UCLA COMPUTER SCIENCE FIELD NOTES

CS / EC ENGR C147A

Neural Networks and Deep Learning

Backpropagation, optimization, regularization, CNNs, sequence models, generative models, and robust training

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	**Neural computation and maximum likelihood**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
2	**Backpropagation and computational graphs**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
3	**Optimization for deep networks**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
4	**Regularization and reliable experiments**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
5	**Convolutional networks**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
6	**Sequence models and attention**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
7	**Autoencoders and generative models**	Explain the model, apply it to a concrete case, and defend the relevant correctness or design decision.
8	**Robustness, debugging, and responsible use**	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/ECENGRC147A. 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 Neural computation and maximum likelihood

The idea

A neural network composes affine maps and nonlinear activations. Output parameterization and loss should match the likelihood model: softmax with cross-entropy for categorical outputs, for example.

Rebuild the chapter from first principles

The claim I need to own is this: A neural network composes affine maps and nonlinear activations. Output parameterization and loss should match the likelihood model: softmax with cross-entropy for categorical outputs, for example. 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 neural computation and maximum likelihood. 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 neural computation and maximum likelihood 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 backpropagation and computational graphs. 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.** A neural network composes affine maps and nonlinear activations.
- **Mechanism.** Output parameterization and loss should match the likelihood model: softmax with cross-entropy for categorical outputs, for example.
- **Boundary.** The useful test is whether I can apply neural computation and maximum likelihood to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when neural computation and maximum likelihood is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I track tensor shapes beside every layer; most implementation bugs reveal themselves as an impossible shape before they become math bugs.

The easy version

A neural network composes affine maps and nonlinear activations.

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 are stacked linear layers still linear? The conclusion is Their matrix products and biases collapse into one affine map. Nonlinear activations are what let depth represent nonlinear functions.

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 are stacked linear layers still linear?
Solution. Their matrix products and biases collapse into one affine map. Nonlinear activations are what let depth represent nonlinear functions.

Practice 2. Give a short oral explanation of neural computation and maximum likelihood 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 neural computation and maximum likelihood 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 neural computation and maximum likelihood 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 Backpropagation and computational graphs

The idea

Backpropagation applies the chain rule in reverse topological order, reusing intermediate derivatives. Automatic differentiation records operations, but correct training still requires understanding gradient flow, broadcasting, and reduction.

Rebuild the chapter from first principles

The claim I need to own is this: Backpropagation applies the chain rule in reverse topological order, reusing intermediate derivatives. Automatic differentiation records operations, but correct training still

requires understanding gradient flow, broadcasting, and reduction. 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 backpropagation and computational graphs. 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 backpropagation and computational graphs 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 neural computation and maximum likelihood to optimization for deep networks. 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.** Backpropagation applies the chain rule in reverse topological order, reusing intermediate derivatives.
- **Mechanism.** Automatic differentiation records operations, but correct training still requires understanding gradient flow, broadcasting, and reduction.
- **Boundary.** The useful test is whether I can apply backpropagation and computational graphs to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when backpropagation and computational graphs is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I derive one scalar path by hand before trusting autograd.

The easy version

Backpropagation applies the chain rule in reverse topological order, reusing intermediate derivatives.

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 accumulate gradients at a reused node? The conclusion is The output depends on that node through multiple paths. The multivariable chain rule adds each path's contribution to the total derivative.

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 accumulate gradients at a reused node?
Solution. The output depends on that node through multiple paths. The multivariable chain rule adds each path's contribution to the total derivative.

Practice 2. Give a short oral explanation of backpropagation and computational graphs 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 backpropagation and computational graphs 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 backpropagation and computational graphs 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 Optimization for deep networks

The idea

Stochastic gradient descent estimates the full gradient from minibatches. Momentum, adaptive methods, learning-rate schedules, normalization, and initialization shape training dynamics. Loss curves diagnose optimization but not generalization alone.

Rebuild the chapter from first principles

The claim I need to own is this: Stochastic gradient descent estimates the full gradient from minibatches. Momentum, adaptive methods, learning-rate schedules, normalization, and initialization shape training dynamics. Loss curves diagnose optimization but not generalization alone. 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 for deep networks. 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 for deep networks 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 backpropagation and computational graphs to regularization and reliable experiments. 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.** Stochastic gradient descent estimates the full gradient from minibatches.
- **Mechanism.** Momentum, adaptive methods, learning-rate schedules, normalization, and initialization shape training dynamics.
- **Boundary.** Loss curves diagnose optimization but not generalization alone.

- **Decision test.** I should be able to say when optimization for deep networks is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

When training fails, I first try to overfit one tiny batch; if that fails, the pipeline or gradient is broken.

The easy version

Stochastic gradient descent estimates the full gradient from minibatches.

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 too-large learning rate diverge? The conclusion is Updates overshoot useful regions, causing oscillating or exploding loss. Reduce the rate, inspect gradient scale, and verify normalization and initialization.

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 too-large learning rate diverge?
Solution. Updates overshoot useful regions, causing oscillating or exploding loss. Reduce the rate, inspect gradient scale, and verify normalization and initialization.

Practice 2. Give a short oral explanation of optimization for deep networks 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 for deep networks 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 for deep networks 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 Regularization and reliable experiments

The idea

Weight decay, data augmentation, dropout, early stopping, and architecture constraints limit overfitting in different ways. Reproducible comparisons control seeds, splits, preprocessing, budgets, and evaluation mode.

Rebuild the chapter from first principles

The claim I need to own is this: Weight decay, data augmentation, dropout, early stopping, and architecture constraints limit overfitting in different ways. Reproducible comparisons control seeds, splits, preprocessing, budgets, and evaluation mode. 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 regularization and reliable experiments. 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 regularization and reliable experiments 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 for deep networks to convolutional networks. 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.** Weight decay, data augmentation, dropout, early stopping, and architecture constraints limit overfitting in different ways.
- **Mechanism.** Reproducible comparisons control seeds, splits, preprocessing, budgets, and evaluation mode.
- **Boundary.** The useful test is whether I can apply regularization and reliable experiments to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when regularization and reliable experiments is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I change one experimental variable at a time and log the exact configuration; otherwise improvement stories are fiction.

The easy version

Weight decay, data augmentation, dropout, early stopping, and architecture constraints limit overfitting in different ways.

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 must dropout be disabled for evaluation? The conclusion is Training dropout samples random subnetworks. Evaluation uses the full network with the learned scaling so predictions are deterministic and match the ensemble approximation.

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 must dropout be disabled for evaluation?
Solution. Training dropout samples random subnetworks. Evaluation uses the full network with the learned scaling so predictions are deterministic and match the ensemble approximation.

Practice 2. Give a short oral explanation of regularization and reliable experiments 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 regularization and reliable experiments 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 regularization and reliable experiments 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 Convolutional networks

The idea

Convolutions share local filters across spatial positions, encoding translation structure and reducing parameters. Receptive field, stride, padding, pooling, residual connections, and normalization determine information flow.

Rebuild the chapter from first principles

The claim I need to own is this: Convolutions share local filters across spatial positions, encoding translation structure and reducing parameters. Receptive field, stride, padding, pooling, residual connections, and normalization determine information flow. 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 convolutional networks. 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 convolutional networks 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 regularization and reliable experiments to sequence models and attention. 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.** Convolutions share local filters across spatial positions, encoding translation structure and reducing parameters.
- **Mechanism.** Receptive field, stride, padding, pooling, residual connections, and normalization determine information flow.
- **Boundary.** The useful test is whether I can apply convolutional networks to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when convolutional networks is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I calculate spatial dimensions layer by layer instead of discovering a mismatch at the classifier head.

The easy version

Convolutions share local filters across spatial positions, encoding translation structure and reducing parameters.

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: Compute output width for kernel k, padding

p, stride s. The conclusion is Use floor((W+2p-k)/s)+1. Dilation modifies effective kernel width and must be included when present.

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. Compute output width for kernel k, padding p, stride s.
Solution. Use floor((W+2p-k)/s)+1. Dilation modifies effective kernel width and must be included when present.

Practice 2. Give a short oral explanation of convolutional networks 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 convolutional networks 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 convolutional networks 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 Sequence models and attention

The idea

Recurrent networks share state transitions through time but can suffer vanishing and exploding gradients; gated units create controlled memory paths. Attention computes content-dependent weighted combinations and supports direct long-range interactions.

Rebuild the chapter from first principles

The claim I need to own is this: Recurrent networks share state transitions through time but can suffer vanishing and exploding gradients; gated units create controlled memory paths. Attention computes content-dependent weighted combinations and supports direct long-range interactions. 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 sequence models and attention. 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 sequence models and attention 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 convolutional networks to autoencoders and generative models. 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.** Recurrent networks share state transitions through time but can suffer vanishing and exploding gradients; gated units create controlled memory paths.
- **Mechanism.** Attention computes content-dependent weighted combinations and supports direct long-range interactions.
- **Boundary.** The useful test is whether I can apply sequence models and attention to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when sequence models and attention is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I distinguish sequence length, feature width, and batch axis before reading any attention equation.

The easy version

Recurrent networks share state transitions through time but can suffer vanishing and exploding gradients; gated units create controlled memory paths.

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 does self-attention cost quadratically in sequence length? The conclusion is It forms pairwise query-key scores for every token pair, producing an n by n matrix before weighting values.

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 does self-attention cost quadratically in sequence length?
Solution. It forms pairwise query-key scores for every token pair, producing an n by n matrix

before weighting values.

Practice 2. Give a short oral explanation of sequence models and attention 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 sequence models and attention 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 sequence models and attention 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 Autoencoders and generative models

The idea

Autoencoders learn compressed representations through reconstruction. Variational autoencoders optimize a reconstruction-plus-KL lower bound; GANs train a generator against a discriminator and may suffer instability or mode collapse.

Rebuild the chapter from first principles

The claim I need to own is this: Autoencoders learn compressed representations through reconstruction. Variational autoencoders optimize a reconstruction-plus-KL lower bound; GANs train a generator against a discriminator and may suffer instability or mode collapse. 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 autoencoders and generative models. 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 autoencoders and generative models 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 sequence models and attention to robustness, debugging, and responsible use. 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 learn compressed representations through reconstruction.
- **Mechanism.** Variational autoencoders optimize a reconstruction-plus-KL lower bound; GANs train a generator against a discriminator and may suffer instability or mode collapse.
- **Boundary.** The useful test is whether I can apply autoencoders and generative models to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when autoencoders and generative models is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

A generative loss is a proxy; I inspect samples, coverage, and failure modes rather than trusting one scalar.

The easy version

Autoencoders learn compressed representations through reconstruction.

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 the VAE reparameterization trick accomplish? The conclusion is It writes a random latent sample as mean plus scale times fixed noise, allowing gradients to flow through distribution parameters.

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 the VAE reparameterization trick accomplish?
Solution. It writes a random latent sample as mean plus scale times fixed noise, allowing gradients to flow through distribution parameters.

Practice 2. Give a short oral explanation of autoencoders and generative models 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 autoencoders and generative models 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 autoencoders and generative models 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, debugging, and responsible use

The idea

Adversarial examples, distribution shift, spurious correlations, privacy leakage, and unequal subgroup behavior expose gaps between benchmark accuracy and dependable use. Debugging needs baselines, ablations, data inspection, and uncertainty.

Rebuild the chapter from first principles

The claim I need to own is this: Adversarial examples, distribution shift, spurious correlations, privacy leakage, and unequal subgroup behavior expose gaps between benchmark accuracy and dependable use. Debugging needs baselines, ablations, data inspection, and uncertainty. 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, debugging, and responsible use. 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, debugging, and responsible use 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 autoencoders and generative models 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.** Adversarial examples, distribution shift, spurious correlations, privacy leakage, and unequal subgroup behavior expose gaps between benchmark accuracy and dependable use.
- **Mechanism.** Debugging needs baselines, ablations, data inspection, and uncertainty.
- **Boundary.** The useful test is whether I can apply robustness, debugging, and responsible use to a new case and explain the assumption that makes the answer valid.
- **Decision test.** I should be able to say when robustness, debugging, and responsible use is the right model, when its assumptions fail, and what evidence would change my choice.

How I actually think about it

I assume the model will exploit the easiest signal available, including one I never intended to provide.

The easy version

Adversarial examples, distribution shift, spurious correlations, privacy leakage, and unequal subgroup behavior expose gaps between benchmark accuracy and dependable use.

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: How can label leakage be detected? The conclusion is Trace feature provenance, test suspicious near-perfect features, compare time-aware splits, remove candidates in ablations, and evaluate on a distribution where the leaked shortcut is unavailable.

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. How can label leakage be detected?
Solution. Trace feature provenance, test suspicious near-perfect features, compare time-aware splits, remove candidates in ablations, and evaluate on a distribution where the leaked shortcut is unavailable.

Practice 2. Give a short oral explanation of robustness, debugging, and responsible use 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, debugging, and responsible use 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, debugging, and responsible use 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.