

# Lesson 36: Overfitting Prevention

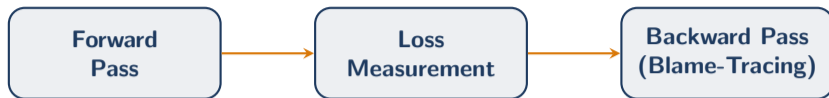
Data Science with Python – BSc Course

Data Science Program

BSc Course

45 Minutes

## Previously on L35...



We learned how networks train. But what if training goes **TOO well**?

---

L35 taught learning. L36 teaches when to **STOP** learning.

# The Suspicious 99%

**Scenario:** Your model gets 99% accuracy on training data but 60% on new data. The CEO wants to deploy it tomorrow. What do you tell them?

**After this lesson, you will be able to:**

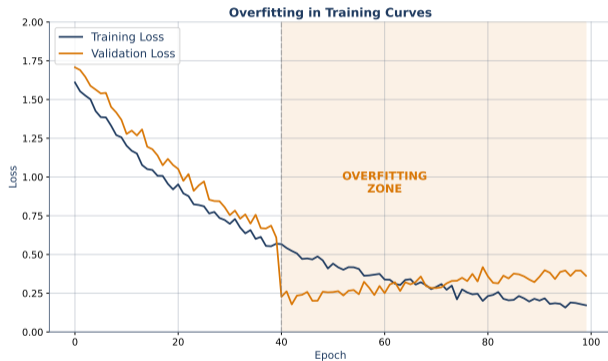
- Recognize overfitting in learning curves
- Apply dropout to prevent co-adaptation
- Use early stopping to halt at the right time
- Apply L2 regularization to control weight magnitude

---

Is 99% training accuracy always good news?

# Overfitting: When Perfect Is Bad

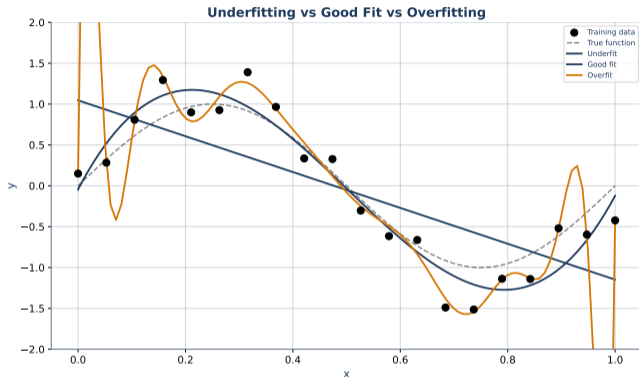
The model is **MEMORIZING** the training data instead of learning general patterns. Like a student who memorizes answers but can't solve new problems.



---

**Overfitting = memorizing noise instead of learning signal**

# Underfitting, Good Fit, Overfitting



Too simple (underfitting) — Just right (good fit) — Too complex (overfitting).

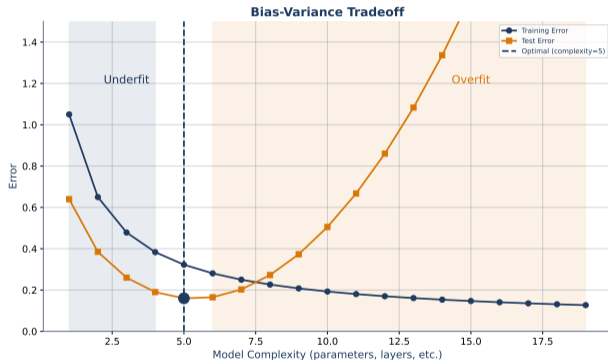
---

The Goldilocks zone: complex enough to learn, simple enough to generalize

# Bias-Variance: The Fundamental Tradeoff

**High bias** = underfitting. **High variance** = overfitting.

The sweet spot is in the middle.



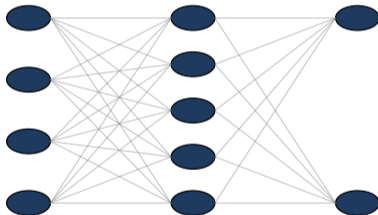
Three weapons ahead: Dropout, Early Stopping, L2 Regularization

# Weapon 1: Dropout

During training, randomly set some neurons to zero. Each batch sees a DIFFERENT sub-network.

Like studying with random pages missing—you learn the core concepts, not specific page numbers.

Standard Network (No Dropout)

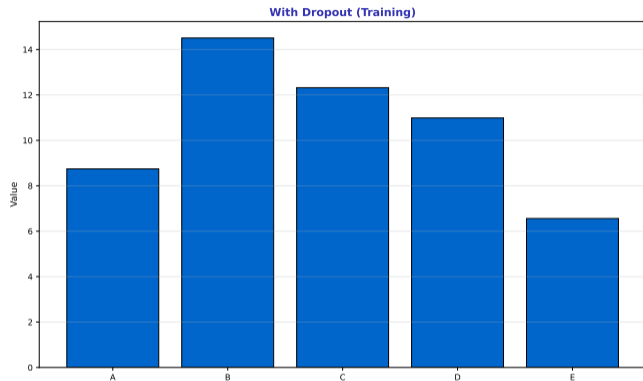


All neurons active, all connections used

---

Dropout forces the network to learn redundant, robust representations

# Dropout in Action



Different neurons dropped each batch—the network can't rely on any single path. It **MUST** learn robust features.

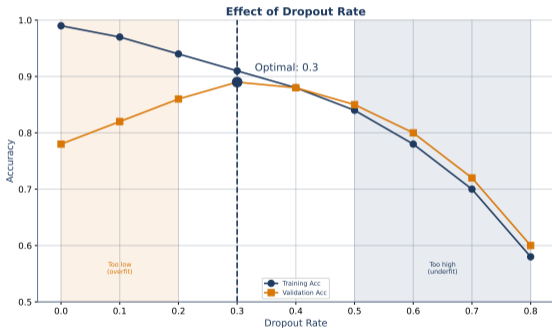
---

Each training step uses a different thinned network

# How Much Dropout?

**Typical rates:** 0.2–0.5 for hidden layers.

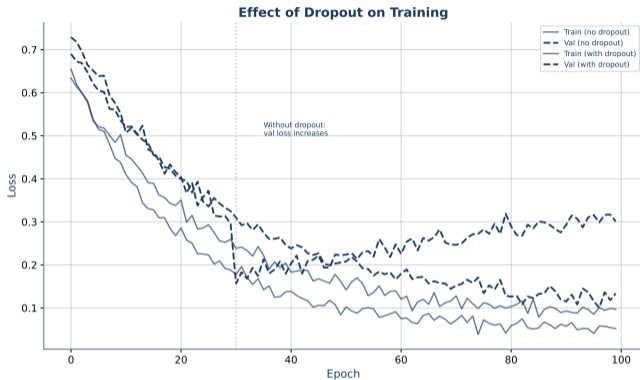
**Keras:** Dropout(0.3) between Dense layers. During INFERENCE, all neurons active—dropout only during training.



---

Keras handles train/inference modes automatically

# Dropout's Effect on Curves



With dropout: training is slower but validation improves. The gap closes. You trade training speed for generalization. Always a good trade.

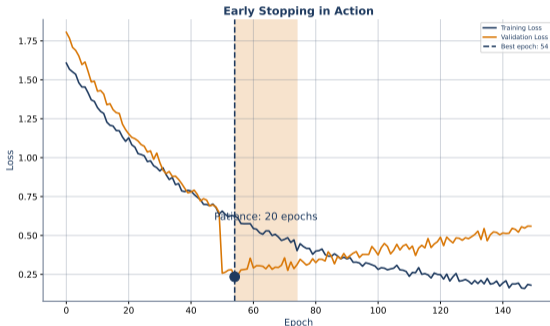
---

Slower training + better generalization = exactly what we want

# Weapon 2: Early Stopping

Set epochs high (500), but **STOP** when validation loss stops improving.

`EarlyStopping(patience=20, restore_best_weights=True)`

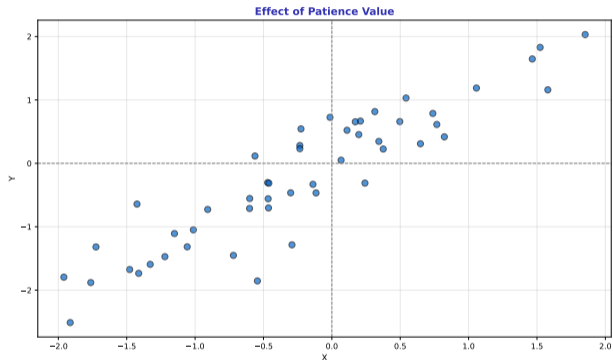


---

Best model = valley of the validation curve, not the last epoch

# Patience: How Long to Wait?

Low patience = stops too early. High patience = waits too long.



10–30 is typical. Always use `restore_best_weights=True`.

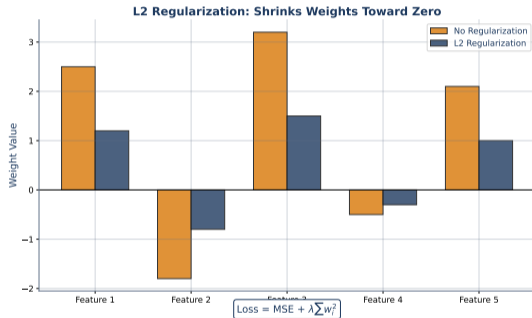
---

`patience=20` is a safe starting point for most problems

# Weapon 3: L2 Regularization

Add penalty for large weights to loss.

Keras: `kernel_regularizer=l2(0.001)`



Same idea as Ridge from L22—large weights = too much faith in one feature.

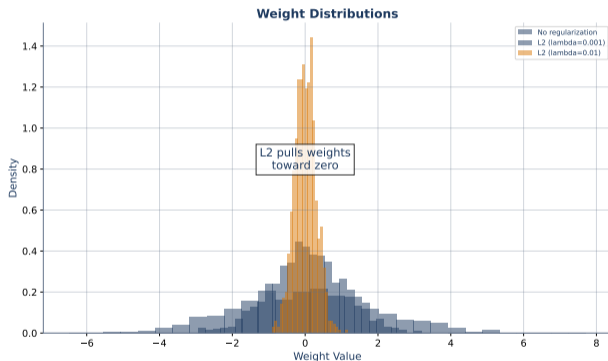
---

L2 in neural networks = Ridge regression for deep learning

# L2 Effect on Weights

**Without L2:** weights spread out, some extreme.

**With L2:** concentrated near zero.



---

Smaller weights → simpler model → better generalization

# Diagnostic Guide: Reading Learning Curves

**Both high, flat**

Underfitting

Fix: add layers/neurons

**Train low, val high**

Overfitting

Fix: dropout/early stop/L2

**Both low, small gap**

Good Fit!

Done – maybe fine-tune

**Both noisy**

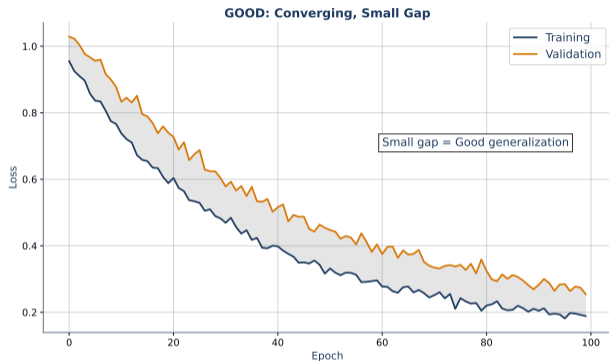
Bad LR or data

Fix: lower LR, larger batch

---

Print this guide. You will use it every time you train a neural network.

# Good Training: Converging, Small Gap

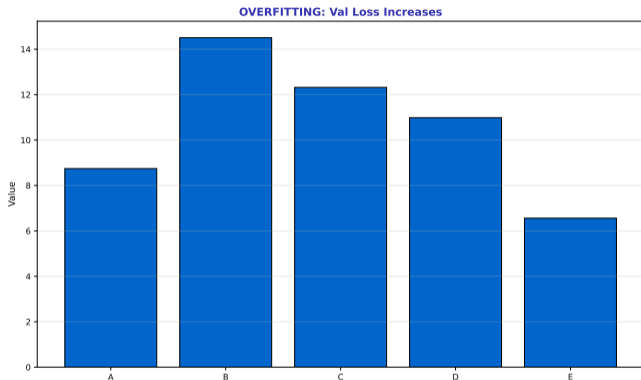


This is what you're LOOKING for: both curves converge, small gap.

---

Goal: train and validation curves converge to similar low values

# Discovery: What's Wrong Here?



**What's happening?** Which weapon(s) would you use? How can you tell it's overfitting and not underfitting?

---

Diagnose first, then prescribe. Which weapons would you deploy? (Pause 15s)

# The Complete Keras Recipe

**Put ALL weapons together in one model:**

```
Dense(64, 'relu') → Dropout(0.3) → Dense(32, 'relu') → Dropout(0.2) →  
Dense(1, 'sigmoid')
```

**Compile:** optimizer='adam', loss='binary\_crossentropy'

**Callbacks:**

- EarlyStopping(patience=20, restore\_best\_weights=True)
- ReduceLROnPlateau(patience=10, factor=0.5)

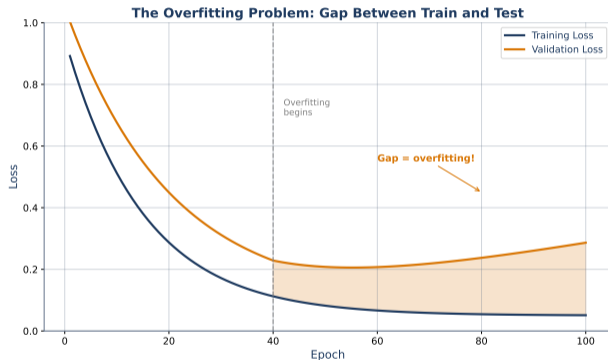
Set epochs=500 and let callbacks decide when to stop.

---

**The recipe: dropout layers + early stopping callback + L2 optional**

# Finance: Fitting to Noise

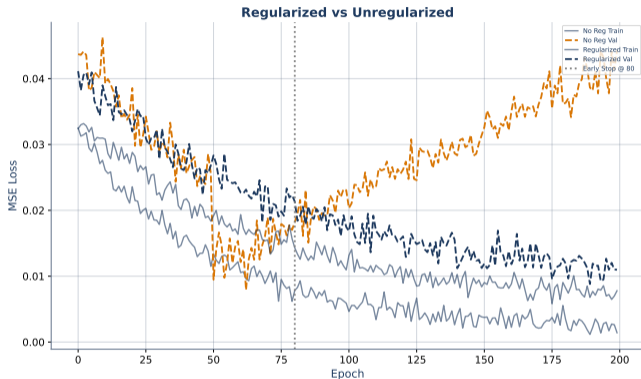
Financial data is the **HARDEST** case for overfitting: noisy, non-stationary, limited samples. Regularization isn't optional—it's mandatory.



---

Financial ML mantra: regularize aggressively, validate relentlessly

# Regularized vs Unregularized



Dropout + early stopping + small network = your finance ML starter kit.

---

Simpler models often outperform complex ones on financial data

# Hands-On Exercise (25 min)

## Task: Compare Regularization Techniques

1. Create overfit-prone dataset (200 samples, 50 features)
2. Train baseline MLP—observe overfitting in curves
3. Add Dropout(0.5)—compare curves
4. Add EarlyStopping—when does it stop?
5. Compare test accuracy across all variants in a table

---

**Extension: Try combining dropout + L2 + early stopping**

# Module 7 Complete: Your Deep Learning Toolkit

Module 7: Complete!



Neuron (L33)    Layers (L34)    Learning (L35)    Regularize (L36)

**Deep Learning Toolkit**

*Always check your validation loss!*

---

Four lessons, one complete toolkit. You're ready for deep learning.

# Key Takeaways

1. **Overfitting** = memorizing noise, not learning patterns
2. **Weapon 1: Dropout**—random neuron removal during training
3. **Weapon 2: Early stopping**—stop at the valley of val loss
4. **Weapon 3: L2**—penalize extreme weights (Ridge for neural networks)
5. **Finance:** regularize aggressively—noisy data demands it

---

Three weapons, one goal: close the gap between train and validation

# What's Next: Module 8 — Teaching Machines to Read

Neural networks process numbers. But finance is full of TEXT: earnings calls, analyst reports, news headlines.

Module 8 teaches machines to read.

**L37: Text Preprocessing**—turning words into numbers.

---

Next module: NLP—from text to predictions