Cats and Dogs CNN Classifier

Ho Huyen Trang

November 29, 2025

Summary

Convolutional neural networks (CNNs) are central to modern computer vision, as they learn hierarchical feature representations directly from images. Beyond traditional supervised learning, recent advances incorporate self-supervision, weak supervision, large-scale pretraining, and knowledge distillation to reduce reliance on labeled data.

In this project, a lightweight CNN (MobileNetV2) is trained for cat–dog classification using two strategies:

- **Self-supervised contrastive learning (SimCLR)**
- **Knowledge distillation** from a pre-trained ResNet50 teacher

Experimental results show that a MobileNetV2 student trained purely via knowledge distillation reaches — and sometimes exceeds — the performance of one initialized with SimCLR pretraining, while requiring significantly less training time.

Problem Statement

Deep CNNs offer strong performance in visual recognition but typically require:

- large labeled datasets, and
- computationally heavy architectures.

Lightweight models are efficient but can't be as concise trained from scratch on limited data. Knowledge transfer techniques—transfer learning, knowledge distillation, and feature-based adaptation—help smaller models inherit rich representations and improve accuracy.

Large labeled datasets are also costly, motivating self-supervised learning methods such as **SimCLR** that extract features from unlabeled data.

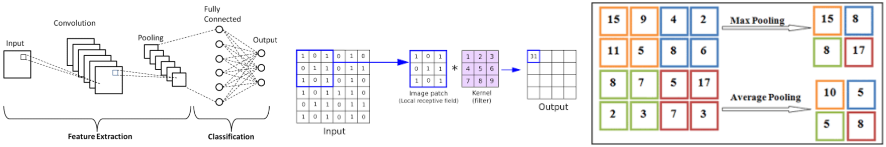
This project evaluates two techniques—SimCLR and logits-based knowledge distillation—to enhance MobileNetV2 on a cats vs. dogs classification task under limited resources (data + computation).

Methods: CNN Structure

Convolutional Neural Networks (CNNs) extract hierarchical spatial features from images through:

- **Convolution layers** — learn local patterns (edges, textures)
- **Pooling layers** — reduce spatial resolution and noise
- **Nonlinear activations (ReLU)** — introduce nonlinearity
- **Fully connected / classifier head** — map features to labels

CNN Structure



Examples of CNN components: full architecture (left), convolution operation (middle), and pooling operation (right).

Methods: SimCLR (Self-Supervised Learning)

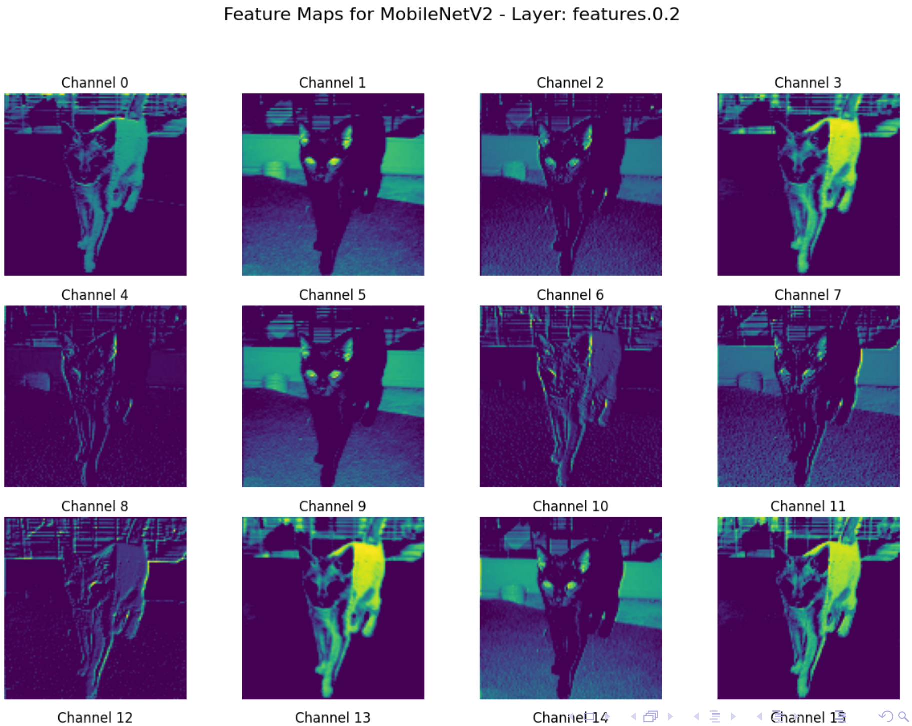
SimCLR learns image representations by maximizing agreement between augmented views of the same image.

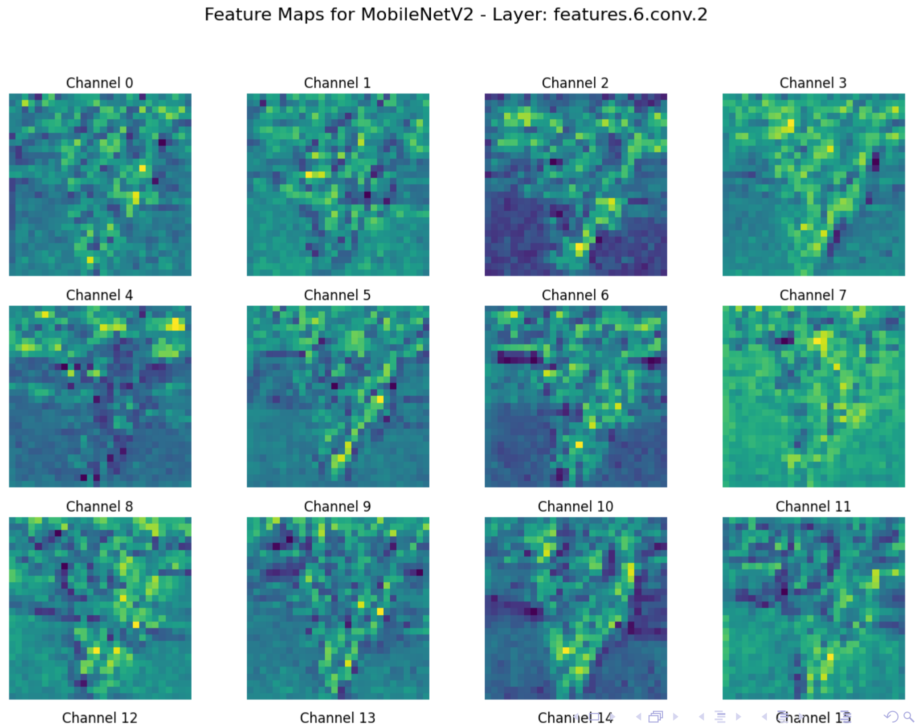
- Create two augmented views (x_i, x_j) of each image.
- Encode both using the same CNN encoder + projection head.
- Use the **NT-Xent contrastive loss**:

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\mathrm{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp(\mathrm{sim}(z_i, z_k)/\tau)}$$

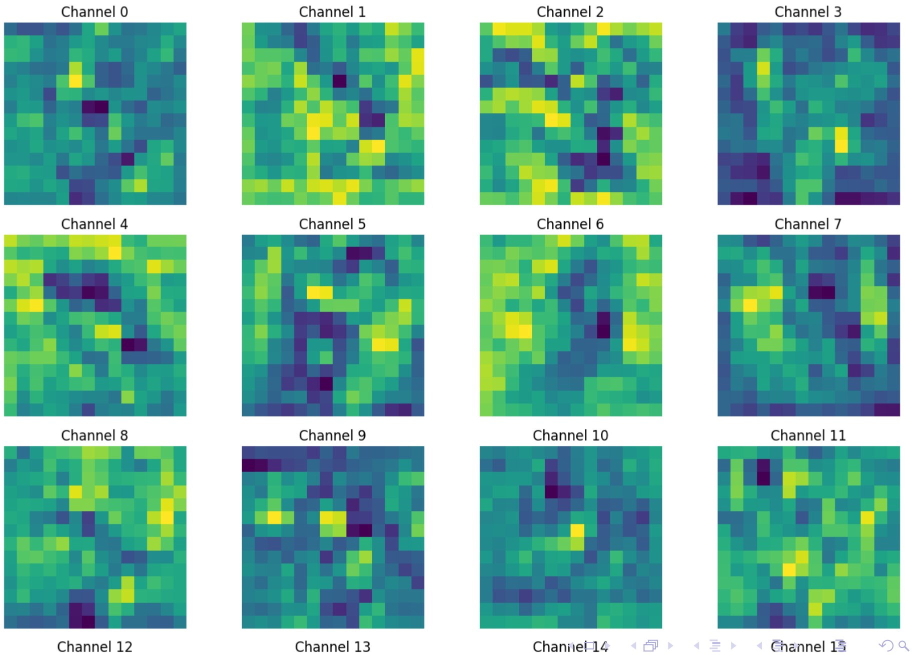
- $\mathrm{sim}(u, v)$: cosine similarity
- τ: temperature
- $2N$: number of augmented samples in the batch

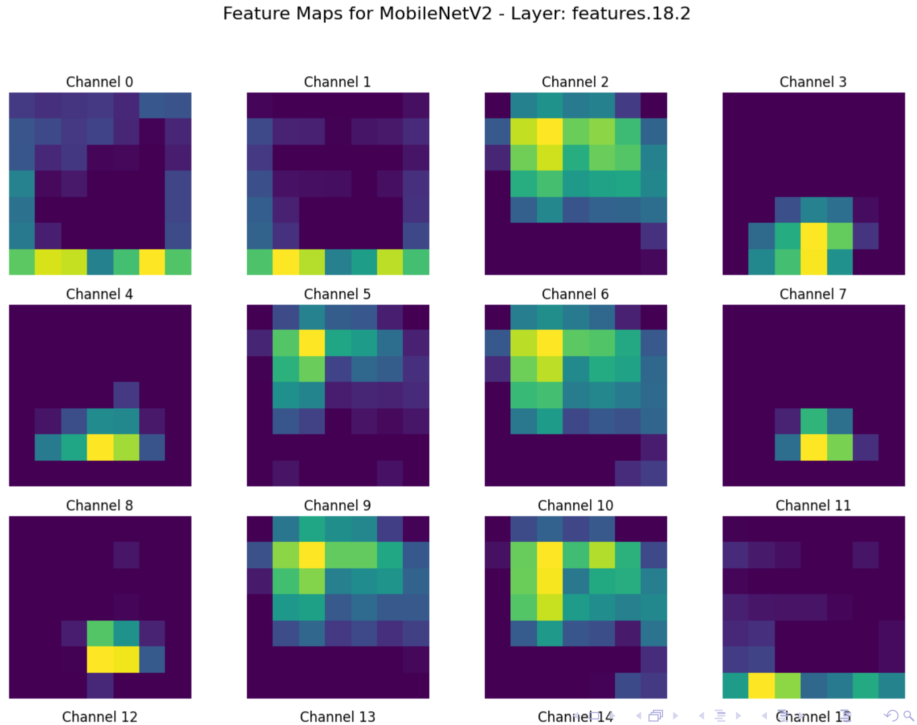
This pulls positive pairs together and pushes all other samples apart.


Feature Maps for MobileNetV2 - Layer: features.0.2
Channel 0
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Channel 6
Channel 7
Channel 8
Channel 9
Channel 10
Channel 11
Channel 12
Channel 13
Channel 14
Channel 15


Feature Maps for MobileNetV2 - Layer: features.6.conv.2
Channel 0
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Channel 6
Channel 7
Channel 8
Channel 9
Channel 10
Channel 11
Channel 12
Channel 13
Channel 14
Channel 15

Feature Maps for MobileNetV2 - Layer: features.13.conv.3




Feature Maps for MobileNetV2 - Layer: features.18.2
Channel 0
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Channel 6
Channel 7
Channel 8
Channel 9
Channel 10
Channel 11
Channel 12
Channel 13
Channel 14
Channel 15

Methods: Knowledge Distillation

Knowledge Distillation (KD) trains a smaller model (student) using soft targets from a larger teacher.

- Teacher outputs softened probabilities:

$$p_t = \mathrm{softmax}\left(\frac{z_t}{T}\right) \qquad p_s = \mathrm{softmax}\left(\frac{z_s}{T}\right)$$

- Student learns to match the teacher's softened output distribution and also by true labels:

$$\mathcal{L} = \alpha\, T^2\, \mathrm{CE}(p_t, p_s) \;+\; (1-\alpha)\, \mathrm{CE}(y, softmax(z_s))$$

- CE: cross-entropy loss
- T: temperature (softens teacher logits)
- α: balance between KD loss and true-label loss
- y: ground-truth one-hot label

Implementation

Datasets:

- Unlabeled training data: 10 000 , 6000, 3000
- Labeled data:
  - 420 for fine-tuning,
  - 180 for validation.
- Test set: 5 000

Environment

- Google Colab (free tier), training primarily on GPU

Libraries

- `torchvision.models`: MobileNetV2 (3M para), ResNet50 (IMAGENET-1K, 25M para)
- `lightly`: SimCLR augmentations and NT-Xent loss
- `torch.nn.functional`: CE, softmax, log_softmax, kl_div for KD loss

Results: SimCLR

Table: SimCLR Test Performance

Train Size	Acc	Prec-C	Rec-C	Prec-D	Rec-D
10k	0.78	0.78	0.78	0.78	0.79
6k	0.76	0.79	0.71	0.73	0.81
3k	0.72	0.72	0.71	0.72	0.72

Observations

- Modest acc decreases as data shrinks (0.78 $\rightarrow$ 0.72).
- Precision/recall remain well-balanced across Cat/Dog.
- Strong stability indicates effective self-supervised features.

Results: Knowledge Distillation

Table: KD Test Performance

Train Size	Acc	Prec-C	Rec-C	Prec-D	Rec-D
10k	0.92	0.94	0.90	0.90	0.94
6k	0.84	0.93	0.73	0.78	0.94
3k	0.75	0.79	0.69	0.72	0.81

Observations

- Very strong performance at 10k (Acc = 0.92).
- At 6k, Unbalances among classes as well as precisions and recalls — likely insufficient epochs.
- At 3k, accuracy is 0.75 with reasonable class balance.
- KD preserves useful boundaries even with limited data.

Comparison & Training Time

SimCLR vs KD

- KD outperforms SimCLR across all dataset sizes.
- Largest gap at 10k: KD 0.92 vs. SimCLR 0.78.
- SimCLR degrades more gradually + stable balance → more robust to data reduction.
- KD offers higher absolute accuracy.

Training Time (Epochs)

Train Size	SimCLR	KD
10k	62	46
6k	57	35
3k	59	25

Insights

- SimCLR generally requires more epochs, slow convergence on small data.
- KD converges faster because the student leverages teacher guidance.
- Efficiency in terms of training time of KD increases as data size decreases.

Conclusion

Key Findings

- Lightweight CNNs (MobileNetV2) perform effectively in low-resource settings.
- **Knowledge Distillation** consistently achieves higher accuracy across all dataset sizes.
- **SimCLR** provides stable and robust performance, even with very small labeled datasets.
- KD converges faster, while SimCLR may require more epochs even for small datasets.

Overall

- Both techniques significantly improve MobileNetV2 under limited data and computation.
- Demonstrates the usefulness of combining CNN efficiency with modern training strategies.

Future Experiments

- **Hyperparameter Tuning:** Improve performance through better augmentations, learning rates, KD temperature, and hard/soft loss ratios.
- **Model Variations:** Explore different teacher–student pairs and architectures to enhance distillation effectiveness.
- **Richer Distillation:** Transfer intermediate features or attention maps—not just logits—to strengthen student learning.
- **Adaptivity:** Investigate how students learn new classes, avoid forgetting, and generalize to new domains, also limits of their learning capabilities.