# VRDL HW1

111550132 張家睿

## Introduction:

From my experience, my first idea is to choose the strongest model and its pre-trained weight. So, I tried ResNeXt101 and ResNeSt200 and found ResNeXt101 can have better performance. Then, I tried different data augmentation methods and hyperparameters to improve one model's accuracy. In the end, I trained 22 models with different random seeds and used them to compose a model using Model Soup or Greedy Soup. It turns out that Greedy Soup can have better accuracy.

## Method:

**Model Architecture:** ResNeXt101 with ImageNet pre-trained weight

**Hyperparameters**:

| Batch size | Epochs | Patience | Learning Rate(layers) | Learning Rate(fc) | Random Seeds |
|---|---|---|---|---|---|
| 64 | 35 | 10 | 1e-4 | 1e-3 | 1025, 102, 132, 140, 156, 158, 192, 33363, 373, 379, 514, 5751, 668, 672, 67, 694, 74, 77, 7810, 78, 960, 9996 |

I used *torchvision.transforms.v2* for **data augmentation**. My data transforms are listed below:

```
train: v2.Compose([
        v2.RandomResizedCrop(224),
        v2.RandomRotation(degrees=20),
        v2.RandomHorizontalFlip(),
        v2.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1),
        v2.ToTensor(),
        v2.ConvertImageDtype(torch.float),
        v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

```
val: v2.Compose([
        v2.Resize(360),
        v2.CenterCrop(224),
        v2.ToTensor(),
        v2.ConvertImageDtype(torch.float),
        v2.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

I would like to specially mention v2.Resize(360) in validation data augmentation, because I found that in this dataset, most of the detected items are in the center of the picture and they are only a small part of the picture. When using v2.Resize(360), the model can get a better cropped pictured to learn. Without adding v2.Resize(360) in front of v2.CenterCrop(224) can lead to 6% accuracy loss.

```
cutmix = v2.CutMix(num_classes=100, alpha=1.0)
```

CutMix is also an important method, it can improve fitting power by randomly cropping a region and replacing it with the corresponding part of another image.

**Model Soup(Uniform Soup)**:

Use multiple random seeds to train the same model. And average all model parameters to construct a stronger model.

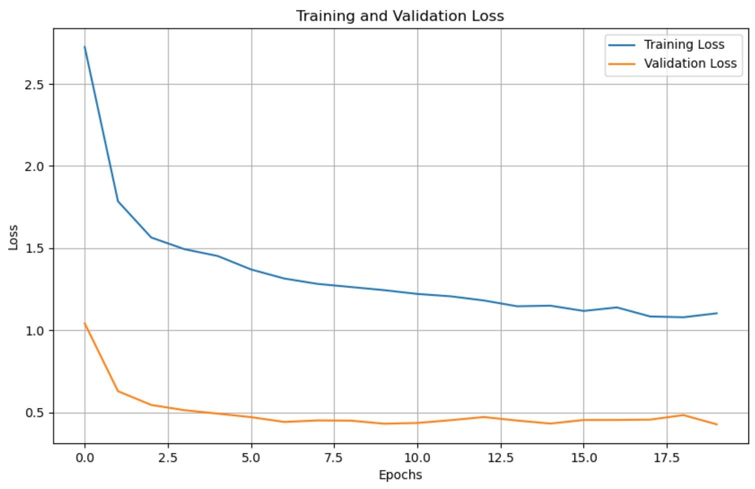
**Greedy Soup**:

Use multiple random seeds to train the same model. And sorted all models in descending order based on their accuracy on a validation set. Next, add each model with a different ratio to the base model. If adding a model improves the accuracy of the averaged-weight ensemble on the validation set, it is retained in the base model; otherwise, discarded.

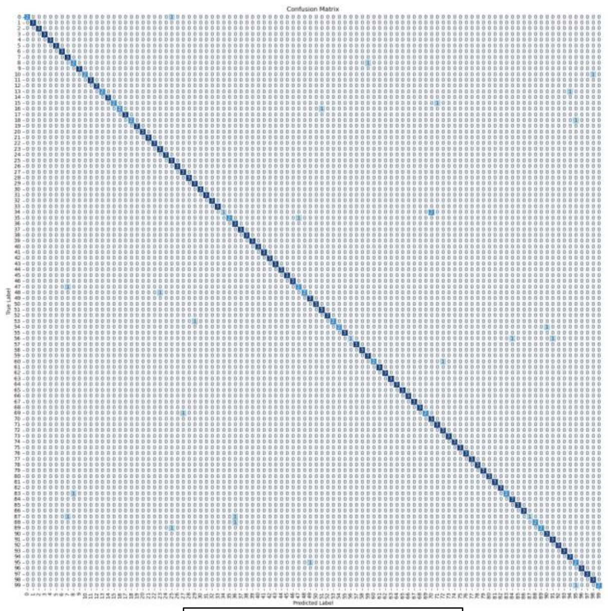
## Results:

**Single ResNeXt model** (with 22 different random seeds):

validation accuracy: 91.33~93.67% / testing accuracy: 95%



Training and Validation Loss



Confusion Matrix

**Model Soup:**

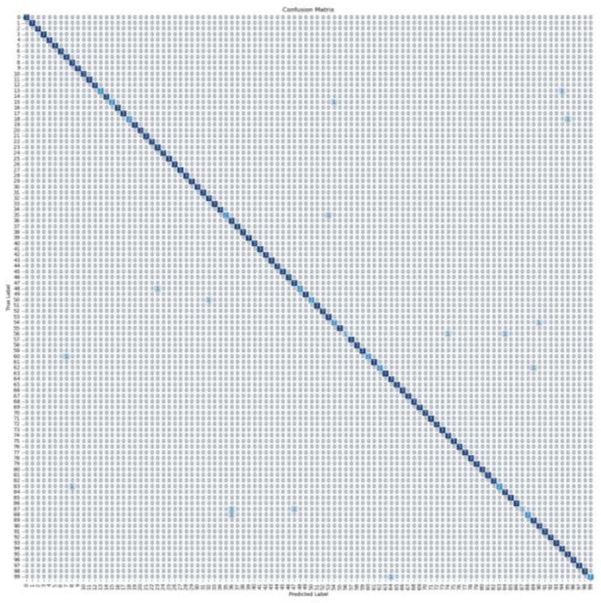
validation accuracy: 93.33% / testing accuracy: 95%

Results: No significant improvement.

**Greedy Soup**:

validation accuracy: 94% / testing accuracy: 96%

Results: Improve testing accuracy by 1%.



Greedy Soup Confusion Matrix

## References:

Model Soup and Greedy Soup paper: https://arxiv.org/pdf/2203.05482

Model Stock paper: https://arxiv.org/pdf/2403.19522

## Additional experiments:

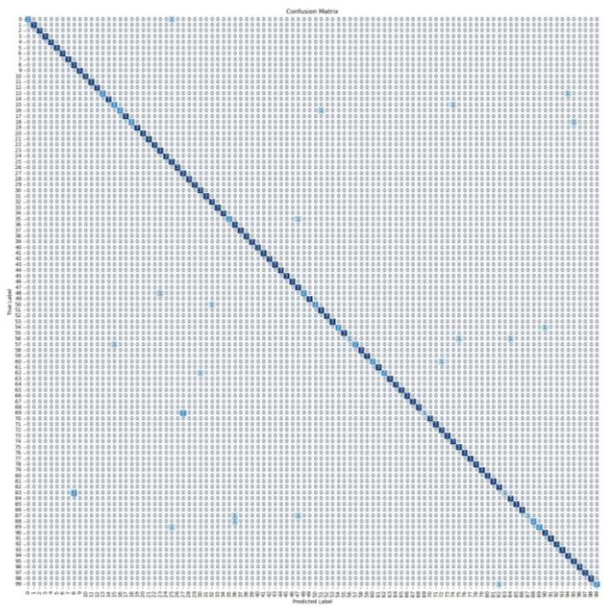
**Model Stock**:

I think that maybe the accuracy of Model Soup(Uniform Soup) didn't improve because the number of models is too small. In the original paper, it uses 50 models, but I only have time to train 22 models. Then I found Model Stock, which is a substitution for Model Soup. It only needs 2 fine-tuned models and can get nearly the same performance as Model Soup.

validation accuracy: 92.33% / testing accuracy: 95%

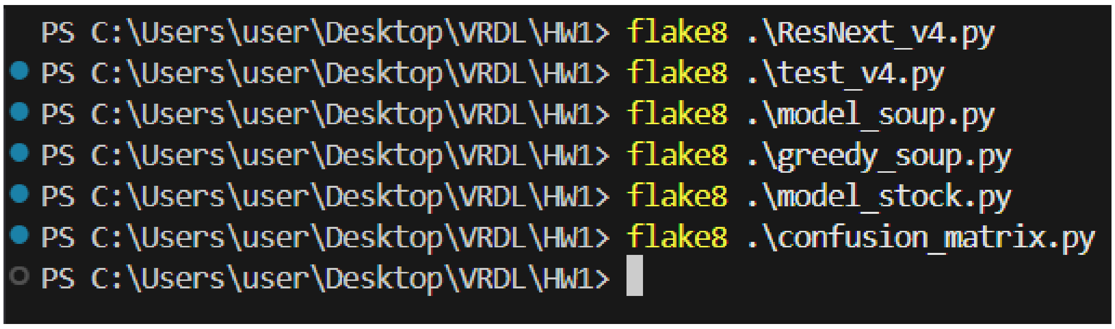
Results: No significant improvement.

The Model Stock model's performance is nearly the same as one model's performance. Maybe it is because the two combined models are too similar, so Model Stock can't have higher accuracy.



Greedy Stock Confusion Matrix

## Code Reliability:



```
  PS C:\Users\user\Desktop\VRDL\HW1> flake8 .\ResNext_v4.py
● PS C:\Users\user\Desktop\VRDL\HW1> flake8 .\test_v4.py
● PS C:\Users\user\Desktop\VRDL\HW1> flake8 .\model_soup.py
● PS C:\Users\user\Desktop\VRDL\HW1> flake8 .\greedy_soup.py
● PS C:\Users\user\Desktop\VRDL\HW1> flake8 .\model_stock.py
● PS C:\Users\user\Desktop\VRDL\HW1> flake8 .\confusion_matrix.py
○ PS C:\Users\user\Desktop\VRDL\HW1> 
```

## GitHub:

https://github.com/LouisChang0126/VRDL-HW1