# CS771A Mini-Project 2: Continual LwP Model Generation on CIFAR-10 Dataset.

**Anirban Sarkar** 241110009
**Nayan Das** 241110045
**Shekhar Pratap Singh** 241110066
**Sushant Gangurde** 241110073
**Junth Basnet** 241110090

## Problem Description

### Task 1 Description

Task 1 involves training a sequence of models $f_1, f_2, \ldots, f_{10}$ using the labeled dataset $D_1$ and progressively updating them with predicted labels from unlabeled datasets $D_2, D_3, \ldots, D_{10}$. Each model is evaluated on held-out labeled datasets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{10}$. The goal is to maintain high accuracy on the current dataset and prevent significant performance degradation on previous datasets.

### Task 2 Description

Task 2 involves incrementally training a series of models $f_{11}, f_{12}, \ldots, f_{20}$ using datasets $D_{11}, D_{12}, \ldots, D_{20}$. Each model is evaluated on its corresponding held-out dataset as well as all previous held-out datasets. The results are presented in a matrix format to compare the performance across different datasets.

## Approach Description

### Feature Extraction

We employed a feature extraction process using a pre-trained **ConvNeXt Tiny** model. The model, pre-trained on ImageNet, provides robust feature representations that generalize well across similar datasets. The feature extractor uses the penultimate layer of the network to generate meaningful representations of the input images, reducing them to compact and informative vectors.

**Steps:**

1. Input images from the datasets are resized to $224 \times 224$ pixels using OpenCV to match the input size requirement of the pre-trained model.
2. The resized images are passed through the ConvNeXt Tiny model, and outputs from the penultimate layer are extracted.
3. This process converts raw image data into high-dimensional feature vectors, which are stored for subsequent processing.

Raw datasets ($D_1$ to $D_{20}$) were initially in PyTorch tensor format. They were converted to NumPy arrays and augmented using the ConvNeXt-based feature extractor. Each dataset is saved as a `.npz` file in argumented dataset folder containing:

- `data`: Feature vectors extracted from the ConvNeXt Tiny model.
- `targets` (if available): Ground truth labels.

### LwP Implementation

We implemented the Learning With Prototype (LWP) model, a simple classifier that leverages class prototypes. For each class, the model maintains a prototype vector, which is updated incrementally during training using a weighted moving average. This method combines the existing prototype with the new training sample in proportion to their respective contributions: the previous prototype is weighted by the number of samples already seen for that class, while the new sample is weighted inversely by the total number of samples for the class. This ensures that the prototype evolves as more data is observed, reflecting the overall distribution of the class. By updating prototypes incrementally, the model achieves efficient continual learning without storing past samples. During prediction, it computes distances (or similarities) between the input samples and the prototypes of all classes. The class with the closest prototype is predicted.

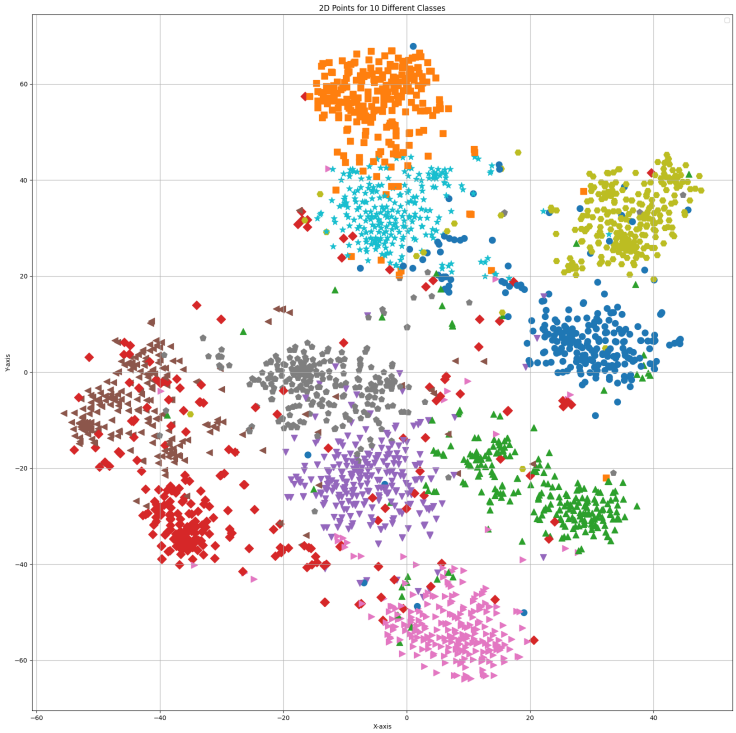


Figure 1: t-SNE plot on $D_1$ dataset

## Continual Model Generation

### Task 1

**Initial Training:** A custom LwP (Learning with Prototypes) classifier $f_1$ is trained on $D_1$ using the extracted features and their corresponding labels.

**Model Update Mechanism:** For each unlabeled dataset $D_i$ ($i \geq 2$):

1. Predictions are generated for the dataset using the current model $f_{i-1}$.
2. Predictions with high confidence (above a threshold, e.g., 0.7) are selected as pseudo-labels.
3. These pseudo-labeled examples are used to update the model, creating $f_i$.

This ensures that new data enhances the model while minimizing noisy updates from low-confidence predictions. All models $f_1$ to $f_{10}$ have a consistent architecture and parameter count, ensuring comparability and scalability.

### Task 2

**Initial Training:** Starting with $f_{10}$ as the initial model that was saved as f10_model.pkl file, we proceed with incremental updates to learn models $f_{11}, f_{12}, \ldots, f_{20}$. The datasets $D_{11}, D_{12}, \ldots, D_{20}$ have slightly different input distributions, requiring an adaptation of the update mechanism from Task 1.

**Model Update Mechanism:** For each dataset $D_i$ ($i \geq 11$):

1. Predictions are generated for the dataset $D_i$ using the current model $f_{i-1}$.
2. Predictions with high confidence (above a threshold, e.g., 0.99999) are selected as pseudo-labels.
3. These pseudo-labeled examples are used to update the model, creating $f_i$, while also considering dataset distribution differences.
    - A domain adaptation technique is applied to address potential shifts in data distributions between $D_i$ and $D_{i-1}$.

This ensures that new data improves the model while maintaining stability across varying input distributions. Models $f_1$1 to $f_{20}$ have a consistent architecture and parameter count.

## Evaluation Results

### Evaluation of Task 1

**Evaluation Datasets:** Each model $f_i$ is evaluated on its corresponding held-out dataset $\hat{D}_i$ and all preceding held-out datasets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{i-1}$.

**Accuracy Matrix:** A $10 \times 10$ accuracy matrix is generated to record the performance of $f_i$ on $\hat{D}_j$ ($1 \leq j \leq i$):

- Rows represent models $f_1, f_2, \ldots, f_{10}$.
- Columns represent datasets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{10}$.

| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 |
|---|---|---|---|---|---|---|---|---|---|---|
| f1 | 0.8732 | | | | | | | | | |
| f2 | 0.8648 | 0.8792 | | | | | | | | |
| f3 | 0.8636 | 0.878 | 0.8704 | | | | | | | |
| f4 | 0.8604 | 0.8772 | 0.8684 | 0.872 | | | | | | |
| f5 | 0.8596 | 0.876 | 0.8688 | 0.8728 | 0.8712 | | | | | |
| f6 | 0.8580 | 0.8764 | 0.8684 | 0.8728 | 0.87 | 0.8624 | | | | |
| f7 | 0.8564 | 0.8756 | 0.8688 | 0.872 | 0.8716 | 0.8628 | 0.86 | | | |
| f8 | 0.8572 | 0.8748 | 0.8684 | 0.8692 | 0.8704 | 0.8616 | 0.8584 | 0.8692 | | |
| f9 | 0.8568 | 0.8744 | 0.8684 | 0.8704 | 0.8712 | 0.8608 | 0.8584 | 0.8684 | 0.8692 | |
| f10 | 0.8560 | 0.8752 | 0.8688 | 0.8688 | 0.8708 | 0.8596 | 0.8584 | 0.8684 | 0.8704 | 0.8764 |

Figure 2: Evaluation Matrix of Task 1

The approach ensures minimal performance degradation on earlier datasets as the models are sequentially updated.

### Evaluation of Task 2

**Accuracy Matrix:** A $10 \times 20$ accuracy matrix is generated to record the performance of models $f_{11}, f_{12}, \ldots, f_{20}$ on held-out datasets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{20}$:

- Rows represent models $f_{11}, f_{12}, \ldots, f_{20}$.
- Columns represent held-out datasets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{20}$.

This matrix highlights the trade-off between adapting to new datasets and retaining performance on earlier datasets.

| | D'1 | D'2 | D'3 | D'4 | D'5 | D'6 | D'7 | D'8 | D'9 | D'10 | D'11 | D'12 | D'13 | D'14 | D'15 | D'16 | D'17 | D'18 | D'19 | D'20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| f11 | 0.8556 | 0.8744 | 0.8672 | 0.8680 | 0.8692 | 0.8580 | 0.8568 | 0.8676 | 0.8684 | 0.8740 | 0.7332 | | | | | | | | | |
| f12 | 0.8528 | 0.8744 | 0.8668 | 0.8672 | 0.8684 | 0.8560 | 0.8580 | 0.8660 | 0.8672 | 0.8728 | 0.7332 | 0.512 | | | | | | | | |
| f13 | 0.8528 | 0.8732 | 0.8660 | 0.8660 | 0.8684 | 0.8552 | 0.8576 | 0.8664 | 0.8660 | 0.8724 | 0.7312 | 0.5112 | 0.7784 | | | | | | | |
| f14 | 0.8512 | 0.8728 | 0.8656 | 0.8648 | 0.8688 | 0.8560 | 0.8560 | 0.8656 | 0.8656 | 0.8720 | 0.7300 | 0.5104 | 0.7788 | 0.8232 | | | | | | |
| f15 | 0.8516 | 0.8720 | 0.8660 | 0.8656 | 0.8668 | 0.8552 | 0.8540 | 0.8652 | 0.8636 | 0.8704 | 0.7312 | 0.5116 | 0.7772 | 0.8228 | 0.8504 | | | | | |
| f16 | 0.8492 | 0.8732 | 0.8664 | 0.8644 | 0.8664 | 0.8552 | 0.8532 | 0.8640 | 0.8636 | 0.8696 | 0.7344 | 0.5128 | 0.7756 | 0.8216 | 0.8508 | 0.7236 | | | | |
| f17 | 0.8500 | 0.8728 | 0.8652 | 0.8640 | 0.8668 | 0.8552 | 0.8532 | 0.8640 | 0.8632 | 0.8688 | 0.7336 | 0.5112 | 0.7756 | 0.8216 | 0.8508 | 0.7248 | 0.7736 | | | |
| f18 | 0.8492 | 0.8720 | 0.8636 | 0.8628 | 0.8660 | 0.8552 | 0.8528 | 0.8632 | 0.8628 | 0.8688 | 0.7320 | 0.51 | 0.774 | 0.8224 | 0.8492 | 0.7232 | 0.7724 | 0.7416 | | |
| f19 | 0.8484 | 0.8724 | 0.8648 | 0.8620 | 0.8664 | 0.8536 | 0.8536 | 0.8620 | 0.8612 | 0.8680 | 0.7308 | 0.5104 | 0.774 | 0.8212 | 0.8484 | 0.7224 | 0.7724 | 0.7388 | 0.6496 | |
| f20 | 0.8492 | 0.8708 | 0.8640 | 0.8628 | 0.8648 | 0.8524 | 0.8540 | 0.8608 | 0.8624 | 0.8672 | 0.7320 | 0.5116 | 0.7736 | 0.8216 | 0.8464 | 0.7224 | 0.7716 | 0.7384 | 0.6468 | 0.8284 |

Figure 3: Evaluation Matrix for Task 2

**Performance Analysis:** The accuracy matrix is analyzed to identify trends and patterns in model performance.

1. After training $f_i$ on $D_i$, evaluate $f_i$ on the corresponding held-out dataset $\hat{D}_i$.
2. Additionally, evaluate $f_i$ on all preceding held-out datasets $\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{i-1}$.
3. Compare the performance of $f_i$ on earlier held-out datasets against $f_{i-1}$ to ensure minimal degradation.

## Conclusion

Based on the accuracy matrices from both Task 1 and Task 2, we can draw several conclusions about the performance and progression of the models:

### Task 1 Summary

- **Model Progression**: The models $f_1$ through $f_{10}$ were trained sequentially on datasets $D_1$ to $D_{10}$. Each model was evaluated on its corresponding held-out dataset as well as all previous datasets.
- **Accuracy Trends**: The accuracy generally decreases as the model is applied to datasets it was not directly trained on, which is expected due to the potential differences in data distribution.
- **Final Model Performance**: The final model, $f_{10}$, achieved a reasonable accuracy across all datasets, indicating that the model was able to generalize well from the sequential training process.

### Task 2 Summary

- **Model Progression**: Starting with $f_{10}$ from Task 1, models $f_{11}$ through $f_{20}$ were trained on datasets $D_{11}$ to $D_{20}$. Each model was evaluated on its corresponding held-out dataset as well as all previous datasets.
- **Accuracy Trends**: Similar to Task 1, the accuracy tends to decrease when models are evaluated on datasets they were not directly trained on. However, the models show a consistent ability to maintain reasonable accuracy across multiple datasets.
- **Final Model Performance**: The final model, $f_{20}$, shows a good level of accuracy across the board, suggesting that the sequential training approach continues to be effective in Task 2.

### Overall Conclusion

- **Sequential Training Effectiveness**: The sequential training approach used in both tasks appears to be effective in building models that can generalize across multiple datasets. This is evidenced by the relatively high accuracy scores across both tasks.
- **Model Generalization**: The models demonstrate a strong ability to generalize, as seen by their performance on datasets they were not directly trained on.
- **Potential for Improvement**: While the models perform well, there is always room for improvement, particularly in handling datasets with significantly different distributions or characteristics.

These results suggest that the methodology used is robust and can be applied to similar tasks where sequential learning and evaluation are required.

## Problem 2

Liu, Chenxi, et al. "DEJA VU: Continual Model Generalization For Unseen Domains." [Online]. Available: `https://arxiv.org/abs/2301.10418`, 2023.

**Youtube link:** `https://www.youtube.com/watch?v=-QMVvgShvB0`

## Acknowledgments

We would like to express our deepest gratitude to Professor Piyush Rai for designing this insightful CS771A Machine Learning course and for providing clear guidance and thought-provoking challenges throughout this mini-project. His expertise in the field of machine learning, coupled with his comprehensive approach to teaching, has been instrumental in our learning journey. We also thank the teaching assistants for their support in answering questions and clarifying doubts during the course and the project. We acknowledge the importance of the tools and frameworks that were integral to our project, including Scikit-learn, Keras, and TensorFlow, for their role in helping us implement machine learning models effectively.

## References

- Liu, Zhuang, et al. "A ConvNet for the 2020s." *CoRR*, Volume abs/2201.03545, 2022. [Online]. Available: `https://arxiv.org/abs/2201.03545`
- Liu, Chenxi, et al. "DEJA VU: Continual Model Generalization For Unseen Domains." [Online]. Available: `https://arxiv.org/abs/2301.10418`, 2023.
- He, Kaiming, et al. "Deep Residual Learning for Image Recognition." [Online]. Available: `https://arxiv.org/abs/1512.03385`, 2015.
- Chollet, François, et al. "Keras: Deep Learning library for Theano and TensorFlow." 2015. [Online]. Available: `https://keras.io`
- Chollet, François. *Deep Learning with Python.* 2nd ed., Manning Publications, 2021.
- Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 3rd ed., O'Reilly Media, 2022.
- Stanford CS229: Machine Learning Course — Summer 2019 (Anand Avati). Stanford Online, [YouTube Playlist]. Available: `https://www.youtube.com/watch?v=KzH1ovd4Ots&list=PLoROMvodv4rNH7qL6-efu_q2_bPuy0adh`
- MIT 6.S191: Convolutional Neural Networks. MIT OpenCourseWare, [YouTube Video]. Available: `https://www.youtube.com/watch?v=2xqkSUhmmXU`
- Deep Learning With Tensorflow 2.0, Keras. [YouTube Playlist]. Available: `https://www.youtube.com/playlist?list=PLeo1K3hjS3uu7CxAacxVndI4bE_o3BDtO`