Modell Verbesserungen

**Daten Input**

* Klassengleichgewicht verbessern (Klasse 1 ist unterrepräsentiert) durch gewichtete Cross Entropy Loss Funktion (nn.CrossEntropyLoss(weight=...) mit z. B. [1.0, 10.0] – oder automatisch berechnet mit sklearn.utils.class\_weight.compute\_class\_weight.) oder SMOTE (für Over-/Undersampling) oder Focal Loss statt Cross Entropy
* *Möglich*: Zusätzliche Input Kanäle (multivariate Zeitreihen?)
* Statistiken der Zeitreihe als Input
* Sliding Windows/Rolling Features
* Positionelle Informationen (Schadenszeitpunkt?)

**Modell Architektur**

* Residual Connections (ResNet)
* Bidirektionale RNN-Schicht (CNN für lokale Features, RNN für Sequenzmodellierung)
* Global Max Pooling und Average Pooling kombinieren statt nur torch.mean()

**Validierung**

* Precision, Recall, F1 pro Klasse plotten → besonders wichtig bei Imbalance.
* ROC-AUC Score berechnen.
* Test & Validation Loss über Epochen loggen → frühes Overfitting erkennen

Feature scaling optimieren:

Statt

scaler = StandardScaler()

X = scaler.fit\_transform(X)

Besser

scaler = StandardScaler()

X\_trainval = scaler.fit\_transform(X\_trainval)

X\_test = scaler.transform(X\_test)

*mit*

class\_weights = torch.tensor([1.0, 5615 / 704], dtype=torch.float32) -> *Bedeutung*?  
criterion = nn.CrossEntropyLoss(weight=class\_weights)  
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)