UCLA COMPUTER SCIENCE FIELD NOTES

CS 163

Deep Learning for Computer Vision

Image tensors, convolutional networks, training, recognition, detection, generation, and robust vision

How these notes are written. I wrote each chapter the way I wish the material had been explained the first time: the real idea, the point where I usually get stuck, the simplest mental model that still stays honest, and a worked decision instead of a polished answer appearing from nowhere. Every chapter closes with practice and a fully written solution. If a sentence sounds impressive but does not help me solve or explain something, it does not belong here.

Daniel Mastick

Curriculum map, working notes, practice, and solutions

Course map

Week	Core thread	What should be solid by the end
1	**Images, vision tasks, and data pipelines**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
2	**Convolution and receptive fields**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
3	**Deep architectures and residual learning**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
4	**Optimization, regularization, and debugging**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
5	**Recognition, transfer, and calibration**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
6	**Detection and segmentation**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
7	**Visual generation and representation learning**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
8	**Robustness, interpretation, and responsible deployment**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
9	**Integration workshop**	Connect at least three chapters in one end-to-end problem and explain where their contracts meet.
10	**Synthesis and project review**	Audit assumptions and explain a complete solution from interface to failure handling.

Scope anchor: https://catalog.registrar.ucla.edu/course/2025/COMSCI163. The sequence is aligned to the official catalog description and expanded into a practical ten-week study plan.

How I would use this packet

Treat each numbered section as one chapter. Read the formal explanation, pause at the student interpretation, and see whether the easy version still says the same thing. Rework the example without looking, then cover the solution in the chapter practice box. The cumulative set at the end is deliberately mixed: knowledge becomes durable when I have to choose the tool instead of being told which chapter I am in.

Contents

CHAPTER 1 Images, vision tasks, and data pipelines

The idea

Computer vision maps pixels or video to classification, localization, detection, segmentation, retrieval, generation, or control. Sampling, color, geometry, labels, augmentation, splits, and provenance define model evidence.

Rebuild the chapter from first principles

The claim I need to own is this: Computer vision maps pixels or video to classification, localization, detection, segmentation, retrieval, generation, or control. Sampling, color, geometry, labels, augmentation, splits, and provenance define model evidence. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of images, vision tasks, and data pipelines. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach images, vision tasks, and data pipelines on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from the course foundations to convolution and receptive fields. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Computer vision maps pixels or video to classification, localization, detection, segmentation, retrieval, generation, or control.
- **Mechanism.** Sampling, color, geometry, labels, augmentation, splits, and provenance define model evidence.
- **Boundary.** The useful test is whether I can apply images, vision tasks, and data pipelines to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when images, vision tasks, and data pipelines is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I decide what invariance the task should have before I augment away information the model might need.

The easy version

Computer vision maps pixels or video to classification, localization, detection, segmentation, retrieval, generation, or control.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why split a medical image dataset by

patient rather than image? The conclusion is Images from one patient share anatomy, acquisition conditions, and near-duplicate content. An image split can leak patient-specific signals and exaggerate generalization.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why split a medical image dataset by patient rather than image?
Solution. Images from one patient share anatomy, acquisition conditions, and near-duplicate content. An image split can leak patient-specific signals and exaggerate generalization.

Practice 2. Give a short oral explanation of images, vision tasks, and data pipelines that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct images, vision tasks, and data pipelines from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a images, vision tasks, and data pipelines problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 2 Convolution and receptive fields

The idea

Convolution uses local shared filters across positions. Kernel size, stride, padding, dilation, channels, pooling, and stacked layers determine output shape, parameter count, equivariance, and receptive field.

Rebuild the chapter from first principles

The claim I need to own is this: Convolution uses local shared filters across positions. Kernel size, stride, padding, dilation, channels, pooling, and stacked layers determine output shape, parameter count, equivariance, and receptive field. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of convolution and receptive fields. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach convolution and receptive fields on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from images, vision tasks, and data pipelines to deep architectures and residual learning. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Convolution uses local shared filters across positions.
- **Mechanism.** Kernel size, stride, padding, dilation, channels, pooling, and stacked layers determine output shape, parameter count, equivariance, and receptive field.
- **Boundary.** The useful test is whether I can apply convolution and receptive fields to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when convolution and receptive fields is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I calculate tensor shapes on paper before debugging a network; shape arithmetic is part of the model.

The easy version

Convolution uses local shared filters across positions.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: A 32 by 32 input uses a 3 by 3 kernel, stride 1, and padding 1. What is the output size? The conclusion is The formula gives floor((32 plus 2 minus 3) divided by 1) plus 1, which is 32 in each spatial dimension. Output channels equal the number of filters.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. A 32 by 32 input uses a 3 by 3 kernel, stride 1, and padding 1. What is the output size?
Solution. The formula gives floor((32 plus 2 minus 3) divided by 1) plus 1, which is 32 in each spatial dimension. Output channels equal the number of filters.

Practice 2. Give a short oral explanation of convolution and receptive fields that includes its

model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct convolution and receptive fields from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a convolution and receptive fields problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 3 Deep architectures and residual learning

The idea

CNN architectures combine convolution, activation, normalization, downsampling, and classification heads. Residual connections learn a correction around an identity path, improving gradient flow and enabling deeper networks.

Rebuild the chapter from first principles

The claim I need to own is this: CNN architectures combine convolution, activation, normalization, downsampling, and classification heads. Residual connections learn a correction around an identity path, improving gradient flow and enabling deeper networks. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of deep architectures and residual learning. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.

3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach deep architectures and residual learning on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from convolution and receptive fields to optimization, regularization, and debugging. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** CNN architectures combine convolution, activation, normalization, downsampling, and classification heads.
- **Mechanism.** Residual connections learn a correction around an identity path, improving gradient flow and enabling deeper networks.

- **Boundary.** The useful test is whether I can apply deep architectures and residual learning to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when deep architectures and residual learning is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

My shorthand for this chapter is: a deeper model is only better when optimization, data, regularization, and task justify its capacity.

The easy version

CNN architectures combine convolution, activation, normalization, downsampling, and classification heads.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Why can a residual block ease optimization? The conclusion is The skip path carries activations and gradients directly. If extra layers are unnecessary, the residual branch can approach zero rather than relearning identity through nonlinear layers.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Why can a residual block ease optimization?
Solution. The skip path carries activations and gradients directly. If extra layers are unnecessary, the residual branch can approach zero rather than relearning identity through nonlinear layers.

Practice 2. Give a short oral explanation of deep architectures and residual learning that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct deep architectures and residual learning from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a deep architectures and residual learning problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 4 Optimization, regularization, and debugging

The idea

Training uses stochastic gradients, momentum or adaptive optimizers, learning-rate schedules, initialization, normalization, weight decay, augmentation, dropout, and early stopping. Debugging begins with data, loss, gradients, and tiny-batch overfitting.

Rebuild the chapter from first principles

The claim I need to own is this: Training uses stochastic gradients, momentum or adaptive optimizers, learning-rate schedules, initialization, normalization, weight decay, augmentation, dropout, and early stopping. Debugging begins with data, loss, gradients, and tiny-batch overfitting. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of optimization, regularization, and debugging. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.

- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach optimization, regularization, and debugging on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from deep architectures and residual learning to recognition, transfer, and calibration. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Training uses stochastic gradients, momentum or adaptive optimizers, learning-rate schedules, initialization, normalization, weight decay, augmentation, dropout, and early stopping.
- **Mechanism.** Debugging begins with data, loss, gradients, and tiny-batch overfitting.
- **Boundary.** The useful test is whether I can apply optimization, regularization, and debugging to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when optimization, regularization, and debugging is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

Before tuning a hundred hyperparameters, I make the model memorize ten examples; failure there usually means a bug.

The easy version

Training uses stochastic gradients, momentum or adaptive optimizers, learning-rate schedules, initialization, normalization, weight decay, augmentation, dropout, and early stopping.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: Training loss is flat from the first step. What should be checked first? The conclusion is Verify labels and preprocessing, loss-input compatibility, nonzero gradients, parameter updates, learning rate, frozen layers, numerical scale, and a tiny-batch overfit test.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. Training loss is flat from the first step. What should be checked first?
Solution. Verify labels and preprocessing, loss-input compatibility, nonzero gradients, parameter updates, learning rate, frozen layers, numerical scale, and a tiny-batch overfit test.

Practice 2. Give a short oral explanation of optimization, regularization, and debugging that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct optimization, regularization, and debugging from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a optimization, regularization, and debugging problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 5 Recognition, transfer, and calibration

The idea

Classification estimates labels from visual evidence. Transfer learning reuses pretrained features through frozen or fine-tuned models; imbalance, long tails, shift, and calibration determine whether confidence supports a decision.

Rebuild the chapter from first principles

The claim I need to own is this: Classification estimates labels from visual evidence. Transfer learning reuses pretrained features through frozen or fine-tuned models; imbalance, long tails, shift, and calibration determine whether confidence supports a decision. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of recognition, transfer, and calibration. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach recognition, transfer, and calibration on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from optimization, regularization, and debugging to detection and segmentation. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Classification estimates labels from visual evidence.
- **Mechanism.** Transfer learning reuses pretrained features through frozen or fine-tuned models; imbalance, long tails, shift, and calibration determine whether confidence supports a decision.
- **Boundary.** The useful test is whether I can apply recognition, transfer, and calibration to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when recognition, transfer, and calibration is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

My shorthand for this chapter is: accuracy hides which classes fail and whether a 90 percent score deserves 90 percent trust.

The easy version

Classification estimates labels from visual evidence.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: When is a confusion matrix more useful than accuracy? The conclusion is When classes have different prevalence or costs, it exposes which labels are confused, minority-class failure, and the need for per-class precision and recall.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. When is a confusion matrix more useful than accuracy?
Solution. When classes have different prevalence or costs, it exposes which labels are confused, minority-class failure, and the need for per-class precision and recall.

Practice 2. Give a short oral explanation of recognition, transfer, and calibration that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct recognition, transfer, and calibration from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a recognition, transfer, and calibration problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 6 Detection and segmentation

The idea

Object detection predicts classes and boxes through proposals or dense heads, matching, nonmaximum suppression, and localization losses. Semantic, instance, and panoptic segmentation assign different pixel-level structures.

Rebuild the chapter from first principles

The claim I need to own is this: Object detection predicts classes and boxes through proposals or dense heads, matching, nonmaximum suppression, and localization losses. Semantic, instance, and panoptic segmentation assign different pixel-level structures. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed,

recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of detection and segmentation. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach detection and segmentation on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from recognition, transfer, and calibration to visual generation and representation learning. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method

valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Object detection predicts classes and boxes through proposals or dense heads, matching, nonmaximum suppression, and localization losses.
- **Mechanism.** Semantic, instance, and panoptic segmentation assign different pixel-level structures.
- **Boundary.** The useful test is whether I can apply detection and segmentation to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when detection and segmentation is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

A correct class with the wrong boundary can still be useless, so I match the metric to the spatial decision.

The easy version

Object detection predicts classes and boxes through proposals or dense heads, matching, nonmaximum suppression, and localization losses.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: What does intersection over union measure? The conclusion is It divides overlap area of predicted and true regions by their union. Thresholded IoU defines matched detections, while averaged IoU summarizes segmentation overlap.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. What does intersection over union measure?
Solution. It divides overlap area of predicted and true regions by their union. Thresholded IoU defines matched detections, while averaged IoU summarizes segmentation overlap.

Practice 2. Give a short oral explanation of detection and segmentation that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a

four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct detection and segmentation from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a detection and segmentation problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 7 Visual generation and representation learning

The idea

Autoencoders, variational autoencoders, adversarial networks, diffusion models, contrastive learning, and self-supervision learn visual representations or distributions with different objectives and sampling procedures.

Rebuild the chapter from first principles

The claim I need to own is this: Autoencoders, variational autoencoders, adversarial networks, diffusion models, contrastive learning, and self-supervision learn visual representations or distributions with different objectives and sampling procedures. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of visual generation and representation learning. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative

derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct procedure under the wrong assumptions is still a wrong answer.
- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach visual generation and representation learning on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from detection and segmentation to robustness, interpretation, and responsible deployment. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Autoencoders, variational autoencoders, adversarial networks, diffusion models, contrastive learning, and self-supervision learn visual representations or distributions with different objectives and sampling procedures.
- **Mechanism.** The useful test is whether I can apply visual generation and representation learning to a new case and explain the assumption that makes the answer valid.
- **Boundary.** The useful test is whether I can apply visual generation and representation learning to a new case and explain the assumption that makes the answer valid.

- **Decision test.** I should be able to say when visual generation and representation learning is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

My shorthand for this chapter is: a realistic generated image is not evidence that the model understood the world or respected its source data.

The easy version

Autoencoders, variational autoencoders, adversarial networks, diffusion models, contrastive learning, and self-supervision learn visual representations or distributions with different objectives and sampling procedures.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: What does a diffusion model learn during common training? The conclusion is It learns to predict added noise, a clean sample, or an equivalent score across noise levels. Sampling iteratively reverses a noise process to construct an image.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. What does a diffusion model learn during common training?
Solution. It learns to predict added noise, a clean sample, or an equivalent score across noise levels. Sampling iteratively reverses a noise process to construct an image.

Practice 2. Give a short oral explanation of visual generation and representation learning that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct visual generation and representation learning from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a visual generation and representation learning problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

CHAPTER 8 Robustness, interpretation, and responsible deployment

The idea

Vision models can depend on shortcuts, backgrounds, camera artifacts, demographic imbalance, adversarial perturbations, and spurious correlations. Deployment needs stress tests, uncertainty, monitoring, privacy, and fallback.

Rebuild the chapter from first principles

The claim I need to own is this: Vision models can depend on shortcuts, backgrounds, camera artifacts, demographic imbalance, adversarial perturbations, and spurious correlations. Deployment needs stress tests, uncertainty, monitoring, privacy, and fallback. I do not count that as learned just because the sentence looks familiar. I should be able to name the objects, state what changes and what stays fixed, recover the rule from the definitions, and explain why the result is allowed.

The move

My working sequence

1. **Translate.** Rewrite the prompt in the language of robustness, interpretation, and responsible deployment. List the known quantities, the unknown, the units or types, and any hidden constraint.
2. **Choose.** State the theorem, model, algorithm, accounting rule, or physical law before substituting values. This is where I make the assumption visible.
3. **Execute.** Work one justified step at a time and keep intermediate state readable enough that I can audit it later.
4. **Verify.** Check a boundary case, sign, unit, invariant, order of magnitude, or alternative derivation. Then translate the result back into an ordinary sentence.

Details I would refuse to skip

- **Definitions:** I should be able to define every technical noun in the chapter without using that same noun in the definition.
- **Assumptions:** I write the condition that makes the method legal beside the method itself. A correct

procedure under the wrong assumptions is still a wrong answer.

- **Representation:** I move freely among words, equations or code, a diagram, and a small numerical example. If one representation disagrees with another, that disagreement is useful evidence.
- **Failure modes:** I keep the nearest counterexample in mind: the boundary where the rule changes, the invariant breaks, or the model stops matching reality.
- **Communication:** My final answer names what the result means and what it does not establish. I do not let notation do the explanatory work for me.

Sanity check

Closed-note check. Can I teach robustness, interpretation, and responsible deployment on a blank page, derive or reconstruct its main rule, work a fresh example, and diagnose a deliberately broken solution? If I can only recognize the finished work, I am not done.

How this chapter connects

I read this chapter as the bridge from visual generation and representation learning to the cumulative course model. The earlier material supplies the language and constraints; this chapter adds a new reasoning move; the later material asks me to use that move without being told its name. That connection matters because exams and real projects mix chapters on purpose.

What I need to be able to do

For this chapter, I should be able to define the objects and state involved, state the governing invariant or contract, trace a small example without hand-waving, and explain which assumption makes the method valid. I should also be able to compare this model with the nearest alternative and say what failure would make me switch approaches.

Deep notes

- **Model.** Vision models can depend on shortcuts, backgrounds, camera artifacts, demographic imbalance, adversarial perturbations, and spurious correlations.
- **Mechanism.** Deployment needs stress tests, uncertainty, monitoring, privacy, and fallback.
- **Boundary.** The useful test is whether I can apply robustness, interpretation, and responsible deployment to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when robustness, interpretation, and responsible deployment is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I ask what visual cue the model could exploit more cheaply than the intended concept.

The easy version

Vision models can depend on shortcuts, backgrounds, camera artifacts, demographic imbalance, adversarial perturbations, and spurious correlations.

Worked example

A useful way to work this chapter is to make the smallest concrete instance, write the state or contract explicitly, and then scale the reasoning only after that instance behaves correctly. The worked question below makes that reasoning concrete: A classifier recognizes cows mainly from green pasture. How can this be reduced? The conclusion is Test counterfactual backgrounds and new scenes, inspect error slices, collect diverse environments, use targeted augmentation or objectives, and validate that performance follows the animal rather than the shortcut.

Common miss

The common mistake is to memorize the visible procedure while dropping its precondition. I write the assumption beside the method and test one boundary case before trusting the result.

Solved chapter practice

Practice. A classifier recognizes cows mainly from green pasture. How can this be reduced?
Solution. Test counterfactual backgrounds and new scenes, inspect error slices, collect diverse environments, use targeted augmentation or objectives, and validate that performance follows the animal rather than the shortcut.

Practice 2. Give a short oral explanation of robustness, interpretation, and responsible deployment that includes its model, one assumption, one failure mode, and one comparison with a neighboring method.
Solution. A strong answer begins with the model in the blue box, names the assumption that licenses the method, uses the common miss above as the failure mode, and chooses the comparison according to the decision test in the deep notes. The goal is not one memorized sentence; it is a four-part explanation I can reproduce on a new example.

Mastery practice A. Reconstruct robustness, interpretation, and responsible deployment from a blank page. Include the central definition, the governing rule, one required assumption, and one boundary case.
Solution. Start from the blue idea box, but rewrite it in my own words. Identify the objects and state, name the rule before using it, copy the relevant assumption from the details audit, and choose a boundary where that assumption becomes equality or fails. A complete solution explains why each part belongs; a list of vocabulary is not enough.

Mastery practice B. A classmate gets a polished-looking answer to a robustness, interpretation, and responsible deployment problem but never states a model or checks the result. Write the shortest useful review of that solution.
Solution. I would ask four concrete questions: What exactly are the inputs and desired output? Which rule licenses the main step? Which assumption could invalidate it? Which independent check supports the conclusion? If the answer cannot survive those four questions, I treat the numerical or verbal result as unverified.

Final study check

I should be able to close this packet and reconstruct the core models, not merely recognize their names. My final check is to explain one complete problem aloud: what the inputs mean, which invariant or contract controls the work, why the method is legal, what it costs, how it can fail, and what evidence would convince another engineer.