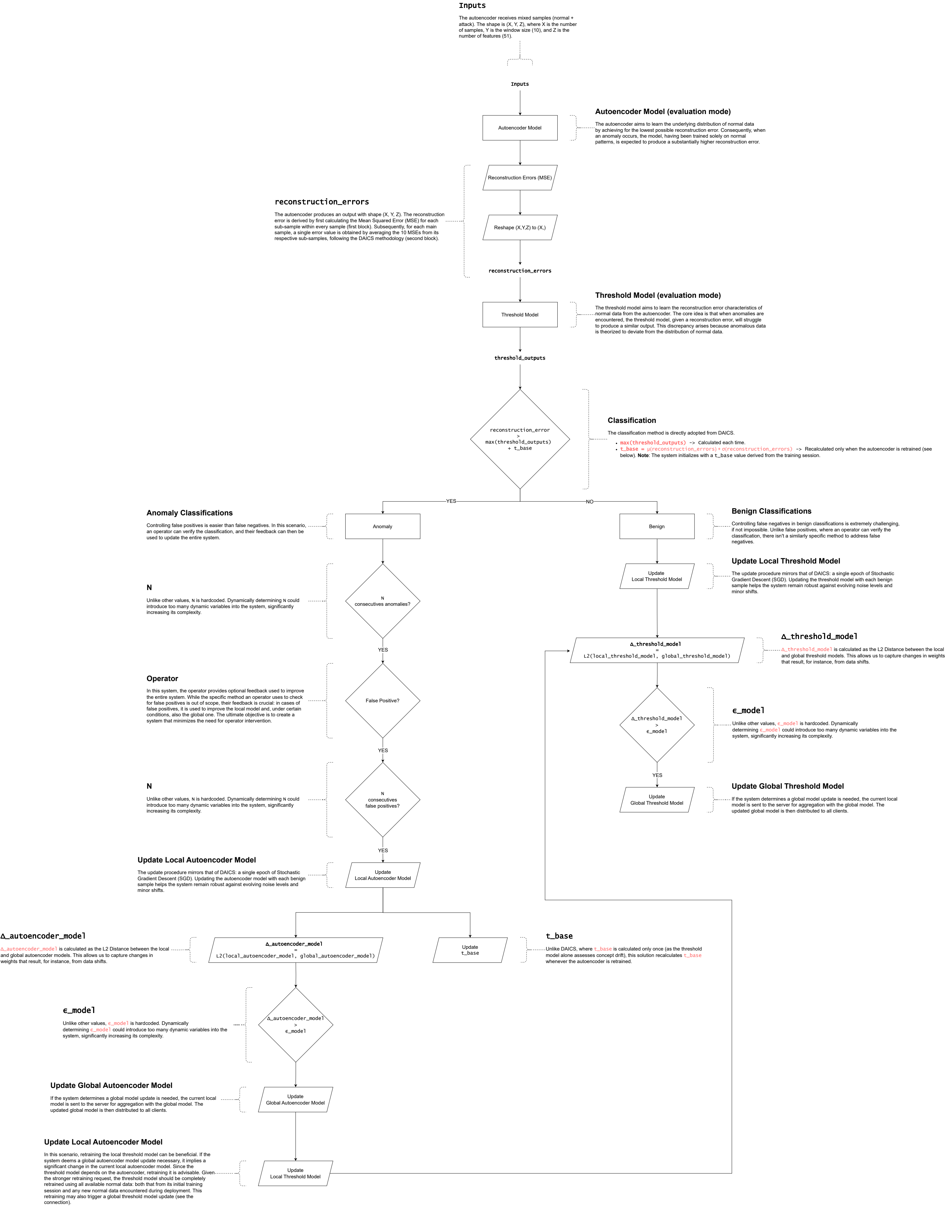
Inputs
The autoencoder receives mixed samples (normal + attack). The shape is (X, Y, Z), where X is the number of samples, Y is the window size (10), and Z is the number of features (51).
Inputs
Autoencoder Model
Autoencoder Model (evaluation mode)
The autoencoder aims to learn the underlying distribution of normal data by achieving for the lowest possible reconstruction error. Consequently, when an anomaly occurs, the model, having been trained solely on normal patterns, is expected to produce a substantially higher reconstruction error.
Reconstruction Errors (MSE)
Reshape (X,Y,Z) to (X,)
reconstruction_errors
reconstruction_errors
The autoencoder produces an output with shape (X, Y, Z). The reconstruction error is derived by first calculating the Mean Squared Error (MSE) for each sub-sample within every sample (first block). Subsequently, for each main sample, a single error value is obtained by averaging the 10 MSEs from its respective sub-samples, following the DAICS methodology (second block).
Threshold Model
Threshold Model (evaluation mode)
The threshold model aims to learn the reconstruction error characteristics of normal data from the autoencoder. The core idea is that when anomalies are encountered, the threshold model, given a reconstruction error, will struggle to produce a similar output. This discrepancy arises because anomalous data is theorized to deviate from the distribution of normal data.
threshold_outputs
reconstruction_error > max(threshold_outputs) + t_base
Classification
The classification method is directly adopted from DAICS.
• max(threshold_outputs) -> Calculated each time.
• t_base = μ(reconstruction_errors) + σ(reconstruction_errors) -> Recalculated only when the autoencoder is retrained (see below). Note: The system initializes with a t_base value derived from the training session.
YES
NO
Anomaly
Anomaly Classifications
Controlling false positives is easier than false negatives. In this scenario, an operator can verify the classification, and their feedback can then be used to update the entire system.
Benign
Benign Classifications
Controlling false negatives in benign classifications is extremely challenging, if not impossible. Unlike false positives, where an operator can verify the classification, there isn't a similarly specific method to address false negatives.
N consecutives anomalies?
N
Unlike other values, N is hardcoded. Dynamically determining N could introduce too many dynamic variables into the system, significantly increasing its complexity.
Update Local Threshold Model
Update Local Threshold Model
The update procedure mirrors that of DAICS: a single epoch of Stochastic Gradient Descent (SGD). Updating the threshold model with each benign sample helps the system remain robust against evolving noise levels and minor shifts.
Δ_threshold_model = L2(local_threshold_model, global_threshold_model)
Δ_threshold_model
Δ_threshold_model is calculated as the L2 Distance between the local and global threshold models. This allows us to capture changes in weights that result, for instance, from data shifts.
YES
False Positive?
Operator
In this system, the operator provides optional feedback used to improve the entire system. While the specific method an operator uses to check for false positives is out of scope, their feedback is crucial: in cases of false positives, it is used to improve the local model and, under certain conditions, also the global one. The ultimate objective is to create a system that minimizes the need for operator intervention.
Δ_threshold_model > ϵ_model
ϵ_model
Unlike other values, ϵ_model is hardcoded. Dynamically determining ϵ_model could introduce too many dynamic variables into the system, significantly increasing its complexity.
YES
YES
Update Global Threshold Model
Update Global Threshold Model
If the system determines a global model update is needed, the current local model is sent to the server for aggregation with the global model. The updated global model is then distributed to all clients.
N consecutives false positives?
N
Unlike other values, N is hardcoded. Dynamically determining N could introduce too many dynamic variables into the system, significantly increasing its complexity.
YES
Update Local Autoencoder Model
Update Local Autoencoder Model
The update procedure mirrors that of DAICS: a single epoch of Stochastic Gradient Descent (SGD). Updating the autoencoder model with each benign sample helps the system remain robust against evolving noise levels and minor shifts.
Δ_autoencoder_model
Δ_autoencoder_model is calculated as the L2 Distance between the local and global autoencoder models. This allows us to capture changes in weights that result, for instance, from data shifts.
Δ_autoencoder_model = L2(local_autoencoder_model, global_autoencoder_model)
Update t_base
t_base
Unlike DAICS, where t_base is calculated only once (as the threshold model alone assesses concept drift), this solution recalculates t_base whenever the autoencoder is retrained.
ϵ_model
Unlike other values, ϵ_model is hardcoded. Dynamically determining ϵ_model could introduce too many dynamic variables into the system, significantly increasing its complexity.
Δ_autoencoder_model > ϵ_model
Update Global Autoencoder Model
If the system determines a global model update is needed, the current local model is sent to the server for aggregation with the global model. The updated global model is then distributed to all clients.
Update Global Autoencoder Model
Update Local Autoencoder Model
In this scenario, retraining the local threshold model can be beneficial. If the system deems a global autoencoder model update necessary, it implies a significant change in the current local autoencoder model. Since the threshold model depends on the autoencoder, retraining it is advisable. Given the stronger retraining request, the threshold model should be completely retrained using all available normal data: both that from its initial training session and any new normal data encountered during deployment. This retraining may also trigger a global threshold model update (see the connection).
Update Local Threshold Model