CS 760: Machine Learning
Neural Networks III

Josiah Hanna

University of Wisconsin-Madison

October 12, 2023

Announcements

- Homework 4 is out (due after midterm)
- Midterm: **next week**
- **My office hours are just Tuesdays or by appointment**

Outline

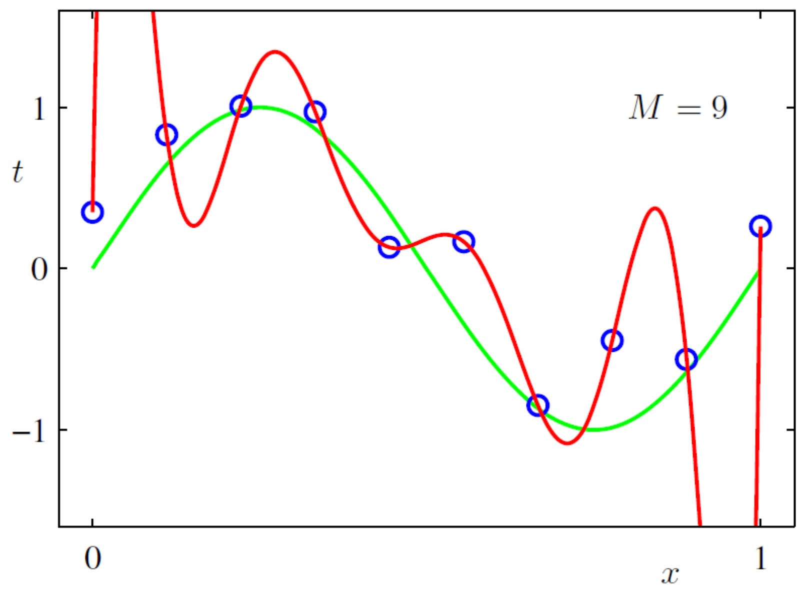
- **Regularization**
  - Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks** (maybe next lecture)

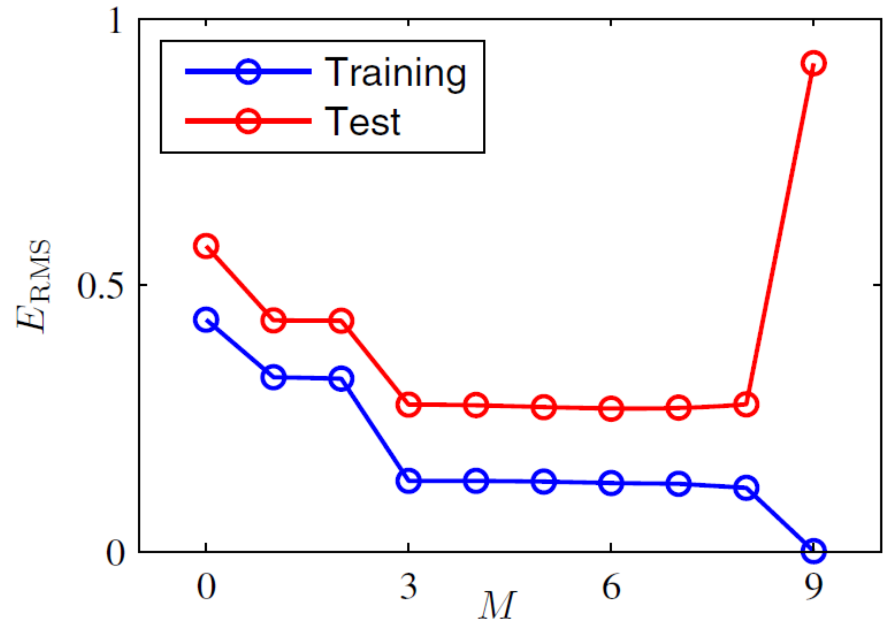
Outline

- **Regularization**
  - Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks** (maybe next lecture)

Review: Overfitting

- What is it? When empirical loss and expected loss are different
- Possible solutions:
  - Larger data set
  - Throwing away useless hypotheses also helps (**regularization**)





Review: Regularization

- In general: any method to **prevent overfitting**

- One approach: modify the optimization objective

- Different “views”
  - Hard constraint,
  - Soft constraint,
  - Bayesian view

Regularization: Hard Constraint View

- Training objective / parametrized version

$$\min_f \hat{L}(f) = \frac{1}{n}\sum_{i=1}^{n} l(f, x_i, y_i)$$

subject to: $f \in \mathcal{H}$

$$\min_\theta \hat{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(\theta, x_i, y_i)$$

subject to: $\theta \in \Omega$

- Constrain $\theta*$ beyond it's natural choice

$$\min_\theta \hat{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(\theta, x_i, y_i)$$

subject to: $R(\theta) \leq r$

$$\min_\theta \hat{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(\theta, x_i, y_i)$$

subject to: $||\theta||_2^2 \leq r^2$

L2 Regularization

Regularization: Soft Constraint View

- Equivalent to, for some parameter $\lambda^* > 0$

$$\min_{\theta} \hat{L}_R(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(\theta, x_i, y_i) + \lambda^* R(\theta)$$

- For L2,

$$\min_{\theta} \hat{L}_R(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(\theta, x_i, y_i) + \lambda^* ||\theta||_2^2$$

- Comes from **Lagrangian duality**

Regularization: Bayesian Prior View

- Recall our MAP version of training. Bayes law:

$$p(\theta|\{(x_i, y_i)\}_{i=1}^n) = \frac{p(\{(x_i, y_i)\}_{i=1}^n|\theta)p(\theta)}{p(\{(x_i, y_i)\}_{i=1}^n)}$$

- MAP (assuming iid data):

$$\begin{aligned}\hat{\theta}_{\text{MAP}} &= \arg\max_{\theta} p(\theta|\{(x_i, y_i)\}_{i=1}^n) \\ &= \arg\max_{\theta} \left(\underbrace{\log(p(\theta))}_{\text{Regularization}} + \underbrace{\sum_{i=1}^n \log p(x_i, y_i|\theta)}_{\text{MLE}} \right)\end{aligned}$$

Choice of View?

- Typical choice for optimization: soft-constraint

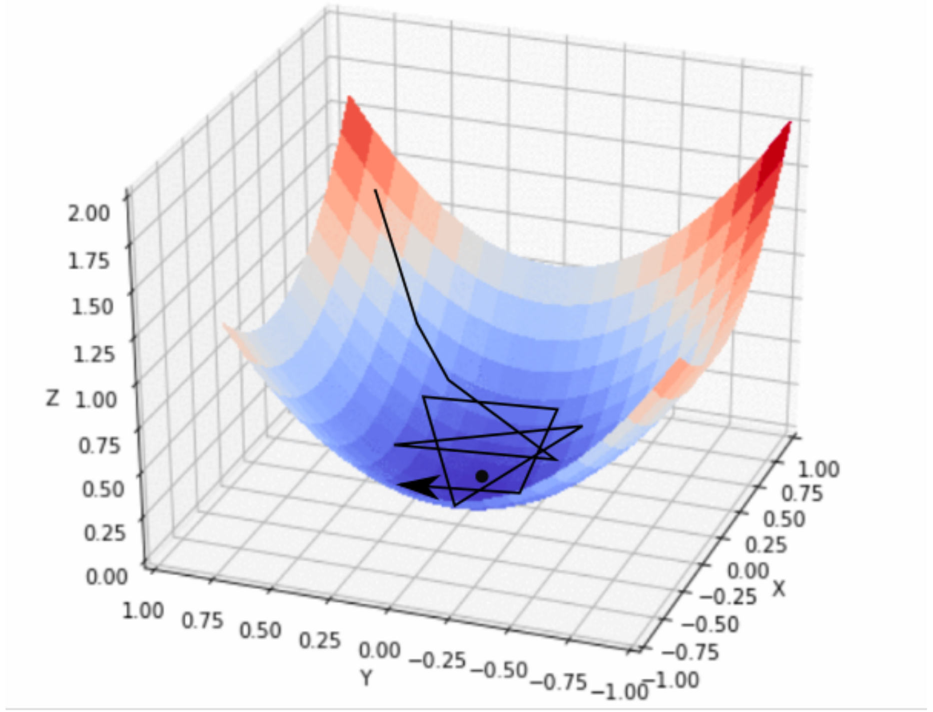
$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \lambda R(\theta)$$

- Hard constraint / Bayesian view: conceptual / for derivation
- Hard-constraint preferred if know the explicit bound
- Bayesian view preferred if domain knowledge easy to represent as a prior

Examples: L2 Regularization

$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \frac{\lambda}{2}||\theta||_2^2$$

- Questions: what are the
  - Effects on (stochastic) gradient descent?
  - Effects on the optimal solution?



L2 Regularization: **Effect on GD**

- Gradient of regularized objective

$$\nabla \hat{L}_R(\theta) = \nabla \hat{L}(\theta) + \lambda\theta$$

- Gradient descent update

$$\theta \leftarrow \theta - \eta \nabla \hat{L}_R(\theta) = \theta - \eta\, \nabla \hat{L}(\theta) - \eta\lambda\theta$$

$$= (1 - \eta\lambda)\theta - \eta\, \nabla \hat{L}(\theta)$$

- In words, **weight decay**

L2 Regularization: **Effect on Optimal Solution**

- Consider a quadratic approximation around $\theta*$ the optimum for the unregularized loss.

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + (\theta - \theta^*)^T \nabla \hat{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H(\theta - \theta^*)$$

Here, H is the hessian at $\theta*$

- Since $\theta*$ is optimal,

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + \frac{1}{2}(\theta - \theta^*)^T H(\theta - \theta^*)$$

$$\nabla \hat{L}(\theta) \approx H(\theta - \theta^*)$$

L2 Regularization: **Effect on Optimal Solution**

- Gradient of regularized objective: $\nabla \hat{L}_R(\theta) \approx H(\theta - \theta^*) + \lambda\theta$

- On the optimal θ_R^*: $0 = \nabla \hat{L}_R(\theta_R^*) \approx H(\theta_R^* - \theta^*) + \lambda\theta_R^*$

$$\theta_R^* \approx (H + \lambda I)^{-1} H \theta^*$$

- H has eigendecomp. $H = Q\Lambda Q^T$, assume $(\Lambda + \lambda I)^{-1}$ exists:

$$\theta_R^* \approx (H + \lambda I)^{-1} H \theta^* = Q(\Lambda + \lambda I)^{-1} \Lambda Q^T \theta^*$$

Effect: **shrink along eigenvectors of** ***H***

Break & Quiz

Q: Which of the following statement(s) is(are) TRUE about regularization parameter λ ?

A. λ is the tuning parameter that decides how much we want to penalize the flexibility of our model.

B. λ is usually set using cross validation.

1. True, True
2. True, False
3. False, True
4. False, False

Q: Which of the following statement(s) is(are) TRUE about regularization parameter λ ?

A. λ is the tuning parameter that decides how much we want to penalize the flexibility of our model.

B. λ is usually set using cross validation.

1. [✓ True, True]
2. True, False
3. False, True
4. False, False

- The optimization problem can be viewed as following:

$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \lambda\,\text{complexity}(\text{Model}))$$

- If the regularization parameter is large then it requires a small model complexity
- We have learned how to use cross validation to set hyperparameters including regularization parameters.

Q: Select the correct option about regression with L2 regularization (also called *Ridge Regression*).

A. Ridge regression technique prevents coefficients from rising too high.

B. As $\lambda \to \infty$, the impact of the penalty grows, and the ridge regression coefficient estimates will approach infinity.

1. Both statements are true.
2. Both statements are false.
3. Statement A is true, Statement B is false.
4. Statement B is true, Statement A is false.

Q: Select the correct option about regression with L2 regularization (also called *Ridge Regression*).

A. *Ridge regression technique prevents coefficients from rising too high.*

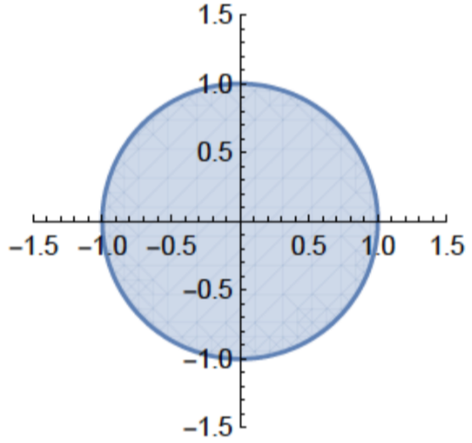
B. *As λ→∞, the impact of the penalty grows, and the ridge regression coefficient estimates will approach infinity.*

1. Both statements are true.
2. Both statements are false.
3. [✓ Statement A is true, Statement B is false.]
4. Statement B is true, Statement A is false.

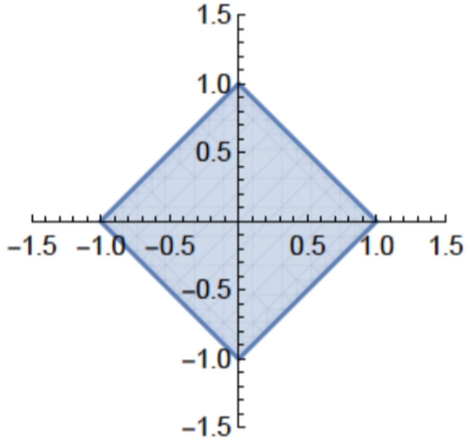
As λ→∞, the impact of the penalty grows, and the ridge regression coefficient estimates will approach zero.

Q: Following figure shows 3-norm sketches: $||x||_p<1$ for p = 1, 2, ∞. Recall that $||x||_\infty = \max\{|x_i| \text{ for all } i\}$

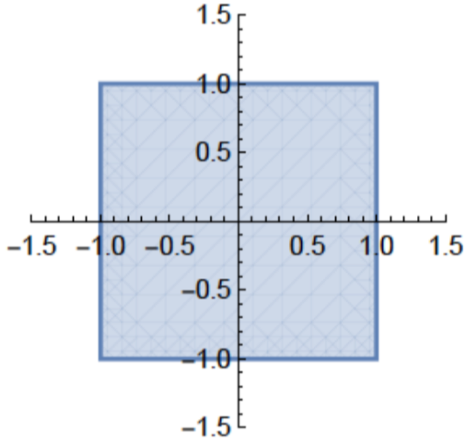
A:



B:



C:

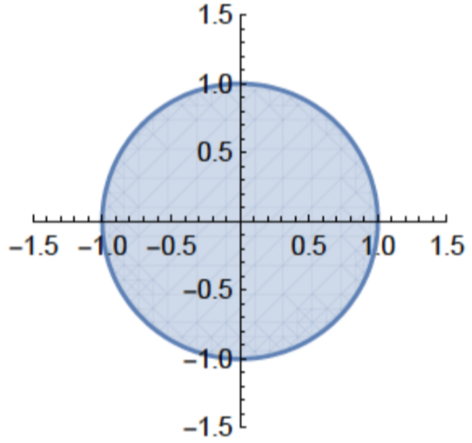


1. A: p=1, B: p=2, C: p=∞
2. A: p=2, B: p=1, C: p=∞
3. A: p=2, B: p=∞, C: p=1
4. A: p=∞, B: p=2, C: p=1

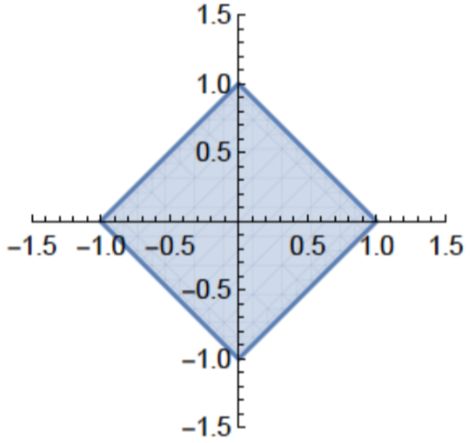
Q: Following figure shows 3-norm sketches: $||x||_p < 1$ for $p = 1, 2, \infty$.

Recall that $||x||_\infty = \max\{|x_i| \text{ for all } i\}$

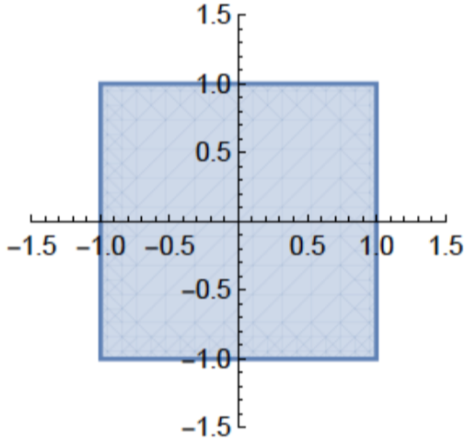
A:



B:



C:



1. A: p=1, B: p=2, C: p=∞
2. [✓ A: p=2, B: p=1, C: p=∞]
3. A: p=2, B: p=∞, C: p=1
4. A: p=∞, B: p=2, C: p=1

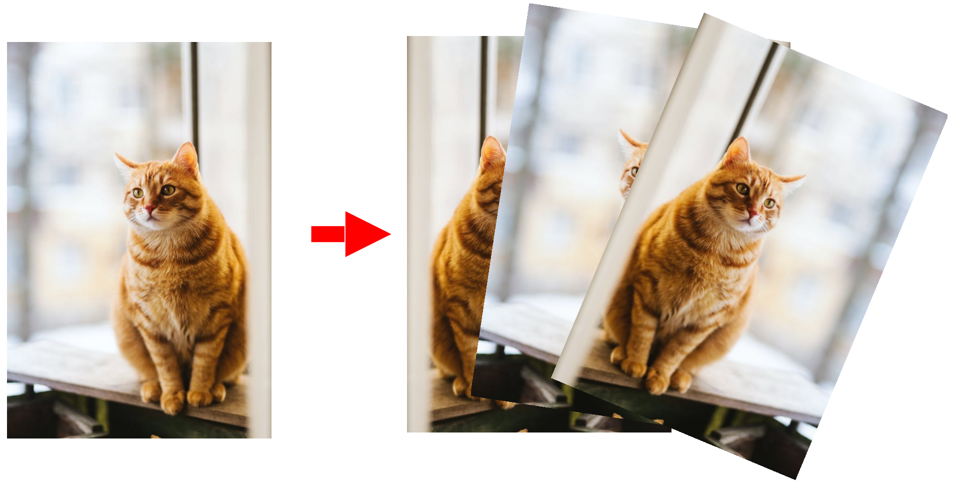
Outline

- **Review & Regularization**
  - Forward/backwards Pass, Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks (next lecture)**

Data Augmentation

Augmentation: transform + add new samples to dataset

- Transformations: based on domain
- Idea: build **invariances** into the model
  - **Ex**: if all images have same alignment, model learns to use it
- Keep the label the same!



Data Augmentation: Examples

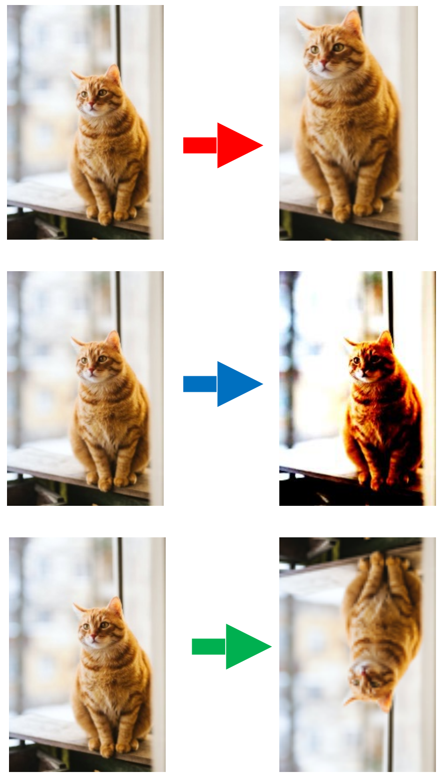
Examples of transformations for images

- **Crop** (and zoom)
- **Color** (change contrast/brightness)
- **Rotations+** (translate, stretch, shear, etc)

Many more possibilities. Combine as well!

Q: how to deal with this at **test time**?

- A: transform, test, average



Combining & Automating Transformations

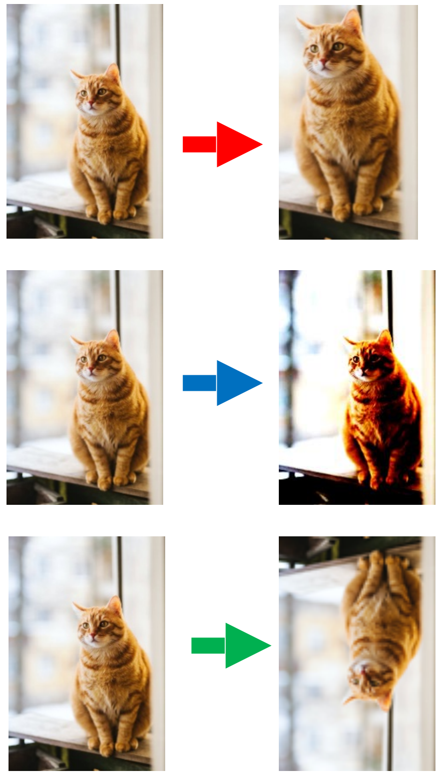
One way to automate the process:

- Apply every transformation and combinations
- **Downside:** most don't help…

Want a good policy, ie, → → → → →

- Active area of research: search for good policies

1. **Ratner et al**: "Learning to Compose Domain-Specific Transformations for Data Augmentation"
2. **Cubuk et al**: "AutoAugment: Learning Augmentation Strategies from Data"



Data Augmentation: Other Domains

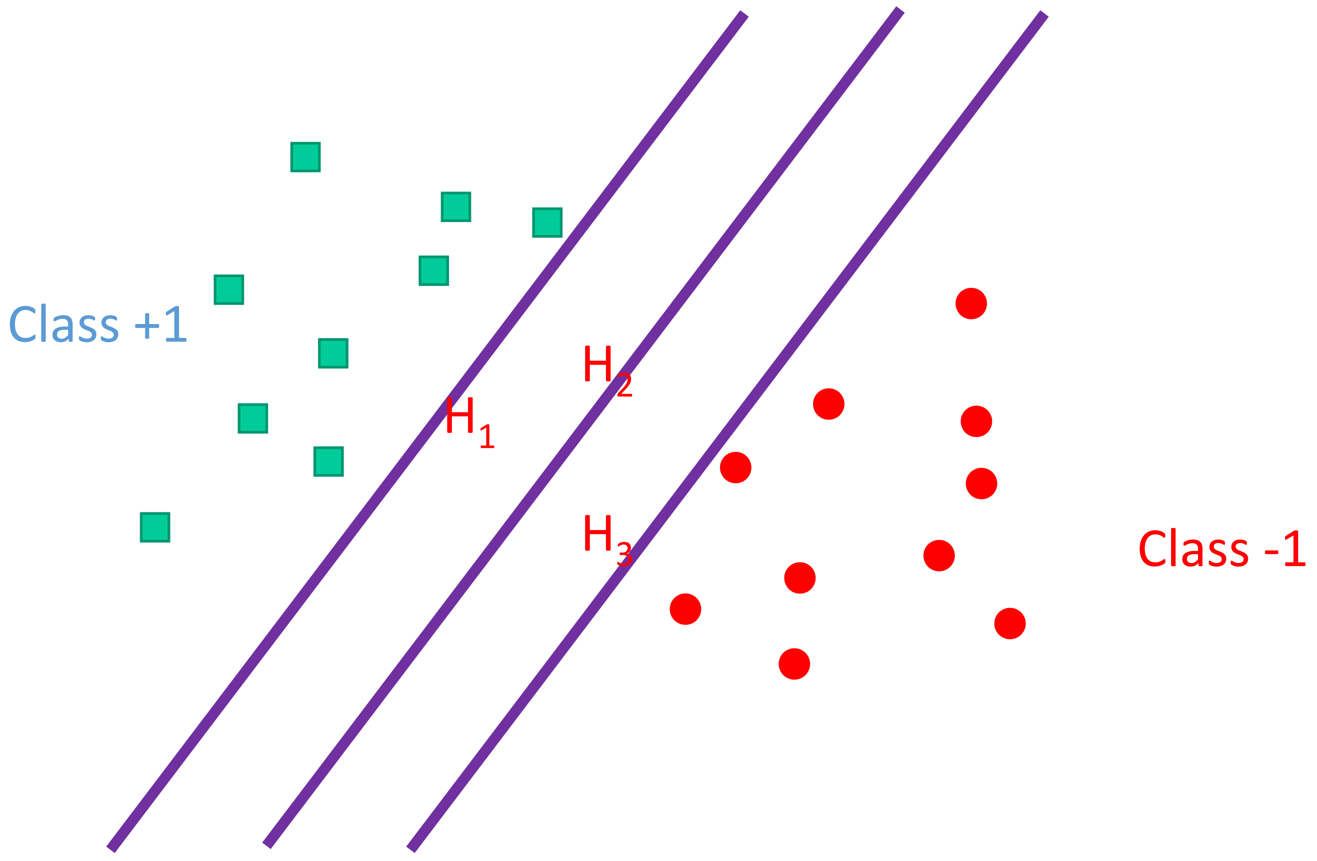
Not just for image data. For example, on text:

- Substitution
  - E.g., “It is a **great** day” $\rightarrow$ “It is a **wonderful** day”
  - Use a thesaurus for particular words
  - Or, use a model. Pre-trained word embeddings, language models
- Back-translation
  - “Given the low budget and production limitations, this movie is very good.” $\rightarrow$ “There are few budget items and production limitations to make this film a really good one”

Xie **et al**: “Unsupervised Data Augmentation for Consistency Training”

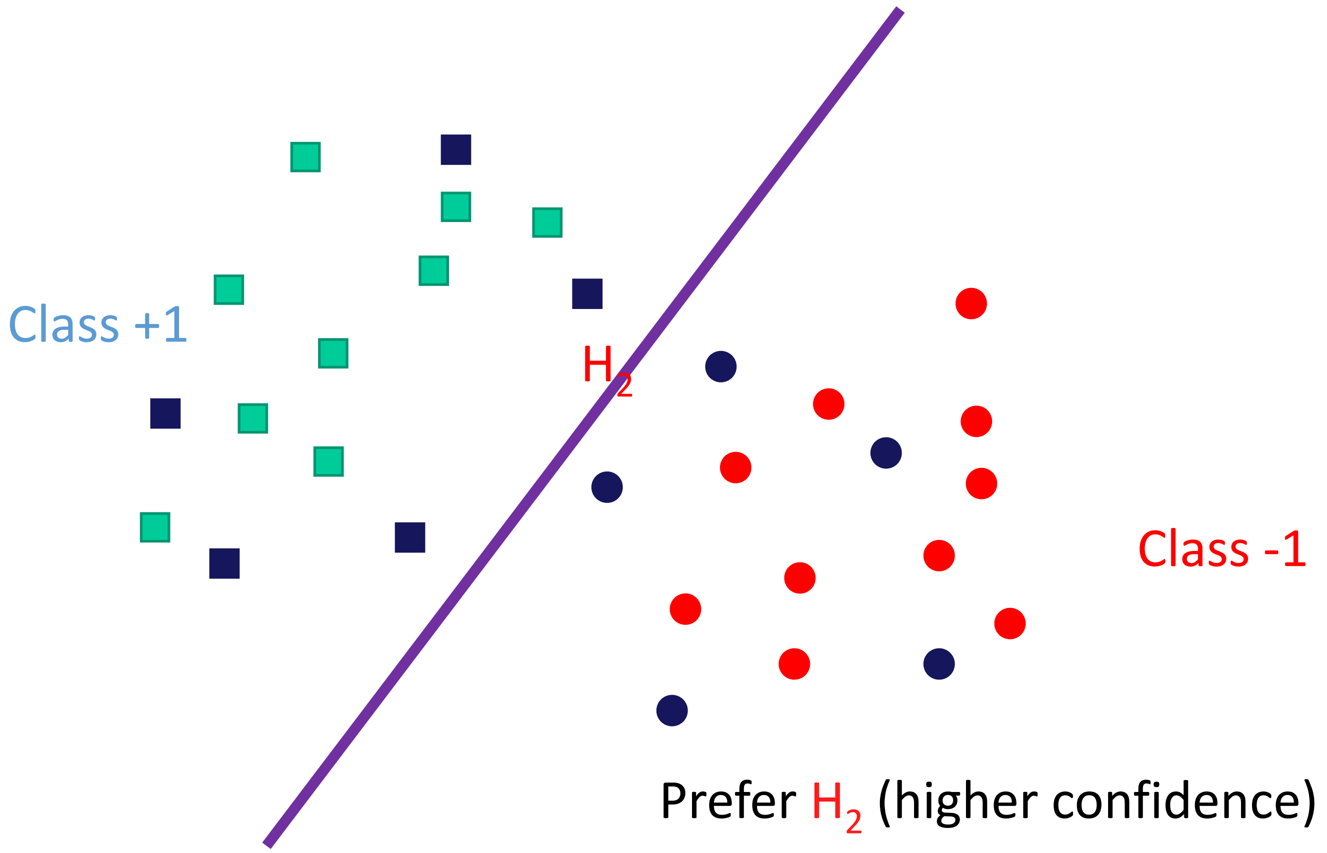
Adding Noise

- What if we have many solutions?



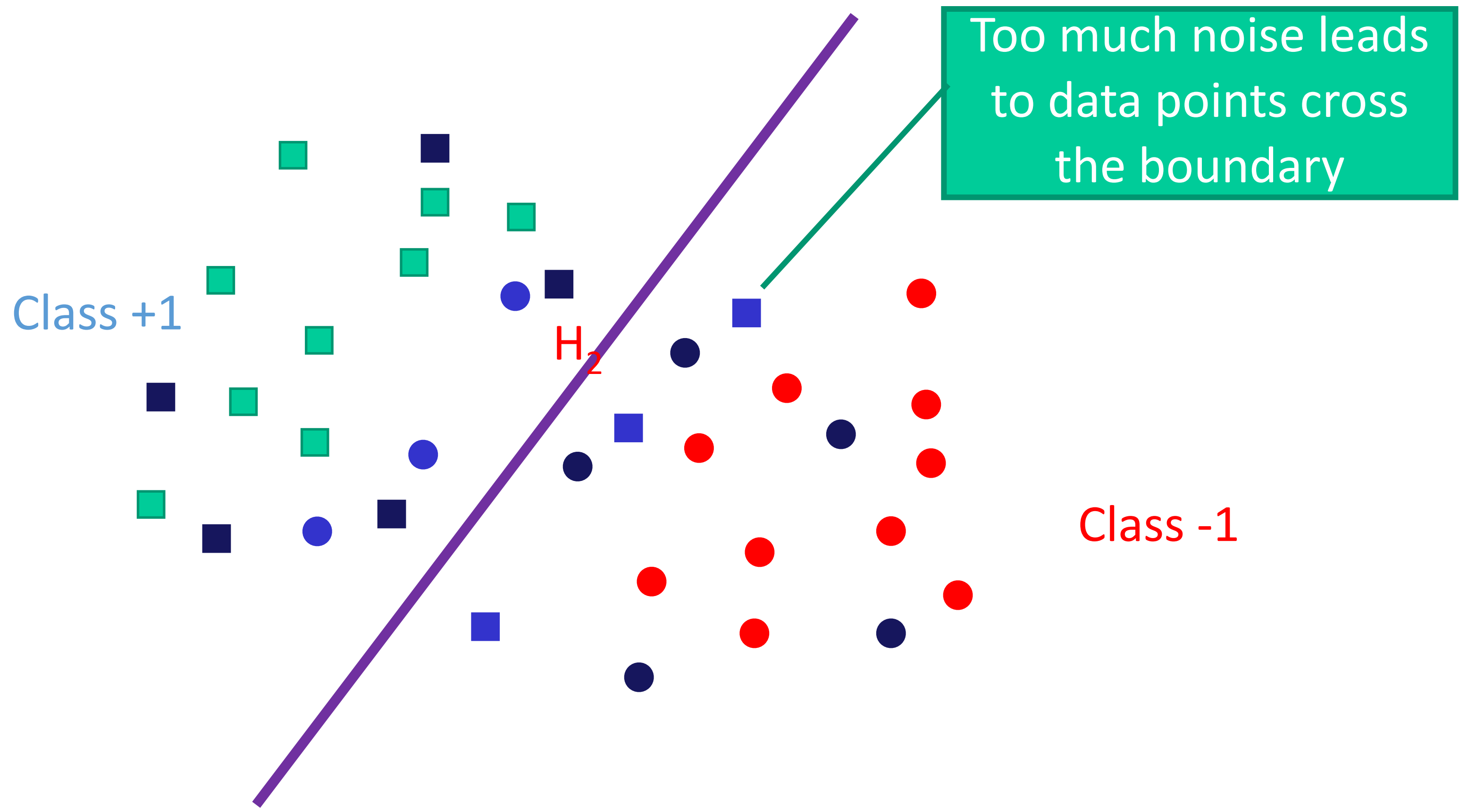
Adding Noise

- Adding some amount of noise helps us pick solution:



Adding Noise

- Too much: hurts instead



Adding Noise: Equivalence to Weight Decay

- Suppose the hypothesis is $f(x) = w^\top x$ noise is $\epsilon \sim N(0, \lambda I)$
- After adding noise, the loss is

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x+\epsilon) - y]^2 = \mathbb{E}_{x,y,\epsilon}[f(x) + w^T\epsilon - y]^2$$

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x) - y]^2 + 2\mathbb{E}_{x,y,\epsilon}[w^T\epsilon(f(x) - y)] + \mathbb{E}_{x,y,\epsilon}[w^T\epsilon]^2$$

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x) - y]^2 + \lambda||w||^2$$

Early Stopping

- **Idea**: don’t train the network to too small training error
  - Larger the hypothesis class, easier to find a hypothesis that fits the difference between the training and test set.
  - So: do not push the hypothesis too much; use validation error to decide when to stop

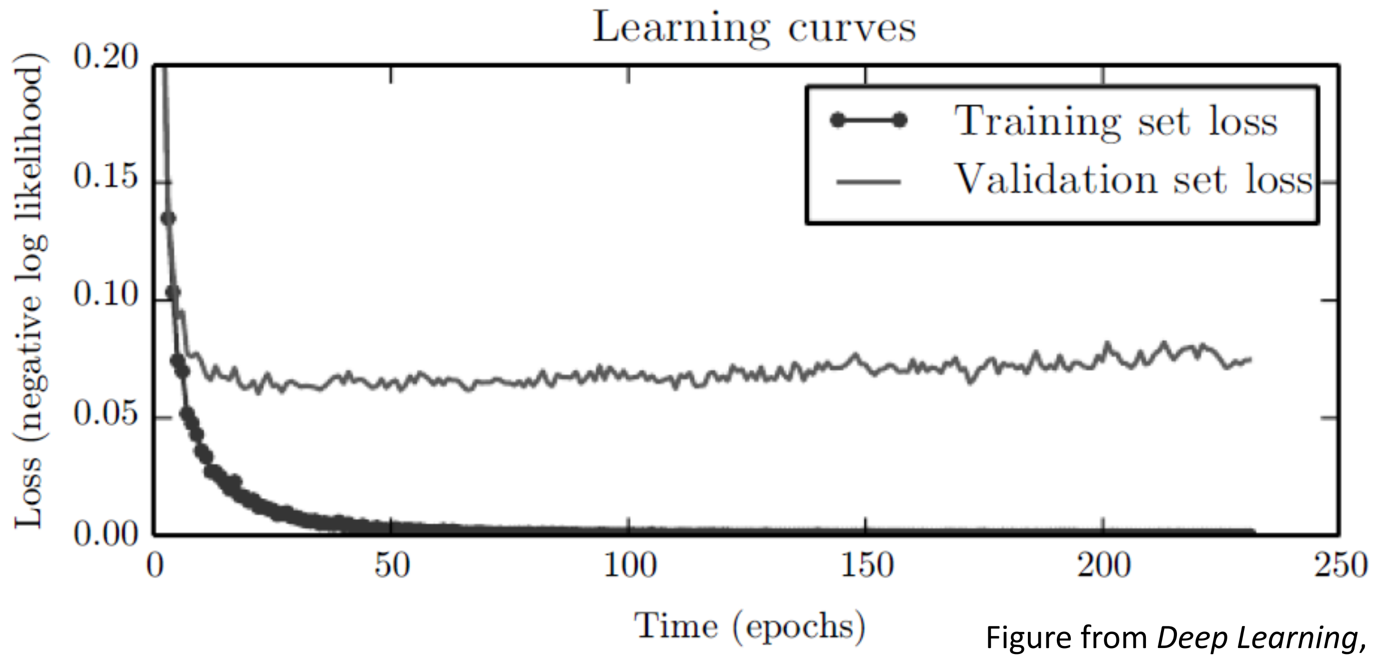


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Early Stopping

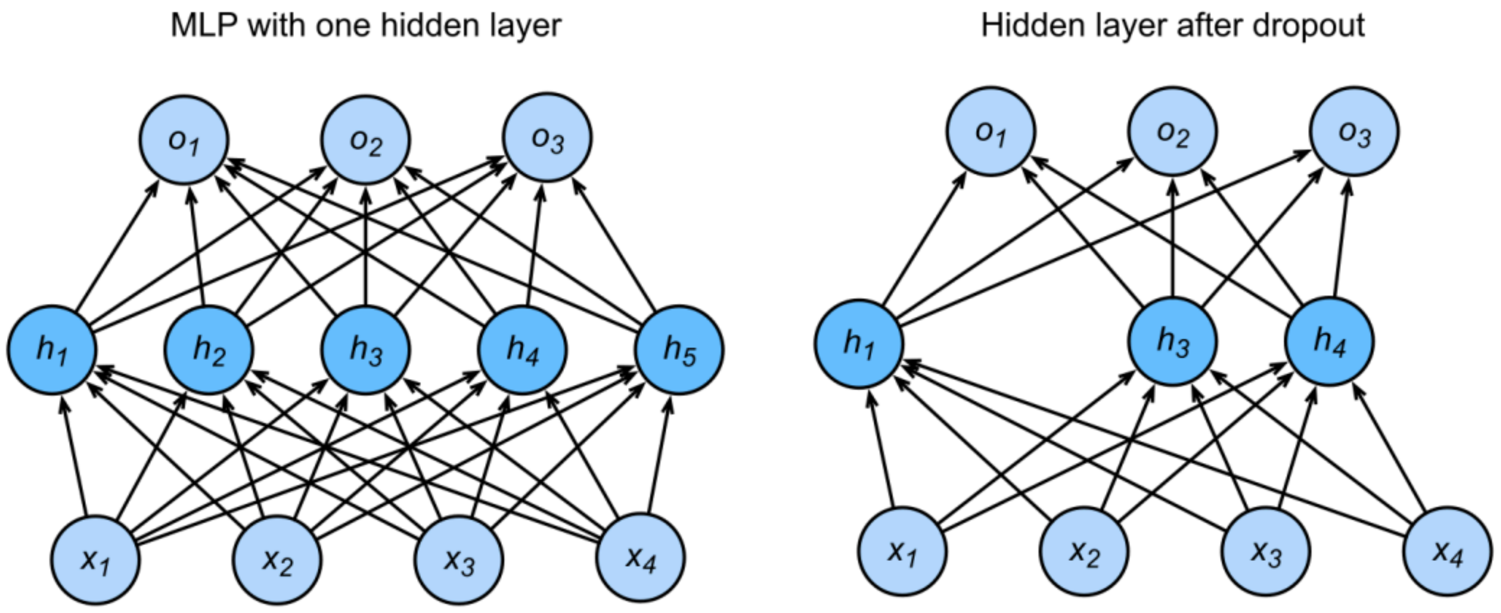
- Practically: when training, also compute validation error
  - Every time validation error improved, store a copy of the weights
  - When validation error not improved for some time, stop
  - Return the copy of the weights stored

Dropout

- **Basic idea**: randomly select weights to update
- In each update step
  - Randomly apply a binary mask to all the input and hidden units
  - Multiply the mask bits with the units and do the update as usual
- Typical dropout prob: 0.2 for input and 0.5 for hidden units

Applying Dropout

$$\mathbf{h} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$
$$\mathbf{h}' = \text{dropout}(\mathbf{h})$$
$$\mathbf{o} = \mathbf{W}^{(2)}\mathbf{h}' + \mathbf{b}^{(2)}$$
$$\mathbf{p} = \text{softmax}(\mathbf{o})$$



Applying Dropout

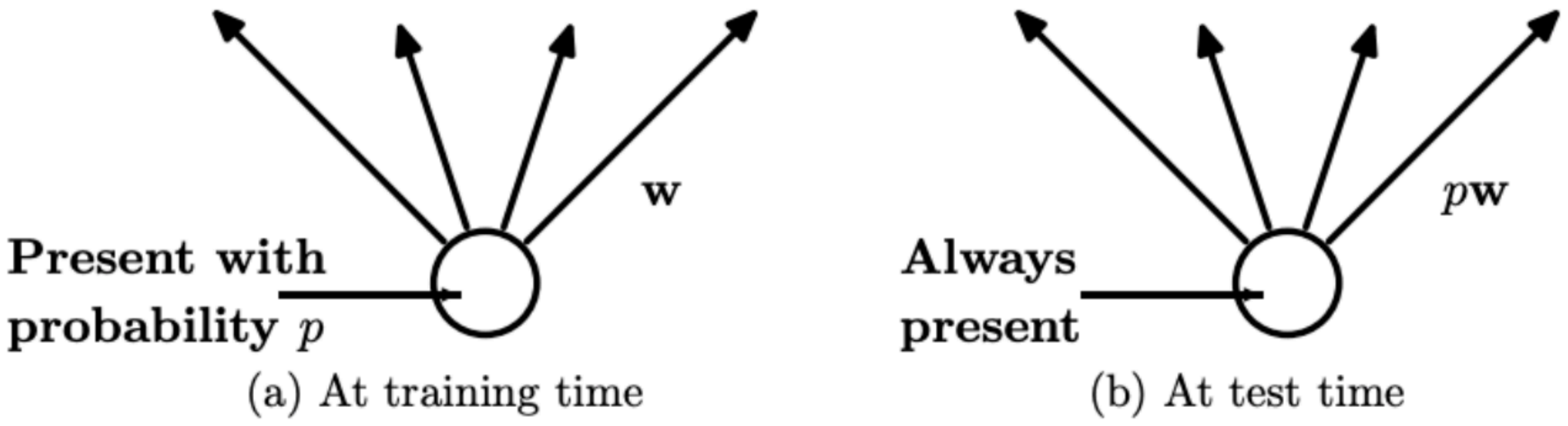


Figure 2: **Left**: A unit at training time that is present with probability p and is connected to units in the next layer with weights $\mathbf{w}$. **Right**: At test time, the unit is always present and the weights are multiplied by p. The output at test time is same as the expected output at training time.

Break & Quiz

Q2-2: Are these statements true or false?
(A) We can use validation data to decide when to stop early.
(B) We can think of early stopping as a regularization to limit the volume of parameter space reachable from the initial parameter.

1. True, True
2. True, False
3. False, True
4. False, False

Q2-2: Are these statements true or false?
(A) We can use validation data to decide when to stop early.
(B) We can think of early stopping as a regularization to limit the volume of parameter space reachable from the initial parameter.

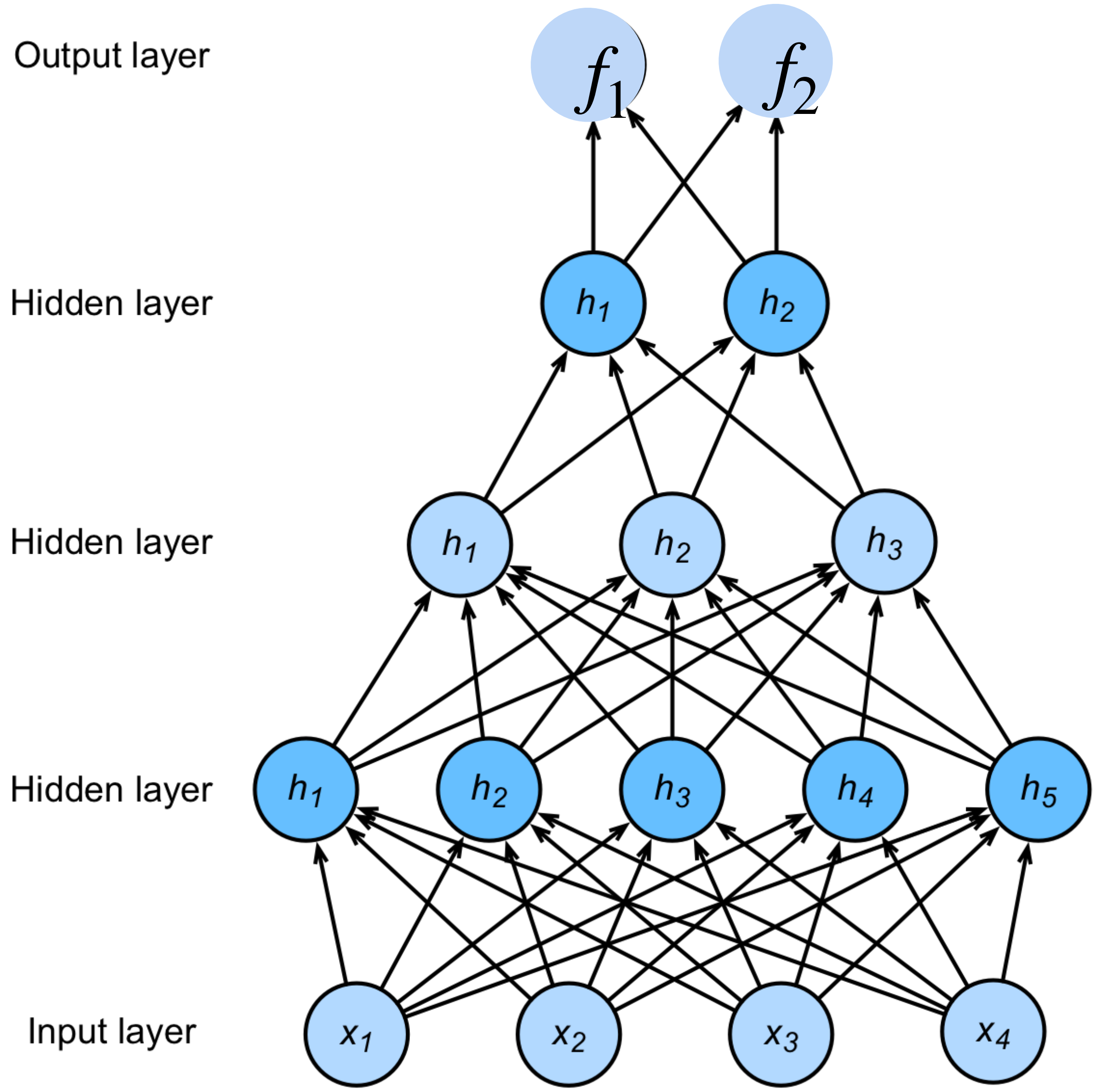
1. [✓ True, True]
2. True, False
3. False, True
4. False, False

(A) As is shown in the lecture.
(B) That’s true. Early stopping will limit the training time and thus potentially limit the space the training can search.

Outline

- **Regularization**
  - Views, L1/L2 Effects
- **Other Forms of Regularization**
  - Data Augmentation, Noise, Early Stopping, Dropout
- **Convolutional Neural Networks** (maybe next lecture)

Multi-layer perceptrons



$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{f} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$
$$\mathbf{y} = \text{softmax}(\mathbf{f})$$

NNs are composition of nonlinear functions

Classifying Images

How to classify

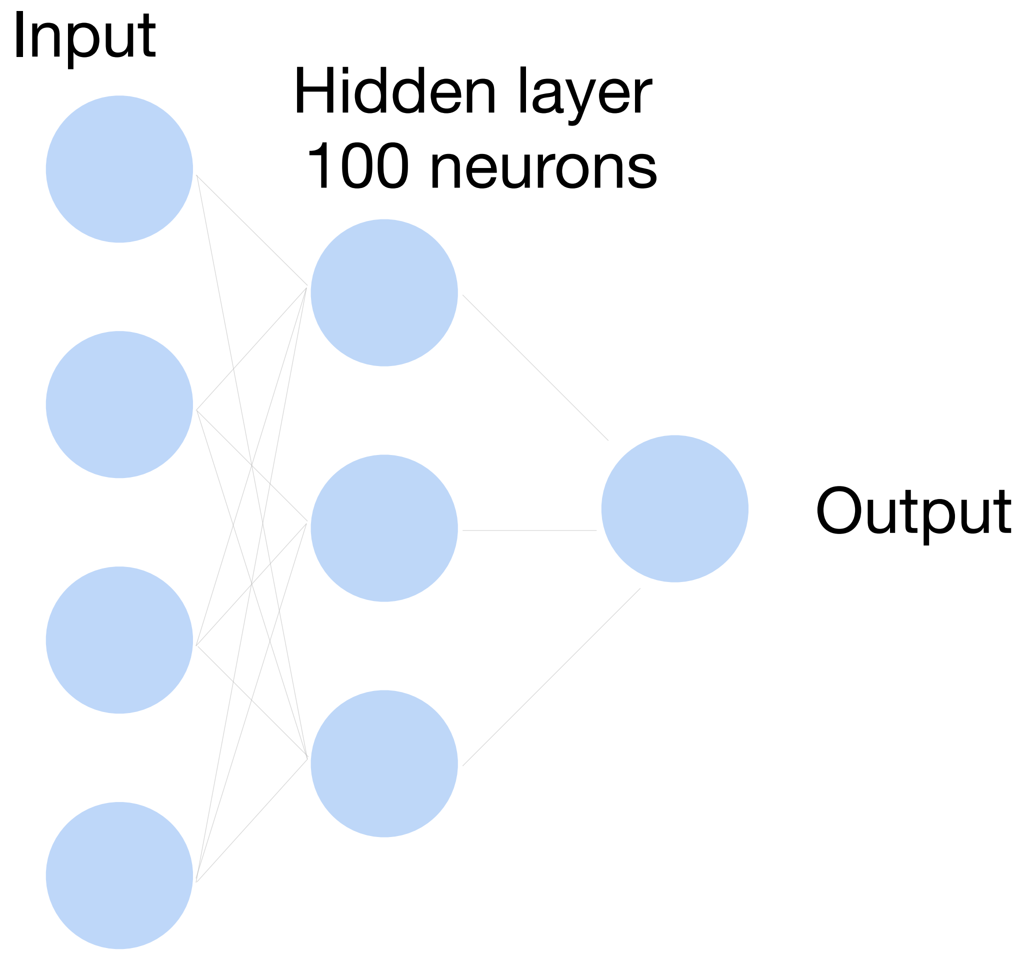
Cats vs. dogs?

Dual
12MP
wide-angle and
telephoto cameras

36M floats in a RGB image!

Classifying Images with fully connected NNs

Cats vs. dogs?



~ 36M elements x 100 = ~**3.6B** parameters!

Convolutions come to rescue!

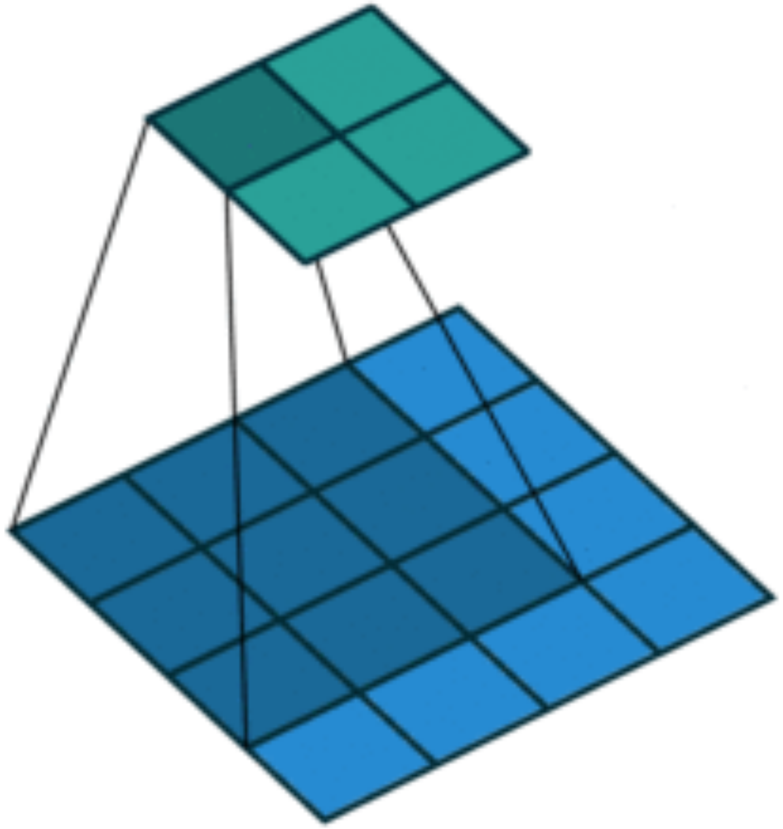
Why Convolution?

- Reduces number of parameters
- Translation Invariance
- Locality

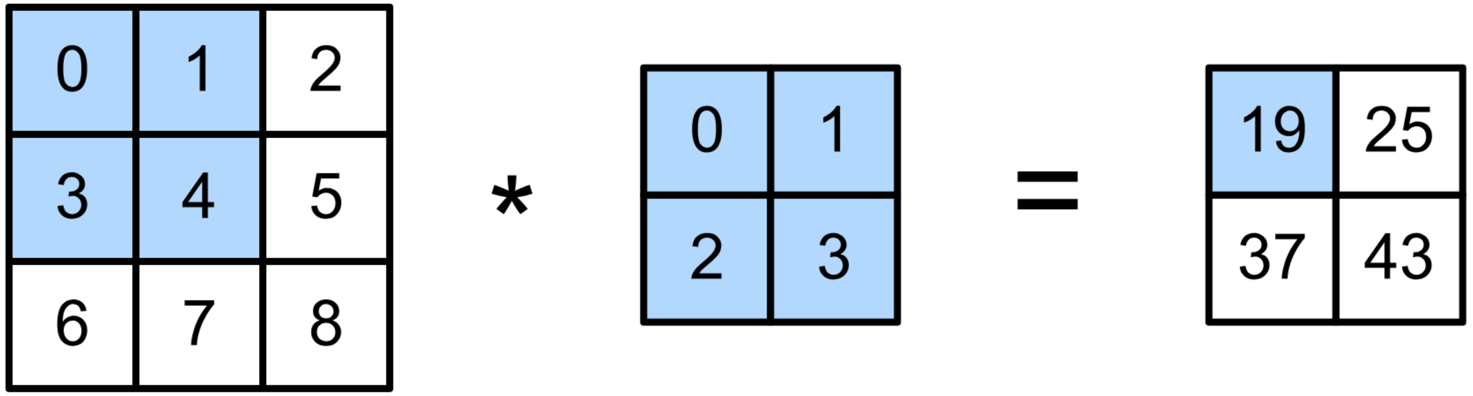
2-D Convolution

$$\begin{array}{ccccc} \text{Input} & & \text{Kernel} & & \text{Output} \\ \begin{array}{|c|c|c|}\hline 0 & 1 & 2 \\ \hline 3 & 4 & 5 \\ \hline 6 & 7 & 8 \\ \hline \end{array} & * & \begin{array}{|c|c|}\hline 0 & 1 \\ \hline 2 & 3 \\ \hline \end{array} & = & \begin{array}{|c|c|}\hline 19 & 25 \\ \hline 37 & 43 \\ \hline \end{array} \end{array}$$

$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$


(vdumoulin@ Github)

2-D Convolution Layer

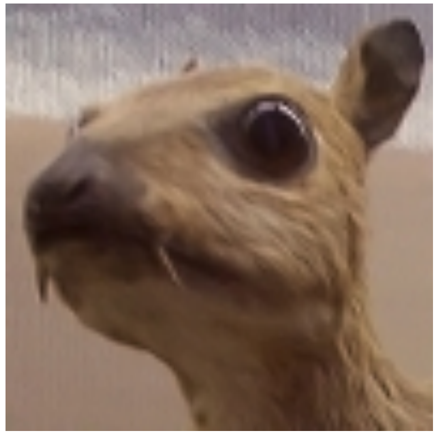


- $\mathbf{X} : n_h \times n_w$ input matrix
- $\mathbf{W} : k_h \times k_w$ kernel matrix
- b: scalar bias
- $\mathbf{Y} : (n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

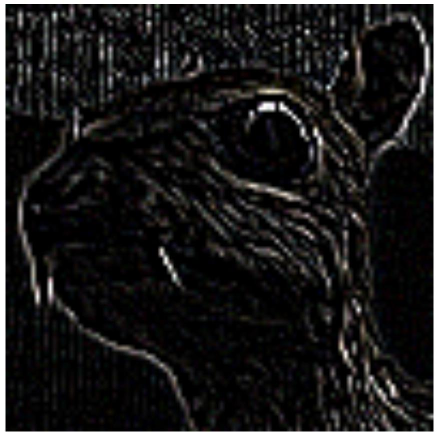
$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$

- **W** and *b* are learnable parameters

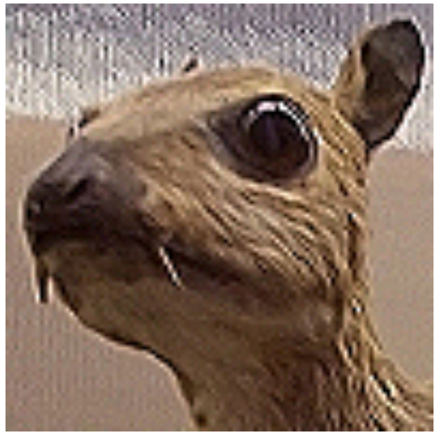
Examples


(wikipedia)

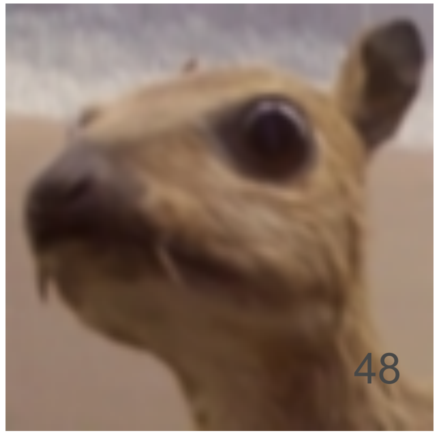
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$


Edge Detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

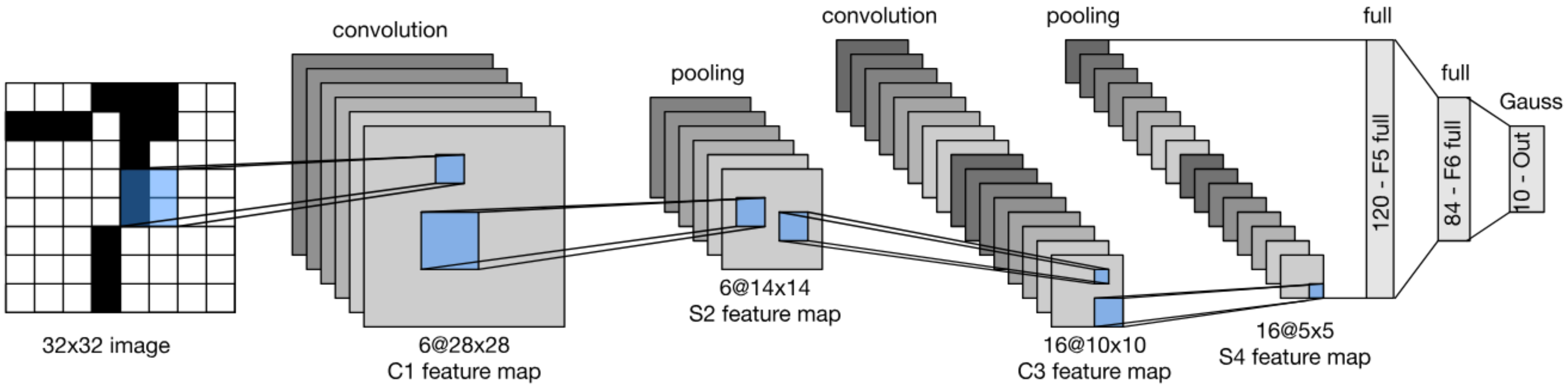

Sharpen

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$


Gaussian Blur

Convolutional Neural Networks

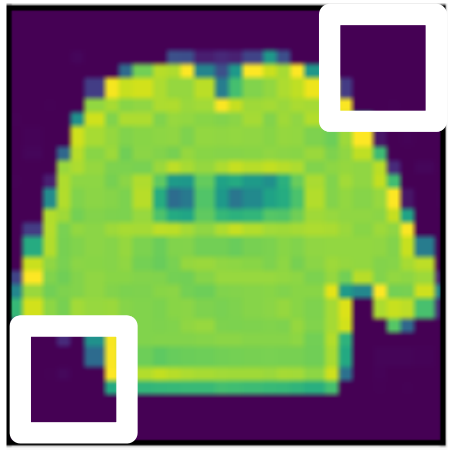
Convolutional networks: neural networks that use convolution in place of general matrix multiplication in at least one of their layers

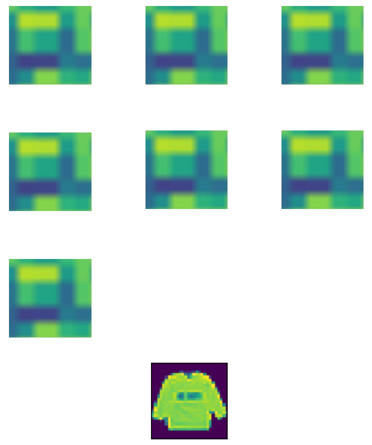


Padding

- Given a 32 x 32 input image
- Apply convolution with 5 x 5 kernel
  - 28 x 28 output with 1 layer
  - 4 x 4 output with 7 layers
- Shape decreases faster with larger kernels

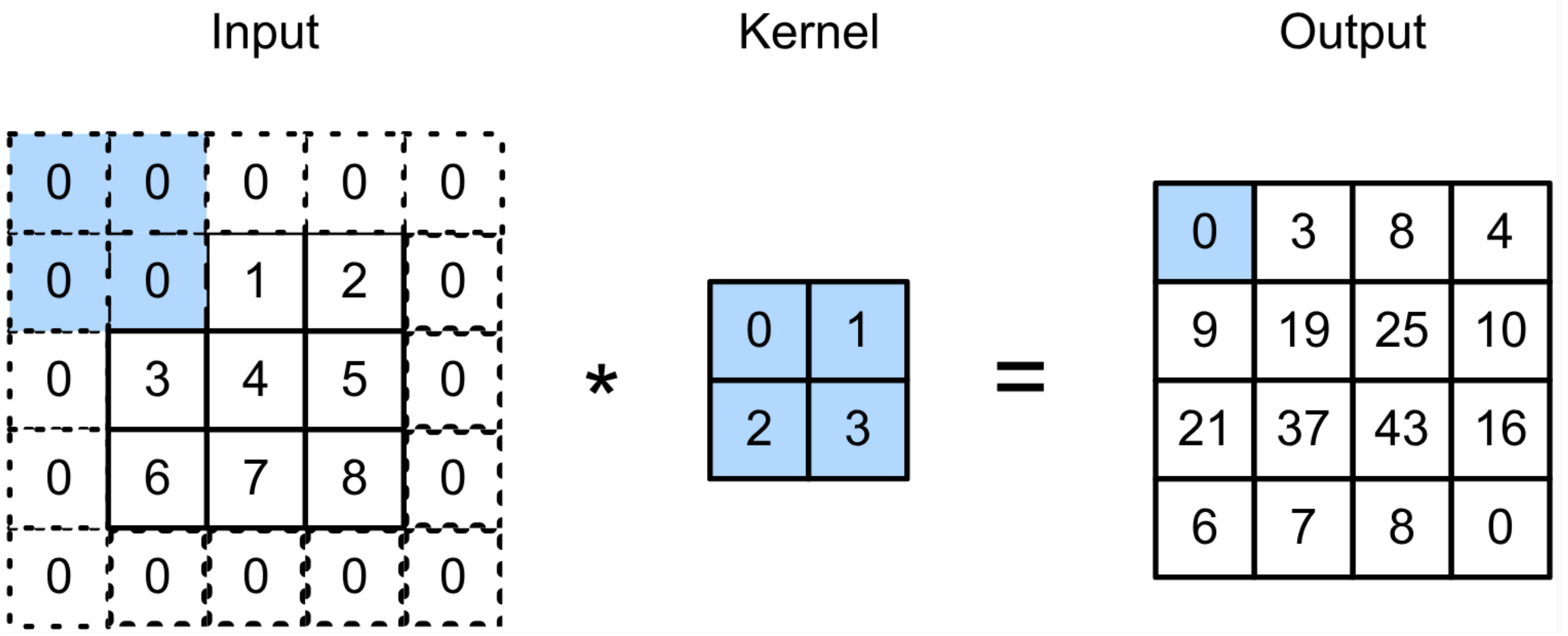
- Padding preserves **edge information**!

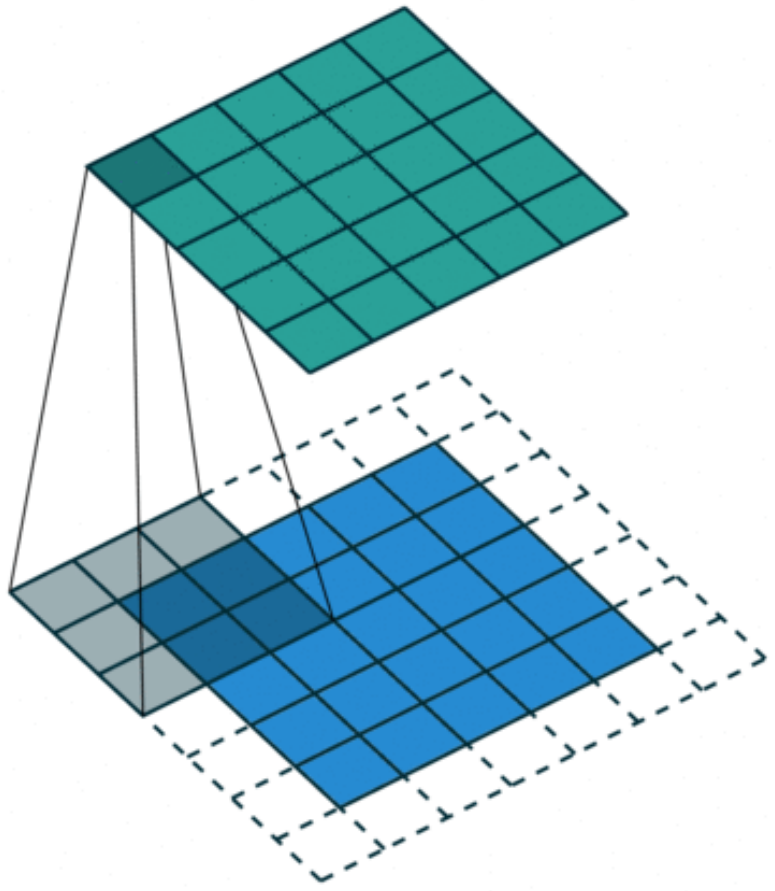




Padding

Padding adds rows/columns around input





$$0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$$

Padding

- Padding p_h rows and p_w columns, output shape will be

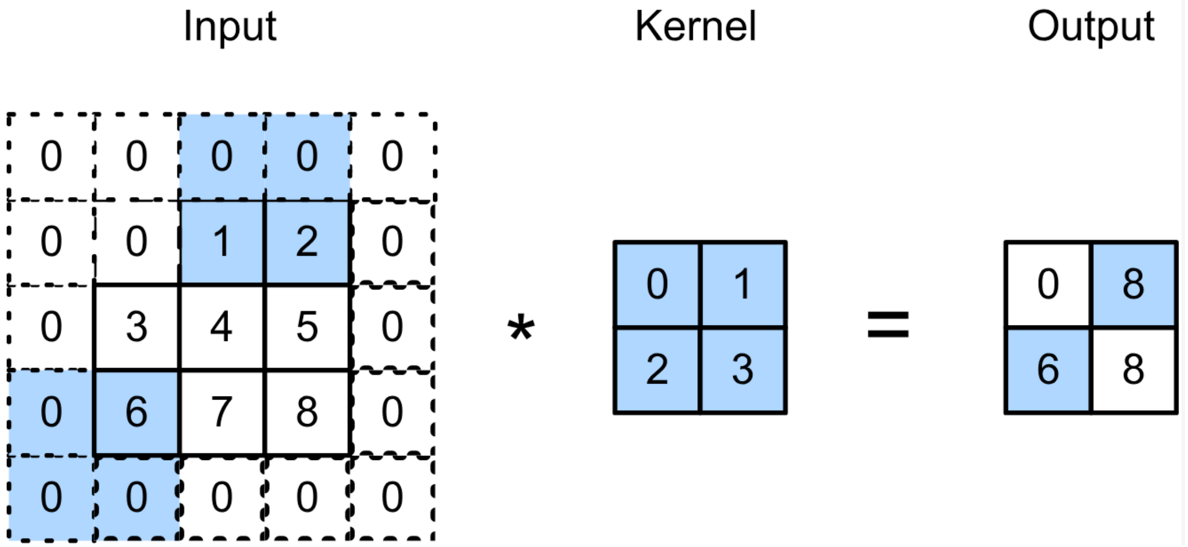
$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$

- A common choice is $p_h = k_h - 1$ and $p_w = k_w - 1$
  - Odd k_h : pad $p_h/2$ on both sides
  - Even k_h : pad $\lceil p_h/2 \rceil$ on top, $\lfloor p_h/2 \rfloor$ on bottom

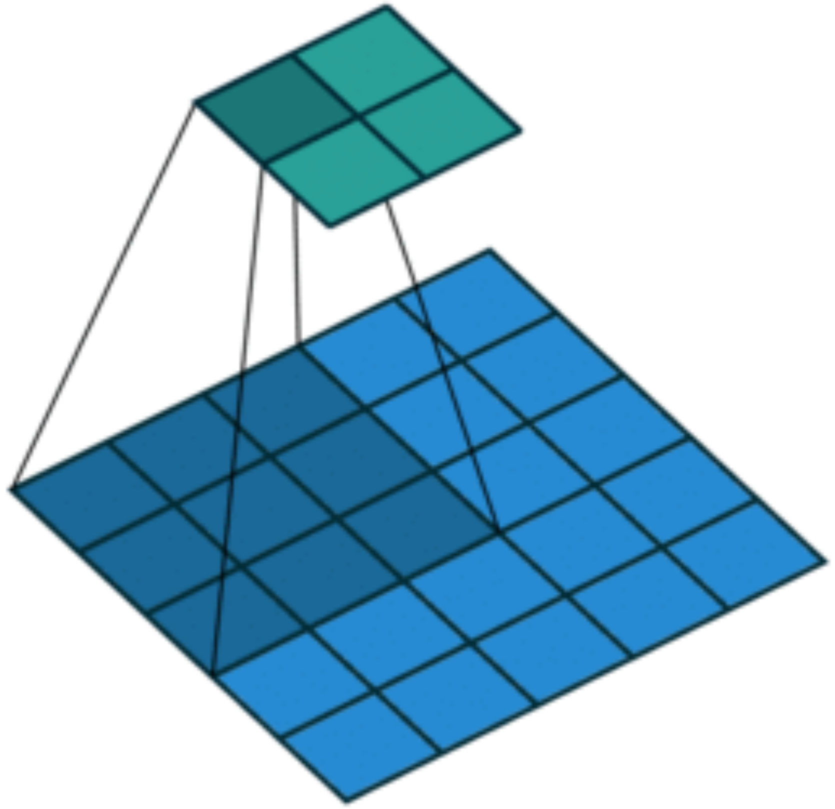
Stride

- Stride is the #rows/#columns per slide

Strides of 3 and 2 for height and width



$$0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$$
$$0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$$



Stride

- Given stride s_h for the height and stride s_w for the width, the output shape is

$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

- With $p_h = k_h - 1$ and $p_w = k_w - 1$

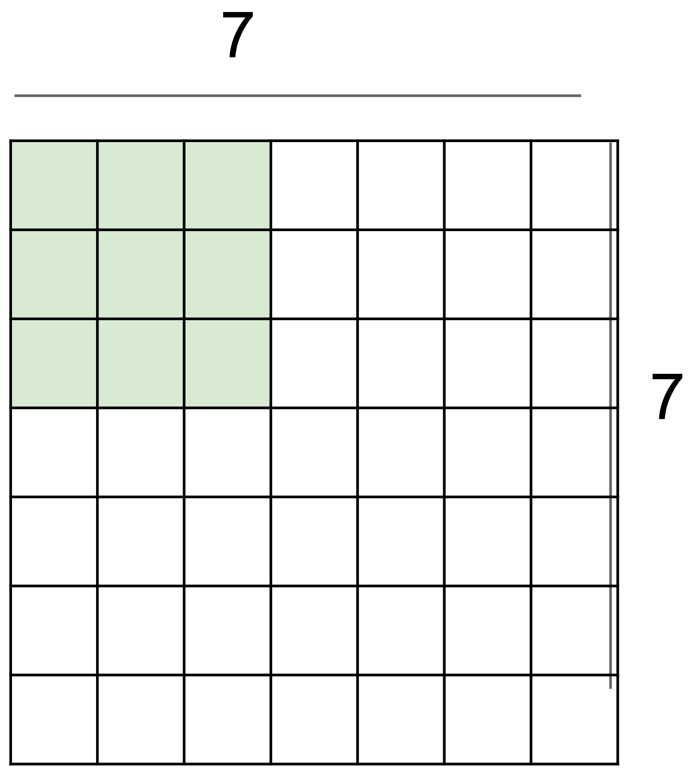
$$\lfloor (n_h + s_h - 1)/s_h \rfloor \times \lfloor (n_w + s_w - 1)/s_w \rfloor$$

- If input height/width are divisible by strides

$$(n_h/s_h) \times (n_w/s_w)$$

Q1. Suppose we want to perform convolution on a single channel image of size 7x7 (no padding) with a kernel of size 3x3, and stride = 2. What is the dimension of the output?

A. 3x3
B. 7x7
C. 5x5
D. 2x2



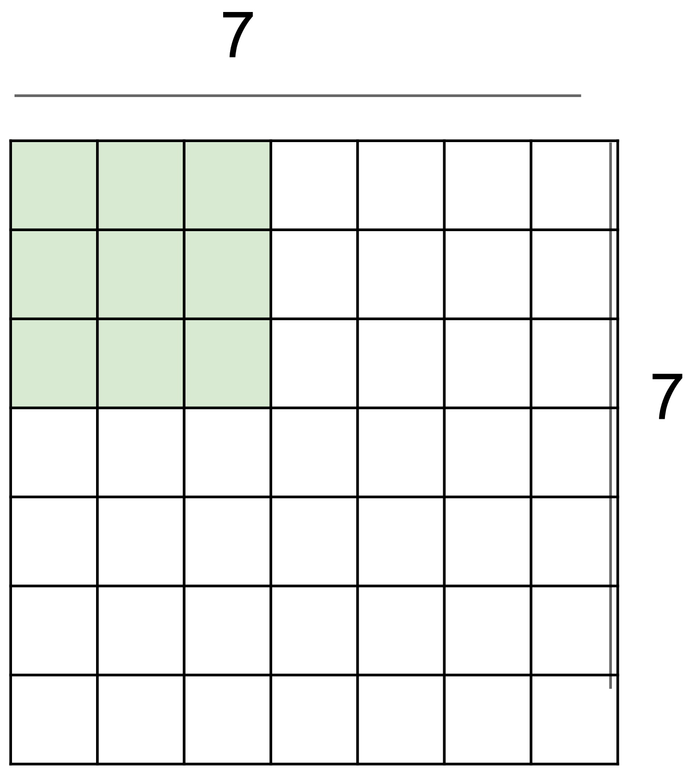
Q1. Suppose we want to perform convolution on a single channel image of size 7x7 (no padding) with a kernel of size 3x3, and stride = 2. What is the dimension of the output?

A. 3x3

B. 7x7

C. 5x5

D. 2x2



$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, Sharon Li, and Fred Sala