## A Measuring Diffused Redundancy

### A.1 CKA Definition

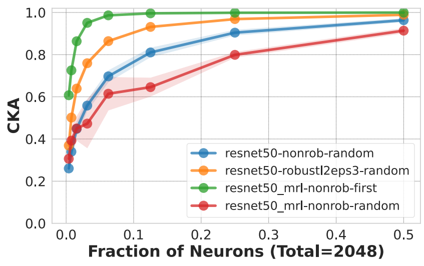
In all our evaluations we use CKA with a linear kernel [24] which essentially amounts to the following steps:

1. Take two representations $Y \in \mathbb{R}^{n \times d1}$ and $Z \in \mathbb{R}^{n \times d2}$
2. Compute dot product similarity within these representation, *i.e.*compute $K = YY^T$, $L = ZZ^T$
3. Normalize $K$ and $L$ to get $K' = HKH$, $L' = HLH$ where $H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$
4. Return $\text{CKA}(Y, Z) = \frac{\text{HSIC}(K,L)}{\sqrt{\text{HSIC}(K,K)\text{HSIC}(L,L)}}$, where $\text{HSIC}(K, L) = \frac{1}{(n-1)^2}(\text{flatten}(K') \cdot \text{flatten}(L'))$

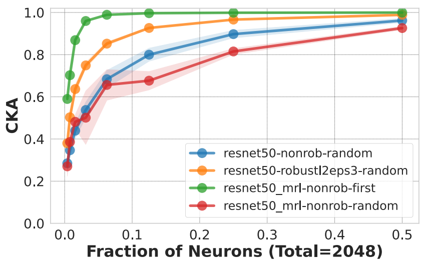
We use the publicly available implementation of [37], which provides an implementation that can be calcuated over multiple mini-batches: `https://github.com/nvedant07/STIR`

### A.2 Additional CKA results

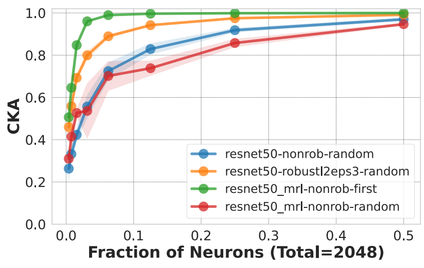
Fig 9 shows CKA comparison between randomly chosen parts of the layer and the full layer for different kinds of ResNet50. We observe that even ResNet50 trained with MRL loss shows a significant amount of diffused redundancy.



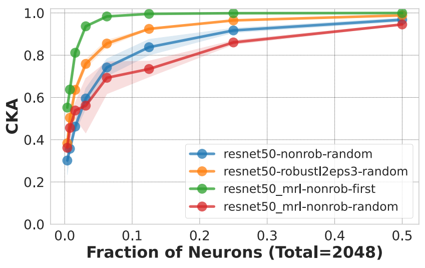
(a) CIFAR10



(b) CIFAR100



(c) Flowers



(d) Oxford-IIIT-Pets

Figure 9: **[Comparison of Diffused Redundancy in MRL vs other losses, through the lens of CKA]** We see a similar trend as reported in Fig 7 in the main paper, where even the MRL model shows a significant amount of diffused redundancy despite being explicitly trained to instead have structured redundancy. The amount of diffused redundancy however is much lesser than the resnets trained using the standard loss and adv. training as denoted by a much lower red line across all datasets.

## B Training and Pre-Processing Details for Reproducibility

Here we list the sources of weights for the various pre-trained models used in our experiments:

- ResNet18 trained on ImageNet1k using standard loss: taken from `timm` v0.6.1.
- ResNet18 trained on ImageNet1k with adv training: taken from Salman et al. [46]:

- ResNet50 trained on ImageNet1k using standard loss: taken from `timm` v0.6.1.
- ResNet50 trained on ImageNet1k with adv training: taken from Salman et al. [46]: `https://github.com/microsoft/robust-models-transfer`.
- ResNet50 trained on ImageNet1k using MRL and with different final layer widths (`resnet50_ffx`): taken from released weights of by Kusupati et al. [26]: `https://github.com/RAIVNLab/MRL`.
- WideResNet50-2 on ImageNet1k both standard and avd. training: taken from Salman et al. [46]: `https://github.com/microsoft/robust-models-transfer`.
- VGG16 trained on ImageNet1k with standard loss: taken from `timm` v0.6.1.
- VGG16 trained on ImageNet1k with adv training: taken from Salman et al. [46]: `https://github.com/microsoft/robust-models-transfer`.
- ViTS32 & ViTS16 trained on ImageNet21k & ImageNet1k: taken from weights released by Steiner et al. [53]: `https://github.com/google-research/vision_transformer`.

All linear probes trained on the representations of these models are trained using SGD with a learning rate of $0.1$, momentum of $0.9$, batch size of $256$, weight decay of $1e-4$. The probe is trained for $50$ epochs with a learning rate scheduler that decays the learning rate by $0.1$ every $10$ epochs. Scripts for training can also be found in the attached code.

For pre-processing, we re-size all inputs to 224x224 (size used for pre-training) and apply the usual composition of RandomHorizontalFlip, ColorJitter(brightness=0.25, contrast=0.25, saturation=0.25, hue=0.25), RandomRotation(degrees=2). All inputs were mean normalized. For imagenet1k pre-trained models: mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. For imagenet21k pre-trained models: mean = [0.5,0.5,0.5], std = [0.5,0.5,0.5].

## C Deeper Analysis of Fairness-Efficiency Tradeoff in Section 4

**Analyzing Error Distributions** To ensure that the higher gini coefficient shown in Fig 8 as we drop more neurons is not merely an artifact of lower overall accuracy, we plot class-wise accuracies as we drop neurons (Figs 11, 12 & 13). We find that for the entire layer, accuracy starts at an almost uniform distribution, and while overall accuracy deteriorates as we drop neurons, the drop comes at a larger cost for a few classes resulting in disparate inter-class accuracies.

**Coeff of Variation for Measuring Inequality in Inter-Class Accuracy** Fig 10 shows results for the same analysis shown in Fig 8 of the main paper and we find similar takeaways even when using the coefficient of variation as a measure of inequality.

## D Corresponding Diffused Redundancy Estimates For Analyses in Section 3 & $\ell_\infty$ Robust Model Results

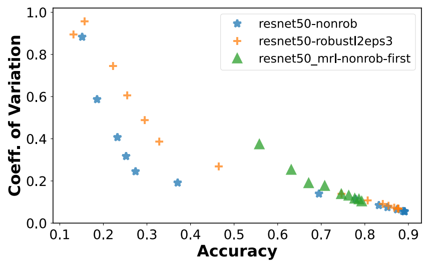
Corresponding diffused redundancy (DR) ablations for Figures 4,5,7. These are shown in Figures 15,16,17 respectively. This should allow for easy comparison of diffused redundancy (lines that are more outside have higher DR). For example, Figure 16 clearly shows higher diffused redundancy in models trained on larger upstream datasets (here ImageNet21k) since these curves lie more on the outside of the same model's curves for ImageNet1k.

Additionally, we show numbers on x-axis for Figure 4 in Figure 18. Figures 5 and 7 compare models with same number of neurons in the final layer and hence trends shown with fraction on the x-axis will be exactly the same with absolute numbers on the x-axis. However, Figure 18 allows a direct comparison of the performance of the same absolute number of neurons across different models.

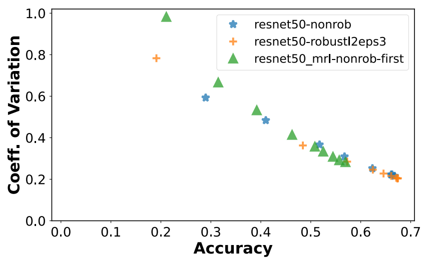
We report results for $\ell_\infty$ robust models (with $\epsilon = 4/255$) in Fig 14 and find that $\ell_2$ model generally shows a greater degree of diffused redundancy.

## E Results on Intermediate Layers

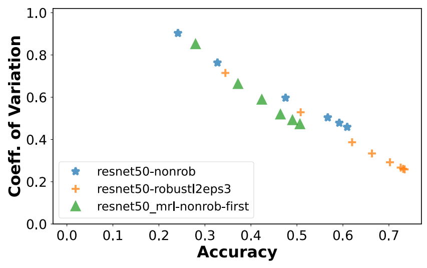
We additionally ran our experiment on other intermediate layers and report the results in Fig 19. We present results for a ResNet50 pretrained on ImageNet1k using the standard CrossEntropy loss. The



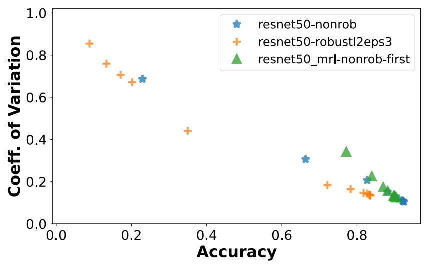
(a) CIFAR10



(b) CIFAR100



(c) Flowers



(d) Oxford-IIIT-Pets

Figure 10: **[Coefficient of Variation As We Drop Neurons]** We see a similar trend as reported in Fig 8 of the main paper where inequality increases as we drop neurons for all models on all datasets.

intermediate layers considered are characterized as activations following each residual connection within distinct ResNet blocks. `layerX.Y.act3` means the $Yth$ residual connection in the $Xth$ ResNet block and act3 indicates that we're taking the value after the activation (ReLU) has been applied.

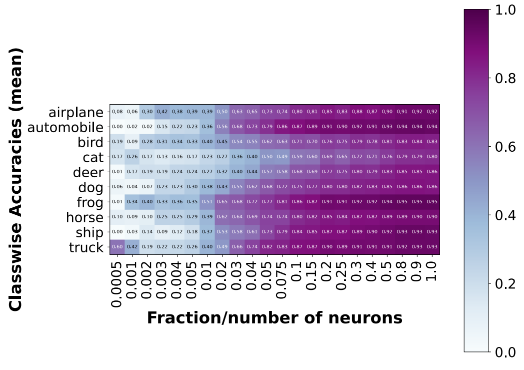
## F Results on Harder Downstream Tasks: ImageNetV2 and Places365

We report results on harder downstream tasks such as ImageNet1k, ImageNetV2, and Places365 in Figure 20. We find that when randomly dropping neurons, the model is still able to generalize to ImageNet1k and Places365 with very few neurons, *i.e.*, the phenomena of diffused redundancy observed for smaller datasets, also holds for harder datasets. Interestingly we also observe that the accuracy gap between ImageNet1k and ImageNetV2 is maintained even as we drop neurons.

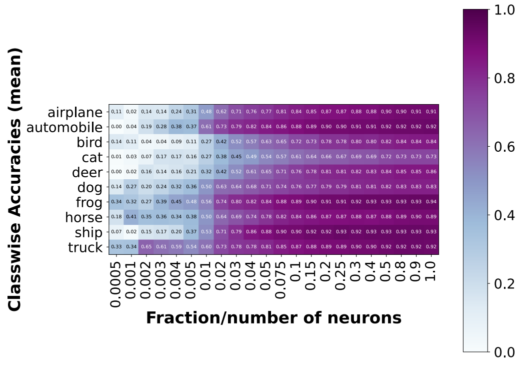
## G Effects of Explicitly Preventing Co-adaptation of Neurons: Analysis of Dropout and DeCov

Regularizers such as dropout and DeCov, force different parts of the representations to not be correlated. Thus these regularizers can be seen as explicitly requiring different, compact parts of the representation to be self-contained for the downstream task. Thus, intuitively, such methods should increase diffused redundancy. Here we investigate if our observation about diffused redundancy is influenced by such regularizers. We evaluate ResNet50 pre-trained on ImageNet1k with dropout in the penultimate layer ranging from a strength of 0.1 all the way to 0.8. We also train another ResNet50 model with the DeCov regularizer added to the usual crossentropy loss and put a weight of 0.0001 on the regularizer to ensure that its numerical range is similar to that of the cross entropy loss term.

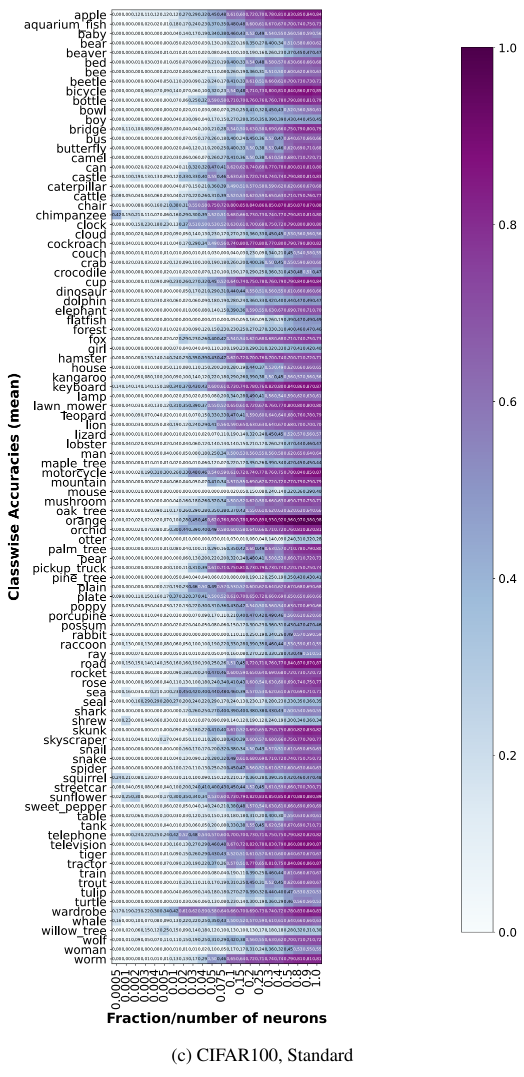
Results in Figures 21 & 22 suggest that such regularizers have *almost no effect* on diffused redundancy. Trends across datasets remain consistent regardless of the strength of dropout or the weight given to DeCov regularizer. This observation further adds to the evidence that diffused redundancy is likely to be a natural property of representations learned by DNNs.



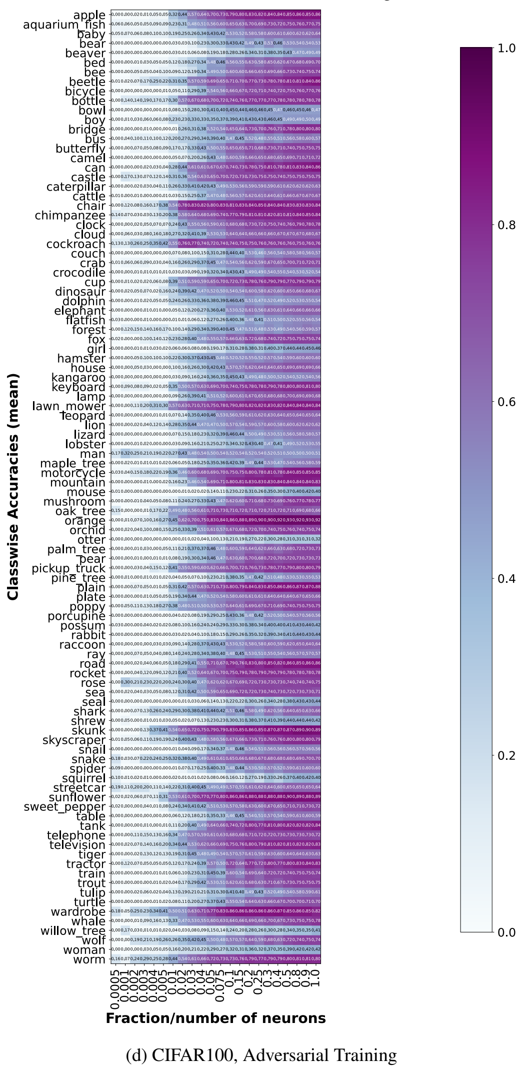
(a) CIFAR10, Standard



(b) CIFAR10, Adversarial Training

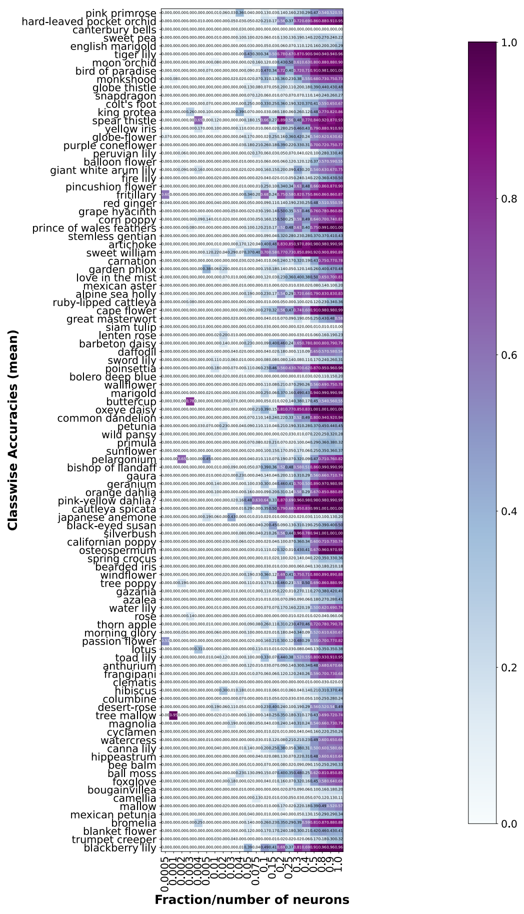


(c) CIFAR100, Standard



(d) CIFAR100, Adversarial Training

Figure 11: **[Error Distributions As A Function of Fraction of Neurons]** We see that accuracy deteriorates as we drop neurons, however, this drop comes at a larger cost for a few classes and results in near homogenous predictions for the least number of neurons on the left.


(a) Flowers, Standard

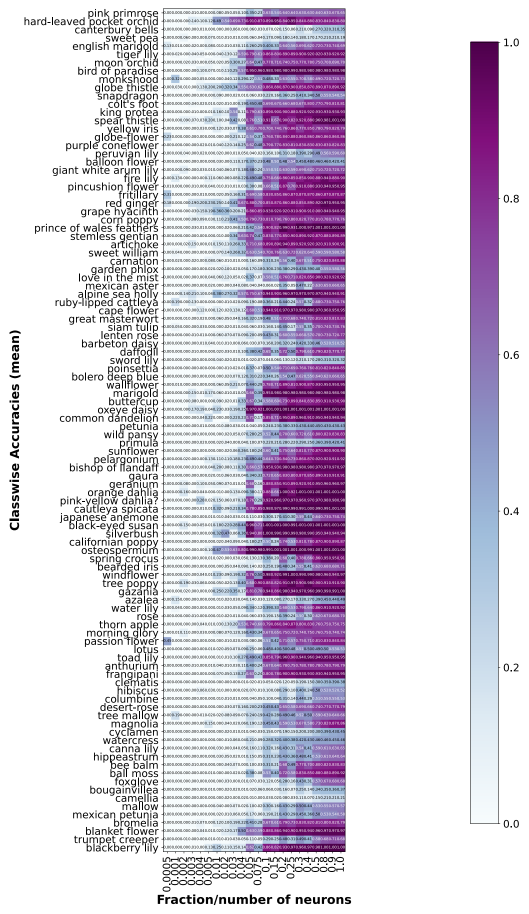
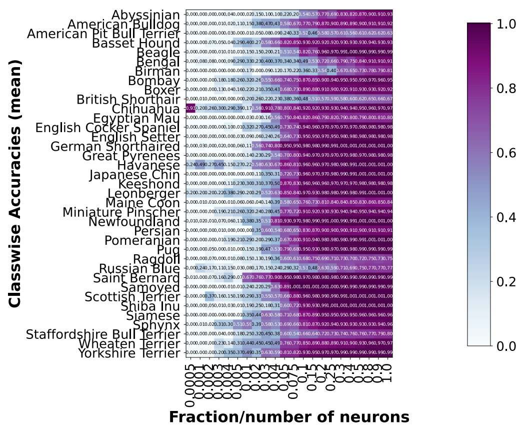

(b) Flowers, Adversarial Training

Figure 12: **[Error Distributions As A Function of Fraction of Neurons – continued]** We see that accuracy deteriorates as we drop neurons, however, this drop comes at a larger cost for a few classes and results in near homogenous predictions for the least number of neurons on the left.


(a) Oxford-IIIT-Pets, Standard

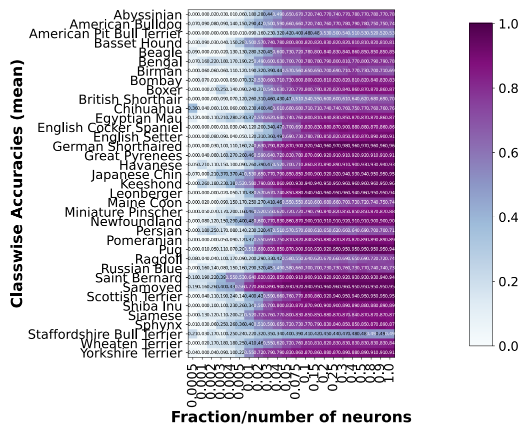

(b) Oxford-IIIT-Pets, Adversarial Training

Figure 13: **[Error Distributions As A Function of Fraction of Neurons – continued]** We see that accuracy deteriorates as we drop neurons, however, this drop comes at a larger cost for a few classes and results in near homogenous predictions for the least number of neurons on the left.

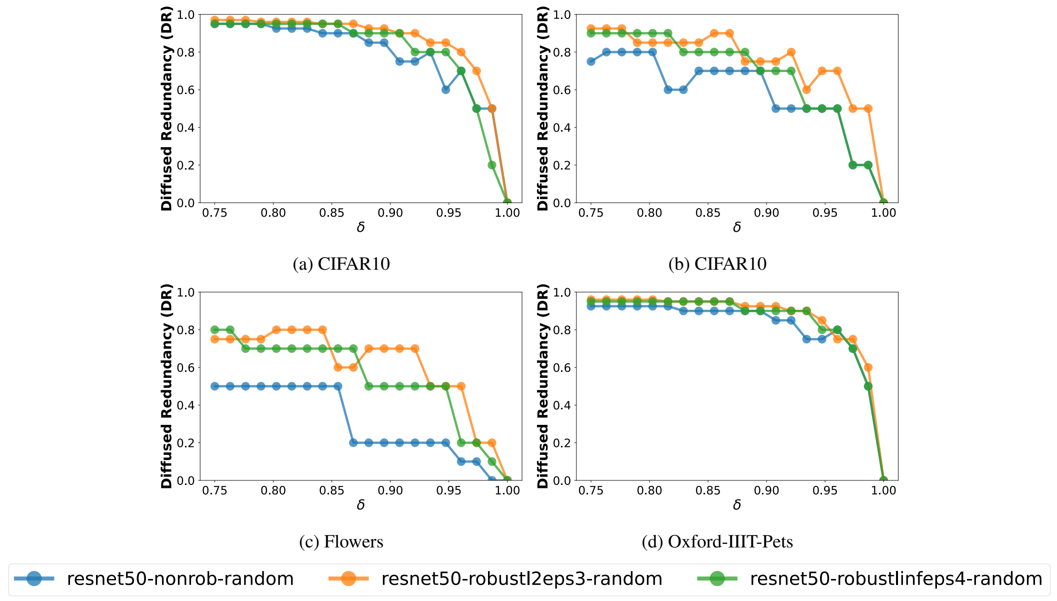


Figure 14: **[Results for $\ell_\infty$ robust model]** We show results for ResNet50 trained with 3 different losses: CrossEntropy (nonrob), Adversarial Training with $\ell_2$ threat model (robl2eps3), and with the $\ell_\infty$ threat model (roblinfeps4). We see that the $\ell_2$ threat model shows the most diffused redundancy. All models are trained on ImageNet1k.

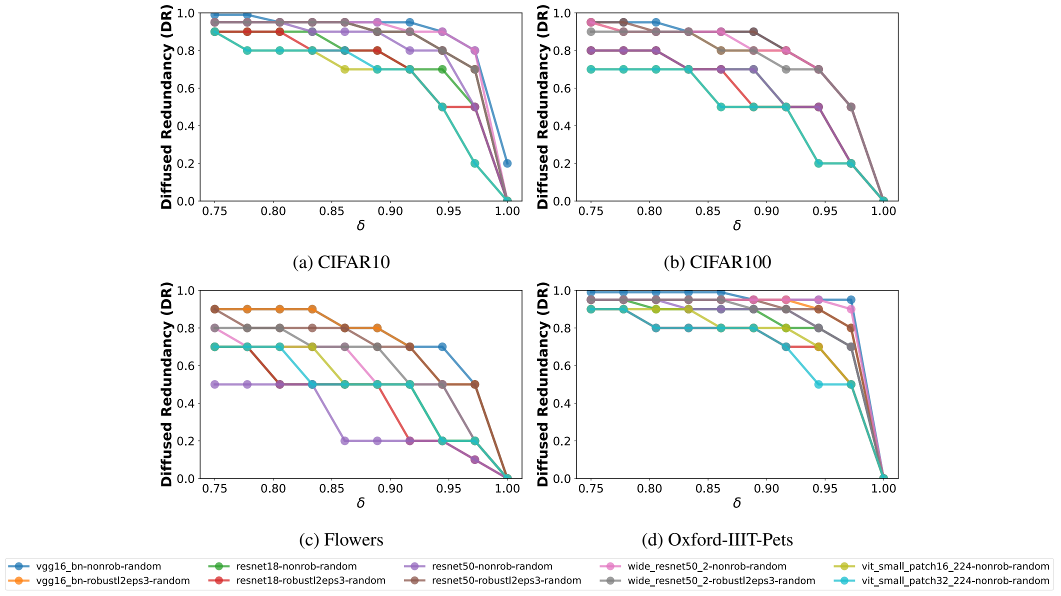


Figure 15: **[Comparisons Across Architectures For Downstream Task Accuracy]** All models shown here are pre-trained on ImageNet1k. This Figure shows corresponding diffused redundancy values for Figure 4 different $\delta$ values. We see that diffused redundancy exists across architectures, and the trend observed in Figure 1c&1a regarding adversarially trained models also holds here as models curves that are more "inside" are the ones trained with standard loss.

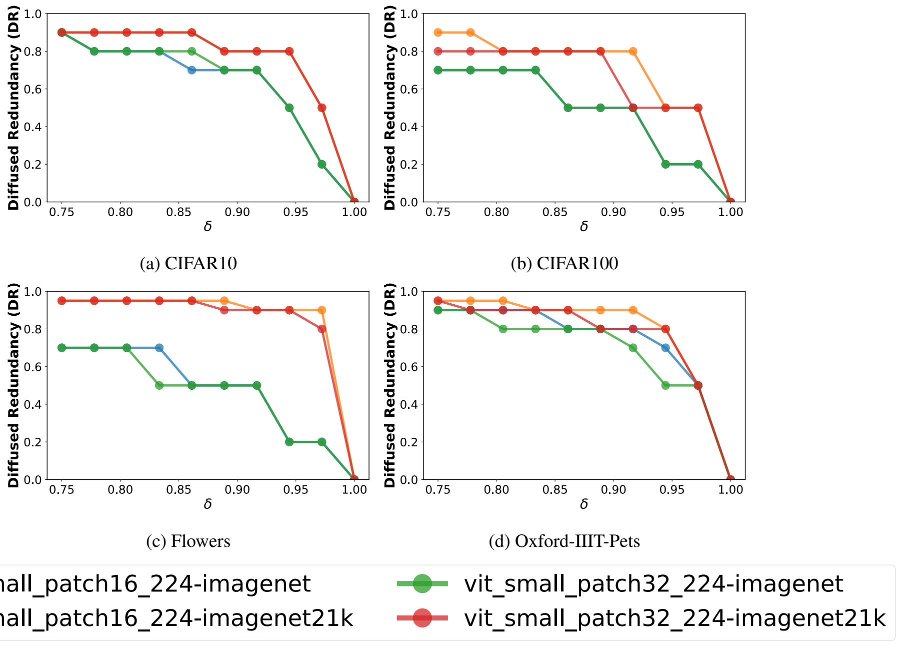


Figure 16: **[Comparison Across Upstream Datasets]** We see that degree of diffused redundancy depends a great deal on the upstream training dataset, in particular models trained on ImageNet21k exhibit a higher degree of diffused redundanacy, although the differences in the degree of diffused redundanacy are downstream task dependent

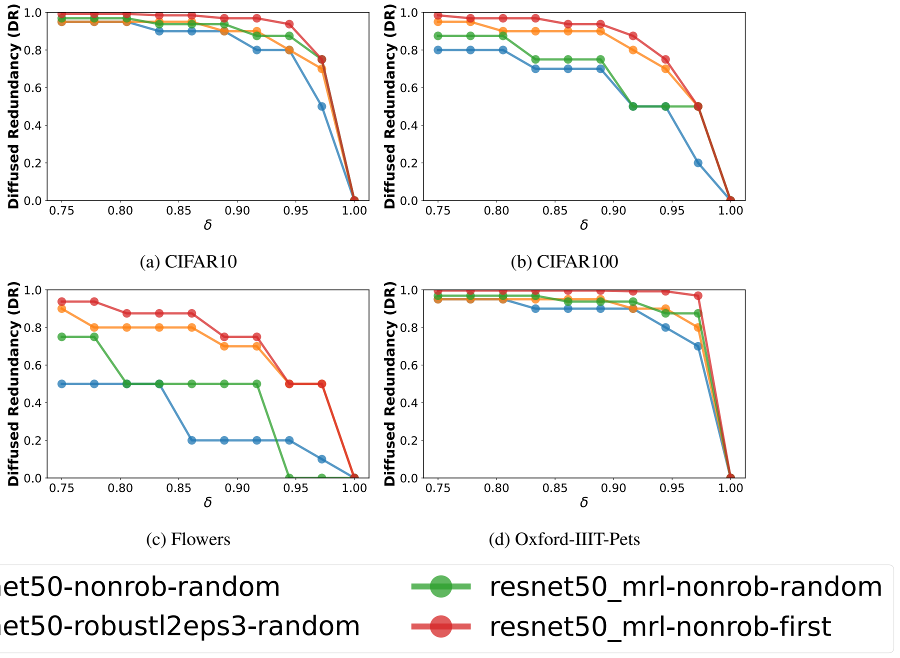


Figure 17: **[Comparison of Diffused Redundancy in MRL vs other losses]** Here we compare ResNet50 trained using multiple losses including MRL [26]. Red line shows results for part of the representation explicitly optimized in MRL, whereas green line shows results for parts that are picked randomly from the same representation. Even the MRL model shows a significant amount of diffused redundancy despite being explicitly trained to instead have structured redundancy. This figure shows diffused redundancy (DR) for all plots in Figure 7.

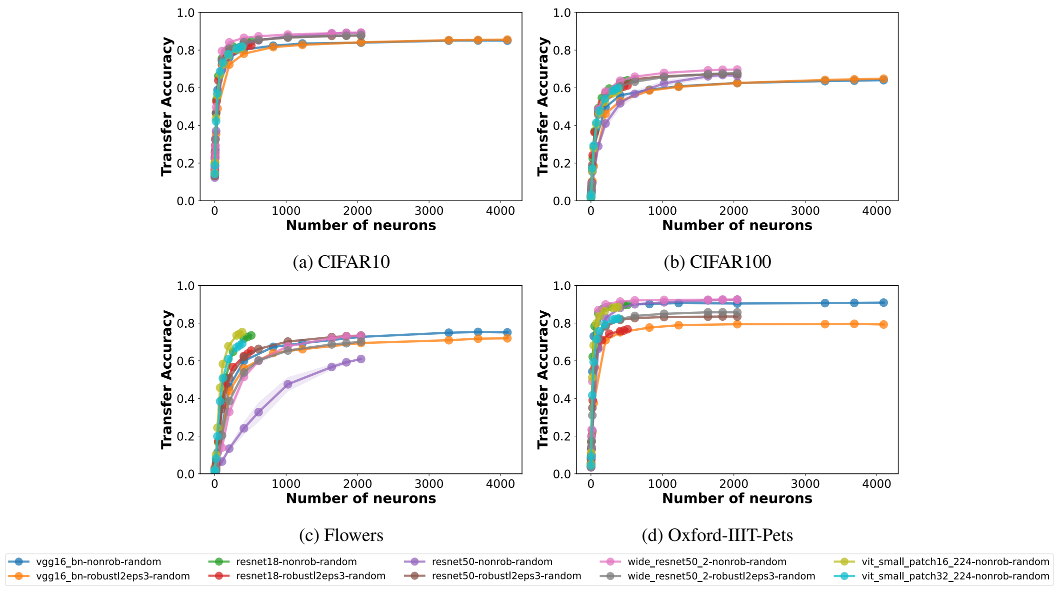


Figure 18: **[Comparisons Across Architectures For Downstream Task Accuracy]** This shows the same plots as Figure 4, except showing absolute number of neurons on the x-axis

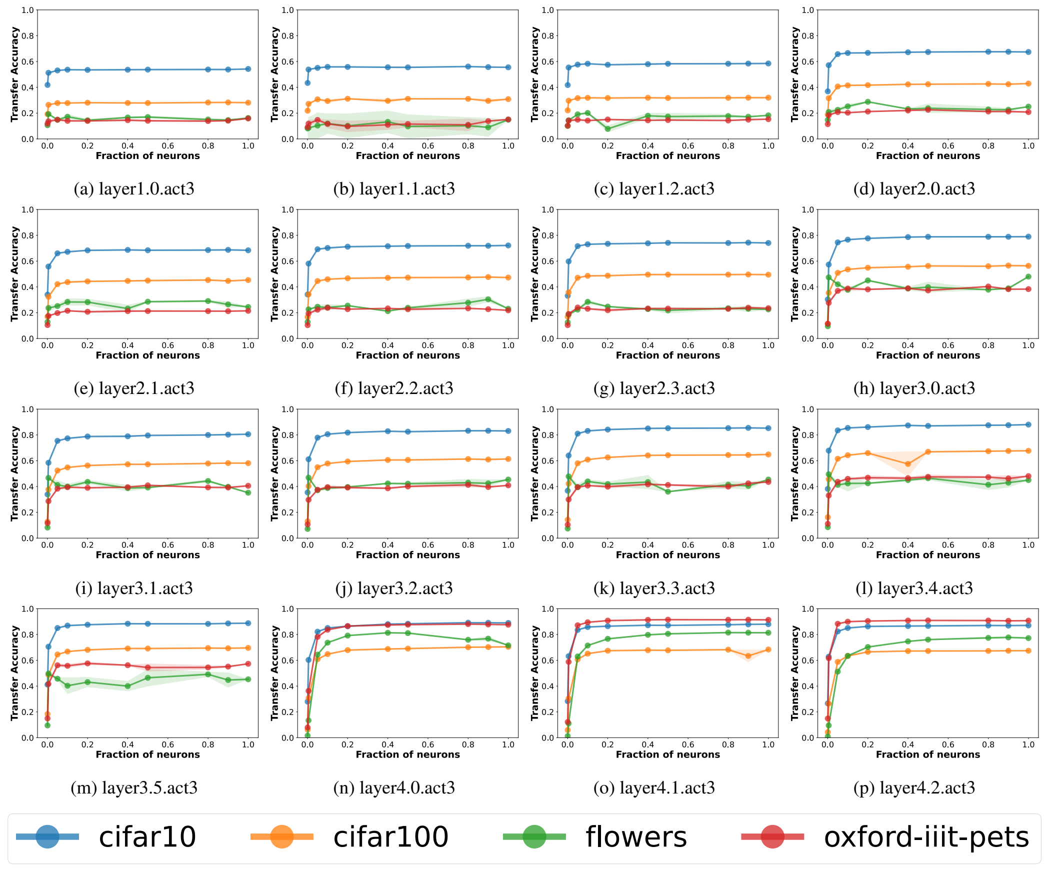
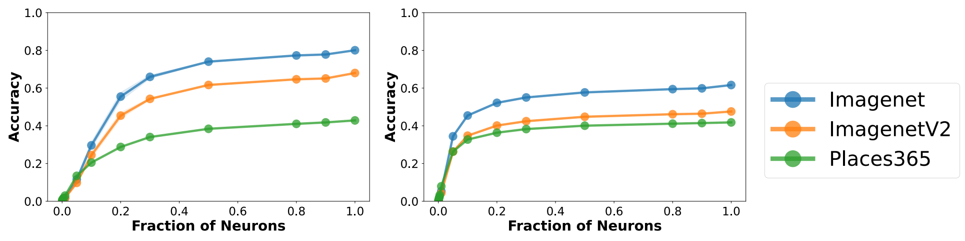
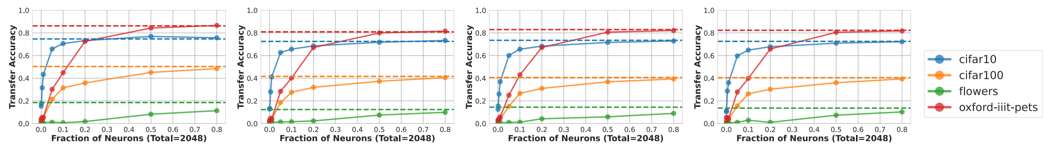


Figure 19: **[Middle Layers; ResNet50, trained with CrossEntropy loss on ImageNet1k]** We see that as we go deeper in the network, accuracy progressively increases. We see even middle layers exhibit diffused redundancy, and accuracy plateaus very quickly for earlier layers. `layerX.Y.act3` refers to the $Yth$ residual connection in the $Xth$ ResNet block and act3 indicates that we're taking the value after the activation (ReLU) has been applied.

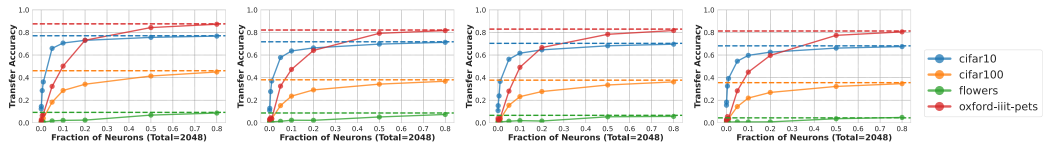


(a) ResNet50 nonrobust (b) ResNet50 robust $\ell_2 \epsilon = 3$

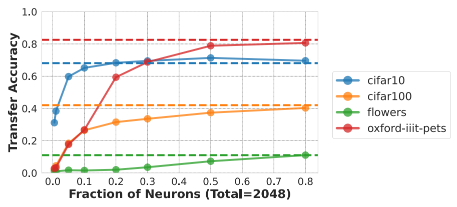
Figure 20: **[Performance on ImageNet1k, ImageNetV2, and Places365]** We check for the performance of randomly chosen subsets of neurons on harder tasks like ImageNet1k, ImageNetV2, and Places365. We find that diffused redundancy holds for all these harder tasks as well. Additionally, we see that randomly dropping neurons still preserves the accuracy gap between ImageNet1k and ImageNetV2.



(a) Dropout = 0.1 (b) Dropout = 0.2 (c) Dropout = 0.3 (d) Dropout = 0.4

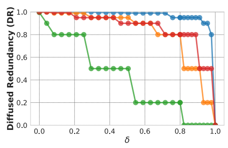


(e) Dropout = 0.5 (f) Dropout = 0.6 (g) Dropout = 0.7 (h) Dropout = 0.8

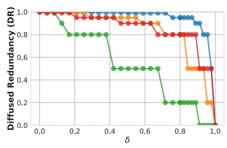


(i) DeCov Regularization (0.0001)

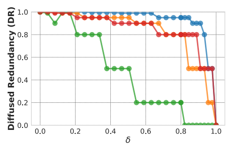
Figure 21: **[Dropout and DeCov regularizer's effect on Diffused Redundancy]**



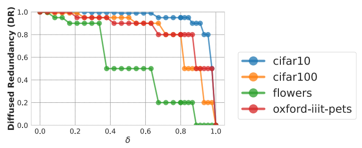
(a) Dropout = $0.1$



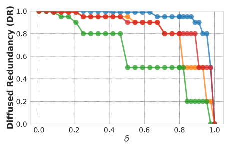
(b) Dropout = $0.2$



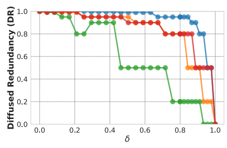
(c) Dropout = $0.3$



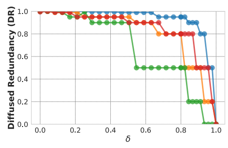
(d) Dropout = $0.4$



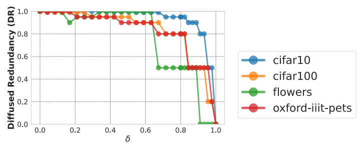
(e) Dropout = $0.5$



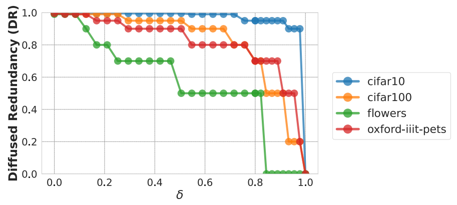
(f) Dropout = $0.6$



(g) Dropout = $0.7$



(h) Dropout = $0.8$



(i) DeCov Regularization $(0.0001)$

Figure 22: **[Dropout and DeCov regularizer's effect on Diffused Redundancy]** Same results as Figure 21, but showing $DR$ estimates (Eq 1). Lines that are more towards the right (*i.e.*more on the "outside") mean they exhibit more diffused redundancy.