



Show exact code lines where scaling is applied in the scripts

Here are the **exact code lines** showing how scaling was fixed:

Original Scripts - Problems [\[1\]](#) [\[2\]](#)

Training Script (LSTM-only_inputs_v2.py)

✗ ISSUE 1: Statistics calculated BEFORE split

```
# Lines ~80-87
X_data = np.concatenate(X_data, axis=0)
Y_data = np.concatenate(Y_data, axis=0)

x_mean=np.mean(X_data.reshape(-1,input_size),axis=0) # ← entire dataset
x_std =np.std(X_data.reshape(-1,input_size),axis=0)
x_min=np.min(X_data.reshape(-1,input_size),axis=0)    # ← entire dataset
x_max =np.max(X_data.reshape(-1,input_size),axis=0)

y_mean=np.mean(Y_data) # ← entire dataset
y_std =np.std(Y_data)
Y_data= (Y_data-y_mean)/(y_std) # Z-score for outputs
```

✗ ISSUE 2: Split happens AFTER statistics computed

```
# Line ~94
X_train, X_eval, y_train, y_eval = train_test_split(X_data, Y_data, test_size=0.2,...)
```

✗ ISSUE 3: MinMax for inputs only

```
# Lines ~102-103
X_train_scaled=(X_train-x_min)/(x_max-x_min+delta)
X_eval_scaled=(X_eval-x_min)/(x_max-x_min+delta)
# But Y already scaled with Z-score above!
```

Test Script (test_lstm_only_inputs_PIP1.py)

✗ ISSUE 4: Wrong stride

```
# Line ~20
stride=1 # Training used stride=2
```

✗ ISSUE 5: Loading wrong statistics

```
# Lines ~27-32
y_mean=loaded_model['output_means'] # From entire dataset, Z-score
y_std=loaded_model['output_std']
```

✗ ISSUE 6: Wrong inverse transform

```
# Line ~75
data_predict = (data_predict*y_std)+y_mean # Z-score inverse (mismatch!)
```

Fixed Scripts - Solutions [\[3\]](#) [\[4\]](#)

Training Script (train_lstm_pip.py)

✓ FIX 1: Split FIRST

```
# Lines ~114-118
X_train, X_eval, y_train, y_eval = train_test_split(
    X_data, Y_data, test_size=0.2, random_state=42, shuffle=True
)
```

✓ FIX 2: Calculate from training only

```
# Lines ~123-127
x_min = np.min(X_train.reshape(-1, input_size), axis=0) # ← training only
x_max = np.max(X_train.reshape(-1, input_size), axis=0)

y_min = np.min(y_train, axis=0) # ← training only
y_max = np.max(y_train, axis=0)
```

✓ FIX 3: Consistent MinMax for both

```
# Lines ~129-133
X_train_scaled = (X_train - x_min) / (x_max - x_min + delta)
X_eval_scaled = (X_eval - x_min) / (x_max - x_min + delta)

y_train_scaled = (y_train - y_min) / (y_max - y_min + delta) # ← MinMax!
y_eval_scaled = (y_eval - y_min) / (y_max - y_min + delta)
```

✓ FIX 4: Save correct statistics

```
# Lines ~201-213
ckpt = {
    'input_min': x_min,      # Training only
    'input_max': x_max,
    'output_min': y_min,    # Training only, MinMax params
    'output_max': y_max,
    'stride': stride,      # Save for consistency
    ...
}
```

Test Script (test_lstm_pip.py)

✓ FIX 5: Load correct statistics

```
# Lines ~27-36
x_min = loaded["input_min"]      # Training stats
x_max = loaded["input_max"]
y_min = loaded["output_min"]     # MinMax params
y_max = loaded["output_max"]
stride = loaded.get("stride", 2) # ← Same as training
```

✓ FIX 6: Same stride

```
# Line ~56
y = y[:, :stride] # stride=2, matches training
```

✓ FIX 7: Correct inverse transform

```
# Lines ~85-86
data_predict = preds_scaled * (y_max - y_min + delta) + y_min # MinMax inverse
```

Key Principle^[5] ^[6]

The fundamental fix is: **compute normalization statistics from training data only, then apply the same transformation consistently to validation and test sets.** This prevents data leakage and ensures the model sees the same data distribution during training and inference.^[4] ^[7]

✱

1. test_lstm_only_inputs_PIP1.py
2. LSTM-only_inputs_v2.py
3. <https://stackoverflow.com/questions/48511048/use-minmaxscaler-on-training-data-to-generate-std-min-and-max-to-be-used-on-test>
4. <https://www.machinelearningmastery.com/data-preparation-without-data-leakage/>

5. <https://towardsdatascience.com/an-introduction-to-data-leakage-f1c58f7c1d64/>
6. <https://www.linkedin.com/pulse/data-scaling-training-space-machine-learning-darko-medin>
7. <https://www.datacamp.com/tutorial/normalization-in-machine-learning>