EEE3314
INTRODUCTION TO AI

2025 Fall Semester

PROJECT

Due Date: 9 December (Tuesday) before midnight to LearnUS

INSTRUCTIONS:

1. This paper consists of 13 pages with 4 Questions only.

2. You may use any deep learning library or toolbox to solve the problems. However, in the introduction, you must include a table summarizing the relevant details (e.g., library name, version, etc.) for each question.

3. Submit **a report only for each group** containing the following items:

Example:
Introduction: student-A(100%) A brief introduction of the problem.
Problems and Solution: student-A(50%), student-B(50%)
Discussion: student-A(50%), student-B(50%)
Conclusion: student-B(100%)
Contributions: Distribute the workload equitably among all students. Submit a table itemizing each student's contributions, and inform us of any sleeping members.

Remark: The **Introduction** and **conclusion** are only written once in the report.

4. Use English in the report.

5. You may show a **code snippet** to aid your explanation; do not show the whole code in the report.

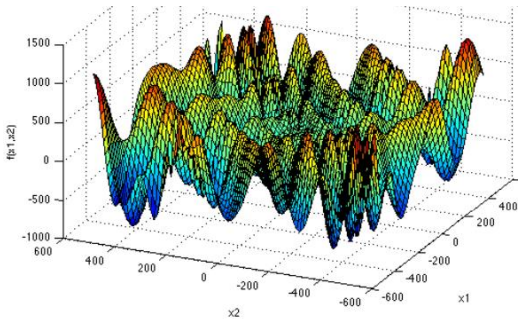
6. What to submit:
   (a) Report in PDF format.
   (b) Provide the source code for each question (a total of four files). If a question includes multiple source files, place them in a folder named "Qs1", "Qs2", etc. Upload all folders together as a single ZIP file. Your submission must run end-to-end; non-runnable code invalidates the report and will not be graded.

Question 1: Regression (Curve Fitting) problem using MLPs (30%)

Design an MLP to approximate the following function:

$$f(x_1, x_2) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1\sin(\sqrt{|x_1 - (x_2 + 47)|})$$

on the cross space [-512 512] x [-512 512]. The curve of $f(x_1, x_2)$ is shown in the following figure. The curve has radial/affine phases, cusps, and amplitude modulation.



1. First, generate 262,114 (512x512) 2-dimension data points based on $f(x_1, x_2)$ where $-512 \leq x_1 \leq 512$ and $-512 \leq x_2 \leq 512$.

2. Add standard Gaussian noises to $f(x_1, x_2)$, i.e., $\hat{f}(x_1, x_2) = f(x_1, x_2) + 0.1\text{N}(0,1)$ (1)
   The **input instance (x_1, x_2)** and the **target value** form a dataset, namely $\{[x_1, x_2], \hat{f}(x_1, x_2)\}$.

3. Standardize inputs and target using the z-score normalization method.

4. Randomly divide the data above into training, validation, and test sets in a ratio of 70%:15%:15%.

5. Train a **single hidden-layer architecture** (e.g., 2-10-1 to start with) with a **ReLU function** in the hidden layer using mini-batch SGD (MBGD).

6. Use the *nested CV protocol* with the **three-fold CV** to determine the **best number of hidden neurons** and estimate the **average generalization errors.** You may review Quiz 7, Qs 6 for the nested CV protocol.

   Specify the **optimizer** (variable learning rate MBGD, ADAM, etc.), **initialization method, and maximum epoch** you adopted, and why you chose them. List all hyperparameters you used in the table below.

	Method	**Hyperparameters**	**Reason**
Minibatch	-		
Optimizer			
Initialization		-	

Max epoch (>500)	-		-

(a) Fill the table below based on the **training set**/**validation set**. Report both $\mathbf{Err_{train}}$ and $\mathbf{Err_{val}}$. Determine the best #hidden neurons using the model selection protocol.

# hidden neurons, H	10	100	200	500
Fold 1	$\mathbf{Err_{train}}$ $\mathbf{Err_{val}}$			
Fold 2				
Fold 3				
Mean±std				
Best H =				

Err_* is the RMSE of the training and validation set

PS: You may increase the number of hidden neurons if you believe that more than 500 are necessary.

(b) Plot the target function (Eq. (1)), and the surface estimated by the optimal MLP determined from (a). Plot the MSE losses of training and validation versus epoch.
Does the model exhibit underfitting, overfitting, or good generalization? Why?

(c) Fill the table below for **generalization errors** based on the testing set.

	Optimal # hidden neurons from (a)	$\mathbf{Err_{test}}$
Fold 1*		
Fold 2		
Fold 3		
Mean±std	-	

* Note that this is a CV for generalization performance, not model selection in (a). For **each fold** in (c), you must determine the best #hidden neurons based on the **Mean of $\mathbf{Err_{val}}$**. In other words, for each fold in (c), three folds for (a) are required. Show all tables required for the nested CV.

(d) Discuss what you observe from the experiments. Discuss the possible issues if you cannot find a reasonably good approximation.

7. Repeat step 6 with your own **hand-designed augmented feature set**. Explain the rationale behind designing such features, grounded in the characteristics of the target function.

8. Repeat step 6 with **a three-hidden-layer network without augmented input features.**

9. Compare the fitting capability, generalization performance (test errors) and optimization difficulty of three models obtained from 6, 7 and 8.

Question 2: Image Dimension Reduction, Visualization, and Classification (25%)

In this question, you will visualize the distribution of the **CIFAR10 dataset** (https://www.cs.toronto.edu/~kriz/cifar.html) ten-class features in a 2D scatter plot using PCA and a CNN. Then perform classification.

The CIFAR-10 dataset includes 60,000 color images of size 32x32 pixels across 10 classes, with 6000 images in each class. It contains 50,000 training images and 10,000 test images.

The dataset is split into five training batches and one test batch, each with 10000 images. The test batch has exactly 1000 randomly selected images from each class. The training batches contain the remaining images in random order, although some batches may have more images from one class than another. Overall, the training batches each have exactly 5000 images per class.

Part A

Dimension Reduction with PCA.

To visualize the CIFAR-10 images in a 2D scatter plot, the images must be projected into a 2D feature space.

(a) Train a PCA projection matrix $\mathbf{W} \in \mathbb{R}^{nxd}$ by using the whole *training set* of CIFAR-10 images (50,000 samples).

Determine *n* and *d* for **W.**

(b) Project the test CIFAR-10 images (10,000 samples) onto a 2D feature space. Then, plot the distribution of the training and test features on a scatter plot (refer to Figure 1 for what a scatter plot looks like). You must plot both the training and test features in two separate figures. Use colors to distinguish the features of the ten classes.

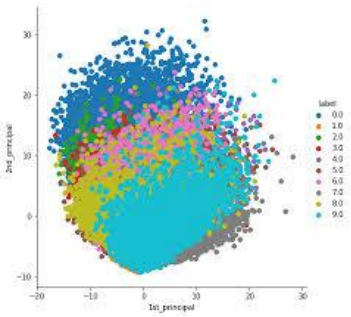


Figure 1: Scatter plot

(c) Train a **softmax regression*** using a dimension-reduced training set. The classifier will be trained via cross-entropy. Plot the training and testing loss curves.

(d) Calculate the accuracy of the training set and testing set using the classifier developed in (c).

Training Accuracy (%)	**Testing Accuracy (%)**

* A linear multi-class classifier constructed as an MLP without hidden layers. Use two input neurons for a 2-dimensional CIFAR-10 feature vector and ten output neurons with a softmax function. Implementation follows [1].

Part B

Dimension Reduction with CNN

Your CNN should follow the specifications as depicted in the table below.

	Stage 1		Stage 2		Stage 3		Stage 4	Stage 5
Layer	Conv	Pool	Conv	Pool	Conv	Pool	FC	Output
	$(5,32)_{/1,2}$ x2 RELU	$2_{/2,0}$	$(5,64)_{/1,2}$ x2 RELU	$2_{/2,0}$	$(5,128)_{/1,2}$ x2 RELU	$2_{/2,0}$	2, linear	10, softmax

where $(m, n)_{/a,b} \times c$ denotes c cascaded convolution layers with n filters of size $m \times m$, where the stride and padding are a and b, respectively. $p_{/r,s}$ denotes the max-pooling layers with a window of $p \times p$, where the stride and padding are r and s, respectively.

Train your network. The network will be trained using Cross-Entropy loss with mini-batch Gradient Descent. Describe the information related to the network training:

- Data Preprocessing (normalization)
- Initialization method, optimizer, regularization etc.
- Optimal hyperparameters (if any) include learning rate schedule, L2 coefficient, dropout rate, batch size, etc.
  Present them systematically in *table form*.

Plot training and testing loss curves.

(a) After training, input both the training images (50,000) and test images (10,000) into the network, then **collect the activation (features) values** (neuron output values) at **stage 4**, and plot the distribution in the feature scatter plot. You must plot both the training and test features in two separate figures.

(b) Train a **softmax regression model** using the CNN dimension-reduced training set. Use the same setting as Part A(c).

(c) Calculate the accuracy of the training set and testing set using the classifier developed in (b).

Training Accuracy (%)	**Testing Accuracy (%)**

(d) Compare the classification performances of two dimension reduction methods. Discuss this in terms of the learning paradigm (supervised, unsupervised, etc.) and the nature of the model (linear or non-linear).

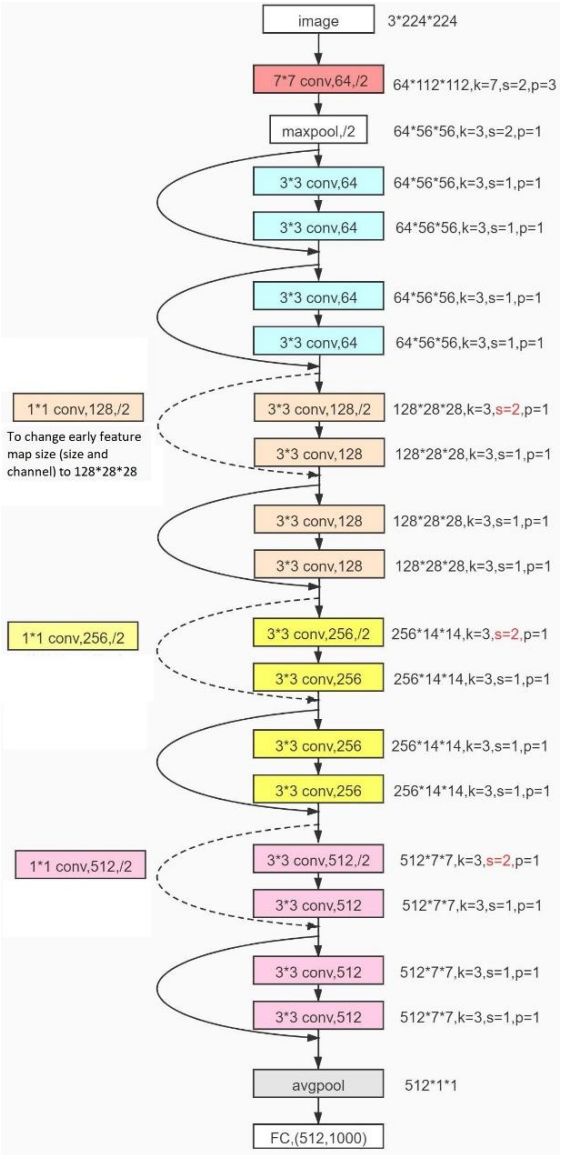
(e) Which model is best for **visualization** purposes, and why?

Question 3: Transfer Learning (30%)

In this question, we aim to transfer the knowledge from the **ImageNet pretrained ResNet18** to **low-resolution face recognition** problems. Here, we use a face dataset of 100 subjects, each with 50 samples, as the target dataset. The face dataset is included in the attachment.

The pretrained ResNet-18 can be found in [2]-[4] or other sources. The figure and table below illustrate the architecture and specifications of ResNet-18, respectively. You may refer to [5] for information on transfer learning and tuning.

The face image size is 64 x 64. With this input size, the feature map size becomes 2 x 2 x 512. Adjust the stride and/or padding to make the final feature map size 8 x 8 x 512. You should not change the network architecture, including the 17 convolution layers, 1 max pooling layer, or the filter size.



Layer name	output size	18-layers	No. of Blocks
conv1	112x112	7x7,64, stride 2	
conv2_x	56x56	3x3 max pool, stride 2	x 2
		3x3,64	
		3x3,64	
conv3_x	28x28	3x3,128	x 2
		3x3,128	
conv4_x	14x14	3x3,256	x 2
		3x3,256	
conv5_x	7x7	3x3,512	x 2
		3x3,512	
	1x1	average pool, 100-d fc, softmax	

(a) Fill out the specification table below for your modified ResNet18. The softmax layer is omitted in the table.

Layer	**Stride/Zero Padding**	**Feature Map Size**	**# of weights**	**Receptive Field**
Input	-	64x64x3	-	1
Conv1				
Pooling				
Conv2-1				
Conv2-2				
Conv2-3				
Conv2-4				
Conv3-1				
Conv3-2				
. . .				
avgpool (Global Average Pooling, GAP)				-

Why does the total weight stay the same as its original version (excluding the weights in the softmax layer)?

(b) First, **retrain** the softmax layer in the pretrained ResNet18 using face training samples (40 samples per subject). This involves freezing the other layers in the network and leaving only a linear classifier. The number of output nodes should be changed to 100 instead of 1000. This creates a **baseline model**. Next, evaluate testing accuracy using a test set of 10 samples per subject.

(c) **Modal A: Fine-tune Conv5_x.** Evaluate the model using the face test set. You may apply data augmentation, BNs, dropout, etc., to enhance test performance. Specify them in the report, if any.

(d) **Modal B: Fine tune Conv4_x and Conv5_x**. Evaluate the model with the face test set. You may apply data augmentation, BNs, dropout, etc., to improve the test performance. Specify them in the report, if any.

(e) **Modal C: Fine-tune all convolutional layers**. Evaluate the model using the face test set. You may use data augmentation, BNs, dropout, etc., to enhance test performance. Specify them in the report, if any.

(f) **Model D:** Freeze all the convolution blocks, add two fully connected (FC) layers before the softmax layer, and train them together with the softmax layer. This forms a 2-hidden-layer MLP. Specify the number of neurons in the FC layers. Evaluate the model using the face test set. You may use data augmentation, BNs, dropout, etc., to enhance test performance. Specify them in the report, if any.

For each model,

(i) Describe clearly the (initial) learning rate (a must) and other hyperparameters that you have set.
(ii) Show the graphs of both training and test loss vs. epoch.
(iii) Show the graphs of both classification accuracy on the training and test set vs. epoch.

Fill in the table

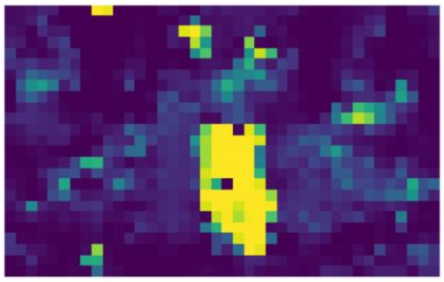
	Learning rate and hyperparameters (if any)	**Training Accuracy (%)**	**Testing Accuracy (%)**
Baseline Model			
Model A			
Model B			
Model C			
Model D			

(g) Analyze the performance of each model and determine if the results meet your expectations based on the **Size-Similarity matrix** discussed in the lecture.

Question 4: A simple, training-free object detector design (15%)

This question aims to design a simple, training-free object detector based on the ImageNet pretrained ResNet18. To do this, the network must be modified into a basic **Fully Convolutional ResNet18 (FCResNet18)**, capable of accepting **arbitrary-sized** input images and producing a response map at the output convolution layer. The response map highlights the labeled object in the input image (detected object). The two figures below illustrate the network's input and output instances.

Input image



Response map

1. To convert the ResNet18 to FCResNet18, the **avgpool layer** and **output layer** must be modified.

(a) Apply **average pooling*** instead of GAP; hence, a tensor with size *n*x*m*x512 is returned. *n* and *m* are the feature map size of Conv5_4 (Refer to question 3).

* Change the max operation of max pooling to the averaging operator

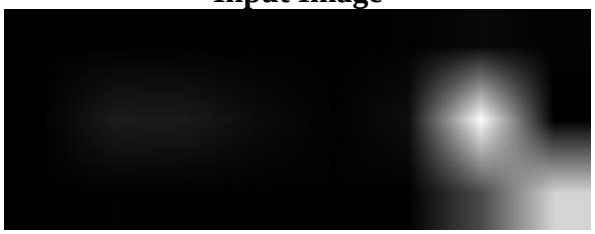
(b) The output layer has to be changed to a tensor with size *n*x*m*x1000, representing a **probability map** of size *n*x*m* for each of the 1000 classes.

2. Perform Inference to obtain the response map.

By inputting the given input image **X** (*camel.jpg*) with size 1920x725 after normalization (**X**-[0.485, 0.456, 0.406]/[0.229, 0.224, 0.225]) to the FCResNet18, you should be able to obtain a response map (score map) with size *n*x*m* (*n*<<1920, *m*<<*725*) where the maximum response occurs at class #354.

The #354 is the class label of "Arabian camel" in the ImageNet dataset. You can look up the names in imagenet_classes.txt, which is included in the attachment.

Input Image



Resize response map

The result can be interpreted as using a sliding window with size 224 x 224 to scan the image for nm times (224 x 224 is the input image size for the original ResNet18, refer to question 3).

The Pytorch code snippet could be helpful for step 2.

```
with torch.no_grad():
    # Perform inference.
    # Instead of a 1x1000 vector, we will get a
    # 1x1000xnxm output ( i.e. a probabibility map
    # of size n x m for each 1000 class,
    # where n and m depend on the size of the image.)
    preds = model(image)
    preds = torch.softmax(preds, dim=1)

    print('Response map shape : ', preds.shape)

    # Find the class with the maximum score in the n x m output map
    pred, class_idx = torch.max(preds, dim=1)
    print(class_idx)

    row_max, row_idx = torch.max(pred, dim=1)
    col_max, col_idx = torch.max(row_max, dim=1)
    predicted_class = class_idx[0, row_idx[0, col_idx], col_idx]

    # Print top predicted class
    print('Predicted Class : ', labels[predicted_class], predicted_class)
```

3. To overlay the response map on the input image, the maximum response map should be enlarged to fit the input image size, and then binarized with thresholding, i.e., choose a threshold and set those pixels to 1 or 0 with respect to the chosen threshold value.

The Pytorch code snippet could be helpful for step 3.

```
# Find the n x m score map for the predicted class
score_map = preds[0, predicted_class, :, :].cpu().numpy()
score_map = score_map[0]

# Resize score map to the original image size
score_map = cv2.resize(score_map, (original_image.shape[1], original_image.shape

# Binarize score map
_, score_map_for_contours = cv2.threshold(score_map, 0.25, 1, type=cv2.THRESH_BI
score_map_for_contours = score_map_for_contours.astype(np.uint8).copy()

# Find the countour of the binary blob
contours, _ = cv2.findContours(score_map_for_contours, mode=cv2.RETR_EXTERNAL, m

# Find bounding box around the object.
rect = cv2.boundingRect(contours[0])
```

To display the results graphically, use the response map to highlight the regions that show a high likelihood of the predicted class.

```
# Apply score map as a mask to original image
score_map = score_map - np.min(score_map[:])
score_map = score_map / np.max(score_map[:])
```

Next, multiply the response map with the original image and display the bounding box (optional).

```
score_map = cv2.cvtColor(score_map, cv2.COLOR_GRAY2BGR)
masked_image = (original_image * score_map).astype(np.uint8)

# Display bounding box
cv2.rectangle(masked_image, rect[:2], (rect[0] + rect[2], rect[1] + rect[3]), (0

# Display images
cv2.imshow("Original Image", original_image)
cv2.imshow("scaled_score_map", score_map)
cv2.imshow("activations_and_bbox", masked_image)
cv2.waitKey(0)
```

An example of the output should look like the figure below.

Detected Object

4. Based on the **last three digits** of the student ID, Z (pick one member ID), look up the class label from *imagenet_classes.txt*, and search for a relevant image of the same class from the internet. Perform object detection with your FCResNet18 and display the result as depicted in (3).

You do not need to resize the input image, as your network should be able to accept an arbitrary-sized input. Repeat with class label = Z + 5 e.g., X=100, X+5=105.

5. Capture **one face image** and **two random images** using your mobile phone, then input them into the network. Identify the response map that produces the highest activation and use it to perform face and random object detection. The photos may or may not include objects defined in the ImageNet dataset.

All the results should be arranged in the table below.

Student id	**Class label**	**Response map size**	**Display Input image (Image size)**	**Display the detected object image**
354 (Example)	Arabian Camel	*n* x *m*	Image (1920x725)	
Z				
Z + 5				
-			Face image (Size)	
-			Random image (Size)	
-			Random image (Size)	

6. Present your observations on the experimental results, identify the detector's limitations, and propose concrete improvements.

References:

[1] https://d2l.ai/chapter_linear-classification/softmax-regression-scratch.html

[2] https://pytorch.org/hub/pytorch_vision_resnet/ (Pytorch)

[3] https://kr.mathworks.com/help/deeplearning/ref/resnet18.html (MATLAB)

[4] https://github.com/qubvel/classification_models (Keras and TensorFlow Keras)

[5] https://d2l.ai/chapter_computer-vision/fine-tuning.html