Machine Learning Final Project: **Machine Unlearning**

Aashai Avadhani
Masters Applied Data Science Candidate

Problem Statement:

- **Goal:** Create an **unlearning algorithm** that can maintain accuracy on the test/validation dataset without the "deleted" data

- Deep Learning Models rely on more training data and mroe parameters to create higher accuracy
  - **GPT 4 from OpenAI** is trained on **45 GB of data** with **1.8 trillion parameters**[1]

- **Privacy Policies** cause users to delete their data from systems cause **deep learning models to have poor performance** and not compliant with Privacy Policies
  - GDPR in Europe allows users to remove their data

- **Data erasures from systems** require models to be retrained on an **entire dataset consistently** which is expensive

- **Unlearning algorithms** take in a pretrained model and "unlearn" the dataset from the deep learning model

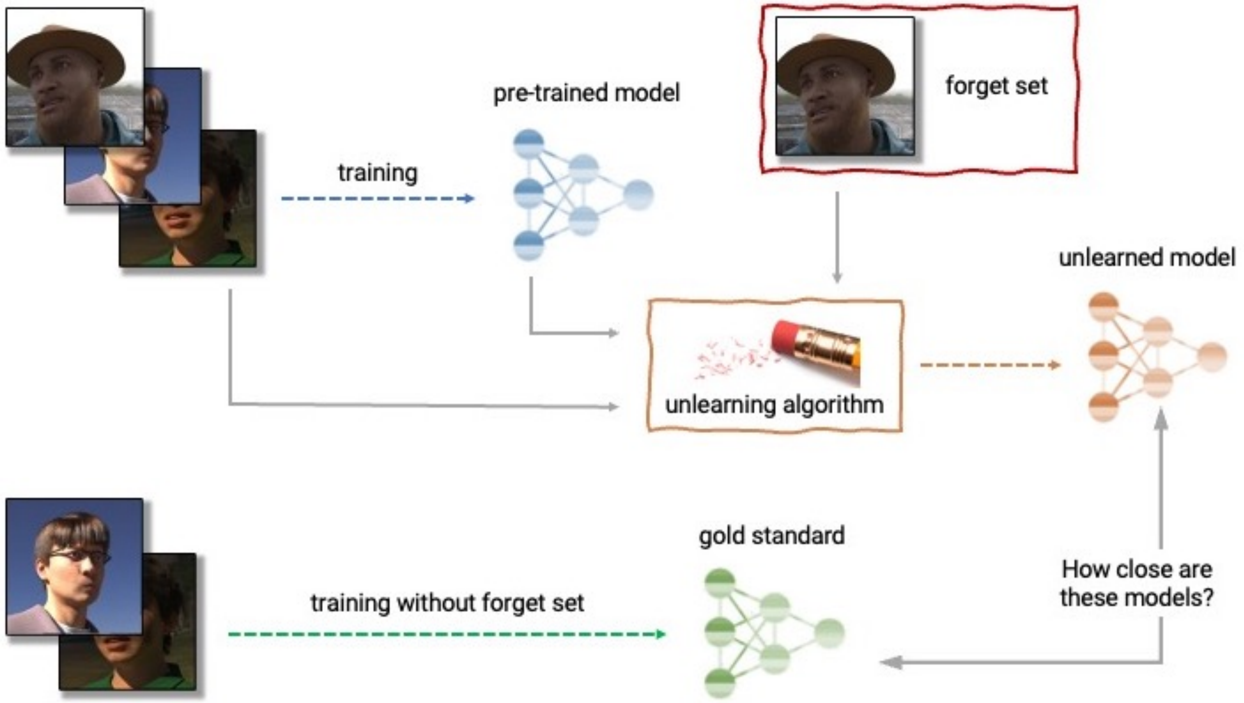


Figure 1: Overall Machine Unlearning Process that shows how an unlearning model is compared vs a model trained on the "forget" dataset

Data Overview and Approach

- **Based from the [Google Neurips Challenge]**
- **Data**
  - CIFAR-10
  - **X: 32x32 color images of 10 classes**
  - **y: Label of each image (Truck, Car etc….)**
- **Model** Pre-trained model ResNet18
  - Convolutional Neural Network that predicts the label of an image
- **Unlearning algorithm (Each Approach has a slide)**
  - **Pruning**
  - **KL loss approach**
- **Evaluation**
  - Compare between the Unlearned Algorithm and the model that is trained on the “retrain set”
  - **Metrics:** Train Accuracy, Test Accuracy, MIA attack
  - **Membership Inference Attacks** determines the probability of

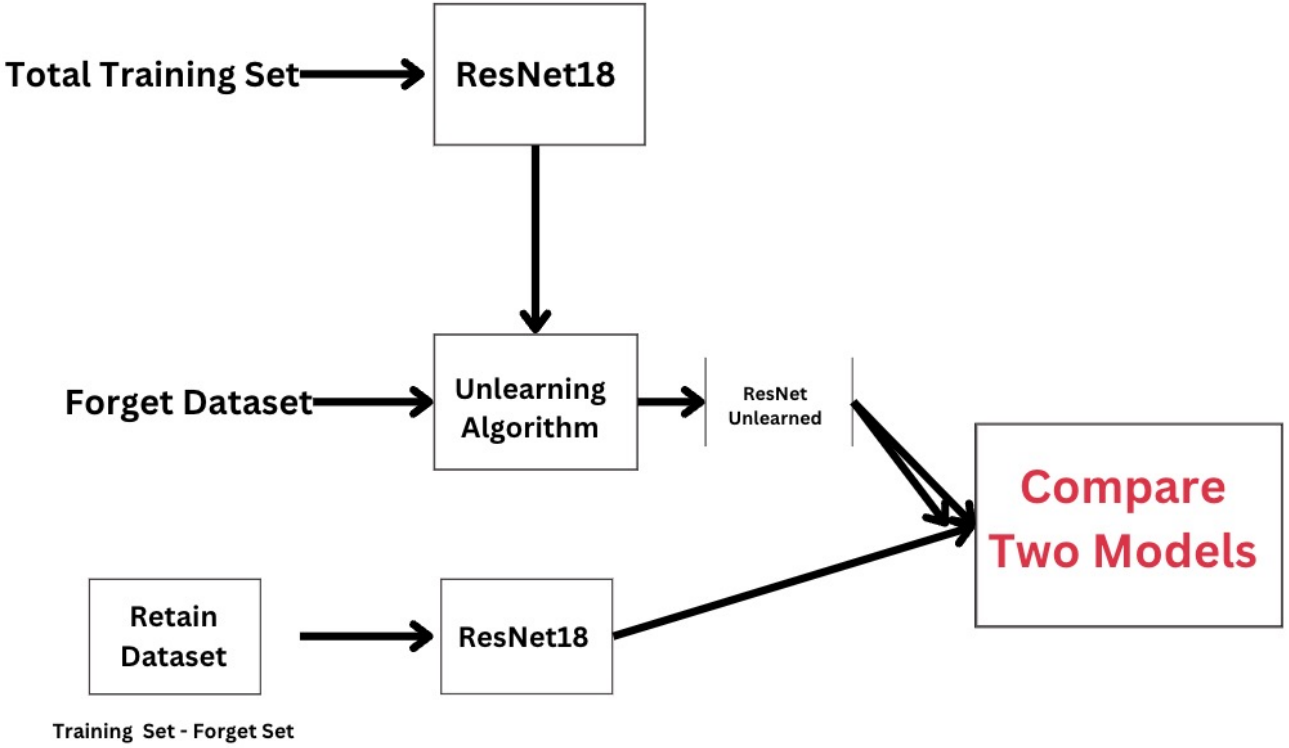


Figure 5: Initial Output of a ResNet18 Model being trained on

	Training Accuracy	Test Set Accuracy
Resnet18	**98.03%**	**84%**

Exploratory Data Analysis

CIFAR 10 Dataset

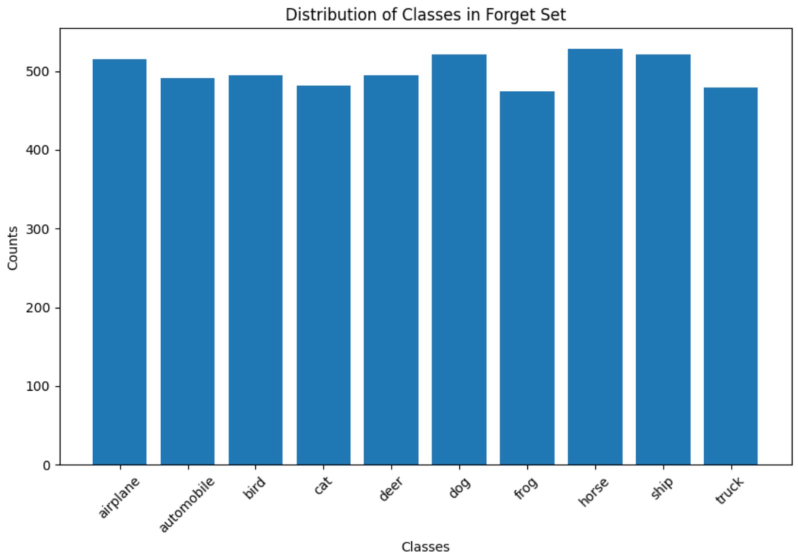
Random Sampled to create the “Forget” and “Retain” Set from the Training Distribution

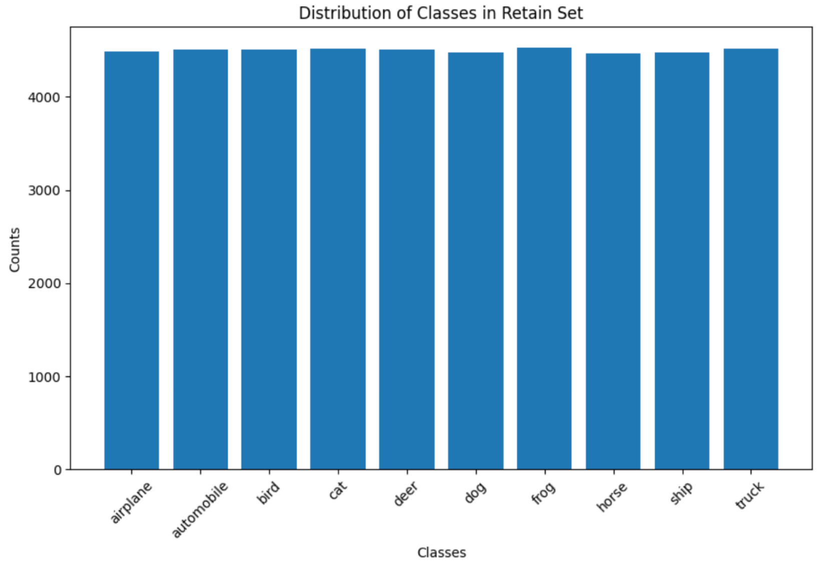
The **Retaining** set has an equal amount of each **class of image**

Each set according to the T-SNE has a more distribution on animals (deers, cats, birds) than other transportation images (trucks, ships)

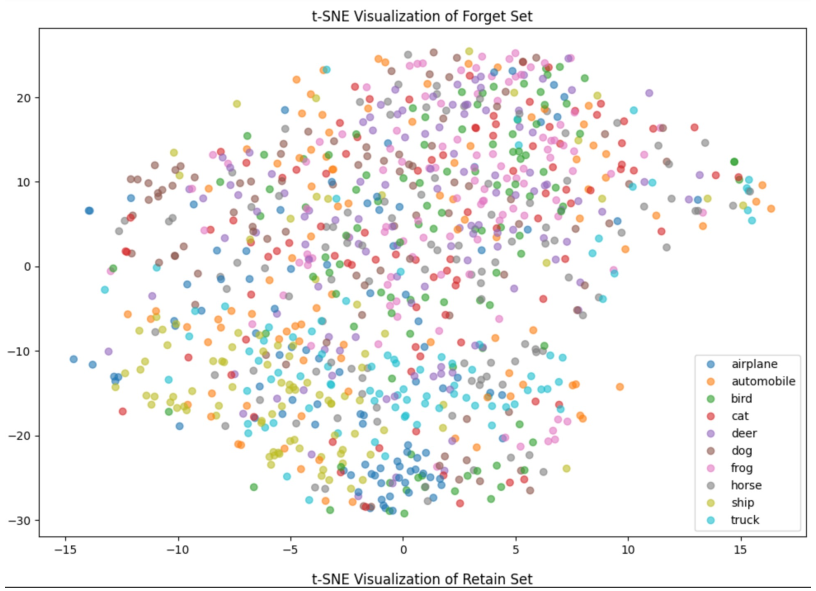
The **ResNet model** will be skewed to animals than transportations

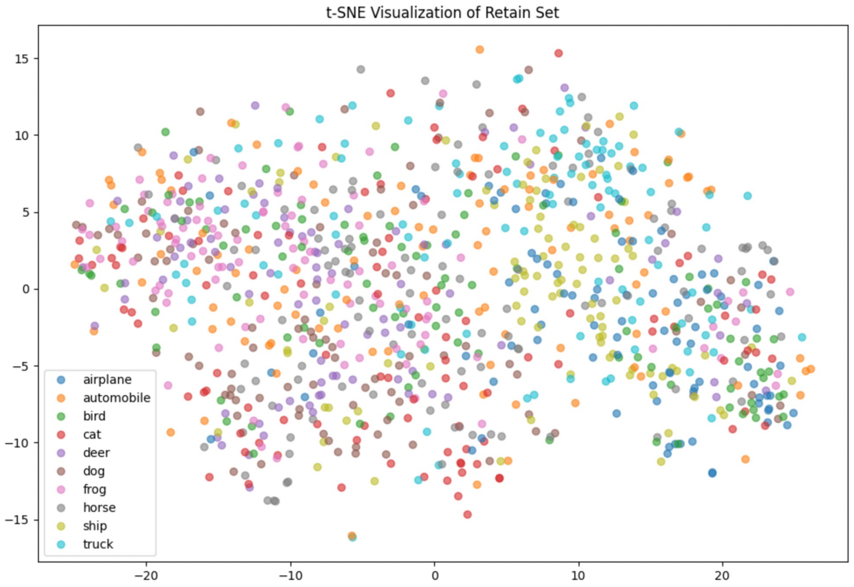
Figure 1: Forget set has an unbalanced number of classes while the retain set has an equal amount of class distribution





Forget set has a lot more deer and other images close together





Feature Engineering and Transformation

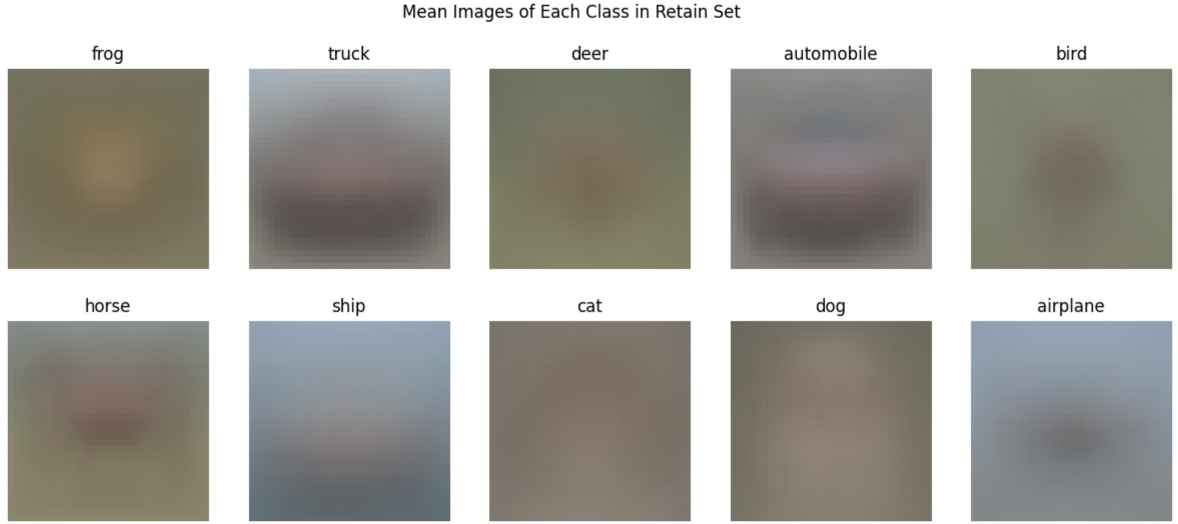
- **The Feature Engineering** main goal was to ensure the ResNet18 model is not tuned in noisy sample data

- **The mean** image of each dataset was a blurry picture (as seen in the right)

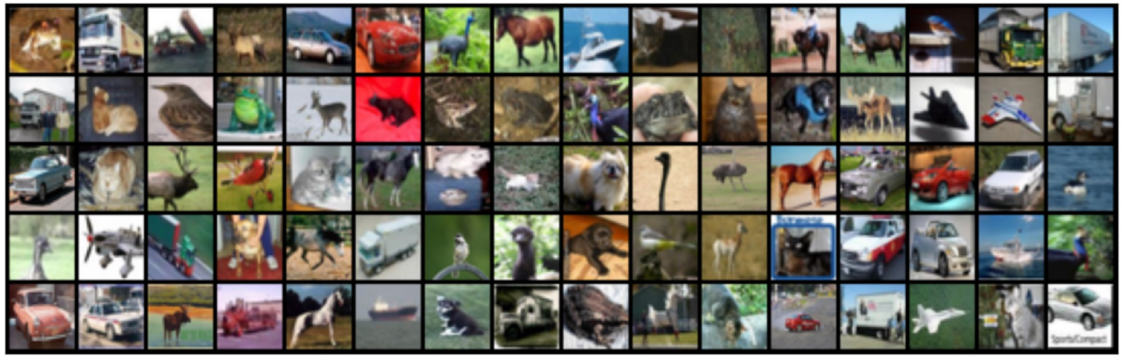
- **The current state of the** dataset would result in the Resnet18 model being fine-tuned on noise rather than the core image class

- **Transformation:** Normalization image using a Normalizing Tensor function

- **Post Transformation:** you can see how the images are slightly more clear which gives more protection against the risk of fine-tuning on noisy sample data

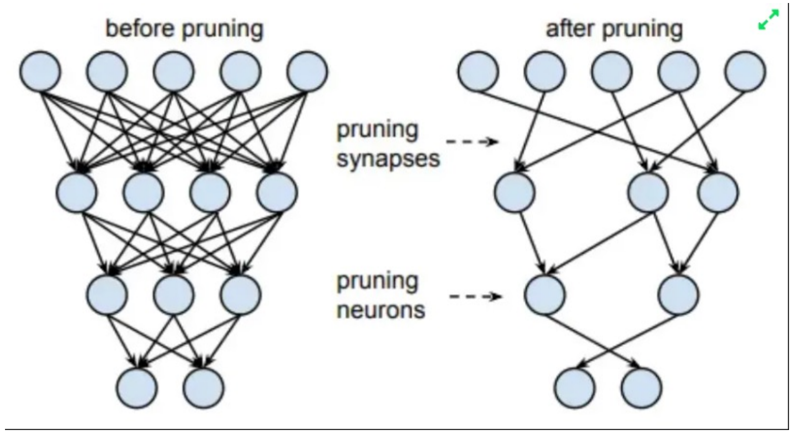



```
normalize = transforms.Compose(
    [
        transforms.ToTensor(),
        transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
    ]
)

train_set = torchvision.datasets.CIFAR10(
    root="./data", train=True, download=True, transform=normalize
)
train_loader = DataLoader(train_set, batch_size=128, shuffle=True, num_workers=2)

held_out = torchvision.datasets.CIFAR10(
    root="./data", train=False, download=True, transform=normalize
)
test_set, val_set = torch.utils.data.random_split(held_out, [0.5, 0.5], generator=RNG)
test_loader = DataLoader(test_set, batch_size=128, shuffle=False, num_workers=2)
val_loader = DataLoader(val_set, batch_size=128, shuffle=False, num_workers=2)
```



Proposed Modeling Approaches for the UnLearning Algorithm

- **CNN Pruning**
  - Deep Learning Models can be pruned to reduce the amount of neurons used per training
  - Prasandi Paper on Pruning Modeling
  - Use L1 Regularization from each neuron to zero out the ineffective neurons



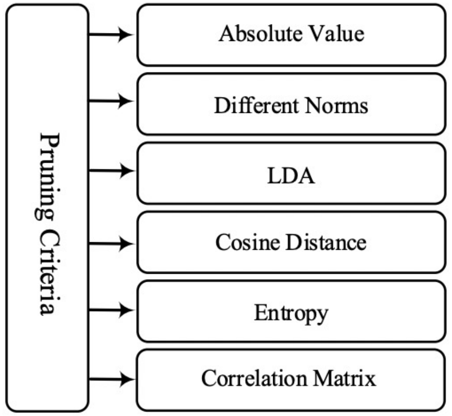


Fig. 5. Pruning criteria

Figure 6: Model Pruning from CNNs about each of the neurons

- **2 Step "Dummy" Labels**
  - 1st Infer the forget data from the pre-trained model
  - Use a simple unlearning (provided by google which is the SGD loss fine tuning)
  - Finetune the pretrained model with the identified data and incorrect inferences as "dummy labels"

Evaluating through Membership Inference Attacks

- **Membership Inference Attack**
  - **Logistic Regression** that predicts whether the model was trained on a particular sample from that samples loss
  - We will use it to predict whether or not **the model was trained using the training set or the retain set**
- Needed as a layer to evaluate the new weights of the
- Similar to a discriminator in a GAN, we aim for the MIA to be **50% scoring since that is "randomly guessing"**
- **Objective: Reach and MIA score of close to 0.50**

Figure 9: The current Train vs Test loss for the pretrained model

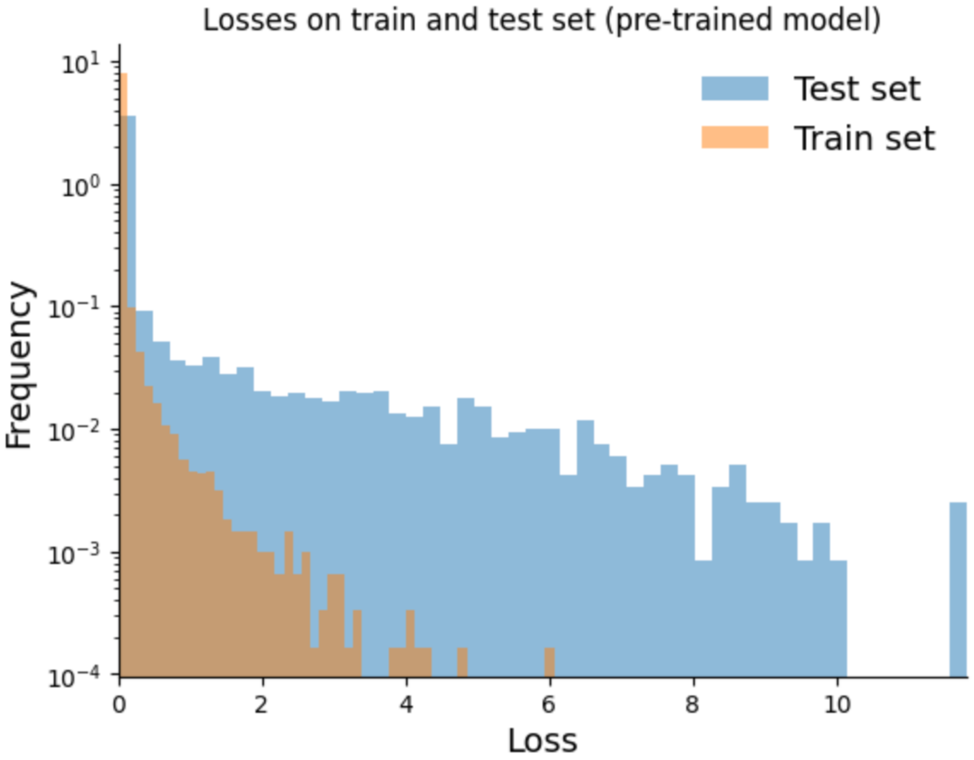


Figure 10: Mmembership inference score with the Pre-trained Resnet18 model

	Training Accuracy	Test Set Accuracy	MIA Score
Resnet18	**98.03%**	**84%**	**0.579**

Final UnLearning Results and Proposed Solution

- **CNN Pruning** retained the most training and test accuracy
  - **However** it did not have the best MIA score which means the model is slightly more prone to not “unlearning” the entire dataset

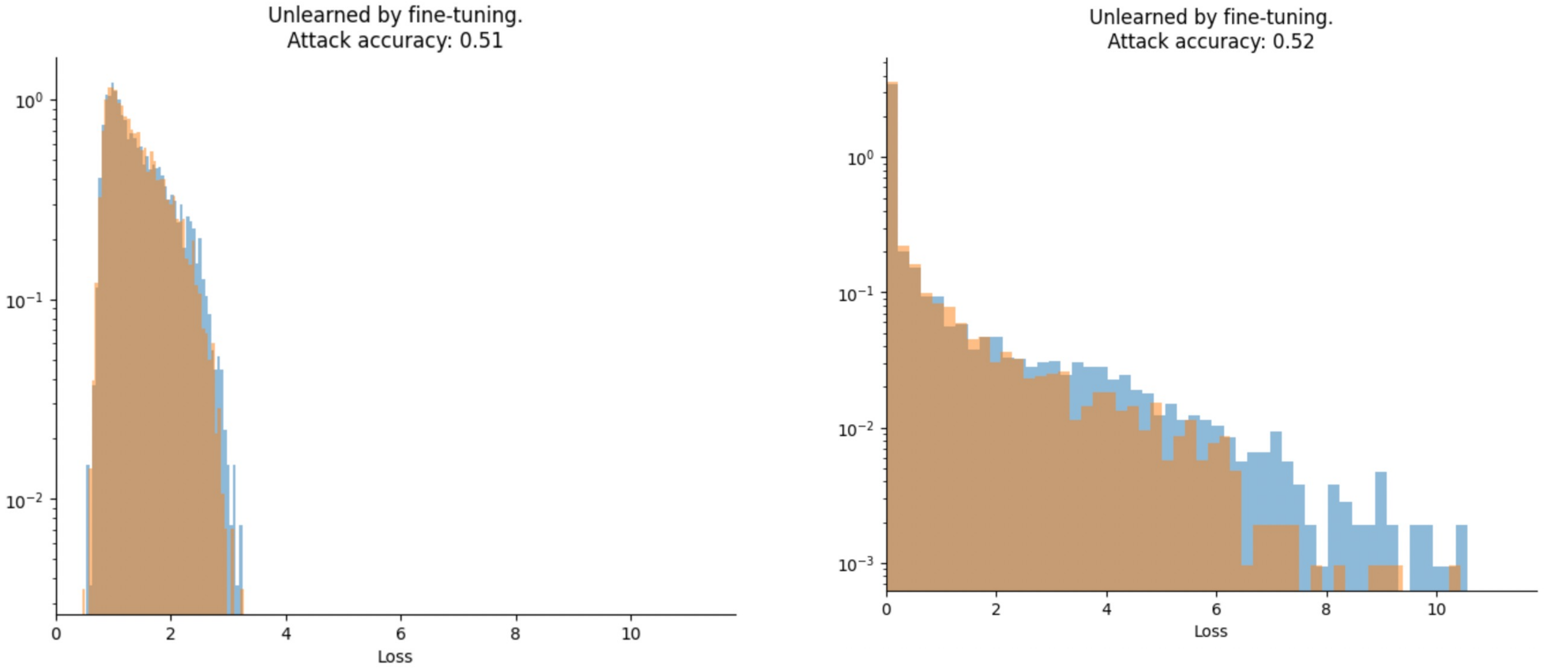
- **Dummy Labels** had the least accuracy but the lowest MIA score
  - This is due to the first inference step to completely forget all the data which eliminates those weights from the beginning

Chosen Methodology: CNN Pruning

Table: The various results of the unlearning algorithms

	Train Accuracy	**Test Accuracy**	**Retain Accuracy**	**MIA Score**
Pre-Trained Resnet	98.5%	84.3%	90%	0.578
CNN Pruning	93.7%	82.1%	93.5%	**0.52**
Dummy Labels	78.3%	72.9%	13,8%	**0.505**

Figure 7: To the left is the MIA prediction for Dummy Labels while to the right is CNN pruning



Future Work

- **Choosing CNN pruning** since it is the more intuitive and we only tried L1 regularization for the weights

- Further work would be building on top of the pruning algorithm to try other pruning algorithms such as cosine distance (similarity between weights and the sample distribution)

- **Generating synthetic data** that could anonymize or “mimic” the forgotten dataset rather than completely unlearning the algorithm

- **Expand to text data** and analyze the performance of transformers loss functions with unlearning algorithms

- Many Use cases will revolve around personalized content/recommendation systems which require a different pruning criteria (correlation matrix)

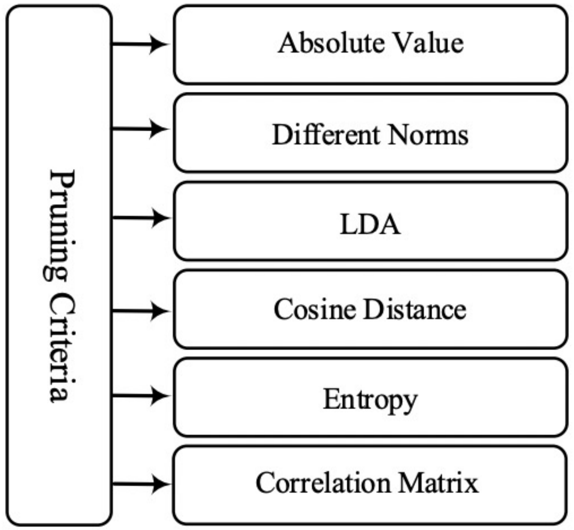


Fig. 5. Pruning criteria