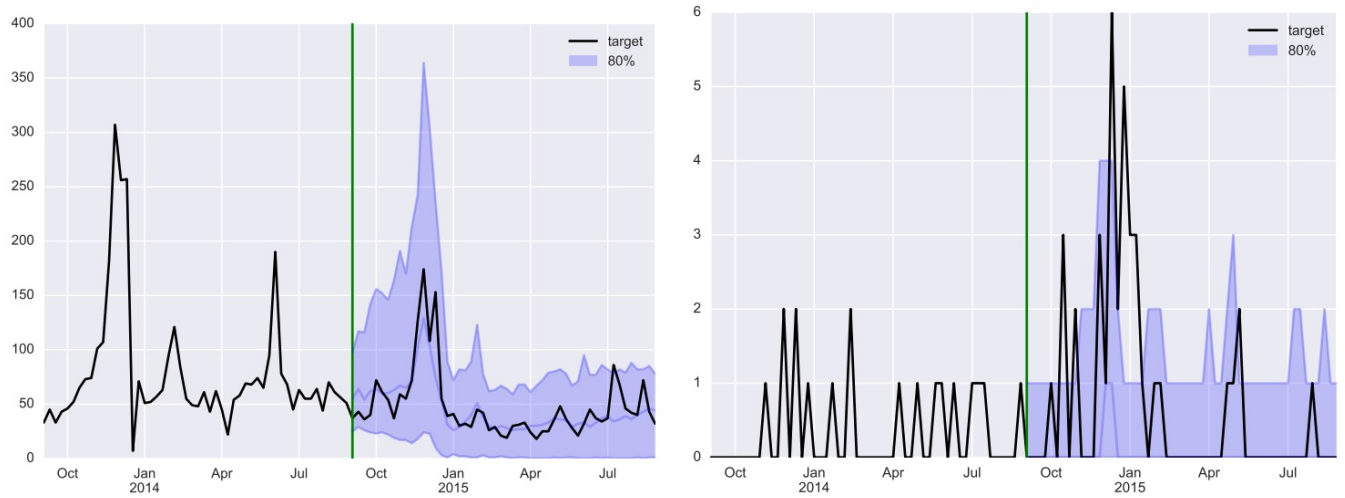
Project proposals by staff -DL 2022 team

Y1: Probabilistic Time Series Modeling (Cap 2)

- Introduction
  - In time series forecasting, people are usually interested in predicting a trend throughout time.
  - However, classic deep learning time series architecture gives deterministic result without quantifying the uncertainty.
  - The goal of this project is to extend the classic DL time series modeling into a probabilistic framework.



Reference:

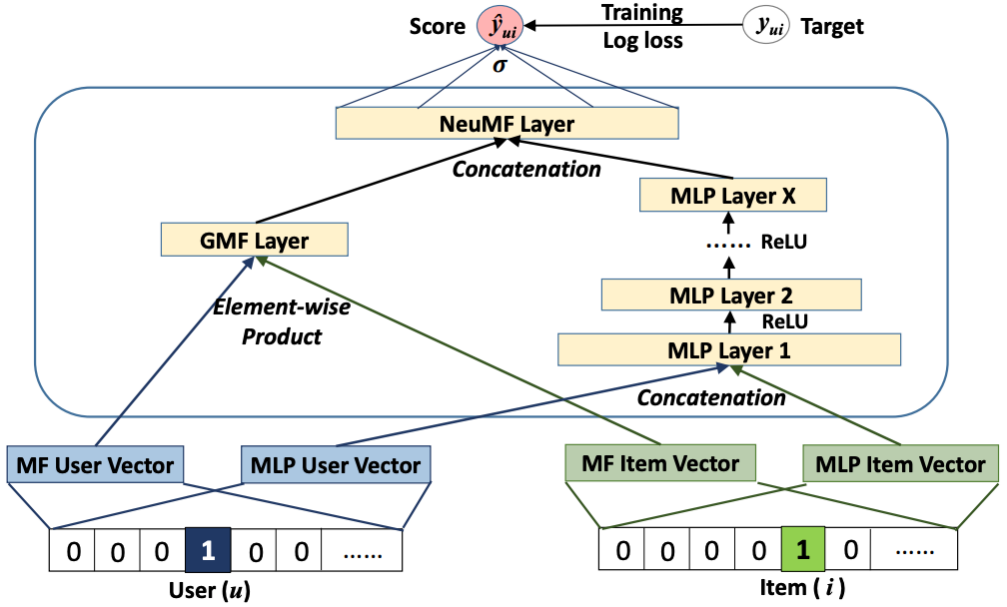
Y1: Probabilistic Time Series Modeling (Cap 2)

- Project Goal:
  - Learn and understand the current methods for time series probabilistic modeling through Deep Learning
- Project Outline:
  - Implement at least 2 of the framework from the list below:
    - https://proceedings.mlr.press/v89/gasthaus19a.html
    - https://proceedings.neurips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf
    - https://arxiv.org/pdf/1704.04110.pdf
    - https://arxiv.org/pdf/1711.11053.pdf
  - Benchmark the models using the S&P dataset, available here: https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks?select=sp500_stocks.csv
    - S&P 500 index is calculated through a weighted average of 500 stocks
    - Provide DL based probabilistic time series modeling/forecasting on the 500 stocks.
    - Forecast the S&P 500 Index (without additional information on the weights) and give an uncertainty quantification on the forecast.
- Responsible TA: Haoming Yang

Reference:

Y2: DL Recommender System (Cap 2)

- Introduction
  - Collaborative Filtering is a standard technique in Recommender systems.
  - Classically this technique was applied through matrix decomposition.
  - Recent developments in DL has empowered a new wave of recommender system based on embeddings.
  - However, simple embeddings are deterministic, meaning that for the same user, the recommendation will be the same.
  - How do we improve this?

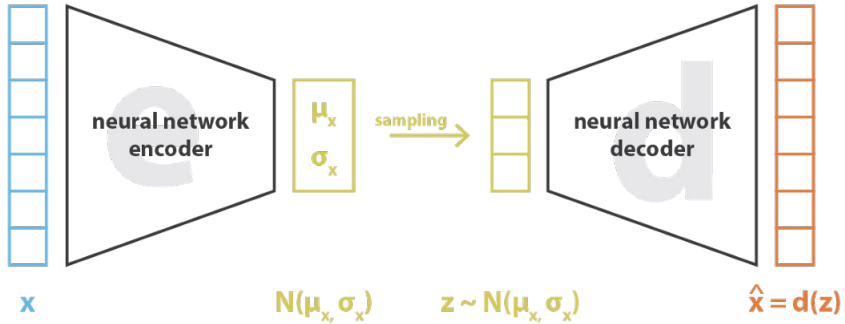


Y2: DL Recommender System (Cap 2)

- Project Goal:
  - Learn and understand the DL methods applied in recommendation problems.
- Project Outline:
  - Literature review on current DL framework for recommender systems:
    - https://arxiv.org/pdf/1708.05031.pdf%E2%80%8B (Neural Collaborative Filtering)
    - https://www.ijcai.org/Proceedings/2017/0447.pdf (Deep Matrix Factorization)
    - https://dl.acm.org/doi/pdf/10.1145/3178876.3186150 (Auto Encoder Collaborative Filtering)
  - Implement at least 1 Deep Learning based recommendation system
    - Modify your neural network such that it allows variation recommendation for the same user
  - Benchmark the model against classic Matrix factorization methods using dataset such as movielens or pinterest
    - http://yifanhu.net/PUB/cf.pdf (classic Matrix factorization)
    - https://grouplens.org/datasets/movielens/ (Movielens)
    - https://github.com/edervishaj/pinterest-recsys-dataset (pinterest)
  - https://github.com/robi56/Deep-Learning-for-Recommendation-Systems (Other DL Recommender)
  - Responsible TA: Haoming Yang

Y3: Repairing Images with Fisher Auto Encoder (Cap 2)

- Introduction
  - VAE is derived from its deterministic predecessor: Autoencoders. By a simple reparametrization tricks, we can generate realistic data through neural networks. There has been many different types of VAE developed since the vanilla VAE is developed.
  - We can also utilize the VAE to re-generate the damaged/noisy/hidden parts of an image.



$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Reference:

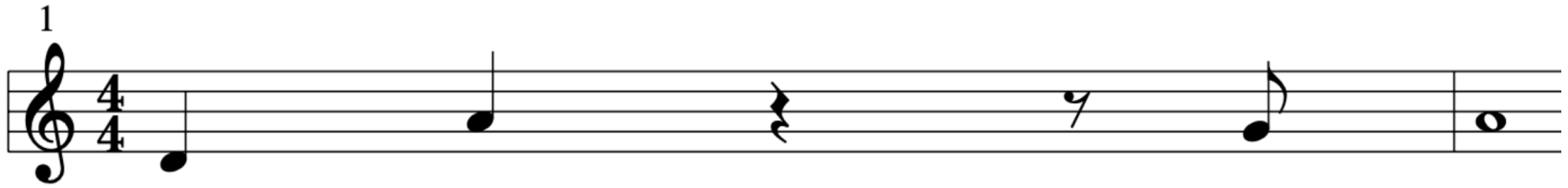
Y3: Repairing Images with Fisher Auto Encoder (Cap 2)

- Goal:
  - Your goal for this project is to learn different types of VAE and their ability to repair noisy/damaged images.
- Pipelines:
  - Literature Review on at least 2 VAE variants and compare them to the original VAE.
  - Implement the Fisher Auto-encoder (https://arxiv.org/pdf/2007.06120.pdf), one VAE from the list (see last bullet point) and original VAE
  - Use the KMNIST dataset (available on pyTorch) or animal-10 dataset (https://www.kaggle.com/datasets/alessiocorrado99/animals10)
    - Add random noise/ set random blocks of pixels to black (try to come up other novel methods to damage the dataset);
    - Find a way to robustly Repair these images through (Fisher) Variational Autoencoder.
  - Study and compare the performance in terms of repairing between the VAE variants and original VAE
    - Consider several levels of noise
    - Consider varying the size of black blocks on data
  - A list of different types of VAE: https://ai.stackexchange.com/questions/23670/how-many-types-of-variational-auto-encoders-are-there
- Responsible TA: Haoming Yang

Reference:

Y4: Deep Music Generation (Cap 2)

- Introduction
  - By treating notes of music as time series data, or a sequence of notes, one can apply deep learning tools such as RNN or VAE to generate music.
  - One can even generate music based on genre or composers to through conditional generation process to give DL generated music their distinct characteristics.



Reference:

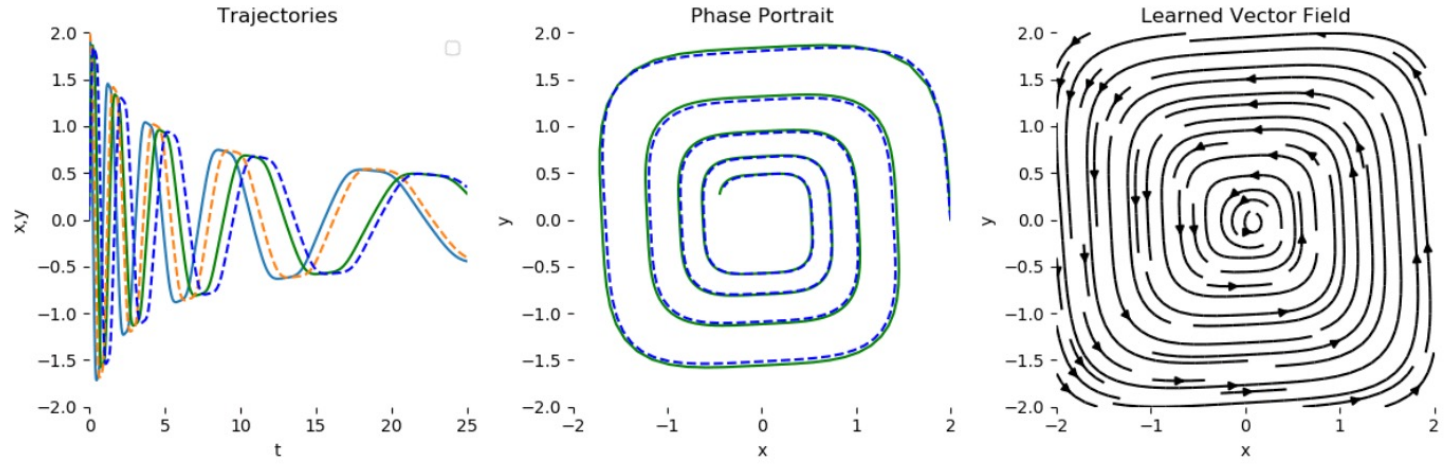
Y4: Deep Music Generation (Cap 2)

- Project Goal:
  - Learn Deep Music Generation technique, provide novel solution to generate music with a mixed style.
- Project Outline:
  - Literature Review on Deep Learning based music generation methods including papers from the following list:
    - https://arxiv.org/abs/1709.06298 (MuseGan)
    - https://arxiv.org/abs/1809.07600 (Midi-VAE)
    - https://arxiv.org/abs/1808.03715 (LSTM-Based)
    - More papers: https://github.com/Chinglohsiu/Deep-Learning-for-Music-Generation
  - After reviewing the papers, implement one of the methods and generate music on this dataset: https://www.kaggle.com/code/ohseokkim/music-generation-let-s-enjoy-new-music/data
    - Requirement:
      - Implement the methods into a music generation methods that allows conditional generation over composers
      - Further, the conditional generation should allow mixing of styles
        - Eg. Generate a short piano piece with 75% Bach style, 20% Beethoven style and 5% Mozart style.
      - Due to the size of the dataset, choosing 3 different composers are sufficient.
- Responsible TA: Haoming Yang

Reference:

X1: Neural ODE for classification (Cap 3)

- Introduction
  - Similar to a residual network, a neural ODE (or ODE-Net) takes a simple layer as a building block, and chains many copies of it together to build a bigger model.



Visualization of the Neural ODE learning the dynamical system

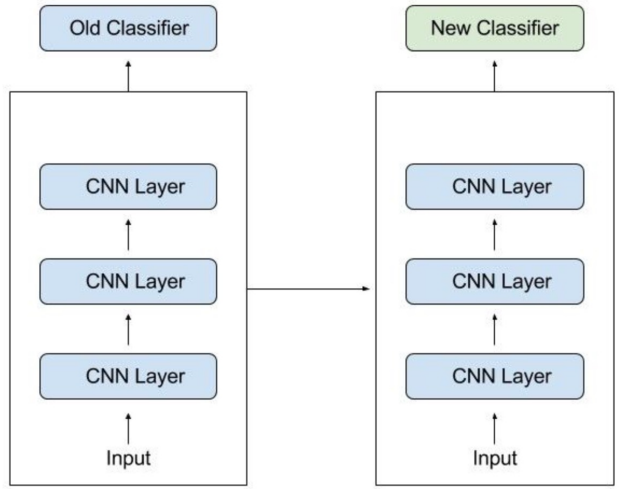
Reference:

X1: Neural ODE for classification (Cap 3)

- Goal:
  - Using neural ODE blocks on image classification.
- Project outline:
  - Train a ResNet 101 or other popular ResNet model on image classification tasks with CIFAR10 dataset. This is the baseline performance we want to compare with.
  - Train a single neural ODE model for classification tasks with CIFAR10 dataset. This is supposed to achieve a significantly worse performance compared to a ResNet.
  - Treat one neural ODE as a network block, we construct a classification model by connecting multiple neural ODE models with convolutional neural networks. We can also make use of neural SDEs, which is equivalent to applying batch norm to neural ODEs.
  - Evaluate the performance of neural ODE block model defined in the last block. and compare to the performance of ResNet in the first step. Neural ODEs usually perform worse than ResNets, due to constraints of ODEs. However, Neural ODEs are memory efficient in training. By using a combination of neural ODEs, we relax some of the constraints. Can this achieve similar performance to ResNets?
  - Reference: https://arxiv.org/abs/1806.07366 https://arxiv.org/abs/2205.14612
- Notes: this project requires students to understand basic ODEs.
- Responsible TA: Xingzi Xu

X2: transfer-learning Literature review (cap 2)

- Introduction
  - Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.
  - It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.



Reference: https://machinelearningmastery.com/transfer-learning-for-deep-learning/

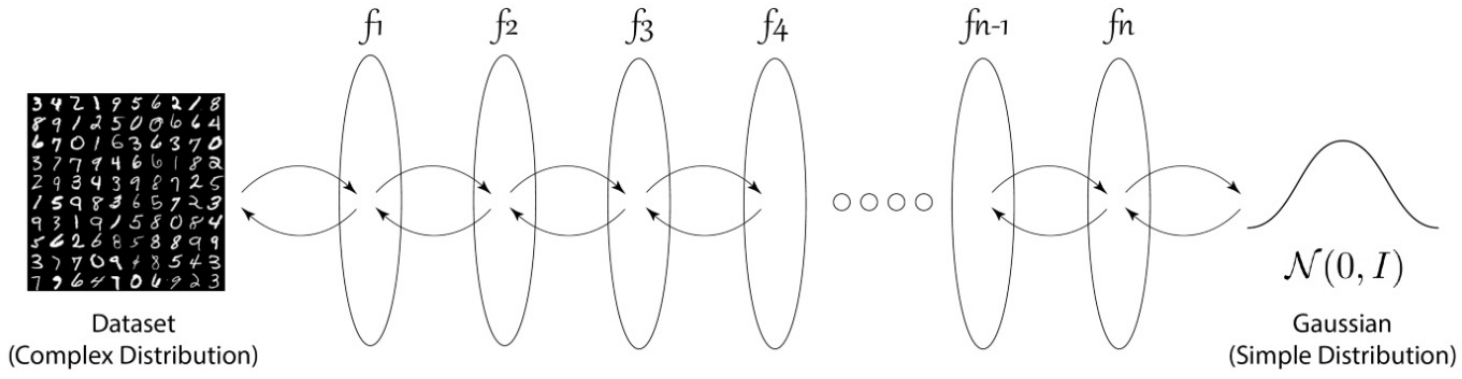
X2: transfer-learning Literature review (cap 2)

- Goal:
  - Review transfer-learning literature and implement one or two popular transfer-learning methods on a task that is not image classification.
- Project outline:
  - Review four different transfer-learning methods. Compare their performances and reason why one method performs better.
  - Implement L2T(http://proceedings.mlr.press/v80/wei18a/wei18a.pdf) or SpotTune (https://openaccess.thecvf.com/content_CVPR_2019/html/Guo_SpotTune_Transfer_Learning_Through_Adaptive_Fine-Tuning_CVPR_2019_paper.html) on a time series prediction task. Specifically, train a LSTM first to predict PhysioNet dataset (https://physionet.org/about/database/) values (use the whole dataset), then perform transfer learning to the LSTM model to predict UCI human activity dataset (https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones) with less than 50% of the whole dataset. Compare the performance of the transfer learning and training from a random initialization.
  - Perform ablation study on the transfer-learning method you tried. (You can look at the impact of number of classes, batch size, backbone network, and more)
- Responsible TA: Xingzi Xu

Reference:

X3: Normalizing flow Literature review (cap 2)

- Introduction
  - normalizing flows is a series of simple functions which are invertible, or the analytical inverse of the function can be calculated. For example, $f(x) = x + 2$ is a reversible function because for each input, a unique output exists and vice-versa whereas $f(x) = x^2$ is not a reversible function. Such functions are also known as bijective functions.



(Source: Author)

Reference: https://towardsdatascience.com/introduction-to-normalizing-flows-d002af262a4b

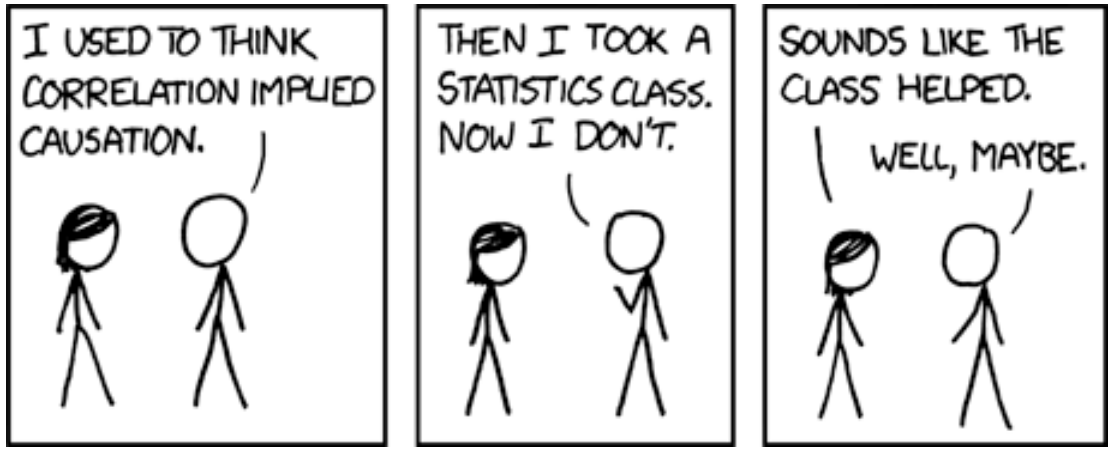
X3: Normalizing flow Literature review (cap 2)

- Goal:
  - Review normalizing flow literature and implement one or two popular normalizing flow methods on EMNIST dataset.
- Project outline:
  - Review four different normalizing flow methods. Including real NVP (https://arxiv.org/abs/1605.08803) and Glow (https://arxiv.org/abs/1807.03039). Compare their performances and reason why one method performs better.
  - Implement 1-2 normalizing flow methods to perform super-resolution tasks. Specifically, we learn the density transformation between MNIST images downsampled to 14x14 and zero padded back to the original 28x28 sizes and the original MNIST dataset.
  - Perform an ablation study on the normalizing flow method you tried. (You can look at the impact of backbone networks, the number of layers, and more)
  - Reference: https://github.com/janosh/awesome-normalizing-flows ;https://arxiv.org/pdf/2006.14200.pdf
- Responsible TA: Xingzi Xu

Reference:

A1: Causal Inference (Cap 2)

- Introduction
  - Causal inference refers to an intellectual discipline that considers the assumptions, study designs, and estimation strategies that allow researchers to draw causal conclusions based on data.



Reference:

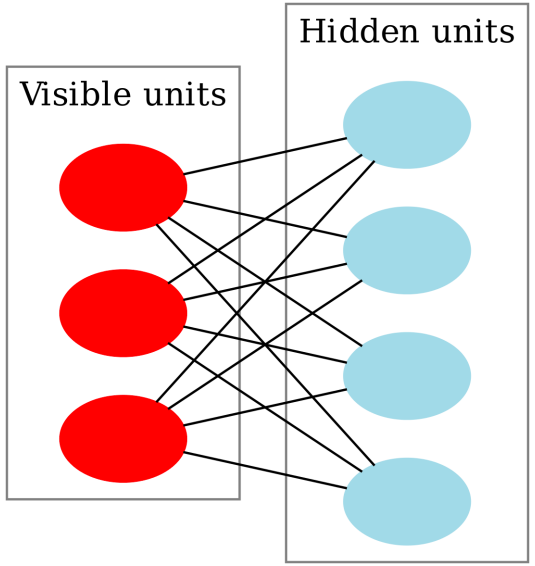
A1: Causal Inference (Cap 2)

- Goal
  - Literature review of causal inference.
  - Implement an S-Learner, a T-Learner, and a TARNet model and compare their performance on the IHDP dataset and two simulated synthetic datasets of your choice.
  - Change the Wasserstein Distance to an Energy Distance and compare this to the performance with Wassertein distance.
  - Use ATE and EPEHE as metrics.
  - References:
    - Basics of Causal Inference
      - https://towardsdatascience.com/implementing-causal-inference-a-key-step-towards-agi-de2cde8ea599.
    - T-Learners and S-Learners
      - https://statisticaloddsandends.wordpress.com/2022/05/20/t-learners-s-learners-and-x-learners/
    - TARNet Paper:
      - https://arxiv.org/pdf/1606.03976.pdf
  - Responsible TA: Ahmed Aloui

Reference:

A2: Training RBM with Fisher Divergence (Cap 2)

- Introduction: Restricted Boltzman Machines are a class of generative artificial neural networks that learn the probability distribution over its set of inputs.



Reference:

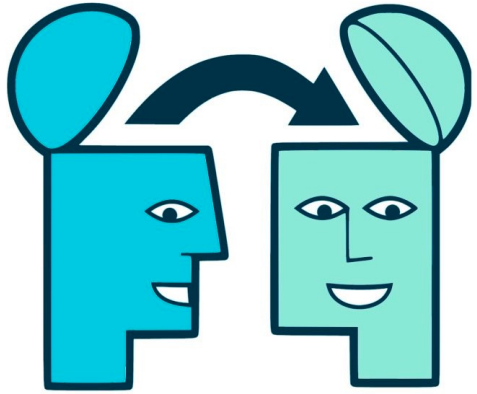
A2: Training RBM with Fisher Divergence (Cap 2)

- Goals
  - Implement an RBM and train it with contrastive divergence.
  - Train the RBM with the Fisher Divergence (also called score matching) instead
  - Compare these two methods.
  - Have an ablation study on the performance of both divergences as a function of your hyperparameters.
- References:
  - Fisher Divergence: check the section 4 in this paper: How to train you energy-based models: https://arxiv.org/pdf/2101.03288.pdf
  - RBM Tutorial: https://medium.com/machine-learning-researcher/boltzmann-machine-c2ce76d94da5
- Responsible TA: Ahmed Aloui

Reference:

A3: Task Affinity based time series knowledge transfer (Cap 4)

- Introduction: Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. A major challenge is defining a "distance" over the tasks space. The Fisher task distance proved to be a powerful tool to come up with this ordering.



Reference:

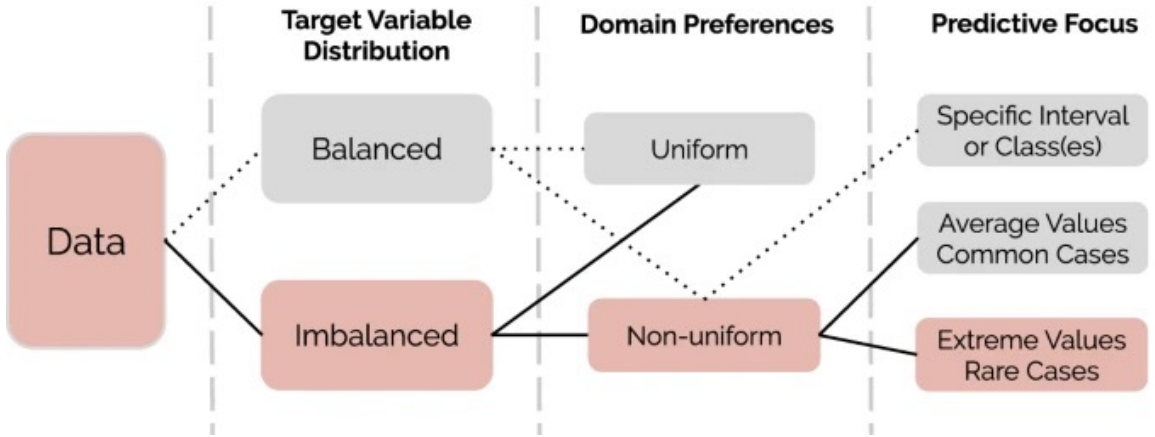
A3: Task Affinity based time series knowledge transfer (Cap 4)

- Pipeline
  - Find a time series dataset and create 10 datasets from it.
  - Use Fisher Task affinity to learn the distance between the source task and the new generated datasets.
  - See if the task affinity recovers the intuitive ordering of the datasets (according to the way you generated it).
  - Print the loss of the target (with/without tuning) as a function of the task distance.
  - Implement the previous steps for 3 different neural networks of your choice.
  - See if the task distance is stable when changing the neural net architecture.
- References:
  - Task Affinity:
    - https://arxiv.org/pdf/2110.02399.pdf
  - MAML (a Meta-Learning method) for time series:
    - https://arxiv.org/pdf/2108.02842.pdf
- Responsible TA: Ahmed Aloui

Reference:

S1: Extreme Value Prediction in Imbalanced Regression (Cap 2)

- Introduction
  - Three factors must be considered when analyzing imbalanced learning tasks: (i) the distribution of the target variable, (ii) domain preferences concerning accurate predictions, and (iii) their predictive focus. A schematic definition of such type of tasks is presented in figure here



Reference:

S1: Extreme Value Prediction in Imbalanced Regression (Cap 2)

- Goal
  - Predict age > 80 with high accuracy in the UTK Face dataset.
- Project outline:
  - Perform Exploratory Data Analysis on UTK Face dataset.
  - Develop a CNN-based regression model to predict age from Face.
  - Plot and analyze true and predicted age for Faces with age > 80.
  - Implement Label Distribution Smoothing (LDS) and Feature Distribution Smoothing (FDS) algorithms mentioned in Reference with your CNN network to predict the same.
  - Analyze how the new model performs on Face with age < 18 and the rest of the age. Ideally there should not be much of a difference in accuracy in the other images.
  - According to the paper, FDS should be applied the final layer (before predictions) of the neural network. Is there a way to apply a novel FDS to convolution layer since CNNs have more representative power? The goal is to explore this domain.
  - Dataset: https://www.kaggle.com/datasets/jangedoo/utkface-new
  - Reference: https://arxiv.org/pdf/2102.09554.pdf
- Responsible TA: Sayan Mandal

Reference:

S2: Weakly Supervised Image Classification (Cap 3)

- Introduction
  - Weakly supervised learning is an essential problem in computer vision tasks, such as image classification, object recognition, etc., because it is expected to work in the scenarios where a large dataset with clean labels is not available.

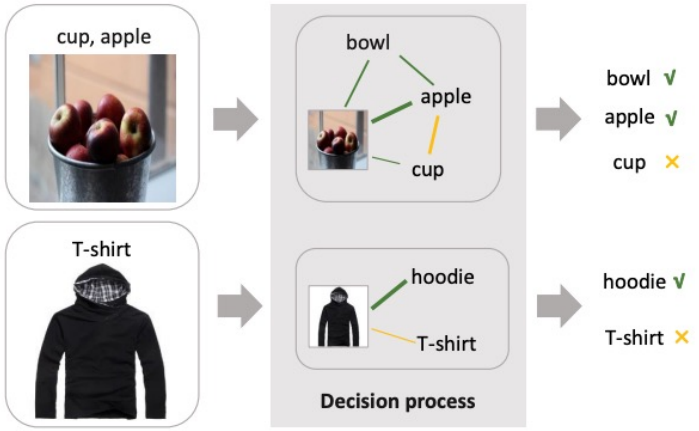


Figure 1. An illustration of distinguishing between the reliable and unreliable parts in the massive noisy labels based on assumptions, such as label-to-label relationship and image-to-label relationship. The green and yellow lines denote positive and negative correlation, respectively. The thickness of a line denotes the strength of the correlation.

Reference:

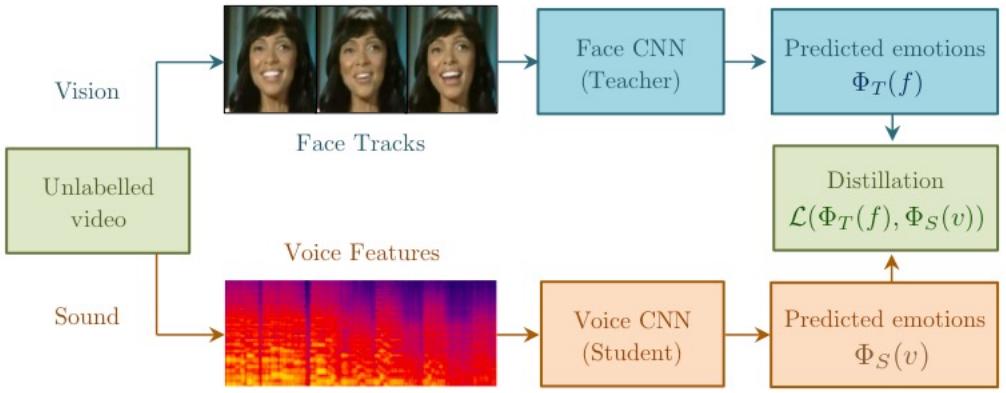
S2: Weakly Supervised Image Classification (Cap 3)

- Goal
  - Implementing a noise regularizing CNN model and Weakly supervised GAN that can classify images with noisy labels.
- Project outline:
  - For training data: Split the dataset into clean labels (<10%) and dirty labels (>90%). For the dirty labels, randomly shuffle the labels among images.
  - Replicate the noise regularizing model shown in https://openaccess.thecvf.com/content_CVPR_2019/papers/Hu_Weakly_Supervised_Image_Classification_Through_Noise_Regularization_CVPR_2019_paper.pdf and train your dataset.
  - Replicate the WSGAN shown in https://arxiv.org/pdf/2111.14605.pdf and train your dataset.
  - Compare the performance on your test data and explore the difference between two methods.
  - Develop a model which can leverage the adversarial learning architecture in GAN to replace the regularizing network to improve performance. Test your model by change the split level for clean labels starting from 10% and getting down as low as 5%. Find the cut off for which the model performance stops improving.
  - Dataset: https://pytorch.org/vision/stable/generated/torchvision.datasets.FashionMNIST.html (or any multilabel dataset of your choice)
- Responsible TA: Sayan Mandal

Reference:

S3: Cross-Modal Representation Learning (Cap 2)

- Introduction
  - Cross-modal representation learning is an essential part of representation learning, which aims to learn latent semantic representations for modalities including texts, audio, images, videos, etc.



Reference:

S3: Cross-Modal Representation Learning (Cap 2)

- Goal:
  - Develop a multi-modal neural network to learn cross-domain representations
- Project Outline:
  - The reference paper implements two different networks and trains them jointly with a contrastive loss to derive representations separately. In this project we are interested in developing and implement a multi-modal Variational Autoencoder that learns joint embeddings of face and voice from Voxceleb dataset.
  - Visualize high dimensional manifolds in different sub-spaces learnt from the VAE: image, audio and joint manifold.
  - Generate and analyze clusters for the representations learnt. Compare the clusters with the emotions predicted on the dataset using teacher model implemented in https://www.robots.ox.ac.uk/~vgg/research/cross-modal-emotions/
  - Dataset/Reference: https://www.robots.ox.ac.uk/~vgg/research/LearnablePins/
- Responsible TA: Sayan Mandal

Reference:

S4: Entity-Level Text Summarization (Cap 2)

- Introduction
  - Abstractive Text Summarization is the task of generating a short and concise summary that captures the salient ideas of the source text. The generated summaries potentially contain new phrases and sentences that may not appear in the source text.

Good quality summary output
S: a man charged with the murder last year of a british back-packer confessed to the slaying on the night he was charged with her killing , according to police evidence presented at a court hearing tuesday . ian douglas previte , ## , is charged with murdering caroline stuttle , ## , of yorkshire , england **T**: man charged with british backpacker 's death confessed to crime police officer claims **O**: man charged with murdering british backpacker con-fessed to murder
S: following are the leading scorers in the english premier league after saturday 's matches : ## - alan shearer -lrb- newcastle united -rrb- , james beattie . **T**: leading scorers in english premier league **O**: english premier league leading scorers
S: volume of transactions at the nigerian stock exchange has continued its decline since last week , a nse official said thursday . the latest statistics showed that a total of ##.### million shares valued at ###.### million naira -lrb- about #.### million us dollars -rrb- were traded on wednesday in #,### deals . **T**: transactions dip at nigerian stock exchange **O**: transactions at nigerian stock exchange down
Poor quality summary output

Reference:

S4: Entity-Level Text Summarization (Cap 2)

- Goal:
  - Summarize abstracts in scientific journal with an emphasis on retrieving the exact title keywords from the article
- Project Outline:
  - Develop or fine-tune a seq2seq model to summarize the text in an abstract from scientific journals.
  - Use beam search or similar methods to generate 1 sentence text which contains at least 2 main title keywords in the journal.
  - Augment the seq2seq model with another simple seq2seq network which takes 'Keywords' in the article as input and repeat the first two steps.
  - Compare the performance of both models using BLEU, ROUGE and Exact Match ratios. Did augmentation help in getting the 1 or 2 title keywords in the summary? How would you approach the problem instead? Implement your solution.
  - Datasets/reference:
    - https://elsevier.digitalcommonsdata.com/datasets/zm33cdndxs/2

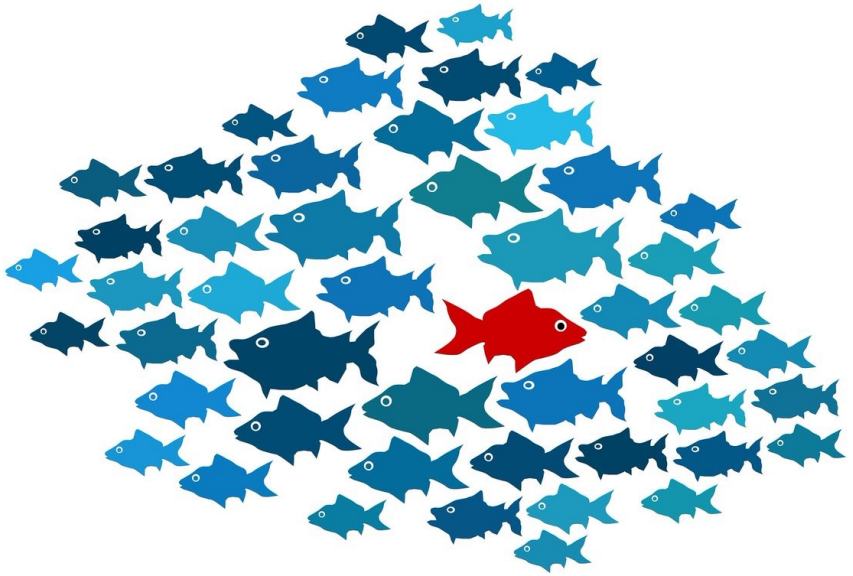
Note: 'Keywords' are simply keywords in the article, whereas title keywords are specific words in the title which makes the title unique.

Responsible TA: Sayan Mandal

Reference:

N1: Anomaly Detection (Cap 3)

- Introduction
  - Anomaly detection is a technique that uses AI to identify abnormal behavior as compared to an established pattern.
  - A common way to implement this is to train a neural network to learn some characteristic about a dataset of only normal examples, and to detect abnormal data when that characteristic differs from normality.



Reference:

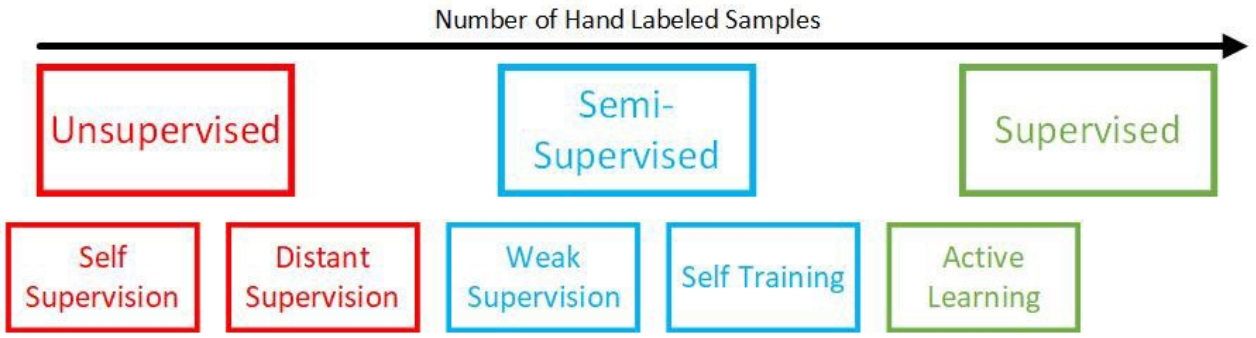
N1: Anomaly Detection (Cap 3)

- Goal:
  - Train a reconstruction autoencoder to perform anomaly detection and localization. Implement different metrics for anomaly detection and localization.
- Project outline:
  1. Train an autoencoder on one class (the normal class) of a simple dataset like CIFAR-10.
  2. Figure out how to use this autoencoder to detect when a test image is either (1) normal (from the training set class) or (2) anomalous (from some other class). Implement metrics of (1) area under the RO curve (image level AUC) and (2) maximum possible accuracy.
  3. Repeat the previous steps but for the more realistic, commonly used industrial anomaly detection benchmark of MVTec-AD (see here https://paperswithcode.com/sota/anomaly-detection-on-mvtec-ad). Can you train a model to work for this more challenging dataset?
  4. Figure out how to use your model to localize/segment anomalies in known anomalous images. Measure this by using your model to predict MVTec-AD's provided anomaly segmentation masks for it's anomalous datapoints (again, while only training your network on non-anomalous data!).
  5. Test anomaly localization by implementing different metrics for it, and compare and contrast their utility:
     1. Pixel AUC
     2. Pixel average precision
     3. Best possible Dice score (I.e, over all possible thresholds)
     4. best possible IoU
     5. best possible PRO (see https://arxiv.org/abs/2106.08265)
  6. Responsible TA: Nick Konz

Reference:

N2: Self- and Semi-Supervised/Contrastive Learning (Cap 3)

- Introduction
  - Self-supervised learning is a process where the model is trained on an automatically-defined *proxy task (*i.e., the model self-labels images), rather than on human labels.
  - Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data (learned via supervision) with a large amount of unlabeled data (learned via self-supervision) during training.



Reference:

N2: Self- and Semi-Supervised/Contrastive Learning (Cap 3)

- Goal:
  - Review self-supervised/contrastive learning literature and try to implement a method to learn the features of an unlabeled image dataset. See if you can use this for labeled image classification (semi-supervised learning).
- Project Outline:
  1. Literature review; to start:
     1. https://arxiv.org/abs/2002.05709
     2. https://arxiv.org/abs/2006.07733
     3. https://arxiv.org/abs/1911.05722
  2. Implement a self-supervised training procedure for a neural network/encoder for MNIST and CIFAR-10. *(hint: you'll want to use something that can work with limited memory; keep your batch size in mind).*
  3. Design a CNN/encoder with 3-4 hidden layers that outputs a 32-dimensional feature vector used for contrastive learning. Use this training procedure to pre-train your network on each of the two datasets.
  4. Use linear evaluation to prepare your pretrained network for classifying a labeled test set, using a small "finetuning" training set that only trains a last, appended layer in your network.
  5. Evaluated your self-supervised network on a classification test set and compare to the supervised result!
  6. Compare steps 3-5 for different encoder output feature dimensions, of 16, 32 and 64. You only need to do this for one of the datasets.
- Responsible TA: Nick Konz

Reference:

N3: Uncertainty Estimation with Dropout (Cap 2)

- Introduction
  - In many fields it is very important to have some sort of confidence/uncertainty score for a result. However, this is neither “built in” or trivial for neural networks, e.g. the maximum softmax score of a classifier is not a robust measure of predictions confidence.
  - Dropout can be to estimate neural network prediction uncertainty!

Figure 4: NYUv2 40-Class segmentation. From top-left: input image, ground truth, segmentation, aleatoric and epistemic uncertainty.

Reference:

N3: Uncertainty Estimation with Dropout (Cap 2)

- Goal/Motivation: Your task is to implement and analyze the use of Dropout layers in a convolutional neural network for prediction uncertainty estimation.
- Tasks:
    1. Start with a neural network for MNIST classification that you've already built in this course (e.g. in Discussion Section 4), and implement Dropout1D layers in the network.
    2. Use the dropout layers *at test time* to sample different predictions for a given image (I.e. using dropout to introduce randomness in your network). Analyze the distribution of these predictions, visually (e.g. with a prediction distribution plot), qualitatively and quantitatively. Use this to estimate a confidence interval for the network's predictions. Do your results make sense? Then, analyze the effects of the following experiments on these results:
        1. What is the effect of modifying the dropout probability hyperparameter?
        2. What happens when you turn off the dropout layers during training, but still use them at test time?
        3. What happens when you only use dropout at the first half, or second half, of the layers of the network?
        4. What happens when you use torch.nn.Dropout2D instead of 1D (where possible)? Determine a theoretical explanation for this behavior, based off of the difference between these dropout types.
        5. What happens when you test on a completely out-of-distribution image? To see, evaluate the confidence of your trained model on some images of the "chestmnist" subset of the MedMNIST medical image MNIST-like dataset (see here to get started: https://github.com/MedMNIST/MedMNIST/blob/main/examples/getting_started.ipynb).
        6. How does using dropout to estimate prediction uncertainty at test time compare to simply using the class softmax probability as a measure of confidence?
    3. Responsible TA: Nick Konz

(See here for more on the probabilistic/Bayesian interpretation of using dropout, but looking at this paper is **not** required, just for the curious)

H1: invisible backdoor attacks on iceberg classifiers (cap 2)

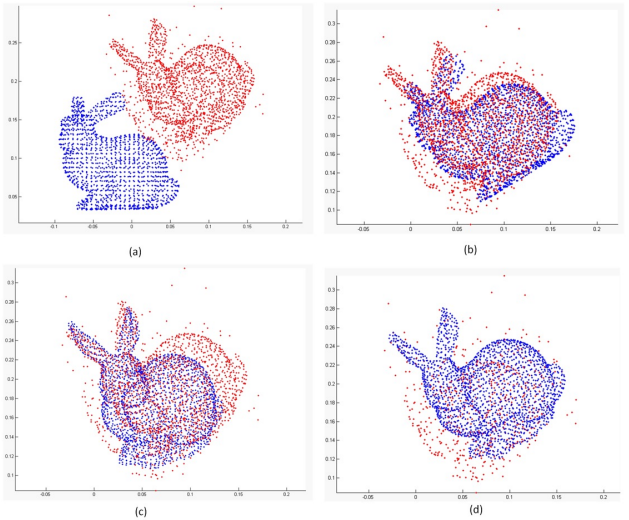
- Introduction
  - With the rapid development and broad application of deep neural networks (DNNs), backdoor attacks have gradually attracted attention due to their great harmfulness. Backdoor attacks are insidious, and poisoned models perform well on benign samples and are only triggered when given specific inputs, which then cause the neural network to produce incorrect outputs.

H1: invisible backdoor attacks on iceberg classifiers (cap 2)

- goals
  - Explore adversarial attack approaches for optical aerial images, and ways to defense against the attacks.
- Pipeline:
  - Literature review of Adversarial Attacks and defenses, get familiar with data poisoning and backdoor attacks
  - Train an iceberg/ship classification model on iceberg dataset https://www.kaggle.com/c/statoil-iceberg-classifier-challenge.
    - You could down-sample the images to 64*64 for usage.
  - Add a frequency domain backdoor attack, and train the DP model on the same task.
  - Try to defense this attack by implementing at least one approach of defense methods.
- Reference:
  - https://arxiv.org/pdf/2207.04209.pdf
- Responsible TA: Hanxue Gu

H2: point cloud registration (cap: 2)

- Introduction
  - Point Cloud Registration is a fundamental problem in 3D computer vision and photogrammetry. Given several sets of points in different coordinate systems, the aim of registration is to find the transformation that best aligns all of them into a common coordinate system. Point Cloud Registration plays a significant role in many vision applications such as 3D model reconstruction, cultural heritage management, landslide monitoring and solar energy analysis.



H2: point cloud registration (cap: 2)

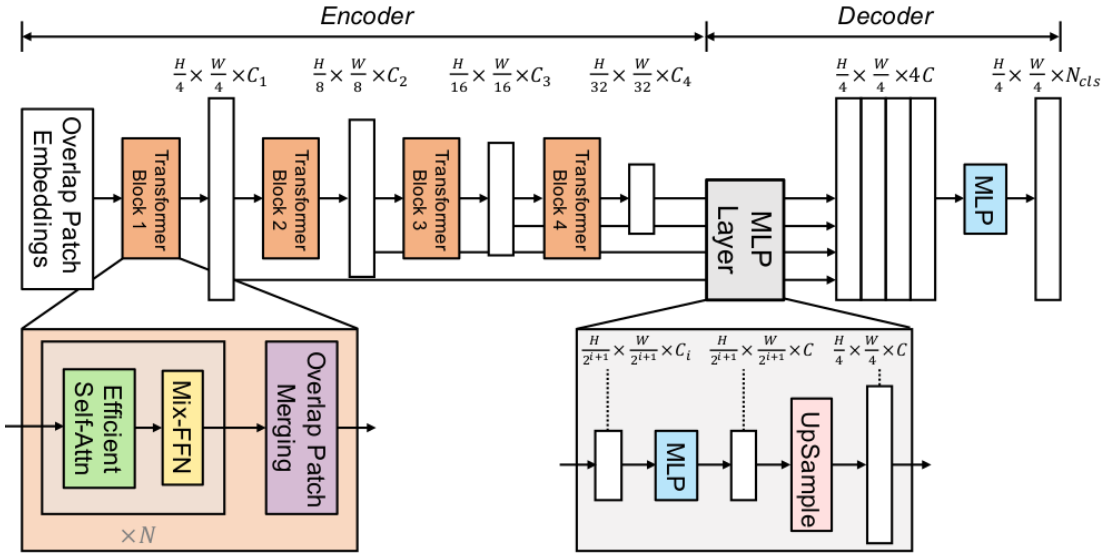
- Goal
  - Understand the point cloud registration and implement by yourselves.
- Pipeline
  - Read the literature of point cloud registration, starting from traditional non-DL methods such as Iterative closest point (ICP), to DL methods.
  - Understand how the DL-methods achieves the alignment and the optimization objectives.

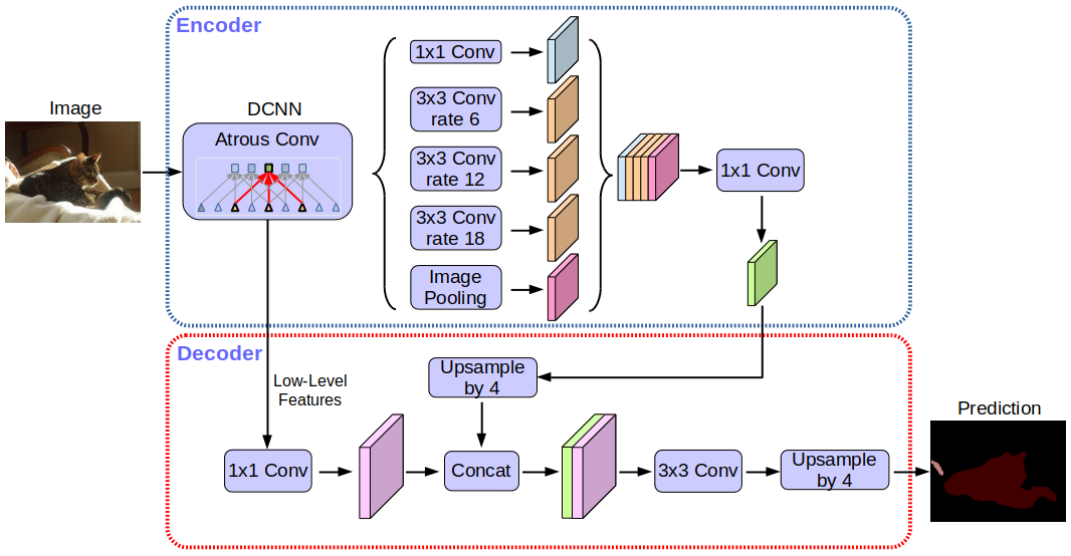
$$E(\boldsymbol{R}_{\mathcal{XY}}, \boldsymbol{t}_{\mathcal{XY}}) = \frac{1}{N}\sum_{i}^{N} \|\boldsymbol{R}_{\mathcal{XY}}\boldsymbol{x}_i + \boldsymbol{t}_{\mathcal{XY}} - \boldsymbol{y}_i\|^2.$$

  - Dataset: 3DMatch dataset
    - https://paperswithcode.com/dataset/3dmatch
  - Implement a DL method which shares a similar objective of ICP
    - Such as CorstNet, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8978671
    - Try to replace their feature extraction structure (they used PointNet).
  - Compare it to ICP.
- Responsible TA: Hanxue Gu

YX1: Convolution vs Attention for segmentation (Cap 4)

- Introduction
  - CNN has long been treated as de facto building component in the field of computer vision. Recently, however, transformer also comes into play and achieves SOTA performance on various vision tasks.





YX1: Convolution vs Attention for segmentation (Cap 4)

- Project goal
  - You will search for and implement the SOTA neural networks which uses CNNs and Transformers as their feature extraction backbones. (e.g., SegFormer, DeepLabv3+) You may only implement them using PyTorch Library. Make sure you comment your code.

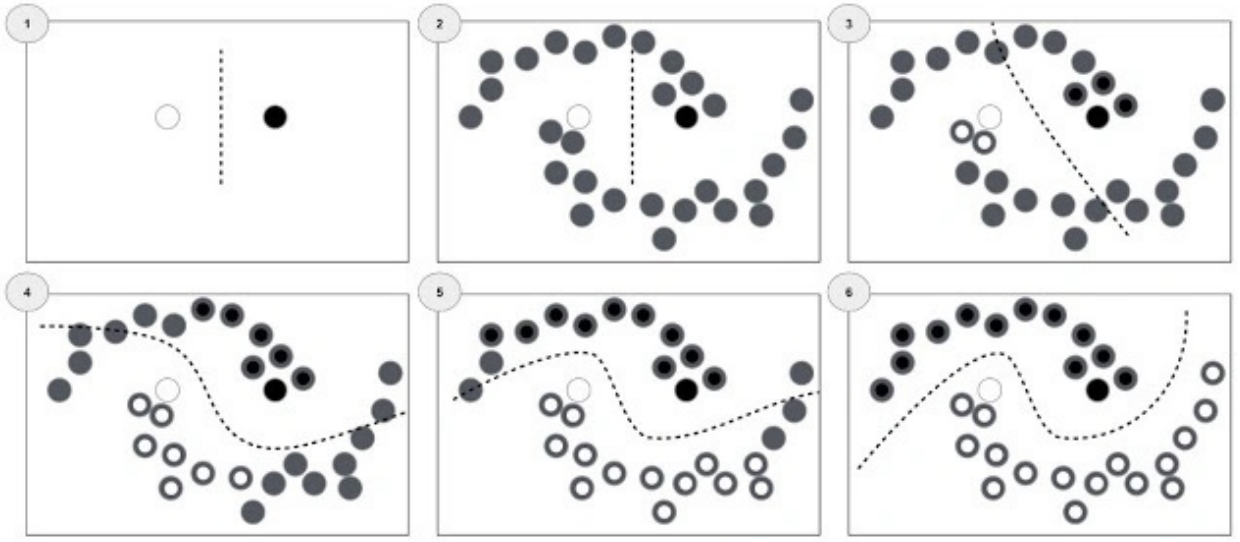
  Comparing their training, validation and testing loss on VOC 2012 and another datasets of your choice. (e.g., mIOU per class, error distribution, failing samples…). The dataset of your own choice should have at least 2000 images in total for training, validation and test sets.

  For each [model, dataset pair], you will also evaluate how pre-training and transfer learning may affect your model performance when only 1/8 of your dataset of interest is available for training. That means for each [model, dataset pair], you will initialize and train each of your model parameters 1. randomly 2. with some pre-trained weight from another dataset.
- Useful links:
  - Required: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) (ox.ac.uk)
    - Note: VOC2012 contains less than 3k official segmentation labels. You may add this SBD augmentation for additional training masks : SegmentationClassAug.zip (dropbox.com)
    - segmentation-paper-reading-notes/Summary of the semantic segmentation datasets.md at master · zhixuanli/segmentation-paper-reading-notes (github.com)
  - Potential models: Semantic Segmentation | Papers With Code
- Responsible TA: Yixin Zhang

YX2: Semi-Supervised Learning for segmentation (Cap 3-4)

- Introduction

Deep learning algorithms generally require large and representative datasets in order to achieve good performance with high generalizability. In a fully supervised learning setting, a ground-truth label is needed for each sample. A large, labeled dataset can hence be costly to obtain, especially in the field of medical imaging. This is especially expensive for the segmentation tasks, where pixel-wise label is need. Unlike labeled data – unlabeled data are rich in quantity and less costly to obtain. Semi-Supervised Learning (SSL) is a training technique designed to leverage the information in the large unlabeled dataset in conjunction with some small, labeled dataset for better generalizability



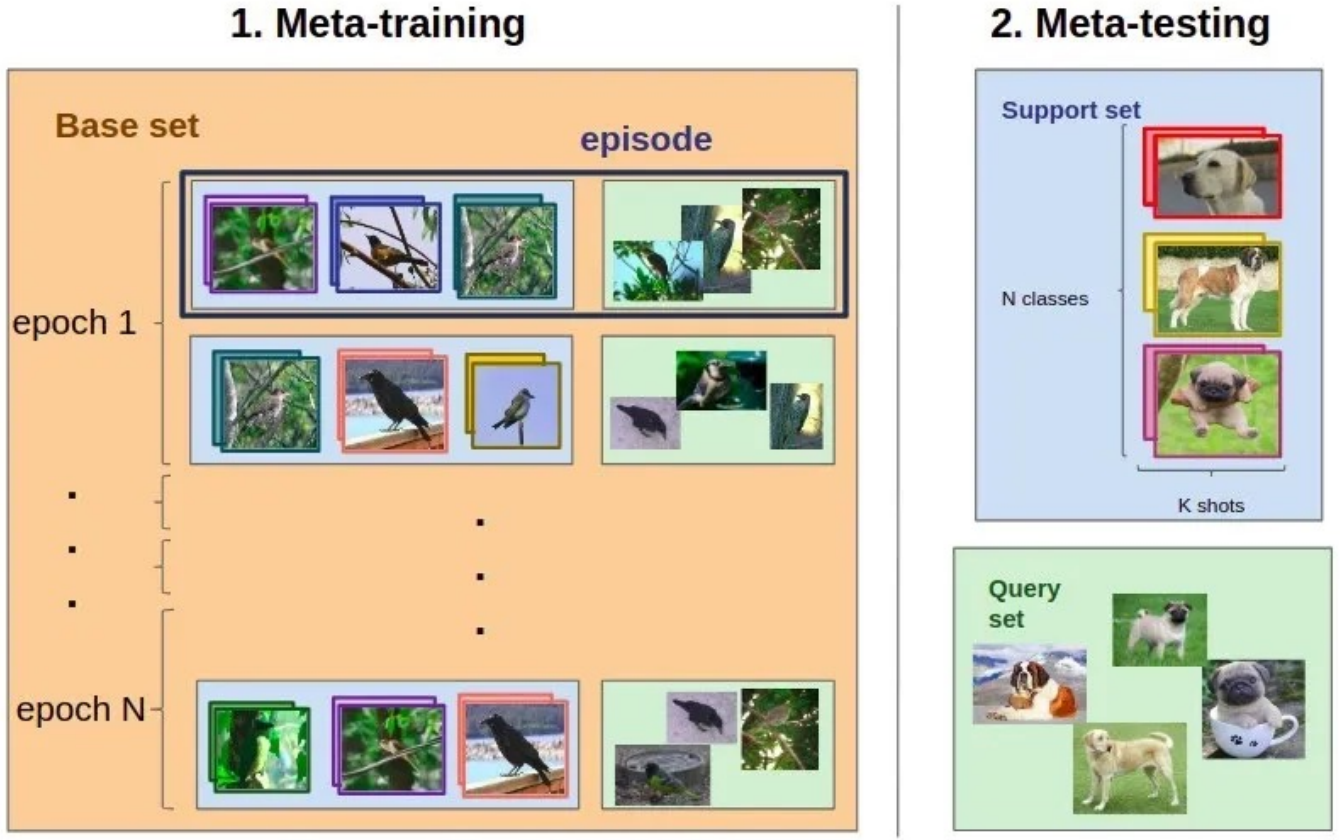
YX2: Semi-Supervised Learning for segmentaion (Cap 3-4)

- In this project, you will implement two algorithms in the realm of SSL for semantic segmentation and test them on two datasets respectively. You may choose your own datasets, and at least one dataset should have more than 10 classes. (e.g., VOC, COCO ...)
- If you use VOC,
  - The PASCAL Visual Object Classes Challenge 2012 (VOC2012) (ox.ac.uk)
    - Note: VOC2012 contains less than 3k official segmentation labels. You may add this SBD augmentation for additional training masks : SegmentationClassAug.zip (dropbox.com)
    - segmentation-paper-reading-notes/Summary of the semantic segmentation datasets.md at master · zhixuanli/segmentation-paper-reading-notes (github.com)
- You will also analyze the effects of pre-training on SSL for segmentation. Specifically, for each [SSL algorithm, dataset of interest] pair, you initialize your feature extraction backbone initialized randomly or with pre-trained weight respectively. The dataset used for pre-training should have minimal overlapping with the dataset of our interest.
- Responsible TA: Yixin Zhang

YX3: Few-Shot Learning (FSL) for classification (Cap 3)

- Introduction

Few-shot image classification is the task of doing image classification with only a few examples for each category (typically < 6 examples). It is a solution in an event where there is an extreme data scarcity.



YX3: Few-Shot Learning (FSL) for classification (Cap 3)

- Recommended literature: Electronics | Free Full-Text | Few-Shot Image Classification: Current Status and Research Trends (mdpi.com)

- In this project, you will do your own literature review. You are required to understand and implement at least two common training settings/mechanisms to train a classifier for few-shots learning. Analyse the performance of each training setting/mechanism under one-shot, five-shot and ten-shots learning on two support & query datasets of your pick from those listed below:
  - 1. BIRDS 450 SPECIES- IMAGE CLASSIFICATION | Kaggle
  - 2. Cat Breeds Dataset | Kaggle
  - 3. Butterfly Image Classification 75 species | Kaggle
  - 4. ImageNet 1k
  - 5. 30 Musical Instruments -Image Classification | Kaggle

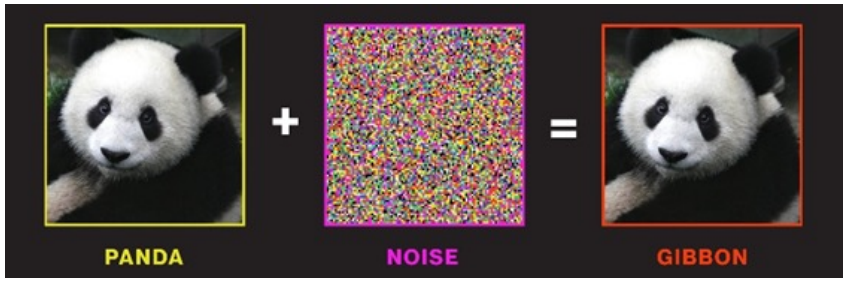
  You may skip classes with less than 100 images in your support and query sets during evaluation.

  Responsible TA: Yixin Zhang

YX4: Black Box Adversarial Attack (Cap 3-4)

- Introduction

In Deep Learning, adversarial attack is a process of generating deceptive data that lead to the malfunction of a neural network. However, such data should be hardly able to fool humans. A classic example of this is the panda to gibbon case as described in the image below.



There are two primary types of adversarial attacks to neural networks: white-box attack and black-box attack. White box attack allows access to model weights. Direct gradient optimization is allowed in this case. However, black box attack allows access to more limited information.

YX4: Black Box Adversarial Attack (Cap 3-4)

- In this project, you will review the literatures relevant to **black box attack** and implement at least two **black box attack** method on some ImageNet pre-trained models. (You may use the built-in models in PyTorch and their pre-trained weights) You would then compare the two methods and explain the general concepts behind black box attack. You may also check to see if the adversarial samples you obtained also fool models with similar performance but different architectures.
- You may use this literature as a starting point

Adversarial Machine Learning in Image Classification: A Survey Towards the Defender's Perspective (arxiv.org)

- You need to: conduct a black box adversarial attack on ImageNet 1k. You should list the metrics you used for optimization and the demonstrate the intensity and locations where the attack occurs.
- You need also: train a classifier with satisfactory testing performance on at least one of the following datasets and conduct attack on it.
    - 1. Tom and Jerry Image classification | Kaggle
    - 2. Butterfly Image Classification 75 species | Kaggle
    - 3. food-11 Image Classification Dataset | Kaggle
- Responsible TA: Yixin Zhang