has ~1.2 million images and 1000 classes

Accuracy is measured as top-5 performance:
Correct prediction if the true label matches one of the top 5 predictions of the model

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Main Breakthrough for CNNs: AlexNet & ImageNet



Note that the actual network inputs were still 224x224 images (random crops from downsampled 256x256 images)

224x224 is still a good/ reasonable size today (224*224*3 = 150,528 features)

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Common CNN Architectures

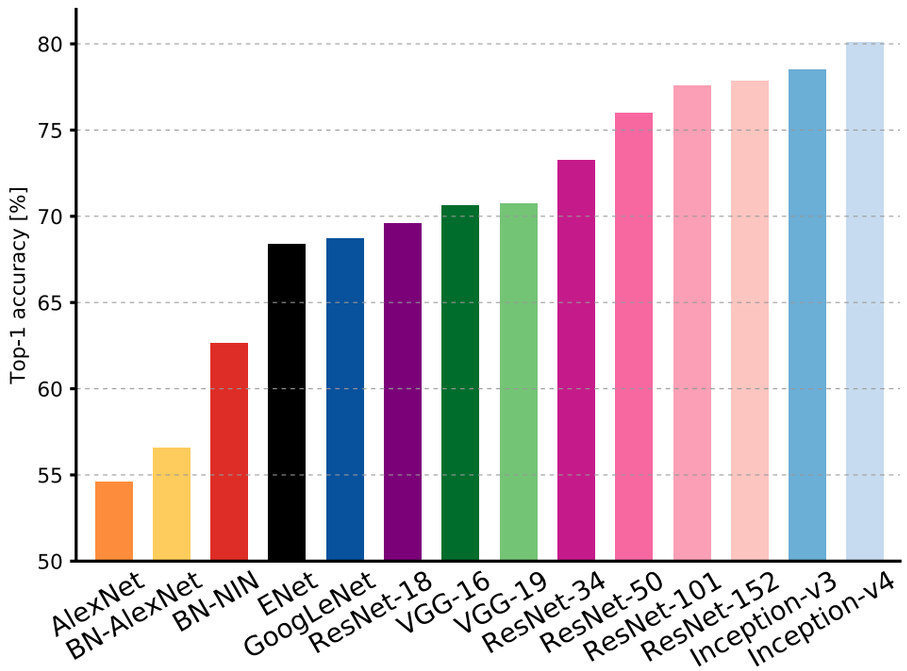


Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that networks of the same group share the same hue, for example ResNet are all variations of pink.

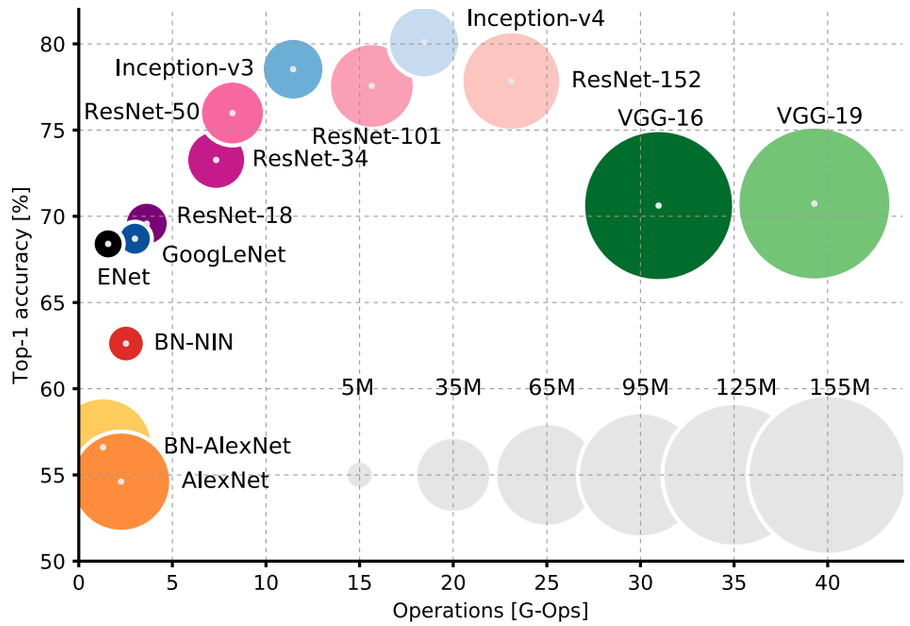


Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params. Both these figures share the same y-axis, and the grey dots highlight the centre of the blobs.

Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*.

Convolutions with Color Channels

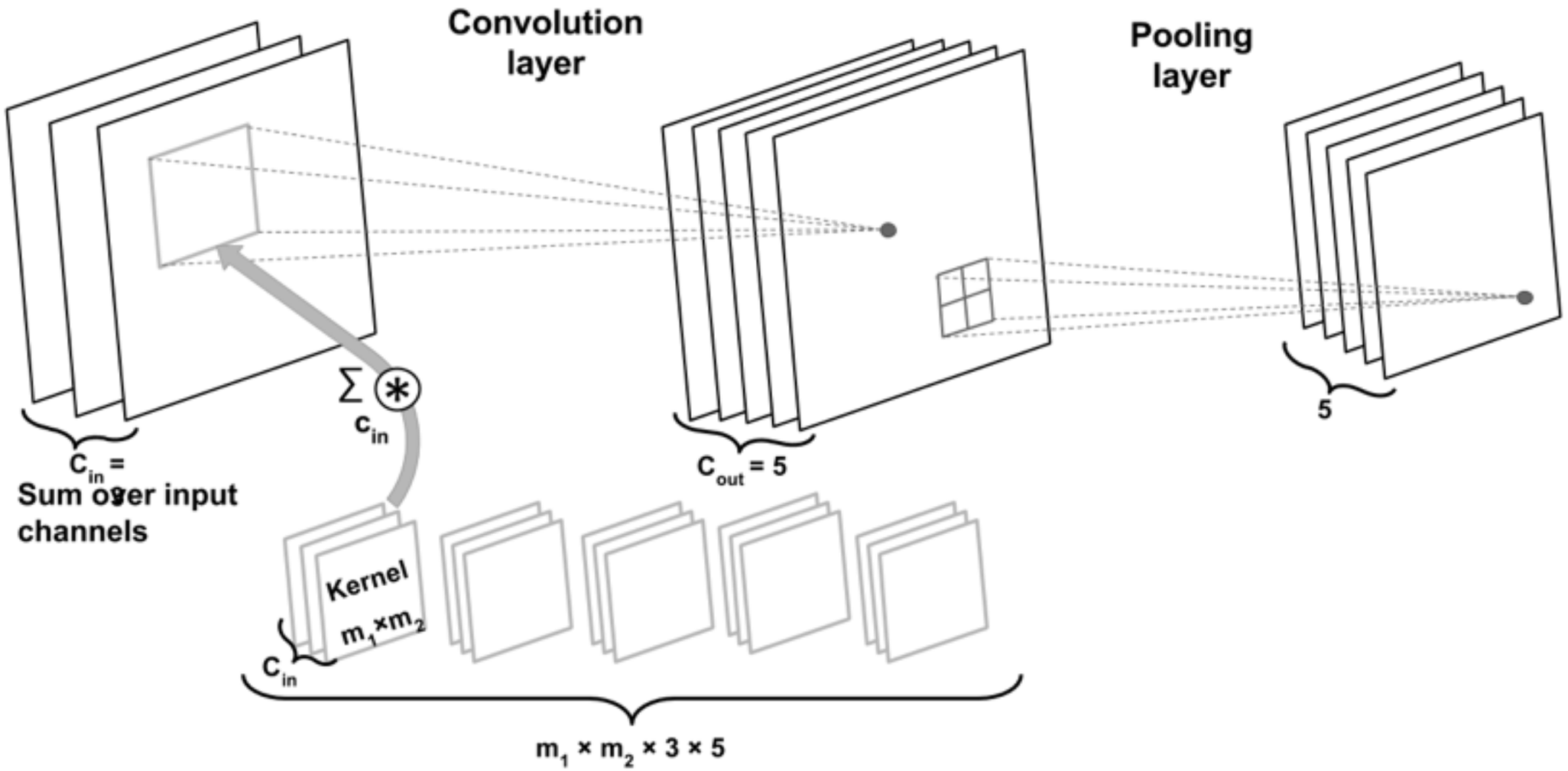


Image dimension: $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times c_{in}}$ in NWHC format,
CUDA & PyTorch use NCWH

Convolutions with Color Channels

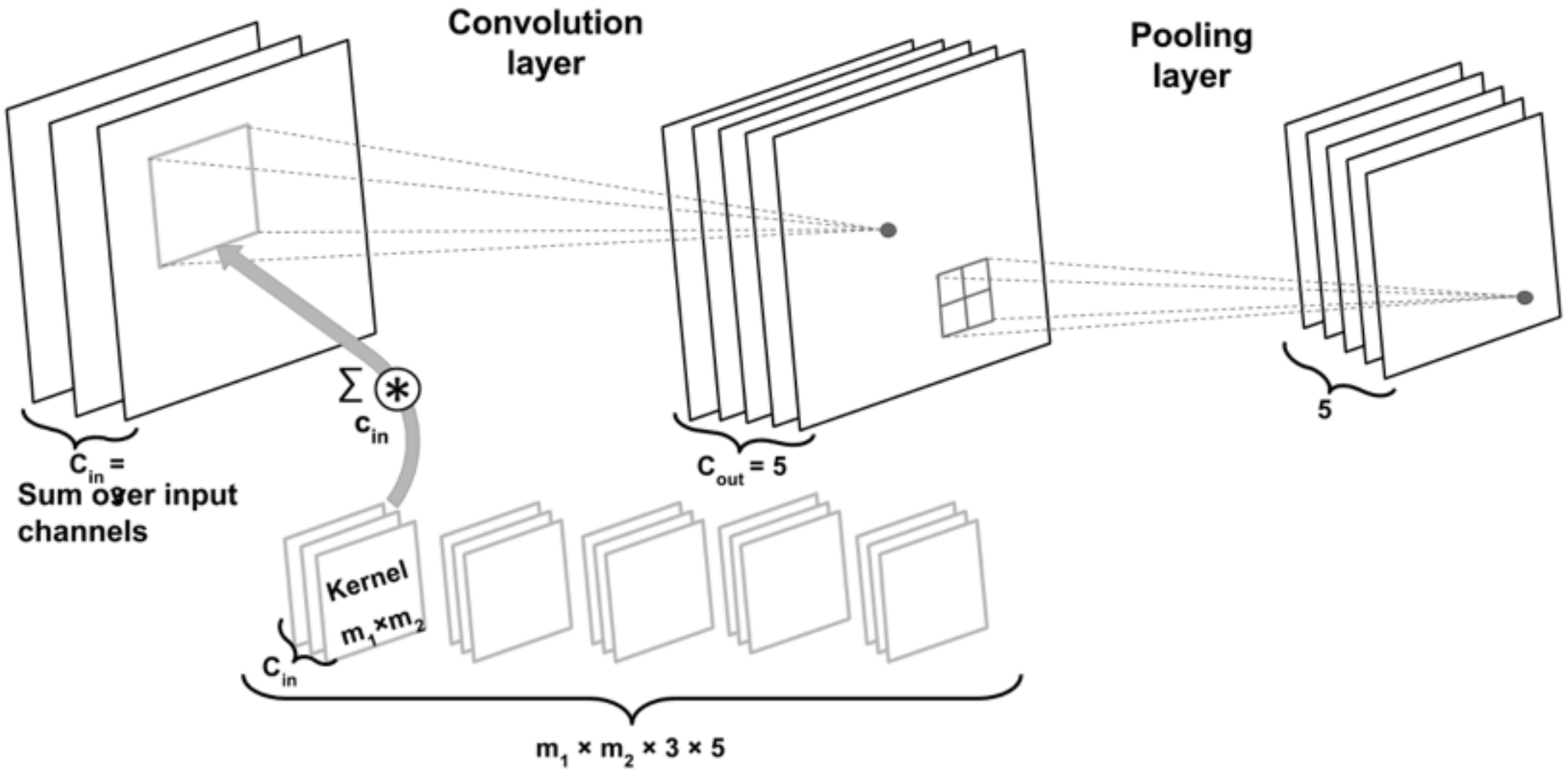


Image dimensions: $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times c_{in}}$

Kernel dimensions: $\mathbf{W} \in \mathbb{R}^{m_1 \times m_2 \times c_{in} \times c_{out}}$ $\mathbf{b} \in \mathbb{R}^{c_{out}}$

Convolutions with Color Channels

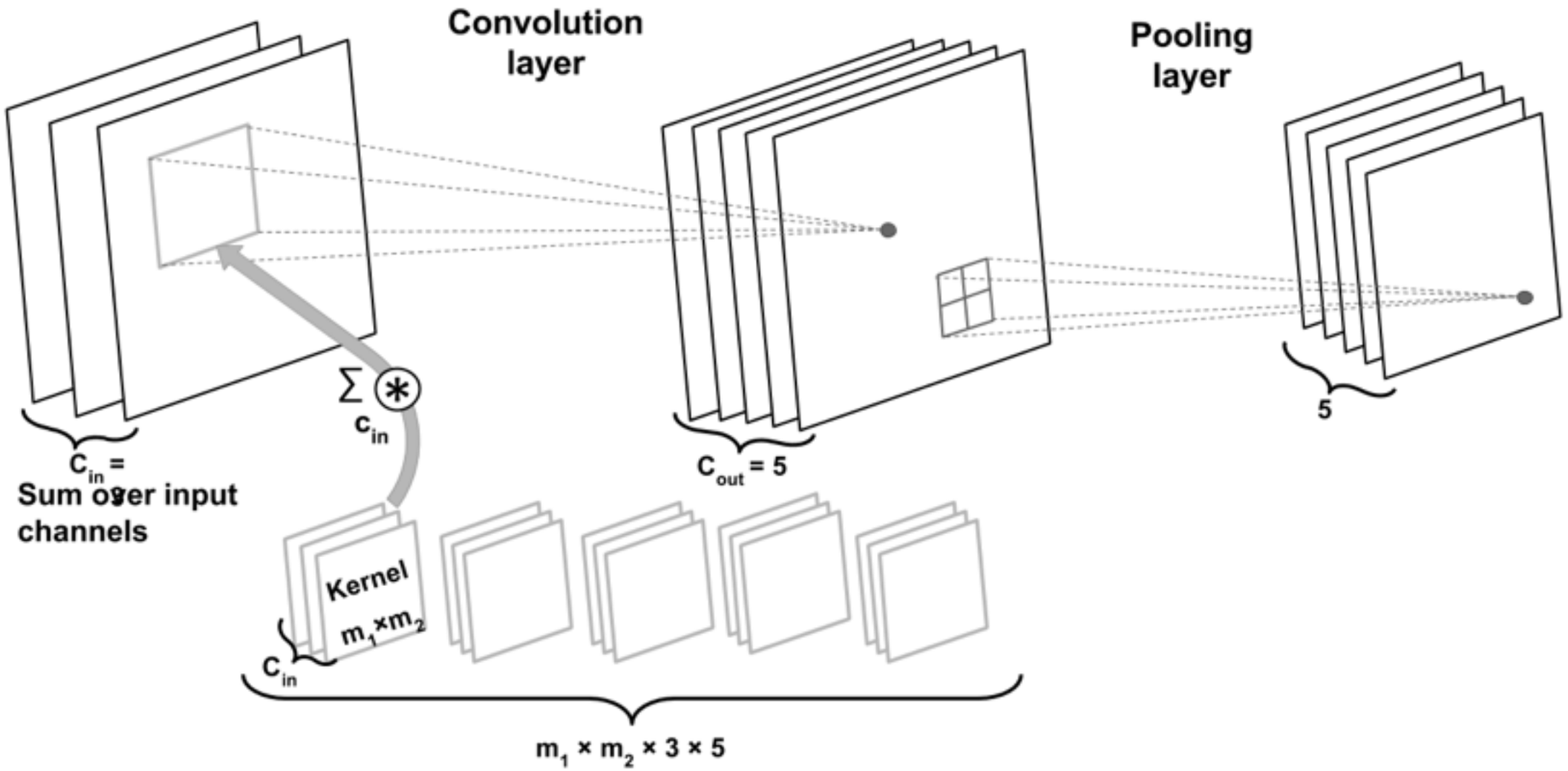
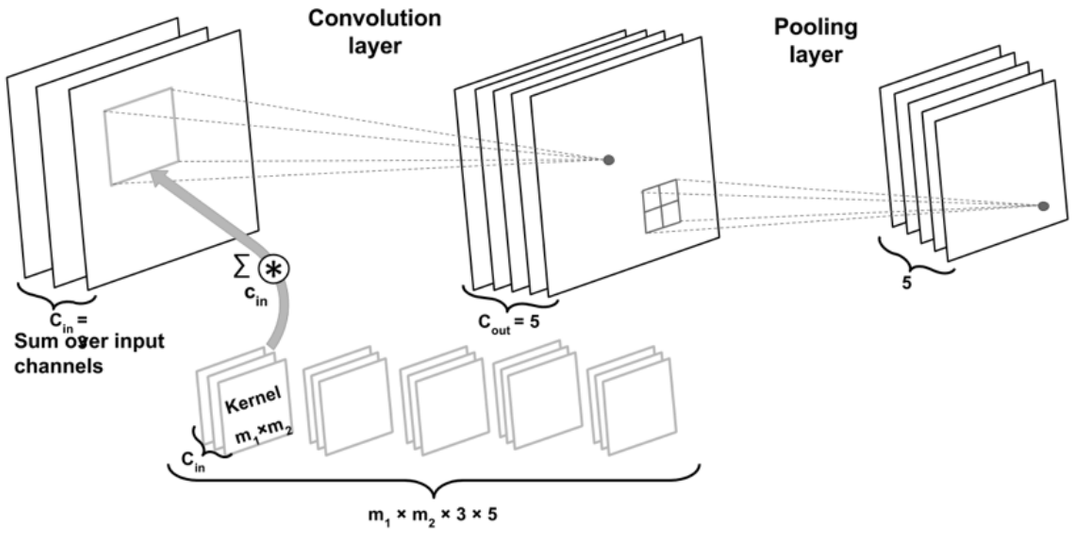


Image dimensions: $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times c_{in}}$

Kernel dimensions: $\mathbf{W} \in \mathbb{R}^{m_1 \times m_2 \times c_{in} \times c_{out}}$ $\mathbf{b} \in \mathbb{R}^{c_{out}}$

Convolutions with Color Channels



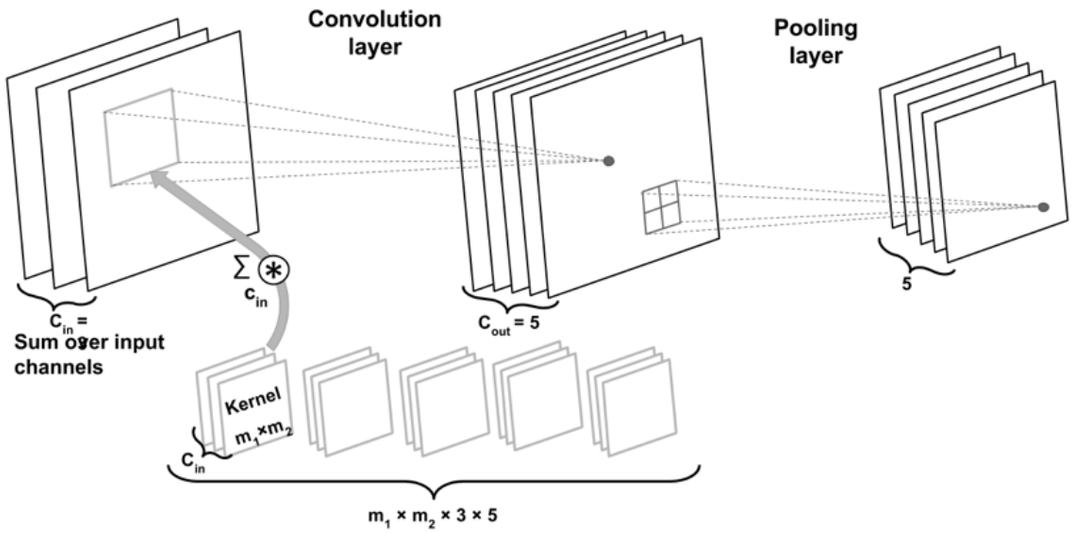
Number of parameters:

$$m_1 \times m_2 \times 3 \times 5 + 5$$

Assume 5x5 kernel:

$$5 \times 5 \times 3 \times 5 + 5 = 380$$

Convolutions with Color Channels



Assume "same" padding (more on padding later), such that the hidden layer has the same height and width as the original images

Assume 128x128 images and 5x5 kernel:

<u>If we use a CNN:</u>

Number of parameters:

$$m_1 \times m_2 \times 3 \times 5 + 5$$

$$5 \times 5 \times 3 \times 5 + 5 = 380$$

<u>If we use a fully connected layer:</u>

Number of parameters:

$$(n_1 \times n_2 \times 3) \times (n_1 \times n_2 \times 5) + (n_1 \times n_2 \times 5)$$

$$(n_1 \times n_2)^2 \times 3 \times 5 + n_1 \times n_2 \times 5$$

$$(128 \times 128)^2 \times 3 \times 5 + 128 \times 128 \times 5$$

$$= 4,026,613,760$$

To be continued ....