Fraud Detection with Neural Networks

Detecting Fraudulent Transactions using Deep Learning

Topics

* Handling Imbalanced Data with SMOTE
* Neural Network Architectures Comparison
* Evaluation Metrics: Precision, Recall, AUC
* Early Stopping and Regularization

Problem Statement

The Challenge

Detect

* 98% legitimate transactions
* Only 2% fraud

Why Sta

SMOTE - Handling Imbalance

Synthetic Minority Over-sampling Technique

Before SMOTE	After SMOTE
100 Fraud	10000 Fraud
10000 Normal	10000 Normal

How SMOTE Works

* Pick fraud point A
* Find nearest fraud neighbor B
* Create new point on line A-B

CRITIC

Neural Network Architectures

Four Models Compared

Model	Architecture	Key Feature
ShallowWide	30-64-32-1	Few layers
DeepNarrow	30-32-32-32-32-1	Many layers
Hybrid	ReLU + Tanh	Mixed activations
Custom	BatchNorm + Dropout	Regularization

Key Concepts

Activation Functions

* ReLU: Hidden layers (no vanishing gradient)
* Sigmoid: Output layer (probability 0-1)

Regular

* Dropout: Randomly disable neurons
* BatchNorm: Normalize layer inputs
* Early Stopping: Stop when validation loss increases

Evaluation Metrics

Why Not Accuracy?

With 99

* Precision: Of predicted fraud, how many are real?
* Recall: Of real fraud, how many caught?
* AUC: Overall discrimination ability (0.5=random, 1.0=perfect)

Results Summary

Model Performance

Model	AUC	Notes
ShallowWide	0.95	Simple, fast
DeepNarrow	0.94	More depth
Hybrid	0.95	Mixed activations
Custom	0.96	Best - regularization

Winner: Custom model with BatchNorm + Dropout

Interview Key Points

Top Questions

* SMOTE: Only training data, never test
* Accuracy misleading for imbalanced data
* ReLU for hidden, Sigmoid for binary output
* Precision vs Recall trade-off

Key Nu

* ReLU derivative: 1 for positive
* Sigmoid range: (0, 1)
* Dropout: 0.2-0.5 typical

Conclusion

Key Learnings

* Always check class distribution first
* SMOTE balances training data effectively
* Regularization (Dropout, BatchNorm) improves generalization
* Early stopping prevents overfitting

Recomm

* Use Precision, Recall, AUC - not accuracy
* Custom model with regularization performs best