ECE655 Project 06

Author: Stewart Schuler

Due: 12/30/2025

Contents

1 Part A

In the *EfficientNet-B3* model the vast majority of the model and parameters make up the feature extractor. In order to apply transfer learning to the model all the weights were frozen and the the final layer, called *classifier* was removed. To make the model work with the *CIFAR-100* dataset, the final layer way repalced with a linear layer with 1536 inputs and 100 outputs representing the fine class labels in the dataset.

```
# Load EfficientNet-B3 with pre-trained ImageNet weights
model = models.efficientnet_b3(weights=models.
    EfficientNet_B3_Weights.IMAGENET1K_V1)
# Remove last layer
model.classifier = nn.Identity()
for par in model.parameters():
    par.requires_grad=False

model.classifier = nn.Sequential(nn.Dropout(.5), nn.Linear
    (1536, 100, bias=True));
model.to(device)
```

Listing 1: part_a.py

Using the new model, the final layer was retrained for 3 epochs, and was found to have a top-1 accuracy value of 0.5441. For this experiment the training data was passed through the entire model on each epoch, even though since the weights in the feature extractor are frozen the 1536 features going to the final layer will be the same. Because of that, the average training time per epoch was 195 seconds.

2 Part B

In order to avoid the long training time discussed in the previous section, the dataset can be preprocessed by passing it through the frozen feature extractor of the model and saving the outputs. Then in order to train the final layer, it can be treated as a separate model with a input of 1536 features. Preventing the training set from needing to be unnecessarily recomputed.

```
def PreprocessedDataset(model_idt, loader, device=None):
    model.to(device)
    for i, (x, y) in enumerate(loader):
        model_idt.eval()
        output = model_idt(x.to(device))
        if i == 0:
            features = output.detach().cpu()
            labels = y.cpu()
        else:
            features = torch.cat(
                [features, output.detach().cpu()])
            labels = torch.cat([labels, y.cpu()])
    dataset = TensorDataset(features, labels)
    return dataset


model = models.efficientnet_b3(weights=models.
   EfficientNet_B3_Weights.IMAGENET1K_V1)
model.classifier = nn.Identity()
for par in model.parameters():
   par.requires_grad=False

filename ="CIFAR100preproc"
start_time = time.time();
pp_dataset_train = PreprocessedDataset(model, train_loader,
   device );
print(f"[{time.time()-start_time:.1f}] Finished Preprocesses
   Training Dataset");
pp_dataset_test  = PreprocessedDataset(model,  test_loader,
   device );
print(f"[{time.time()-start_time:.1f}] Finished Preprocesses
   Testing  Dataset");

torch.save(pp_dataset_train.tensors, f'dataset/{filename}
   _train.pth');
torch.save(pp_dataset_test.tensors,  f'dataset/{filename}
   _test.pth' );
```

```
31 print(f"[{time.time()-start_time:.1f}] Finished Writing
       Dataset");
```

Listing 2: part_b.py

With the training set preprocessed the new classifier was training for 30 epochs. The top-1 and top-5 accuracies are report in Figure 1. As can be seen the model converged after only a few epochs, but because of the inclusion of a dropout layer the continued training didn't hurt the test set accuracy. After 30 epochs the classifier layer had a top-5 test accuracy of 0.95 and a top-1 accuracy of 0.78. Because of the dataset preprocessing the average training time per epoch for this model was 0.89 seconds. A significant reduction compared to recomputing the frozen features.

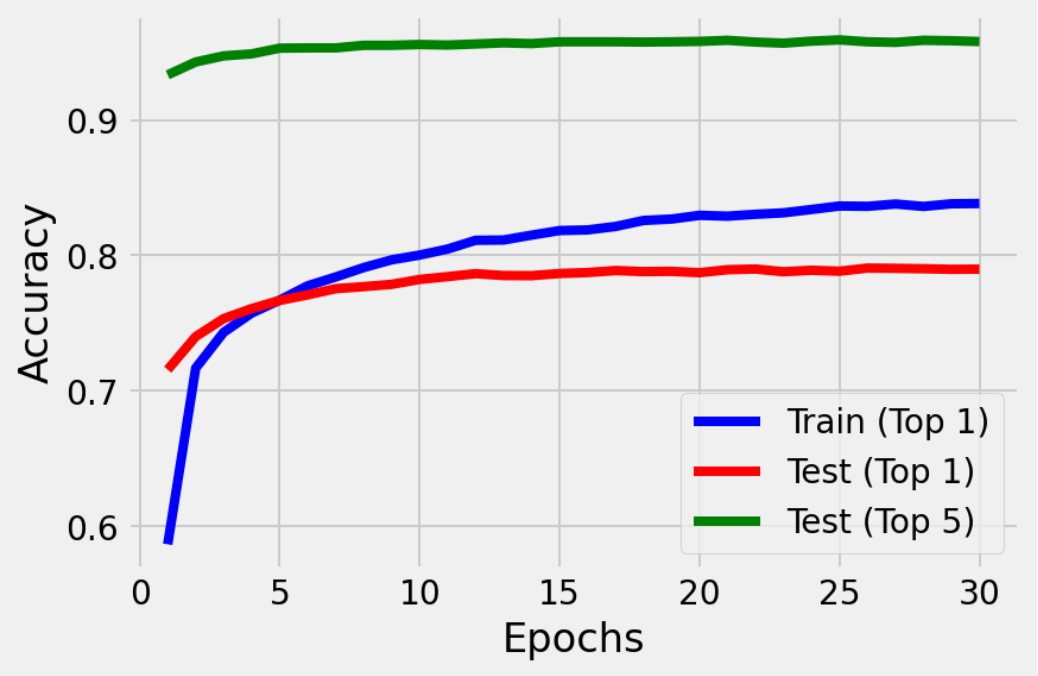

Figure 1: Fine Accuracies

3 Part C

The *CIFAR-100* dataset, contains both fine and coarse class labels. The dataset has 100 fine labels (used in the previous section) and 20 coarse classes. For this experiment the same procedure was applied as in Part B however the classifier was instead trained to learn the coarse labels.

Like before, the top-1 and top-5 accuracies are report in Figure 2. Unsurprisingly the accuracies increase across the board compared to the fine labels because this is a more forgiving classification problem. After 30 epochs the classifier had a top-5 test accuracy of 0.98 and a top-1 accuracy of 0.85.

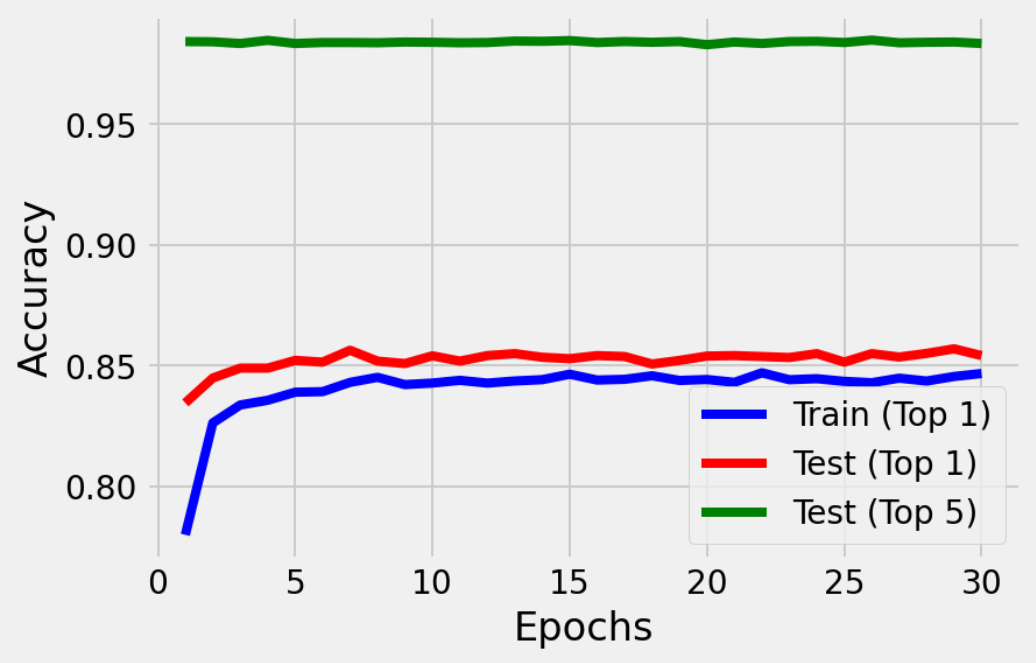

Figure 2: Coarse Accuracies

4 Part D

Next, the two fine and coarse classifiers were combined into a single class called *BrainNet*. Which when given an inputs returns both the fine and coarse classification while sharing a common feature extractor to prevent redundant computations.

```
class BrainNet:
    def __init__(self, model_common, model_coarse, model_fine
    , device='cpu'):
        self.model_common_ = model_common;
        self.model_coarse_ = model_coarse;
        self.model_fine_   = model_fine;
        self.model_common_.to(device);
        self.model_coarse_.to(device);
        self.model_fine_.to(  device);
        self.model_common_.eval();
        self.model_coarse_.eval();
        self.model_fine_.eval();

    def __call__(self, x ):
        common_output = self.model_common_(x.to(device))
        y_pred_fine   = model_classifier_fine(
    common_output );
        y_pred_coarse = model_classifier_coarse(
    common_output );
        return y_pred_coarse, y_pred_fine
```

Listing 3: part_d.py

In order to test out the *BrainNet* model a non-*CIFAR-100* dataset was chosen. The test data comes from the *Tiny-ImageNet* dataset downloaded from *Hugging Face*[1]. This dataset contained 100,000 64x64 images spread across 200 class labels. The images were resized and normalized to match the *CIFAR-100* images used in training. Because this data is coming from a different dataset with different labels it wouldn't be expected the the *BrainNet* model would get equal performance to the test set used in the previous sections. Additionally a large number of the *Tiny-ImageNet* don't have a direct analogous *CIFAR-100* label. To address this problem 11 classes were chosen from the *Tiny-ImageNet* labels that either directly overlapped with *CIFAR-100* labels or I determined were similar enough that the model should be able to classify them. The mapping of the chosen 11 classes is as follows, for the coarse labels they were chosen to be the corresponding coarse label to the associated fine label.

[1]https://huggingface.co/datasets/zh-plus/tiny-imagenet

```
common_classes = {
  "186": {"cfar_fine": 53 }, # Orange -> Orange
  "185": {"cfar_fine": 51 }, # Mushroom -> Mushroom
  "177": {"cfar_fine": 61 }, # Plate -> Plate
  "8":   {"cfar_fine": 79 }, # Black Widow -> Spider
  "9":   {"cfar_fine": 79 }, # Tarantual -> Spider
  "77":  {"cfar_fine": 15 }, # Snail -> Snail
  "18":  {"cfar_fine": 45 }, # Lobster -> Lobster
  "34":  {"cfar_fine": 43 }, # Lion -> Lion
  "92":  {"cfar_fine": 39 }, # Computer Keyboard -> Keyboard
  "114": {"cfar_fine": 41 }, # Lawn Mower -> Lawn Mower
  "135":  {"cfar_fine": 9 }, # Soda Bottle -> Bottle
}
```

Listing 4: part_d_2.py

Using the chosen labels as a subset of the *Tiny-ImageNet* dataset, there were 5440 new images that could be tested. After running them through the *BrainNet* model (which was training on *CIFAR-100* data), the model produced the accuracies shown in Table 1. As can be seen the model did quite well at classifying these new images.

BrainNet	**Top-1**	**Top-5**
Fine	0.68	0.84
Coarse	0.74	0.92

Table 1: *BrainNet* Accuracies

For further results the confusion matrix was generated for the coarse labels in Figure 3. As can be seen the biggest issue was with the images labeled, either *snail* or *lobster* by the *Tiny-ImageNet* dataset. They were frequently confused with the *CIFAR-100* coarse label *insects*. It is easy to imagine how those could be challenging for the classifier. For completeness the fine classification confusion matrix is attached in Figure 5, however due to the number of classes it is difficult to read.

Next we consider 100 randomly chosen images from the *Tiny-ImageNet* dataset. The trail results are shown in Figure 4. From these 100 predictions, it was found that in the case of the coarse labels where teh model is correct the average confidence was 97.5%, and 69.4% when incorrect. For the fine model the average correct confidence was 99.2% and 51.4% when incorrect.

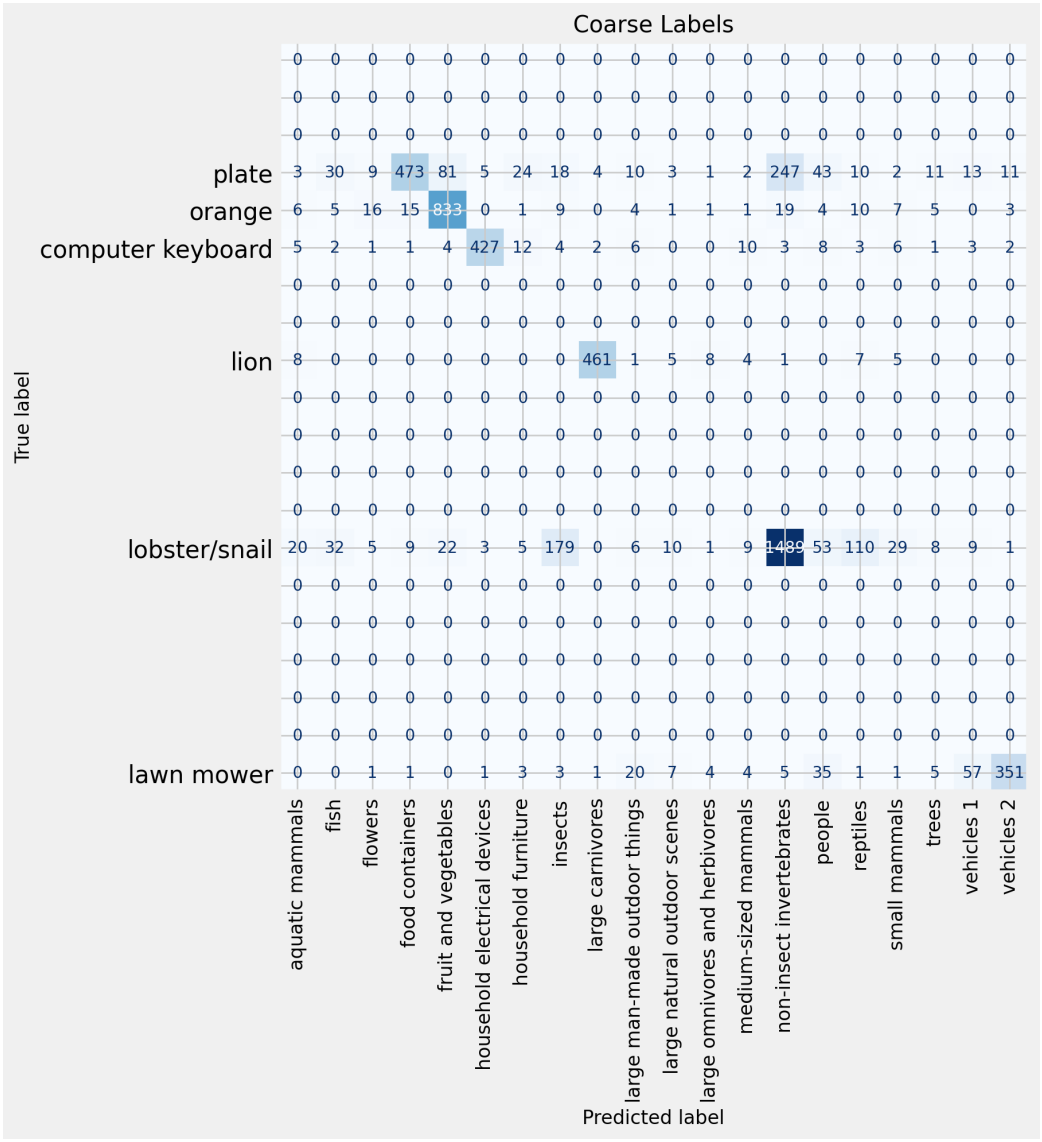

Figure 3: Coarse Accuracies

Data Index	True Label (Tiny ImageNet)	Coarse Label (CIFAR100)	Confidence	Fine Label (CIFAR100)	Confidence
0	lawn mower	vehicles 2	99.93	lawn_mower	99.98
1	tarantula	non-insect invertebrates	98.75	spider	99.90
2	mushroom	fruit and vegetables	99.91	mushroom	99.91
3	mushroom	fruit and vegetables	100.00	mushroom	100.00
4	orange	fruit and vegetables	99.99	orange	100.00
5	plate	food containers	61.86	plate	96.15
6	lawn mower	vehicles 2	99.99	lawn_mower	99.99
7	plate	food containers	99.14	plate	99.59
8	plate	food containers	64.53	plate	92.01
9	mushroom	fruit and vegetables	75.86	mushroom	96.97
10	orange	fruit and vegetables	96.04	orange	98.49
11	lobster	non-insect invertebrates	99.87	lobster	99.99
12	lawn mower	vehicles 2	98.16	lawn_mower	99.20
13	black widow	non-insect invertebrates	99.99	spider	100.00
14	orange	fruit and vegetables	100.00	orange	100.00
15	plate	food containers	87.65	plate	96.06
16	snail	large omnivores and herbivores	99.76	snail	99.90
17	plate	food containers	10.75	plate	70.71
18	plate	food containers	80.71	plate	95.70
19	plate	food containers	98.83	plate	99.55
20	mushroom	fruit and vegetables	73.82	mushroom	96.22
21	lobster	non-insect invertebrates	99.74	lobster	99.97
22	lawn mower	vehicles 2	99.99	lawn_mower	100.00
23	lion	large carnivores	99.94	lion	99.96
24	snail	non-insect invertebrates	99.94	snail	99.51
25	tarantula	non-insect invertebrates	5.58	spider	46.35
26	lawn mower	vehicles 2	26.32	lawn_mower	79.73
27	snail	non-insect invertebrates	100.00	snail	99.97
28	lawn mower	vehicles 2	23.57	lawn_mower	99.75
29	lobster	non-insect invertebrates	99.53	lobster	99.92
30	computer keyboard	household electrical devices	98.38	keyboard	97.78
31	computer keyboard	household electrical devices	100.00	keyboard	100.00
32	mushroom	fruit and vegetables	99.94	mushroom	100.00
33	lawn mower	vehicles 2	100.00	lawn_mower	100.00
34	lobster	non-insect invertebrates	99.96	lobster	99.99
35	plate	food containers	64.45	plate	97.21
36	mushroom	fruit and vegetables	99.99	mushroom	100.00
37	soda bottle	food containers	92.55	bottle	99.94
38	mushroom	fruit and vegetables	100.00	mushroom	100.00
39	tarantula	non-insect invertebrates	44.62	spider	88.01
40	orange	fruit and vegetables	99.96	orange	99.97
41	mushroom	fruit and vegetables	99.64	mushroom	99.96
42	lobster	non-insect invertebrates	99.83	lobster	99.98
43	computer keyboard	household electrical devices	100.00	keyboard	100.00
44	lawn mower	vehicles 2	56.48	lawn_mower	98.60
45	black widow	non-insect invertebrates	99.94	spider	99.99
46	snail	large omnivores and herbivores	93.44	snail	98.68
47	snail	large omnivores and herbivores	99.97	snail	99.88
48	lobster	non-insect invertebrates	99.85	lobster	99.96
49	lobster	non-insect invertebrates	99.97	lobster	99.99
50	black widow	non-insect invertebrates	99.99	spider	100.00
51	black widow	non-insect invertebrates	100.00	spider	100.00
52	plate	food containers	83.47	plate	87.71
53	black widow	non-insect invertebrates	99.99	spider	100.00
54	mushroom	fruit and vegetables	99.61	mushroom	99.74
55	tarantula	non-insect invertebrates	99.94	spider	100.00
56	mushroom	fruit and vegetables	100.00	mushroom	100.00
57	orange	fruit and vegetables	99.97	orange	100.00
58	lion	large carnivores	99.80	lion	99.96
59	tarantula	non-insect invertebrates	99.56	spider	99.74
60	mushroom	fruit and vegetables	100.00	mushroom	100.00
61	snail	non-insect invertebrates	99.72	snail	99.52
62	lion	large carnivores	99.99	lion	99.98
63	lobster	non-insect invertebrates	99.98	lobster	99.99
64	lawn mower	vehicles 2	99.51	lawn_mower	99.05
65	orange	fruit and vegetables	99.95	orange	99.99
66	black widow	non-insect invertebrates	99.99	spider	100.00
67	mushroom	fruit and vegetables	99.94	mushroom	99.92
68	computer keyboard	household electrical devices	8.34	keyboard	33.87
69	tarantula	non-insect invertebrates	99.84	spider	99.93
70	plate	food containers	91.55	plate	98.34
71	mushroom	fruit and vegetables	100.00	mushroom	100.00
72	snail	non-insect invertebrates	99.26	snail	98.94
73	snail	non-insect invertebrates	99.96	snail	99.89
74	soda bottle	food containers	99.88	bottle	99.52
75	soda bottle	food containers	99.97	bottle	99.99
76	mushroom	fruit and vegetables	40.54	mushroom	88.66
77	lobster	non-insect invertebrates	99.99	lobster	100.00
78	soda bottle	food containers	87.12	bottle	93.29
79	soda bottle	food containers	99.99	bottle	99.98
80	lobster	non-insect invertebrates	96.94	lobster	99.91
81	lobster	non-insect invertebrates	99.99	lobster	100.00
82	plate	food containers	61.41	plate	57.96
83	orange	fruit and vegetables	100.00	orange	100.00
84	snail	non-insect invertebrates	99.48	snail	99.63
85	computer keyboard	household electrical devices	99.94	keyboard	99.76
86	soda bottle	food containers	99.98	bottle	99.99
87	tarantula	non-insect invertebrates	97.30	spider	99.86
88	orange	fruit and vegetables	98.91	orange	99.18
89	snail	non-insect invertebrates	98.09	snail	99.84
90	plate	food containers	90.74	plate	97.66
91	lion	large carnivores	90.07	lion	95.41
92	lion	large carnivores	99.98	lion	100.00
93	lobster	non-insect invertebrates	99.98	lobster	99.99
94	computer keyboard	household electrical devices	96.27	keyboard	99.87
95	tarantula	non-insect invertebrates	98.75	spider	98.06
96	snail	non-insect invertebrates	96.27	snail	81.43
97	orange	fruit and vegetables	100.00	orange	99.99
98	lawn mower	vehicles 2	18.66	lawn_mower	80.81
99	orange	fruit and vegetables	62.72	orange	97.30

Figure 4: 100 *Tiny-ImageNet* Test Points

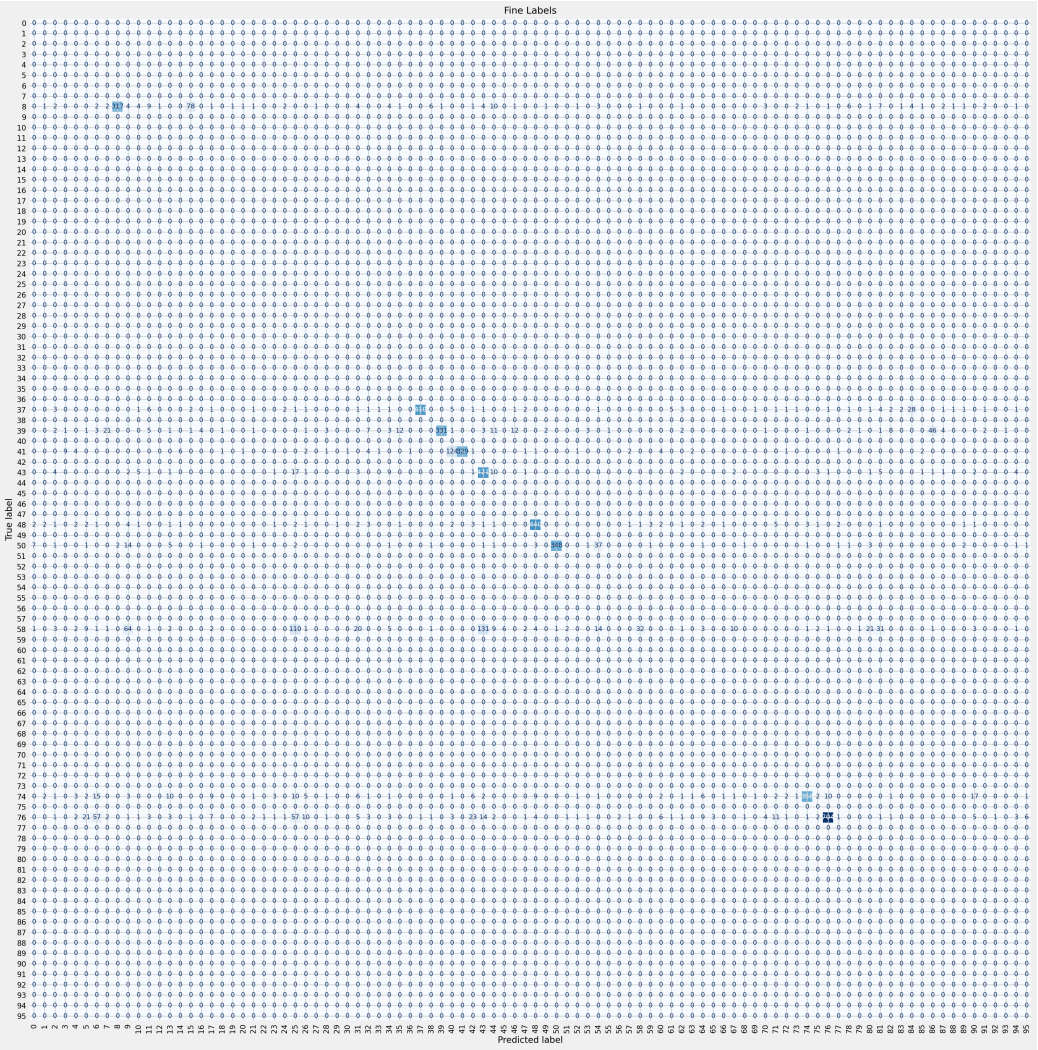

Figure 5: Fine Accuracies