ARTIFICIAL INTELLIGENCE AND NEURAL NETWORKS

Lecture 3b – Hidden Layers and Backpropagation

Chizhi Chris ZHANG

zhangchizhi@ciomp.ac.cn

Advanced Computing and Digital Technology Research Center

University of Chinese Academy of Sciences

Spring 2026

Today's Question

What we are trying to answer

How does a neural network learn useful internal features instead of only drawing one final boundary on the raw input?

Why this lecture matters

NN2 showed what one layer can do. NN3 explains why adding hidden layers changes the game and how those layers are trained together.

What changes from NN2

The story is no longer only about one score and one boundary. It is now about representation and training.

From NN2 to NN3

Last time

We deliberately stayed with one layer so the geometry was easy to see: one score, one threshold, one boundary.

Today

We let the network build intermediate features for itself and then learn those features together with the final decision.

One simple sentence

We are moving from “draw one boundary” to “build a better internal view first”.

One Hidden Layer Changes the Story

Without hidden layers

The output unit must work directly on the original variables, even if those variables are awkward for the task.

With hidden layers

The network can combine simple clues into intermediate patterns and pass those patterns forward.

The gain is not just “more numbers.” The gain is a new internal representation that can make the final task easier.

Why Modern AI Still Depends on This

Across many domains

Image models combine edges into parts and parts into objects. Language models combine local token relations into larger context. Recommendation systems combine many weak signals into a useful user state.

What stays the same

The details change from field to field, but the basic idea is the same: useful layers build more useful representations.

From Boundary to Representation

NN2 intuition

The last layer draws a boundary in the space it receives.

NN3 intuition

Hidden layers try to deliver a better space to that last layer, one in which the final separation becomes easier.

A picture in words

If the raw input space is tangled, hidden layers can stretch, rotate, and regroup it until the final decision becomes simpler.

XOR Is the Reminder, Not the Whole Lecture

What XOR already taught us

One straight boundary is not enough for every pattern. Some problems need the model to break the task into smaller internal steps.

Why we do not stay there today

NN2 already made the geometric point. NN3 asks the next question: if hidden units are useful, how are they actually learned?

Layer-by-Layer Computation

One hidden layer

$$h = g\left(W^{(1)}x + b^{(1)}\right)$$

Output layer

$$\hat{y} = f\left(W^{(2)}h + b^{(2)}\right)$$

How to read it

Each layer takes the current representation, transforms it, and hands a new representation to the next layer.

Why Activation Changes Everything

Without activation

Stacking linear layers still collapses into one larger linear transformation. Depth alone does not buy you a new kind of behavior.

With activation

Nonlinearity lets the network bend, gate, and reshape the representation between layers.

The key point

The real turning point is not “more layers.” It is “more layers with nonlinearity.”

Forward Pass Builds the Current Answer

What happens in the forward pass

The input moves through the network, and each layer produces a more processed version of the signal until the output layer gives a prediction.

A classroom metaphor

It is like a team editing a draft. Each person rewrites the current draft into something more useful for the next person.

This is why the final answer depends on many earlier transforms, not on one magical jump at the end.

Hidden Units Are Learned Features

What a hidden unit is not

It does not need to match a neat human label such as “ear”, “verb”, or “trustworthy”.

What it can still do

It can capture a pattern that repeatedly helps later prediction, even if that pattern is hard for us to name cleanly.

Why this matters

The network is not limited to the hand-designed features we happened to think of first.

Output Layers Follow the Task

Classification output

$$\hat{y} = \mathrm{softmax}(Wh + b)$$

What it means

The network turns the learned representation into class scores or probabilities.

Regression output

$$\hat{y} = w^{\top} h + b$$

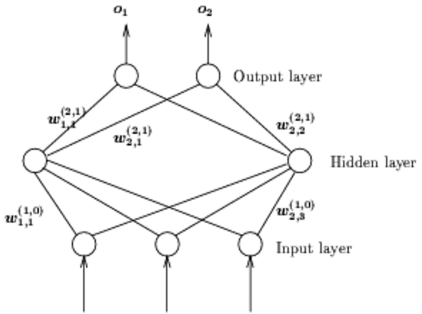
What it means

The same hidden-feature idea can also feed a numerical prediction rather than a category.

One lesson

The hidden layers may be similar, but the last step still depends on what kind of answer the task needs.

Backpropagation in One Picture



Forward

The network produces a prediction by moving information from input to output.

Backward

The correction signal then moves back through the network so each weight learns how it contributed to the final error.

The Chain Rule in Plain Language

Compact form

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial h} \frac{\partial h}{\partial w}$$

How to read it

If the loss depends on the output, and the output depends on hidden activations, and those activations depend on earlier weights, then earlier weights can still be updated through that chain.

The practical meaning

Backpropagation keeps asking: if this earlier weight changed a little, how would the final loss change after all later steps react?

Credit Assignment Is the Hard Part

The real difficulty

When the final answer is wrong, which hidden unit deserves how much blame, and which early weight should move most?

What backpropagation solves

It turns one final error into many local update signals that are consistent with the whole network.

Why one output error is not enough

The network may have many layers and many paths. A good update must distribute responsibility instead of changing everything blindly.

Training Is a Repeated Cycle

One pass through the loop

- make a prediction
- measure the loss
- send the correction backward
- update the parameters
- try again

One sentence version

The network makes a guess, gets criticized, learns from the criticism, and tries again.

Gradient Signals Must Survive the Trip

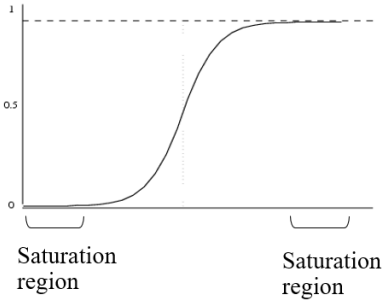
What can go wrong

As the signal moves backward through many layers, it can shrink too much, explode too much, or become noisy enough that learning slows or becomes unstable.

Why depth gets harder

Depth gives more expressive power, but it also makes optimization a harder coordination problem.

Why Saturation Slows Learning



What the figure shows

Near the flat ends of the sigmoid curve, changing the input no longer changes the output very much.

Why that hurts training

If a unit sits in a saturated region, the backward correction becomes weak and earlier layers learn slowly.

Generalization Is Still the Real Target

The old lesson returns

A larger network can reduce training error impressively and still become worse on new data.

A neural-network version

The network may learn a very detailed internal code for the training set and still fail when the next batch looks a little different.

What matters most

Representation power is useful only if it carries forward into honest performance beyond the training set.

Neural Networks Are Not Only for Classification

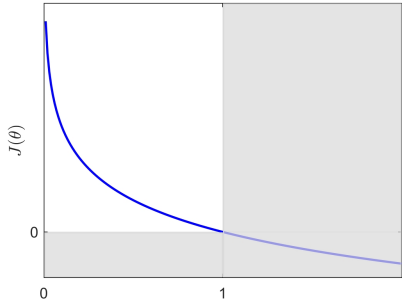
A common misunderstanding

Students often meet neural networks through digit recognition or image labels and then assume the method is mainly about categories.

The broader reality

The same hidden-layer idea also supports continuous prediction, ranking, recommendation, forecasting, and representation learning for later tasks.

Validation and Early Stopping Help



What the curve says

Training error may keep falling while test error begins to rise. That is the point where more fitting stops helping.

Why early stopping works

We stop near the point where held-out performance is best instead of chasing a prettier training curve.

Architecture and Optimization Are Different Questions

Architecture question

Does the model family have enough representational power for the task?

Optimization question

Can we actually train that model stably with the data, loss, initialization, and learning schedule we chose?

Why students should separate them

Poor results do not automatically mean the idea was wrong. They may mean the training process never reached a good solution.

A Debugging Order That Saves Time

Start outside the model

Check data quality, label quality, train-validation split, and obvious preprocessing mistakes before changing the network.

Then move inward

Only after that do learning rate, depth, width, regularization, and activation choices become the right questions.

Many frustrating neural failures are not deep mysteries. They are ordinary pipeline problems wearing a modern name.

What Useful Internal Features Feel Like

How you know the features are helping

Later layers should make the final task look simpler than it looked in the raw input space.

A plain example

In speech, a useful representation may separate speaker identity from spoken content. In recommendation, it may separate long-term taste from short-term mood.

The real win

The best hidden features make downstream decisions easier, not more mysterious.

Width and Depth Help in Different Ways

More width

Adding more units to one layer gives the model more parallel features at the same stage of abstraction.

More depth

Adding more layers gives the model a longer chain of transformations, where later features depend on earlier learned features.

Why this distinction matters

“Bigger” is not one thing. Different kinds of size help in different ways and create different training problems.

More Layers Do Not Cancel Bad Data

What depth cannot do

It cannot repair misleading labels, missing context, dishonest splits, or a task definition that never made sense.

Why this repeats AI2

Powerful models still learn from the examples we gave them, not from the world directly.

Two Common Myths About Backpropagation

Myth 1

"Backpropagation means the network reasons backward like a person."

Better reading

It is a calculus-based update rule for distributing error through a computational graph.

Myth 2

"If a network is deep enough, training details barely matter."

Better reading

Depth can increase expressive power, but poor optimization can still keep that power out of reach.

Fragility Is Not the Same as Failure

What fragility means

Training a multilayer model can be sensitive to initialization, scaling, activation choice, optimization settings, and data quality.

What this does not mean

It does not mean representation learning was a bad idea. It means the idea needs careful machinery to work well.

Much of later neural-network progress came from making deep training more stable, not from abandoning the basic principle.

Useful Representation Is the Real Target

What students should remember

The goal is not to stack layers because layers sound advanced. The goal is to create internal features that make the task easier, more robust, or more transferable.

The closing idea

When a neural network works well, what we are praising is not depth by itself. We are praising the representation that depth made possible.

Why AI3 and NN3 Belong Together

What AI3 established

Prediction, residuals, loss, validation, and uncertainty gave us a mature language for what a model output means.

What NN3 adds

Neural networks give one answer to the question of how a model can build richer internal structure before making that prediction.

Why NN4 Follows Naturally

What this lecture opened

We introduced hidden layers and backpropagation, but only at the level needed to understand the basic mechanism.

What NN4 will do

The next lecture looks more directly at activation behavior, training stability, and the design choices that make multilayer models usable.

What Later Neural Lectures Add

What NN3 gave you

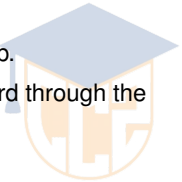
Representation, forward pass, backward correction, training fragility, and the link between hidden layers and useful features.

What later lectures specialize

Better activations, more stable training, architectures for different data types, and the path from early multilayer nets to today's large-scale systems.

Summary

- Hidden layers matter because they let the network build internal features before making the final prediction.
- Nonlinearity is what makes depth useful instead of collapsing back into one linear map.
- Backpropagation trains the whole stack by sending responsibility for the error backward through the network.
- Training success still depends on optimization, validation, and data quality.
- The real goal is useful representation, not depth for its own sake.



Next Time

Where this story goes next

NN4 stays inside the neural-network family and looks more closely at activation behavior, training stability, and the practical choices that make deep models usable.

Carry this question with you

If hidden layers are useful because they reshape representation, which design choices help that reshaping stay trainable instead of collapsing into instability?

Thank You