ECEN 5672 – DIGITAL IMAGE PROCESSING - ASSIGNMENT 03 (Module 5 & 6)

This Homework is due in two weeks starting from today.
Due date: November 17th, 2025 11:59 PM (MDT)

Instructions:

- You are encouraged to work with other students to understand the course material. However, sharing source code, images, or other answers from any homework is strictly forbidden. You are to submit your own work.
- Please include your written answer and/or output images for each problem.
- The instructions are written with Matlab and Python in mind, but feel free to use any programming language or tool for this assignment.
- Please show all steps for full credit and attach the code to each problem.
- All the images needed for this homework can be downloaded as .jpg files from Canvas.
- The homework should be formatted as a **PDF file**. While submitting on Canvas, zip the PDF file with the images you generated, and submit as a single zipped file.
- You are allowed to use ChatGPT or any other LLM, but you have to specify where used it in the code and also explain it. Its alright having the syntax or the idea of some functions from these LLM models but **NOT THE WHOLE CODE!**

Problem 1: NON-LOCAL MEANS FILTER

In this problem, you'll work with the **Non-Local Means (NLM)** algorithm and see how it compares to regular neighborhood-based smoothing.

TOM.jpg

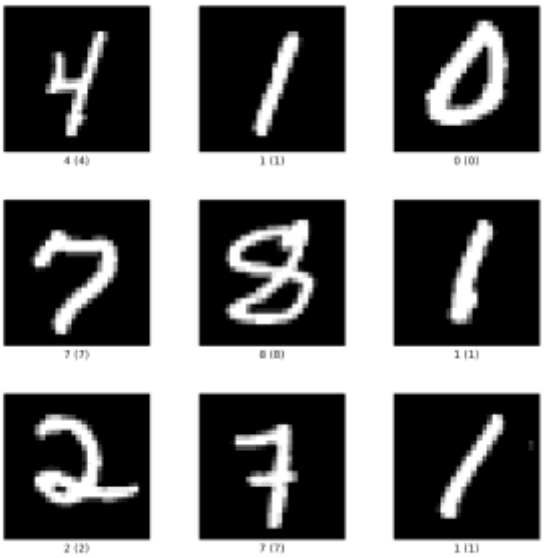
A. Start with a clean grayscale image of the given image **TOM.jpg**. Add **Gaussian noise** with a known standard deviation (for example, $\sigma = 20$). Show the original and noisy images side by side.

B. Apply simple **mean** and **Gaussian filters** of the same kernel size to the noisy image. Display both results and note what happens to fine details and edges.

C. Implement a **Non-Local Means filter** following the description from the lecture. Each pixel should be replaced by a weighted average of pixels in a larger search area, where the weights depend on patch similarity. Try patch sizes of 3×3 and 7×7, and search windows of 11×11 and 21×21. Discuss briefly how changing these sizes affects smoothness and runtime.

D. ***(For Grad Students)*** Extend your NLM to a **color image** such as peppers.png. You can run it either directly in RGB or after converting to YCbCr. Comment on which version looks cleaner and why.

Problem 2 - AUTOENCODERS AND DENOISING AUTOENCODERS



MNIST_example.jpg

A. Using the **MNIST dataset (the above image is an example)** design a **convolutional autoencoder** with an encoder–decoder structure. The encoder should progressively reduce the spatial dimension to reach a low-dimensional **bottleneck** representation, while the decoder reconstructs the image back to its original size. Train the network using a mean-squared-error loss until the reconstruction appears visually similar to the input.

"Use 10000 training images and 2000 test images from the dataset."

Link: https://www.tensorflow.org/datasets/catalog/mnist

B. Display and briefly describe the feature maps or latent representations learned at the bottleneck. Comment on whether they appear to capture meaningful shapes, textures, or other structural information.

C. Next, create a **denoising autoencoder** by adding Gaussian noise (for example, $\sigma = 0.3$) to the training images while keeping the original clean images as targets. Retrain the same network and compare its performance on noisy test images with that of the basic autoencoder. Visualize several examples showing the noisy input, the network's output, and the original image.

D. Discuss how adding noise during training changes the learned representation and why the denoising autoencoder is more robust to distortions. Relate your explanation to the regularization and sparsity concepts mentioned in the lecture slides.

Problem 3 – Image Compression (Image Entropy & Huffman Algorithm)

In this problem, you will implement Huffman coding for image compression and relate the achieved results to the entropy of the image.

Lena.png

A. Choose a grayscale image such as **lena.png.**
Compute its histogram and corresponding probability distribution of gray-level intensities (0–255).
Display both plots and briefly describe the overall distribution of pixel values.

B. Implement Huffman coding based on the intensity probabilities obtained in part A. Construct the Huffman tree, assign prefix-free codewords to each gray level, and encode the image into a binary bitstream.Record the total number of bits in the encoded image and compare it with the original 8-bit representation.

C. Decode the compressed bitstream using the same Huffman dictionary and confirm that the reconstructed image matches the original.
Display both images side by side to verify that the compression is lossless.

D. Compute the entropy $H = -\sum p(i)\log_2 p(i)$of the image and compare it with the average Huffman code length. Discuss in a few lines what this difference indicates about coding efficiency and how closely your result approaches the theoretical entropy limit.

Problem 4 – Block Truncation Coding (BTC)

In this problem, you will explore the idea of representing image blocks using only a few statistical parameters, as introduced in the **Block Truncation Coding (BTC)** technique.

Squirrel.jpg

A. Begin with a clean grayscale image of the given **Squirrel.jpg**. Convert it to double precision in the range [0,1] if required. Divide the image into non-overlapping **4×4 blocks** and compute the **mean** and **standard deviation** for each block.

B. For every block, generate a **binary pattern** where each pixel is assigned 1 if its intensity is greater than or equal to the block mean, and 0 otherwise. Display this binary pattern image for a few selected blocks to visualize how the thresholding separates lighter and darker regions.

C. Using the block mean (A) and standard deviation (σ), reconstruct each block with two quantized gray levels: **one level for all "1" pixels and another for all "0" pixels.**
Form the complete reconstructed image and display it alongside the original.

D. Repeat the reconstruction process using quantized values of the mean and standard deviation (for example, 6 bits for the mean and 4 bits for the standard deviation).
Observe how the reduction in bit depth affects both image fidelity and compression ratio.

E. ***For Grad Students***: Apply your BTC implementation to a **color image** such as peppers.png by compressing each channel (R, G, B) independently. Discuss whether color distortions or blocking artifacts become more noticeable and suggest a possible way to reduce them.

Problem 5 – Discrete Cosine Transform (DCT)

In this problem, you will study the **Discrete Cosine Transform (DCT)** and its role in image compression. The goal is to understand how the DCT represents image information efficiently in the frequency domain.

Earth.jpg

A. Convert the image **Earth.jpg** into its grayscale form and then solve. Partition the image into 8×8 blocks and compute the 2-D DCT of each block. For one sample block, display the DCT coefficients (or their log-magnitude) to show that most of the energy is concentrated in the low-frequency corner.

B. Set a selected number of the high-frequency DCT coefficients to zero in every 8×8 block (for example, keep only the lowest 10, then 20, then 30 coefficients following the usual top-left ordering). Reconstruct the image using the inverse DCT and display the results for the different retention levels. Comment on the visual changes (blurring, blocking, loss of fine detail) as more coefficients are removed.

C. For each retention level, estimate the amount of data saved by counting how many DCT coefficients are kept vs. the full 8×8 block. Express this as a simple compression measure (for example, “kept 20/64 coefficients per block → about 3.2× reduction”). Relate this observation to why JPEG first transforms, then quantizes, then entropy-codes the data.

D. Repeat part B using the **Discrete Fourier Transform (DFT)** on the same 8×8 blocks, keeping the same number of lowest-frequency terms. Compare the reconstructed images from DCT and DFT and explain, in a few lines, why the DCT version generally looks cleaner and has fewer block artifacts. (Hint: the DFT assumes periodic extension, while the DCT corresponds to an even/reflective extension, which reduces artificial discontinuities.)

Problem 6 – ALEXNET AND RESNET: EVOLUTION OF DEEP CONVOLUTIONAL NETWORKS

AlexNet and ResNet represent two key milestones in the evolution of deep convolutional neural networks (CNNs). Both are discussed in your lecture slides—AlexNet as the first successful deep CNN that revolutionized large-scale image recognition, and ResNet as the model that overcame degradation and vanishing-gradient issues in very deep networks through skip connections.

Question:
Explain how the architectural design choices of **AlexNet** and **ResNet** reflect the changing challenges in deep learning from 2012 to the modern era. In your answer, discuss the differences in network depth, feature extraction strategy, training stability, and generalization capability. You should also interpret why **skip connections** in ResNet enable much deeper models to converge effectively compared to AlexNet's straightforward layered design, and how these developments have influenced current CNN architectures.