Object oriented design and implementation: image density estimation and classification

LOCATIONs: Lectures NL3.114 large conference room
Office hours: cubicles outside of J9.130B

Instructors: Albert Montillo (AM) Albert.Montillo@UTSouthwestern.edu
TAs:

- Aixa Andrade (AA) aixaxiuhyolotzin.andradehernandez@utsouthwestern.edu
- Son Nguyen (SG) Son.Nguyen@UTSouthwestern.edu
- Austin Marckx (ATM) Austin.Marckx@UTsouthwestern.edu

Overall goals: This block will illustrate the fundamentals of *object oriented code design and implementation.* Throughout students will learn software debugging techniques and the use of the GPU hardware at the BioHPC. As a testbed for learning these concepts, students will generate their own VAE-GAN and apply it to synthesize and classify image data. This model combines the strengths of a variational autoencoder (VAE) with those of a generative adversarial network (GAN). Students are to (1) demonstrate that they can create a VAE-GAN whose encoder can learn compact representation, z, and visualize the learned class distribution across the compressed space. (2) Construct a decoder / generator that produces new images, x, that are very similar to the actual images, $\tilde{x}$, and describe what aspects of the images the model represents and suppresses. (3) Generate new images from each class. (4) Show that representation can also substantial performance for disease diagnosis.

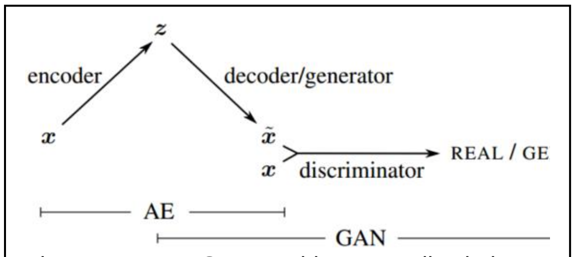

Figure 1: VAE-GAN architecture: discriminator from GAN receives input from VAE decoder

The traditional VAE allows for image compression but is plagued by blurry image quality. The VAE-GAN allows for *greater fidelity* in image compression and image generation because the primary loss function is not pixelwise, like the traditional VAE, but instead the loss aims to maximize fidelity to a semantic representation learned by the internal layers of the GAN's discriminator. The architecture of a VAE-GAN is illustrated in Fig 1. The VAE's decoder takes on the role of the GAN's generator. The entire model is trained end-to-end.

Students will also learn how to handle high dimensional data, in the form of images; how to optimize hyperparameters of sophisticated neural networks. They will also learn how to develop neural networks capable of exploiting the richness of information in such data with approaches such as, convolutional neural networks. Search strategies for hyperparameter optimization which are suitable for any neural network model will be illustrated. Along the way we will have opportunity to reinforce good programming practices, such as code documentation and version control, using the *git* version control system. Finally, students will learn how to use a symbolic debugger including advanced concepts such as the use of conditional break points.

Materials provided:

1. Test dataset: a dataset of 60,000 handwritten digits, MNIST, is provided, as testbed for rapid algorithm development and prototyping.
2. The shell of the subnetworks and integrator class which the students have to implement to increasing degrees
3. A suggested train/test split is provided.
4. Examples of fake and real images is provided

5. A separate CNN classifier of AD diagnosis is provided.
    a. Biomedical image datasets: MRI whole brain coronal slice (2D image) through the right hippocampus.
        i. grayscale intensity MRI images from an Alzheimer's Disease (AD) dataset
        ii. each image is labeled with subject diagnosis: AD or Cognitively Normal (CN).
6. Packet of reference material covering SWE (OO, Debugging, BioHPC GPU), and science (e.g. deep learning: VAE, GAN).
7. Access to a BioHPC compute node (each *with a GPU*) is provided to each student for the duration of this block. Students are expected to have and use their BioHPC account.

Grading:

You will receive a score based on 20 points (this block is 20% of your grade).
Note that below there are 24 points possible, and a maximum of 20 will be awarded. e.g. if your total score is 18 then you will get 18 points. If your total score is 22 you will get 20 points.

Weekend-Monday (5 pts) Reminder you have an extra day to complete since Monday is a holiday.

Weekend:

- 1 pt MyImageClass exercise
- 1 pt Using subclassing implement a DFNN MNIST classifier
- 1 pt Using subclassing implement a CNN CIFAR10 classifier

Monday (part of long weekend):

- 2 pts Complete a subclassed VAE implementation using starter code provided
- Monday's topic (VAE) uses a flipped class style. Students are to read the Monday_VAE slides on their own and at least start the VAE implementation exercise (slides 47-54).
  Any questions/concerns email TAs on Monday or bring your questions to Tuesday morning lecture.
  Tuesday morning: Complete your VAE implementation and \implement GANs.

Tuesday (5 pts)

- Morning and early afternoon
  - Complete your implementation of VAE that you started on Monday (see 2pts above).
  - 3 pts Complete a subclassed GAN implementation using starter code provided
- Late afternoon
  - 2 pts Debugging exercise

Wed (5 pts)

- 4 pts Subclass Monday's VAE implementation and create a cVAE
  - 2pts make_conditional_inputs
  - 2pts train_step
- 1 pt   Using code provided, run cGAN on the MNIST dataset. Compare to the GAN implemented Tuesday.
  - Write up your comparison as bullet points as comments at the start of the cGAN class
- Remember to check in your code!

Thurs (5 pts)

- 5 pts Using code provided, implement a hyperparameter tuning experiment (using at least two hyperparameters) on (choose one model) a cGAN, a cVAE, or a cVAE-cGAN.
- Describe the rationale for choosing your HPOs and how did each hyperparameter impacted performance.

Fri (4pts, present on the following)

- 2 pts: For a cGAN, a cVAE, or a cVAE-cGAN (choose one), explore meaning of latent space dimensions for at least three different MNIST digits

- 2 pts: Create a plot of the training curves (train and validation/test) for either a VAE or a GAN over at least 20 epochs. Loss vs Epochs

Schedule:

Friday May 23

5:00p-5:40p	Brief intro to weekend exercise and verify all students can clone block3 repo. *Brief Intro of Tensorfow OO deep learning framework* and its core base classes and methods such as Model and the Callback methods.	AA

Saturday/Sunday/Monday (holiday), May 24 + 25 + 26

Objectives: Learn 1) principles of object oriented code design and implementation, 2) how to manipulate biomedical images

Main task: Read/do the basic background exercises: Object Oriented (OO) implementation and Image input/output. *See next two rows below*. Second task: Read: SWE (OO, Debugging, BioHPC GPU), and science (VAE, GAN). Email AM questions by Sunday evening for Monday morning's discussion.	Self-study
Exercises: slides and Jupyter Notebook: *Principles of object oriented code design and implementation*. • Install Visual Studio Code in your BioHPC account. • Inheritance: is-a vs has-a. Polymorphism. Method overloading. Documenting a class and class library.	Self-study
Exercises: slides and Jupyter Notebook: *Density estimation and the manipulation of high dimensional biomedical images* • Load and display the provided images with Python. • Compute mean and variance across a set of images to characterize as a Gaussian distribution. • Qualitatively compare original and provided reconstructed images and explain the differences. • Quantify the reconstruction error between original and provided images.	Self-study

1. Read the reference materials listed below covering SWE (OO, Debugging, Python on the BioHPC), and Deep Learning density estimation (VAE, GAN). Self-study. Email TAs if questions.

a) Object oriented code development in Python: Tutorial link
b) Python on the BioHPC: tutorial slides
c) Image manipulation
    a. Numpy arrays, read through (up to and including) the section : "Indexing, Slicing and Iterating"
    https://numpy.org/doc/stable/user/quickstart.html
    b. Data visualization in python via matplotlib imshow:
    https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.imshow.html
    https://matplotlib.org/3.5.0/tutorials/introductory/images.html#sphx-glr-tutorials-introductory-images-py

d) Deep Learning (DL) and Density Estimation
1. If you have no Deep Learning knowledge:
    a. MLP: Skim through: Ch 6, Ian Goodfellow et al.: Deep Learning. Vol. 1. Cambridge: MIT press, 2016 (http://www.deeplearningbook.org/).
    b. CNN: Ch 9, first 10 pages, Ian Goodfellow et al.: Deep Learning. Vol. 1. Cambridge: MIT press, 2016 (http://www.deeplearningbook.org/).
    c. Autoencoders: Ch 14, Ian Goodfellow et al.: Deep Learning. Vol. 1. Cambridge: MIT press, 2016 (http://www.deeplearningbook.org/).
2. Density Estimation:
    a. VAE Understanding VAEs tutorial
    b. GAN, Understanding GANs tutorial
    c. Combining VAE+GAN: Boesen et al., J. Machine Learning Research, Autoencoding beyond pixels using a learned similarity metric, 2016 pdf link

e) The following are reading material for VSCode IDE and its symbolic debugger
#1 and #2 apply for any language, not only for Python
#3 is specific debugging for Python.

1. Start with VSCode https://code.visualstudio.com/docs/getstarted/userinterface

Visual Studio Code User Interface
User Interface. At its heart, Visual Studio Code is a code editor. Like many other code editors, VS Code adopts a common user interface and layout of an explorer on the left, showing all of the files and folders you have access to, and an editor on the

right, showing the content of the files you have opened.
code.visualstudio.com

2. Debug in VSCode https://code.visualstudio.com/Docs/editor/debugging

Debugging in Visual Studio Code
One of the great things in Visual Studio Code is debugging support. Set breakpoints, step-in, inspect variables and more.
code.visualstudio.com

3. A short (6 minutes) videos on Python debugging in VSCode

   https://www.youtube.com/watch?v=w8QHoVam1-I&ab_channel=Jayanam

```
class User:
    def __init__(self, name, email):
        self._name = name
        self._email = email

    def get_name(self):
        return self._name

    def get_email(self):
        return self._email

    def do_something(self):
        raise ValueError()
```

```
Exception has occurred: ValueError
exception: no description
  File "C:\Projects\Python\python_demo.py", line 13, in do_something
    raise ValueError()
  File "C:\Projects\Python\python_demo.py", line 21, in <module>
    user.do_something()
```

```
    def __str__(self):
        return self._name + ", " + self._email
```

How to Debug Python with VSCode - YouTube
In this short tutorial I want to show you how to debug Python scripts with VSCode. I explain simple and conditional breakpoints, and how to log messages inst...
www.youtube.com

Schedule for the week (continued):

Tuesday, May 27

Objectives: 1) Learn about OO implementations, through an example of a NN that performs density estimation of images

2) PyTorch as an alternative OO framework to Tensorflow,
3) how to debug with a symbolic debugger.
4) unsupervised learning with generative adversarial networks

9:00 AM – 10:00	Answer questions about OO implementation and image input/output from weekend background exercises. *Review your questions on VAEs: Unsupervised learning with variational autoencoders (VAEs)*. • Why is unsupervised learning important? • What is an encoder? What is a decoder? • Why do we need variational AEs? Principles of their design including subnetworks and the loss functions. *Introduce unsupervised learning with generative adversarial networks (GANs)*. • What is a GAN? • What is a generator? What is a discriminator? • What is a GAN useful for? • Core design principles including generator and discriminator subnetworks and their loss functions.	AM
10:00 AM- 3:15 PM	Coding session Part 1: • Finish implementing (started on Monday) your VAE for MNIST by subclassing the base *Model* class • Implement *learning curves* to tell whether the model is learning. • Visualize generated images. Visualize the compressed space of digits. Compute the reconstruction error. Part 2: Writing your own GAN w/OO implementation • Implement a GAN for *MNIST* datasets by subclassing the base *Model* class for the generator/discriminator components. • Implement a *custom class method* for the training loop which alternates between gradient descent of the generator and discriminator. • Compare your VAEs and GANs for synthesizing new images. • Write down how you would combine VAEs and GANs to combine their strengths.	Self-study
3:15 PM – 4:00	*Part 1: Debugging with the Python symbolic debugger in Visual Studio Code*	Son

4:00-6:00 PM	Coding session Part 1: Debug w/symbolic debugger • Load the provided bug-laden VAE code. • Set conditional breakpoints • Identify and resolve bugs.	Self-study
5:00 PM – 6:00 PM	Office hours	AA, SN
Evening	Continue exercise	Self-study

Wednesday, May 28

Objectives: Learn 1) learn what are and rational for: conditional VAEs and classification GANs. 2) learn about combining unsupervised and supervised learning with cVAE-cGANs with OO implementation

9:00 AM	Lecture on *Conditional* VAEs and *classification* GANs • What is a conditional VAE? • What is a classification GAN? • What are these models used for? • How are they implemented and evaluated? • Why might you combine them into a cVAE+cGAN?	AM
10:00AM-4:00 PM	Coding session Part 1: • Implement a cVAE (conditional) for MNIST Part 2: • Run the provided cGAN for MNIST show the output. Describe what performance differences you observe between GAN, cGAN, VAE, cVAE.	Self-study
4:00 PM – 5:00 PM	Office hours	SN, AA
Evening	Continue exercise	Self-study

Thursday, May 29

Objectives: 1) Learn to explore and tune parameters of a fully object oriented class design for a cVAE-cGAN. 2) form your own test exploring the effects of hyperparameter combinations.

9:00 AM	Lecture on putting it all together: hyperparameter optimization and tuning. Experimenting with a cVAE-cGANs which provides end-to-end learning for density estimation, generation and classification	AA
10:00AM-4:00 PM	• Study a combined cVAE-cGAN in OO for MNIST Compare its performance to networks you ran on Monday, Tuesday, and Wednesdays • Note how model definition can impact ease of HPO • Apply a (provided) cVAE-cGAN on MNIST	Self-study

	• Explore the impact of at least two hyperparameters. Either use two separate 1D searches, or a combined 2D grid search. (Parameter values and tips will be given.) • The HP search may be applied to either a cVAE or cGAN or cVAE-cGAN	
4:00 PM – 5:00 PM	Office hours	SN, AA
Evening	Continue exercise	Self-study

Friday, May 30 - Delivery day NL3.114 Conf room **from 1:00p.m.-3:00p.m.**

9:00-1:00	• Explore parameter values to reach the week's goal of learning an image density distribution from which realistic new images can be generated. Complete your Thursday's tuning of the training to reach optimal classification performance. • For a cGAN, a cVAE, or a cVAE-cGAN (choose one), explore meaning of latent space dimensions for at least three different MNIST digits. Implement a visualization (plot of your choosing) to illustrate the structure of the latent space • Create a plot of the training curves (train and validation/test) for one model VAE, GAN or VAE-GAN, over at least 20 epochs. Loss vs Epochs • Prepare for short presentations and code demos of highlights of your results (10 min)	Self-study
1:00-3:00 PM	Students present and discuss their projects individually with the instructors **Note this is in NL3.114 conf room**	AM/AA/Students

Directions to TA desks for Office hours:
J building 9th floor The TAs sit in cubicles near J9.130b