Watched a video on how to read research paper and took notes, anything missing?

How to Read Deep Learning Papers

1. Prepare external context

- **Goal:** Fill gaps in your background before deep-diving into the paper.
- **How:** Watch 5–7 short videos or read quick summaries on concepts you don't know that are directly referenced by the paper (e.g., if the paper builds on VGG, learn VGG first).

2. First read: internal context

- **Just read.** Resist the urge to search or debug on the first pass, read straight through and *mark* what you don't understand.
- **Mark categories while reading:**
    - **External Unknowns:** Concepts *outside* the paper that you don't know (new techniques, architectures, background theory).
    - **Internal Unknowns:** Things *inside* the paper you don't understand (why a matrix is used, what a given output represents, how a block works).
    - **Author fault:** Claims or reasoning that seem unclear or unjustified.
    - **Author fact-check:** Clear errors or inaccuracies from the authors.

3. Close the gaps

- **Fill external gaps** first (read/watch quick primers).
- **Minimize internal unknowns:** go line-by-line to understand what each section is doing.
- **Understand the motivation / problem statement.** Why was this research done? What problem does it solve?

4. Jump to the conclusion (summary)

- Read the conclusion/abstract early to get the high-level gist, this helps guide what to look for in the rest of the paper.

5. Figures = gold

- **Extract every figure** and understand it fully. Paste it somewhere (like a notion page or google doc) to carefully anaylse it, or just for memories/notes
- Images/figures often convey intuition faster than text (as we all already know) parse axes, labels, legends, and captions.

6. Understand the code (if provided)

- **Run the code** in your IDE and observe inputs/outputs.
- **Tweak and play** with parameters to see how behavior changes.
- **Track dependencies:** what is imported, which functions are defined where.
- **Note unknown lines** and isolate them to study separately.

7. Methodology deep-dive

- **Data:** What data is used? How is it fed to the model (batch size, num_workers, preprocessing, augmentations)? Look at the data yourself if possible.
- **Architecture:** Fully map out the model architecture — every block, layer, and connection. This builds intuition about training and behavior.
- **Training routine:** What does the train loop look like? Optimizer? Scheduler? Loss functions? Metrics?
- **Pipeline:** Sketch the entire pipeline — from raw data → preprocessing → model → loss → evaluation. *UNDERSTAND. THE. PIPELINE.*
- **Re-read** after this pass and see what still feels fuzzy.

8. Revisit remaining unknowns

- If something is still unclear after the above, loop back to targeted reading (papers, docs, short videos) or ask a focused question to a helper (peer, forum, assistant).

How to Read the Math

1. **Identify every formula** used or referenced in the paper — list them somewhere visible.
2. **Get intuition:** watch short explainer videos or ask for an intuition (e.g., ChatGPT/Claude) for each formula you don't immediately understand.
3. **Sketch Input** → **Process** → **Output** for each formula on paper:

    - What are the inputs?
    - What operation is applied?
    - What is the output and its shape/meaning?
4. **Symbol drill-down:** list and define every symbol/variable in isolation, then reconnect them.
5. **Why these formulas?** Connect each formula to the motivation — how does it help achieve the research goal?
6. **Consolidate:** write down what *you* understand and try to teach it (even if only to an imaginary student). Teaching reveals gaps.

How to Understand the Code

- **Run it** and observe inputs/outputs. Confirm behavior matches what the paper claims.
- **Trace data flow:** from first cell to last — sketch it with arrows and boxes.
- **Isolate unknowns:** if a function or loop confuses you, extract it and test it alone.
- **Understand structure:** classes, functions, arguments, return values — what goes in/out and why.
- **Document decisions:** why this op vs. that op, shapes chosen, nested loops, etc.
- **Nitpick:** read docs for used functions, evaluate time/space implications, and consider naming improvements.
- **Pen-and-paper mapping:** redraw the entire script or notebook flow. Focus how data transforms between steps (e.g., after scaling → after Conv block 1 → after Conv block 2).

Tools to Use

- **Notion** — notes, highlights, diagrams, todos, permanent record of your learning.
- **Excalidraw** — quick sketches, whiteboard-style architecture and pipeline drawings.
- **Claude with Explanatory Mode** — for niche clarifications when you can't find a clear explanation elsewhere.

Note -> I DID NOT use ChatGPT to take the notes, I wrote it myself on the notes app but the formatting was ruined while copy-pasting, and I was too lazy to manually do it. Anyway, if you guys wanna add onto this/give feedback, let me know!

Understanding Deep Learning Research Tutorial - Theory, Code and Math
Beens - how to read a ML paper (with maths)