THE LAB — DAY 07

Standing on Giants' Shoulders

Transfer learning — borrow a giant's brain, teach it your trick

Day 07 · Week 2 · Unlocks 🎀 RPS Arena

1 Mission briefing

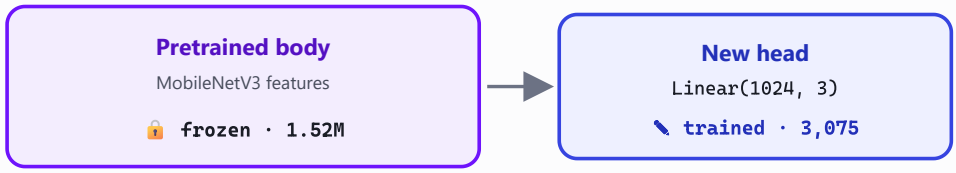
Training the Photo Detective from scratch was real work — and it still tops out near 70%. What if you could borrow a network that already spent weeks studying millions of images, and teach it just *one* new trick? That is **transfer learning**. Today you stand on a giant's shoulders: freeze a pretrained MobileNetV3, transplant a fresh head, and teach it rock-paper-scissors with photos you collect yourself. *Previously:* you unlocked 🕵️ Photo Detective, a CNN that classifies real photos.

- ☐ Understand **transfer learning** (an expert chef, one new dish)
- ☐ Multiply your data with **augmentation** (flip / rotate)
- ☐ Prove the giant matters: **pretrained vs random**
- ☐ Freeze the body, transplant a new **head** (1.52M vs 3,075)
- ☐ Collect your own photos by webcam or phone
- ☐ Wire the model into the RPS referee

2 Field guide the concepts

Transfer learning. An expert chef learning one new dish does not relearn how to hold a knife. A network pretrained on a million photos already knows edges, textures, and shapes — knowledge that transfers to *your* problem for free.

Freeze the body, transplant the head. Keep the giant's feature extractor (its "body") frozen, and replace only the final classifier (its "head") with a fresh 3-class layer. That is 1,517,856 frozen parameters and 3,075 you actually train — fast, and it needs very little data.



Freeze the giant's body and train only a fresh 3-class head — 3,075 trainable parameters, including biases.

Free data, real proof. **Augmentation** flips and rotates each photo into several free training examples. And to *prove* the giant matters, compare a frozen pretrained body with a new head against an entirely trainable randomly initialized network. This demonstrates two practical strategies; it does not isolate initialization alone.

> 🧮 **Matrix Watch —** that frozen giant is millions of stored weights, and every forward pass through it is a stack of matrix multiplies. You retrain the final layer; the frozen layers still compute fresh features for every image.

A precise reading

The recorded lab freezes all existing parameters, then replaces the final Linear(1024, 1000) layer with Linear(1024, 3): 3,075 trainable parameters and 1,517,856 retained frozen parameters. The comparison trains the entire random network, so initialization and the training strategy both differ. Frozen weights do not freeze batch-normalization running statistics in train mode.

3 Lab missions check each off in the notebook

1 Finish the validation pipeline ☐

Assemble the eval transform: `Resize(256)` → `CenterCrop(224)` → ImageNet `Normalize`.

Expect: tensors shaped (3, 224, 224) in the giant's color world.

2 Freeze the existing parameters 🧊 ☐

Set `requires_grad = False` across the feature body.

Expect: trainable parameters become zero; replacing the final layer next creates 3,075.

3 Perform the transplant ☐

Swap `classifier[3]` for `nn.Linear(1024, 3)`.

Expect: a model that now outputs exactly 3 scores.

4 Point the optimizer at the new head ☐

Give `Adam` only the parameters that still require grad.

Expect: training touch the head and leave the body untouched.

5 Build a giant with no experience ☐

Make the same architecture from random initialization and train all its parameters.

Expect: the pretrained model crush the random one — proof of transfer.

6 Teach the model to referee ☐

Wire predictions into the rock-paper-scissors judge.

Expect: it call your move and pick a winner (try 😈 cheat mode).

4 Cheat sheet transfer learning in PyTorch

`mobilenet_v3_small(weights="DEFAULT")`	Download a pretrained giant.
`net.features.requires_grad_(False)`	Freeze the whole body in one call.
`model.classifier[3] = nn.Linear(1024, 3)`	Transplant a fresh 3-class head.
`datasets.ImageFolder(root, transform=tf)`	Read images; labels come from folder names.
`Resize(256) · CenterCrop(224)`	Match the giant's expected input size.
`Normalize(IMAGENET_MEAN, IMAGENET_STD)`	Use the exact colors the giant learned on.
`RandomHorizontalFlip()`	Free extra training data from a mirror.
`Adam(head.parameters(), lr=1e-3)`	Train only the unfrozen head.
`torch.jit.trace(model, ex).save(path)`	Freeze the fine-tuned model to a file.

5 🔓 Unlock your Studio module

✂️ RPS Arena

Fine-tune, trace, and save your model. The unlock cell must write exactly:

`ai_studio/models/rps_arena.pt` `ai_studio/models/rps_labels.json`

Then play rock-paper-scissors against your own model — and flip on 😈 cheat mode:

```
$ python ai_studio/app.py
```

6 Mini-glossary

transfer learning
Reusing a model trained on one task to jump-start another.

fine-tuning
Lightly retraining part of a pretrained model on your data.

classifier head
The final layer(s) that turn features into class scores.

pretrained weights
The knowledge a network already learned on a huge dataset.

frozen layer
A layer whose weights are held fixed during training.

data augmentation
Making new training examples by transforming existing ones.

ImageNet
The million-image dataset most giants are pretrained on.

7 Show & tell

1. How many photos did you train on, and how good is your Arena? Does cheat mode feel fair?
2. Pretrained vs random init — how big was the gap? What does that prove about the giant?
3. Where does your model get confused (lighting, angle, background)? How would more data help?

© 2026 Dr. Abdelghafour HALIMI · ahalimi.com · All rights reserved