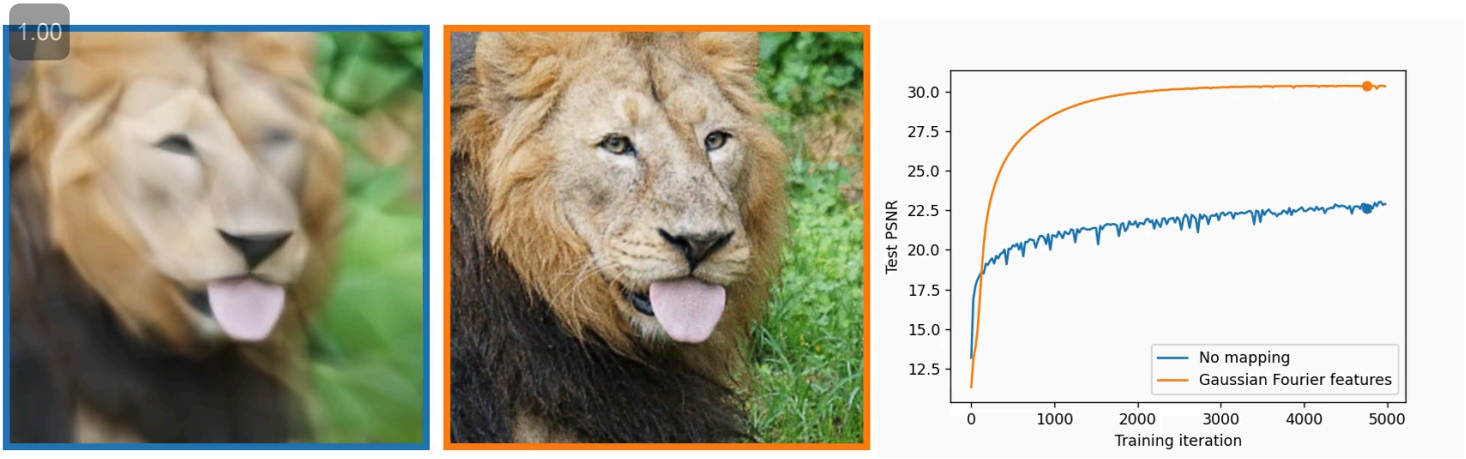
Spring 2024 CS 444

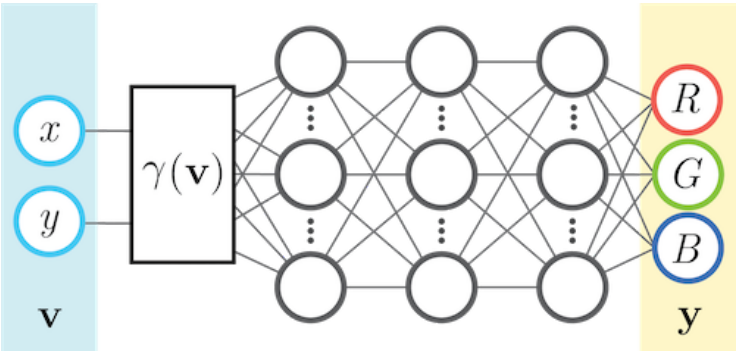
Assignment 2: Multi-Layer Neural Networks

Due date: Wednesday, March 6th, 11:59:59 PM



Source: https://bmild.github.io/fourfeat/

In this assignment you will implement multi-layer neural networks and backpropagation to "memorize" an image. As shown in the figure below, the inputs to the multi-layer network are the 2-dimensional (x,y) coordinates and the outputs of the network are the 3-dimensional RGB values. Therefore, this network will be used to reconstruct an image based on pixel coordinate inputs. Once you have trained a model using the (x, y) coordinates directly, you will then explore using different forms of input feature mapping as outlined in this paper to improve the final image reconstruction.



Source: https://bmild.github.io/fourfeat/

Specifically, you will write your own forward and backward pass and train a four-layer network with SGD and Adam optimizers. The network's weights will be updated to minimize a mean square error (MSE) loss between the original and reconstructed images. Other hyperparameters (e.g. hidden layer size, learning rate, activation function choices) are given to you in the neural_network.ipnyb notebook.

Assignment Setup

Download the starting code here.

The top-level notebook (`neural_network.ipynb`) will guide you through all the steps of training a neural network for this task. You will implement the multi-layer neural network in the `neural_net.py` file.

We also provide you with a notebook to help with debugging and testing your neural network implementation by using a toy dataset along with numeric gradient checks. It is found in `develop_neural_network.ipynb`. **NOTE: This file is only for debugging/experimenting/testing purposes. It will not be graded, and you are not required to submit it.**

None of the parts of this assignment require use of a machine with a GPU. You may complete the assignment using your local machine or you may use Google Colaboratory.

Google Colab Setup

Download the assignment zip file, decompress, and upload to Google Drive. In the assignment notebook, the first cell provides the command necessary to mount to Google Drive and access those files. You can edit the .py files by double clicking in the file explorer once it is mounted.

Local Setup

If you will be completing the assignment on a local machine then you will need a python environment set up with the appropriate packages. We suggest that you use Anaconda to manage python package dependencies (https://www.anaconda.com/download). This guide provides useful information on how to use Conda: https://conda.io/docs/user-guide/getting-started.html. Ensure that IPython is installed (https://ipython.org/install.html). You may then navigate the assignment directory in terminal and start a local IPython server using the `jupyter notebook` command.

Implementation Instructions

Toy Dataset Gradient Check

First you will implement the neural network components with forward pass and backward pass in neural_net.py. We recommend you use develop_neural_network.ipynb to perform a gradient check to confirm your backpropagation implementation is correct. This notebook will not be submitted for grading, but will be very helpful as an intermediate step before proceeding to the main assignment.

Low Resolution Image Reconstruction

Then you will use your neural network implementation to memorize a low resolution image. First follow neural_network.ipynb to implement the helper functions, including the input feature mapping and experiment runner, which will give the required functions to run the memorization experiments. Training on the low resolution image is relatively fast compared to training on a higher resolution image, so it will be easier to debug your implementation on this task. In particular, you will try the SGD and Adam optimizers on the following input feature mapping strategies: none, basic, and fourier feature mapping. For each training run requested above, please copy the training curves and image reconstruction results into the report template. Please discuss which optimizer worked best and why. Please discuss which feature mapping strategy worked best and why.

High Resolution Image Reconstruction

Now that you have made sure that your implementation works on the low resolution image, you will use your network to memorize a higher resolution image. You should first start with the example image and run the various feature mapping experiments to compare with the low resolution setting. Then, you will use an image of your own choice and train the network using the best feature mapping strategy. In particular, you will try an optimizer of your choice on the following input feature mapping strategies: none, basic, and fourier feature mapping. For each training run requested above, please copy the training curves and image reconstruction results into the report template. Please discuss which feature mapping strategy worked best and why. When choosing an image select one that you think will give you interesting results or a better insight into the performance and explain why in your report template.

Extra Credit

You are welcome to implement any extensions or advanced applications that come to your mind. You should provide results and discussion in your report, clearly marked as "Extra Credit," and upload any relevant code as well. We will consider any such attempts and give points provided there is sufficient evidence of additional learning and effort. Here are some suggestions for this assignment:

- Implement a deeper network -- try to go as deep as you can -- and see how the results, computational requirements, training behavior, etc. change.
- Experiment with the hyperparameters of the Gaussian Fourier Feature mapping (m, sigma). Discuss how the choice of these values affects the results.
- Experiment with alternative losses (e.g., L1 loss), regularization, or normalization techniques covered in class and see what difference they make.
- Implement the same network in PyTorch with autograd and compare the behavior (training curves, computational requirements, output quality) of your numpy and PyTorch networks.
- Apply your model to interesting image inpainting or restoration scenarios. You can also attempt to do hint-based colorization (i.e., colorization based on user-provided scribbles or reference color values at a sparse set of locations).
- Attempt a more advanced application like learning of 3D occupancy networks (see Section E.2 paper). This is probably more of a project idea, though.

Submission Instructions:

The assignment deliverables are as follows. If you are working in a pair, only one designated student should make the submission.

1. Upload four files to **Canvas**:
    1. **neural_net.py**: Your main multi-layer NN **python** implementation for ease of inspection
    2. **neural_network.ipynb**: Your NN training **IPython notebook** implementation for ease of inspection
    3. **netid_mp2_code.zip**: All of your code (python files and ipynb file) **in a single ZIP file**
    4. **netid_mp2_output.pdf**: Your IPython notebook with output cells converted to **PDF format**
    5. **netid_mp2_report.pdf**: Your assignment report (using this template) **in PDF format**

    Don't forget to hit "Submit" after uploading your files, otherwise we will not receive your submission!

Please refer to course policies on collaborations, late submission, etc.