CS182/282A Designing, Visualizing & Understanding Deep Neural Networks SP2019

Midterm 1

This is a closed-book exam with 7 questions. You can use one two-sided sheet of notes. Please write all your answers in this book. There are a total of 80 points for all questions, and you have 80 minutes to complete the exam. Please budget your time accordingly. Good luck!

NAME_____ **SID**_____

NAME TO YOUR LEFT_____

NAME TO YOUR RIGHT_____

1. **(40 points)** Give short answers for each of the following questions – 2 points each

a) Describe one architectural difference (other than difference in number and size of its layers) between the LeNet text recognizer from 1989 and AlexNet from 2012. (2 points)

One difference is that AlexNet used ReLU instead of Sigmoid non-linearities. (If you only said "ReLU" without mentioning "sigmoids" or the actual network names, you needed to say a little more to get full credit.)

b) What is multi-task learning, and why does it often improve the performance of deep network models? (2 points)

Multi-task learning is when networks are applied to multiple tasks with some shared representation. It usually refers to when layers (typically those nearest the input) are shared.

The shared part of the network is trained on data for multiple tasks, typically with different output modules (sets of neurons) specialized to each task. But the shared part can be re-used and provide extra information to different tasks due to capturing the essence of the task similarity. Finally, another possibility is to avoid overfitting to a particular task as the network is forced to learn more general, robust features.

c) Give one advantage of generative ML models and one advantage of discriminative models. (2 points)

Generative advantage: can sample ("generate") data from distribution, insights into the distribution. Don't just say "it models p(x,y)".

Discriminative advantage: requires fewer assumptions about data, may model more complex distributions (don't just say 'easier to train').

d) Squared error is a loss commonly used for real-valued predictors. What loss is commonly used with predictors of discrete values? (2 points)

Cross entropy loss (or binary case, log loss...)

e) Like SVMs, logistic regression classifiers typically exhibit a margin around their decision boundary. Explain why with a picture. (2 points)

Picture should be similar to slide 36 in lecture 3: Two sets of points on either side of a decision boundary. Loss is non-zero even at the boundary and for points on either side of it. The loss gradients "push" the boundary to move points outside the margin.

f) Why is there a tradeoff between the bias and variance of a prediction model? Where do deep networks lie on the spectrum of this tradeoff? (2 points)

Tradeoff is due to model complexity: more complex models can fit data better (lower bias) but do so at the expense of sensitivity to data (higher variance). Regularization reduces variance at the expense of bias.

Deep networks lie at the low-bias, high variance end of the spectrum. You didn't get credit if you said neural nets tried to 'balance' the spectrum; this misses the idea.

g) When would you expect a naïve bayes model to be more accurate than a logistic regression model? When would you expect the logistic regression model to be more accurate? (2 points)

Naive Bayes makes the strong assumption that feature values are conditionally independent given the class label. It is very accurate when this assumption holds. Don't say that features are independent --- they are *conditionally* independent based on the *class*.

Logistic regression makes no assumption about feature independence and thus is more accurate if features are not conditionally independent.

h) For a function $f(X)$, at what kinds of points does the gradient of f vanish? List all that apply. (2 points)

Local optima, saddle points.

i) If an input data block in a convolutional network has dimension $C \times H \times W = 96 \times 128 \times 128$, (96 channels, spatial dim 128x128) and we apply a convolutional filter to it of dimension $D \times C \times H_F \times W_F = 128 \times 96 \times 7 \times 7$, (i.e. a block of D=128 filters) with stride 2 and pad 3, what is the dimension of the output data block? (2 points)

128 x 64 x 64, assumes a floor in the division.

j) With the aid of a picture, explain how cross-validation is used to estimate model performance. (2 points)

Divide data into k equal-sized, disjoint folds

At step i, hold out fold i, train on other folds, then check on data on fold i (which is validation data).

k) Given that matrix multiplication is associative and the derivative of the loss with respect to a weight layer in a deep network is a product of Jacobian matrices, why is the derivative computed backwards from the output (i.e. using backpropagation)? (2 points)

Derivative is a product of jacobians. The last one (loss) is a vector .

Going output-to-input involves only matrix-vector multiplies (and complexity is proportional to the total model complexity). Going input-to-output involves matrix-matrix multiplies which are much more expensive.

l) Consider the image below. Apply max pooling to it with a 2x2 max pool filter with stride 2 and no padding. Draw the result to the right of the image. (2 points)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Correct answer is

6 8

14 16

m) ReLU layers have non-negative outputs. Given one negative consequence of this. Give an example of another layer type that was developed to address this issue? (2 points)

Negative consequence: Gradient of a weight neuron in the next layer is either all-positive or all-negative.

One of: ELU, Leaky-RELU, PRELU, Maxout

n) What is Xavier initialization and why is it used? (2 points)

Xavier: initialize weights with random values with zero mean and variance 1/F where F is fan-in. F = Fh x Fw x Cin for a conv layer, or Cin for a fully-connected layer.

Rationale (with definition): Initial activations and gradients depend on weights. If weights are badly designed, gradients may explode or vanish.

o) What other benefits does batch normalization exhibit, besides scaling/normalizing activations? (2 points)

Can correct for bad weight initialization,

Improves gradient flow/Clips Gradients/Stabilizes Learning

p) What is inverted dropout and what is its advantage? (2 points)

Scale activation by 1/p during training where p is the probability the neuron is kept.

The scaling preserves the expected value of the output. OR (with definition): No modifications necessary at test time

q) What are bagging and boosting? Which method is more commonly used with deep networks and why? (2 points)

Bagging; Bootstrap aggregation: models are trained on different bootstrap re-samples of a dataset. Reduces variance but not bias compared to the base model.

Boosting: A hierarchy of models where mis-classified examples receive a higher weight in later models. Reduces bias and possibly variance.

Bagging is typically used with deep networks because the base models have low bias but high variance. Also the bagging approach is trivially parallel, which is important when deep model training is very expensive. (or other assorted reasons)

r) What is the main difference between Fast R-CNN and Faster R-CNN? What's the reason for this difference? (2 points)?

Faster R-CNN uses a region proposal neural network after the CNN feature map instead of an external region proposal network.

Advantages: higher speed (about 10x), same accuracy; End-to-end training; RPN is learned.

s) What is a limitation of vanilla recurrent networks (RNNs) and how is this addressed in LSTM networks? (2 points)

Limitation: exploding and/or vanishing gradients

LSTM carefully manages gradients through the memory cell; Memory cell creates shortcuts to avoid exploding/vanishing gradients through time.

t) When using activation maximization to visualize a neuron, why is initialization of the image important? (2 points).

A good initialization (for example, initializing on the image manifold) can contribute to more human interpretable outputs.

2. **(6 points)** A soft-margin SVM minimizes the following loss function:

$$L = \sum_{i=1}^{n} (0, 1 - y_i f(x_i)) \ + \lambda ||w||^2$$

where $f(x) = w^T x + b$, and $y_i \in \{-1, 1\}$ is a label, and n is the number of training samples.

a) Ignoring the regularizing term (with factor λ), how does the loss vary as a function of $||w||$?

b) How does the margin of this classifier vary as a function of λ ?

c) What is the derivative of the loss with respect to w ?

a) Assume the weight is scaled by a scalar $w = \alpha v$ and substituting:
$L = \sum_{i=1}^{n} \max(0, 1 - y_i(b + \alpha v^T x_i))$
then consider $\alpha = 0$, the loss is
$L = \sum_{i=1}^{n} \max(0, 1 - y_i b)$
as α increases, the loss is a linear function which either increases or decreases away from zero, according to the sign of
$L = v^T \sum_{i=1}^{n} y_i x_i (0 < 1 - y_i(b + \alpha v^T x_i)$
where $(a < b) = 1$ iff $a < b$, and 0 otherwise.

b) As λ increases the weight vector w decreases (since its share of the gradient is larger), and the margin becomes larger. That's because f(x) computes the distance from x to the decision boundary, and this distance scales with ||w||. When ||w|| is smaller, the point must be further from the decision boundary to produce the same f(x).

c) **The loss is:**
$L = \sum_{i=1}^{n} \max(0, 1 - y_i(b + w^T x_i)) + \lambda ||ww||^2$
and the derivative inside the max function is $-y_i x_i$.
This derivative is restricted by the max function to values where its second argument is positive. So the full derivative is
$\frac{dL}{dw} = \sum_{i=1}^{n} -y_i x_i (y_i f(x_i) < 1) + 2\lambda w$

3. **(8 points)** A deep network implements a parametric function $f(X, W)$ where X is the input and W is a vector of weights. Assume that the loss being optimized is symmetric (spherical) about an optimum weight W_0.
   Now consider a function $g(X, V) = f(X, D\,V)$ where D is a diagonal matrix, with $D_{\{ii\}} = i$. Suppose you have successfully trained the function f to some loss L_{final} after N iterations. What if anything can you say about training g assuming both networks were trained with:
   a) Vanilla SGD (no momentum)
   b) SGD with momentum
   c) RMSprop
   d) ADAM

a) **Vanilla SGD should take longer to reach L_{final}, or have higher loss after N iterations. While gradients for f point directly at the optimum (spherical loss about W_0), gradients for g do not (the loss for g is ellipsoidal). SGD without momentum follows the gradient flow. Its also possible that SGD will oscillate at high learning rates on g, making slow overall progress.**

b) SGD with momentum should do a better job optimizing g than vanilla SGD. At high learning rates, oscillatory gradients should cancel each other, leading to a net gradient that is close to optimal. Loss after N steps may be close to L_{final}, and time to reach L_{final} may be close to N when optimzing g with SGD with momentum.

c) **For RMSprop, gradients are diagonally scaled by inverse gradient magnitude. Since the gradients for f and g are diagonally scaled first, the final gradients are the same, and convergence rate and final loss will be the same.**

d)

For Adam, gradients are diagonally scaled by inverse gradient magnitude. Since the gradients for f and g are diagonally scaled first, the final gradients are the same, and convergence rate and final loss will be the same.

4. **(8 points)** A deep network contains a new layer type called "BiasScale" that implements the following operation:

$$y_i = s_i x_i + b_i$$

where y is an output vector, x is the input vector, and s and b are scale and bias weight vectors respectively. Subscripts denote the i^{th} elements of these vectors, which are of equal dimension.

Given the output loss gradient

$$\frac{dL}{dy_i}$$

compute the loss gradient wrt to the parameters s and b.

Would it make sense to place this layer before or after an affine (matrix multiply) layer? Why?

Gradient with respect to s_i is

$$\frac{dL}{ds_i} = \frac{dL}{dy_i} x_i$$

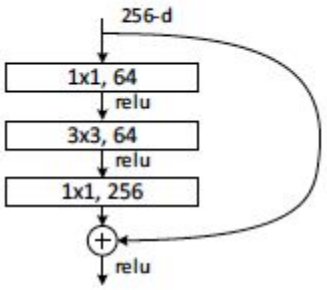
Gradient with respect to b_i is

$$\frac{dL}{db_i} = \frac{dL}{dy_i}$$

No, it does not make sense to place this layer either before or after an affine layer.

The affine layer is capable of element-wise scale and bias. If there were any loss reduction possible by these changes, the weight layer should learn them.

5. **(6 points)** A residual network features bottleneck layers like the one below that use 1x1 convolutions. What is the effect of this design on:
    a) Training and test time (forward and backward passes)
    b) Model complexity (number of parameters)
    c) Overall network accuracy



a) The bottleneck reduces the number of channels which speeds up both forward and backward passes. Convolution cost is proportional to the product of number of input and output channels.

b) Model complexity grows as the product of number of input and output channels, and is much lower for bottleneck layers.

c) Accuracy may increase or decrease. Non-bottleneck layers have more parameters and therefore potentially lower bias if there is enough training data. On the other hand, the lower parameter count for bottleneck reduces the risk of overfitting on modest-sized training datasets.

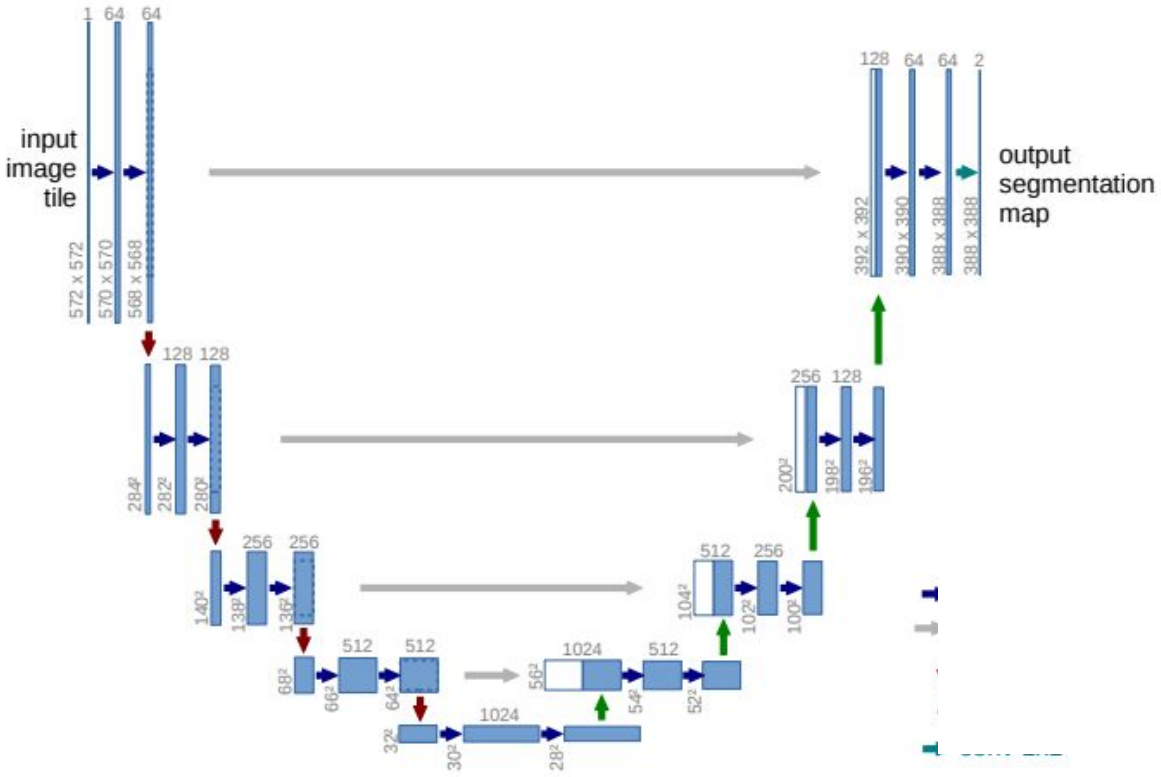
6. **(6 points)** Consider the following collections of models. For each collection, say how would you combine their outputs to improve accuracy: prediction averaging or parameter averaging, and then say what kinds of improvements you would expect to see: reduced bias or reduced variance:
    a) An ensemble of CNN models, all with the same number and dimensions of layers, with different initialization.
    b) An ensemble of CNN models, with different numbers and dimensions of layers.
    c) A series of checkpoints (snapshots) of the same model at different points during training. Assume training has reached the optimum loss, but is continuing.

a) We can bag the ensemble of CNN models, combining with prediction averaging. Since models are the same, bias is not reduced but variance should be.

b) We should combine these models via prediction averaging. This should produce both a bias reduction (the models are different) and a variance reduction.

c) Parameter averaging is suitable here, and much more efficient. The model is the same, so there can only be variance reduction.

7. **(6 points)** The figure below shows a U-Net which is used to compute a semantic segmentation of an image.
    a) What operations are represented by the upward arrows in the figure ?
    b) What is the role of the rightward arrows ?
    c) Which of the transformations (arrows in any direction) in the network have learned parameters ? Circle all the arrows that have learned parameters.



a) Up-sampling or strided transposed convolution.

b) The long rightward arrows are residual connections going from down-sampled images to corresponding-sized upsampled images. The small rightward arrows represent unstrided convolutions.

c) Upward arrows are strided transposed convolutions which have learnable parameters. Small right arrows (convolutions) have learnable parameters.