CS 482/682 Machine Learning: Deep Learning
Mock Exam – Fall 2023

Note: Please note that the primary purpose of this mock exam is for you to review your understanding of important concepts taught in this class. Specifically, the mock exam **is not** representative of the true midterm exam with respect to length or composition of questions.

Question:	1	2	3	4	Total
Points:	12	10	15	0	37
Score:					

Overall Problem Setting

Please read this part carefully before starting work on the individual questions. The information provided on this page applies to every single question in this exam. It will be a good idea to ensure that this page and the information herein is easily available to you in case you want to double check any information.

Your Role

Upon your graduation from JHU you have been hired as an applied machine learning engineer in an up-and-coming startup company that focuses on building advanced algorithms for autonomous driving. The company has just started working on a new exciting project that seeks to dramatically enhance its portfolio with new deep learning-enabled products. This is why you were hired!

Your Tasks

Your job is to lead the development of these deep learning algorithms. Your responsibilities will include:

1. Pre-processing (Q1)
2. Developing algorithms for determining which country the car is in (Q2);
3. Consider pedestrian and car detection to enhance safety (Q3).
4. Improvement of object detection module using new architectures (Q4).

While your team of highly qualified interns will be able to handle most issues arising during these tasks independently, they will have several questions for you that you must answer adequately to ensure that generalizable and performant deep learning algorithms are shipped to your customers.

The Dataset

Of course, your company has been in the self-driving sector for some time and has collected relevant data for you to use in these projects. Specifically, you are provided with the following:

- A dataset with 10,000,000 images of street scenes acquired in 10 different countries (1,000,000 images per country). It is safe to assume that these images are independent (i. e., not consecutive frames from videos) and well capture the expected variation (i. e., identically distributed).
- Labels
  - Country labels in one-hot encoding (one label per image)
  - Segmentations for the "car" class (one segmentation map per image)
- Every image is 1024×1024 pixels encoded in 8-bit RGB (i. e., 3 channels with 256 intensity values $\in [0, 255]$).

An Important Reminder

Every question and part may have several sub-questions, indicated by keywords such as "name, explain, describe, ...". Make sure your answer addresses <u>all</u> of these points.

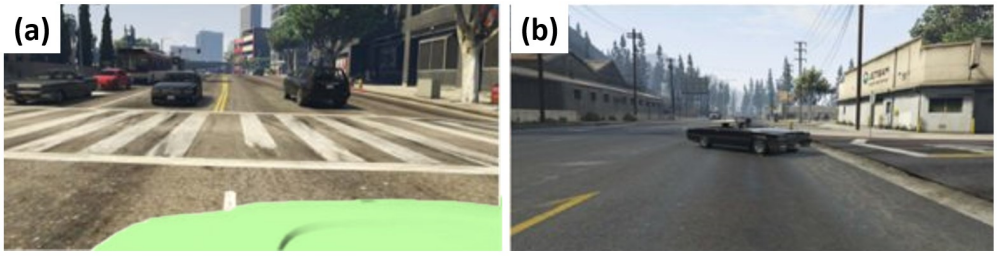

Figure 1: Representative images from (a) the initial self-driving dataset and (b) the final camera system. Both images represent the same country. Note how in images from the initial dataset shown in (a) the brightly colored hood of the car is visible.

1. **Pre-processing**

   (a) (12 points) One of your interns notices that the images are in RGB, but remembers that normalization is beneficial. Unfortunately, they are no longer sure how to do this correctly. Thus, you give the formula for normalizing your input images and describe using a few bullet points how you would normalize images during training. Make sure to introduce all variables and give explanation (e. g., using math) how to calculate them. Then, briefly explain how normalization differs during testing?

   - (4pt) $\tilde{x}_i^{\{r,g,b\}} = (x_i^{\{r,g,b\}} - \mu^{\{r,g,b\}})/\sigma^{\{r,g,b\}}$ (one point for each correct term), where ...
   - (4pt) ...the superscript denotes the channel (either red, green, or blue) as normalization is carried out independently over channels, $\tilde{x}_i^{\{r,g,b\}}$ is the normalized intensity value, $x_i^{\{r,g,b\}}$ is the unnormalized intensity value, $\mu^{\{r,g,b\}}$ is the channel mean, and $\sigma^{\{r,g,b\}}$ is the channel variance (one point for each correctly introduced variable)
   - (2pt) Important to note: μ, σ are calculated over *training set only*
   - (2pt) During testing, the same normalization is applied however we still use the mean and variance from the training set.

2. **Classification**

   To understand which traffic rules to implement, your team is developing a deep learning algorithm that classifies images with respect to the country that they were taken in.

   (a) (10 points) When collecting a first and smaller version of the initial dataset described on the first page (only 300,000 images of three countries: Germany, US, and Japan) only a single car was equipped with the camera for data collection: A violet Chevrolet in the US, a green Mercedes in Germany, and an orange Toyota in Japan. Further, as can be seen in Fig. 1(a), the camera was angled such that the hood of the car was well visible in all images. *Your team trained an initial model – Task Model 1 – on this dataset, achieving very high accuracy for the country classification task.*

   Shortly thereafter, the company started collecting the final dataset described on the first page. For this dataset, camera systems were installed in many different cars. Further, the camera system no longer took images of the hood of the car (see Fig. 1(b). To validate your initial model, your team applied Task Model 1 to this newly collected data of the three countries. Unfortunately, the performance you observe is dramatically worse than what you reported on the initial dataset.

   First, explain why you observe deteriorated model performance when applying Task Model 1 trained on the initial data to the newly collected dataset and name the phenomenon you observe (Hint: We are looking for answers around possible spurious associations.). Then, describe another scenario we have discussed in class where this problem was observed, and briefly discuss how it was addressed. Finally, describe a strategy that might allow you to train a model on the initial data (the one shown in Fig. 1(a)) such that it could generalize to images of the new dataset (like the ones shown in Fig. 1(b)).

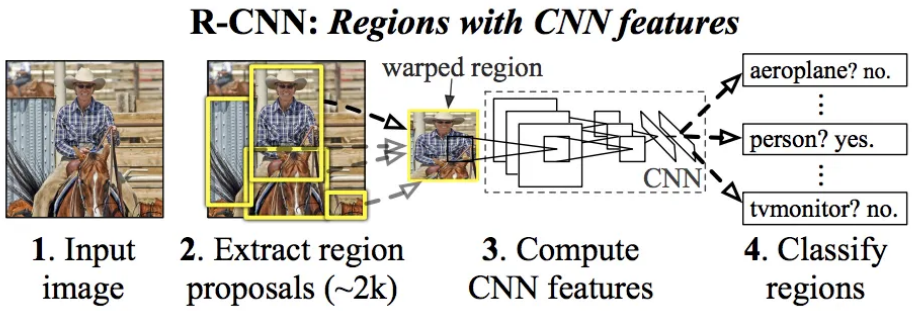

Figure 2: An ineffective approach to object detection: R-CNN.

- (5pts) There is a shortcut during training where the model can distinguish labels simply from the color of the car instead of the street scene. However, this shortcut doesn't exist in the test scenario.
- (3pts) Any scenario that is reasonable
- (2pts) data augmentation (Color jittering, Random Crop, etc)

3. **Object Detection** An *Object Detection* module enables the self-driving system to recognize surrounding objects, e.g., cars, pedestrians and road signs, and their locations. Your goal now is to lead the development of the Object Detection module. With your team, you identified 100 object classes that are of interest to the self-driving system and should be detected reliably. Thus, the Object Detection module will take an image as its input and output (1) all objects in the image and (2) their (minimal) bounding boxes containing the respective object.

   (a) (15 points) An intern suggests an R-CNN framework to realize the functionality of the module (see Figure 2). Specifically, for an input image, R-CNN

   1. proposes a set of $2,000$ candidate object regions;
   2. then uses a pre-trained CNN network to extract a $4,096$-dimensional feature vector for each candidate region;
   3. - (*training.*) then uses the feature vectors and the labels (i.e., the class of the object within each candidate region) of the candidate regions to fit a linear classification layer;
      - (*inference.*) then feeds the feature vector of a candidate region to the trained linear classification layer to obtain a prediction of the object within the region.

   Your team tests this method but finds both the training and the inference of R-CNN take a very long time. Identify the most important aspect that makes this approach to R-CNNs slow during training and inference, explain why this approach is ineffective, and discuss (in good detail) an idea to improve it.

   This is a partially open question. The following examples are motivated by fast R-CNN and faster R-CNN.

   (5pt) Reason + (5pt) explanation.

   - Training/Inference of R-CNN is slow because the model has to classify $2,000$ region proposals per image.
   - The process of proposing $2,000$ candidate regions is not trained/learned end-to-end, and bad candidates can exist that slow down the follow-up process.
   - The feature extraction is done separately for each candidate region, the time of which accumulates when the number of candidate regions is large.

   (5pt) How to improve

- Feeding the image instead of each region to the CNN for extracting the feature. Then applying RoI pooling to connect the feature with the regions.
- Region proposal network for proposing less amount but more useful candidate regions.

4. **Enhancing Object Detection**

   Your team managed to find a way to speed up the R-CNN-based object detection module. Your team presented an example of the result after they trained and fine-tuned the network for car detection, as shown in Fig. 3 below.

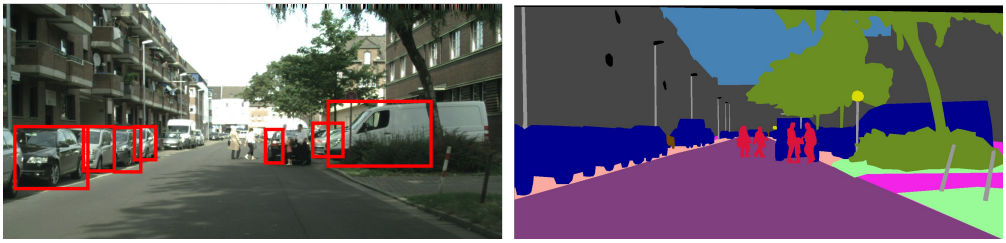

Figure 3: An example of the R-CNN result for car detection. The left image is the output from your model (red bounding box), and the right image is the ground truth semantic segmentation for various labels (the car label is represented with the color blue.

However, you realize that there are several issues in the detection model. The model is unable to correctly identify the boundaries of cars that are occluded by objects like pedestrians and trees (as shown in the center and right of the image). The model also performs worse in identifying cars at longer distances (as shown on the left of the image).

To address this issue, you propose to utilize a new architecture called Feature Pyramid Networks (FPN). FPN is an architecture that builds a multi-level pyramid of features. Each level corresponds to a different resolution and it enhances the standard CNN with a top-down architecture with lateral connections. An example of the architecture is shown in Fig. 4 below.

(a) (6 points) You try to convince your team this new FPN model is capable of solving the issues identified in Fig.3. Based on the architecture description in Fig.4, please explain how the various portions of the FPN architecture (e.g. different pathways, use of 1x1 conv layers or residual blocks) could potentially help: (1) identifying cars at longer distances (2) improving detection boundaries for cars occluded by various objects.

Solutions:

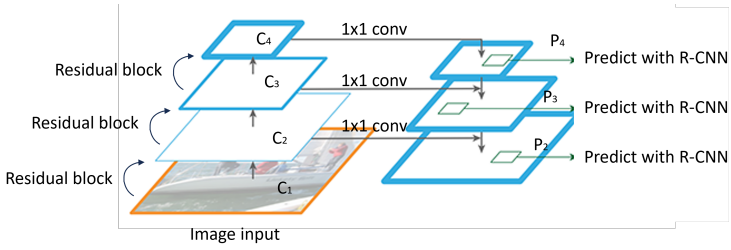

Figure 4: The FPN architecture you proposed. The input image will be passed through residual blocks, creating feature maps at different scales (C_2 to C_4) in the bottom-up pathway on the left. In the top-down pathway on the right, we use simple upsampling convolutional layers and then merge them with the corresponding lateral feature map (e.g. C_4) through a 1x1 convolutional layer, creating enhanced feature maps (P_4 to P_2), which you will use together with the R-CNN model trained earlier for prediction.

```python
class C2ResidualBlock(nn.Module):

    def __init__(self):
        super(C2ResidualBlock, self).__init__()
        self.downsample = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=1, stride=2),
            nn.BatchNorm2d(128)
        )
        self.conv1 = nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1)
        self.bn1 = nn.BatchNorm2d(128)
        self.relu = nn.ReLU()
        self.conv2 = nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1)
        self.bn2 = nn.BatchNorm2d(128)

    def forward(self, x):

        residual = self.downsample(x)

        out = self.conv1(x)
        out = self.bn1(out)
        out = self.relu(out)
        out = self.conv2(out)
        out = self.bn2(out)

        out += residual
        out = self.relu(out)
        return out
```

Figure 5: The sample code your team created for the residual block between C_1 and C_2.

- Identifying cars at longer distances: FPN generates a pyramid of feature maps at different scales (P_4 to P_2), which are then used to detect the objects using R-CNN from the previous task. Cars at longer distances are rather small in the image. If FPN is employed, we can use R-CNNs with smaller region sizes across all feature maps, so that the model works well on detecting smaller objects on higher-resolution images, but will still work well detecting larger objects (which will be picked up in high-level feature map like P_4).
- Improving detection boundaries for occluded cars: The same principle still applies here. With the introduction of a multi-level feature pyramid, it allows the car to be detected at the appropriate level. The use of residual blocks and the 1x1 convolutional blocks will also help discard unnecessary features (like poles, trees and pedestrians in this case) and help the network learn identity mappings, improving the detection boundaries.

(b) (6 points) Your team created a simplified Pytorch implementation of a residual block as shown in Fig. 5 below. You know that the input tensor to this block would be C_1 in Fig. 4, which has a shape of (batch_size, 64, 1024, 1024). Please answer the following:

- Describe two advantages of using residual blocks with residual connections instead of only using several 2D convolutional layers in sequence.
- What is the expected output shape after passing the input tensor through C2ResidualBlock? If we change the padding in conv1 and conv2 to zero ("valid" padding), what would be the expected output shape? Are there any potential issues using a different padding value?
- If we select a single pixel from the output tensor, what is the receptive field of this pixel in the input tensor?

Solutions:

- Using residual blocks instead of stacks of convolutional layers could:
  1. Prevent vanishing gradient problems with the use of residual connections
  2. Allows the construction of very deep models

3. Easier to train as the residual blocks try to learn the residual functions with reference of layer inputs. The direct path of data flow may also simplify and stabilize the model learning process.

- 1. SAME padding: In the residual path, the downsample layer uses a stride of 2 with the same padding, so the output shape would be (batch_size, 128, 512, 512). This is the same as conv1 layer, which has an output shape (batch_size, 128, 512, 512). The output shape remains the same after conv2, which is (batch_size, 128, 512, 512). The final output shape would be (batch_size, 128, 512, 512). Note that batch normalization does not change the shape of the tensors.
  2. zero padding: If padding is set to 0, the downsampling path is not affected, which has the same output shape as the same padding (batch_size, 128, 512, 512). For conv1, the output size is calculated by floor($\frac{1024-3}{2}$), which is 510, so the output shape is (batch_size, 128, 510, 510). The output shape after conv2 remains the same. The issue here is that if we use zero padding, the residual tensor and the output cannot be directly added in the forward function as they have different shapes, which will raise errors.
- Receptive field:
  1. The receptive field after the downsample layer remains as 1x1.
  2. The receptive field after the conv2 layer becomes 3x3 due to the kernel size used in this layer.
  3. The receptive field after the conv1 layer (with stride 2) should be 7x7 (there are 2 overlapping pixel between 3x3 kernels using stride=2, so it would be 3x3-2 = 7)
  4. The final receptive field of a single output pixel would be 7x7.