Credit card fraud detection

Niccolò Salvi - Beatrice Zani

Why?

- As card fraud grows more sophisticated, financial institutions increasingly rely on machine learning to detect rare threats hidden in vast volumes of legitimate transactions.
- Card frauds are rapidly rising, with losses reaching €4.3 billion in 2022 alone.

Dataset

Each transaction sample includes 31 features:

- 28 anonymized features using PCA for privacy reasons, labeled V1-V28.
- 3 original features:
  - **time:** elapsed time from the first transaction;
  - **amount:** transaction value;
  - **class:** label indicating fraud (1) or legitimate (0).

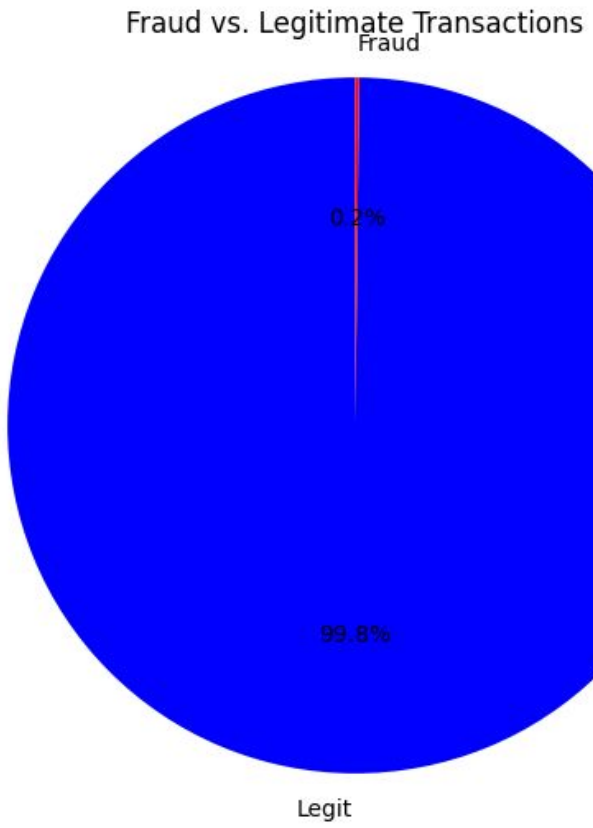
Time	V1	…	V28	Amount	Class

Methodology

1. Data preprocessing
2. Autoencoder training
3. Generation of fraud samples using SVM for filtering
4. Attention-based LSTM Classifier
5. Gradient Boosting integration
6. Test of the results

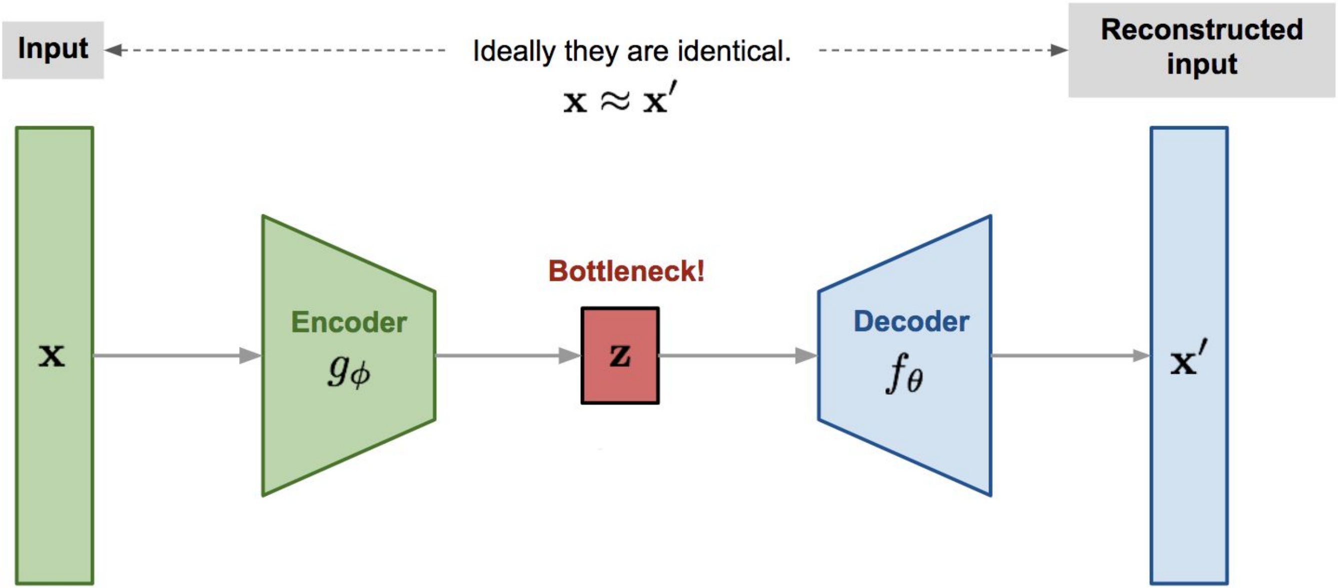
Data preprocessing

- Extracted features values from the dataset
- Scaled ‘Amount’ and ‘Time’ features using the RobustScaler to minimize the impact of outliers
- Separated features (X) and labels (y)
- Split data into **training** and **testing** sets



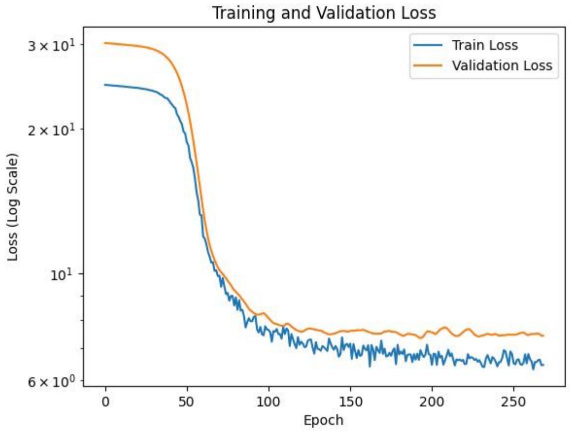
Autoencoder

Encoder: Compresses input data into a low-dimensional latent space
Decoder: Reconstructs the original input from the latent representation



Our structure

- Reduce from **29 input features** to a compact latent space (8 neurons), then expand back to 29 outputs. This compression forces the model to learn meaningful, compact representations of the fraud patterns.
- **Dropout layers** are added to prevent overfitting by randomly deactivating neurons during training
- Trained on fraud data only and to minimize the **reconstruction error**



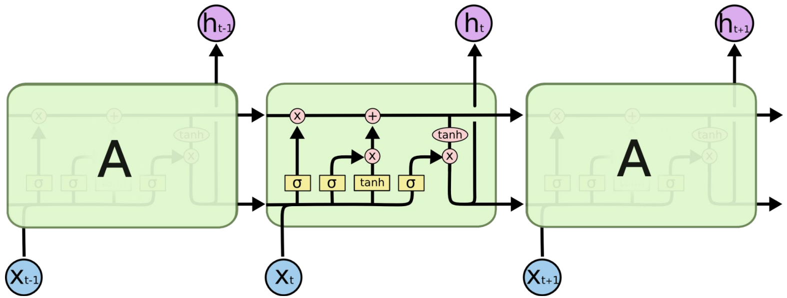
Encoder	Decoder
29 features (input)	8 neurons, fully connected
23 neurons (fully connected)	17 neurons, fully connected
dropout = 0.1	dropout = 0.2
19 neurons, fully connected	19 neurons, fully connected
dropout = 0.2	dropout = 0.1
17 neurons, fully connected	23 neurons (fully connected)
8 neurons, fully connected	29 features (output)

Fraud samples Generation

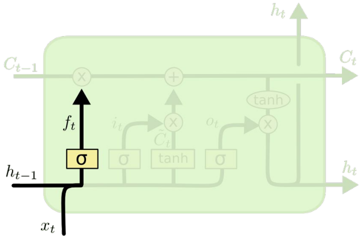
- Train an **SVM model** to distinguish between fraud and legitimate samples, this step will discard unrealistic or noisy synthetic samples generate, that could harm the training of the final classifier.
- **Synthetic samples generation:** generated new fraud samples by interpolating between encoded fraud representations in the autoencoder's latent space, adding noise to increase diversity.
- **Decoding step:** transformed these latent vectors back into the original space using the decoder.
- **Filtering:** filtered out unrealistic fraud samples from the generated data.
- **Repeat** until dataset is balanced.

```
Final shape of data: (398016, 29)
Final # Fraud: 199008
Final # Legit: 199008
```

LSTM

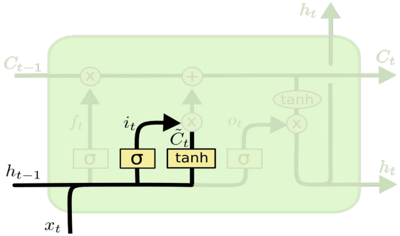


1. **Forget** Gate Layer



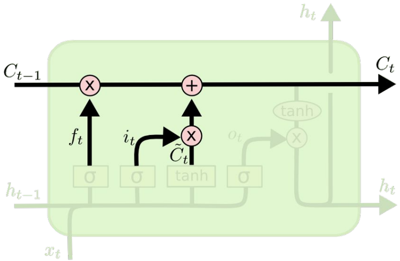
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

2. **Input** Gate Layer



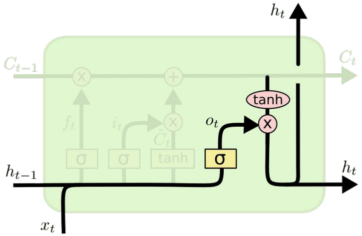
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

3. **Output** Gate Layer



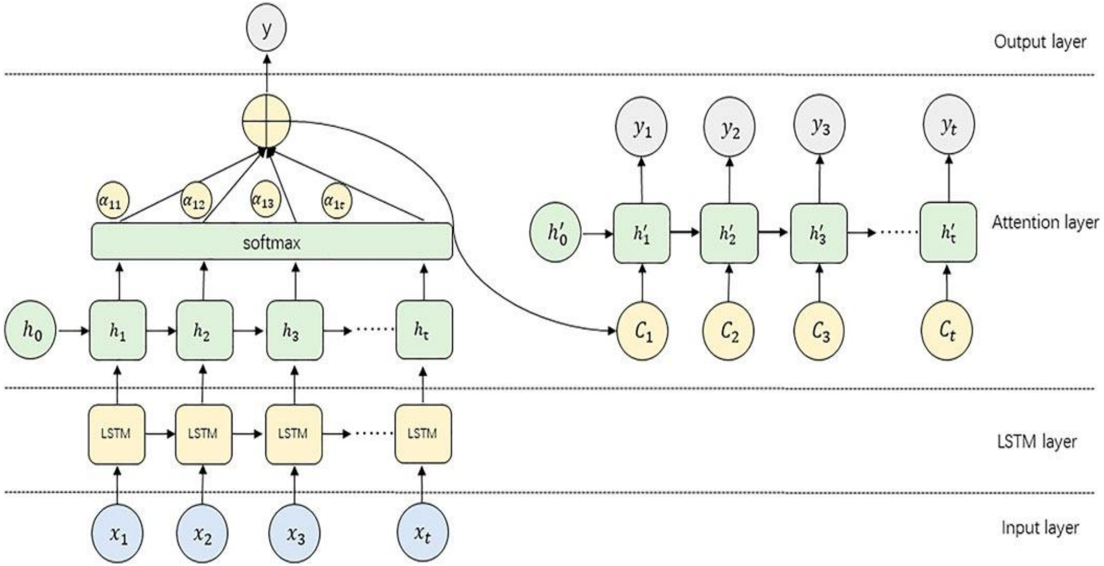
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

ALSTM

→ **Why?**

- fraud is often a **sequential** phenomenon
- needs to capture **temporal dependencies**



→ **Architecture:**

- uses **gates** (forget, input, output) to **retain** or **discard** old and new information
- **attention mechanism** used to to not only encapture sequences linearly, but enhancing performance by dynamically assigning weights to different elements within a sequence based on their relevance. Given *hi* the hidden states:

$$a_{ij} = \frac{\exp(\alpha(s_{i-1}, h_j))}{\sum_{k=1}^{n} \exp(\alpha(s_{i-1}, h_k))}$$

Gradient Boosting integration

- Boosting is an **ensemble learning technique** that builds a strong predictive model by combining multiple weak learners sequentially.
- Each new model is trained to **correct the errors (residuals)** of the combined previous models.
- It optimizes a loss function by using **gradient descent in function space**, gradually improving prediction accuracy.

Algorithm 2 GB-ALSTM Algorithm.

Input: Training set $T = \{(x_i, y_i)\}_{i=1}^{m}$, ALSTM as base learner, number of estimators ($n\_estimators$)
Initialize $h^0(x)$ with a constant: $h^0(x) = \arg\min \sum_{i=1}^{n} L(y_i, 0)$
for $t = 1, \ldots, n\_estimators$ **do**
    Compute residuals: $r_i^t = -\frac{\partial L}{\partial h^{t-1}(x_i)}$
    Train $ALSTM_t(x)$ using the dataset (x_i, r_i^t)
    Calculate step length: $\alpha^t = \arg\min_\alpha \sum_{i=1}^{n} L(y_i, h^{t-1}(x_i) + \alpha h^t(x_i))$
    Update the model: $h(x) = h(x) + \alpha^t h_t(x)$
end for
Output: Final ensemble model $h(x)$

Results

After training our model, we tested it on unseen data using the following metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$