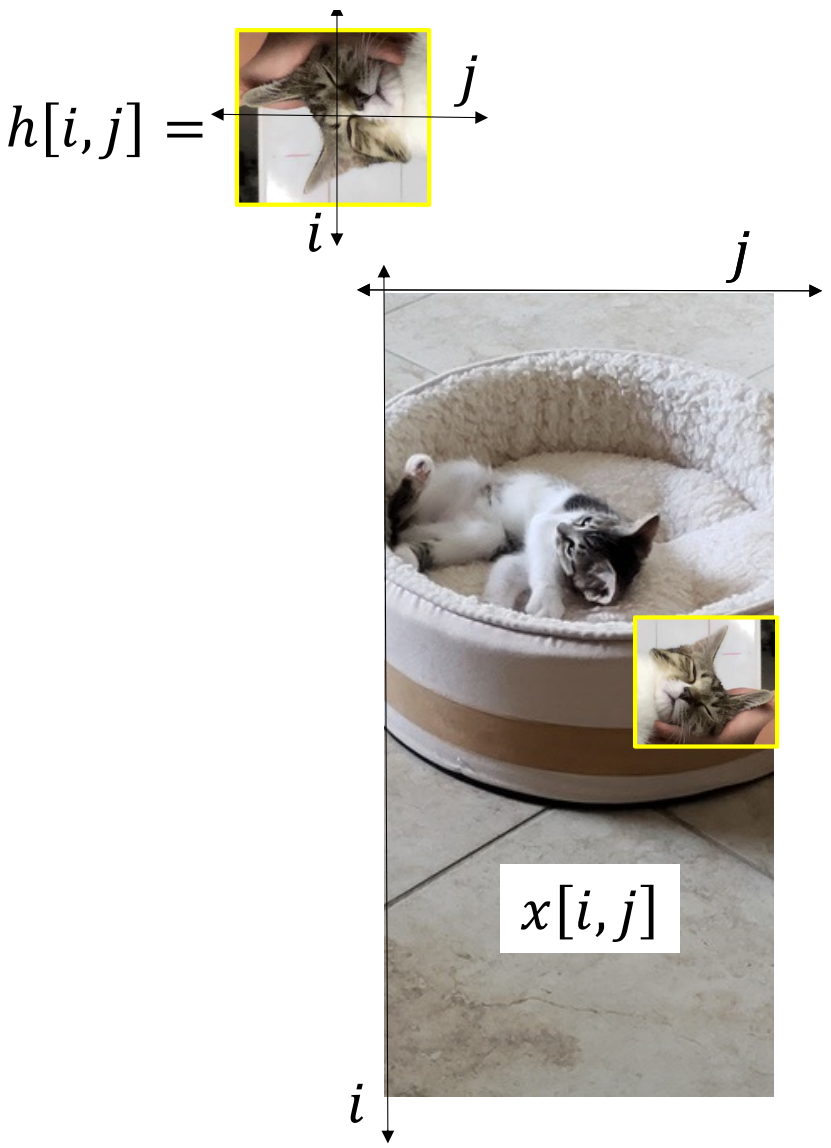
CS440/ECE448 Lecture 17 Computer Vision: Convolutional Networks

Mark Hasegawa-Johnson

Slides are in Public Domain: Re-Mix, Re-Use, Re-Distribute at will

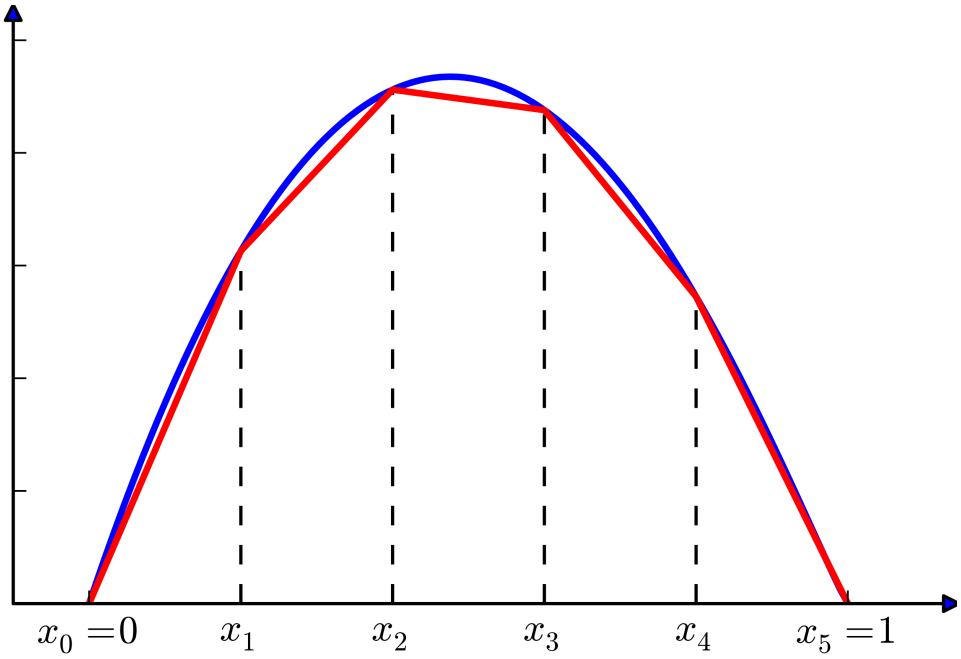


Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Remember: a neural network learns a PWL approximation of a nonlinear function

- We can approximate any nonlinear classifier using a PWL classifier
- In the limit, as the number of hidden nodes goes to infinity, the approximation becomes provably perfect

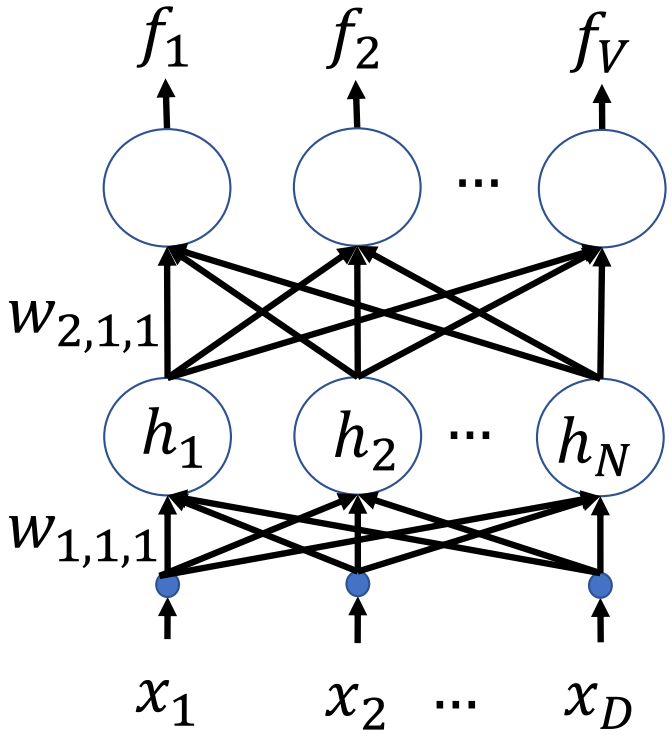


Public domain image, Krishnavedala, 2011

The first layer is a matrix followed by a nonlinearity.
The second layer is a linear classifier.

$$\boldsymbol{f} = \text{softmax}(\boldsymbol{W}_2\boldsymbol{h})$$

$$\boldsymbol{h} = \text{ReLU}(\boldsymbol{W}_1\boldsymbol{x})$$

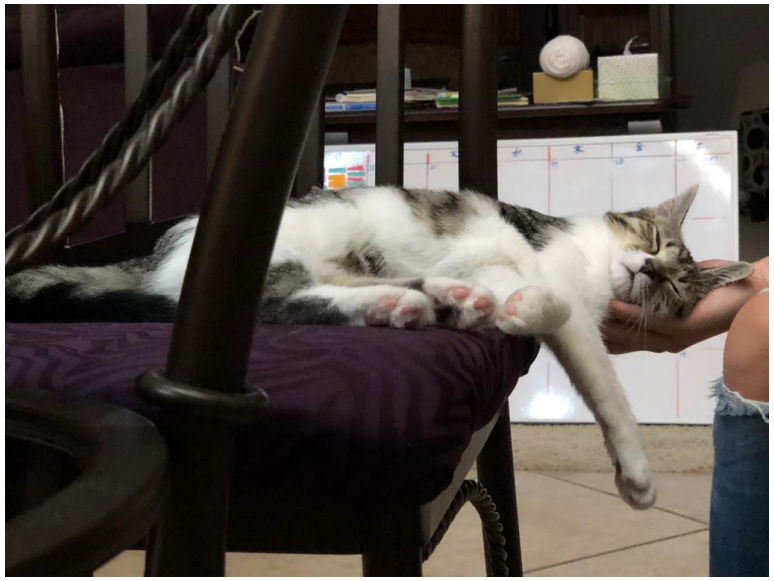


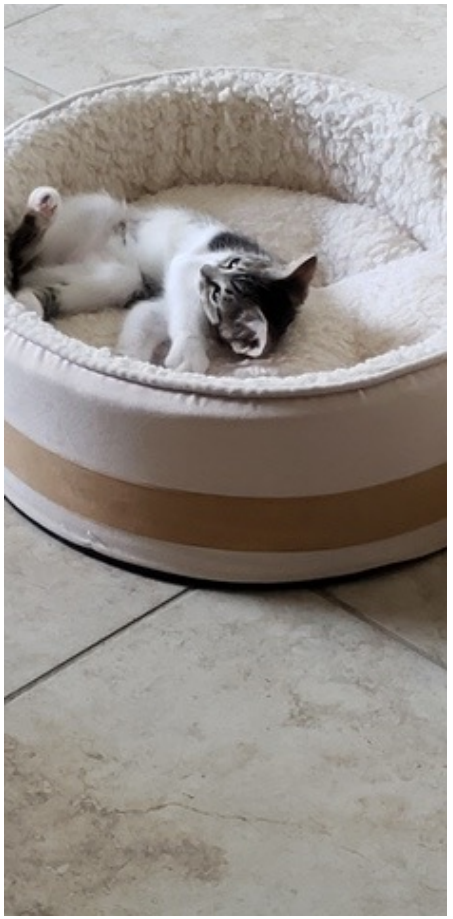
Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

The problem of shift invariance

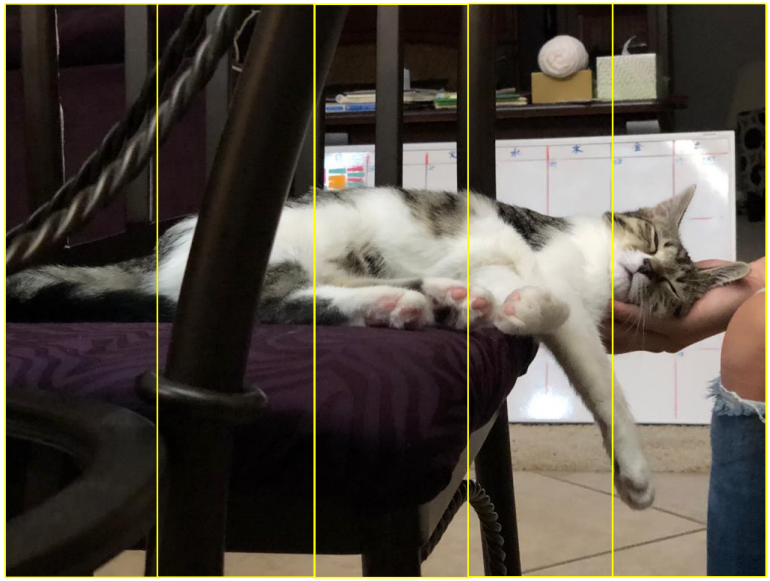
- These two images each contain a cat.
- How can a neural network tell?





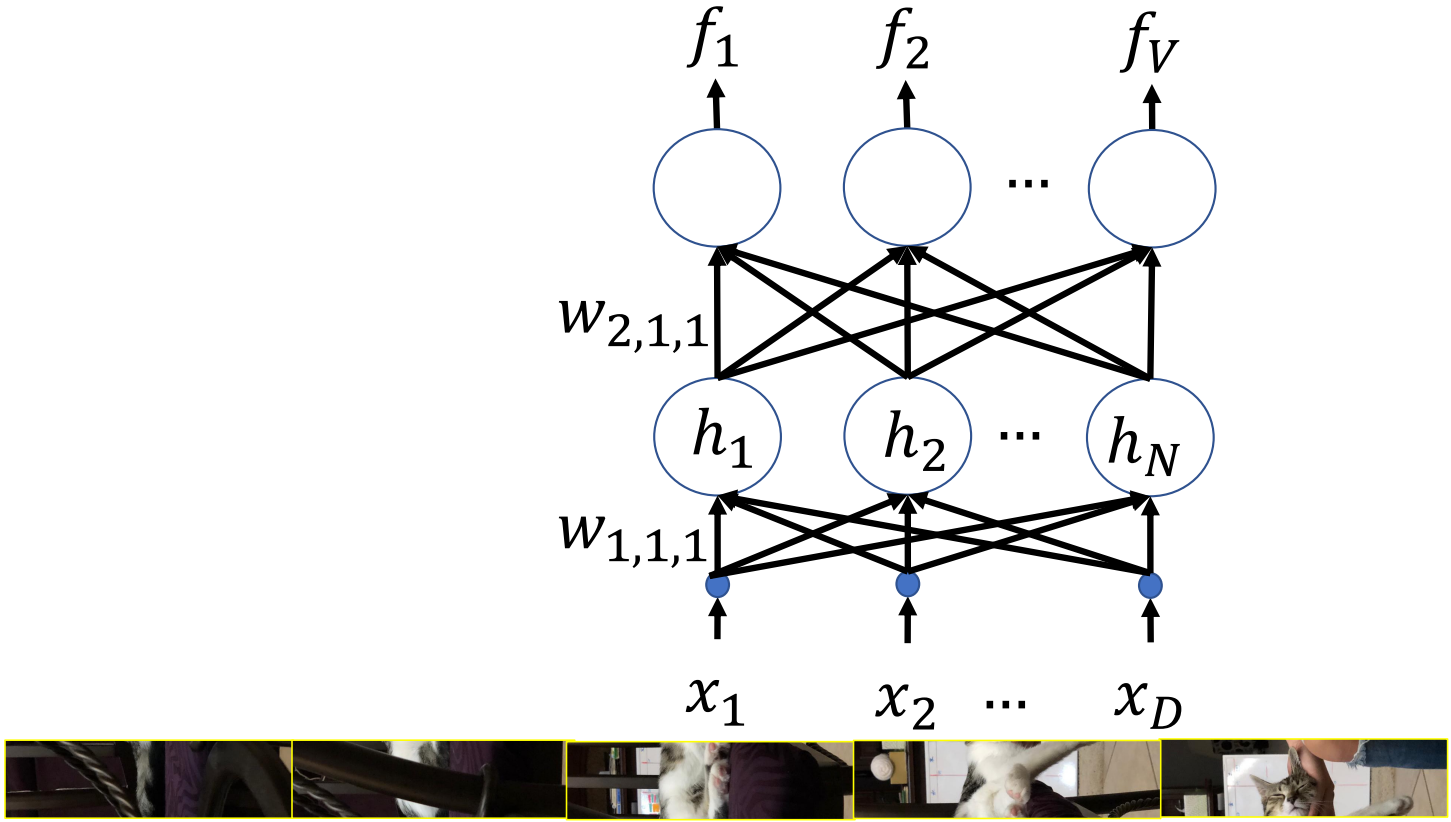
Fully-connected network

A fully-connected network converts the image into a vector, then multiplies it by a matrix.



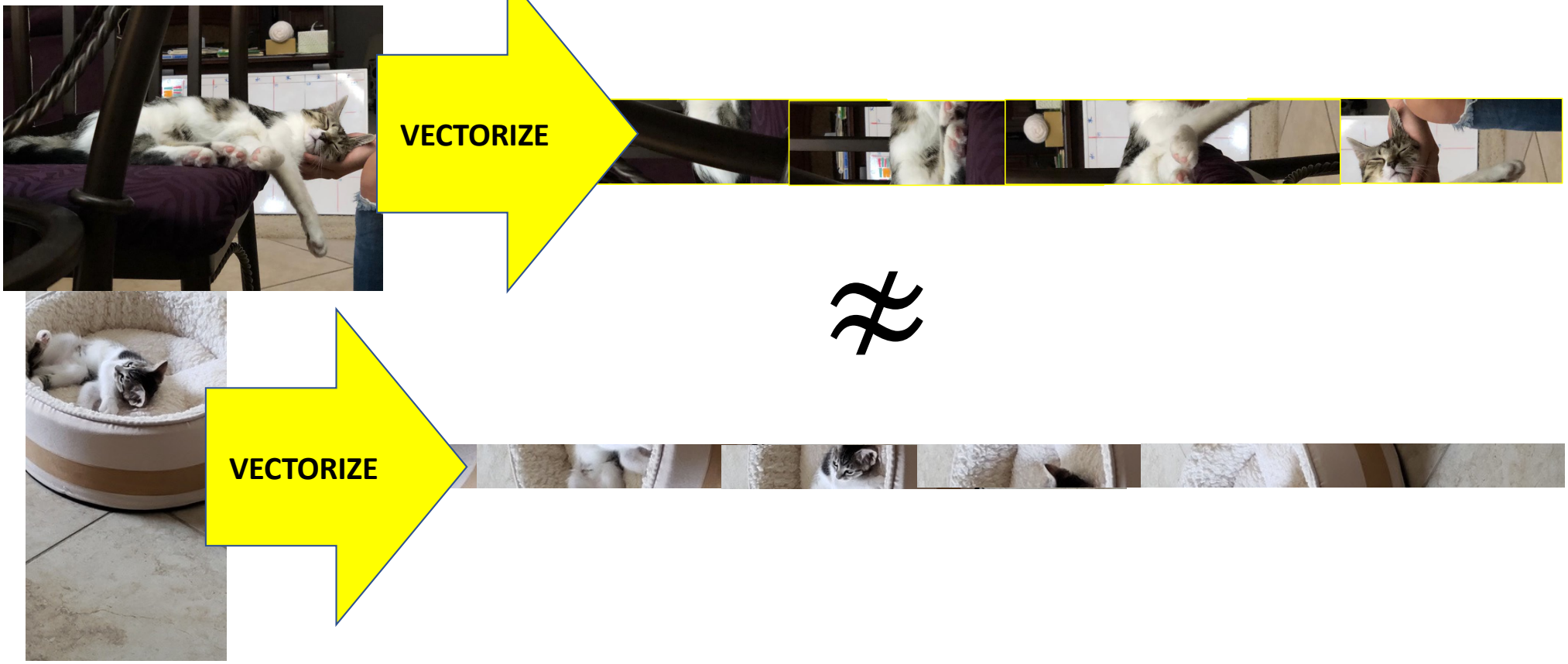
Fully-connected network

A fully-connected network converts the image into a vector, then multiplies it by a matrix.



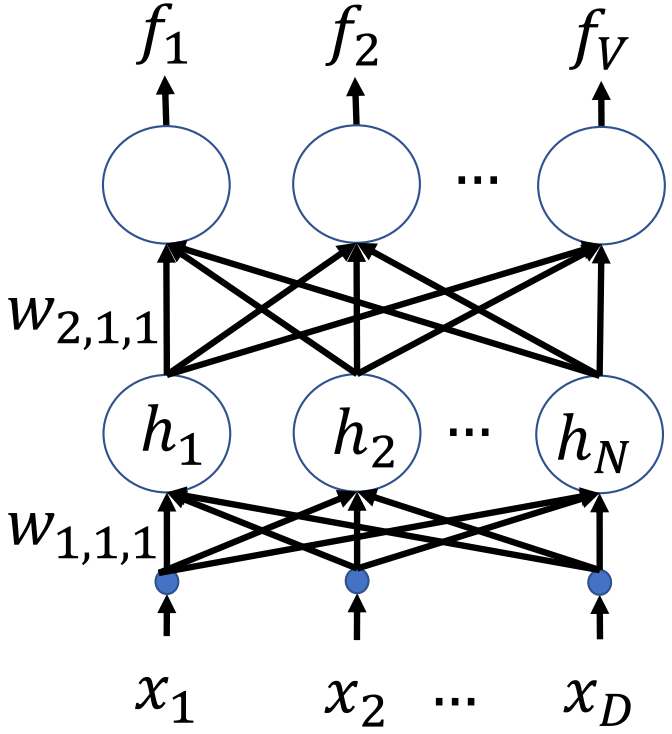
The problem of shift invariance

…is that, even though both images contain a cat, the vectorized versions of these two images are dissimilar.



The first layer is a matrix followed by a nonlinearity.
The second layer is a linear classifier.

- Since the vectors aren't similar, they won't produce similar hidden features.
- Since the hidden features aren't similar, they won't be recognized as the same animal.

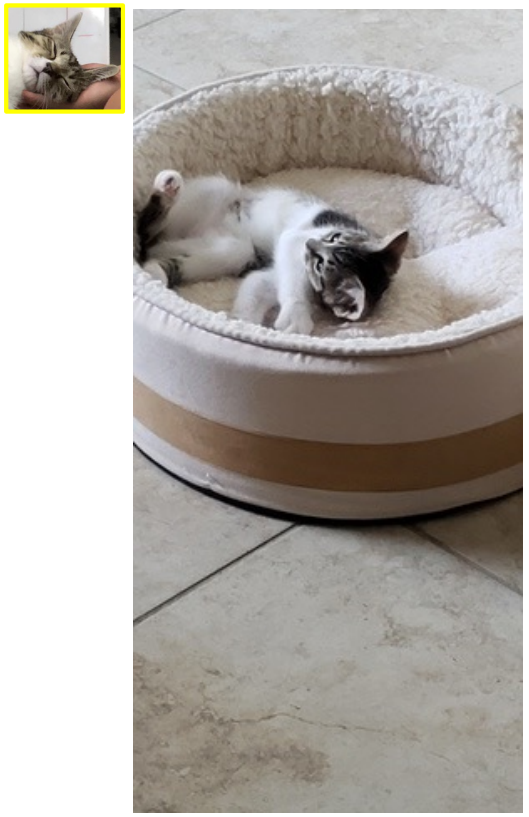


Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

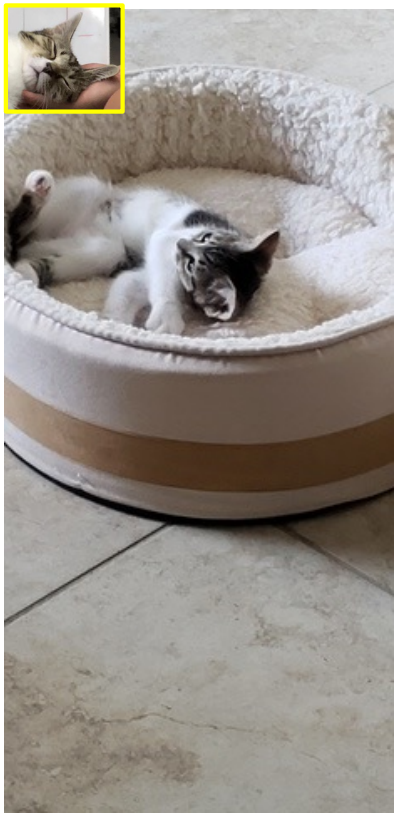




Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

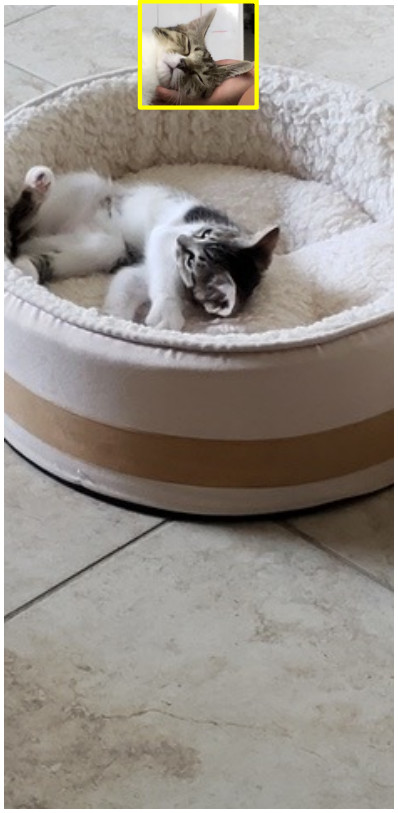


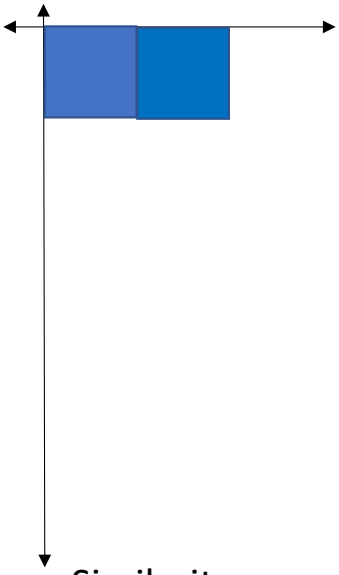


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

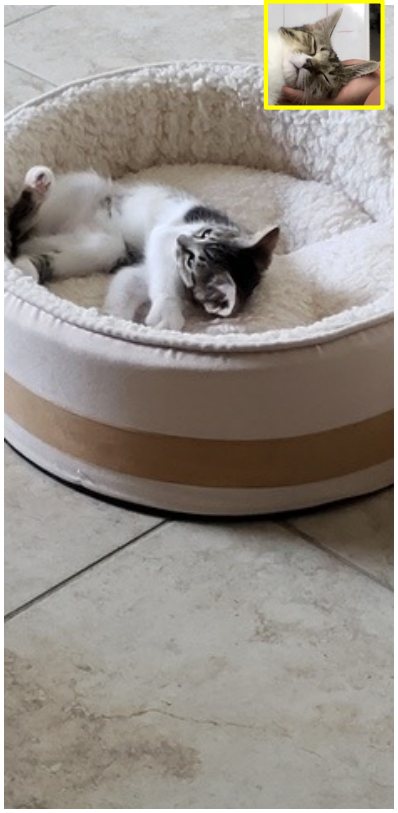


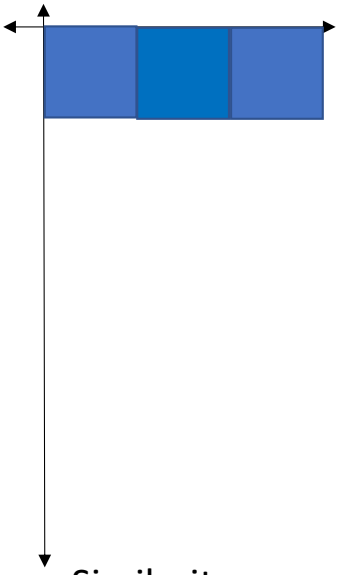


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

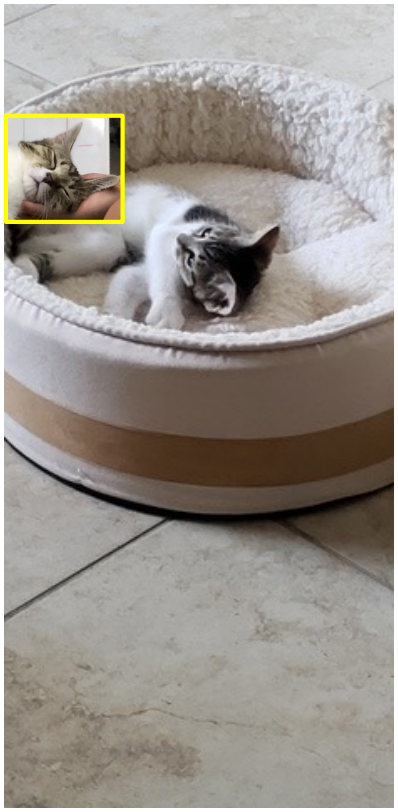


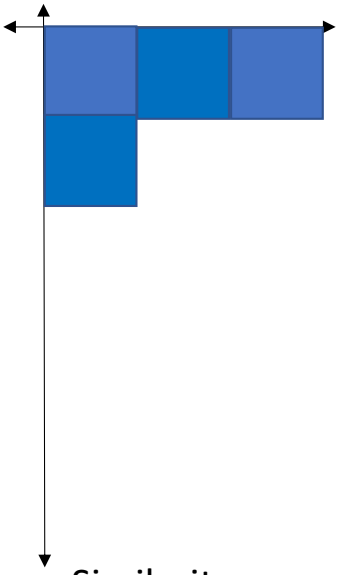


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

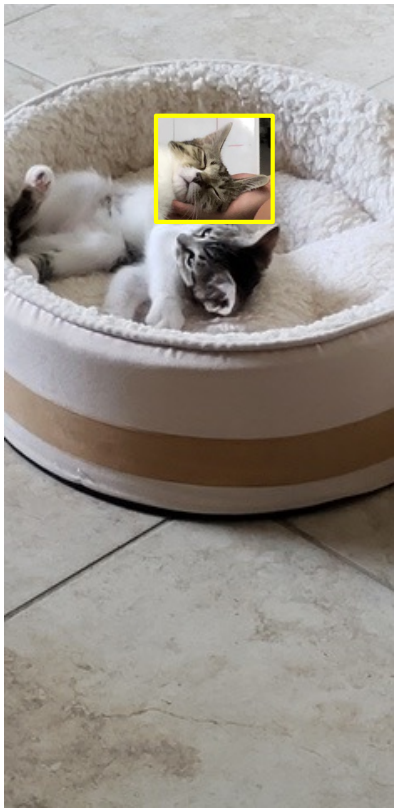


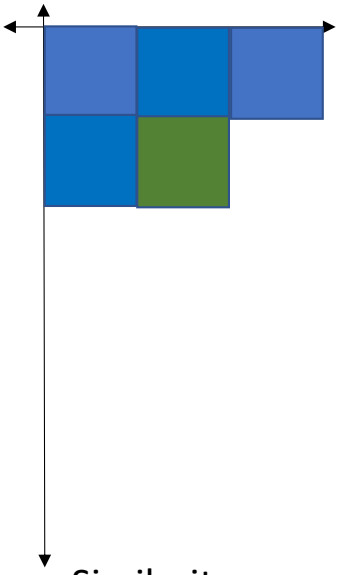


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

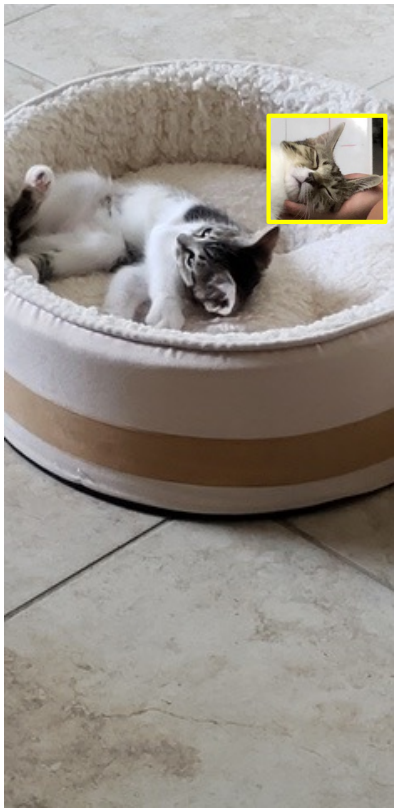


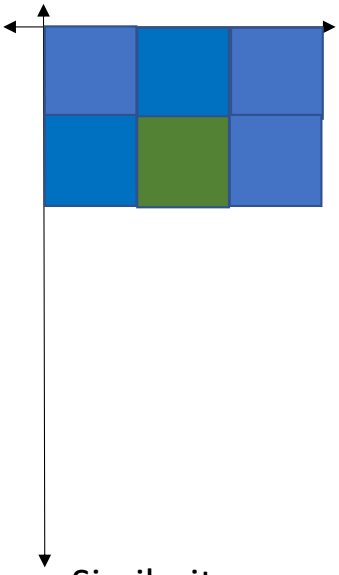


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

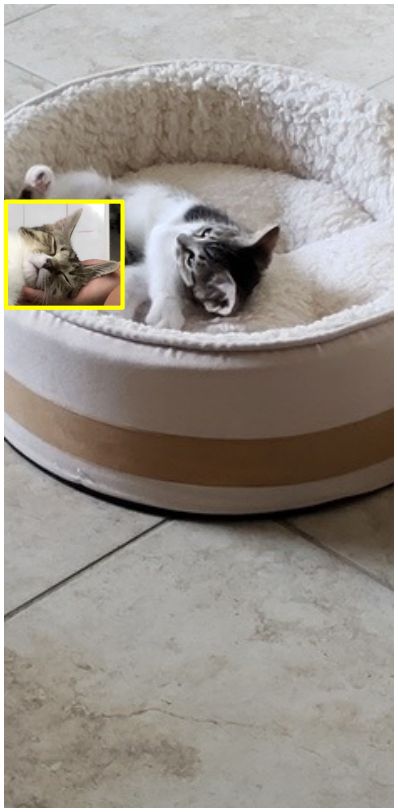


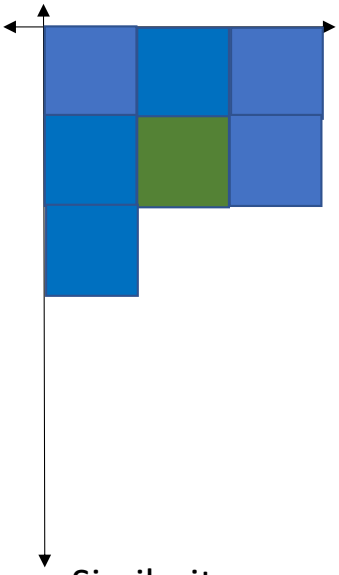


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

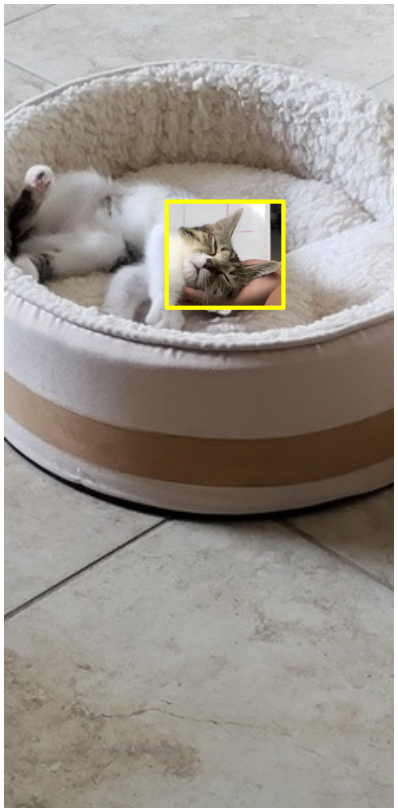


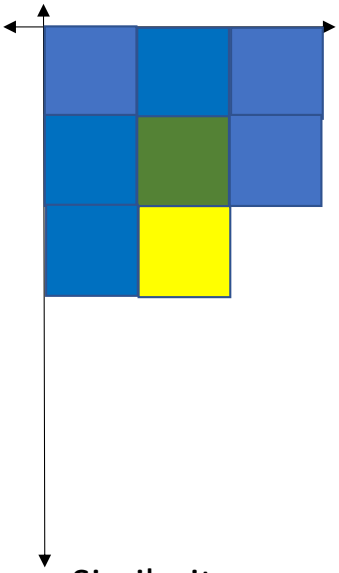


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

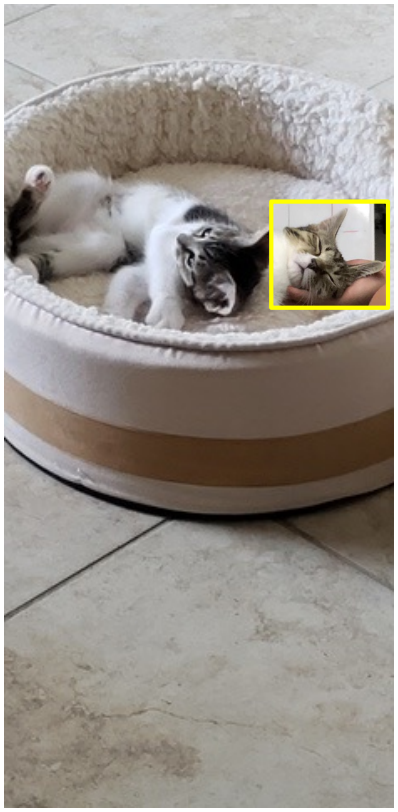


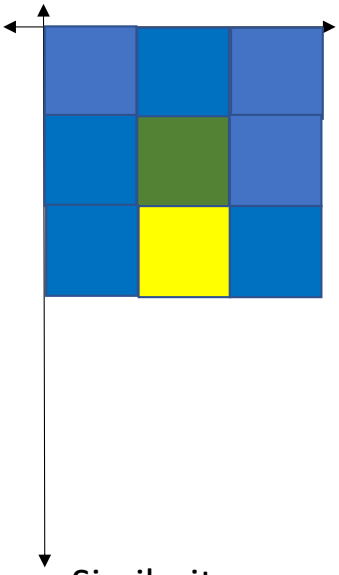


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

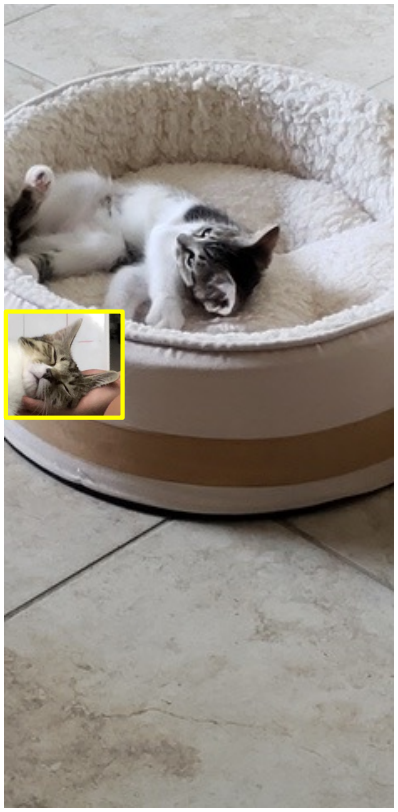


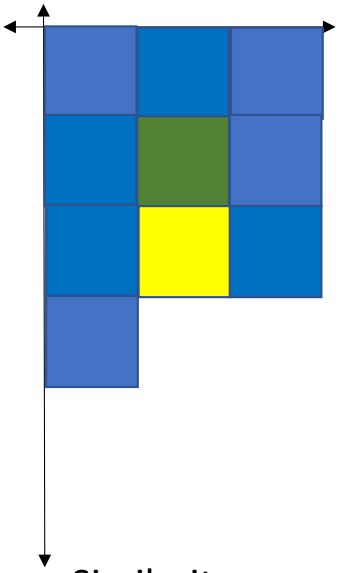


Similarity as a function of row and column indices

The key idea of a convolutional neural network: local similarity judgments

- The key idea of a convolutional neural network is that we perform the similarity judgments locally, in every patch of the image.
- If any patch of the image resembles the target image, we score a recognition

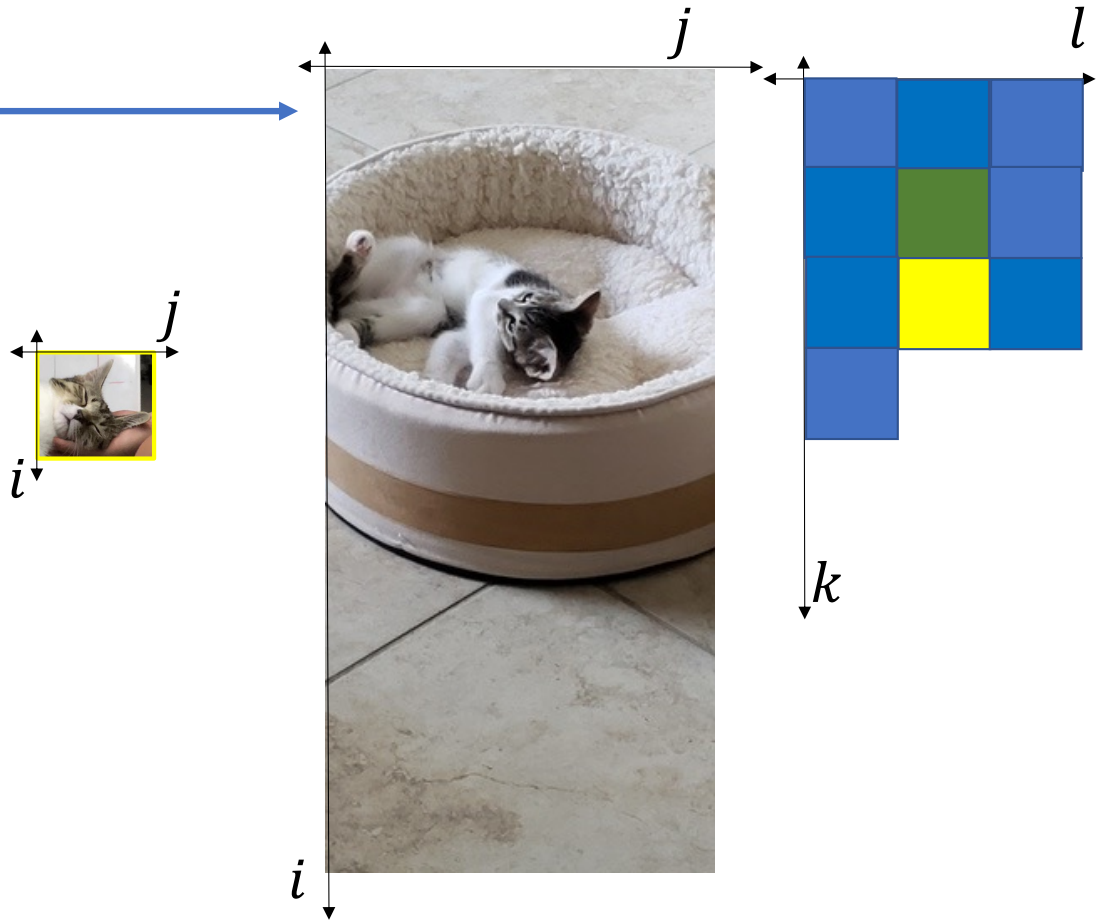




Similarity as a function of row and column indices

How do we measure similarity? …. The math.

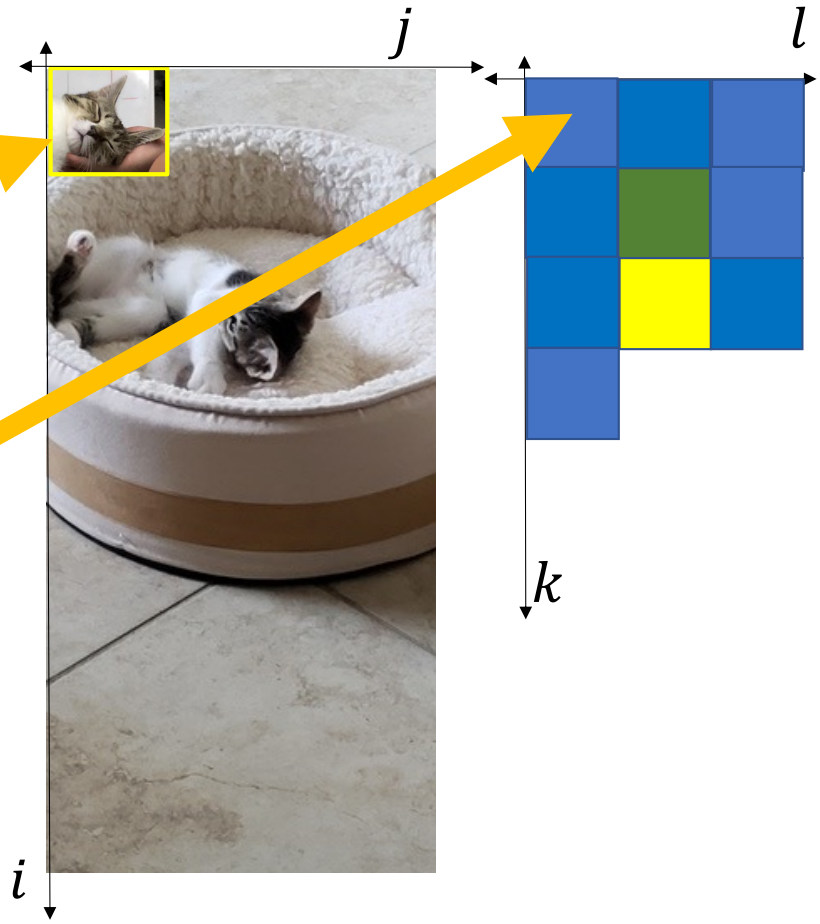
- Input image = $x[i, j]$
- Filter (pattern being detected) = $w[i, j]$
- Output image (grid of local detection strengths) = $y[k, l]$



How do we measure similarity?
Answer: local dot product = correlation

$$\sum_i \sum_j w[i,j]x[i,j]$$

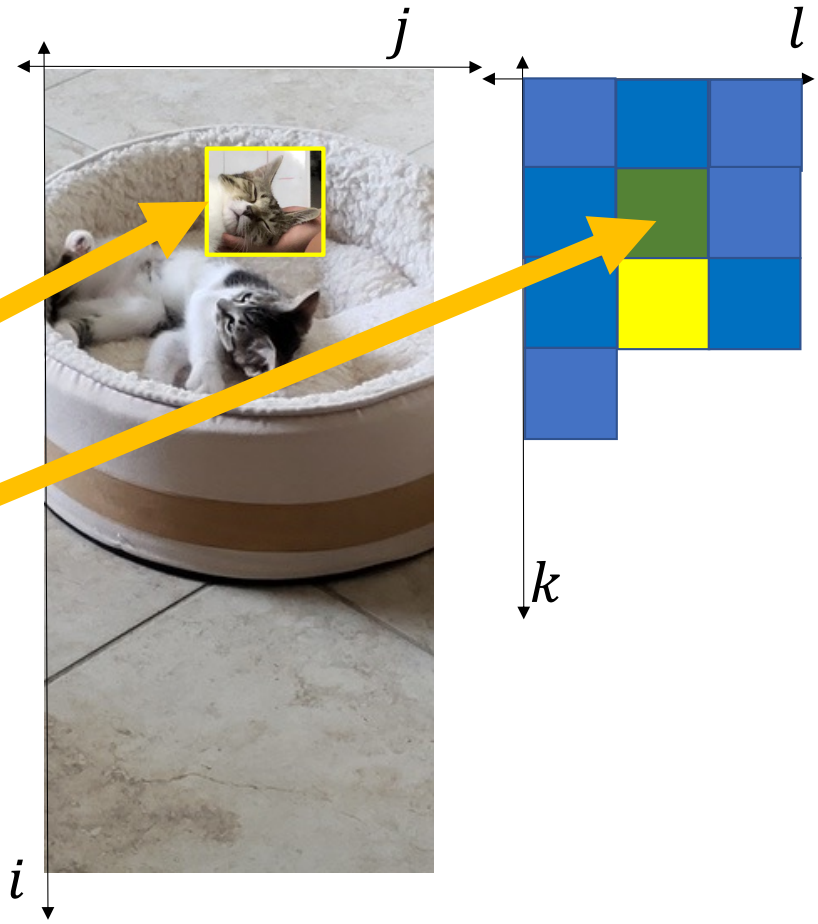
$$= y[0,0]$$



How do we measure similarity?
Answer: local dot product = correlation

$$\sum_i \sum_j w[i-k, j-l]x[i,j]$$

$$= y[k,l]$$



Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Correlation versus Convolution

- This formula is called the ***correlation*** of $w[i,j]$ with $x[i,j]$:

$$y[k,l] = \sum_i \sum_j w[i-k, j-l]x[i,j]$$

- This formula is called the ***convolution*** of $w[i,j]$ with $x[i,j]$:

$$y[k,l] = \sum_i \sum_j w[k-i, l-j]x[i,j]$$

Why use convolution if correlation is easier to visualize?

- Historical reason: Convolution is related to Fourier transform
- Reason it's still used: Convolution is symmetric:

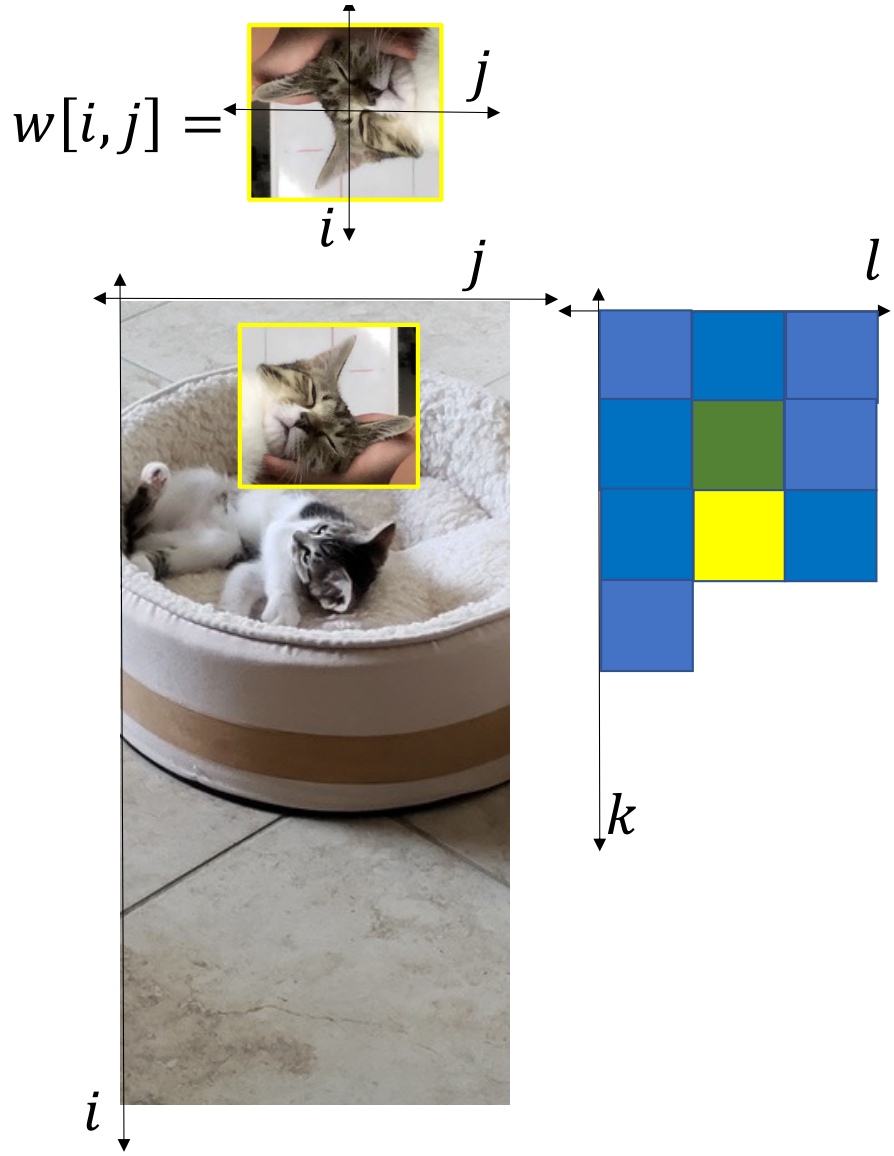
$$\sum_i \sum_j w[k-i, l-j]x[i,j] = \sum_i \sum_j w[i,j]x[k-i, l-j]$$

… whereas correlation is antisymmetric:

$$\sum_i \sum_j w[i-k, j-l]x[i,j] = \sum_i \sum_j w[i,j]x[k+i, l+j]$$

Convolutional neural net

A convolutional neural net is the same thing as a correlational neural net, except that the filter is defined upside down and backward, like this.

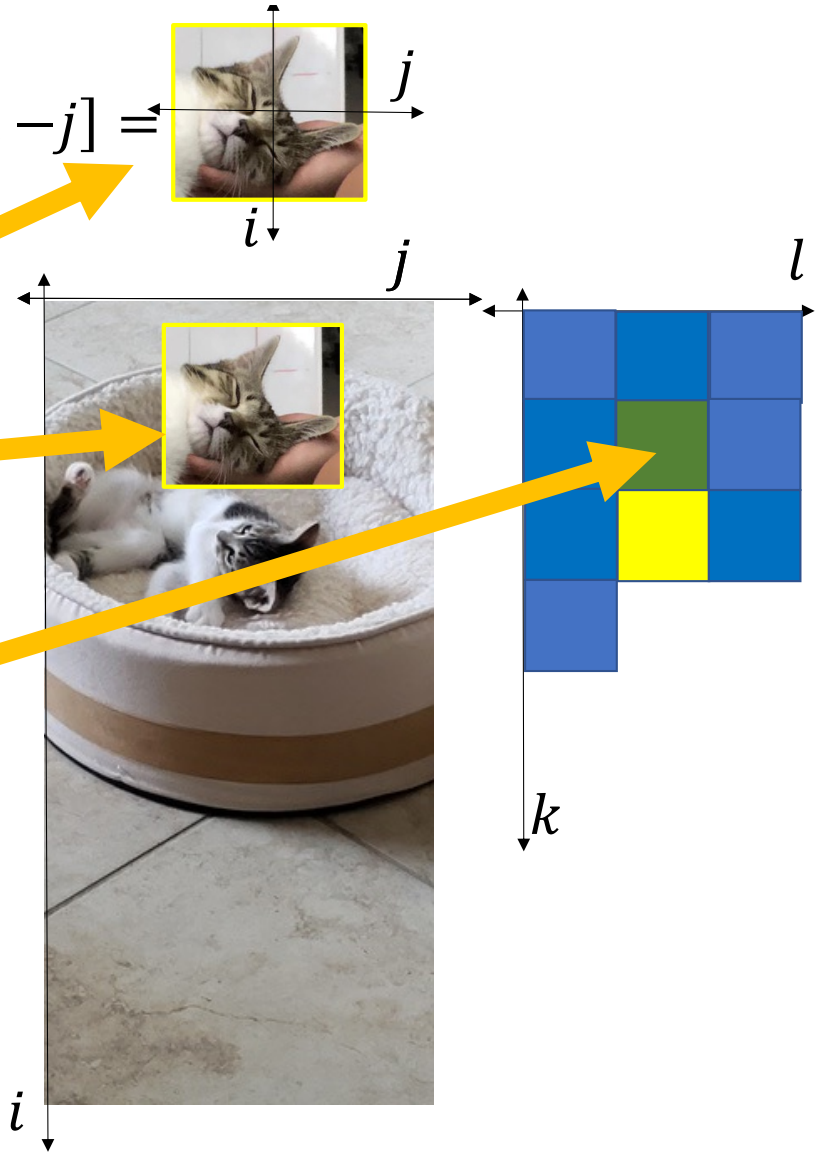


Convolutional neural net

The convolution operation then

- Flips the filter rightside up,
- Shifts it to $w[k-i, l-j]$
- Computes the similarity:

$$y[k,l] = \sum_i \sum_j w[k-i, l-j] x[i,j]$$

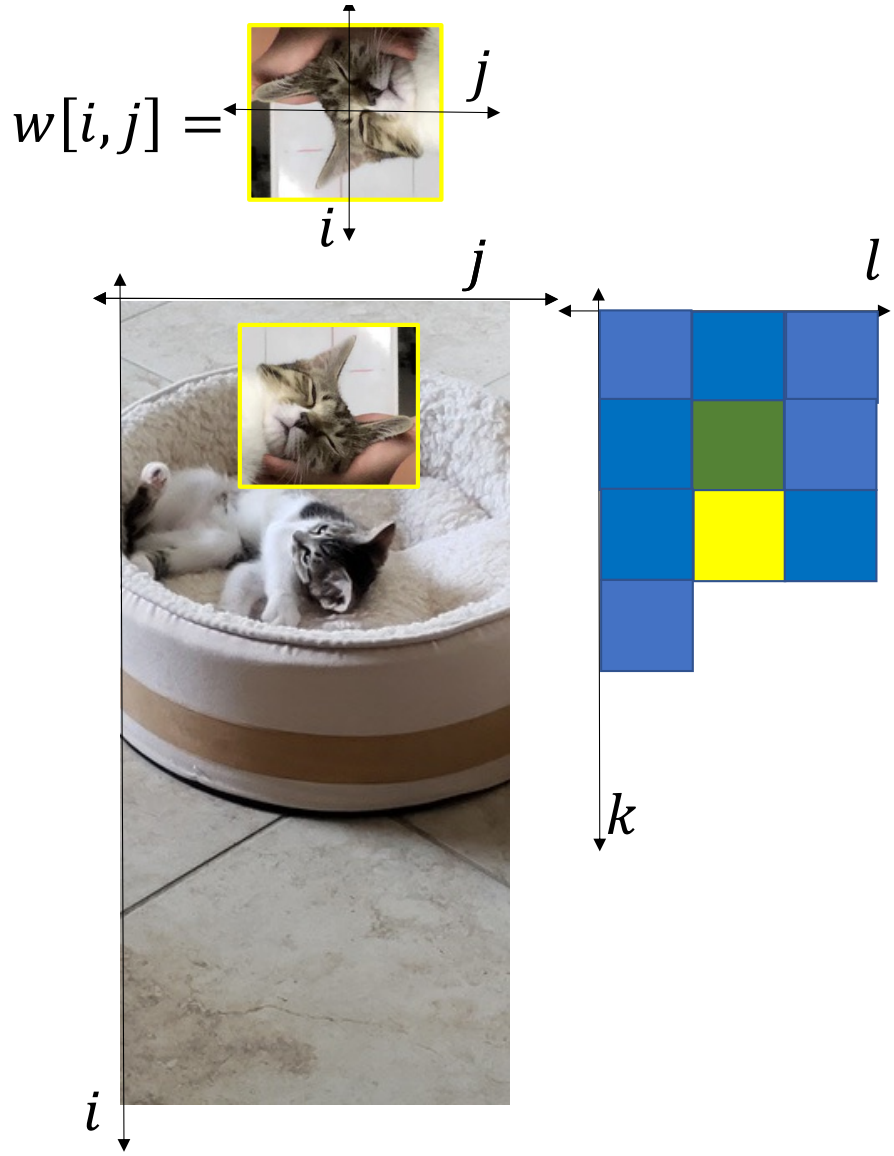


Convolutional neural net

$$y[k,l] = \sum_i \sum_j w[k-i, l-j] x[i,j]$$

This is often written as

$$y[k,l] = w[k,l] * x[k,l]$$



Quiz

- Try the quiz!

Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Back-prop through convolution

- Suppose we've computed $y[k,l] = \sum_i \sum_j w[i,j]x[k-i,l-j]$
- Now suppose we know $\frac{d\mathcal{L}}{dy[k,l]}$
- We want to train $w[i,j] = w[i,j] - \eta \frac{d\mathcal{L}}{dw[i,j]}$
- How do we find $\frac{d\mathcal{L}}{dw[i,j]}$?

Back-prop through convolution = correlation!

Answer: use the chain rule!

$y[k,l] = \sum_i \sum_j w[i,j]x[k-i, l-j]$, so…

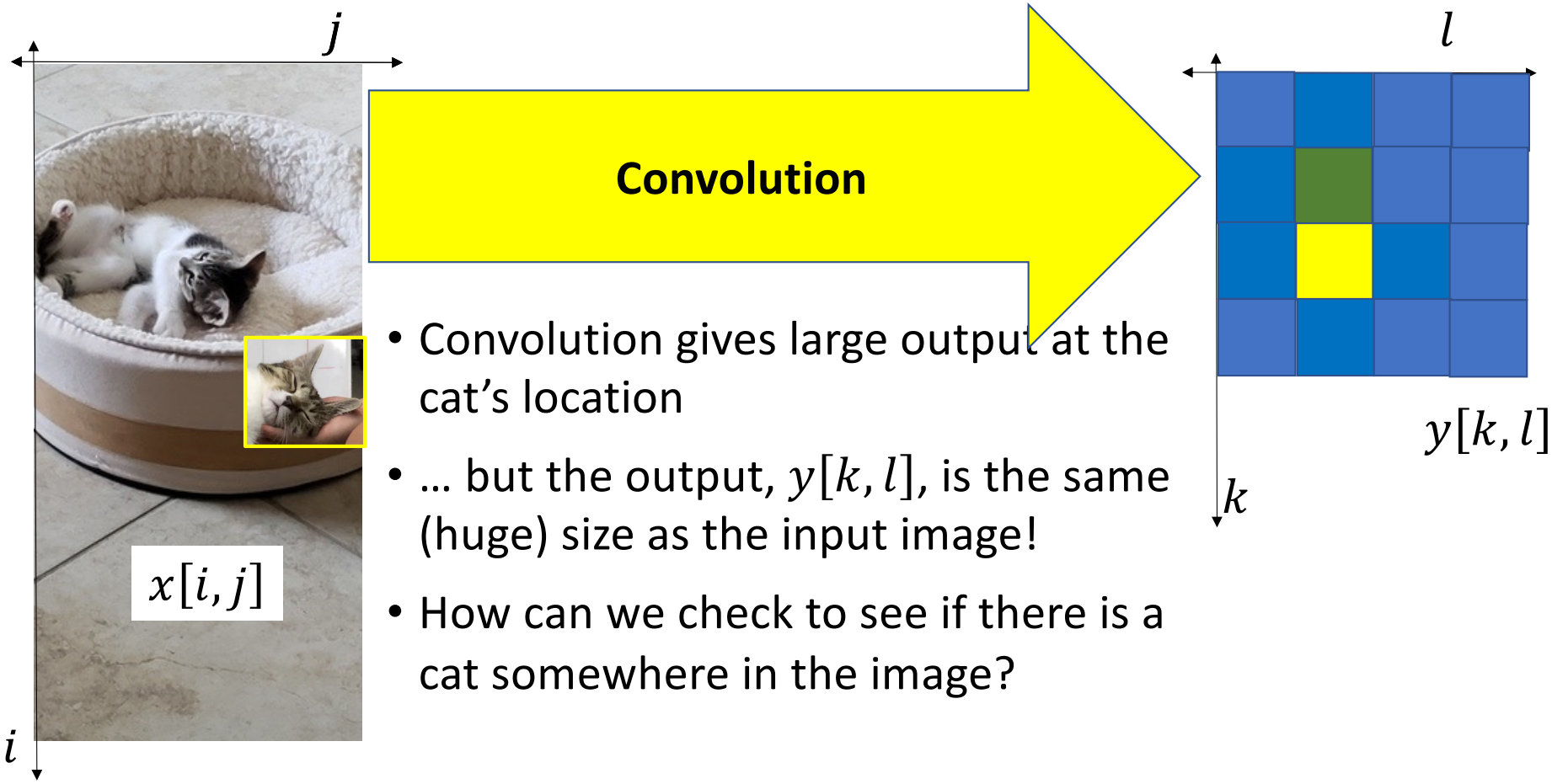
$$\frac{d\mathcal{L}}{dw[i,j]} = \sum_k \sum_l \frac{d\mathcal{L}}{dy[k,l]} \frac{dy[k,l]}{dw[i,j]} = \sum_k \sum_l \frac{d\mathcal{L}}{dy[k,l]} x[k-i, l-j]$$

= correlate($x[k,l], \frac{d\mathcal{L}}{dy[k,l]}$)!

Computer Vision: Object Recognition

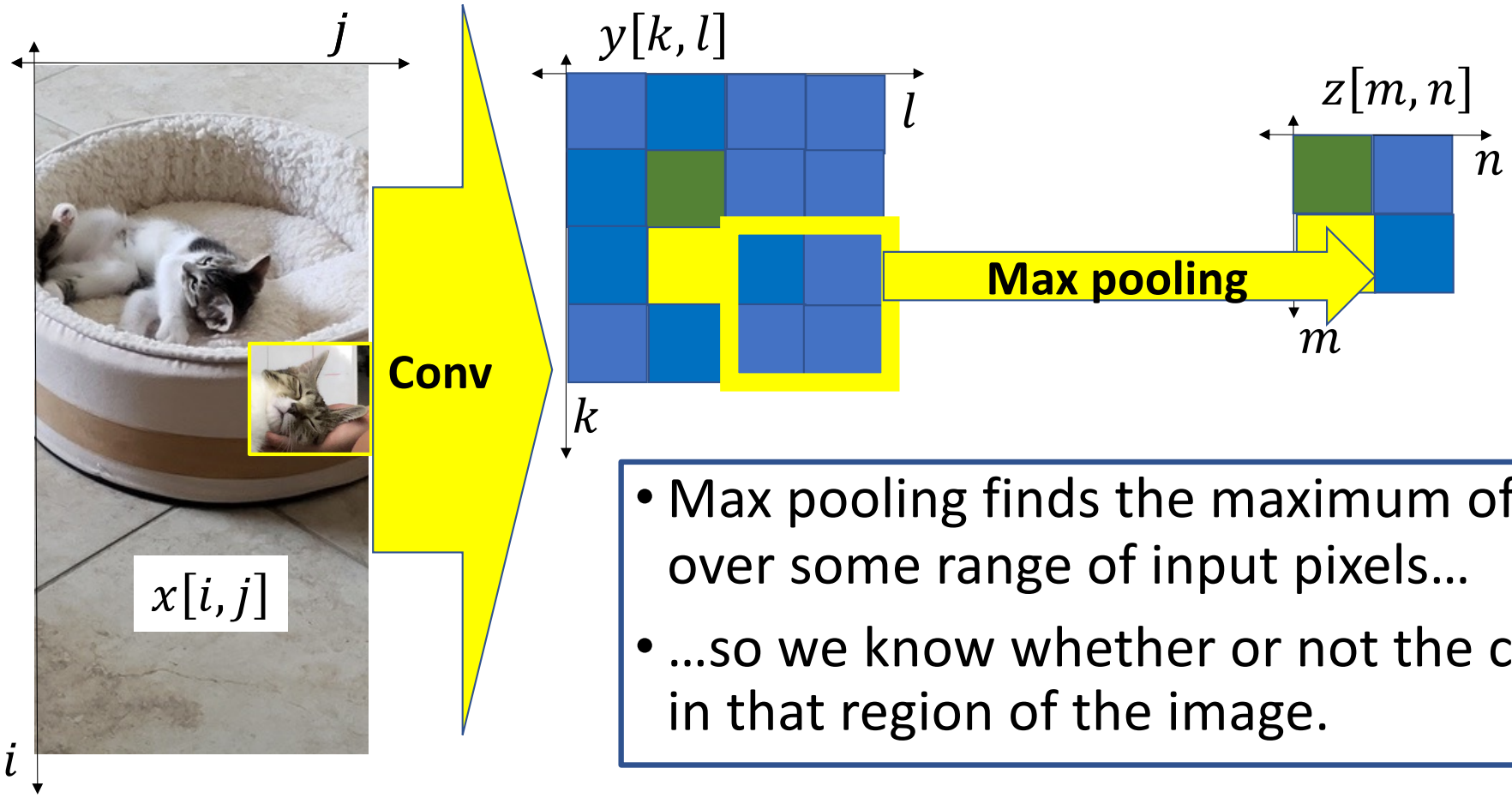
- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Too much information



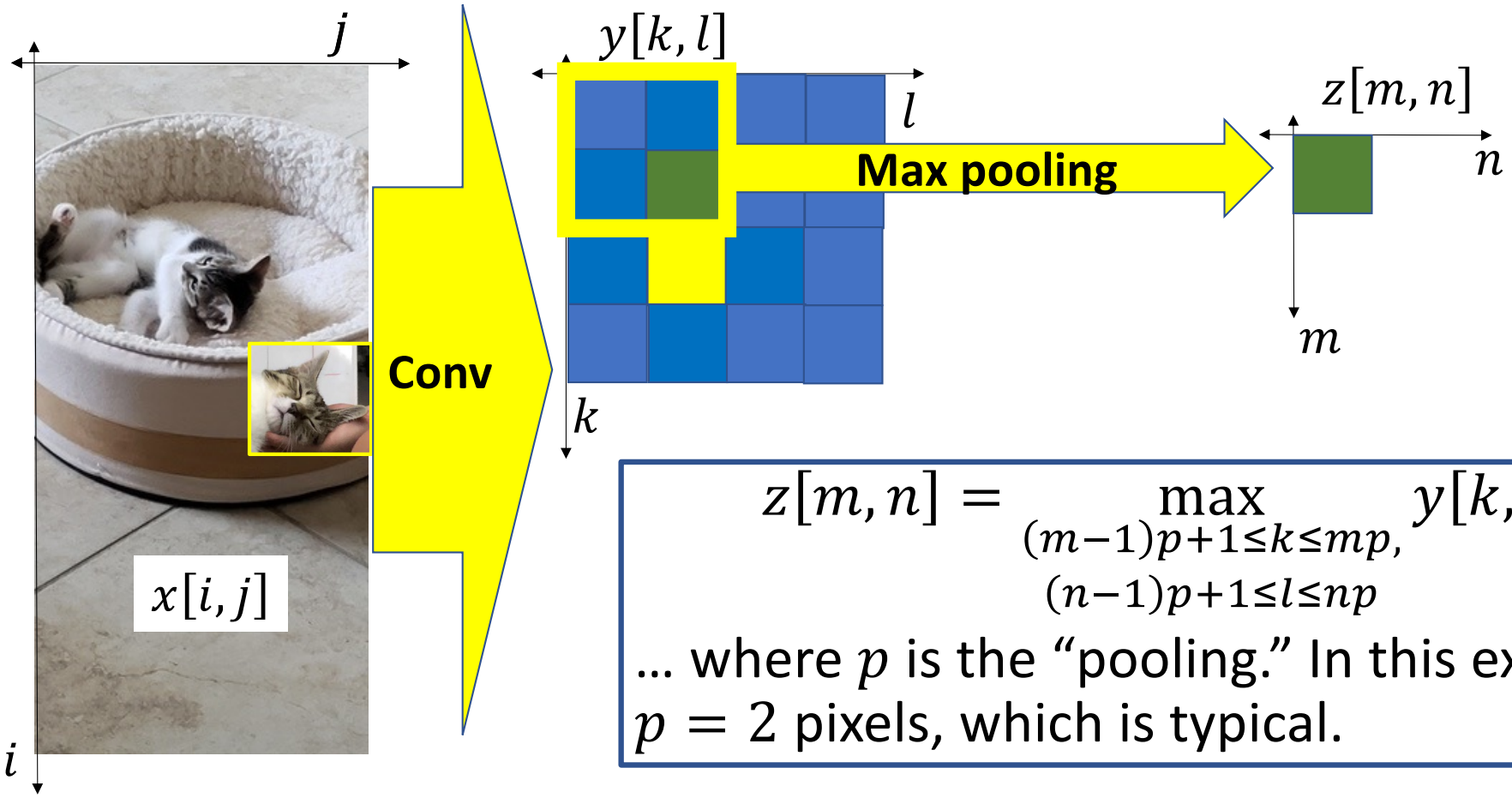
- Convolution gives large output at the cat's location
- … but the output, $y[k,l]$, is the same (huge) size as the input image!
- How can we check to see if there is a cat somewhere in the image?

Max Pooling



- Max pooling finds the maximum of $y[k,l]$ over some range of input pixels…
- …so we know whether or not the cat was in that region of the image.

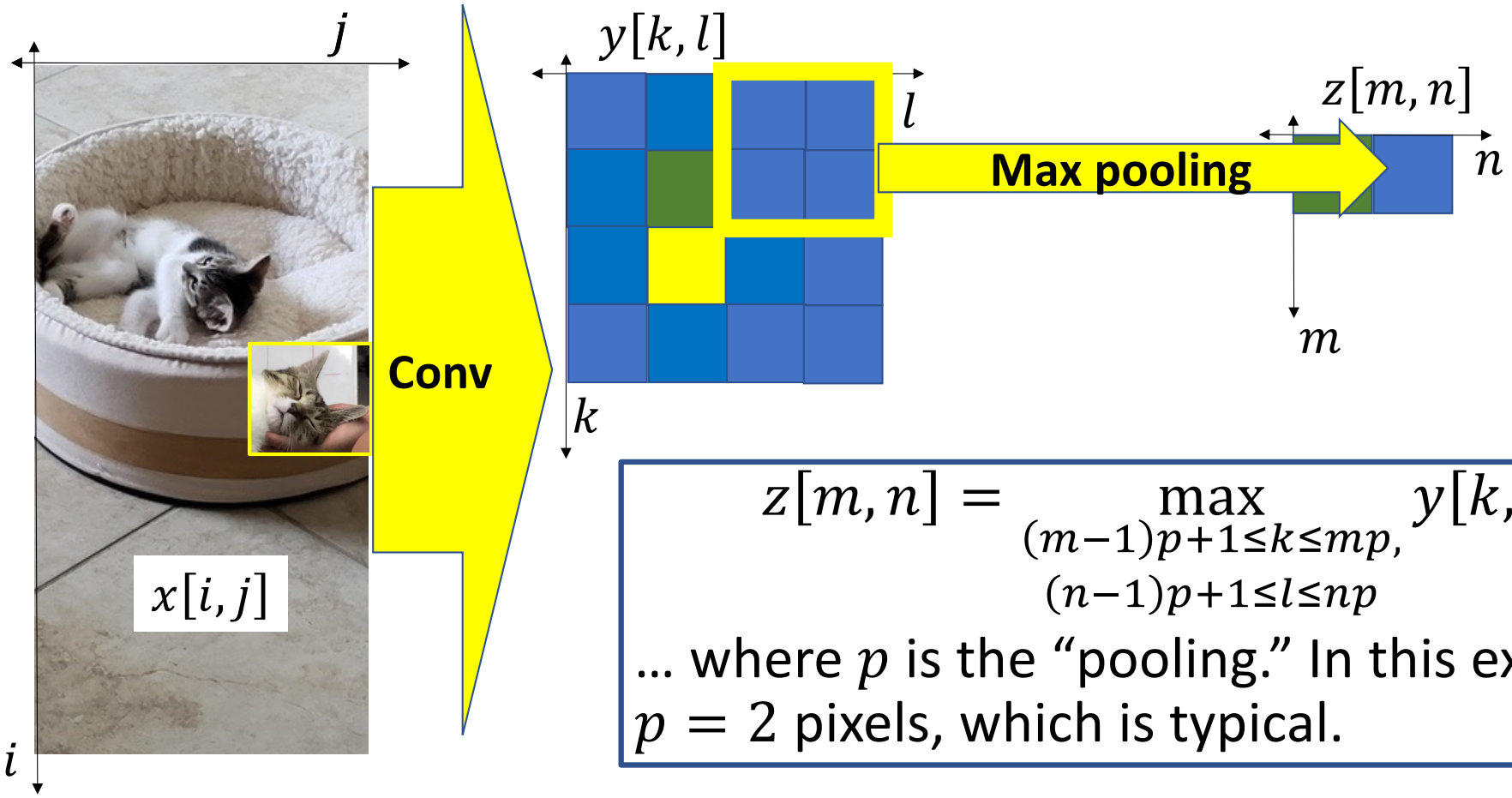
Max Pooling



$$z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$$

… where p is the "pooling." In this example, $p = 2$ pixels, which is typical.

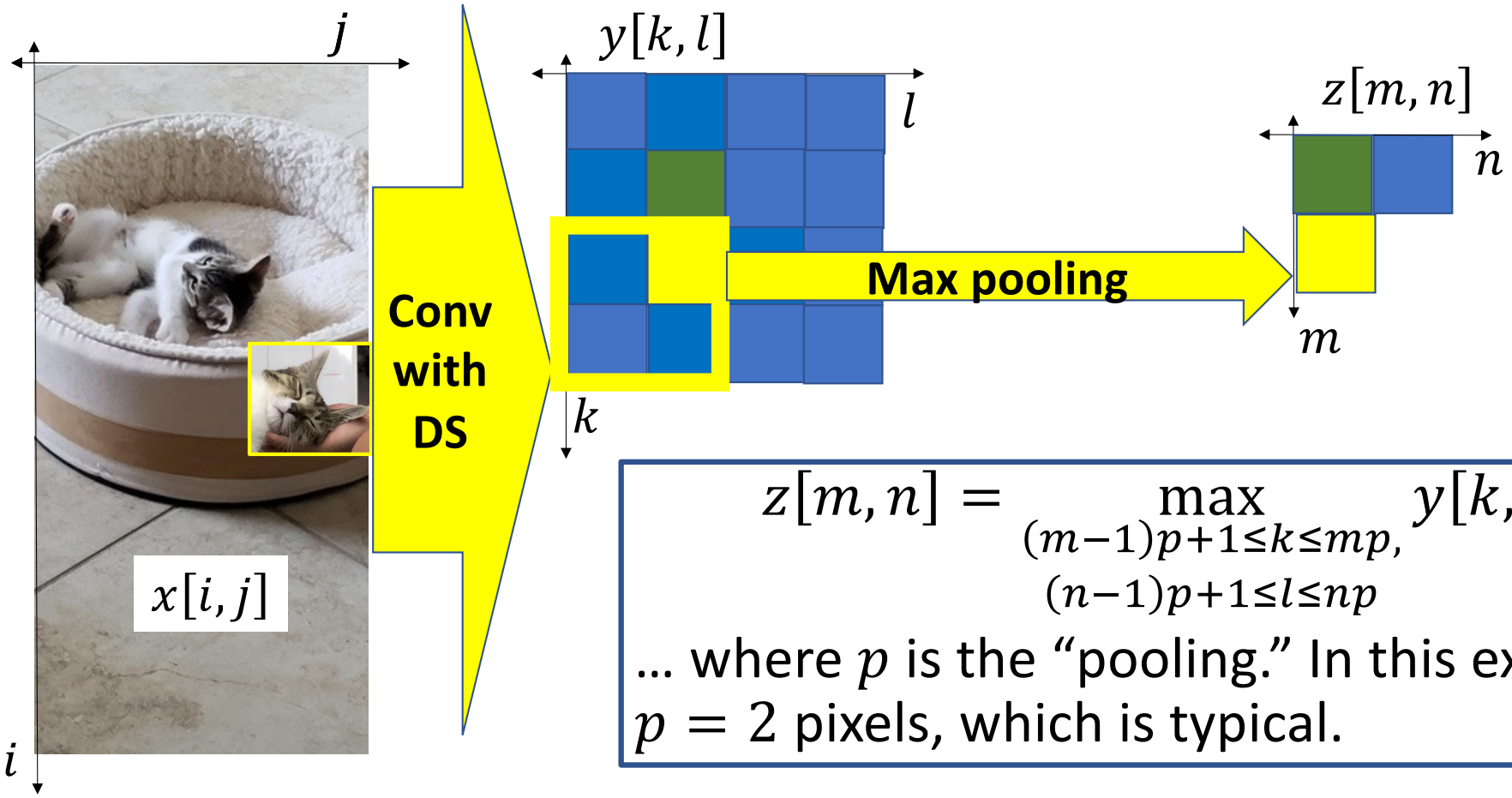
Max Pooling



$$z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$$

… where p is the "pooling." In this example, $p = 2$ pixels, which is typical.

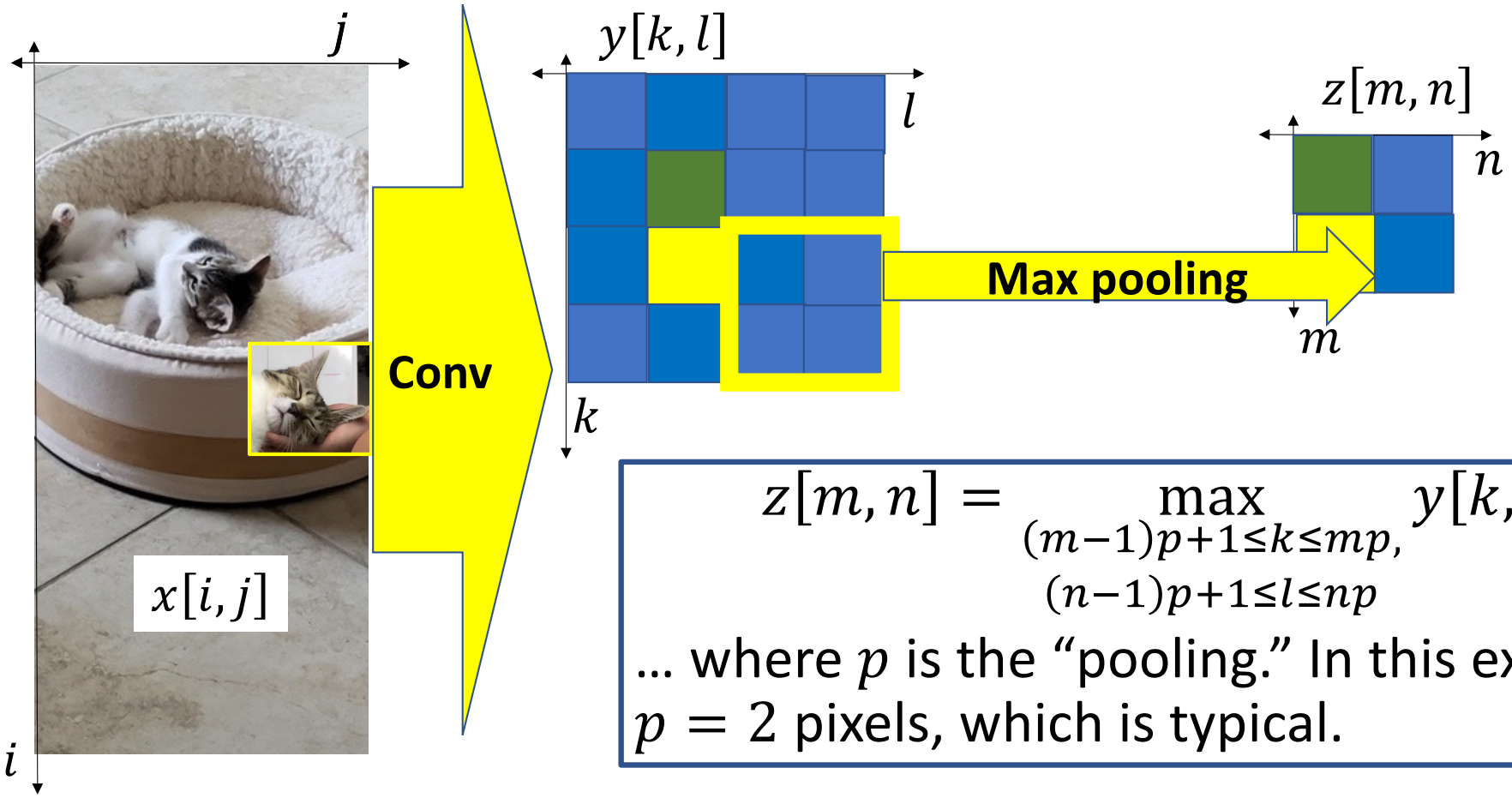
Max Pooling



$$z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$$

… where p is the "pooling." In this example, $p = 2$ pixels, which is typical.

Max Pooling

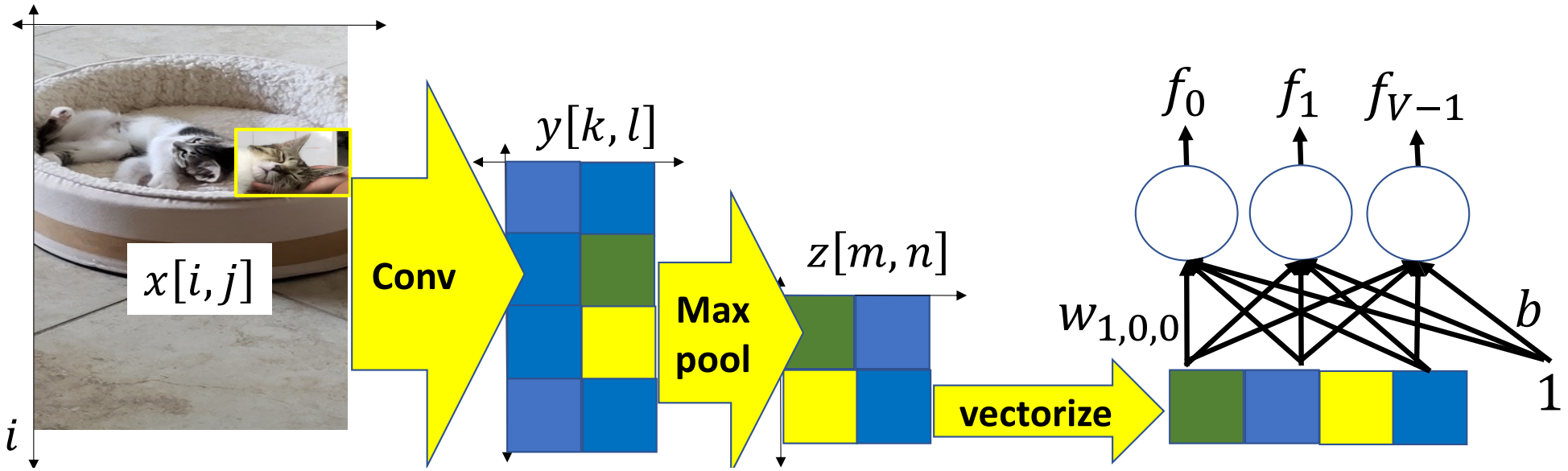


$$z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$$

… where p is the "pooling." In this example, $p = 2$ pixels, which is typical.

Object detection: convolution, max pooling, vectorize, then use a linear classifier

- Convolution detects an object, no matter where it is,
- Max pooling keeps the best detection of the object in each region,
- Linear classifier tests if the best detection is good enough.



Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through max pooling
- Real-World CNNs

Back-prop through max-pooling

- Suppose we've computed $z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$
- Now suppose we know $\frac{d\mathcal{L}}{dz[m,n]}$
- In order to train the neural net, we need $\frac{d\mathcal{L}}{dy[k,l]}$
- How do we find $\frac{d\mathcal{L}}{dy[k,l]}$?

Back-prop through max-pooling

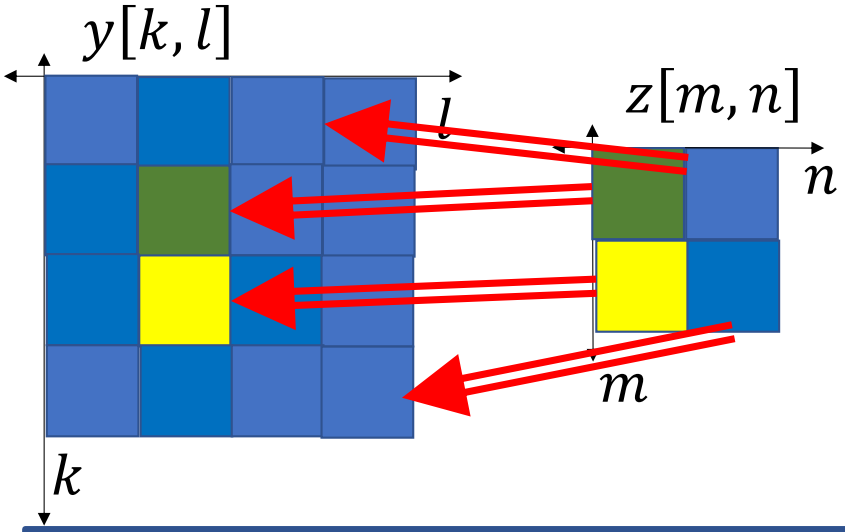
Answer: use the chain rule!

$z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$, so…

$$\frac{d\mathcal{L}}{dy[k,l]} = \frac{d\mathcal{L}}{dz[m,n]} \frac{dz[m,n]}{dy[k,l]}$$

$$= \begin{cases} \dfrac{d\mathcal{L}}{dz[m,n]} & \text{if } y[k,l] = \max\limits_{\substack{(m-1)p+1 \le i \le mp, \\ (n-1)p+1 \le j \le np}} y[i,j] \\ 0 & \text{otherwise} \end{cases}$$

Back-prop through max-pooling



$$\frac{d\mathcal{L}}{dy[k,l]} = \begin{cases} \frac{d\mathcal{L}}{dz[m,n]} & \text{if } y[k,l] = \max_{\substack{(m-1)p+1 \le i \le mp, \\ (n-1)p+1 \le j \le np}} y[i,j] \\ 0 & \text{otherwise} \end{cases}$$

In every 2x2 block of $y[k,l]$, only one pixel has nonzero $\frac{d\mathcal{L}}{dy[k,l]}$

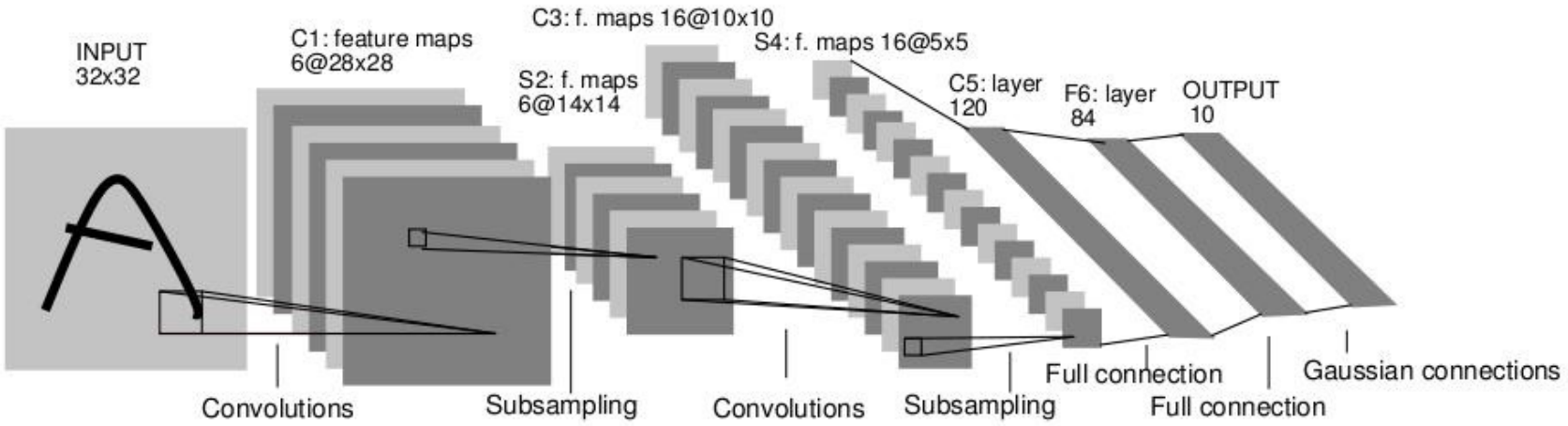
All the other pixels have zero gradient.

Computer Vision: Object Recognition

- Review: Neural Networks
- Object recognition: the problem of shift-invariance
- Correlation = local similarity judgements
- Convolution = Correlation that has been flipped upside down and backward
- Back-propagation through convolution
- Object detection: Max Pooling & Vectorization
- Back-propagation through convolution
- Real-World CNNs

LeNet, 1998

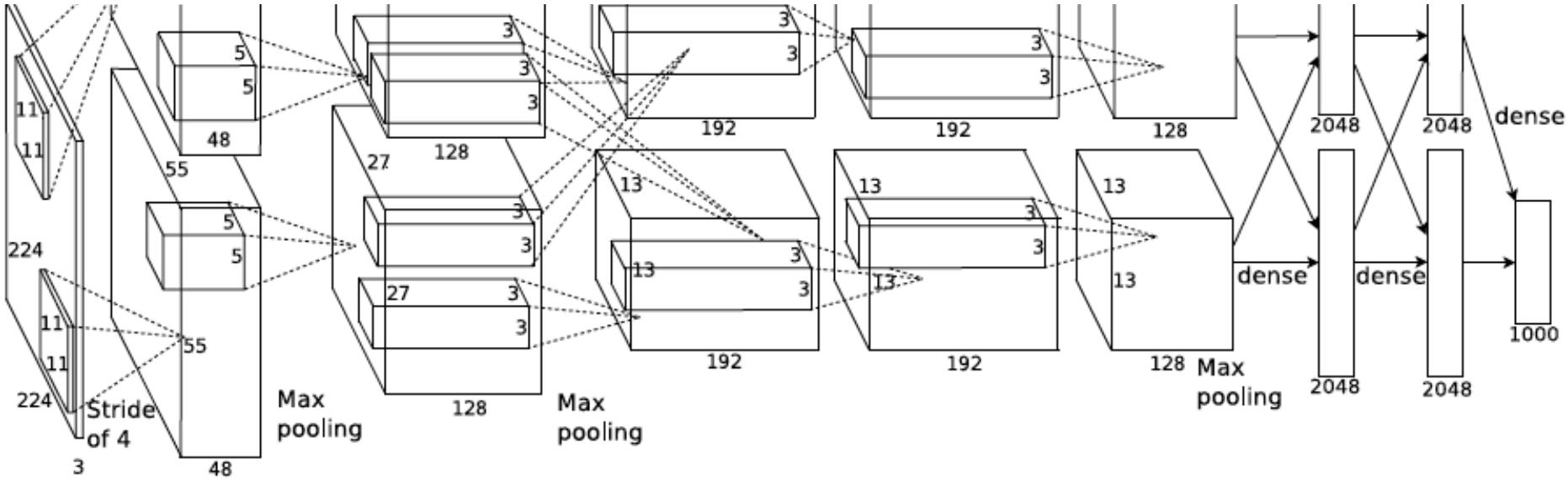
- (Convolution, Nonlinearity, Max-pooling, Normalization) X 2 layers
- After the second subsampling, the resulting features are concatenated to form a vector, which is then classified.



Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86(11): 2278–2324, 1998.

AlexNet

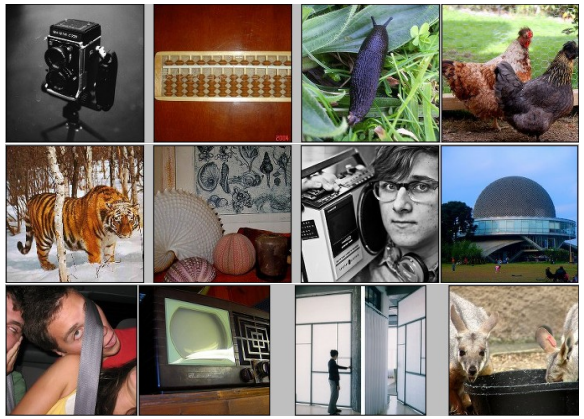
- Similar framework to LeNet but:
  - Bigger model (7 hidden layers, 650,000 units, 60,000,000 params)
  - More data (10^6 vs. 10^3 images)
  - GPU implementation (50x speedup over CPU)
    - Trained on two GPUs for a week



A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

ImageNet Challenge

IMAGENET

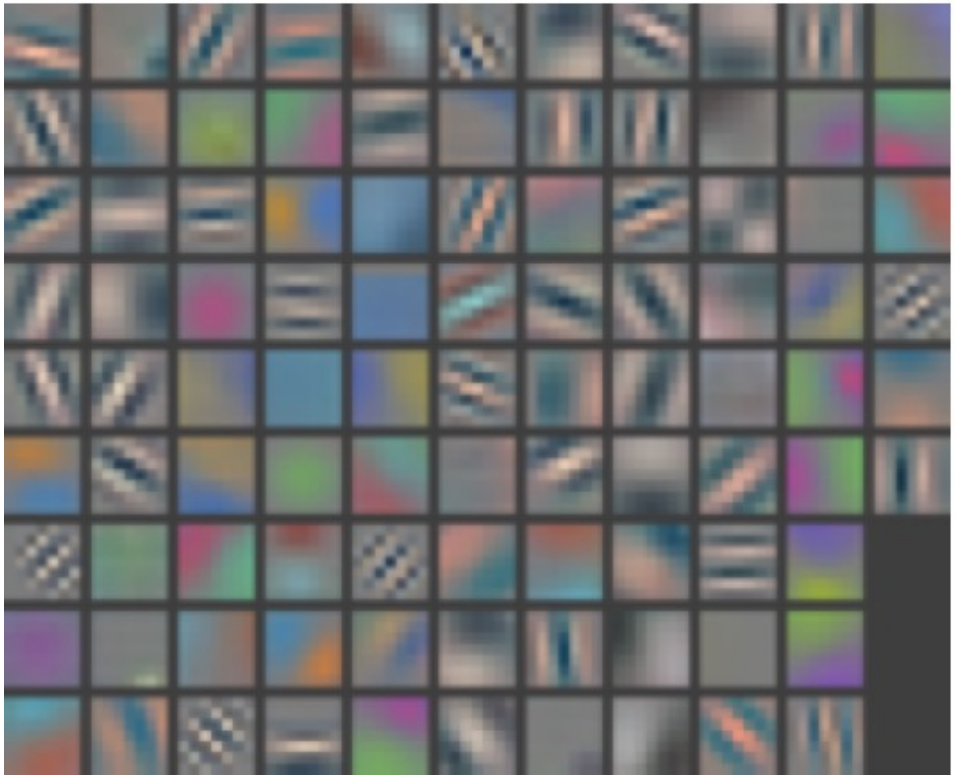


[Deng et al. CVPR 2009]

- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- Challenge: 1.2 million training images, 1000 classes

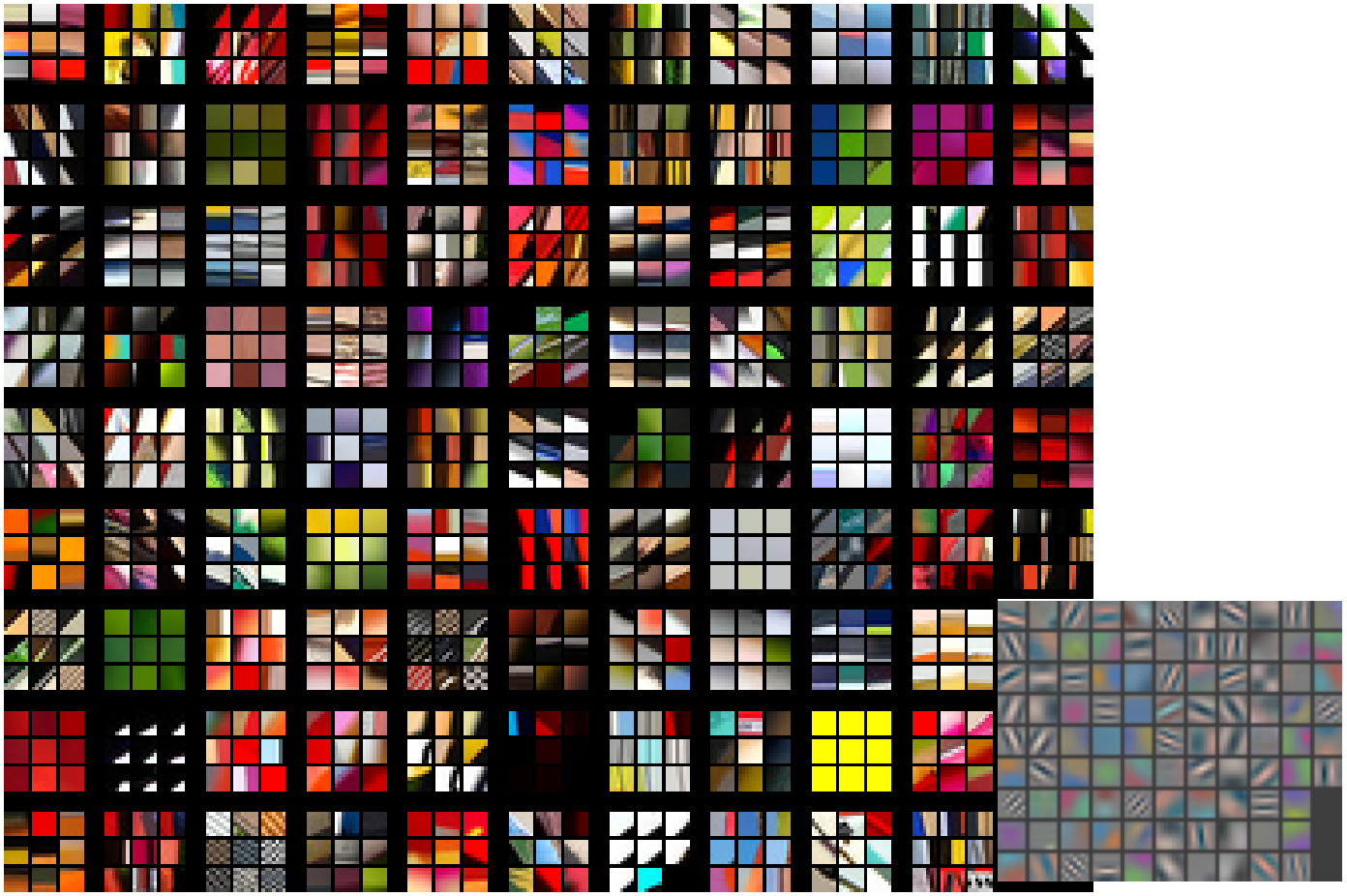
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Layer 1 Filters



M. Zeiler and R. Fergus, Visualizing and Understanding Convolutional Networks, arXiv preprint, 2013

Layer 1: Top-9 Patches



Layer 2: Top-9 Patches

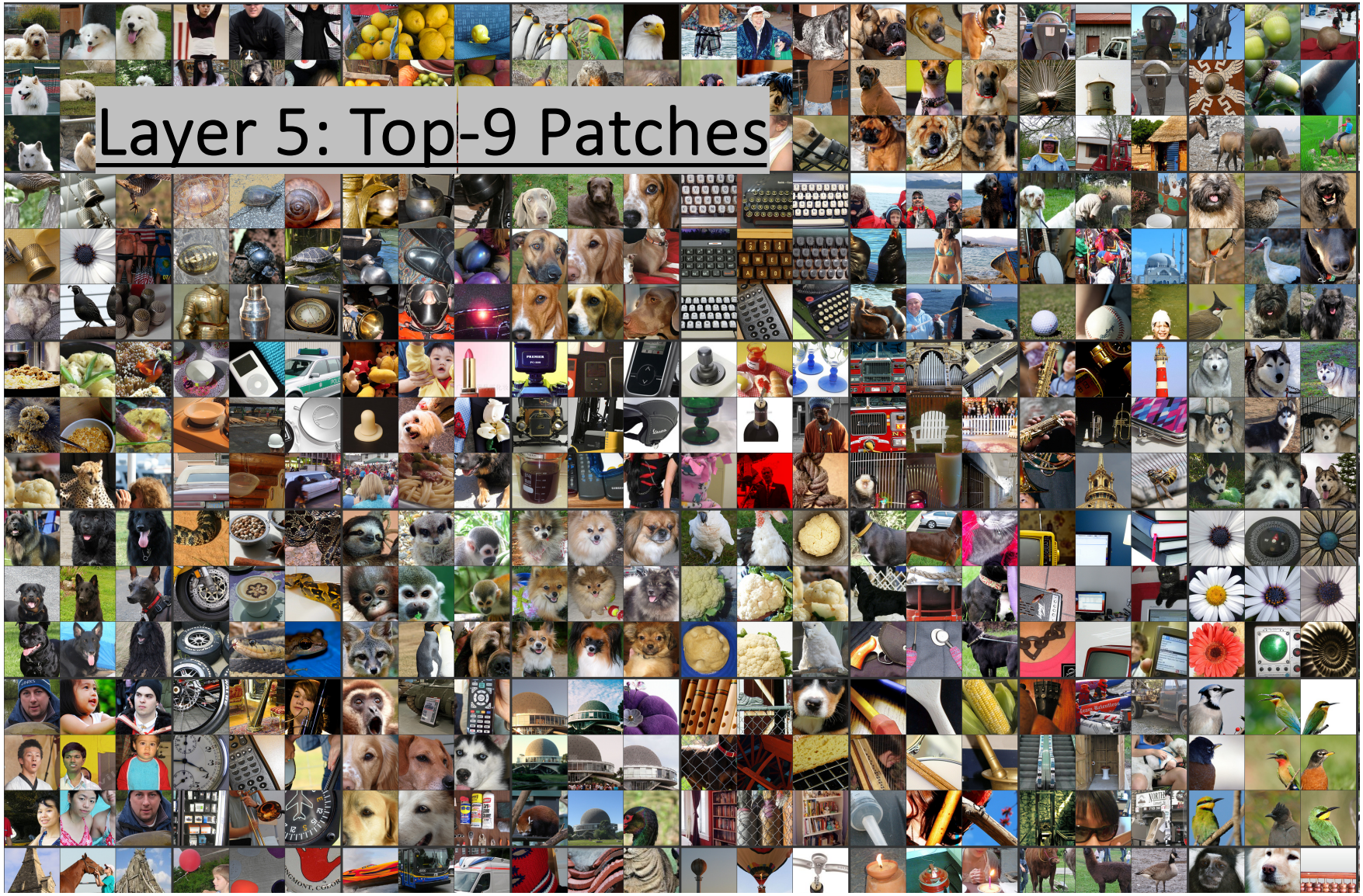


- Patches from validation images that give maximal activation of a given feature map

Layer 3: Top-9 Patches




Layer 4: Top-9 Patches


Layer 5: Top-9 Patches

Conclusion: Convolution and Max Pooling

$$y[k,l] = w[k,l] * x[k,l] = \sum_i \sum_j w[k-i, l-j] x[i,j] = \sum_i \sum_j w[i,j] x[k-i, l-j]$$

$$\frac{d\mathcal{L}}{dw[i,j]} = \sum_k \sum_l \frac{d\mathcal{L}}{dy[k,l]} \frac{dy[k,l]}{dw[i,j]}$$

$$z[m,n] = \max_{\substack{(m-1)p+1 \le k \le mp, \\ (n-1)p+1 \le l \le np}} y[k,l]$$

$$\frac{d\mathcal{L}}{dy[k,l]} = \begin{cases} \dfrac{d\mathcal{L}}{dz[m,n]} & \text{if } y[k,l] = \max\limits_{\substack{(m-1)p+1 \le i \le mp, \\ (n-1)p+1 \le j \le np}} y[i,j] \\ 0 & \text{otherwise} \end{cases}$$