Powercast AI - Model Architecture Documentation

Overview

Powercast AI uses a **Multi-Horizon XGBoost Ensemble** for electrical load forecasting. The model predicts grid load at 15-minute intervals for a 24-hour horizon (96 time steps), with conformal prediction intervals for uncertainty quantification.

```
                                POWERCAST AI MODEL

   Historical
   Load Data              Feature          96 XGBoost Models           24-Hour
   (1 week)             Engineering      (One per 15-min step)       Forecast
                                                         + Intervals

   Weather Data
   (optional)

                                           Conformal Prediction
                                               Intervals
```

Model Specifications

Core Architecture

Component	Specification
Model Type	XGBoost Gradient Boosted Trees
Architecture	Multi-output via 96 independent models
Output Horizon	24 hours (96 steps × 15 minutes)
Input Features	21 engineered features
Uncertainty	Conformal Prediction (80%, 90%, 95% intervals)
Model Size	115 MB (joblib serialized)

Hyperparameters (Optuna-Tuned)

```
{
  "max_depth": 7,
  "learning_rate": 0.0612,
  "subsample": 0.823,
  "colsample_bytree": 0.919,
```

```
  "min_child_weight": 5,
  "gamma": 0.0165,
  "reg_alpha": 0.00185,
  "reg_lambda": 0.656,
  "n_estimators": 500
}
```

Tuning Details: - Optimization: Optuna with 50 trials - Objective: Minimize cross-validation MAPE - Best CV MAPE: 1.91% - Tuning Time: 1.21 minutes

---

Performance Metrics

Overall Performance

Metric	Value	Description
Test MAPE	0.91%	Mean Absolute Percentage Error
Test MAE	69.16 MW	Mean Absolute Error
Coverage (90%)	91.04%	% of actuals within prediction interval
Inference Time	157.83 ms	Time to generate 96-step forecast

Horizon-Specific MAPE

Horizon	MAPE	Interpretation
15 min	0.93%	Excellent short-term accuracy
3 hours	0.91%	Very high accuracy
6 hours	0.89%	Best accuracy at this horizon
12 hours	0.90%	Consistent mid-range accuracy
18 hours	0.96%	Slight degradation expected
24 hours	0.95%	Strong end-of-day accuracy

```
MAPE by Forecast Horizon
1.0%

0.95%

0.90%

0.85%
     0h   4h   8h  12h  16h  20h  24h
```

Feature Engineering

Input Feature Vector (21 Features)

The model uses a carefully engineered feature set optimized for electrical load forecasting:

1. Lag Features (4 features)

Feature	Description	Lag
`lag_1h`	Load 1 hour ago	4 steps
`lag_6h`	Load 6 hours ago	24 steps
`lag_24h`	Load same time yesterday	96 steps
`lag_168h`	Load same time last week	672 steps

```
features = [
    recent_load[-4],    # 1 hour ago
    recent_load[-24],   # 6 hours ago
    recent_load[-96],   # 24 hours ago (same time yesterday)
    recent_load[-672],  # 168 hours ago (same time last week)
]
```

2. Rolling Statistics (4 features)

Feature	Window	Statistic
`roll_24h_mean`	24 hours	Mean
`roll_24h_std`	24 hours	Standard deviation
`roll_168h_mean`	1 week	Mean
`roll_168h_std`	1 week	Standard deviation

```
w24 = recent_load[-96:]   # Last 24 hours
w168 = recent_load[-672:] # Last week
features += [w24.mean(), w24.std(), w168.mean(), w168.std()]
```

3. Calendar/Cyclical Features (8 features)

Feature	Encoding	Purpose
`hour_sin`	sin(2 × hour/24)	Circular hour encoding
`hour_cos`	cos(2 × hour/24)	Circular hour encoding
`dow_sin`	sin(2 × weekday/7)	Day of week (cyclic)
`dow_cos`	cos(2 × weekday/7)	Day of week (cyclic)
`month_sin`	sin(2 × month/12)	Seasonal pattern
`month_cos`	cos(2 × month/12)	Seasonal pattern

Feature	Encoding	Purpose
`is_weekend`	0 or 1	Weekend flag
`is_peak_hour`	0 or 1	Peak hours (7:00-21:00)

```python
hour = forecast_start.hour + forecast_start.minute / 60
features += [
    np.sin(2 * np.pi * hour / 24),
    np.cos(2 * np.pi * hour / 24),
    np.sin(2 * np.pi * forecast_start.weekday() / 7),
    np.cos(2 * np.pi * forecast_start.weekday() / 7),
    np.sin(2 * np.pi * forecast_start.month / 12),
    np.cos(2 * np.pi * forecast_start.month / 12),
    1.0 if forecast_start.weekday() >= 5 else 0.0,
    1.0 if 7 <= forecast_start.hour <= 21 else 0.0,
]
```

4. Weather Features (5 features)

Feature	Unit	Default
`temperature`	°C	15.0
`humidity`	%	50.0
`cloud_cover`	%	30.0
`wind_speed`	m/s	5.0
`temp_x_humidity`	interaction	7.5

```python
# Weather features (from API or defaults)
features += [temperature, humidity, cloud_cover, wind_speed, temp_x_humidity]
```

Model Architecture Details

Multi-Horizon Strategy

Instead of a single model predicting all 96 steps, we train **96 independent XGBoost models**, one for each time step in the forecast horizon:

```
                     Input Features (21)




    XGBoost           XGBoost           XGBoost
    Model 1           Model 2           Model 96
    (+15 min)         (+30 min)         (+24 hr)
```

```
     Forecast[0]          Forecast[1]    ...   Forecast[95]
```

Why this approach? - Each horizon has different optimal hyperparameters - Error patterns vary by forecast distance - Easier to interpret and debug individual horizons - Parallel training possible

Conformal Prediction Intervals

We use **split conformal prediction** for uncertainty quantification:

```
                     CONFORMAL PREDICTION

   1. Split data: Training (80%) + Calibration (20%)
   2. Train models on training set
   3. Compute residuals on calibration set:
      residual[i] = |actual[i] - predicted[i]|
   4. Compute quantiles of residuals:
      q80 = 80th percentile
      q90 = 90th percentile
      q95 = 95th percentile
   5. At inference:
      lower = prediction - margin
      upper = prediction + margin
```

Stored Margins:

```
conformal_margins = {
    "q80": <array of 96 values>,  # 80% confidence
    "q90": <array of 96 values>,  # 90% confidence (default)
    "q95": <array of 96 values>,  # 95% confidence
}
```

Inference Pipeline

Data Flow

```
                         INFERENCE PIPELINE


     Raw Load            Feature            Normalization
     History           Engineering         (Z-score)
```

```
   (672 pts)         (21 feat)



   Output           Conformal        96 XGBoost
   Response           Intervals          Predictions
   (JSON)             (q10, q90)
```

API Response Format

```json
{
  "predictions": [
    {
      "timestamp": "2026-01-29T11:15:00",
      "point": 8523.45,
      "q10": 8023.45,
      "q90": 9023.45
    },
    ...
  ],
  "metadata": {
    "model_type": "xgboost",
    "horizon_hours": 24,
    "interval_minutes": 15,
    "plant_type": "mixed",
    "generated_at": "2026-01-29T11:00:00",
    "confidence": 0.90,
    "test_mape": 0.9108
  }
}
```

Model Artifacts

File Structure

```
backend/app/models/
  xgboost_model.joblib     # 115 MB (Git LFS tracked)
  training_config.json     # Model metadata
```

Joblib Contents

```python
model_data = {
    "models": List[XGBRegressor],      # 96 trained models
```

```python
    "feature_means": np.ndarray,        # Shape: (21,)
    "feature_stds": np.ndarray,         # Shape: (21,)
    "conformal_margins": {
        "q80": np.ndarray,              # Shape: (96,)
        "q90": np.ndarray,              # Shape: (96,)
        "q95": np.ndarray,              # Shape: (96,)
    }
}
```

Training Details

Data Requirements

Requirement	Minimum	Recommended
Historical data	1 month	6+ months
Granularity	15 minutes	15 minutes
Missing data	< 5%	< 1%

Training Process

1. **Data Preparation**
    - Load historical load data (15-min intervals)
    - Handle missing values (interpolation)
    - Create rolling features
2. **Feature Engineering**
    - Extract lag features
    - Compute rolling statistics
    - Encode calendar features
    - Merge weather data (optional)
3. **Train/Test Split**
    - Training: 80% of data
    - Calibration: 10% of data (for conformal)
    - Test: 10% of data
4. **Hyperparameter Tuning**
    - Optuna optimization (50 trials)
    - 3-fold time-series cross-validation
    - Objective: Minimize MAPE
5. **Model Training**
    - Train 96 XGBoost models (one per horizon)
    - Early stopping on validation set
    - Save feature normalization parameters
6. **Conformal Calibration**
    - Compute residuals on calibration set

    - Calculate quantile margins (80%, 90%, 95%)
7. **Validation**
    - Test set evaluation
    - Horizon-specific MAPE analysis
    - Coverage verification

Training Configuration

```json
{
  "model_type": "xgboost_fast_tuned",
  "output_horizon": 96,
  "tuning": {
    "n_trials": 50,
    "best_cv_mape": 1.9093
  },
  "training_time_seconds": 372.46,
  "trained_at": "2026-01-16T13:20:41"
}
```

Deployment Architecture

Production Stack

```
                    VERCEL EDGE

                 Next.js Frontend
                 (React Dashboard)

                         API Routes

                /api/forecast
                (Proxy to FastAPI)




                  FASTAPI BACKEND

              MLInferenceService
              (Singleton Pattern)
```

```
XGBoost Model (96 models)
115 MB loaded in memory
```

Fallback Behavior

If the XGBoost model fails to load, the system automatically falls back to **mock predictions** that generate realistic Swiss grid patterns:

```python
# Mock prediction pattern
base_load = 8500  # MW
daily_variation = 2000 * np.sin(2 * np.pi * (hour - 4) / 24)
noise = np.random.normal(0, 150)
point = base_load + daily_variation + noise
```

Health Check API

```
GET /api/forecast/health

Response:
{
  "status": "healthy",           # or "degraded" if mock mode
  "model_loaded": true,
  "model_type": "xgboost",
  "test_mape": 0.9108,
  "test_mae": 69.16,
  "coverage_90": 91.04,
  "inference_time_ms": 157.83,
  "model_path": "/app/backend/app/models/xgboost_model.joblib",
  "model_exists": true
}
```

Future Improvements

Planned Enhancements

1. **Real-time Weather Integration**
   - Replace default weather features with live API data
   - Improve accuracy during extreme weather events
2. **Online Learning**
   - Periodic retraining on recent data
   - Adaptive model updates
3. **Ensemble Expansion**

- Add LSTM for capturing long-term patterns
- Gradient boosting + neural network ensemble

4. **Regional Models**
    - Train separate models for India/Switzerland grids
    - Account for regional load patterns

Model Versioning

Version	Date	Test MAPE	Notes
v1.0.0	2026-01-16	0.91%	Initial production model

References

- XGBoost Documentation: https://xgboost.readthedocs.io/
- Conformal Prediction: https://arxiv.org/abs/2107.07511
- Optuna: https://optuna.org/

Documentation generated for Powercast AI v1.0 Last updated: 2026-01-29