Convolutional Networks: From Sliding Windows to ResNets

Lecture Notes Consolidated with Claude AI — CS6140 Machine Learning

---

Instructor's lecture plan for the note of the same title (`CNN_claude_1017.pdf`), Mon 11/16. Section numbers (§) refer to that note.

One lecture carries all of HW6 Problems 1, 2 and 6, so the plan is tight: the board does the ideas and one worked example each; the HW-sized numbers (accuracies, shifts, transfer) come from the note's figures, shown for a minute each. HW5 is due the next day (Tue 11/17), when HW6 opens; the RNN lectures follow on Thu 11/19 and Mon 11/23, so this is the only CNN lecture. The first 55 minutes must be enough to start Problem 1 on their own.

Lecture plan (100 minutes, mostly on the board)

Time	On the board	Worked example / demo
0–8	What an MLP does to an image: 203,530 weights, no idea of neighbours, a sleeve learned twice. The two issues: locality, the same detector at every position (a moving map is not an unchanged label). Draw LeNet with its shapes; leave it up all lecture. (§1)	"Frame × kernel window", the stick figure in two frames.
8–26	Cross-correlation (not convolution: no flip, and why it doesn't matter); output size; padding (the two reasons) and stride. (§2)	The 3×3 image by hand: 19, 25, 37, 43; the padded 4×4. The edge kernel $[1, -1]$ on a bar.
26–44	Channels: R, G, B in, d detectors out; the 4-index formula; parameter count independent of image size; 1×1 convolution. Unfold: one window per column, one matrix product; the offset-slice view. **Shapes in code**: $(n, c_i, h, w) \to (n, c_i k_h k_w, h'w') \to (n, c_o, h'w') \to (n, c_o, h', w')$; channel axis before the pixels. (§3)	56/72/104/120. The 4×4 patch matrix times the kernel; the batch axis riding along.
44–52	Pooling: max vs average; equivariance vs invariance; a move inside vs across a window. LeNet's layer table: where the parameters are. (§4–5)	The 1-D pooling table (positions 2, 3, 4).
52–56	Sigmoid + average → ReLU + max; what to expect against the MLP, and on shifted images. (§5)	Demo Segment 4 (shift curve), 1 minute.
56–70	Batch norm: the problem (layer inputs change scale), the formula, which axes (draw the grid), statistics of length c viewed as $(1, c, 1, 1)$; γ, β; train vs eval mode; running averages; the two PyTorch conventions. (§6)	The toy matrix: every column → $(-1.336, 1.069, 0.267)$; running_var 3.0.
70–84	Depth and the product of factors; $H(x) = x + R(x)$: redraw the handwritten residual block with its bullets; $\partial H/\partial x = I + \partial R/\partial x$; the 1×1 skip; pool outside the block. Global average pooling: $(n, 128, 8, 8) \to (n, 128)$. (§7)	Shape walk-through $16 \to 32$ channels, stride 2, $32 \times 32 \to 16 \times 16$.
84–96	Overfitting on CIFAR: augmentation (train only); equivariance vs invariance revisited (an augmentation teaches only what it applies: P6). Transfer: scratch / probe / fine-tune; **frozen vs `.eval()`** and why the probe's features can be computed once; pretrain → fine-tune = the LLM recipe. (§8)	Demo Segment 7 (augmented images; probe), 2 minutes.
95–100	HW6 map for P1, P2, P6; "debug small, run full". (§9)	

If time runs short (fallback): shorten the pooling table and the sigmoid → ReLU block to the board sketch (saves about 5 minutes). Do not cut the two shape strips or frozen-vs-eval: those are what students need for P1(A,D) and P2(C).

Reading instead of board time: the Weeks 11–14 roadmap (the remark at the end of §1) and the layer-norm half of the batch-norm example (it comes back in Week 14).

Board layout suggestion. Keep LeNet's pipeline across the top of the board all lecture; each new idea (padding, pooling, BN, the residual block) gets an arrow to the place in the pipeline where it lives.

The story, in six beats

Not a script — the one-paragraph version of each stage. The whole lecture is one sentence stretched out: *use the same small detector everywhere, and keep a deep stack of them trainable.*

1. **An MLP wastes its weights on images.** It has one weight per pixel per unit, so it learns a sleeve on the left and a sleeve on the right separately, and it would learn exactly as well from scrambled pixels. Images have structure the MLP ignores: neighbours matter, and a pattern means the same thing anywhere.

2. **One small kernel, slid everywhere.** Locality (a small window) plus weight sharing (the same window weights everywhere) is cross-correlation. Do 19 by hand; the rest is bookkeeping (padding, stride, channels). The punchline for the parameter count: it no longer depends on the image size.

3. **It is still a matrix product.** Unfold turns every window into a column, and the whole layer becomes HW5's XW again. This is how GPUs run it, and it is why the from-scratch version is slow: every pixel is copied many times.

4. **Convolution moves with the image; pooling forgives small moves.** Equivariance vs invariance, with the 1-D pooling table. Neither handles rotation or scale — hold that thought for augmentation.

5. **Two tricks keep deep stacks trainable.** Batch norm re-standardizes every layer's input at every step (draw which values share a mean); residual blocks give the gradient a path around each block. Both come back in every Transformer in Week 14.

6. **With little data, borrow.** Augmentation teaches the invariances you know; pretrained features give you what 1.3 million images learned. Fine-tuning 5,000 images to 0.88 is the cleanest preview of how large language models are used.

If you remember one line: *a CNN is an MLP that reuses one small set of weights at every position — and everything after LeNet is about making that stack deeper without breaking training.*