Midterm Review

Logistics

- The midterm is scheduled on **May 12th (Tuesday).**
- Choose an 1hr40min **contiguous** slot yourself.
- **Open book and web** (although you won't need it)**.**

- You will take the test **entirely online** in gradescope:
  - The format will be the same as practice midterm. **Do try it in advance.**
  - A 1hr40min timer will start as soon as you open the test page, and won't stop in between.

- Questions during midterm:
  - Raise **all** questions **privately on piazza.**
  - We will make a pinned thread to update all necessary exam clarifications. So pay attention to that.

Logistics

- Question types:
    - True / False:
        - 10 questions. 2 points each.
        - Correct answer is worth 2 points. To discourage guessing, incorrect answers are worth **-1 points**. Leaving a question blank will give **0 points**.)
    - Multiple Choice:
        - 10 questions. 4 points each.
        - Selecting all of the correct options and none of the incorrect options will get full credits. For each incorrect selection, you'll get 2 points off **from the 4**. The minimum score is 0.
        - I know this is confusing, so examples:
            - Correct answer is A. You select A -> 4 points; AB -> 2 points; ABC -> 0 points; ABCD -> 0 points.
            - Correct answer is AB. You select AB -> 4 points; A -> 2 points; **ABC -> 2 points; AC -> 0 points;** CD -> 0 points.

Logistics

- Question types:
    - Short Answers:
        - 40 points. ? questions.
        - You could either type in your answers (math in latex is available by $$$$💰) or write, scan (or photo) and upload. Please make it visually clear if you are writing. We may take points off if we can't easily understand your handwritten work.
        - Note: certain questions might be easier done with the write option. So do prepare a pen, paper, and camera.

Session Objectives

50 minutes is short!

This is just to help you get going with your studies.

- You should understand **all** we've learned so far
- **We won't cover everything that the midterm has and we might cover things not on the midterm**
- Want to synthesize concepts of the course to give intuition/high level picture.
- Try to clarify things we've seen people have more difficulties with
- Would highly recommend really understand the fundamental concepts, details, and reasoning behind the many topics covered in lectures

Overview of today's session

Summary of Course Material:

- Basics of neural networks:
  - Loss function & Regularization
  - Optimization
  - Activation Functions
- How we build complex network models
  - Convolutional Layers
  - Recurrent Neural Networks

Practice Midterm Problems

Q&A, time permitting

Overview of today's session

Summary of Course Material:

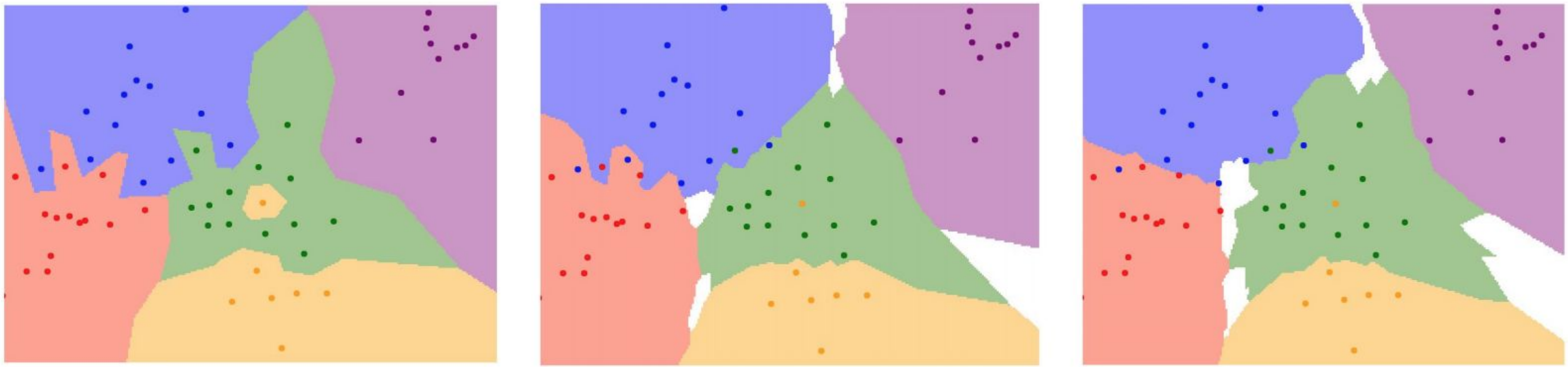
- Basics of neural networks:
  - Loss function & Regularization
  - Optimization
  - Activation Functions
- How we build complex network models
  - Convolutional Layers
  - Recurrent Neural Networks

Practice Midterm Problems

Q&A, time permitting

KNN

- How does it work? Train? Test?
- What positive / negative effect would using larger / smaller k value have?
- Distance function? L1 vs. L2?



K = 1 K = 3 K = 5

Linear Classifier

Image

Array of **32x32x3** numbers (3072 numbers total)

$\longrightarrow$ f(**x**,**W**) $\longrightarrow$ **10** numbers giving class scores

$\uparrow$

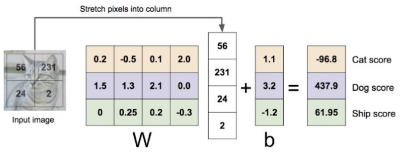
W

parameters or weights

$$f(x,W) = Wx + b$$

Algebraic Viewpoint

f(x,W) = Wx

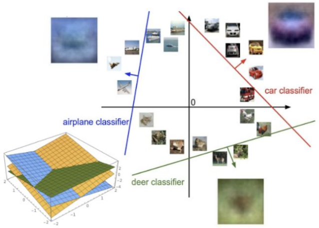


Visual Viewpoint

One template per class

Geometric Viewpoint

Hyperplanes cutting up space



Suppose: 3 training examples, 3 classes.
With some W the scores $f(x,W) = Wx$ are:

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

A **loss function** tells how good our current classifier is

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where x_i is image and y_i is (integer) label

Loss over the dataset is a average of loss over examples:

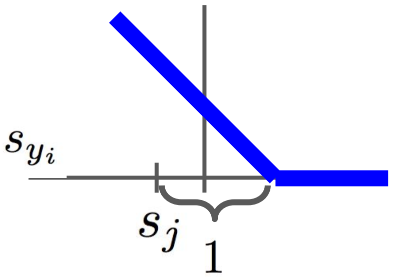
$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

Loss Function: **Multiclass SVM Loss**

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

"Hinge loss"

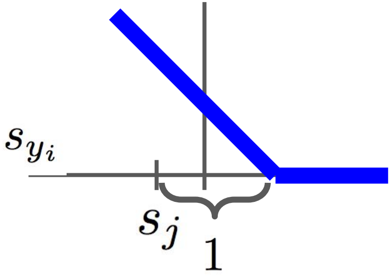


Loss Function: **Multiclass SVM Loss**

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

"Hinge loss"



cat	**3.2**
car	5.1
frog	-1.7
Losses:	2.9

= max(0, 5.1 - 3.2 + 1) +
max(0, -1.7 - 3.2 + 1)

= max(0, 2.9) + max(0, -3.9)

= 2.9 + 0

= 2.9

Aside: Regularization

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a W such that L = 0.
Is this W unique?

No! 2W also has L = 0!
How do we choose between W and 2W?
<u>Regularization</u>

Aside: Regularization

λ = regularization strength (hyperparameter)

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)} + \underbrace{\lambda R(W)}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

Why regularize?

- Express preferences over weights
- Make the model *simpler* to avoid overfitting

Loss Function: **Softmax Classifier**

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$

Softmax Function

$$L_i = -\log P(Y = y_i|X = x_i)$$

Loss Function: **Softmax Classifier**

Want to interpret raw classifier scores as **probabilities**

$$\boxed{s = f(x_i; W)}$$

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}}$$ Softmax Function

Probabilities must be >= 0

Probabilities must sum to 1

$$L_i = -\log P(Y = y_i|X = x_i)$$

cat	**3.2**		**24.5**		**0.13**
car	5.1	$\xrightarrow{\text{exp}}$	164.0	$\xrightarrow{\text{normalize}}$	0.87
frog	-1.7		0.18		0.00
	Unnormalized log-probabilities / logits		unnormalized probabilities		probabilities

$\rightarrow$ L_i = -log(0.13)
= **2.04**

Loss Intuition:

- **Maximizing Likelihood**
- **Comparing Probabilities**

Optimization Motivation

- We have some dataset of (x,y)
- We have a **score function:**
- We have a **loss function**:

$$s = f(x; W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + R(W)$$

How do we find the best W?

Gradient Descent

$$L(W) = \frac{1}{N} \sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$$

$$\nabla_W L(W) = \frac{1}{N} \sum_{i=1}^{N} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$$

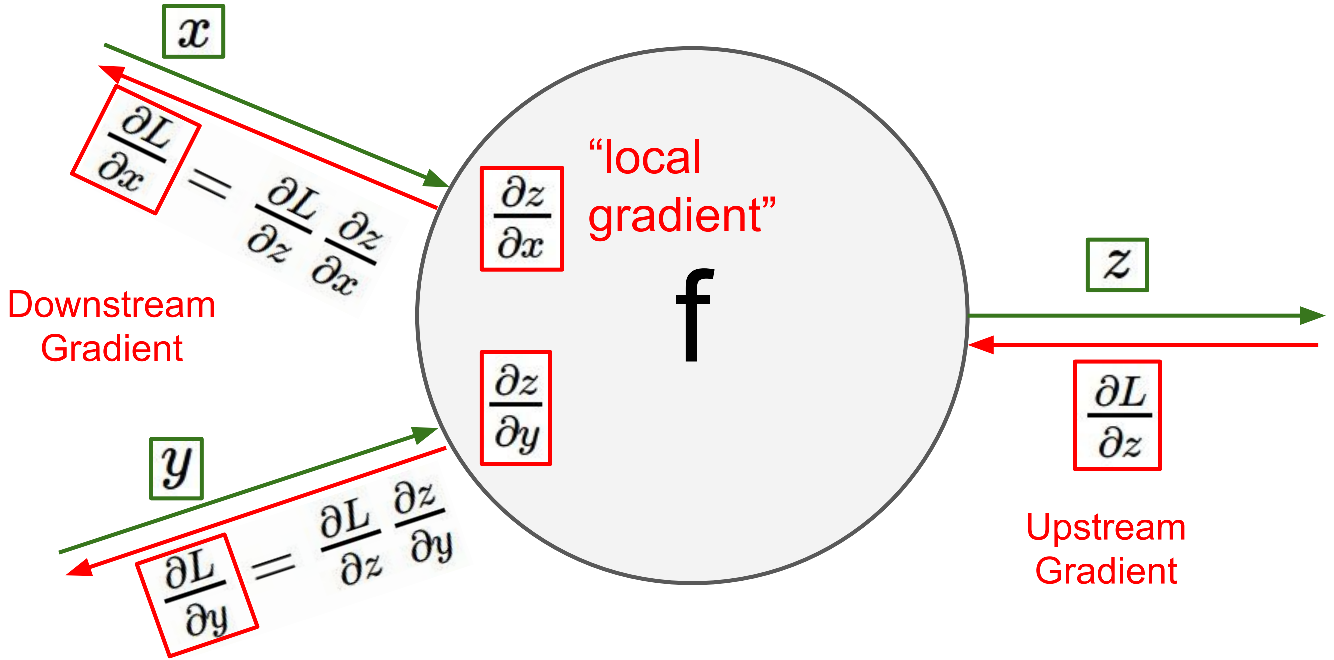
Full sum expensive when N is large!

Approximate sum using a **minibatch** of examples
32 / 64 / 128 common

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

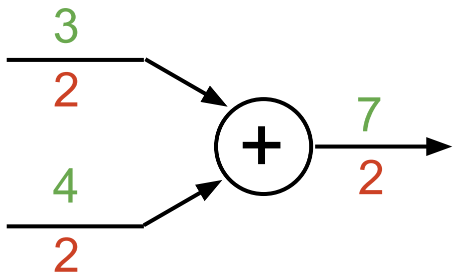
Optimization, Point 1: Calculating Gradients for Updates



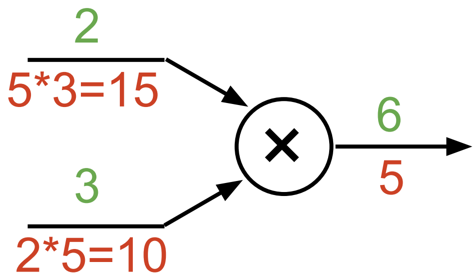
Computational Graphs + Backpropagation (Chain Rule)

Patterns in Gradient Flow

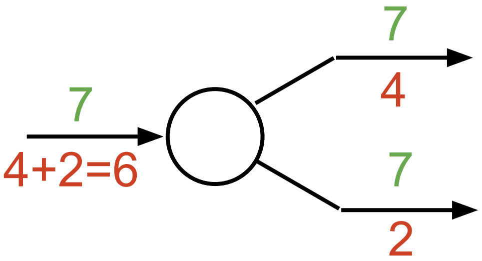
add gate: gradient distributor



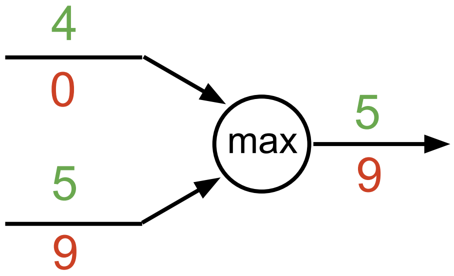
mul gate: “swap multiplier”



copy gate: gradient adder

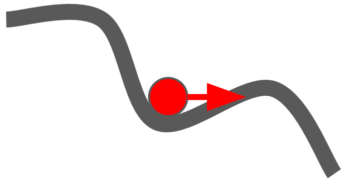


max gate: gradient router

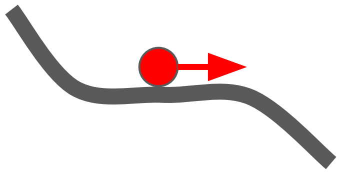


Optimization, Point 2: Things to take care!

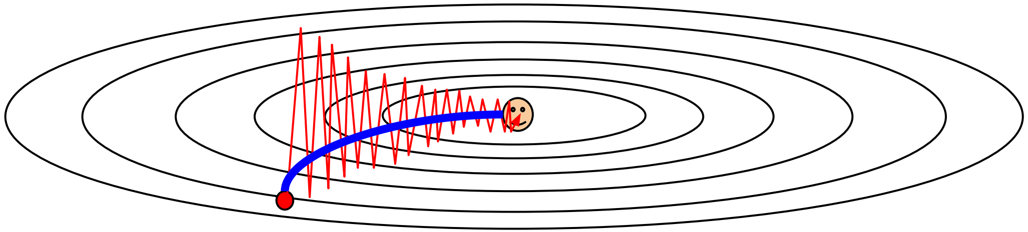
Local Minima

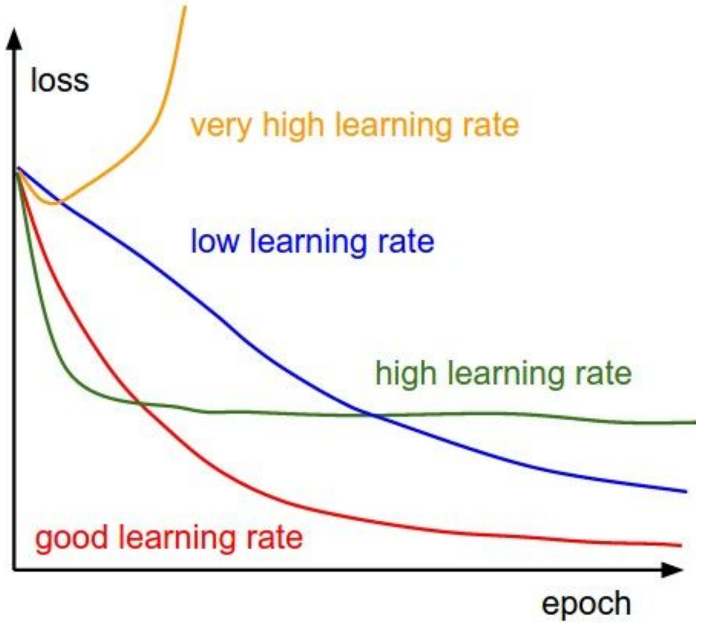


Saddle points



Poor Conditioning





Other Algos: SGD+momentum, AdaGrad, RMSProp, Adam

Problem 2.1

1. You start training your Neural Network but the total loss (cross entropy loss + regularization loss) is almost completely flat from the start. What could be the cause?

   (a) The learning rate could be too low

   (b) The regularization strength could be too high

   (c) The class distribution could be very uneven in the dataset

   (d) The weight initialization scale could be incorrectly set

Problem 2.1: Solution

1. You start training your Neural Network but the total loss (cross entropy loss + regularization loss) is almost completely flat from the start. What could be the cause?

   (a) The learning rate could be too low

   (b) The regularization strength could be too high

   (c) The class distribution could be very uneven in the dataset

   (d) The weight initialization scale could be incorrectly set

Problem 2.1: Solution

1. You start training your Neural Network but the total loss (cross entropy loss + regularization loss) is almost completely flat from the start. What could be the cause?
   (a) The learning rate could be too low
   (b) ~~The regularization strength could be too high~~
   (c) The class distribution could be very uneven in the dataset
   (d) The weight initialization scale could be incorrectly set

Problem 2.1: Solution

1. You start training your Neural Network but the total loss (cross entropy loss + regularization loss) is almost completely flat from the start. What could be the cause?

   (a) The learning rate could be too low

   (b) ~~The regularization strength could be too high~~

   (c) ~~The class distribution could be very uneven in the dataset~~

   (d) The weight initialization scale could be incorrectly set

Problem 2.1: Solution

1. You start training your Neural Network but the total loss (cross entropy loss + regularization loss) is almost completely flat from the start. What could be the cause?

   (a) The learning rate could be too low

   (b) ~~The regularization strength could be too high~~

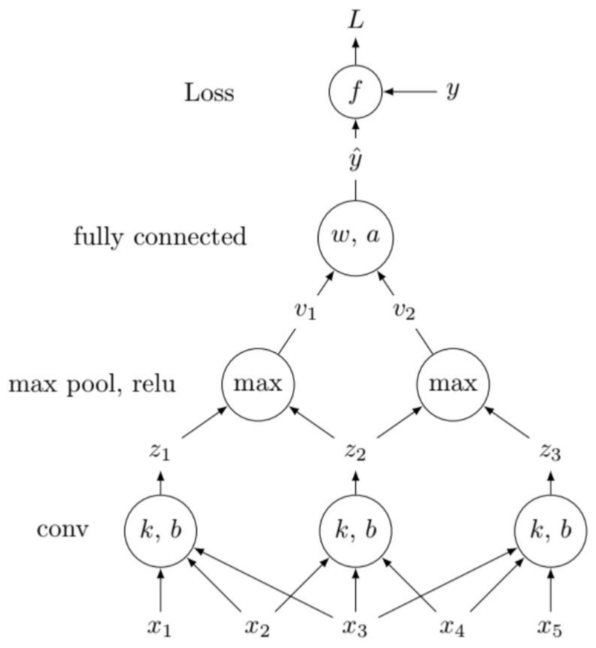
   (c) ~~The class distribution could be very uneven in the dataset~~

   (d) The weight initialization scale could be incorrectly set

Problem 3.4

3.4 Simple ConvNet (12 points)

Consider the following 1-dimensional ConvNet, where all variables are scalars:



$$L = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + a$$

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \max\{z_1, z_2, 0\} \\ \max\{z_2, z_3, 0\} \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

Problem 3.4, 3): Solution

3. **(3 points)** Given the gradients of the loss L with respect to the second layer activations v, derive the gradient of the loss with respect to the first layer activations z. More precisely, given

$$\frac{\partial L}{\partial v_1} = \delta_1 \qquad \frac{\partial L}{\partial v_2} = \delta_2$$

Determine the following

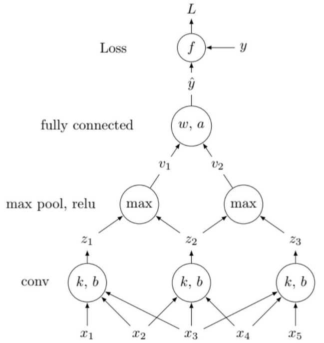
$$\frac{\partial L}{\partial z_1} =$$

$$\frac{\partial L}{\partial z_2} =$$

$$\frac{\partial L}{\partial z_3} =$$

3.4 Simple ConvNet (12 points)

Consider the following 1-dimensional ConvNet, where all variables are scalars:



$$L = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + a$$

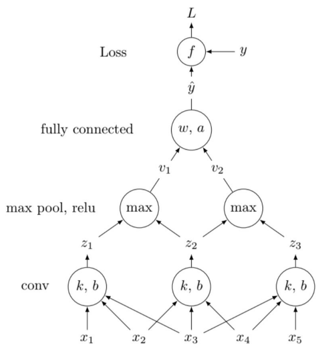
$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \max\{z_1, z_2, 0\} \\ \max\{z_2, z_3, 0\} \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

Problem 3.4, 3): Solution

3.4 Simple ConvNet (12 points)

Consider the following 1-dimensional ConvNet, where all variables are scalars:



$$L = \frac{1}{2}(y-\hat{y})^2$$

$$\hat{y} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + a$$

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \max\{z_1, z_2, 0\} \\ \max\{z_2, z_3, 0\} \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

3. **(3 points)** Given the gradients of the loss L with respect to the second layer activations v, derive the gradient of the loss with respect to the first layer activations z. More precisely, given

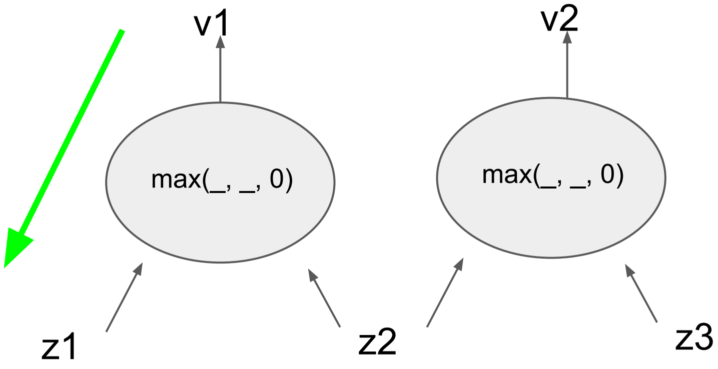
$$\frac{\partial L}{\partial v_1} = \delta_1 \qquad \frac{\partial L}{\partial v_2} = \delta_2$$

Determine the following

$$\frac{\partial L}{\partial z_1} =$$

$$\frac{\partial L}{\partial z_2} =$$

$$\frac{\partial L}{\partial z_3} =$$

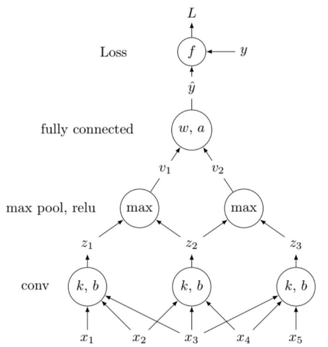


$$\frac{\partial L}{\partial z_1} = \begin{cases} \delta_1 & \text{if } z_1 = \max\{z_1, z_2, 0\} \\ 0 & otherwise \end{cases}$$

Problem 3.4, 3): Solution

3.4 Simple ConvNet (12 points)

Consider the following 1-dimensional ConvNet, where all variables are scalars:



$$L = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + a$$

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \max\{z_1, z_2, 0\} \\ \max\{z_2, z_3, 0\} \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

3. **(3 points)** Given the gradients of the loss L with respect to the second layer activations v, derive the gradient of the loss with respect to the first layer activations z. More precisely, given

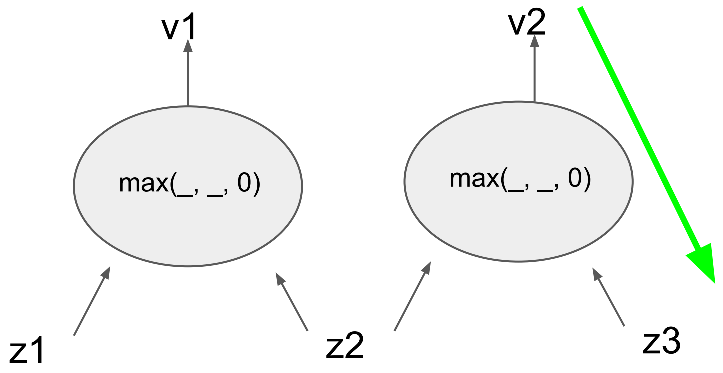
$$\frac{\partial L}{\partial v_1} = \delta_1 \qquad \frac{\partial L}{\partial v_2} = \delta_2$$

Determine the following

$$\frac{\partial L}{\partial z_1} = \begin{cases} \delta_1 & \text{if } z_1 = \max\{z_1, z_2, 0\} \\ 0 & \textit{otherwise} \end{cases}$$

$$\frac{\partial L}{\partial z_2} =$$

$$\frac{\partial L}{\partial z_3} =$$

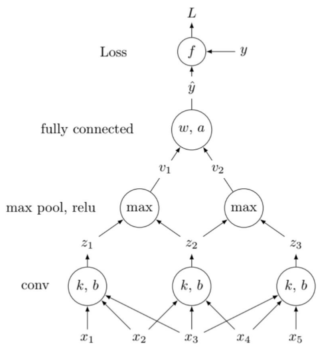


$$\frac{\partial L}{\partial z_3} = \begin{cases} \delta_2 & \text{if } z_3 = \max\{z_2, z_3, 0\} \\ 0 & \textit{otherwise} \end{cases}$$

Problem 3.4, 3): Solution

3.4 Simple ConvNet (12 points)

Consider the following 1-dimensional ConvNet, where all variables are scalars:



$$L = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + a$$

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \max\{z_1, z_2, 0\} \\ \max\{z_2, z_3, 0\} \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} k_1 & k_2 & k_3 & 0 & 0 \\ 0 & k_1 & k_2 & k_3 & 0 \\ 0 & 0 & k_1 & k_2 & k_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

3. **(3 points)** Given the gradients of the loss L with respect to the second layer activations v, derive the gradient of the loss with respect to the first layer activations z. More precisely, given

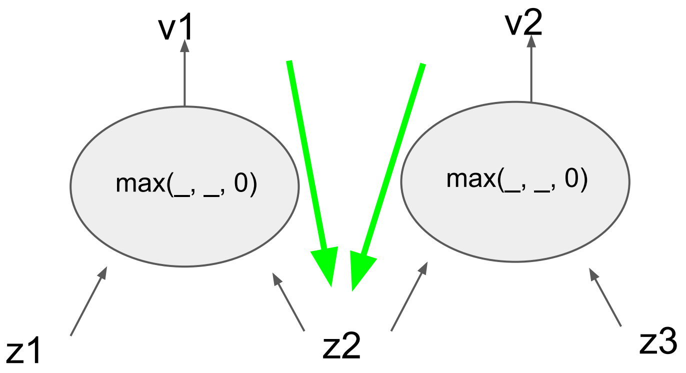
$$\frac{\partial L}{\partial v_1} = \delta_1 \qquad \frac{\partial L}{\partial v_2} = \delta_2$$

Determine the following

$$\frac{\partial L}{\partial z_1} = \begin{cases} \delta_1 & \text{if } z_1 = \max\{z_1, z_2, 0\} \\ 0 & \textit{otherwise} \end{cases}$$

$$\frac{\partial L}{\partial z_2} =$$

$$\frac{\partial L}{\partial z_3} = \begin{cases} \delta_2 & \text{if } z_3 = \max\{z_2, z_3, 0\} \\ 0 & \textit{otherwise} \end{cases}$$



$$\frac{\partial L}{\partial z_2} = \begin{cases} \delta_1 + \delta_2 & \text{if } z_2 = \max\{z_1, z_2, 0\} \text{ and } z_2 = \max\{z_2, z_3, 0\} \\ \delta_1 & \text{else if } z_2 = \max\{z_1, z_2, 0\} \\ \delta_2 & \text{else if } z_2 = \max\{z_2, z_3, 0\} \\ 0 & \text{otherwise} \end{cases}$$

Overview of today's session

Summary of Course Material:

- Basics of neural networks:
  - ~~Loss function $ Regularization~~
  - ~~Optimization~~
  - Activation Functions
- How we build complex network models
  - Convolutional Layers
  - Recurrent Neural Networks

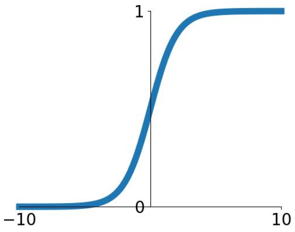
Practice Midterm Problems

Q&A, time permitting

Activation Functions

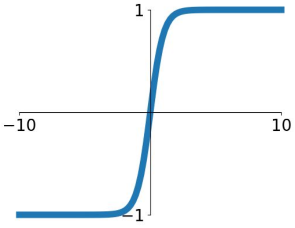
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



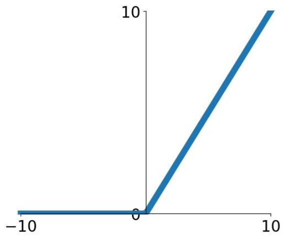
tanh

$\tanh(x)$



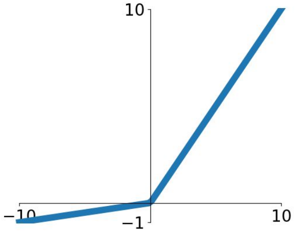
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

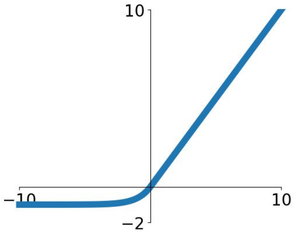


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Problem 2.4:

4. Which of the following are valid activation functions (elementwise non-linearities) you could use in a neural network? (That is, which functions could be effective when training a neural net in practice?)

○ A: $f(x) = \max(0.25x, 0.75x)$

○ B: $f(x) = \min(0, x)$

○ C: $f(x) = 0.7x$

○ D: $f(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ -1 & \text{else} \end{cases}$

Problem 2.4: Solution

4. Which of the following are valid activation functions (elementwise non-linearities) you could use in a neural network? (That is, which functions could be effective when training a neural net in practice?)

- ■ A: $f(x) = \max(0.25x, 0.75x)$
- ○ B: $f(x) = \min(0, x)$
- ○ C: $f(x) = 0.7x$
- ○ D: $f(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ -1 & \text{else} \end{cases}$

Problem 2.4: Solution

4. Which of the following are valid activation functions (elementwise non-linearities) you could use in a neural network? (That is, which functions could be effective when training a neural net in practice?)

- A: $f(x) = \max(0.25x, 0.75x)$
- B: $f(x) = \min(0, x)$
- C: $f(x) = 0.7x$
- D: $f(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ -1 & \text{else} \end{cases}$

Problem 2.4: Solution

4. Which of the following are valid activation functions (elementwise non-linearities) you could use in a neural network? (That is, which functions could be effective when training a neural net in practice?)

- A: $f(x) = \max(0.25x, 0.75x)$
- B: $f(x) = \min(0, x)$
- C: $f(x) = 0.7x$
- D: $f(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ -1 & \text{else} \end{cases}$

Problem 2.4: Solution

4. Which of the following are valid activation functions (elementwise non-linearities) you could use in a neural network? (That is, which functions could be effective when training a neural net in practice?)

A: $f(x) = \max(0.25x, 0.75x)$

B: $f(x) = \min(0, x)$

C: $f(x) = 0.7x$

D: $f(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ -1 & \text{else} \end{cases}$

Overview of today's session

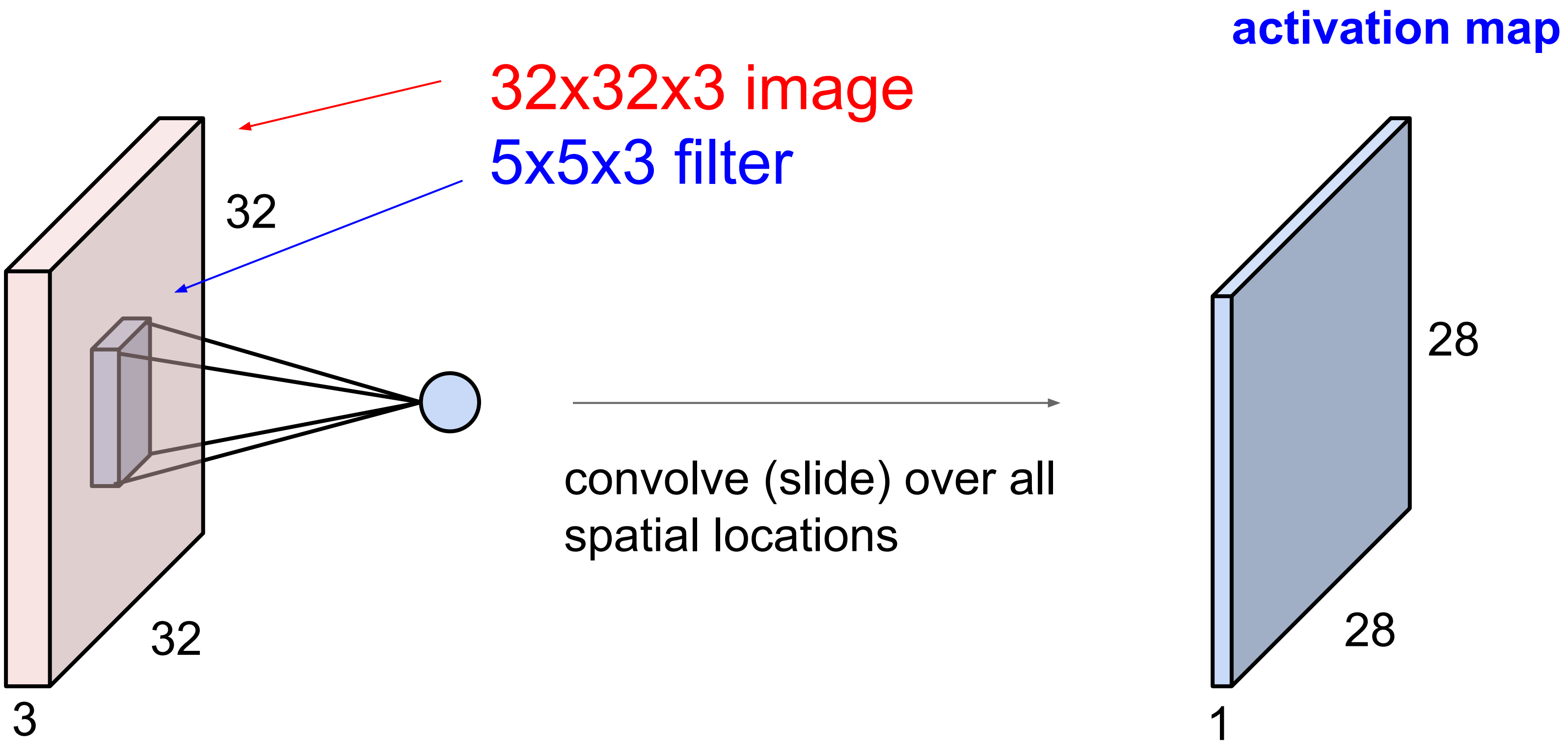
Summary of Course Material:

- Basics of neural networks:
  - Loss function & Regularization
  - Optimization
  - Activation Functions
- How we build complex network models
  - Convolutional Layers
  - Batch/Layer Normalization
  - Recurrent Neural Networks

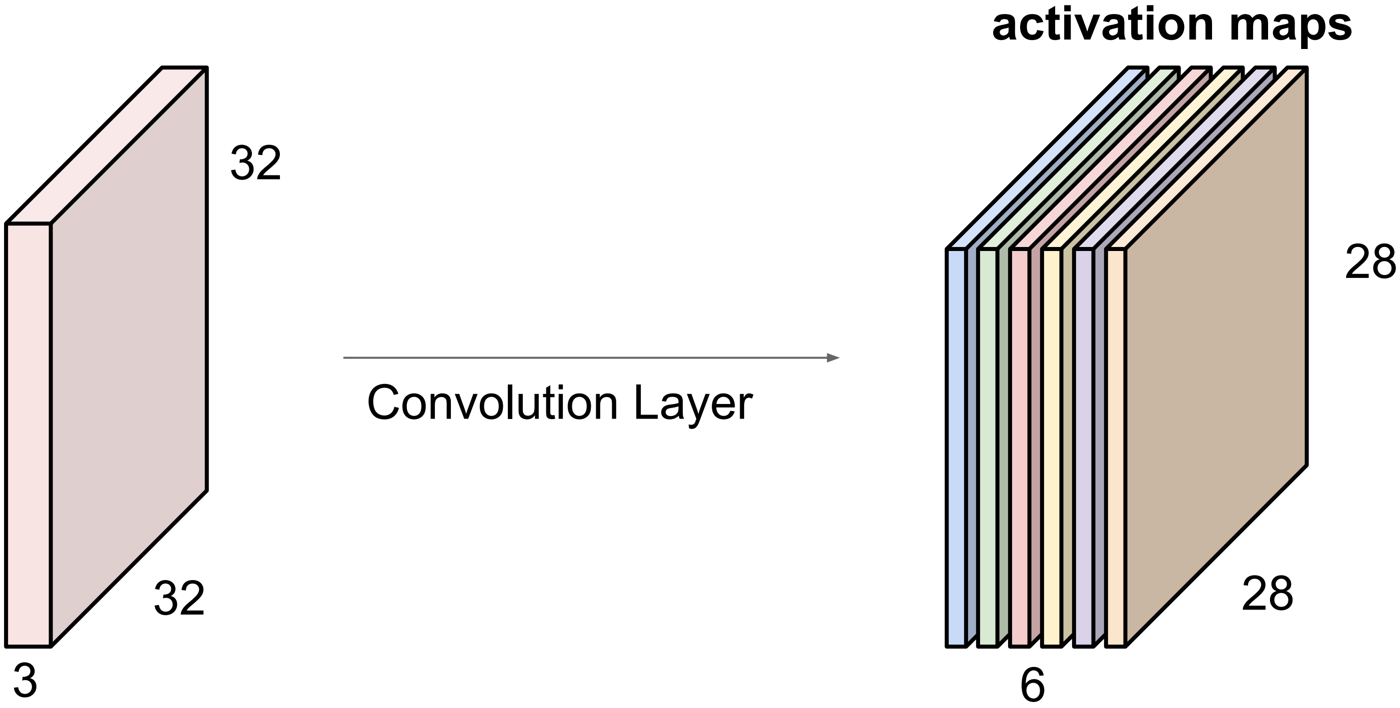
Practice Midterm Problems

Q&A, time permitting

Convolution Layer



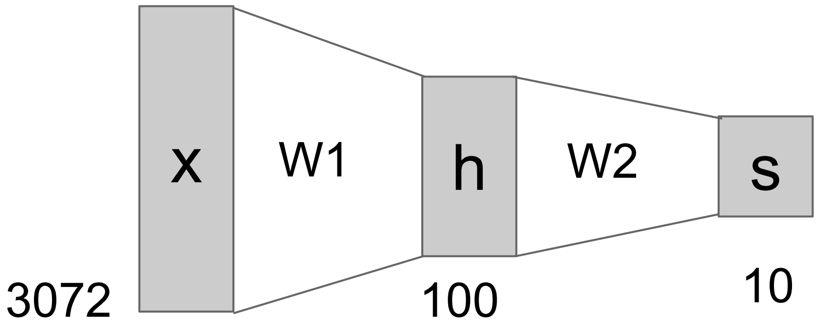
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Convolution Layer

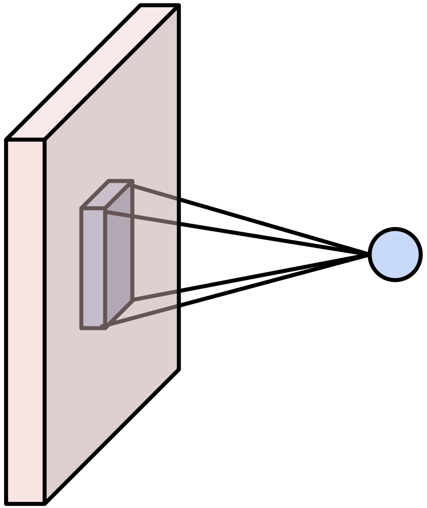
In contrast to fully connected layer,
Each term in output is dependent on spatially local 'subregions' of input



$$out_i = \sum_{j=1}^{H \times W \times C} w_{ij} \cdot in_j + b_i$$

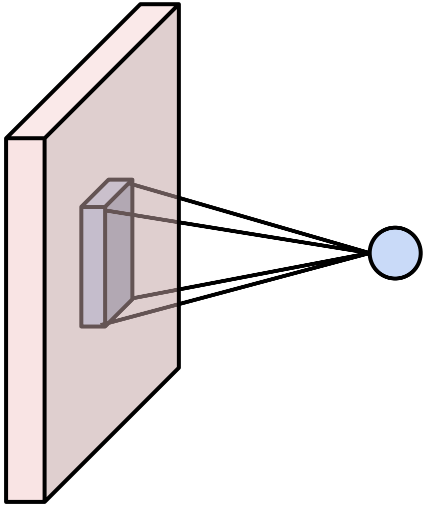
Convolution Layer

In contrast to fully connected layer,
Each term in output is dependent on spatially local ‘subregions’ of input



Convolution Layer

In contrast to fully connected layer,
Each term in output is dependent on spatially local 'subregions' of input



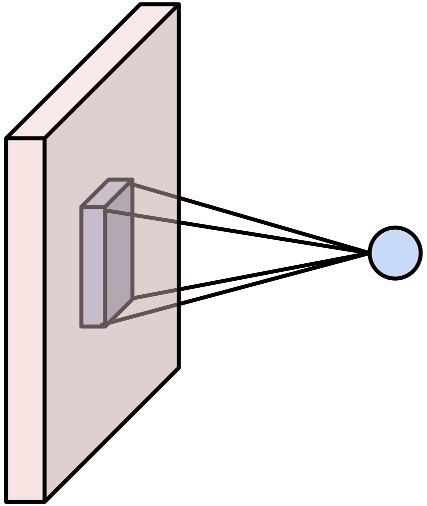
Question: connection between an FC layer and a convolutional layer?

$$out_i = \sum_{j=1}^{H \times W \times C} w_{ij} \cdot in_j + b_i$$

$$out_i = \sum_{j=1}^{HH \times WW \times C} w_{ij} \cdot in(patch_i)_j + b_i$$

Convolution Layer

In contrast to fully connected layer,
Each term in output is dependent on spatially local 'subregions' of input



Question: connection between an FC layer and a convolutional layer?

Answer: FC looks like convolution layer with filter size HxW

$$out_i = \sum_{j=1}^{H \times W \times C} w_{ij} \cdot in_j + b_i$$

$$out_i = \sum_{j=1}^{HH \times WW \times C} w_{ij} \cdot in(patch_i)_j + b_i$$

Problem 3.3

3.3 Convolutional Architectures

Consider the convolutional network defined by the layers in the left column below. Fill in the size of the activation volumes at each layer, and the number of parameters at each layer. You can write your answer as a multiplication (e.g. 128x128x3).

- CONV5-N denotes a convolutional layer with N filters, each of size 5x5xD, where D is the depth of the activation volume at the previous layer. Padding is 2, and stride is 1.
- POOL2 denotes a 2x2 max-pooling layer with stride 2 (pad 0)
- FC-N denotes a fully-connected layer with N output neurons.

Layer	Activation Volume Dimensions (memory)	Number of parameters
INPUT	32x32x1	0
CONV5-10		
POOL2		
CONV5-10		
POOL2		
FC-10		

Prob 3.3 - Solution

3.3 Convolutional Architectures

Consider the convolutional network defined by the layers in the left column below. Fill in the size of the activation volumes at each layer, and the number of parameters at each layer. You can write your answer as a multiplication (e.g. 128x128x3).

- CONV5-N denotes a convolutional layer with N filters, each of size 5x5xD, where D is the depth of the activation volume at the previous layer. Padding is 2, and stride is 1.
- POOL2 denotes a 2x2 max-pooling layer with stride 2 (pad 0)
- FC-N denotes a fully-connected layer with N output neurons.

Layer	Activation Volume Dimensions (memory)	Number of parameters
INPUT	32x32x1	0
CONV5-10		
POOL2		
CONV5-10		
POOL2		
FC-10		

$$w_{out} = \frac{w_{in} + w_{pad} - k}{s} + 1$$

Solution Cont.

Layer	Activation volume	No. of Parameters
Input	32x32x1	0
Conv5-10	32x32x10	10x1x5x5 + 10
Pool-2	16x16x10	0
Conv5-10		

$$W_{out} = \frac{32 - 5 + 2*2}{1} + 1$$
$$= 31 + 1$$
$$W_{out} = 32$$

Parameters = NxCxHHxWW (number of elements in weight matrix) + N(bias)

Solution Contd.

Layer	Activation volume	No. of Parameters
Input	32x32x1	0
Conv5-10	32x32x10	10x1x5x5 + 10
Pool-2	16x16x10	0
Conv5-10	16x16x10	10x10x5x5 + 10

$$W_{out} = \frac{16 - 5 + 2 * 2}{1} + 1$$
$$= 15 + 1$$
$$W_{out} = 16$$

Parameters = NxCxHHxWW (number of elements in weight matrix) + N(bias)

Solution Contd.

Layer	Activation volume	No. of Parameters
Input	32x32x1	0
Conv5-10	32x32x10	10x1x5x5 + 10
Pool-2	16x16x10	0
Conv5-10	16x16x10	10x10x5x5 + 10
Pool2	8x8x10	0
FC-N	10	8x8x10x10 + 10

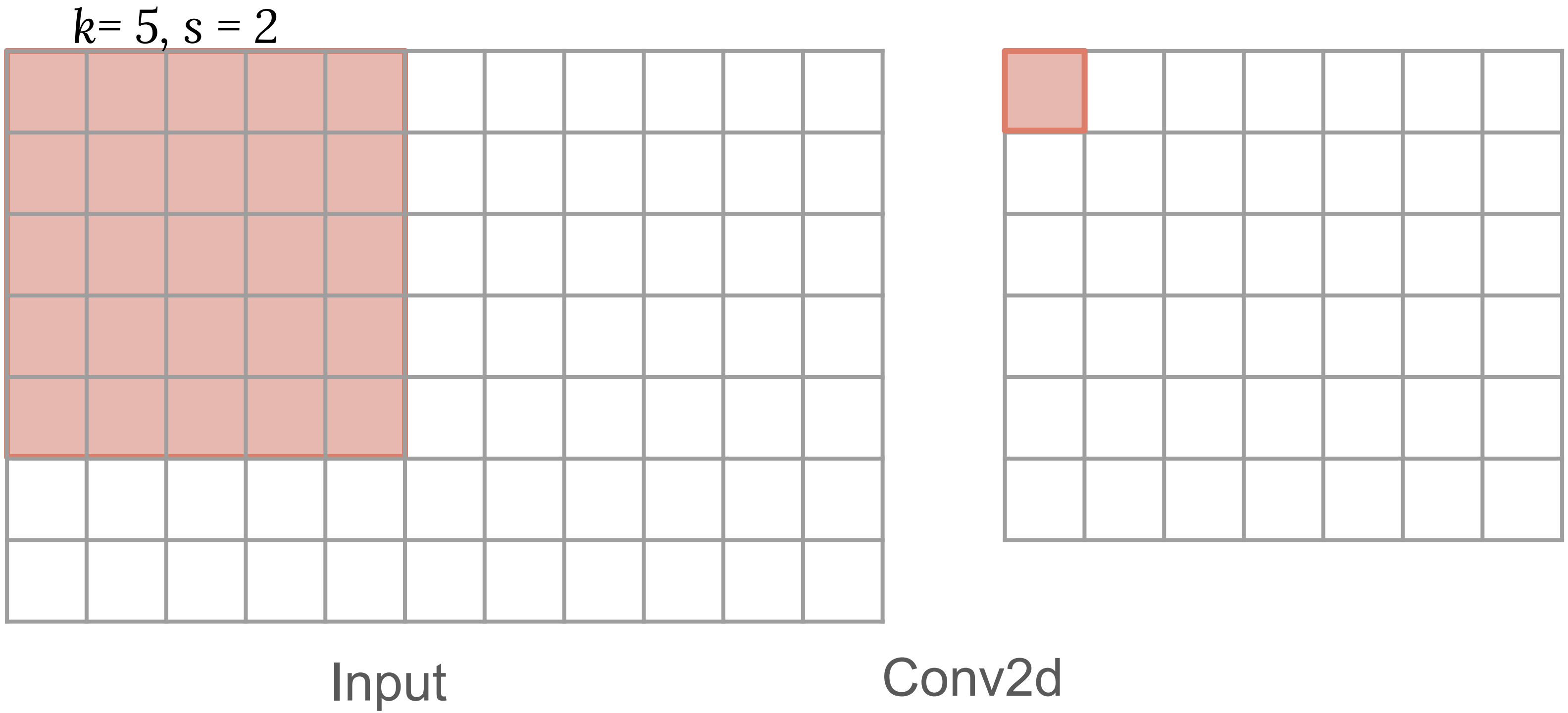
Summary

For kernel width $\mathbf{k}$ and stride $\mathbf{s}$,
Input width $\mathbf{w}_{in}$ and total padding $\mathbf{w}_{pad}$,
Output width $\mathbf{w}_{out}$ is

$$w_{out} = \frac{1}{s}(w_{in} + w_{pad} - k) + 1$$

Deriving Output size using Receptive Field

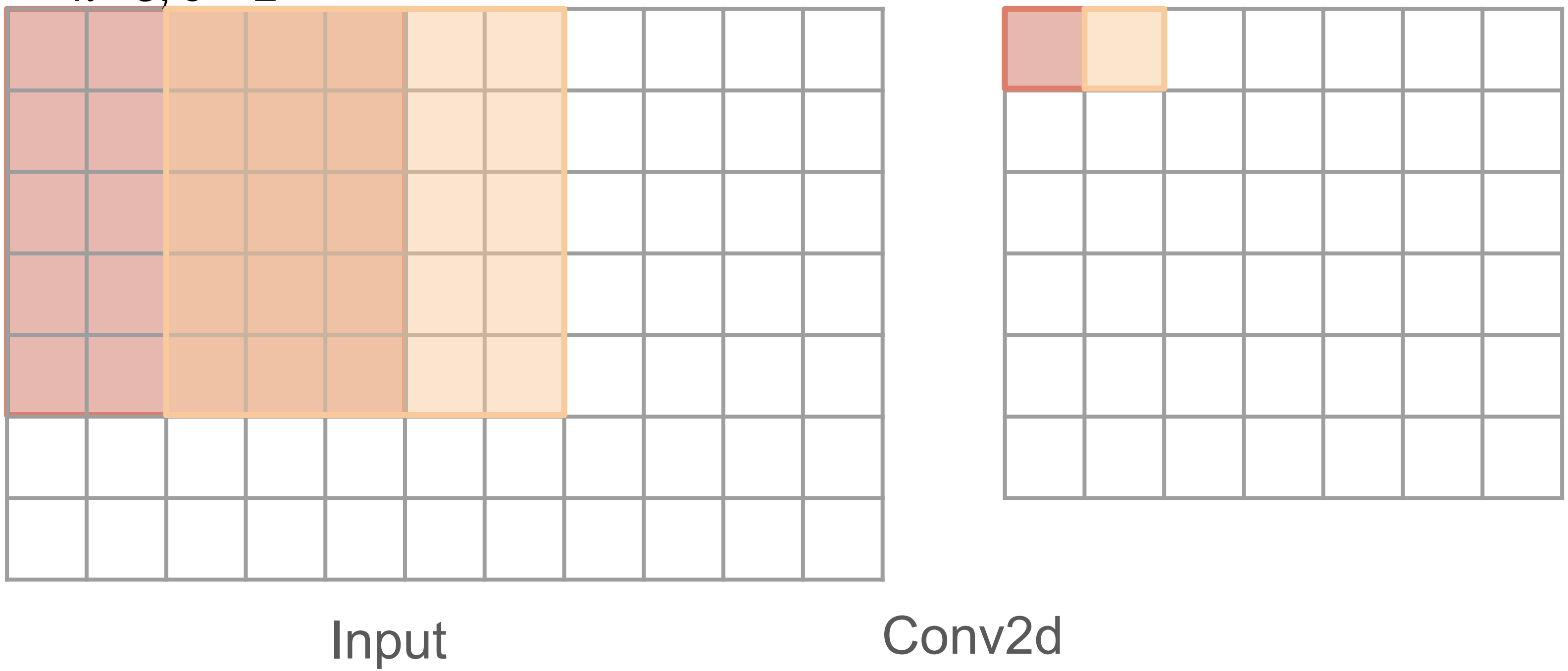
'Input data seen/received' in single activation layer 'pixel'



Deriving Output size using Receptive Field

‘Input data seen/received’ in single activation layer ‘pixel’

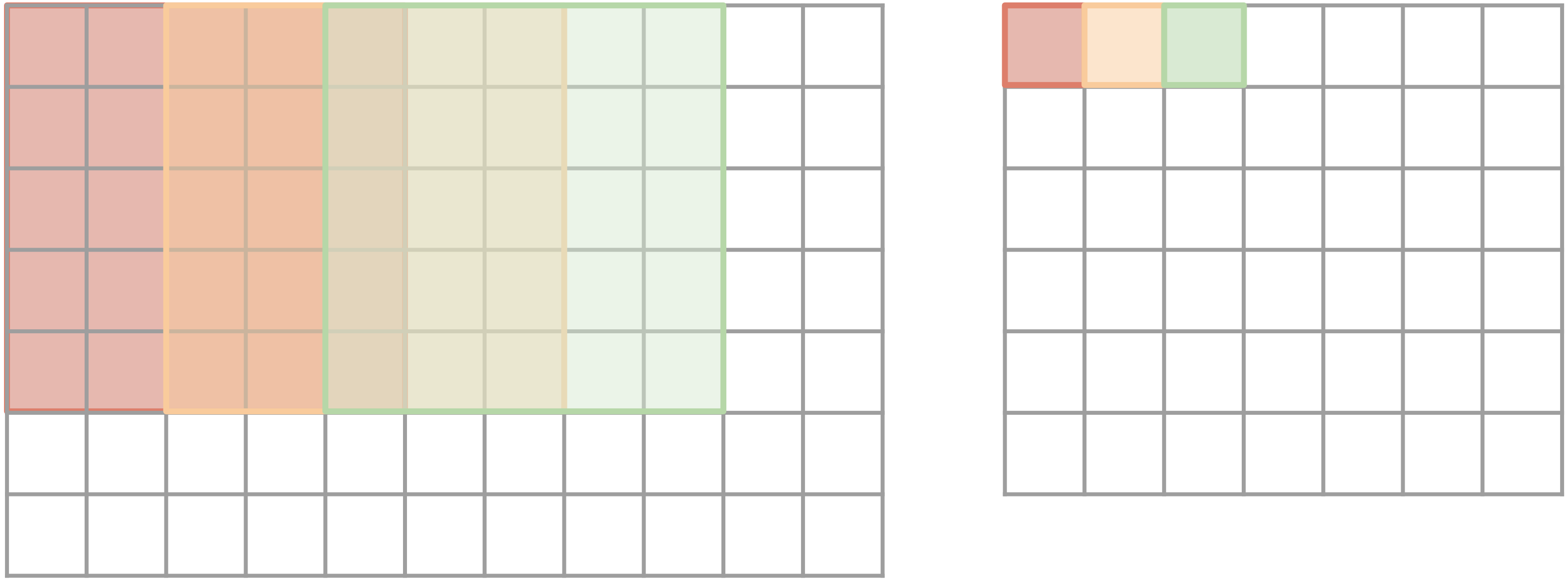
$k = 5$, $s = 2$



Deriving Output size using Receptive Field

'Input data seen/received' in single activation layer 'pixel'

$k = 5$, $s = 2$

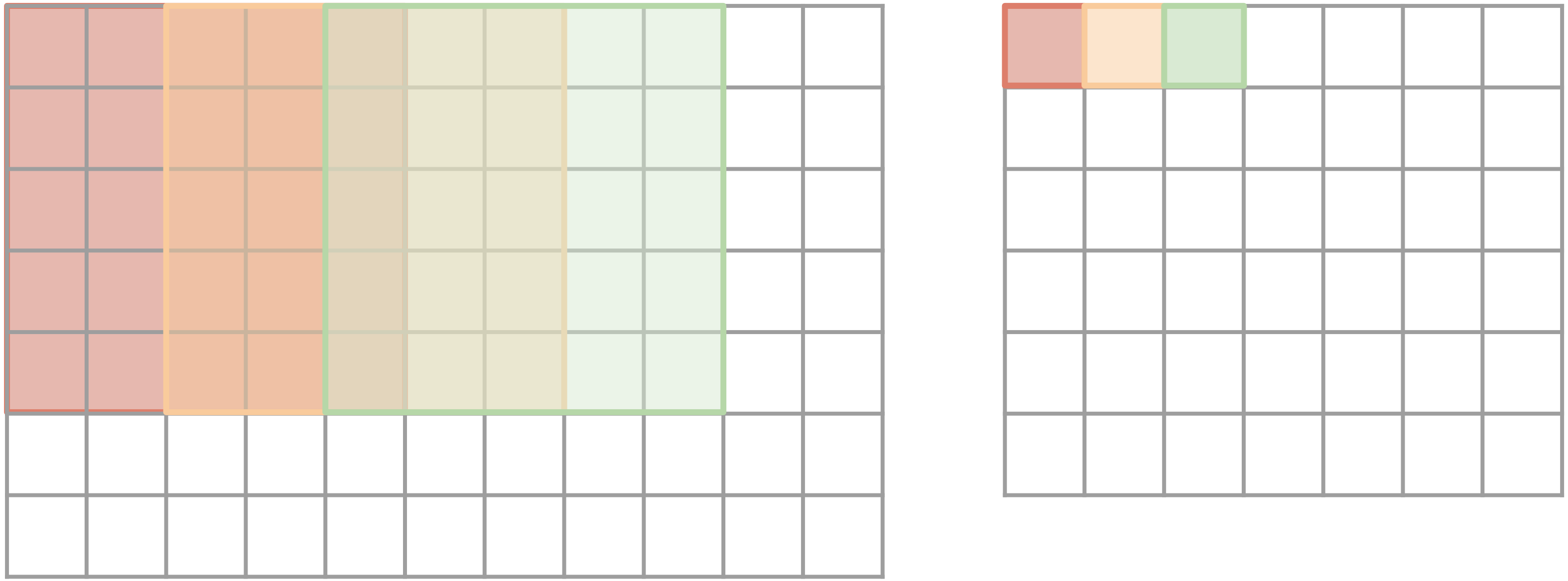


Deriving Output size using Receptive Field

'Input data seen/received' in single activation layer 'pixel'

k= 5, s = 2

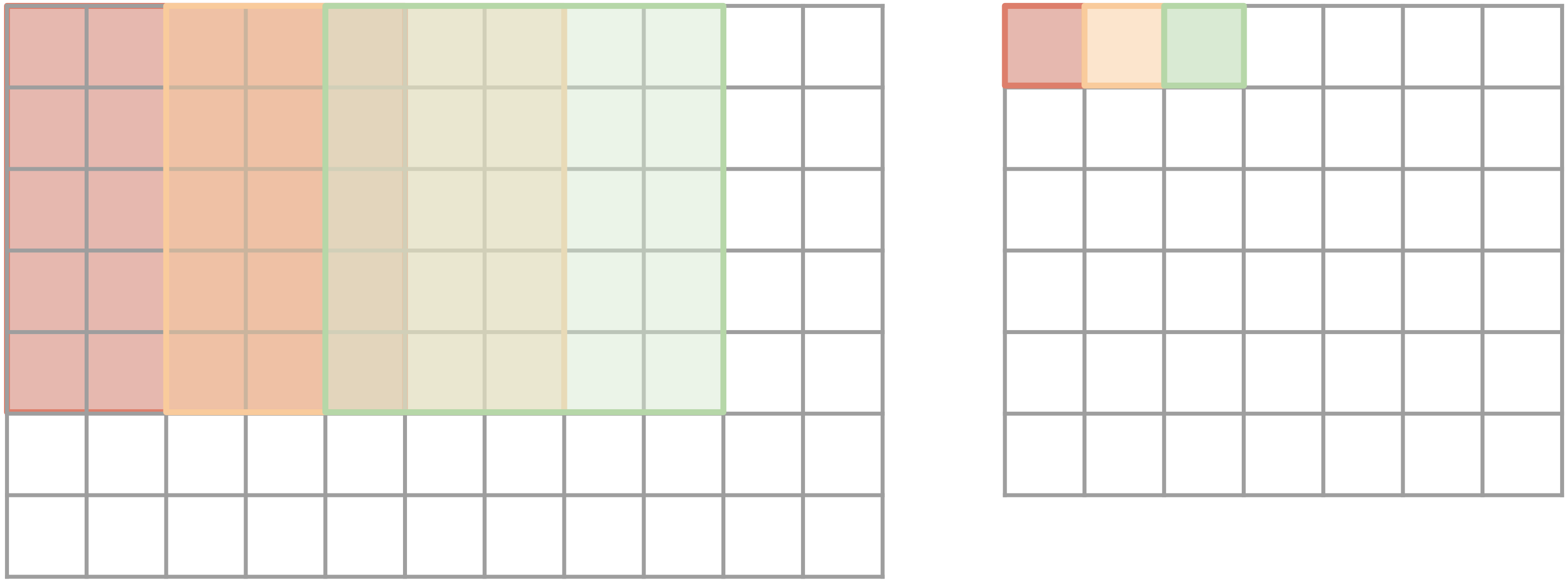
$$n = 5 + 2 \times (3 - 1)$$



Deriving Output size using Receptive Field

'Input data seen/received' in single activation layer 'pixel'

k= 5, s = 2 $$n = 5 + 2 \times (3 - 1) \Rightarrow n = k + s(m - 1)$$



Summary

Given kernel width **k** and stride **s**,
For **m** adjacent pixels in the activation output,
Cumulative receptive field **n** *with respect to layer input* is

$$n = k + s(m - 1)$$

Receptive field size

Given kernel width **k** and stride **s**,
For **m** adjacent pixels in the activation output,
Cumulative receptive field **n** *with respect to layer input* is

$$n = k + s(m - 1)$$

Note: Generally when we refer to ‘ effective receptive field’,
we mean with respect to **input data/layer 0/original image**,
not with respect to **direct input to the layer**

Receptive field size

Given kernel width **k** and stride **s**,
For **m** adjacent pixels in the activation output,
Cumulative receptive field **n** *with respect to layer input* is

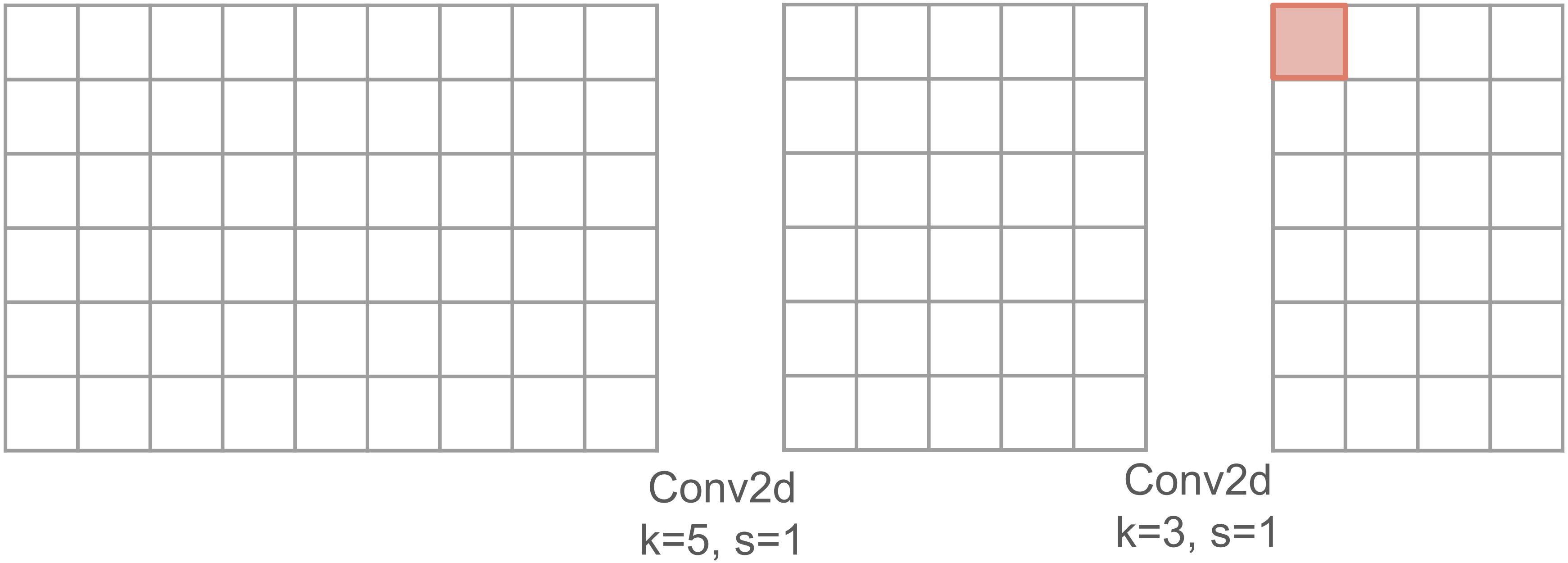
$$n = k + s(m - 1)$$

Note: Need to be computed recursively

Receptive field size computation

$$n = k + s(m - 1)$$

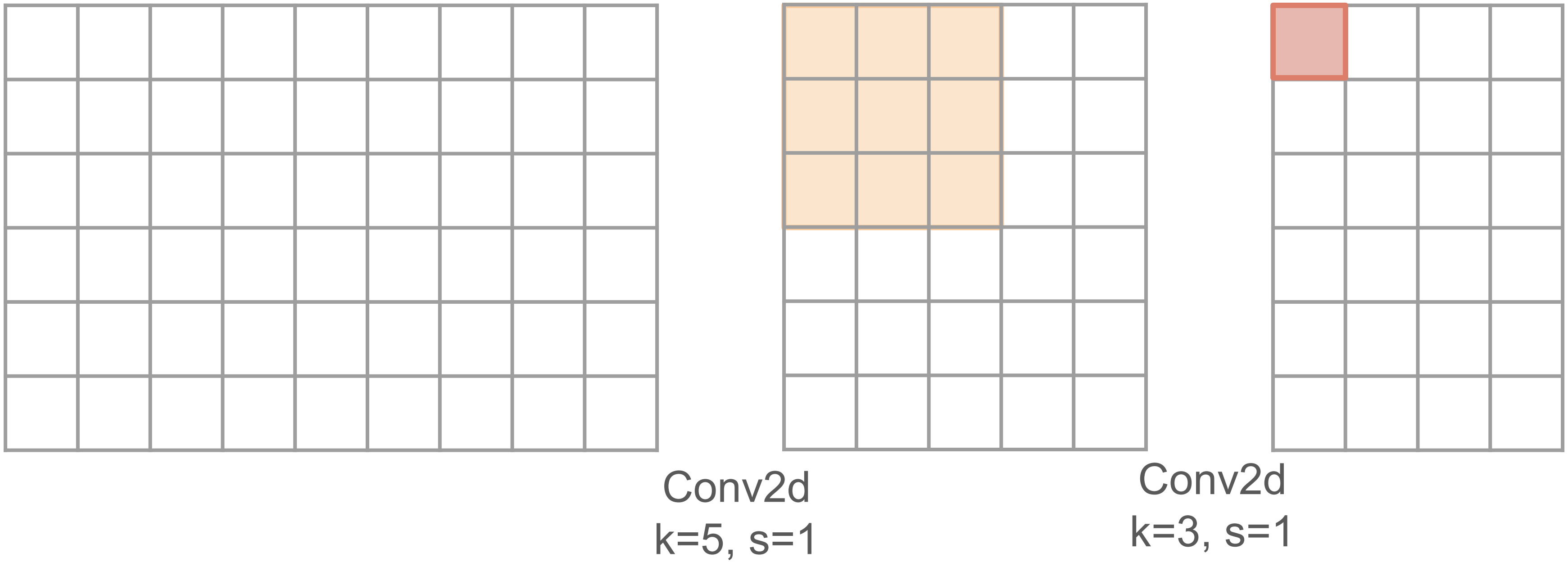
k=3, s=1, m=1



Receptive field size computation

$$n = k + s(m - 1)$$

k=3, s=1, m=1

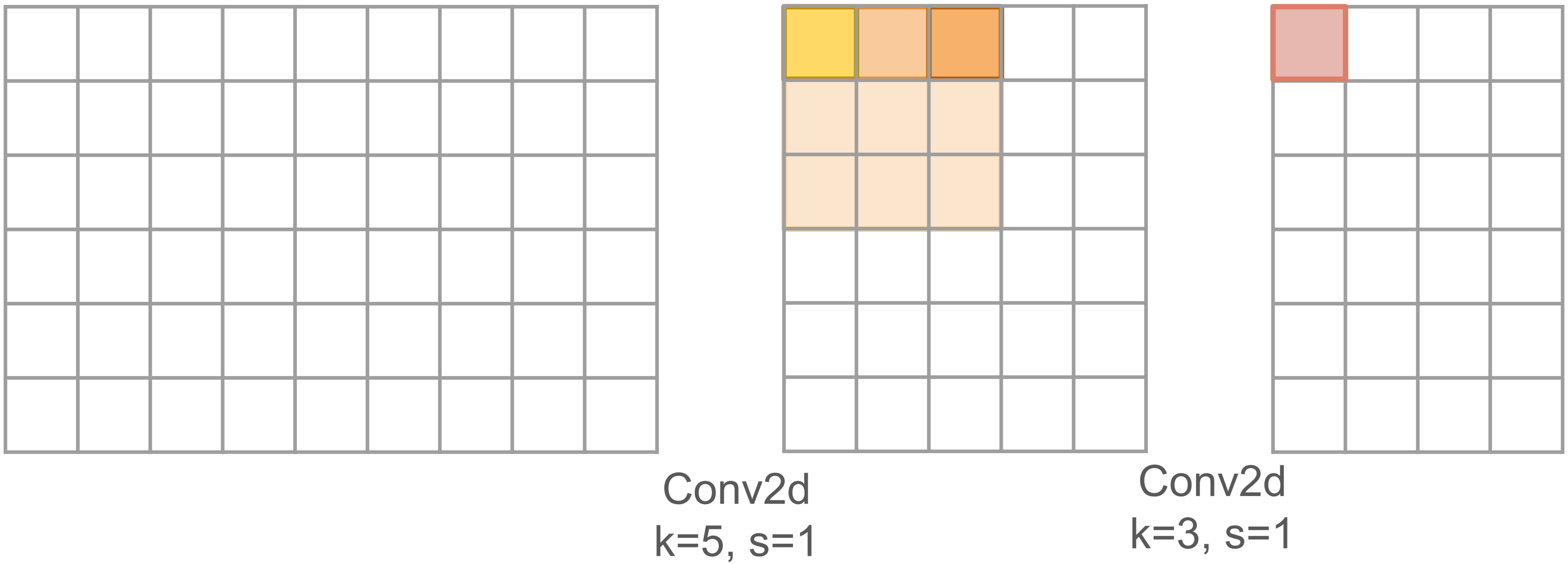


Receptive field size computation

$$n = k + s(m - 1)$$

k=3, s=1, m=1

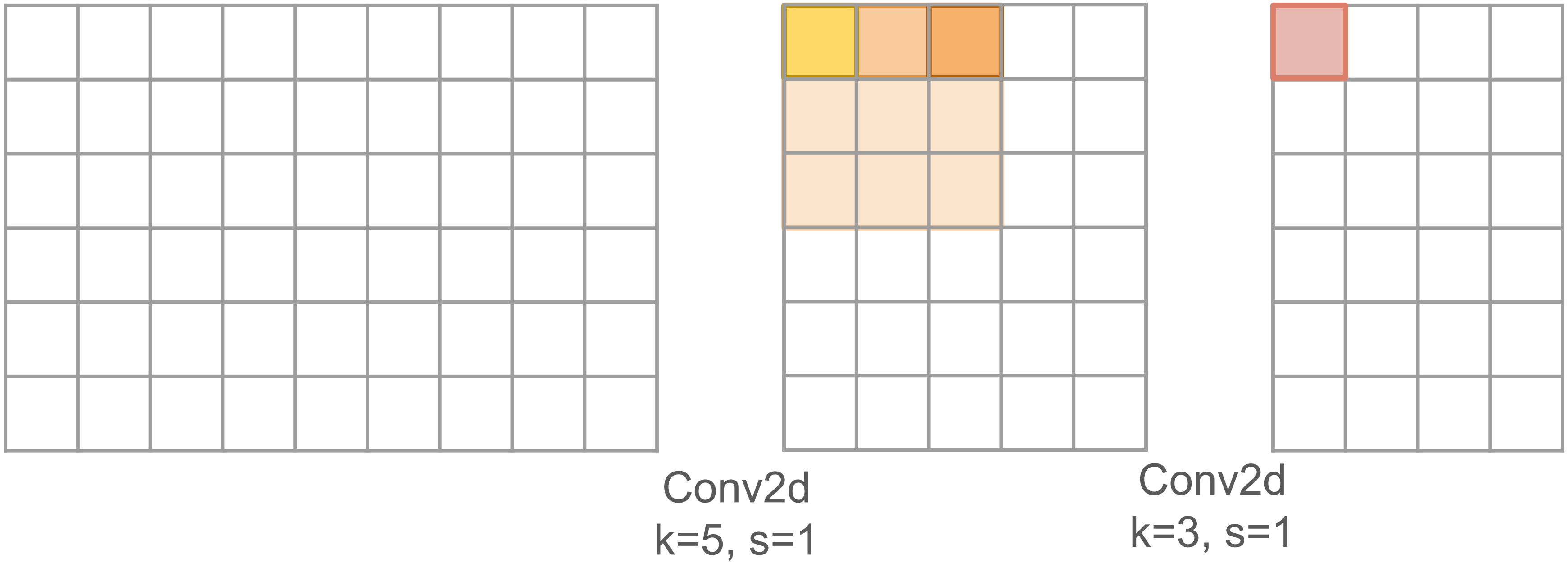
Intermediate, n=3



Receptive field size computation

$$n = k + s(m - 1)$$

k=5, s=1, m=3

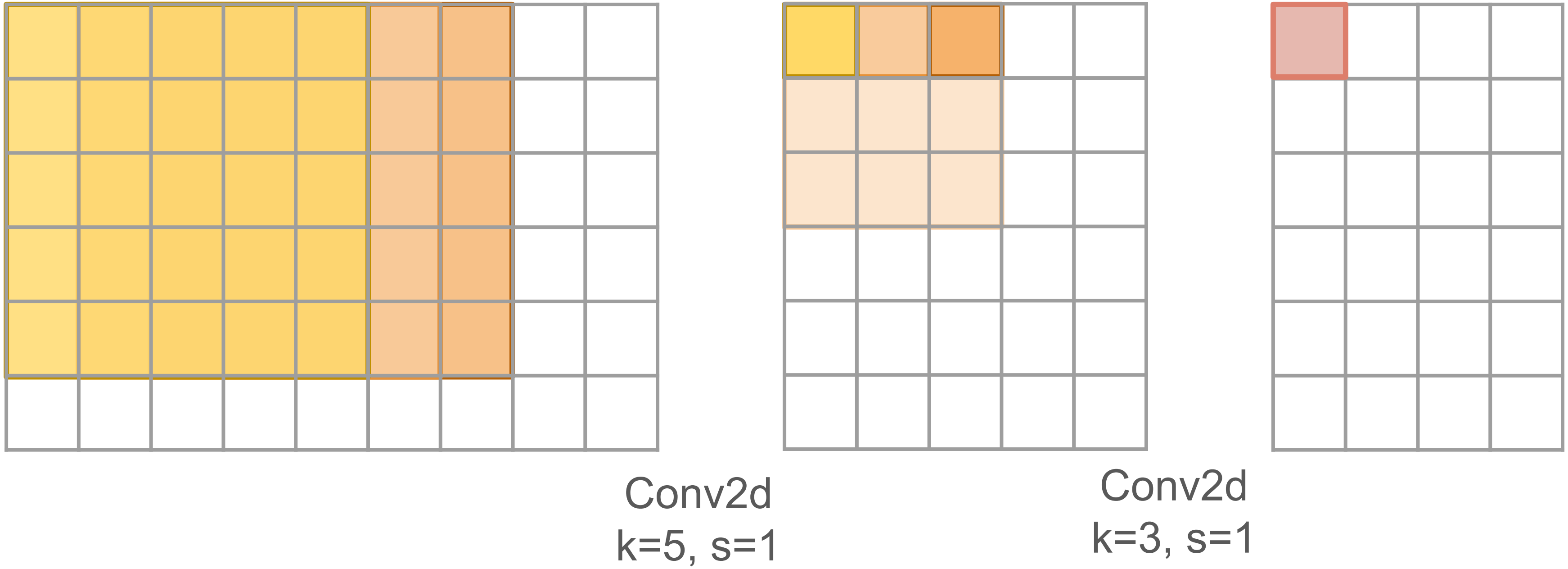


Receptive field size computation

$$n = k + s(m - 1)$$

k=5, s=1, m=3

Receptive size,
n = 7



Batch Normalization for ConvNets

Batch Normalization for **fully-connected** networks

$$\mathbf{x}: N \times D$$

Normalize

$$\boldsymbol{\mu}, \boldsymbol{\sigma}: 1 \times D$$

$$\gamma, \beta: 1 \times D$$

$$y = \gamma(x - \mu)/\sigma + \beta$$

Batch Normalization for **convolutional** networks (Spatial Batchnorm, BatchNorm2D)

$$\mathbf{x}: N \times C \times H \times W$$

Normalize

$$\boldsymbol{\mu}, \boldsymbol{\sigma}: 1 \times C \times 1 \times 1$$

$$\gamma, \beta: 1 \times C \times 1 \times 1$$

$$y = \gamma(x - \mu)/\sigma + \beta$$

Layer Normalization

Batch Normalization for fully-connected networks

$$\mathbf{x}: N \times D$$

$$\boxed{\begin{array}{l}\text{Normalize} \downarrow \\ \boldsymbol{\mu}, \boldsymbol{\sigma}: 1 \times D\end{array}}$$

$$\gamma, \beta: 1 \times D$$

$$y = \gamma(x-\mu)/\sigma+\beta$$

Layer Normalization for fully-connected networks
Same behavior at train and test!
Can be used in recurrent networks

$$\mathbf{x}: N \times D$$

$$\boxed{\begin{array}{l}\text{Normalize} \downarrow \\ \boldsymbol{\mu}, \boldsymbol{\sigma}: N \times 1\end{array}}$$

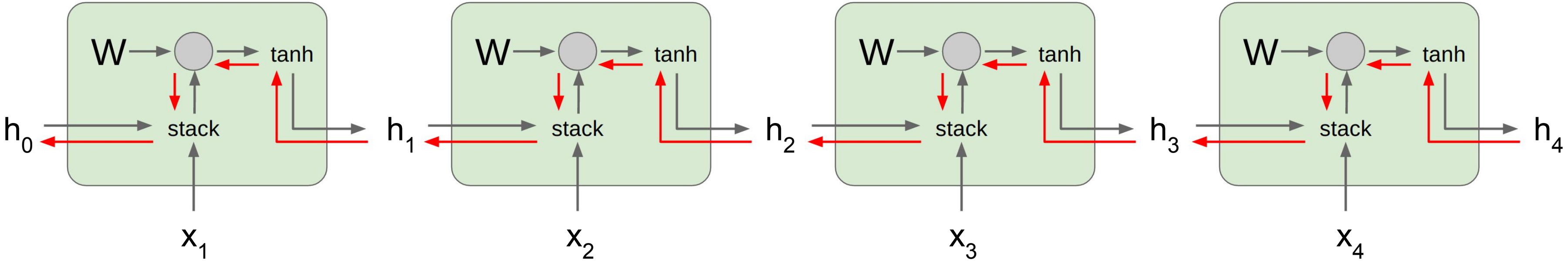
$$\gamma, \beta: 1 \times D$$

$$y = \gamma(x-\mu)/\sigma+\beta$$

Ba, Kiros, and Hinton, “Layer Normalization”, arXiv 2016

Vanilla RNN Gradient Flow

Bengio et al, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 1994
Pascanu et al, “On the difficulty of training recurrent neural networks”, ICML 2013



Computing gradient of h_0 involves many factors of W
(and repeated tanh)

Vanishing/Exploding Gradient

Vanishing Gradient:

- Gradient becomes too small

Vanishing/Exploding Gradient

Vanishing Gradient:

- Gradient becomes too small
- Some causes:
  - Choice of activation function
  - Multiplying many small numbers together

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T}\left(\prod_{t=2}^{T} \boxed{tanh'(W_{hh}h_{t-1} + W_{xh}x_t)}\right)W_{hh}^{T-1}\frac{\partial h_1}{\partial W}$$

Vanishing/Exploding Gradient

Vanishing Gradient:
- Gradient becomes too small
- Some causes:
  - Choice of activation function
  - Multiplying many small numbers together

$$\frac{\partial L_T}{\partial W} = \frac{\partial L_T}{\partial h_T}\left(\prod_{t=2}^{T} \boxed{tanh'(W_{hh}h_{t-1} + W_{xh}x_t)}\right)W_{hh}^{T-1}\frac{\partial h_1}{\partial W}$$

Exploding Gradient:
- Gradient becomes too large
- Gradient clipping: Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

Solution

Modified versions of the RNN cells like,

- LSTM
- GRU

Vanishing gradient problem is controlled by computing some intermediate values also called as gates, that give greater control over the values you want to write, pass to next time step and reveal as hidden state.

For exploding gradients, gradient clipping is a standard technique.

All the best for your midterm!

Sample Problem

3. A max pooling layer in a ConvNet:

   (a) Is approximately as fast to compute in both forward and backward pass as a CONV layer (with the same filter size and strides).

   (b) Is similar to batch normalization in that it will keep all of your neuron activities in a similar range.

   (c) Could contribute to difficulties during gradient checking (higher error than usual, as in the SVM).

   (d) Could contribute to the vanishing gradient problem (recall: this is a problem where by the end of a backward pass the gradients are very small)

3. A max pooling layer in a ConvNet:

(a) ~~Is approximately as fast to compute in both forward and backward pass as a CONV layer (with~~ the same filter size and strides).

(b) Is similar to batch normalization in that it will keep all of your neuron activities in a similar range.

(c) Could contribute to difficulties during gradient checking (higher error than usual, as in the SVM).

(d) Could contribute to the vanishing gradient problem (recall: this is a problem where by the end of a backward pass the gradients are very small)

3. A max pooling layer in a ConvNet:

   (a) ~~Is approximately as fast to compute in both forward and backward pass as a CONV layer (with~~ the same filter size and strides).

   (b) ~~Is similar to batch normalization in that it will keep all of your neuron activities in a similar~~ range.

   (c) Could contribute to difficulties during gradient checking (higher error than usual, as in the SVM).

   (d) Could contribute to the vanishing gradient problem (recall: this is a problem where by the end of a backward pass the gradients are very small)

3. A max pooling layer in a ConvNet:

   (a) ~~Is approximately as fast to compute in both forward and backward pass as a CONV layer (with~~ the same filter size and strides).

   (b) ~~Is similar to batch normalization in that it will keep all of your neuron activities in a similar~~ range.

   (c) Could contribute to difficulties during gradient checking (higher error than usual, as in the SVM).

   (d) Could contribute to the vanishing gradient problem (recall: this is a problem where by the end of a backward pass the gradients are very small)

3. A max pooling layer in a ConvNet:

   (a) ~~Is approximately as fast to compute in both forward and backward pass as a CONV layer (with~~ the same filter size and strides).

   (b) ~~Is similar to batch normalization in that it will keep all of your neuron activities in a similar~~ range.

   (c) Could contribute to difficulties during gradient checking (higher error than usual, as in the SVM).

   (d) Could contribute to the vanishing gradient problem (recall: this is a problem where by the end of a backward pass the gradients are very small)