TIME INSPIRED DECAY MECHANISM FOR MACHINE UNLEARNING WITH IMPORTANCE ESTIMATION

Anirudh Kaluri

Instructor: Dr. Jung-Eun Kim

Hypothesis

In image classification tasks, high-accuracy for a particular class is due to a fraction of parameters and filters that are important to that class. A time-decay of the magnitude of these parameters simulates weakening or fading of memory in Humans. This will result in the accuracy drop for the class.

TESTING THE HYPOTHESIS

1. Train a model with a dataset. Importance is estimated during inference.
2. Remove each weight/filter and calculate new accuracy by inferring forget class.
3. Determine the top-K important parameters for the forget class.

Importance of Parameter = Original Accuracy - Accuracy after removing Parameter

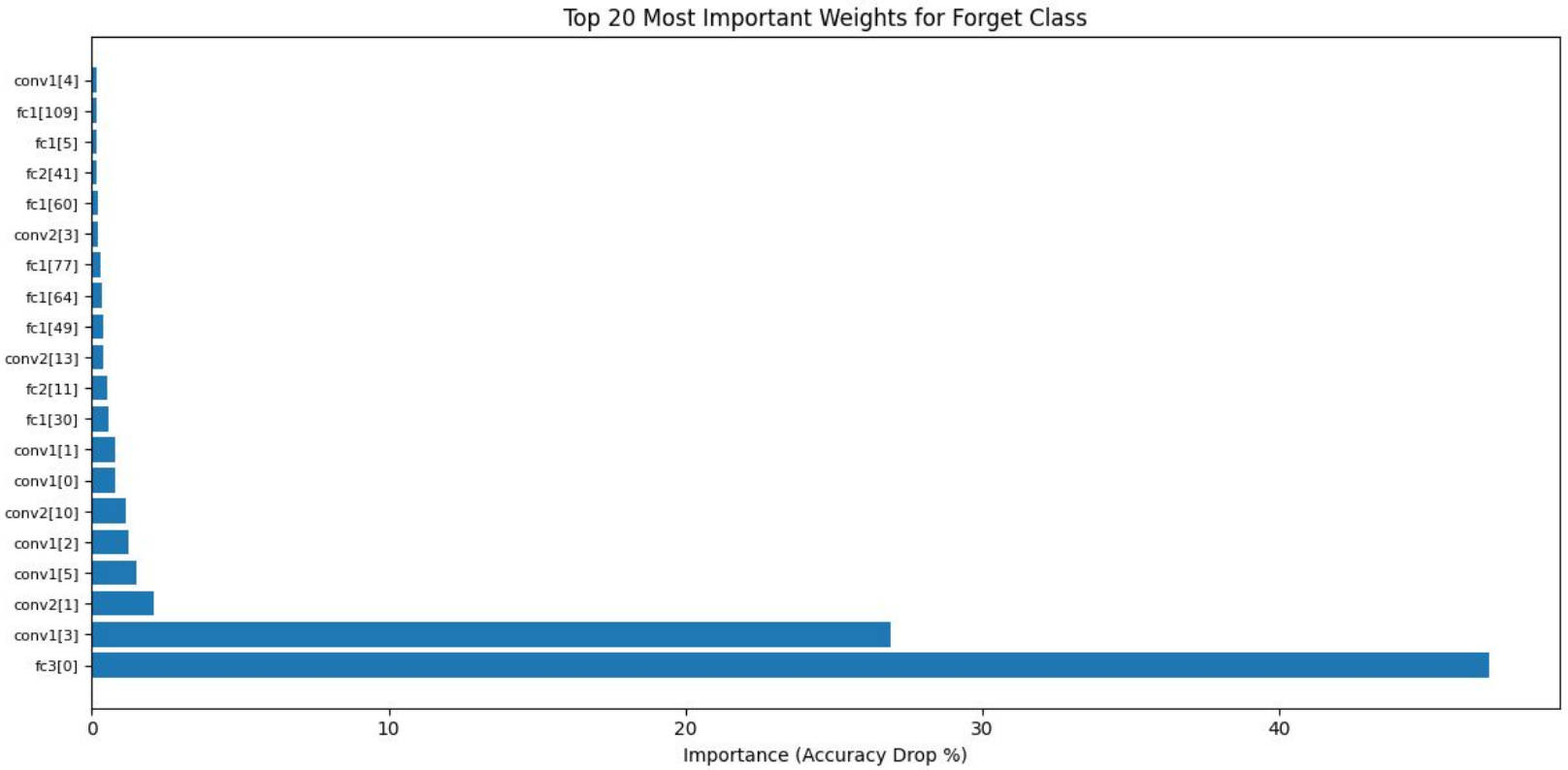
4. For each important parameter, decay the weights with the following equation:

$$\theta_i(t+1) = \theta_i(t) \cdot e^{-\lambda \Delta t}$$

5. Calculate new accuracies with and without fine-tuning on retain training dataset.

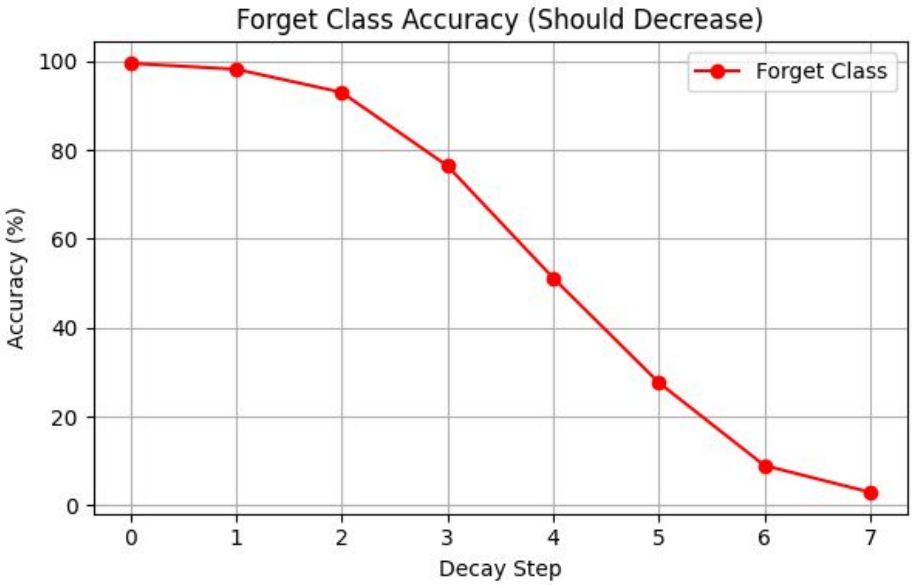
Testing the Hypothesis- LeNet MNIST

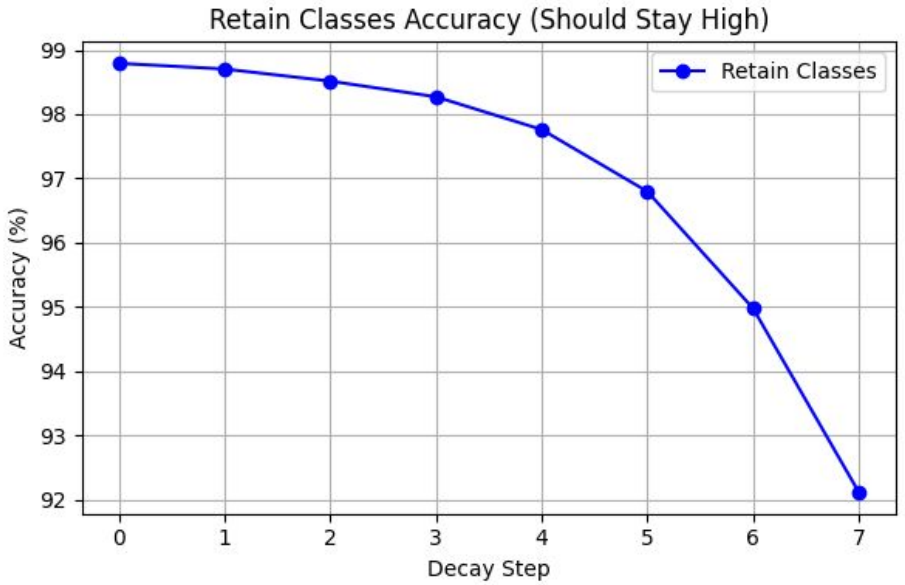
Forget Class=[0]
Retain Class=[1,2,3,4,5,6,7,8,9]
Fine-tuning= False



Testing the Hypothesis- LeNet MNIST

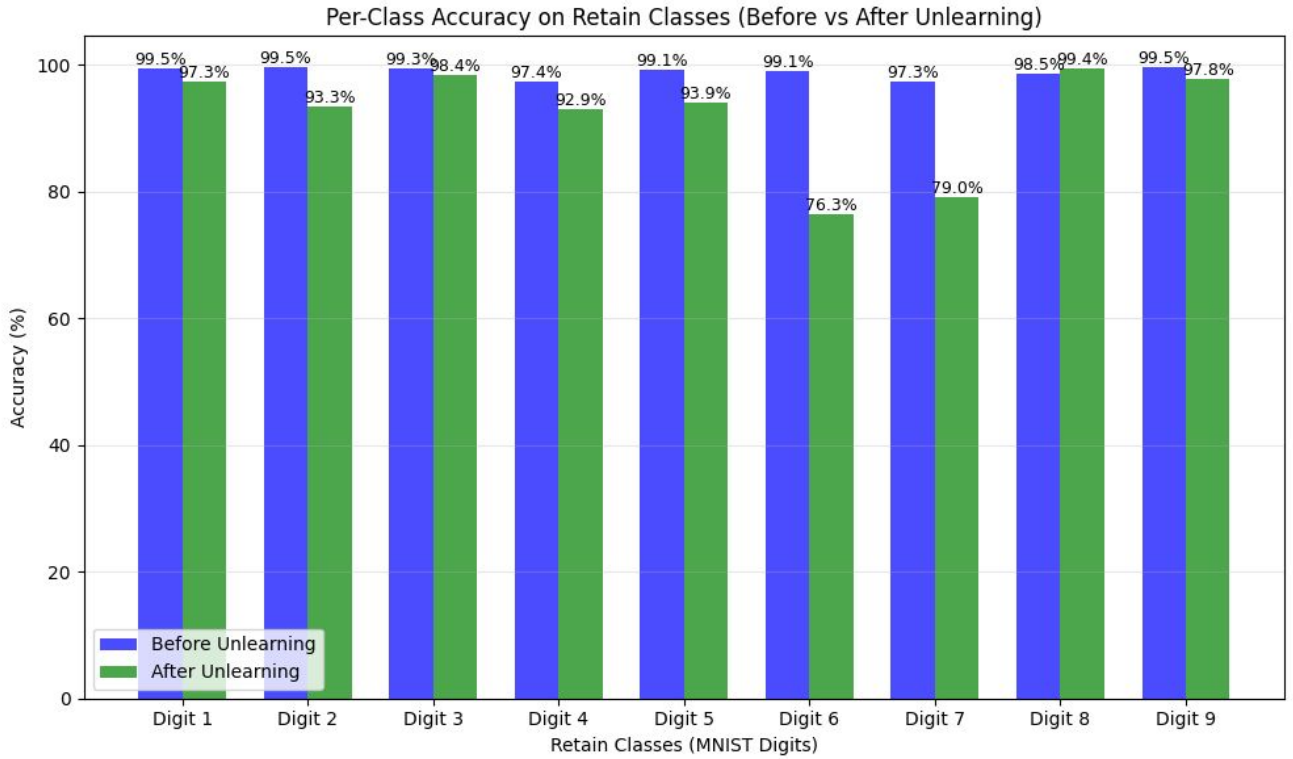
Forget Class=[0]
Post-unlearning accuracy Forget Class= 2.96%
Retain Class=[1,2,3,4,5,6,7,8,9]
Post-unlearning accuracy Retain Class= 92.12%
Fine-tuning= False





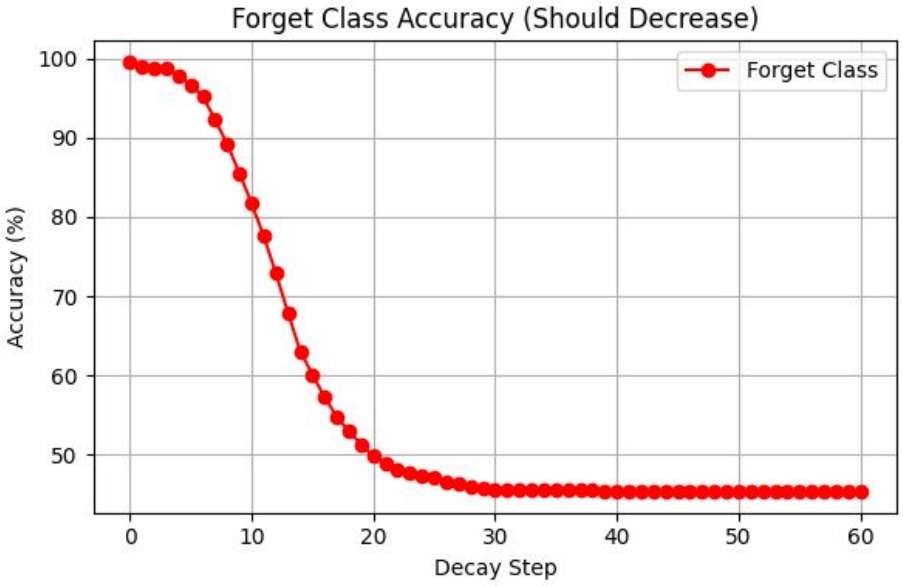
Testing the Hypothesis- LeNet MNIST

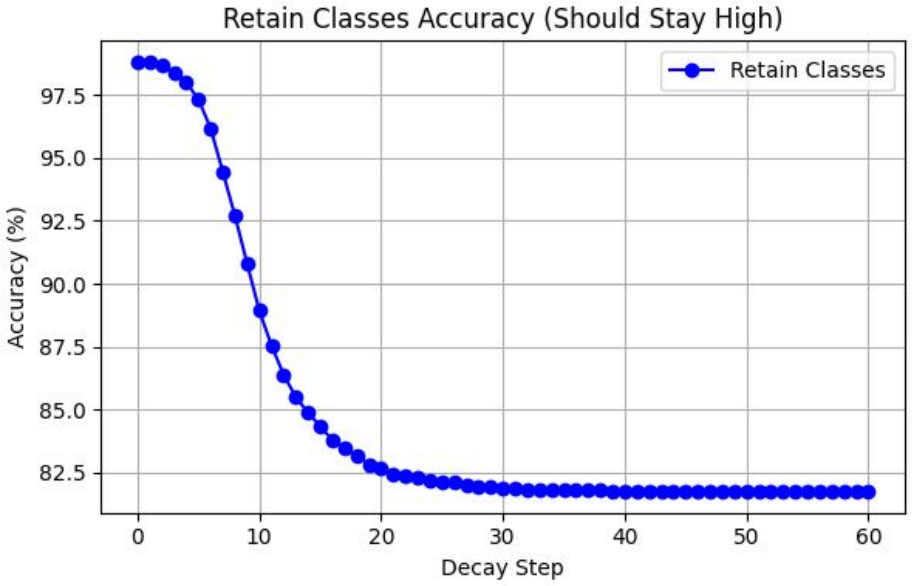
Forget Class=[0]
Post-unlearning accuracy Forget Class= 2.96%
Retain Class=[1,2,3,4,5,6,7,8,9]
Post-unlearning accuracy Retain Class= 92.12%
Fine-tuning= False



Testing the Hypothesis- LeNet MNIST

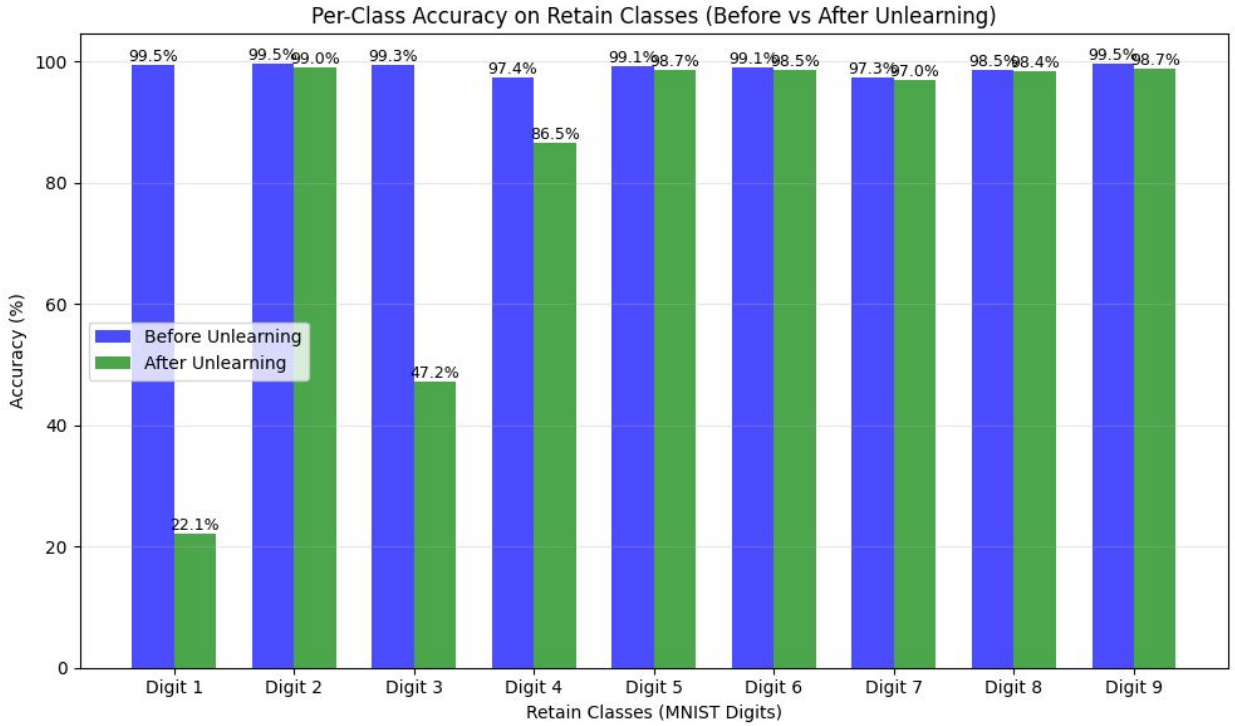
Forget Class=[0]
Post-unlearning Accuracy= 45.41%
Retain Class=[1,2,3,4,5,6,7,8,9]
Post-unlearning Accuracy=81.75%
Fine-tuning= False
Decay Random weight/filters instead of Important weight/filters
Forget Class accuracy still high, Retain Class accuracy dropped





Testing the Hypothesis- LeNet MNIST

Forget Class=[0]
Post-unlearning Accuracy= 45.41%
Retain Class=[1,2,3,4,5,6,7,8,9]
post-unlearning Accuracy=81.75%
Fine-tuning= False
Decay Random weight/filters instead of Important weight/filters
Affects accuracies of other classes severely



Testing the Hypothesis- LeNet MNIST

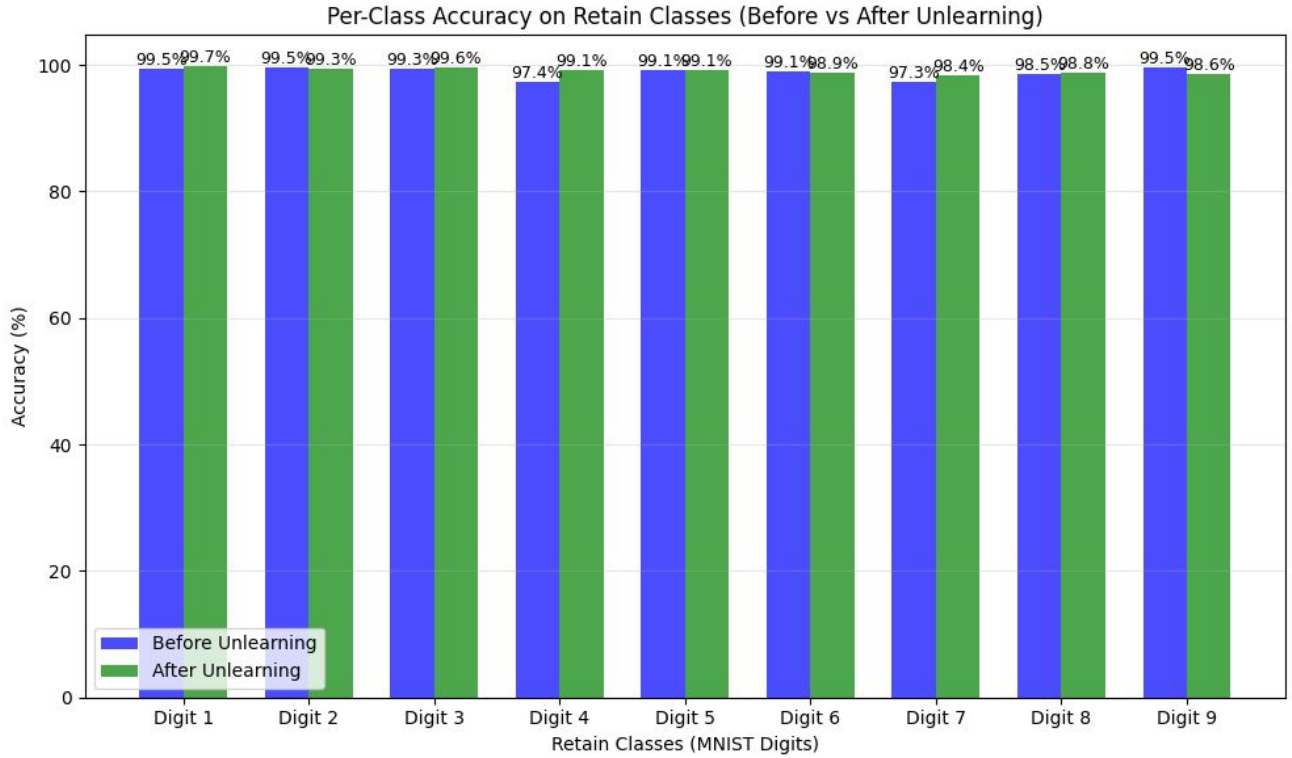
Till now no fine-tuning after weight decay. Lets finetune now

2 ways to fine-tune:

1. Finetune after all decay steps
2. Finetune after each decay step

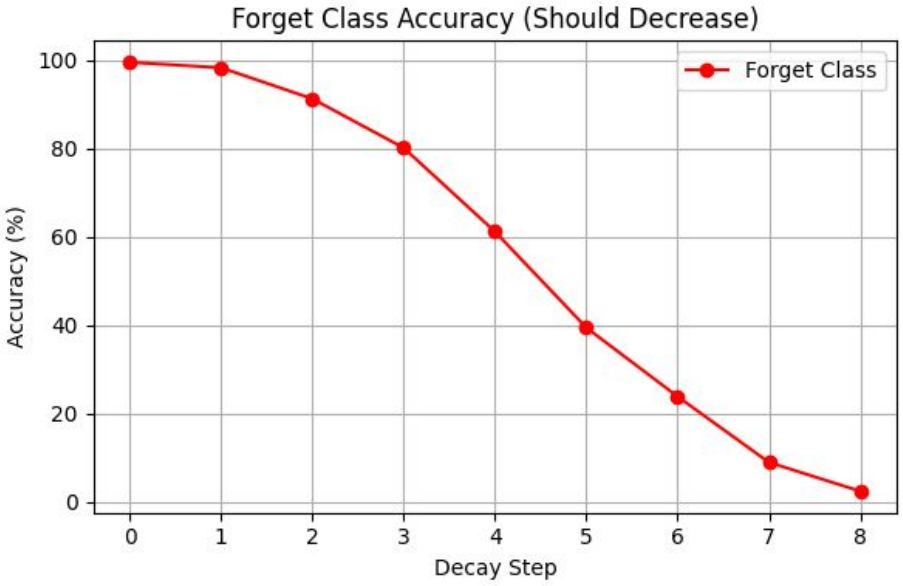
Testing the Hypothesis- LeNet MNIST

Lets fine-tune now….
Forget Class=[0],
Retain Class=[1,2,3,4,5,6,7,8,9],
Fine-tuning after all 7 decay steps with 1 epoch of retain train data
Post-unlearning+Fine Tuning Forget Class Accuracy: 0.61%
Post-unlearning+Fine Tuning Retain Classes Accuracy: 99.07%



Testing the Hypothesis- LeNet MNIST

Forget Class=[0]
Retain Class=[1,2,3,4,5,6,7,8,9],
Fine-tuning after each decay step with 21 batches each step
Post-unlearning + Fine-tuning Forget Class Accuracy: 2.35%
Post-unlearning + Fine-tuning Retain Classes Accuracy: 98.66%



Testing the Hypothesis- LeNet MNIST

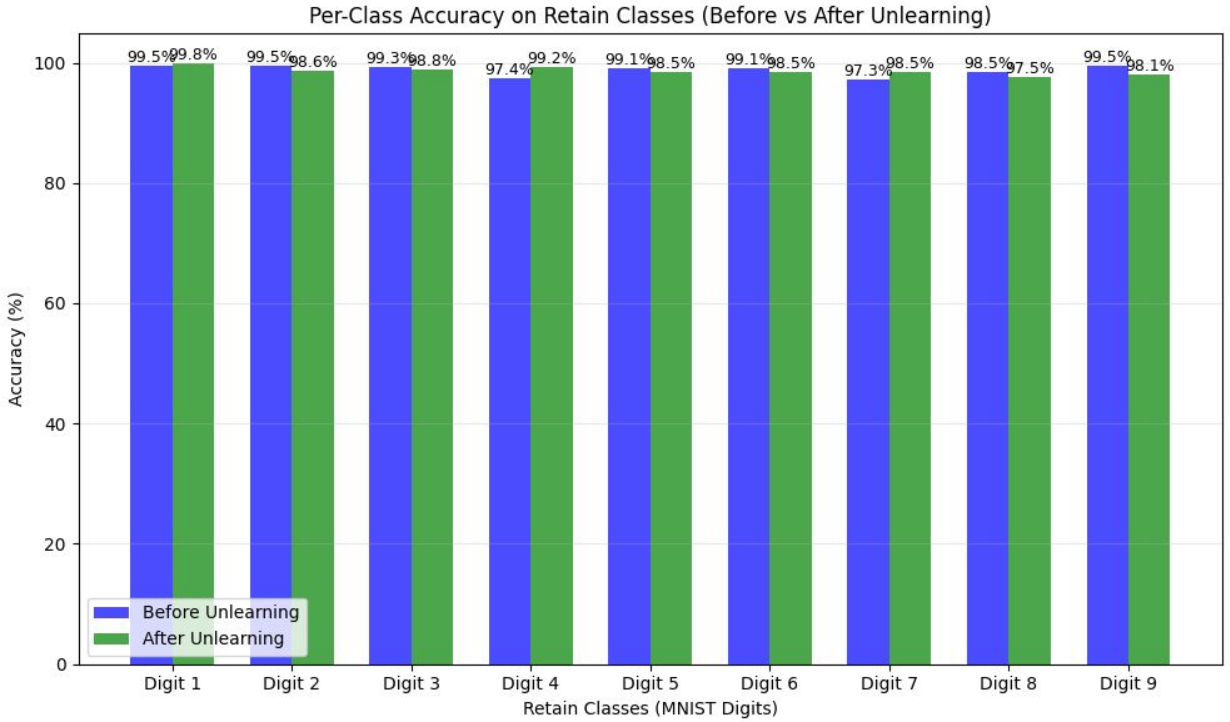
Forget Class=[0]

Retain Class=[1,2,3,4,5,6,7,8,9],

Fine-tuning after each decay step with 21 batches

Post-unlearning + Fine-tuning Forget Class Accuracy: 2.35%

Post-unlearning + Fine-tuning Retain Classes Accuracy: 98.66%

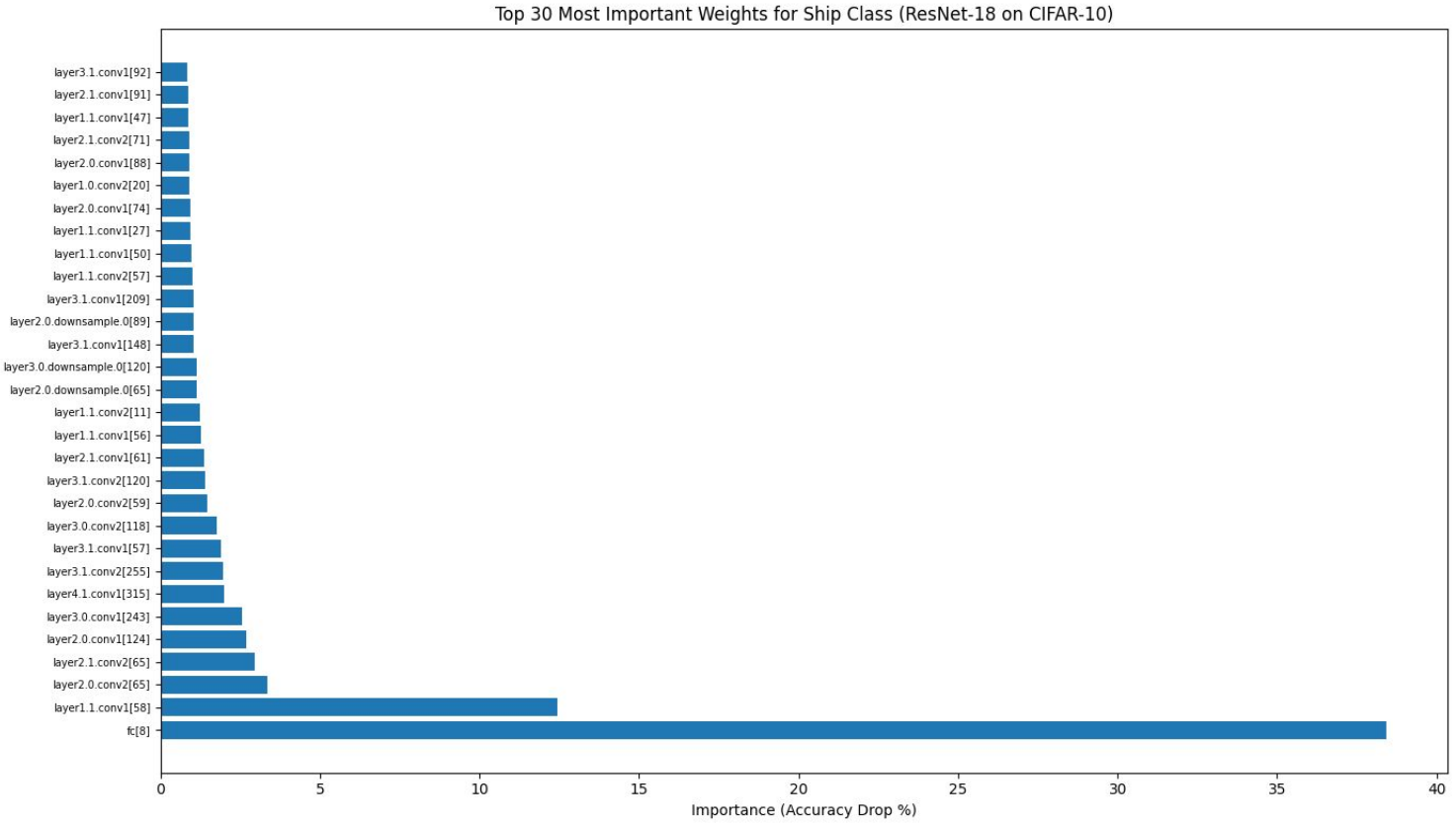


Takeaways till now

1. Hypothesis is correct.
2. Time-decay of important parameters for a class helps to forget the class
3. Choosing random parameters instead of important parameters for decay affects accuracies of other classes.
4. Fine-tuning after all decay steps is more effective - (Doesn't hold in ResNet-18)

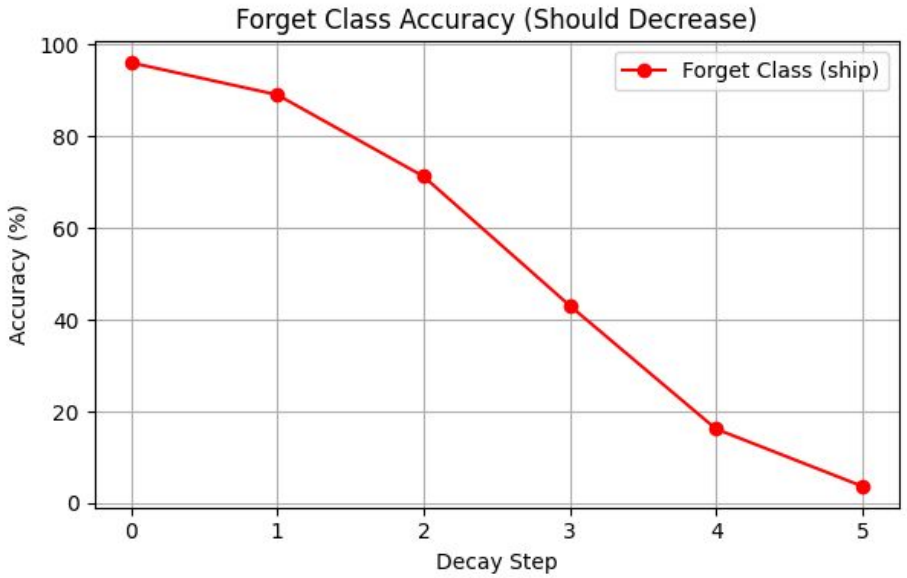
Resnet-18 on CIFAR-10

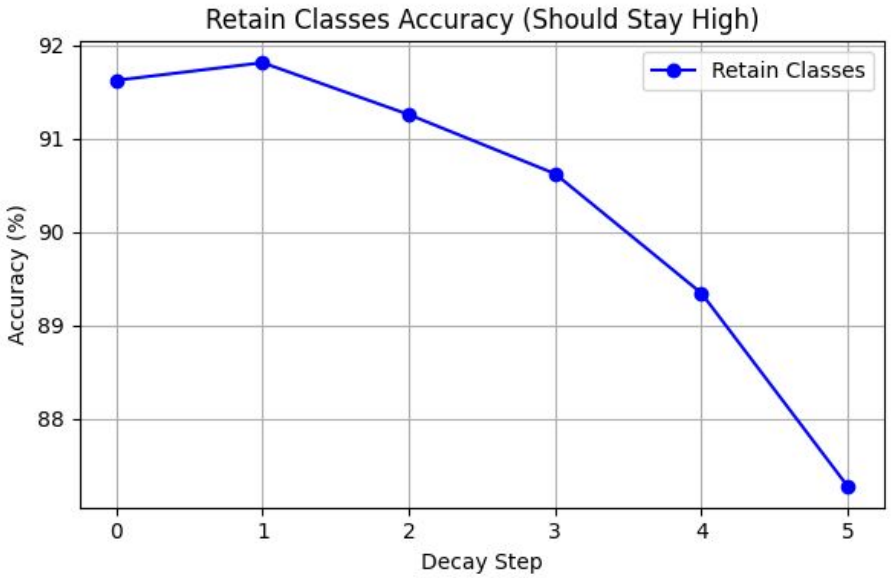
Pre-Unlearning Forget Class (ship) Accuracy: 96.10%
Pre-Unlearning Retain Classes Accuracy: 91.62%
Pre-Unlearning Overall Test Accuracy: 92.07%
Training Epochs: 20



Resnet-18 on CIFAR-10

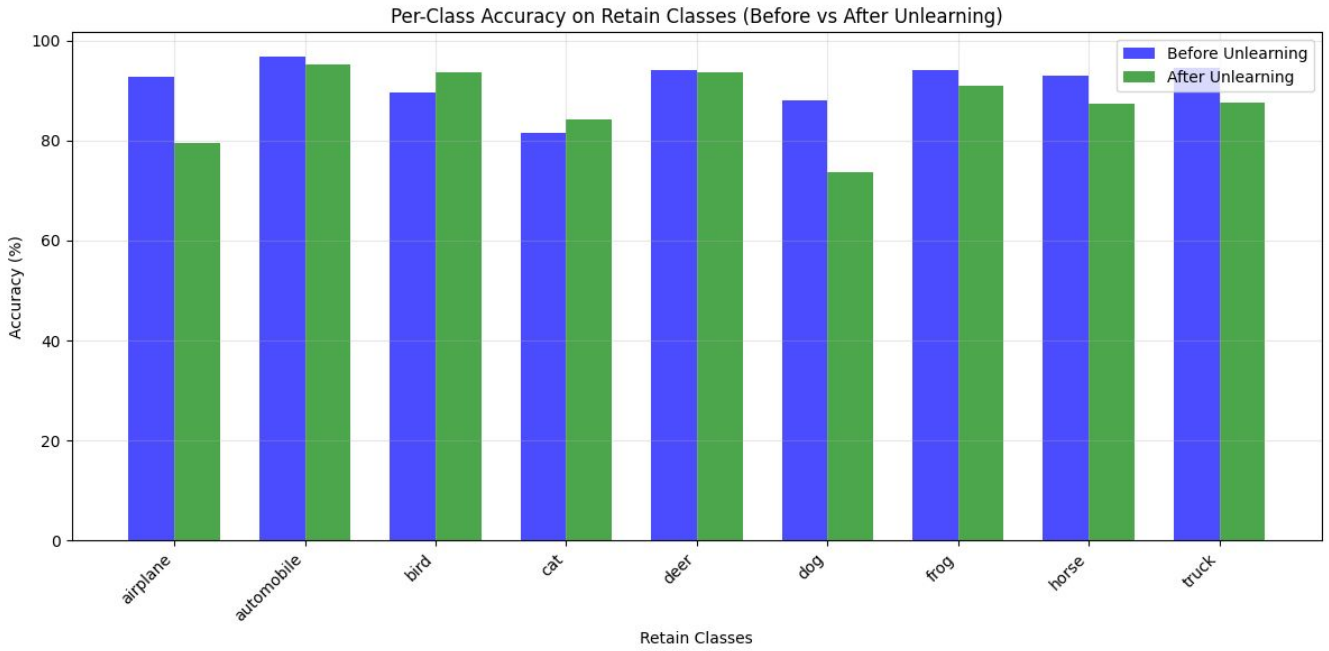
Decay rate= Lambda = 0.05
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning= False
Post-unlearning Forget Class (ship) Accuracy: 3.70%
Post-unlearning Retain Classes Accuracy: 87.28%





Resnet-18 on CIFAR-10

Decay rate= Lambda = 0.05
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning= False
Post-unlearning Forget Class (ship) Accuracy: 3.70%
Post-unlearning Retain Classes Accuracy: 87.28%



Resnet-18 on CIFAR-10

Decay rate= Lambda = 0.05
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning after all 5 decay steps with 1 epoch of training retain dataset (5%)
Post-unlearning+Fine Tuning Forget Class (ship) Accuracy: **63.20%**
Post-unlearning+ Fine Tuning Retain Classes Accuracy:91.54%

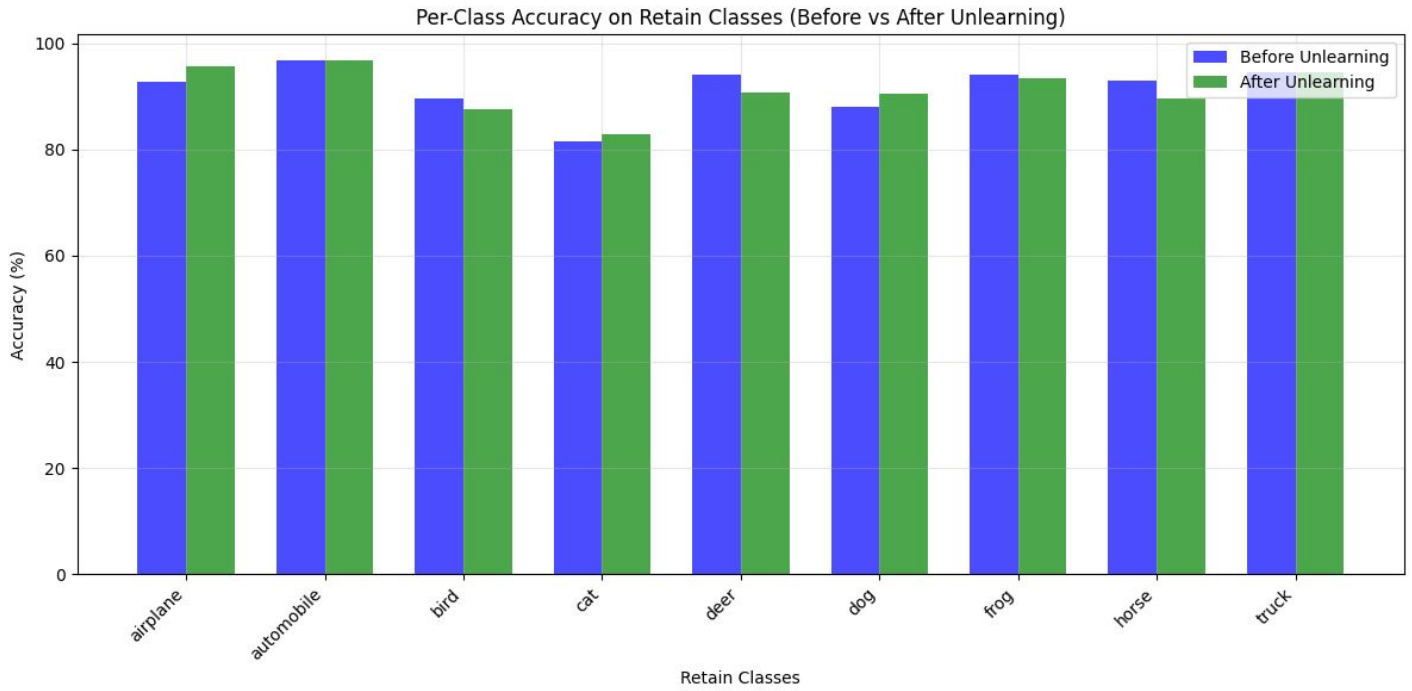
Forget Class Accuracy increased after fine-tuning

Resnet-18 on CIFAR-10

Decay rate= Lambda = 0.05
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning after all 5 decay steps with 2 epoch of training retain dataset (10%)
Post-unlearning+Fine Tuning Forget Class (ship) Accuracy: 51.20%
Post-unlearning+ Fine Tuning Retain Classes Accuracy:91.22%
Forget class accuracy increased with 1 and decreased with 2 epoch fine-tuning

Resnet-18 on CIFAR-10

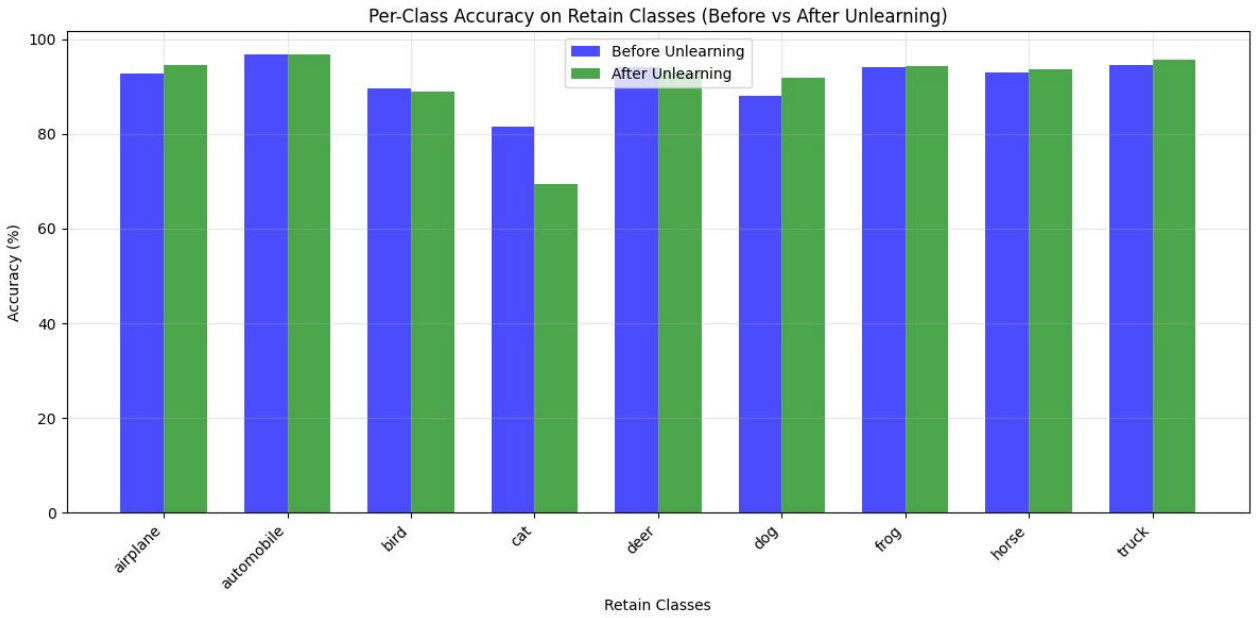
Decay rate= Lambda = 0.05
Fine-tuning after all 5 decay steps with 6 epochs of training retain dataset (30%)
Post-unlearning+Fine Tuning Forget Class (ship) Accuracy: 6.40%
Post-unlearning+ Fine Tuning Retain Classes Accuracy:91.28%
Fine-tuning more after decay caused Forget-class accuracy to drop



So is it enough to just retrain with additional epochs of retain set to forget the “Ship” class WITHOUT TIME DECAY?

Resnet-18 on CIFAR-10

Decay Steps= 0
Fine-tuning with 6 epochs of training retain dataset (30%)
Post-unlearning+Fine Tuning Forget Class (ship) Accuracy: 12.60% (**>5%)**
Post-unlearning+ Fine Tuning Retain Classes Accuracy:90.92%



Resnet-18 on CIFAR-10

Why even perform decay then? Why not just train/fine-tune with training retain dataset

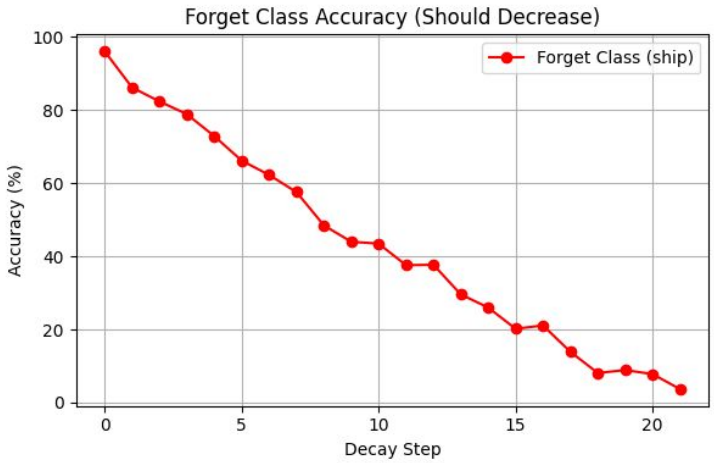
	Decay with No Fine-tuning	No Decay with Fine-tuning
Retain Test Accuracy	87.28%	91.28%
Forget Test Accuracy	3.70%	6.40%

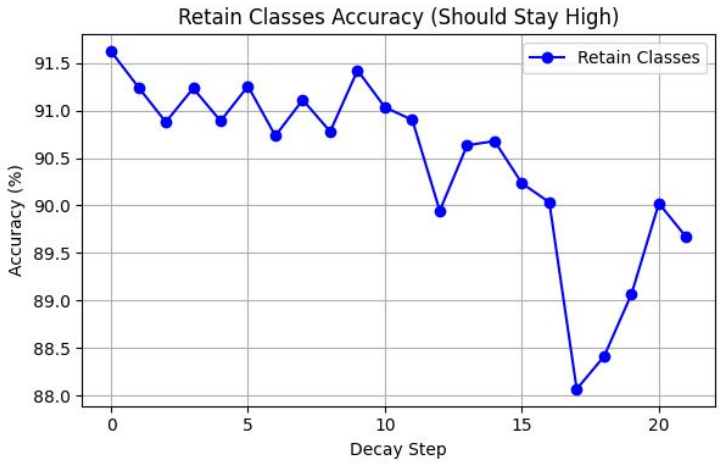
But….
Fine-tuning without decay on Retain Class requires 6 Epochs.
Fine-tuning with decay **AFTER EACH DECAY STEP** requires less than 1 Epoch.

Resnet-18 on CIFAR-10

Decay rate= Lambda = 0.1
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning after each decay step on 11 batches
Post-Learning Forget Class (ship) Accuracy: 3.70% **(<5% forget accuracy)**
Post-Learning Retain Classes Accuracy: 89.67%

Retain Batches per Epoch=352
Total Batches used in fine-tuning = 21X11 = 231 < 1 EPOCH

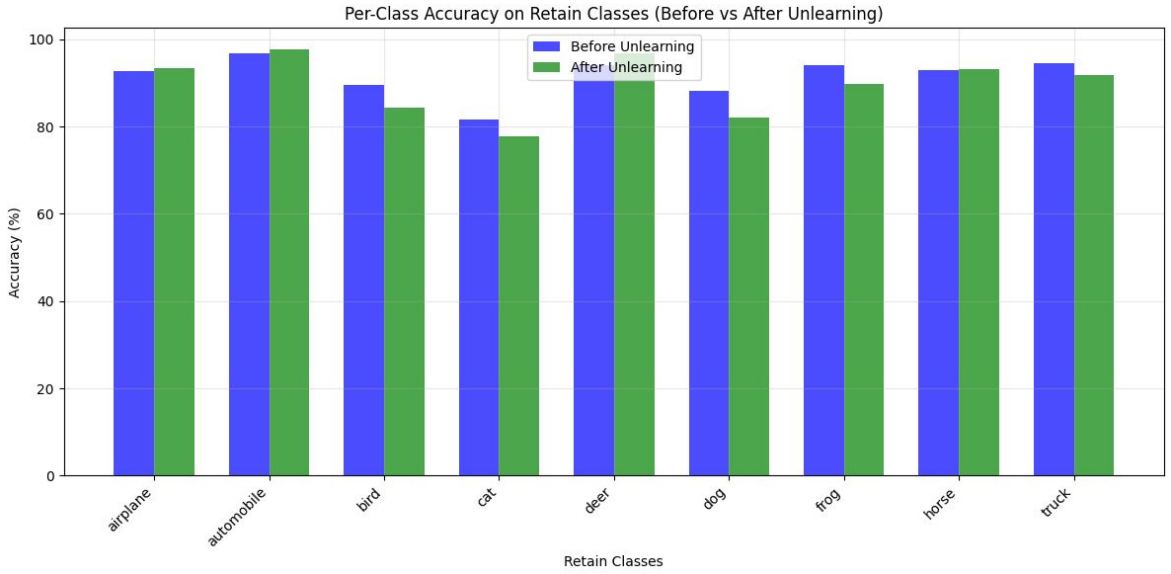




Resnet-18 on CIFAR-10

Decay rate= Lambda = 0.1
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning after each decay step on 11 batches
Post-Learning Forget Class (ship) Accuracy: 3.70%
Post-Learning Retain Classes Accuracy: 89.67%

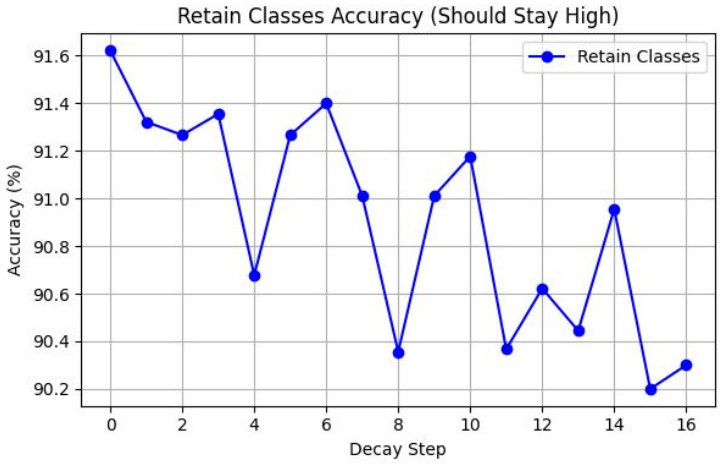
Retain Batches per Epoch=352
Total Batches used in fine-tuning = 21X11 = 231 < 1 EPOCH



Resnet-18 on CIFAR-10

Decay rate= Lambda = 0.1
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning after each decay step on 17 batches
Post-Learning Forget Class (ship) Accuracy: 4.40%
Post-Learning Retain Classes Accuracy: 90.30% (**increased with 17 batches instead of 11)**

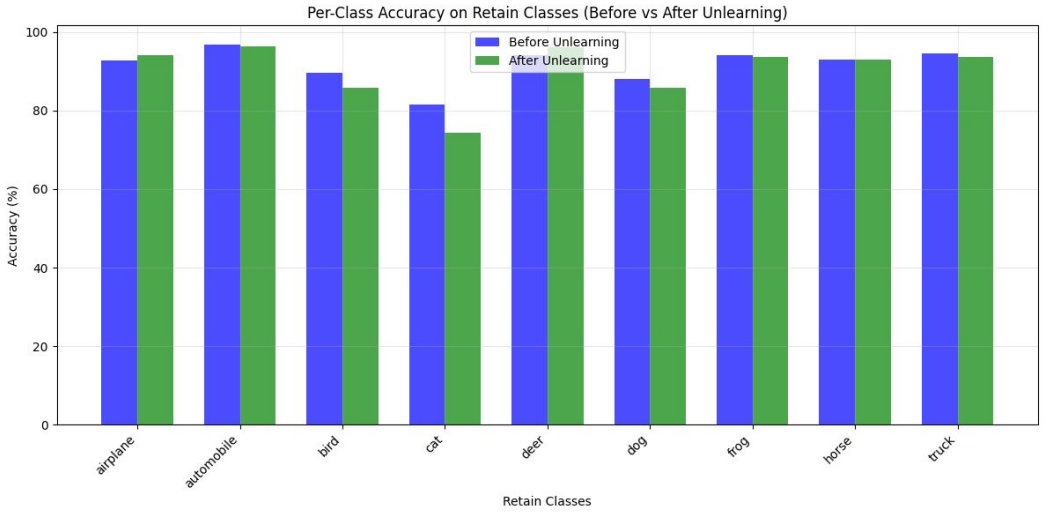
Retain Batches per Epoch=352
Total Batches used in fine-tuning = 17X16 = 272 batches < 1 EPOCH



Resnet-18 on CIFAR-10

Decay rate= Lambda = 0.1
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Fine-tuning after each decay step on 17 batches
Post-Learning Forget Class (ship) Accuracy: 4.40%
Post-Learning Retain Classes Accuracy: 90.30%

Retain Batches per Epoch=352
Total Batches used in fine-tuning = 21X11 = 231 < 1 EPOCH



Resnet-18 on CIFAR-10

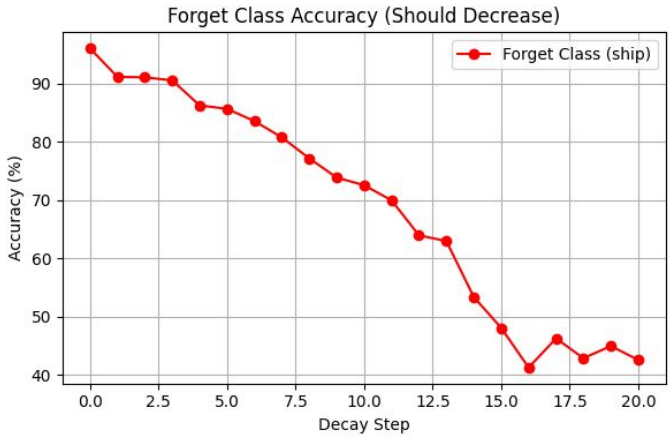
Fine-tuning after every decay step is more efficient for unlearning than just fine-tuning on retain dataset without decay.

Why don't we decay random weights and then fine-tune on Retain Data? That way there is no need for computational cost to calculate importance of each parameter for a class.

DOESNT WORK

Resnet-18 on CIFAR-10

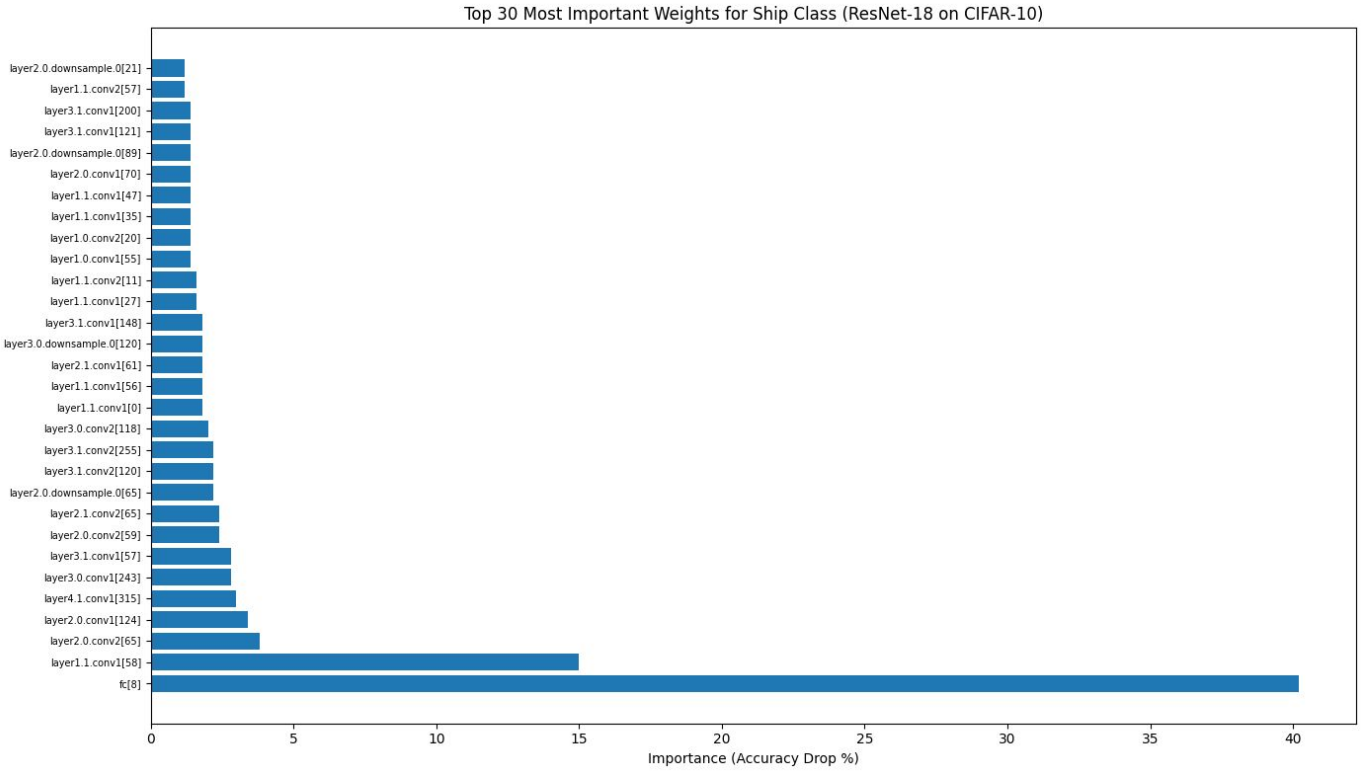
Decay rate= Lambda = 0.1
Forget Class=[‘ship’]
Retain Class=['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
Decay on random parameters instead of top-k important parameters
Fine-tuning after each decay step on 17 batches
Forget Class (ship) Accuracy: 42.60% (**very high)**
Retain Classes Accuracy: 90.66%



IMPORTANCE ESTIMATION BOTTLENECK

Estimating importance score of each parameter for a class is expensive

Before: Importance of each parameter computed with whole forget_train_data
Now: Importance of each parameter computed with 10% of forget_train_data



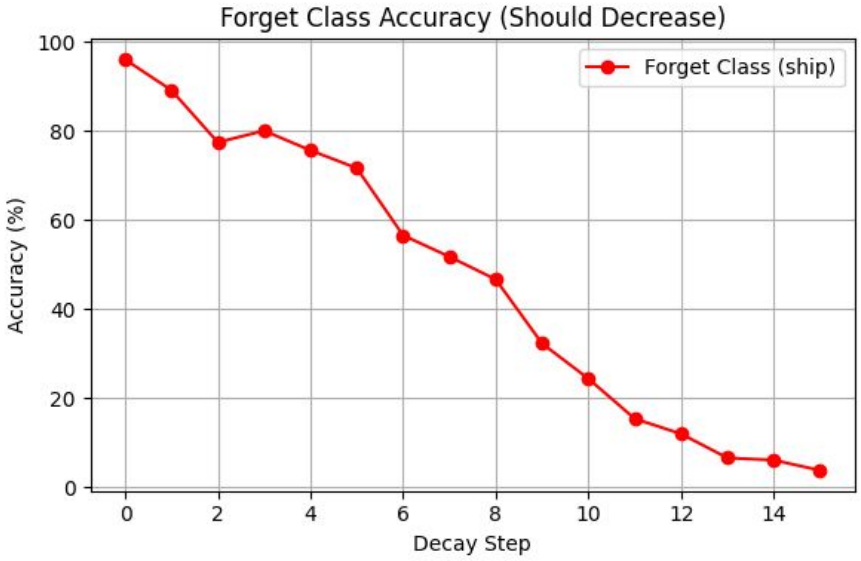
IMPORTANCE ESTIMATION BOTTLENECK

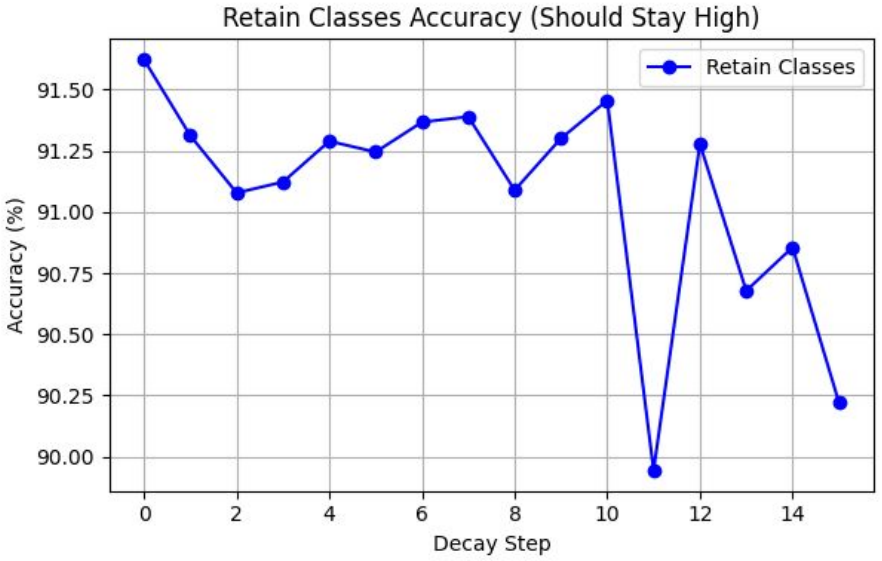
Estimating importance score of each parameter for a class is expensive

Before: Importance of each parameter computed with whole forget_train_data
Now: Importance of each parameter computed with 10% of forget_train_data

Forget Class (ship) Accuracy: 3.80%
Retain Classes Accuracy: 90.22%

Fine-tuning Batch count: 345 (<1 EPOCH)



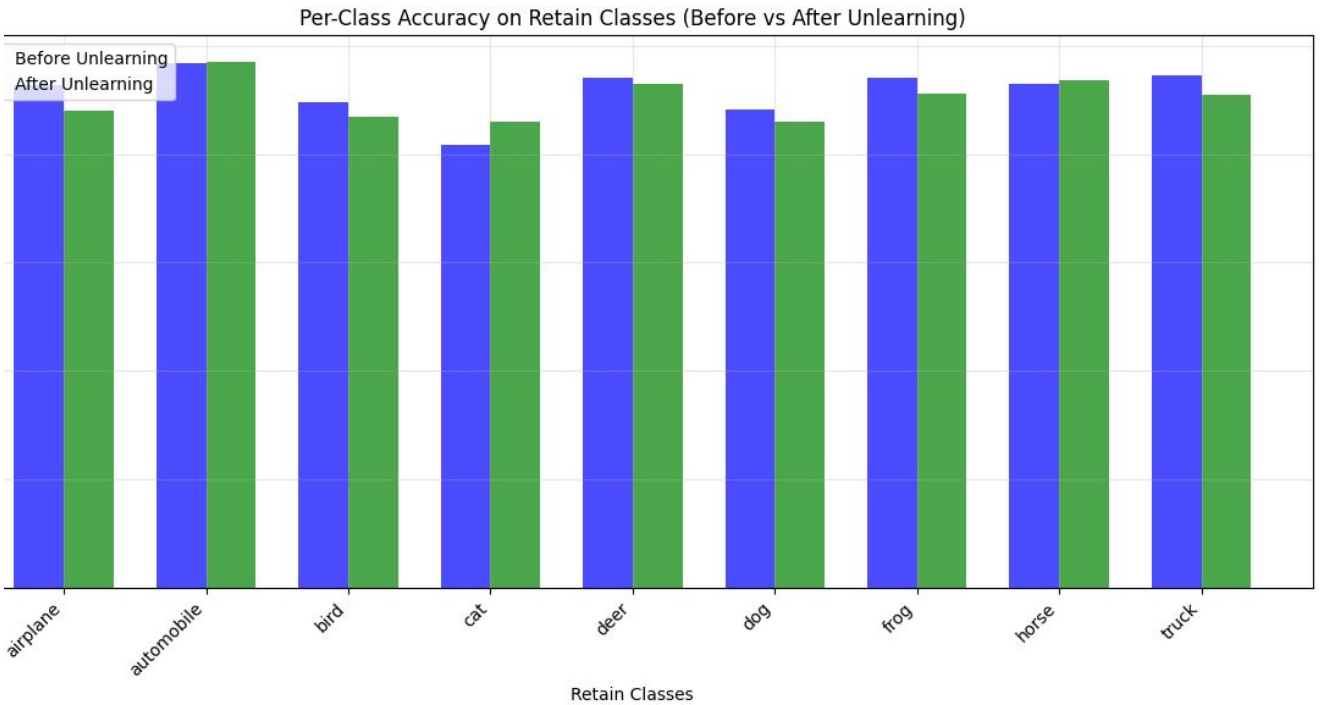


IMPORTANCE ESTIMATION BOTTLENECK

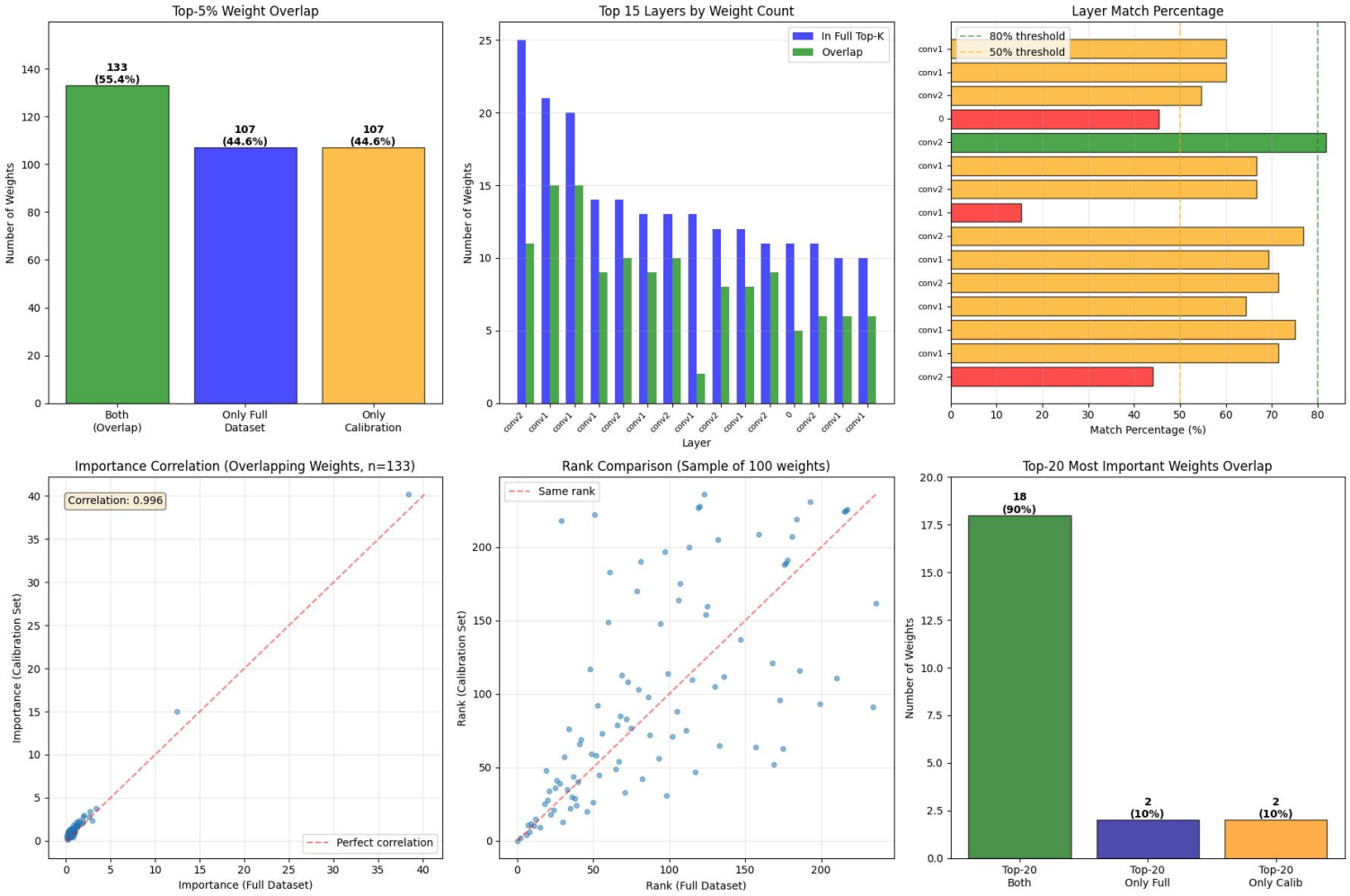
Estimating importance score of each parameter for a class is expensive

Before: Importance of each parameter computed with whole forget_train_data
Now: Importance of each parameter computed with 10% of forget_train_data

Forget Class (ship) Accuracy: 3.80%
Retain Classes Accuracy: 90.22%
Fine-tuning Batch count: 345 (<1 EPOCH)



IMPORTANCE ESTIMATION BOTTLENECK



BIGGEST BOTTLENECK

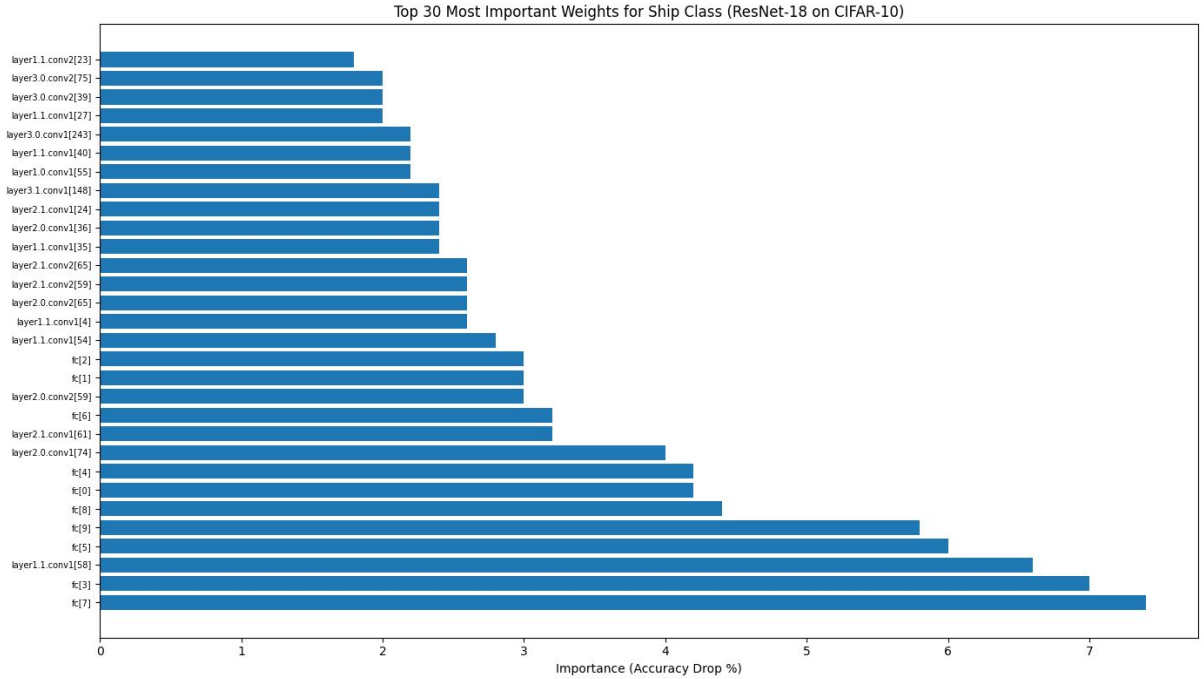
Before: Importance of each parameter computed with whole forget_train_data
Now: Importance of each parameter computed with 10% of forget_train_data

What if we identify the top parameters with all class train data rather than forget_train_data ?

IMPORTANCE ESTIMATION BOTTLENECK

Estimating importance score of each parameter for a class is expensive
Before: Importance of each parameter computed with whole forget_train_data
Previous: Importance of each parameter computed with 10% of forget_train_data
Now: Importance of each parameter computed with 10% of all train data

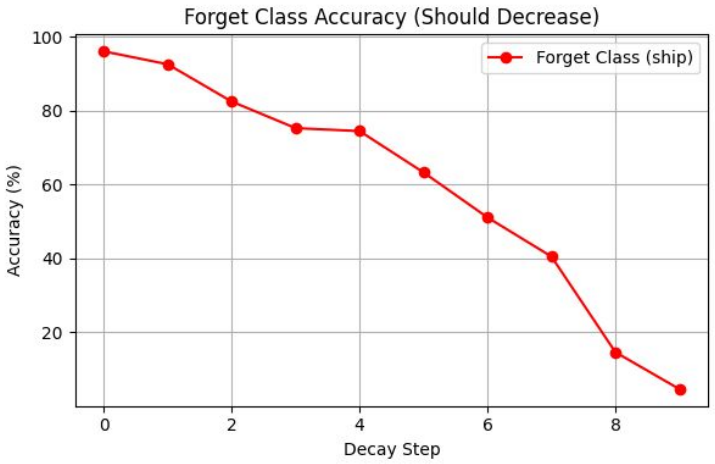
Forget Class (ship) Accuracy: 4.60%
Retain Classes Accuracy: 90.61%
Fine-tuning Batch count: 345 (<1 EPOCH)

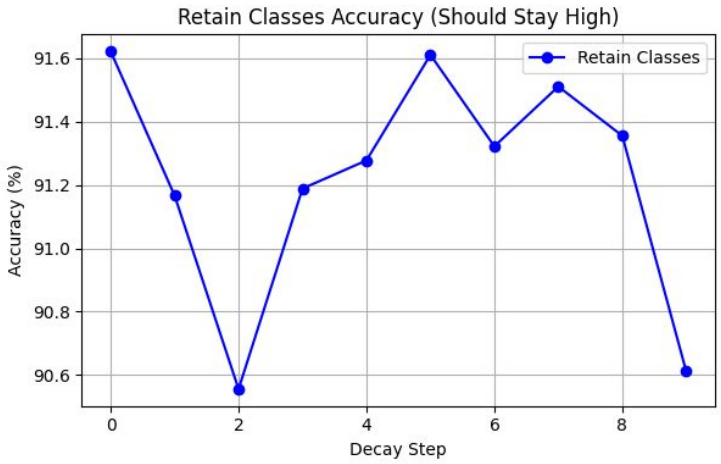


IMPORTANCE ESTIMATION BOTTLENECK

Estimating importance score of each parameter for a class is expensive
Before: Importance of each parameter computed with whole forget_train_data
Previous: Importance of each parameter computed with 10% of forget_train_data
Now: Importance of each parameter computed with 10% of all train data

Forget Class (ship) Accuracy: 4.60%
Retain Classes Accuracy: 90.61%
Fine-tuning Batch count: 345 (<1 EPOCH)

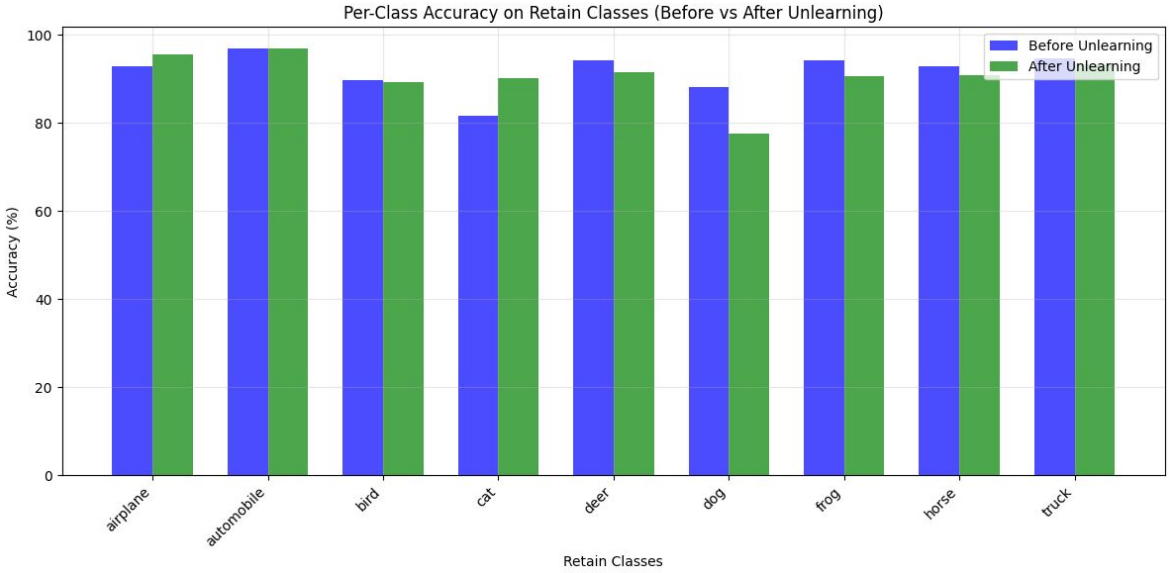




IMPORTANCE ESTIMATION BOTTLENECK

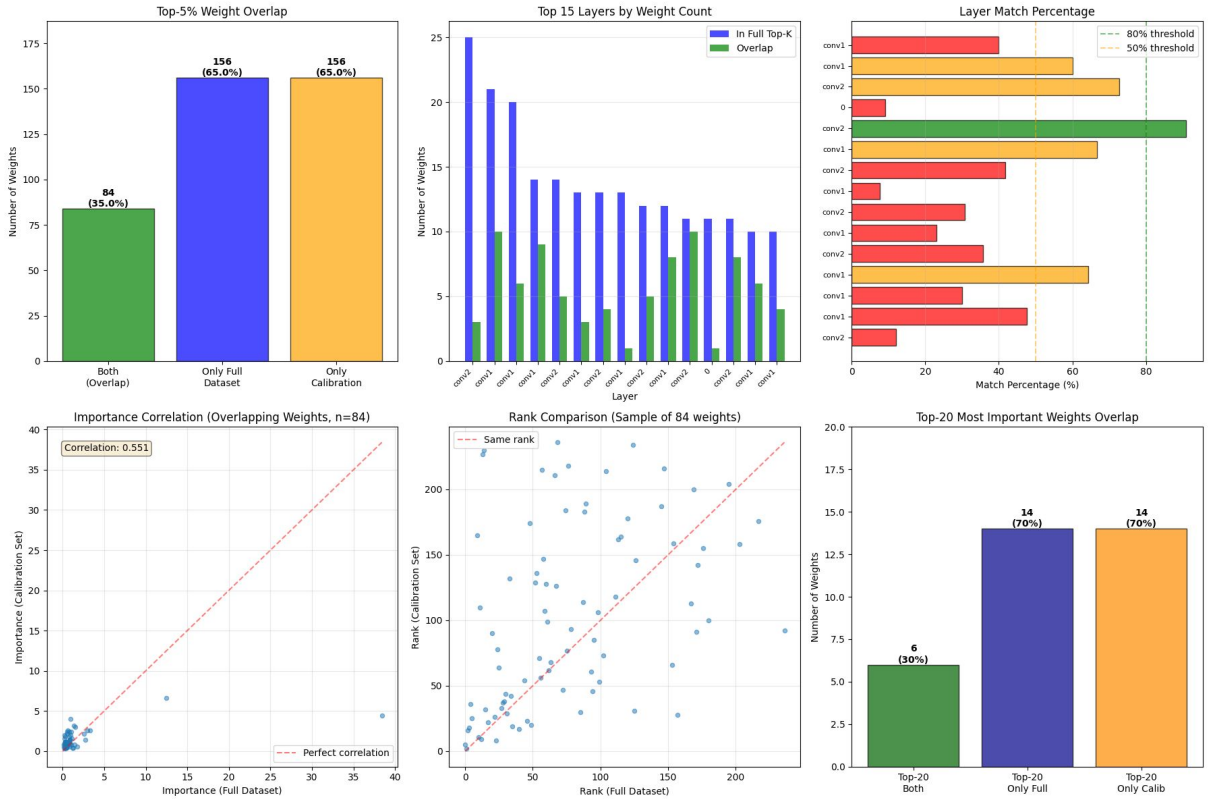
Estimating importance score of each parameter for a class is expensive
Before: Importance of each parameter computed with whole forget_train_data
Previous: Importance of each parameter computed with 10% of forget_train_data
Now: Importance of each parameter computed with 10% of all train data

Forget Class (ship) Accuracy: 4.60%
Retain Classes Accuracy: 90.61%
Fine-tuning Batch count: 345 (<1 EPOCH)



IMPORTANCE ESTIMATION BOTTLENECK

Estimating importance score of each parameter for a class is expensive
Before: Importance of each parameter computed with whole forget_train_data
Previous: Importance of each parameter computed with 10% of forget_train_data
Now: Importance of each parameter computed with 10% of all train data



FUTURE SCOPE

1. Test with larger datasets on larger models
2. Test with different combination of hyper-parameters.
3. Coming up with a more efficient way to determine the importance of parameters during inference for a particular class.