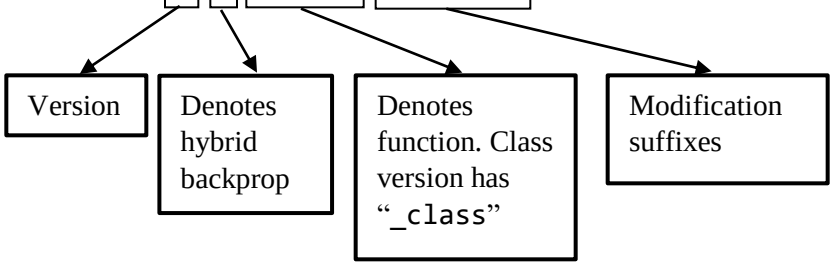
Model Key and Description

Nomenclature: **lstm_v2_c_CSV_FED_FORE_AUTO**



Unless otherwise mentioned, all models were trained through hybrid backpropagation

For each structure, there exists two versions: a class version and a csv_fed version. The class version is used in genetic optimization, and the csv_fed version is used for real training and testing

Naïve:

- Uses current value as prediction for next timestep
- Base value control—anything worse than this is not worth pursuing

Vanilla RNN:

- Uses standard (non-gated) RNN to demonstrate benefits of LSTM

ARIMA:

- Statistical prediction model
- Used as a high-bound standard—anything statistically significantly better would have outperformed statistical time-series methods

Version 0: Vanilla LSTM

- Standard LSTM with forget, input, suggestion, and output gates with hybrid (continuous) cell state and same dimension cell dimensions
- Used as another high-bound standard—any model modification that performs statistically significantly better would have outperformed the standard LSTM

Version 1: Tanh with Reduce Sum

- Hyperbolic tangent node is added to cell state to constrain values, increasing stability
- Cell dimensions are augmented with a element-wise summation to produce a single scalar output

Version 2: Tanh with reLU layer

a. Discrete Propagation
    - Hyperbolic tangent node with augmented cell dimensions, but with a fully connected reLU layer to propagate hidden states to output
    - Initializes states to zero for every footprint
b. Hybrid Backpropagation
    - Same structure as discrete propagation, but instead of initializing states to zero, the gradient flattened but still represented numerically by initializing states using previous states value

Model Key and Description

c. Hybrid with Compression Layer
   - Same propagation as the Hybrid Backpropagation model but contains a fully-connected compressional ($21 \rightarrow 1$) layer to reduce dimensionality of input

d. Hybrid with Absolute Value Loss Function
   - Same propagation and structure as the Hybrid Backpropagation model but uses an absolute value loss function to improve sub-one convergence rates

e. Hybrid with High Forget Bias
   - Same propagation and structure as the Hybrid Backpropagation model but uses a high initial bias to encourage past data retention

f. Hybrid with PCA
   - Same as the Hybrid with Compression Layer but instead of a compression layer, a PCA is used to compare compression algorithms

Version 3-4 non-existent due to changing naming standards

Version 5: Tanh with reLU layer and Cell State Peepholes

- Hyperbolic tangent node with reLU output layer, and the cell state is concatenated with hidden layer and model input to be used as inputs to the gates, attempting to increase long-term accuracy

Version 6: Tanh with reLU layer and Input/Forget Concatenation

a. Normal Hybrid Backpropagation
   - Hyperbolic tangent node with reLU output layer, and the forget (removal) and input (insertion) gates are merged into one, with one being the $1 - [k]$ of the other, attempting to reduce cell null space

b. Normal Hybrid Backpropagation with High Forget Bias
   - Same structure as Normal Hybrid Backpropagation, but uses a high forget bias to encourage data retention and long-term stability

Version 7: Tanh with reLU layer, Input/Forget Concatenation, and Cell State Peepholes

- Combination of version 5 and 6: uses cell states for gate context and also combines the forget and input gates

Version 8 non-existent due to the removal of multilayer LSTMs from analysis

Version 9: Tanh with reLU layer and Input/Forget Concatenation and Hidden Propagation

- Hyperbolic tangent node with reLU output layer and a hyperbolic tangent fully-connected layer to propagate some current hidden information to the next hidden to increase information carry-over