

Transfer Learning for LSTM-Based Anomaly Detection

CYSE 420 — Assignment 07

Carl Lochstampfor

Setup

No pre-trained LSTM checkpoint exists for cybersecurity sequence data, so one was created. The NSL-KDD intrusion dataset was split by transport protocol into two domains that share a feature space but not a distribution: a source domain of 20,526 TCP records and a target domain of 4,666 UDP and ICMP records. This is the domain-shift case described in section 7.3.1 — the same task, a different $P(X)$. Records were windowed into sequences of ten consecutive connections labelled by the last, giving the many-to-one form from 7.2.2. Splits are positional rather than shuffled, because overlapping windows would otherwise leak between training and test data, and the scaler was fit on source training rows only so that the shift was preserved rather than normalized away.

How transfer learning was applied

An LSTM with 64 units and 27,969 parameters was pre-trained on 14,359 source windows, reaching 0.9997 ROC-AUC on source test data, and saved as `lstm_source.keras`. It was then reloaded and adapted in two phases, following 7.3.3. First, the `lstm_encoder` layer was frozen and a fresh classification head attached, leaving 2,113 of 27,969 parameters trainable; only the head learned, preserving the encoder's representation. Second, the encoder was unfrozen and training continued at a learning rate of $1e-4$, ten times lower, so that gradients from the new head could not destroy the pre-trained weights. Both phases used only 1,200 target windows, a deliberately reduced training set representing scarce labels.

Comparison

A structurally identical model trained from random initialization on the same 1,200 windows, with a larger epoch budget, served as the control. The transfer model is nominally ahead, but the 0.0043 accuracy difference is three windows out of 691. Retraining the scratch model under three different random seeds moved its accuracy by the same three windows, so the difference falls within run-to-run noise. On accuracy the two models are equivalent. They differ in cost. The scratch model converged in 25 epochs and 4.4 seconds; the transfer model required 183 epochs and 24.9 seconds to reach the same result, 5.7 times longer. Section 7.3.4 lists reduced training time as transfer learning's first benefit; here it was a penalty.

Model	Accuracy	Precision	Recall	F1	ROC-AUC	Epochs	Time (s)
Transfer (pre-trained)	0.9508	0.9204	0.9603	0.9399	0.9917	183	24.9
Scratch (random init)	0.9465	0.9167	0.9531	0.9345	0.9915	25	4.4

Table 1. Both models evaluated on the same 691 held-out target windows.

Transfer vs. scratch on the target domain (UDP / ICMP)

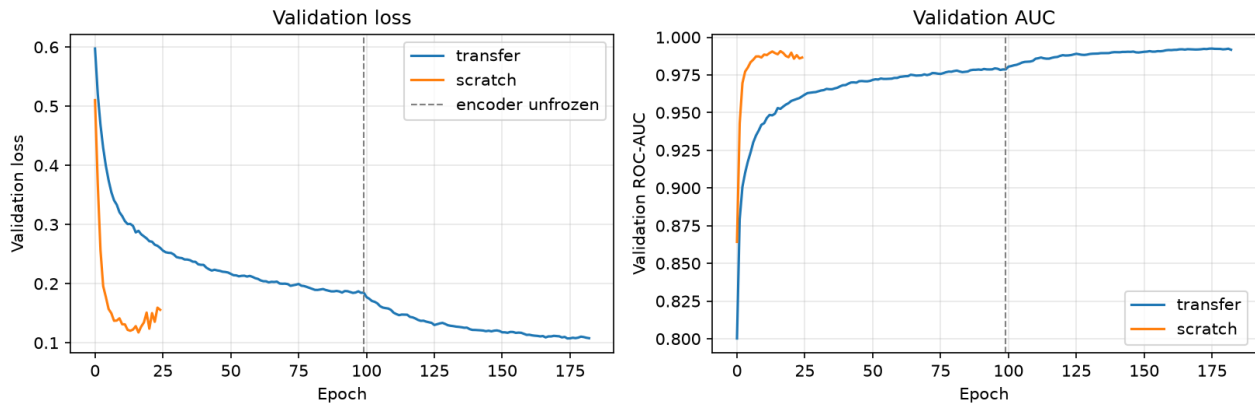


Figure 1. Validation loss and AUC on the target domain. The scratch model converges and stops within 25 epochs. The transfer model spends 100 epochs with the encoder frozen and only begins closing the gap after the unfreeze, reaching equivalent performance at epoch 183.

Interpretation

Section 7.3.4 also supplies the explanation: domain mismatch. The two domains share a feature space only by construction. Two of the ten services present in the target domain occur in the source domain, and the `flag` field takes eleven values in the source and one in the target. The encoder had learned TCP session structure, which fine-tuning had to largely overwrite before it became useful for connectionless traffic. A second factor is that the target task is easy — the scratch model exceeded 0.94 validation AUC within two epochs — leaving little headroom for prior knowledge to contribute. Transfer learning is not free, and whether it pays depends on the distance between domains rather than on the technique itself.

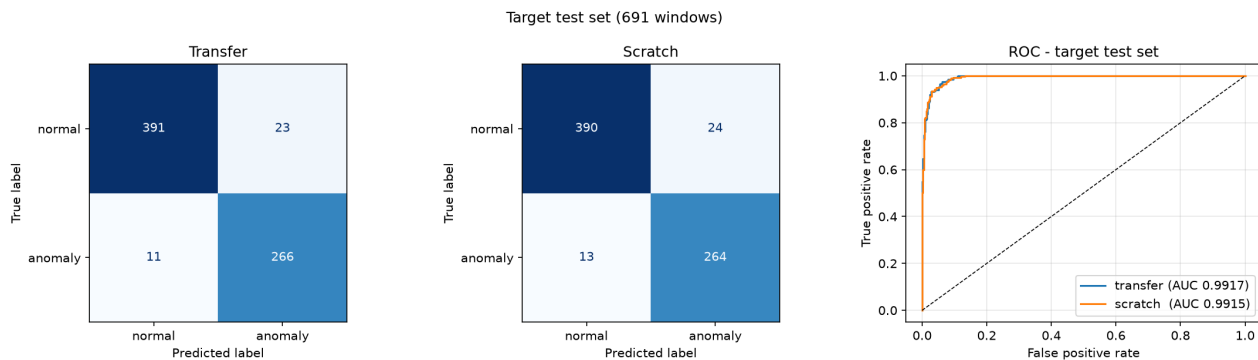


Figure 2. Target test set. The two confusion matrices differ by one false positive and two false negatives, and the ROC curves are superimposed.