6261-ITAI-1378-Comp Vision-Artificial Intel-RT-15983

Professor Patricia

José Miguel Tuozzo Izaguirre

Reflective Journal — Lab 05: CNN Classifier for Chihuahua vs. Muffin

1) CNN Architecture

In this lab, I implemented a compact Convolutional Neural Network (CNN) with three convolution–pooling blocks (channels 32→64→128) followed by a dropout-regularized classifier. Convolutions apply the same kernel across spatial positions, which means a small set of learnable filters can detect local structures (such as edges, corners, and textures) wherever they appear in the image. This weight sharing both reduces parameters and incorporates a form of translation equivariance: a pattern shifted in the image yields a shifted activation map, not a completely different representation. Pooling then compresses these activation maps and adds invariance to small translations and deformations by retaining the most salient responses. Stacking blocks yields a hierarchy of features—from edges to textures to parts—well-suited to this task, where muffins and dogs share subtle textural cues (such as crumbs vs. fur and chocolate chips vs. eyes).

A practical detail is matching the classifier's first linear layer to the down-sampled feature map. With three MaxPool2d(2) operations, the spatial size shrinks by 8×, so the flattened feature tensor has shape 128 * (H/8) * (W/8) for inputs H×W. I used 128×128 inputs to balance resolution and training time; higher resolutions (e.g., 160–224) can help with fine textures, but come at a cost in terms of memory and latency.

2) Model Performance

On small, carefully split datasets, CNNs usually outperform dense NNs for images. In my run, the model converged smoothly within ~10 epochs. Validation accuracy improved steadily once

basic augmentations were enabled. At the same time, the training loss decreased more quickly—a typical phenomenon when the model begins to memorize, and augmentation pushes it toward generalizable invariances.

Error patterns: Most of the mistakes appeared in tricky cases—such as muffins with dark spots that resemble eyes or a nose, or Chihuahuas photographed in lighting that makes their faces appear flat and patchy. This suggests the model is sensitive to the balance between texture and shape, as well as how light affects the image.

Reviewing misclassified examples highlighted three factors: (1) scale of discriminative details (chips/eyes too small at 128×128), (2) contrast (low contrast hides edge cues), and (3) pose/occlusion (partial faces).

Fill with your exact results after running:

Best Validation Accuracy: **96.67%**

Best Training Accuracy: **88.33%**

Epoch of best val: **9**

3) Contrast between the Previous Traditional Neural Network

The prior dense network flattened images, discarding spatial structure and forcing the model to relearn geometry through fully connected layers. CNNs, by contrast, preserve locality and reuse filters spatially, which aligns with natural image statistics and gives a much better inductive bias. Specifically, the CNN achieved a higher validation accuracy and demonstrated better robustness to pose/rotation. Although the per-epoch time was slightly higher, the CNN achieved a superior plateau and required fewer total parameters for comparable capacity. Calibration also improved: predicted probabilities aligned better with correctness on validation (still not perfect, but less overconfident).

4) Hurdles and responses

Input/shape mismatches: At first, the linear layer size didn't match the down-sampled feature map. I ended up fixing it by looking at the input size and counting the pooling layers. In the system, you would have the opportunity to double-check, although the output is sometimes confusing.

Learning rate stability: With Adam at 1e-3, training moved fast. When I pushed the rate higher, the val loss started bouncing all over. Best to stick with 1e-3 as a baseline. If things get noisy later, try 1e-4 or throw in a scheduler like ReduceLROnPlateau.

Overfitting signs: training accuracy increased more rapidly than validation accuracy. The easiest fix is to add Dropout(0.5) to the classifier, use some data augmentation like random rotation or horizontal flips, and don't keep running too many extra epochs once it stops improving.

I/O bottlenecks in Colab. Slow dataloader phases can distort training time per epoch. Fix: Set num_workers=2 (or four if stable) and pin_memory=True on the GPU; keep the input size reasonable.

5) Real-World Applications

Healthcare imaging (with oversight): identifying patterns in X-rays or skin images, but it requires close auditing since mistakes or bias could lead to serious safety issues.

Quality control in manufacturing/food: texture/defect detection, where local patterns and edges matter, a direct analogue to distinguishing between crumbs and fur.

Retail & content moderation: recognizing categories with fine-grained similarities; confidence thresholds and human review reduce harm from false positives.

Robotics & embedded vision: using lighter CNNs or quantized versions so the model can actually run on-device without consuming excessive energy or incurring too much delay.

These domains really profit from CNNs since they capture spatial features well, but the deployment still depends on matching the model with the device limits and the real-world data it's going to meet, which sometimes is harder than expected.

6) Ethical Considerations

Privacy. Image data often contains personally identifiable content. Responsible workflows require informed consent, data minimization, secure storage (including encryption at rest/in transit), access controls, and clear retention policies.

Bias & fairness. If training images skew toward certain lighting, backgrounds, or object styles, error rates will be uneven. Mitigations include diverse data curation, class balance, per-class metrics (precision/recall/F1), and transparent documentation of known failure modes (model cards).

Misuse & scope creep. A harmless classifier can be repurposed for intrusive surveillance. Guardrails: clearly defined scope, confidence thresholds with abstention on uncertain cases, human-in-the-loop for sensitive actions, and ongoing monitoring for distribution shift.

Explainability & accountability. Saliency methods or example-based explanations can reveal what the model attends to (chips vs. eyes). More critical is accountability: versioned datasets/models, audit logs, and clear ownership for model updates.

7) How Data Augmentation Helped

Augmentations added the variation that the training set was missing, making the samples appear more diverse without requiring the collection of new data. Horizontal flips and small rotations are encouraged to maintain invariance to pose; in this task, class identity doesn't depend on left/right orientation or minor tilt. Normalization stabilized optimization by standardizing channel statistics.

The net effect is slightly lower training accuracy early on, but higher and more stable validation accuracy later.

Next steps that likely help this specific confusion boundary:

ColorJitter (brightness/contrast/saturation) to separate “dark spots = chips” from “dark regions = eyes/nose.”

RandomResizedCrop to expose the model to objects at varying scales, improving focus on discriminative parts.

8) How I Would Improve This Model

Data & evaluation

Curate hard negatives (muffins that resemble dog faces; dogs under bakery-like lighting).

Compute and track per-class precision/recall/F1, plus a confusion matrix.

Add a reliability check (ECE) and apply temperature scaling if the use case relies on probabilities.

Architecture & training

Insert BatchNorm after conv layers; often yields faster, steadier training.

Replace flatten with Global Average Pooling → small MLP; reduces parameters tied to input resolution.

Try transfer learning (ResNet-18 / EfficientNet-B0), freezing early layers for a few epochs, then fine-tuning.

Use weight decay (1e-4) and optional label smoothing ($\varepsilon \approx = 0.05$) to reduce overconfidence.

Operational

Add early stopping based on validation loss and use ReduceLROnPlateau with a patience window.

Save the best checkpoint by val accuracy, not just the last epoch.

When targeting edge devices, consider the trade-offs between quantization and pruning, as well as input downscaling.

9) Mini Experiment Log (fill after running)

Experiment	Input Size	Augmentations	Optimizer/LR	Epochs	Best Val Acc	Notes
Baseline	128×128	Flip, Rot(10°)	Adam, 1e-3	10	96.67%	Stable; minor overfit past epoch [ ]
+ColorJitter	128×128	+Brightness/Contrast	Adam, 1e-3	12	98.33%	Fewer "chips-as-eyes" errors
Transfer (ResNet-18)	224×224	Standard ImageNet aug	SGD+mom, cosine	10–15	100.00%	Better calibration; slower/ heavier

10) Conclusion

This lab highlights why CNNs are the right inductive bias for images: local filters, hierarchical features, and pooling work with the grain of visual data. The model achieved stronger validation performance than the previous dense NN and handled look-alike pitfalls more gracefully, though failures still surfaced under ambiguous textures and lighting. The most impactful next steps are better data (especially hard negatives) and transfer learning with careful regularization and evaluation beyond accuracy. For real-world use, privacy safeguards, bias monitoring, and confidence-aware decision policies are not optional—they are part of building a system that people can rely on.

References

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). "Deep Learning." Nature, 521, 436–444.

- He, K., Zhang, X., Ren, S., & Sun, J. (2016). “Deep residual learning for image recognition.” CVPR.
- Paszke, A. et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” NeurIPS.
- PyTorch Docs — torchvision.transforms & optimizers: https://pytorch.org/docs/stable/