



Module 4 - Session 1.5: Transfer Learning

What is Transfer Learning?

Standing on the Shoulders of Giants

- Start with a model pre-trained on a massive dataset (vs. random weights)
- **Goal:** Apply learned visual features to a new problem

Two key approaches:

1. **Feature Extraction:** Find patterns, train only the classifier
2. **Fine-Tuning:** Adjust valid weights to fit specific task



The Power of ImageNet

The “University” for Computer Vision Models

Dataset Stats:

- **14 Million+** Images and **1,000** Categories

Why it matters:

- Models learn universal visual features (edges, textures, shapes)
- These features transfer to *your* specific problem
- Even transfers to different domains (e.g. medical imaging)



Why Use Transfer Learning?

1. **Less Data Required:** You don't need millions of images; hundreds or thousands can suffice.
2. **Faster Training:** The model already knows how to "see"; it just needs to learn your specific classes.
3. **Better Performance:** Starting with good weights usually leads to higher accuracy than starting from scratch.



PyTorch Implementation

Step 1: Load a Pre-trained Model

```
1 from torchvision import models
2
3 # Load ResNet18 with default (ImageNet) weights
4 model = models.resnet18(weights='DEFAULT')
```



Step 2: Freeze the Feature Extractor

```
1 # Prevent backprop through these layers
2 for param in model.parameters():
3     param.requires_grad = False
```



PyTorch Implementation (Continued)

Step 3: Replace the “Head” (Output Layer)

The original ResNet output 1000 classes. We need it to output *our* number of classes (e.g., 2).

```
1 # Check input size of the final layer
2 num_ftrs = model.fc.in_features
3
4 # Create a new linear layer for our specific problem
5 model.fc = nn.Linear(num_ftrs, 2)
```

Now, only `model.fc` has `requires_grad=True` by default.



Summary

- **Don't reinvent the wheel:** Use pre-trained models.
- **ImageNet:** The massive dataset that gives models their "vision".
- **Workflow:** Load Model → Freeze Parameters → Replace Head → Train.