Announcements

[30 Apr 2023]

Zoom link for 1 May 2023

Invite Link
https://cmi-ac-in.zoom.us/j/82395345386?pwd=QzJEeHM0ZUZLWEpEdVlJakRxYmRYQT09
https://cmi-ac-in.zoom.us/j/82395345386?pwd=QzJEeHM0ZUZLWEpEdVlJakRxYmRYQT09

[25 Apr 2023]

Zoom link for 26 Apr 2023

https://cmi-ac-in.zoom.us/j/81112805739?pwd=RWtSMkZHcjFiN2tuR2NyT0ZLS2pjZz09

[20 Apr 2023]

Presentation Slots

—-
Topic - Poetry Classification from Speech to Text using Transformers.
Date - 25/04/23.
Team - Raktim, Avik, Srijit, Sinjini.
—-
1. Group name: Sentiment Analysis
2. Group members:
Sucheta Jhunjhunwala
Shreyansh Rastogi
Rajas Vaidya
Ishan Singh
3. Proposed date of presentation: 11am, 1st of May,2023
—-
Group Members:
1. Naheli
2. Neha
3. Nevetha
We would like to present on 25th. But if there are no slots
available on that day, we would like to present on 26th.
—-
PFB project presentations details -

Group Members: Rohan Dharmadhikari, Soham Biswas, Soham Pyne, Krishna Gupta

Proposed Date: 25th April, 2023

—-

Group: Ayush Srivastava, Pragya Jaiswal, Rishika Tibrewal, Ved Prakash

Proposed date: 25th April 2023, 2-4 PM

—-

Topic: Dog breed image classification

Group: Shouvik Ghosh and Siddhant Chaudhary

Proposed date: 25th April

[18 Apr 2023]

Guidelines for final project presentations

Slots for dates of presentation

1. 25th Apr 2023 : 2pm-4pm
2. 26th Apr 2023 : 9 30am - 12 30pm
3. 1st May: 9 30am - 12 30pm
4. 2nd May 2023: 2pm-4pm

Please send email to kulraghav@gmail.com with following information

1. group name
2. group members
3. proposed date of presentation

Guidelines for presentation

1. presentation time: 10-15min
2. use some online presentation tool slides/google doc/google jam to prepare your presentation
3. introduce the contributions/primary responsibilities of each team member
4. introduce the problem statement and motivation
5. introduce techniques/approaches/solutions
6. reference to github

[13 Apr 2023]

Project Update: due 18 Apr 2023

Please give a short update on the projects in the next class

[11 Apr 2023]

Assignment 5: due date 27 April 2023: extended 30 Apr 2023

1. Transfer Learning for image data using CNN

- Download about 100 images of ducks and 100 images of chickens from the internet
- In a google colab notebook, fine-tune a pre-trained convolutional neural network to classify duck vs chicken and output the classification report

https://www.analyticsvidhya.com/blog/2021/07/step-by-step-guide-for-image-classification-on-custom-datasets/

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

2. Transfer Learning for text data using Transformer

- Download the sentiment analysis dataset from https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset

- Build a sentiment analysis classifier to classify the sentiment into positive, neutral, and negative by fine-tuning a pre-trained transformer model and print your classification report

https://alvinntnu.github.io/NTNU_ENC2045_LECTURES/temp/sentiment-analysis-using-bert-keras-movie-reviews.html

https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/

[30 Mar 2023]

Please come to lecture hall 3 if you are in CMI

[21 Mar 2023]

https://jalammar.github.io/illustrated-transformer/

https://jalammar.github.io/illustrated-bert/

https://jalammar.github.io/illustrated-stable-diffusion/

https://en.wikipedia.org/wiki/Large_language_model

https://towardsdatascience.com/stable-diffusion-using-hugging-face-501d8dbdd8

[16 Mar 2023]

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

https://debuggercafe.com/implementing-resnet18-in-pytorch-from-scratch/

https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/

https://www.kaggle.com/code/blurredmachine/vggnet-16-architecture-a-complete-guide

https://idiotdeveloper.com/unet-implementation-in-pytorch/

[9 Mar 2023]

Assignment 4: due date 23 March 2023: extended 30th March 2023

1. **containerization**

- create a docker container for the flask app created in Assignment 3
- create a Dockerfile which contains the instructions to build the container, which include
    - installing the dependencies
    - copying app.py and score.py
    - launching the app by running “python app.py” upon entry
- build the docker image using Dockerfile
- run the docker container with appropriate port bindings
- in test.py write test_docker(..) function which does the following
    - launches the docker container using commandline (e.g. os.sys(..), docker build and docker run)
    - sends a request to the localhold endpoint /score (e.g. using requests library) for a sample text
    - checks if the response is as expected
    - close the docker container

In coverage.txt, produce the coverage report for the tests in test.py

2. **continuous integration**

- write a pre-commit git hook that will run the test.py automatically every time you try to commit the code to your local ‘main’ branch
- copy and push this pre-commit git hook file to your git repo

References

https://docker-curriculum.com/
https://www.tutorialspoint.com/docker/docker_overview.htm

https://www.freecodecamp.org/news/how-to-dockerize-a-flask-app/

https://githooks.com/
https://www.giacomodebidda.com/posts/a-simple-git-hook-for-your-python-projects/

[7 Mar 2023]

Project Proposal: due date 14 March 2023

Please email (one email per group) to kulraghav@gmail.com
final project proposal

1. proposed group (max 4 people: each person takes some primary responsibility)
2. proposed topic (NLP, computer vision, audio, video, predictive analytics etc)
3. proposed title (e.g. face mask detection)
4. proposed data sources (e.g. from internet or self-created)
5. proposed technique/model (e.g. deep learning, time-series prediction etc)

Browsing references for deep learning architectures:
https://www.v7labs.com/blog/neural-network-architectures-guide

Also read chapters 6, 7, 8, 9 from the https://www.deeplearningbook.org/

[2 Mar 2023]

Browsing references for Nerual Network Basics:
https://en.wikipedia.org/wiki/Stochastic_gradient_descent
https://towardsdatascience.com/animations-of-gradient-descent-and-loss-landscapes-of-neural-networks-in-python-e757f3584057
https://gbhat.com/machine_learning/gradient_descent_anim.html

https://machinelearningknowledge.ai/animated-explanation-of-feed-forward-neural-network-architecture/
https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.68438&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY

=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

https://heartbeat.comet.ml/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/
https://pub.towardsai.net/how-to-choose-your-loss-function-where-i-disagree-with-cassie-kozyrkov-2038d19b5e0a

https://towardsdatascience.com/how-to-choose-the-right-activation-function-for-neural-networks-3941ff0e6f9c

[28 Feb 2023]

Start browsing the project ideas. You may refer to the following links:
https://thecleverprogrammer.com/2020/11/22/deep-learning-projects-with-python/

https://becominghuman.ai/130-machine-learning-projects-solved-and-explained-897638335f1a

https://machinelearningprojects.net/deep-learning-projects/

https://huggingface.co/tasks

Next week: 7th March
first project proposal

1. proposed group (max 4 people: each person takes some primary responsibility)
2. proposed topic (NLP, computer vision, audio, video, predictive analytics etc)
3. proposed title (e.g. face mask detection)
4. proposed data sources (e.g. from internet or self-created)
5. proposed technique/model (e.g. deep learning, time-series prediction etc)

[16 Feb 2023]

Assignment 3: due date 9 March 2023: extended 14 March 2023

1. **unit testing**
    - In score.py, write a function with the following signature that scores a trained model on a text:

      def score(text:str,
                model:sklearn.estimator,
                threshold:float) -> prediction:bool,
                                    propensity:float

- In test.py, write a unit test function test_score(...) to test the score function. You may reload and use the best model saved during experiments in train.ipynb (in joblib/pkl format) for testing the score function.
  You may consider the following points to construct your test cases:
  - does the function produce some output without crashing (smoke test)
  - are the input/output formats/types as expected (format test)
  - is prediction value 0 or 1
  - is propensity score between 0 and 1
  - if you put the threshold to 0 does the prediction always become 1
  - if you put the threshold to 1 does the prediction always become 0
  - on an obvious spam input text is the prediction 1
  - on an obvious non-spam input text is the prediction 0

2. **flask serving**
   - In app.py, create a flask endpoint /score that receives a text as a POST request and gives a response in the json format consisting of prediction and propensity
   - In test.py, write an **integration test** function test_flask(...) that does the following:
     - launches the flask app using command line (e.g. use os.system)
     - test the response from the localhost endpoint
     - closes the flask app using command line

In coverage.txt produce the coverage report output of the unit test and integration test using pytest

[14 Feb 2023]

Mid-term preparation:
Please go through the class lecture notes and first 3 chapters of ISLR.
https://www.statlearning.com/

Midterm datetime: 20th Feb 2023: 2pm-5pm

[2 Feb 2023]

Familiarize yourself with

- **unit testing** using pytest coverage report
  https://docs.pytest.org/en/7.2.x/
  https://pytest-cov.readthedocs.io/en/latest/reporting.html
  https://realpython.com/python-debugging-pdb/

- **continuous integration** using git hooks
  https://githooks.com/
  https://www.giacomodebidda.com/posts/a-simple-git-hook-for-your-python-projects/

[31 Jan 2023]

Assignment 2: due date ~~16 Feb 2023~~ : extended 27th Feb 2023

1. **data version control**
   in prepare.ipynb track the versions of data using dvc
   - load the raw data into raw_data.csv and save the split data into train.csv/validation.csv/test.csv
   - update train/validation/test split by choosing different random seed
   - checkout the first version (before update) using dvc and print the distribution of target variable (number of 0s and number of 1s) in train.csv, validation.csv, and test.csv
   - checkout the updated version using dvc and print the distribution of target variable in train.csv, validation.csv, test.csv
   - **bonus:** (decouple compute and storage) track the data versions using google drive as storage
2. **model version control and experiment tracking**
   in train.ipynb track the experiments and model versions using mlflow
   - build, track, and register 3 benchmark models using MLflow
   - checkout and print AUCPR for each of the three benchmark models

References
https://dvc.org/doc/start/data-management/data-versioning
https://towardsdatascience.com/how-to-manage-files-in-google-drive-with-python-d26471d91ecd
https://realpython.com/python-data-version-control/
https://mlflow.org/docs/latest/tracking.html
https://mlflow.org/docs/latest/tutorials-and-examples/tutorial.html
https://towardsdatascience.com/experiment-tracking-with-mlflow-in-10-minutes-f7c2128b8f2c

[24 Jan 2023]

Familiarize yourself with **distributed version control** systems.
You may refer to the following links:
https://about.gitlab.com/topics/version-control/what-is-git-workflow/
https://uidaholib.github.io/get-git/3workflow.html

https://mlflow.org/docs/latest/tutorials-and-examples/tutorial.html

https://dvc.org/doc/use-cases/versioning-data-and-models/tutorial
https://dvc.org/doc/user-guide/how-to/setup-google-drive-remote
https://www.codementor.io/@oussamalouati4/versioning-data-and-models-for-machine-learning-projects-with-dvc-1k3glsgqyf

[19 Jan 2023]

Assignment 1 : due date 31 Jan 2023

Build a **prototype** for *sms spam classification*

1. in prepare.ipynb write the functions to
    - load the data from a given file path
    - preprocess the data (if needed)
    - split the data into train/validation/test
    - store the splits at train.csv/validation.csv/test.csv
2. in train.ipynb write the functions to
    - fit a model on train data
    - score a model on given data
    - evaluate the model predictions
    - validate the model
        i. fit on train
        ii. score on train and validation
        iii. evaluate on train and validation
        iv. fine-tune using train and validation (if necessary)
    - score 3 benchmark models on test data and select the best one

Notes:

- You may download sms spam data from https://archive.ics.uci.edu/ml/datasets/sms+spam+collection and
- You may refer to https://radimrehurek.com/data_science_python/ for building a prototype.
- You may refer to first 3 chapters of https://www.statlearning.com/ for basic ML concepts.
- You may refer to the Solution Design example covered in the class as a guideline for experiment design.
- You may refer to the Code Design example covered in the class as a guideline for writing python functions.

[17 Jan 2023]

Please download data from:

sms spam data:

https://archive.ics.uci.edu/ml/datasets/sms+spam+collection

and play with the data.

You may want to try out https://radimrehurek.com/data_science_python/.

[12 Jan 2023]

1. Please create a github account and AML repository and share your github handle and path to the AML repository

   to email: kulraghav@gmail.com

   This repository will be used for pushing your assignments and projects for this course.

2. Prepare fraud_detection.txt containing Problem Framing, Why ML, and Solution Design and push it to the AML repository. We will have class participation discussion on this on 19th.

[5 Jan 2023]

Please send

Name:

Batch: BSc/MSc CS/MSc DS

Python: beginner/intermediate/advanced

ML: beginner/intermediate/advanced

to email: kulraghav@gmail.com

Course Overview

Applied Machine Learning

—----------

Learning Objectives

- problem space: understand the typical use cases of Machine Learning methods in practice
- solution space: understand the basic building blocks used for the end-to-end application of Machine Learning

Course Structure

- 3 parts: ML Basics, MLOps, Deep Learning
- running use cases: spam filter, fraud detection, diabetic retinopathy detection, ads recommendation, relevance ranking
- 5 assignments: 2 jupyter notebooks (ML Basics), 2 python codes (MLOps), 1 google colab (Deep Learning)
- 1 mid term + 1 final project and presentation

Grading Structure

- Assignments: 50%
- Midterm: 20%
- Final Project: 20%
- Class Participation: 10%

Text books:

- An Introduction to Statistical Learning (authors: Gareth M. James, Trevor Hastie, Daniela Witten, Robert Tibshirani)
- Deep Learning: (authods: Ian Goodfelow, Yoshua Bengio (F.R.S.) , and Aaron Courville)

References:

- Basic ML: https://www.statlearning.com/
- Deep Learning: https://www.deeplearningbook.org/
- MLOps: https://madewithml.com/

Applied ML Basics

- ML use case design (problem framing + why ML)
- ML solution design
- ML system design

Running use case: sms spam filter

ML Use Case Design

Problem Framing

	qualitative	quantitative	question
Current State	too much spam => bad user experience => less engagement => less revenue	20% spam => 10% less engagement => 10% loss in revenue	what is the current situation (pains/desires) that we want to address and why?
Objectives	• build a model that can filter spam messages as soon as when received by the user • decrease spam => improve user experience => improve user engagement => more revenue => improve topline • ~~automate spam review => less labor requirement => less cost=>improve bottomline~~	reduce at least half of the spam(from 20% to 10%) => 5% more engagement => 5% more revenue	what is that we want to do and why? (to improve topline/bottomline?)

<table>
<tr>
<td>Benefit/ Cost Tradeoff and Prioratization</td>
<td>

cost of errors:

FN => spam stays => bad user experience => low engagement => less revenue
FP => genuine message filtered as spam => very bad user experience => more complaints => moer churn => big loss of revenue

benefits of correct predictions:

TP => correctly filtered spam => better user experience => better engagement => more revenue
TN => correctly kept non-spam => maintained user experience as expected => no significant impact on revenue
</td>
<td>
cost-benefit matrix
<table>
<tr><td>c(TP)</td><td>c(FP)</td></tr>
<tr><td>c(FN)</td><td>c(TN)</td></tr>
</table>
1% TP => + 0.5% engagement => + 0.5% revenue

1% FP => 1% very bad experiences => 0.1% risk of churn => 10% less engagement over customer lifetime => 10% less revenue

1% FN => -0.5% engagement => -0.5% revenue

1% TN => no significant impact on revenue
</td>
<td>what are the cost of errors/benefits of correct predictions and why?</td>
</tr>
<tr>
<td>Constraints</td>
<td>can only afford a small FP percent => very small percent of very bad user experience => limited risk of churn => limited loss of revenue</td>
<td>at most 10% FP => 1% very bad experiences => 0.1% churn => 0.1% risk of revenue loss => acceptable risk for 5% potential upside in revenue</td>
<td>what are the acceptable risks/budgets and why?</td>
</tr>
<tr>
<td>Desired State</td>
<td>

benefit: significantly lesser spam => significantly better user experience => significantly better engagement => significantly better revenue
cost: very few false positives => limited risk of very bad user experience => limited risk of churn => limited risk to revenue

</td>
<td>

at least 50% decrease in spam (from 20% to 10%) => 5% better engagement => 5% more revenue
at most 10% false positives

</td>
<td>what is the desired outcome (benefits/costs) that we want to see and why?</td>
</tr>
</table>

		=> 1% very bad experiences => 0.1% churns => 0.1% risk to revenue	

Why ML

	qualitative	quantitative	question
best non-ML alternative hypothesis	classify based on a curated list of keywords => too many FP and FN => very bad user experience => lesser engagement => loss of revenue	50% FP 70% FN => not cleaning enough spam and causing more complaints for misplacing genuine sms as spam => 5% revenue loss risk	what are the non-ML alternatives and why are they problematic? (pains/missed gains)?
ML value proposition hypothesis	much fewer FP and FN => much better user experience => much better revenue	10% FP 50% FN => 50% decrease in spam (from 20% to 10%) at the expense of 1% bad engagements => 5% increase in revenue at the expense of 0.1% risk	what are the advantages (pain relievers/gain creators) of ML solution and why?
ML feasibility hypothesis	• data: labelled samples of historic sms text data is available • model: state of the art review suggests promising candidates are available	• data: around five thousand samples • model: state of the art claim solutions with 10% FP 20% FN	what data and model are good candidates and why?

ML Solution Design

	choices	metrics	experiment
data	(labelled) sms text data	• label imbalance	• randomized 70/15/15 train/validation/test split
model	pr(spam)	• AUCPR (precision recall curve)	• rule based heuristic • tf-idf + logistic regression • tf-idf + random forest • BERT + logistic regression train these benchmark models (from simpler to more complex) using train data. validate and tune using validation data. select the model with best AUCPR on test data
action	if pr(spam) > threshold: auto take down	• precision • recall • confusion matrix	• choose a threshold to maximize the recall (estimated reward) subject to precision > 90%
reward	• decrease in spam • cost of misclassification	• % decrease in spam • % increase in daily active users	• shadow test • A/B test

[>> add guiding questions: what is the choice you are making and why?]

ML System Design

	service	flow
data	/prepare	1. request arrives at the server endpoint 2. server program pulls the data from Data Store to RAM 3. server program processes the data 4. server program stores the processed data back in Data Store
model	/train	1. request arrives at the server endpoint 2. server program pulls the data from Data Store 3. server program trains the model in RAM 4. server program logs the results in the Data Store 5. server program registers the model in Model Store (optional)
action	/score	1. server program pulls the model from Model Store to RAM and keeps it ready for scoring 2. request arrives at the server endpoint 3. server program pulls the data from Data Store to RAM 4. server program obtains the predictions and propensity scores of the model on the data in RAM 5. server program stores the scoring results in Data Store 6. downstream program takes appropriate action (e.g. auto take down) based on the stored results
reward	/evaluate	1. request arrives at the server endpoint 2. server pulls the data from Data Store to RAM 3. server computes evaluation metrics on the data 4. server stores the results back to Data Store

Code Design

	train	score
data	• load_data(file_path) -> df • save_data(df, file_path)->file_path • prepare_train_validation_test_split(df) -> df_train, df_val, df_test	• prepare_scoring_data(df) -> df_scoring
model	• fit(model, df) -> model • score(model, df, threshold)-> df_scored • evaluate(df_scored, metrics) -> results • validate(model, df_train, df_test) -> results	• load_model(path)->model • save_model(model, path)->path
action		• score(df, model, threshold) -> df_scored
reward		• evaluate(df_scored, metrics) -> results

Core Functions

function name	description	input	output	file name
prepare_training_data				prepare.ipynb
prepare_scoring_data				
prepare_train_validation_test_split				
load_data				
save_data				
validate				train.ipynb
fit				
score				
evaluate				
load_model				

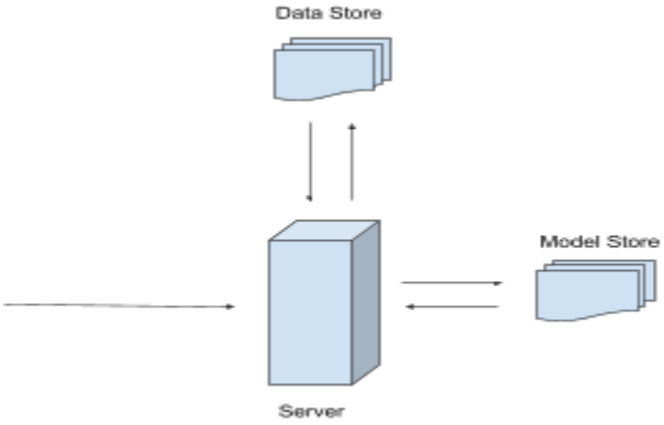
save_model				

Folder Structure

- README
- Dockerfile
- requirements.txt
- /src
    - /prepare.py
    - /train.py
    - /score.py
    - /evaluate.py
    - /helpers.py
    - /constructors.py
    - /app.py
- /config
    - /prepare.yaml
    - /train.yaml
    - /score.yaml
    - /evaluate.yaml
- /test
    - /units.py
    - /integration.py
- /<path_to_data_store>
    - /raw
    - /split
        - /train.csv
        - /validation.csv

        - /test.csv
- /<path_to_model_store>
    - /<model_name_version>.joblib
- .git/hooks
    - /pre-commit
    - /post-receive

[>> add component diagram + distributed system + scaling]



decouple compute and storage

MLOps

	Data	Model	Code
version control	dvc	mlflow	git
testing	• target distribution • key predictor distributions • out of range value checks	• overfit check • eyeballing sample positive/negative predictions	• unit tests • integration tests • automation using pytest • debug using pdb
modularity	• raw data • processed data • feature store • predictions	• separate propensity prediction and threshold	• prepare • train • score • evaluate
automation	• automatic data profiling upon change	• mlflow experiment tracking	• CI using git hook
deployment	• backup	• web-serving using flask	• CD using docker
monitoring	• distribution drift	• performance metrics • concept drift • bias	• test coverage
documentation	• wiki: data model	• wiki: experiment design • wiki:	• sphinx • system design diagram

		production models (params, training data, algo etc)	
reproducibility	dvc logs	log model params + data version + code version + docker file	docker
security	• database isolation • authentication	• file permissions • making only predictions available via API	• file permissions • using .pyc in production

Tasks

- Track the data versions
    - use dvc + git for versioning data from google drive
    - automate the process: dvc push after every update
- Track the model training process
    - use MLFlow to log params and metrics
    - automate the process: create a python decorator and apply it to train function
- Register a selected model
    - use MLFlow model registry
    - automate the process: registering the best model under an experiment
- Use a registered model to score
    - use MLFlow to load a registered model to score
    - automate the process of picking the latest version and falling back to previous version in case of failure
- Write test functions for the following functions:
    - prepare()
    - train()
    - score()
    - evaluate()
- Automate the test functions using pytest and githook
- Write a function to create a profile of data
- Write a function to validate the model training results
- Integrate sphinx with your documented python code to create a documentation
- Create wiki pages in gitlab to document your problem framing, experiment design, and code design
- Create a flask service with following API endpoints:
    - /score
    - /prepare

    - /train
    - /evaluate
- Create a docker container to host your service

Deep Learning

Neural Network Basics

loss functions	
SGD optimization	
multilayer perceprotn	
activation function	
backpropagation	

Deep Learning Architectures

feed forward networks	
CNN	
RNN	
LSTM	
Transformers	

Transfer Learning

Guiding Principles

	high data availability	low data availability
high domain similarity	finetune near-last layers	near-last layer as feature
low domain similarity	finetune deeper layers	deeper layer as feature

Use Cases

use case	image classification	text classification
model	ResNet	BERT
architecture	CNN + residual connections	transformer + self-attention
trained on		
num layers		
num params		

Building Blocks of Modern Architectures

Stable Diffusion

- Variational Auto Encoder (VAE)
- Diffusion Process
- Unet
    - convolution
    - transpose convolution
- Attention/Transformer
- Multi-modal embeddings (e.g. image + caption)

VAE to “reconstruct” an image
input_image-> latent_encoding -> sampling -> latent_decoding -> output_image

- reconstruction_loss(input_image, output_image)
- latent_loss(latent_encoding, Gaussian Normal)

ChatGPT

- Large Language Models (LLM)
    - next word prediction
    - next sentence prediction
    - fill in the blanks
- Attention/Transformer
- Reinforcement Learning with Human in the Loop

Students

[>> add Ayush and others]

e-mail	Name	Batch	Py	ML
shourjya@cmi.ac.in	Shourjya Basu	MSc CS ?	I?	I?
shramana@cmi.ac.in	Shramana Guin	Msc DS 2	I	I

amank@cmi.ac.in	Aman Kumar	MSc DS	A	A
krishnaguptacmi@gmail.com	Krishna Gupta	MScDS 2nd Year	I	I
deepmalya@cmi.ac.in	Deepmalya Dutta	MSc DS 1	I	I
dona@cmi.ac.in	Dona Ghosh	MSc Data Science ??	I	I
sohamp@cmi.ac.in	Soham Pyne	Msc Data Science ??	A	A
shreyansh.cmi@gmail.com	Shreyansh Rastogi	MSc DS ??	I	I
bharathr@cmi.ac.in	Ravilla Bharath Kumar	MSc DS ??	I	I
deeptikcmi@gmail.com	Deepti Kumawat	M.Sc. DS ??	I	I
shankar.ram1998@gmail.com	Shankar Ram V	MSc CS 2nd year	I	I
akhoury@cmi.ac.in	Akhoury Shauryam	BSc Maths 3rd Year	I	I
abhishekm@cmi.ac.in	Abhishek Mishra	M.Sc. DS ?? year	I	I
rajas@cmi.ac.in	Rajas Vaidya	MSc Ds II	I	I
ashutoshm@cmi.ac.in	Ashutosh Maurya	MSc DS ?? year	I	I
chaudhary@cmi.ac.in	Siddhant Chaudhary	MSc CS 1st Year	I	I
agnija@cmi.ac.in	Agnija Ashrita	MSc. Data Science II	I	I
ankushd@cmi.ac.in	Ankush Dey	M.Sc DS ?? year	I	I
ananya@cmi.ac.in	Ananya Sankaranarayanan	MSc DS ??	I	I
prashant@cmi.ac.in	Prashant Bajpai	MScDS 2nd Year	I	I
arghadeep@cmi.ac.in	Arghadeep Ghosh	MSc CS II	I	I
sampaddavcspur2001@gmail.com	Sampad Kumar Kar	MSc. CS ??	I	I
nehan@cmi.ac.in	Neha	MscDS 2	I	I
nevetha@cmi.ac.in	Nevetha N G	MSc DS 2nd Year	I	I

saikatb@cmi.ac.in	Saikat Bera	M.Sc. DS 1	I	B
aniket@cmi.ac.in	Aniket Santra	MSc DS II	I	I
bhaskar@cmi.ac.in	Bhaskar Pandey	MSc CS	I	B
shouvikg.cal@gmail.com	Shouvik Ghosh	BSc 3rd year Maths and CS	I	B
sucheta@cmi.ac.in	Sucheta Jhunjhunwala	MSc. DS II	I	I
avik@cmi.ac.in	Avik Das	MSc DS	I	I
roudranil@cmi.ac.in, dasroudranil@gmail.com	Roudranil Das	Msc DS (first year)	I	I
raktim022dey@gmail.com	Raktim Dey	MSc Data Science, 2nd year.	I	I
psrinivas@cmi.ac.in	Putta Ramlal Srinivas	MSc CS (II)	I	I
srijit@cmi.ac.in	Srijit Saha	MSc DS-2	I	I
deepmalya@cmi.ac.in	Deepmalya Dutta	MSc DS 1	I	I
ishan.singh.cmi@gmail.com	Ishan Singh	MSc. Data Science 2nd Year	I	I
biswassoham434@gmail.com	Soham Biswas	MSc DS 2nd Year	I	A
vedp@cmi.ac.in	Ved Prakash	Msc Data Science	I	I
rohand@cmi.ac.in	Rohan Dharmadhikari	MSc Data Science ??	A	A

Projects

https://medium.com/coders-camp/230-machine-learning-projects-with-python-5d0c7abf8265

https://thecleverprogrammer.com/2020/11/15/machine-learning-projects/

https://thecleverprogrammer.com/2020/11/22/deep-learning-projects-with-python/

https://becominghuman.ai/130-machine-learning-projects-solved-and-explained-897638335f1a

https://machinelearningprojects.net/deep-learning-projects/

https://huggingface.co/tasks

https://machinelearningknowledge.ai/animated-explanation-of-feed-forward-neural-network-architecture/