DEPARTMENT OF COMPUTER SCIENCE, UCLA
COM SCI 188: COMPUTER VISION

**INSTRUCTOR:** Prof. Achuta Kadambi, Prof. Stefano Soatto
**TAS:** Aditya Golatkar, Albert Zhao, Chinmay Talegaonkar

**NAME:** Your Name
**UID:** Your UID

# HOMEWORK 4

| PROBLEM | TYPE | TOPIC | MAX. POINTS |
|---|---|---|---|
| 1 | Analytical | Shape and Reflectance | 10 |
| 2 | Coding | Photometric Stereo | 15 |
| 3 | Analytical | Machine Learning Basics | 10 |
| 4 | Coding | Training a Classifier | 15 |
| 5 | Interview Questions (Bonus) | Miscellaneous | 15 |

## Motivation

So far we have seen geometric approaches to obtaining 3D features of a scene such as depth and shape. This problem set aims to expose you to a photometric approach to shape estimation, more commonly known as *photometric stereo*. The second half of the problem set gives you a basic exposure to machine learning approaches and techniques used for computer vision tasks such as image classification. You will train a simple classifier network on CIFAR-10 dataset using google colab. We have provided pytorch code for the classification question, you are free to use any other framework if that's more comfortable.

The problem set consists of two types of problems:

- analytical questions to solidify the concepts covered in the class, and
- coding questions to provide a basic exposure to implementing photometric stereo and building a machine learning classifier using pytorch.

*This problem set also exposes you to a variety of machine learning questions commonly asked in job/internship interviews*

## Homework Layout

The homework consists of 5 problems in total, with subparts for each problem. There are 2 types of problems in this homework - analytical and coding. All the problems need to be answered in the Overleaf document. Make a copy of the Overleaf project, and fill in your answers for the questions in the solution boxes provided.

For the analytical questions you will be directly writing their answers in the space provided below the questions. For the coding problems you need to use the Jupyter notebooks (see the Jupyter notebook for each sub-part which involves coding). You are provided with 2 jupyter notebooks, for Problem 2 and Problem 4. After writing your code in the Jupyter notebook you need to copy paste the same code in the space provided below that question on Overleaf. In some questions you are also required to copy the saved images (from Jupyter) into the solution boxes in Overleaf. For the classification question, upload the provided notebook to google colab, and change the runtime type to GPU for training the classifier on GPU. Refer to question 4 for more instructions/details.

## Submission

You will need to make two submissions: (1) Gradescope: You will submit the Overleaf PDF with all the answers on Gradescope. (2) CCLE: You will submit your Jupyter notebooks (.ipynb file) with all the cells executed on CCLE.

## Software Installation

You will need Jupyter to solve the homework. You may find these links helpful:

- Jupyter (https://jupyter.org/install)
- Anaconda (https://docs.anaconda.com/anaconda/install/)

## 1 Shape and Reflectance

### 1.1 Plenoptic function (1.0 points)

How many degrees of freedom does the plenoptic function $L(x, \omega, t, \lambda)$ have. Justify.

### 1.2 Lambertian surface (3.0 points)

At a given pixel $(u, v)$ corresponding to a point on the image of a 3D surface, assume $I(u, v)$ to be the observed intensity, $l$ to the direction of the incident light at the point corresponding to the pixel, and $N(u, v)$ to be a unit vector denoting the surface normal at this point. $N(u, v)$ and $l$ are both $3 \times 1$ vectors.

(i) Assuming a Lambertian surface with an albedo at each point given by $\rho(u, v)$, write an expression for $I(u, v)$.

(ii) In above expression, the lighting vector $l$ and the measured intensity $I(u, v)$ is known to us. How many unknown variables we need to solve for at the pixel $(u, v)$? Specify these unknown variables? The 3 components of the normal vector $N(u, v)$ can be denoted as $N_x(u, v), N_y(u, v), N_z(u, v)$. Remember that $N(u, v)$ is a unit vector.

(iii) What is the minimum number of measurements required to solve for the unknown quantities at a given pixel $(u, v)$

### 1.3 Solving Photometric Stereo (4.0 points)

Having set the stage and notations for photometric stereo in the previous subparts, you will now solve for the unknown surface normals and albedos, thus solving the photometric stereo problem.

(i) Assume that for each pixel $(u,v)$ you obtain $m$ intensity measurements under $m$ different lighting vectors. Let's denote $L$ to be a $m \times 3$ matrix where each row $l$ corresponds to a lighting direction. $I_m(u,v)$ denotes a $m \times 1$ vector of the obtained intensity measurements under $L$. Use $G(u,v) = \rho(u,v)N(u,v)$. $G(u,v)$ denotes all the unknowns we are trying to solve for. Obtain $G(u,v)$ as a function of $I_m(u,v)$ and $L$. Justify your answer briefly.
(ii) Write an expression to obtain $\rho(u,v)$ from $G(u,v)$.
(iii) Having obtained $\rho(u,v)$, now derive an expression to obtain $N(u,v)$ from $G(u,v)$.
(iv) Solving the above 2 parts solves the photometric stereo problem for 1 pixel in the image. A naive and inefficient way to scale this up to the whole image is to sequentially apply this operation for each pixel in the image. Let's denote $\hat{N}$ to be a $3 \times n$ matrix, whose each row is the surface normal at a given point on the object surface. We similarly define a $3 \times n$ matrix $\hat{G}$. $n$ is the total number of pixels in the image. Modify your above answers to obtain $\hat{G}, \hat{N}$, thus solving the photometric stereo for the full image in one go. (Assume $I_m$ to be $m \times n$)

### 1.4 Assumptions to be wary of (2.0 points)

(i) Will the equations derived above work on non-lambertian surfaces? Justify in one sentence.
(ii) Even for a lambertian surface, can you list 2 scenarios in which the above equations may not give exact results. Assuming no standard photometric stereo assumptions are violated, i.e. the camera is fixed w.r.t object.

## 2 Implementing Photometric Stereo (15.0 points)

In this question you will implement the equations derived in the previous problem, thus implementing photometric stereo. You will be using numpy for this part. Please refer to the given jupyter-notebook for the code setup. Follow the instructions in jupyter-notebook to complete the missing parts. For this part, you will use the notebook named PSET4_PS. The mathematical notations will be the same used in problem 1.

### 2.1 Loading data (4.0 points)

(See the jupyter-notebook ) The first step is to images and corresponding lighting vectors for an object. Refer to the corresponding cell (or question) in the jupyter-notebook. In the notebook, we have printed 3 images of the CAT object along with the mask.

(i) Follow the same template and paste 3 images for the BEAR object along with its mask in the box below.

(ii) In the box, paste the code to compute the mean and standard deviation of the magnitude of lighting vectors for all images of a given object. Briefly justify the obtained mean and standard deviation values.

(iii) Write the dimensions of $I_m$ and $L$, for the data loaded in the jupyter notebook. Note that $I_m$ will be a $m \times n$ matrix where $n$ is the number of pixels in the image.

## 2.2 Implementation (10 points)

You will now implement photometric stereo using the equations derived in problem 1.3.

(i) (See the jupyter notebook) Write the code to compute $\hat{G}$ as a function of $I_m$ and $L$ using the expression obtained in (iii) of problem 1.3.

(ii) (See the jupyter notebook) Write the code to compute a vector $\hat{\rho}$ of size $n \times 1$ which stores the albedo at each pixel.

(iii) (See the jupyter notebook) Having computed $\hat{G}$ and $\hat{\rho}$, write the code to compute $\hat{N}$.

(iv) (See the jupyter notebook) In the box below, replace the images shown, and paste the image of the obtained reconstruction, surface normal, and an image showing the error between the two for the CAT object.

Normal Ground Truth
Reconstruction
Error Map

(v) Repeat part (iv) for the BALL, READING, POT2, COW objects.

## 2.3 Qualitative Observation (1 points)

(vi)You might observe that for some objects, such as CAT the error map denotes lower error compared to regions of some objects such as BALL. Explain this observation.

## 3 Machine Learning Basics (10 points)

### 3.1 Calculating gradients (2.0 points)

A major aspect of neural network training is identifying optimal values for all the network parameters (weights and biases). Computing gradients of the loss function w.r.t these parameters is an essential operation in this regard (gradient descent). For some parameter $w$ (a scalar weight at some layer of the network), and for a loss function $L$, the weight update is given by $w := w - \alpha \frac{\partial L}{\partial w}$, where $\alpha$ is the learning rate/step size.

Consider (a) $w$, a scalar, (b) $\mathbf{x}$, a vector of size $(m \times 1)$, (c) $\mathbf{y}$, a vector of size $(n \times 1)$ and (d) $\mathbf{A}$, a matrix of size $(m \times n)$. Find the following gradients, and express them in the simplest possible form (boldface lowercase letters represent vectors, boldface uppercase letters represent matrices, plain lowercase letters represent scalars):

- $z = \mathbf{x}^T\mathbf{x}$, find $\frac{dz}{d\mathbf{x}}$
- $z = Trace(\mathbf{A}^T\mathbf{A})$, find $\frac{dz}{d\mathbf{A}}$
- $z = \mathbf{x}^T\mathbf{A}\mathbf{y}$, find $\frac{\partial z}{\partial \mathbf{y}}$
- $\mathbf{z} = \mathbf{A}\mathbf{y}$, find $\frac{d\mathbf{z}}{d\mathbf{y}}$

You may use the following formulae for reference:

$$\frac{\partial z}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \\ \vdots \\ \frac{\partial z}{\partial x_m} \end{bmatrix}, \frac{\partial z}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial z}{\partial A_{11}} & \frac{\partial z}{\partial A_{12}} & \cdots & \frac{\partial z}{\partial A_{1n}} \\ \frac{\partial z}{\partial A_{21}} & \frac{\partial z}{\partial A_{22}} & \cdots & \frac{\partial z}{\partial A_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z}{\partial A_{m1}} & \frac{\partial z}{\partial A_{m2}} & \cdots & \frac{\partial z}{\partial A_{mn}} \end{bmatrix}, \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_m} & \frac{\partial y_2}{\partial x_m} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$$

### 3.2 Deriving Cross entropy Loss (6.0 points)

In this problem, we derive the cross entropy loss for binary classification tasks. Let $\hat{y}$ be the output of a classifier for a given input $x$. $y$ denotes the true label (0 or 1) for the input $x$. Since $y$ has only 2 possible values, we can assume it follow a Bernoulli distribution w.r.t the input $x$. We hence wish to come up with a loss function $L(y, \hat{y})$, which we would like to minimize so that the difference between $\hat{y}$ and $y$ reduces. A Bernoulli random variable (refresh your pre-test material) takes a value of 1 with a probability $k$, and 0 with a probability of $1 - k$.

(i) Write an expression for $p(y|x)$, which is the probability that the classifier produces an observation $\hat{y}$ for a given input. Your answer would be in terms of $y, \hat{y}$. Justify your answer briefly.

(ii) Using (i), write an expression for $\log p(y|x)$. $\log p(y|x)$ denotes the log-likelihood, which should be maximized.

(iii) How do we obtain $L(y, \hat{y})$ from $\log p(x|y)$? Note that $L(y, \hat{y})$ is to be minimized

### 3.3 Perfect Classifier (?) (2.0 points)

You train a classifier on a training set, achieving an impressive accuracy of 100 %. However to your disappointment, you obtain a test set accuracy of 20 %. You bring up this issue in one of the discussions, and the TA suggests some possible steps you could take. For each suggestion below, explain why (or why not) if these suggestions may help improve the testing accuracy.

1. Use more training data
2. Add L2 regularization to your model
3. Increase your model size, i.e. increase the number of parameters in your model
4. Create a validation set by partitioning your training data. Use the model with highest accuracy on the validation set, not the training set.

## 4 Implementing an image classifier using PyTorch (15.0 points)

In this problem you will implement a CNN based image classifier in pytorch. We will work with the CIFAR-10 dataset. Follow the instructions in jupyter-notebook to complete the missing parts. For this part, you will use the notebook named PSET4_Classification. For training the model on colab gpus, upload the notebook on google colab, and change the runtime type to GPU.

### 4.1 Loading Data (2.0 points)

(i) Explain the function of transforms.Normalize() function (See the Jupyter notebook Q1 cell). How will you modify the arguments of this function for gray scale images instead of RGB images. (ii) Write the code snippets to print the number of training and test samples loaded.

Make sure that your answer is within the bounding box.

### 4.2 Classifier Architecture (6.0 points)

(See the Jupyter Notebook) Please go through the supplied code that defines the architecture (cell Q2 in the Jupyter Notebook), and answer the following questions.

1. Describe the entire architecture. The description for each layer should include details about kernel sizes, number of channels, activation functions and the type of the layer.
2. What does the padding parameter control?
3. Briefly explain the max pool layer.
4. What would happen if you change the kernel size to 3 for the CNN layers without changing anything else? Are you able to pass a test input through the network and get back an output of the same size? Why/why not? If not, what would you have to change to make it work?
5. While backpropagating through this network, for which layer you don't need to compute any additional gradients? Explain Briefly Why.

### 4.3 Training the network (3.0 points)

(i) (See the Jupyter notebook.) Complete the code in the jupyter notebook for training the network on a CPU, and paste the code in the notebook. Train your network for 3 epochs. Plot the running loss (in the notebook) w.r.t epochs.

(ii) (See the Jupyter notebook.) Modify your training code, to train the network on the GPU. Paste here the lines that need to be modified to train the network on google colab GPUs. Train the network for 20 epochs

(iii) Explain why you need to reset the parameter gradients for each pass of the network

### 4.4 Testing the network (4.0 points)

(i) (See the jupyter-notebook) Complete the code in the jupyter-notebook to test the accuracy of the network on the entire test set.

(ii) Train the network on the GPU with the following configurations, and report the testing accuracies and running loss curves -

- Training Batch Size 4, 20 training epochs

- Training Batch Size 4, 5 epochs
- Training Batch Size 16, 5 epochs
- Training Batch Size 16, 20 epochs

(iii) Explain your observations in (ii)

## 5 Interview Questions (15 points)

### 5.1 Batch Normalization (4 points)

Explain
(i) Why batch normalization acts as a regularizer.
(ii) Difference in using batch normalization at training vs inference (testing) time.

### 5.2 CNN filter sizes (4 points)

Assume a convolution layer in a CNN with parameters $C_{in} = 32$, $C_{out} = 64, k = 3$. If the input to this layer has the parameters $C = 32, H = 64, W = 64$.

(i) What will be the size of the output of this layer, if there is no padding, and stride = 1
(ii) What should be the padding and stride for the output size to be $C = 64, H = 32, W = 32$

## 5.3 L2 regularization and Weight Decay (4 points)

Assume a loss function of the form $L(y,\hat{y})$ where $y$ is the ground truth and $\hat{y} = f(x,w)$. $x$ denotes the input to a neural network (or any differentiable function) $f()$ with paramters/weights denoted by $w$. Adding $L2$ regularization to $L(y,\hat{y})$ we get a new loss function $L'(y,\hat{y}) = L(y,\hat{y}) + \lambda w^T w$, where $\lambda$ is a hyperparameter. Briefly explain why $L2$ regularization causes weight decay. Hint: Compare the gradient descent updates to $w$ for $L(y,\hat{y})$ and $L'(y,\hat{y})$. Your answer should fit in the given solution box.

## 5.4 Why CNNs? (3 points)

Give 2 reasons why using CNNs is better than using fully connected networks for image data.