CS 182/282A Designing, Visualizing and Understanding Deep Neural Networks
Spring 2021 Sergey Levine **Discussion 5**

This discussion covers vision applications and introduction to visualizing networks (style transfer, Dream Machine)

1 Review of Vision Problems

For most of the class thus far, when we discuss applying neural networks in practice to vision applications, we have largely assumed an image classification task. That is, given an image, we let the network output the probabilities of the true label belonging to a variety of classes.

However, there are more types of standard computer vision problems, namely, *object localization, object detection, semantic segmentation.* Below, we outline the four main types of computer vision problems.

Image Classification Given an image, we would like the network output the probabilities of the true label belonging to a variety of classes. This type of problem was the main focus of the course so far.

Object Localization Determine a *bounding box* for the object in the image that determines the class. In this type of problem, only one object is involved, and indeed, we know ahead of time that there is only one object class of interest in the image. Often, the bounding box objective may be simultaneously trained with the classification objective, resulting in a loss objective that is the sum of the two loss terms, the L_2 and the cross-entropy loss, respectively.

Object Detection Determine multiple objects in an image and their bounding boxes, with performance measured by *mean average precision* (mAP). There may be many objects, and several instances of the same object class (for e.g., several dogs) in the same picture. This means that, in contrast to image classification where the network only has to identify one object, the network has to predict a varying number of bounding boxes. In literature, object detection can be solved using Faster R-CNN.

Semantic Segmentation Label every pixel in the image. Here, we can naively run a CNN classifier for each pixel. However, better solutions, like `UNet`, exists in literature. *Semantic* segmentation means we do not worry about distinguishing between different instances of a class, in contrast to the aptly-named *instance* segmentation problem.

Problem 1: Calculate mAP

A common metric for object detection is *mean average precision* (mAP), where we compute average precision (AP) separately for each class, then average over classes. We say that a detection is a true positive if it has IOU (Intersection over Union) with a ground truth box greater than some threshold, and we can calculate the AP as the area under the precision / recall curve for each class.
You run an object detector, and get the following results:

Example	Predicted/Ground Truth IOU
A	0.29
B	0.11
C	0.701
D	0.001
E	0.92
F	0.45

If all candidates are true positives and we threshold the IOU at 0.5, what is the Average Precision of our object detector? What is the Mean Average Precision (mAP) when using the thresholds $(0, 1)$? Please note that we did not explicitly cover mAP in lecture.

2 Object Detection: R-CNN

Faster R-CNN is a popular technique for object detection problems, and stands for Faster *Region*-CNN. Faster R-CNN uses these regions as areas in the image that are likely to contain objects. More precisely, a *Region Proposal Network* predicts proposals from CNN features. The CNN features were obtained from passing the original input image through several convolutional layers.

The network is trained jointly using four losses, which normally means adding up the objectives (possibly with different weights).

3 Segmentation: Transposed Convolution & U-Net

We briefly comment on an operator called the *transpose convolution*, because it's often used for *upsampling* a convolutional neural network during segmentation tasks. This operator increases the resolution of the intermediate tensors, which we often want if we want the output of our network to be an image (e.g., of the same size as the input images). Note that early convolutional and pooling layers tend to *downsample* or reduce the size of tensors. Please note that it is sometimes referred to as a *deconvolution* operator, but it is not the preferred wording because it is an overloaded term with other definitions commonly used.

The transpose convolution can be thought of as flipping the forward and backward passes of the convolution step. In addition, the naming comes from how it can be implemented in a similar manner as in convolution but with the weight matrix transposed (along with different padding).

Convolutional layers typically downsample images spatially but sometimes we want to upsample. For example in semantic segmentation or in DCGAN where we generate images from random noise of a lower dimension.

Problem 2: 2D Transpose Convolution Mechanics

Let our input be

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$$

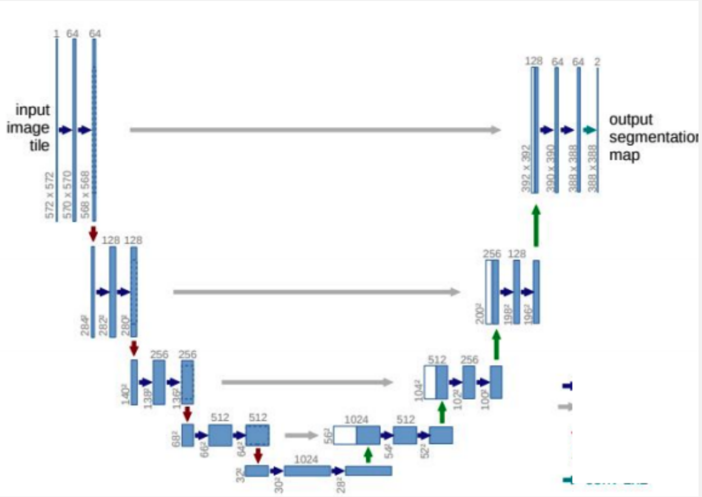
and kernel be

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$$

Assume input, output channels of 1, padding of 0 and stride of 1, what is the output of transposed convolution layer?

Problem 3: U-Net Potpourri

The figure below shows a U-Net architecture.



Answer the following questions,

(a) What operations are represented by upward arrows in the figure?

(b) What is the role of rightward arrows?

(c) Which of the transformations in the network have learned parameters?

4 DeepDream

The main idea behind DeepDream is to exaggerate details in an image that look like recognizable objects. Simply put, the procedure for DeepDream is as follows,

1. Pick a layer
2. Forward propagate to the layer
3. Set the gradient at that layer to the activation at that layer
4. Backpropagate to update the image

5 Style Transfer

Style transfer is a class of algorithms to manipulate digital images, or videos, in order to adopt the appearance or visual style of another image.

We assume an input image p and a style image a. Then, the image p is fed through a CNN (in the original paper, through a VGG-19 architecture), and network layer activations are sampled at the early to middle layers of the CNN. The style image a is also fed through the same CNN, and network activations are sampled at early to middle layers of the CNN, and are encoded into a Gram matrix, $S(a)$. The Gram matrix encodes the correlation between different features.

Then, the goal of style transfer is to synthesize some output x, such that $C(x)$ approximates $C(p)$, and $S(x)$ approximates $S(a)$. Then, our loss function is,

$$\mathcal{L}(x) = \|C(x) - C(p)\|_2 + k\|S(x) - S(a)\|_2$$