LSTM VS TRANSFORMER

PRÉVISION ÉNERGÉTIQUE SUR SMART GRID

Client : AI Energy Solutions
Auteur : Giulia Governatori
Date : 17 Janvier 2026

AGENDA DE LA PRÉSENTATION

1. Le Défi Business : Pourquoi prédire la consommation ?
2. Mon Parcours : De débutante à praticienne Deep Learning
3. Les Données : Comprendre avant de modéliser
4. Feature Engineering : L'art de créer les bonnes variables
5. LSTM : Le modèle de référence
6. Transformer : L'architecture révolutionnaire
7. Résultats : La bataille LSTM vs Transformer
8. Ce que j'ai appris : Leçons et surprises
9. Et après ? Vision pour le futur

LE DÉFI BUSINESS

Un problème à 62 millions d'euros

Imaginez une entreprise qui gère l'électricité de 50 000 foyers en Île-de-France. Chaque jour, elle doit acheter de l'électricité sur le marché européen (EPEX) pour le lendemain.

Le problème : **Comment savoir combien acheter ?**

TROP ACHETER	PAS ASSEZ ACHETER
→ Gaspillage : 45 M€/an	→ Pénalités : 17.5 M€/an

Coût total des erreurs de prévision : **62.5 M€/an**

MA MISSION

Objectif : Réduire l'erreur de prévision avec l'IA

	SITUATION ACTUELLE	MON OBJECTIF
MÉTHODE	Moyennes historiques	Deep Learning (LSTM, Transformer)
ERREUR (MAE)	0.75 kW/foyer	< 0.5 kW/foyer
FRAIS ANNUELS	62.5 M€	40-30 M€

Pourquoi le Deep Learning ?

- Les méthodes statistiques classiques ne captent pas les patterns complexes
- La consommation dépend de nombreux facteurs (météo, jour, heure, habitudes...)
- Les réseaux de neurones peuvent apprendre ces relations non-linéaires

MON PARCOURS DANS CE PROJET

Premier projet Deep Learning : Une aventure d'apprentissage

Ce projet représente ma première expérience complète en Deep Learning.
J'ai dû apprendre, expérimenter, me tromper, et recommencer.

Les étapes de mon apprentissage :

ÉTAPE	CE QUE J'AI APPRIS
1	Comprendre les séries temporelles et leurs spécificités
2	Éviter le data leakage (utiliser shift(1) pour les lags !)
3	Construire un LSTM de zéro avec TensorFlow/Keras
4	Implémenter un Transformer pour séries temporelles
5	Gérer l'overfitting avec régularisation L2, Dropout, Early Stopping
6	Comparer rigoureusement deux modèles et choisir le meilleur

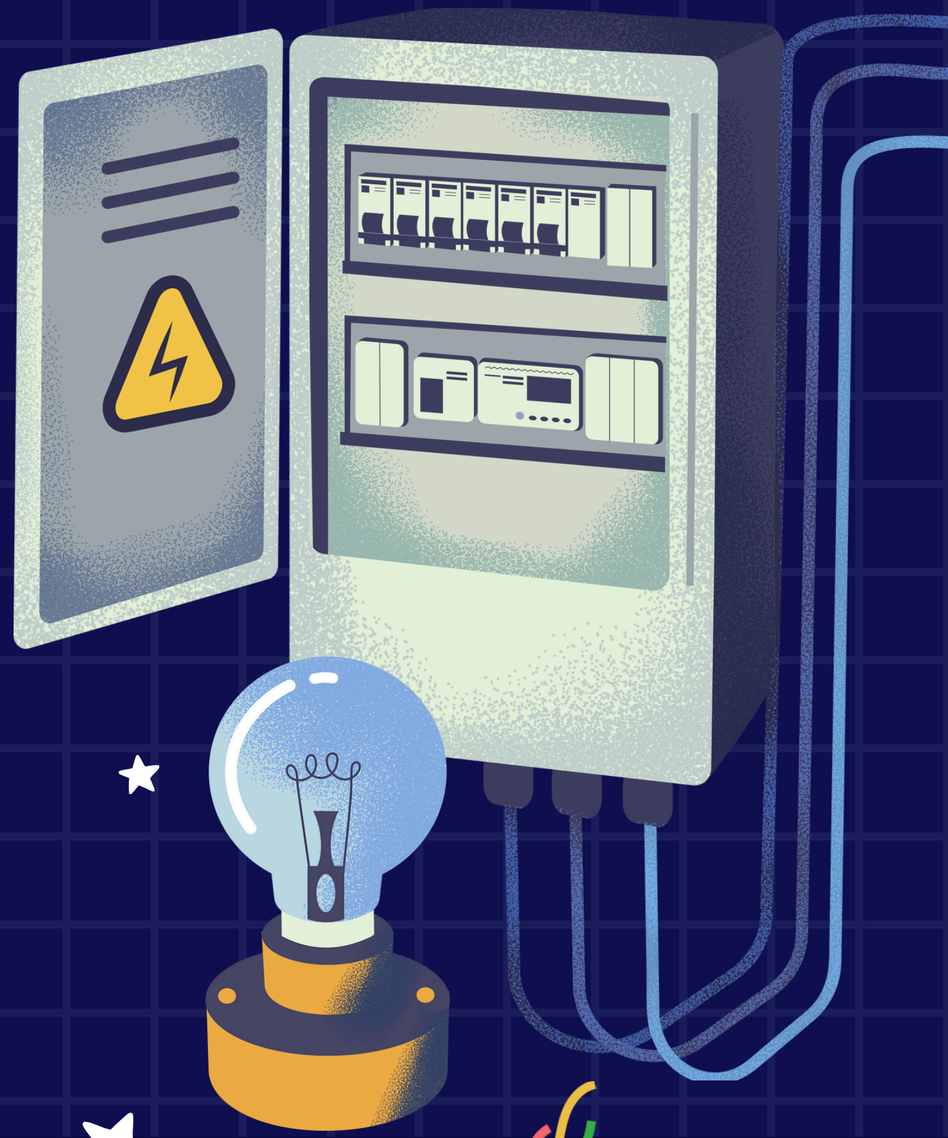
LES DONNÉES : POINT DE DÉPART

Dataset UCI : 4 ans de consommation réelle

Source	UCI Machine Learning Repository
Localisation	Sceaux, banlieue parisienne
Période	Décembre 2006 → Novembre 2010 (4 ans)
Résolution brute	1 mesure par minute (~2 millions d'observations)
Résolution finale	1 mesure par heure (34 589 observations)

Pourquoi ce dataset ?

- Données réelles d'un vrai foyer français (pas de simulation)
- Période suffisamment longue pour capturer la saisonnalité
- Plusieurs sous-compteurs (cuisine, buanderie, chauffage)



EXPLORATION DES DONNÉES (EDA)

Comprendre avant de modéliser

Observations clés :

- Distribution très asymétrique : médiane (0.60 kW) << moyenne (1.09 kW)

- Pics rares mais importants : max = 11 kW (18× la médiane)

🎯 FOCUS SUR LA VARIABLE CIBLE: Global_active_power

Moyenne:	1.092 kW
Médiane:	0.602 kW
Écart-type:	1.057 kW
Minimum:	0.076 kW
Maximum:	11.122 kW

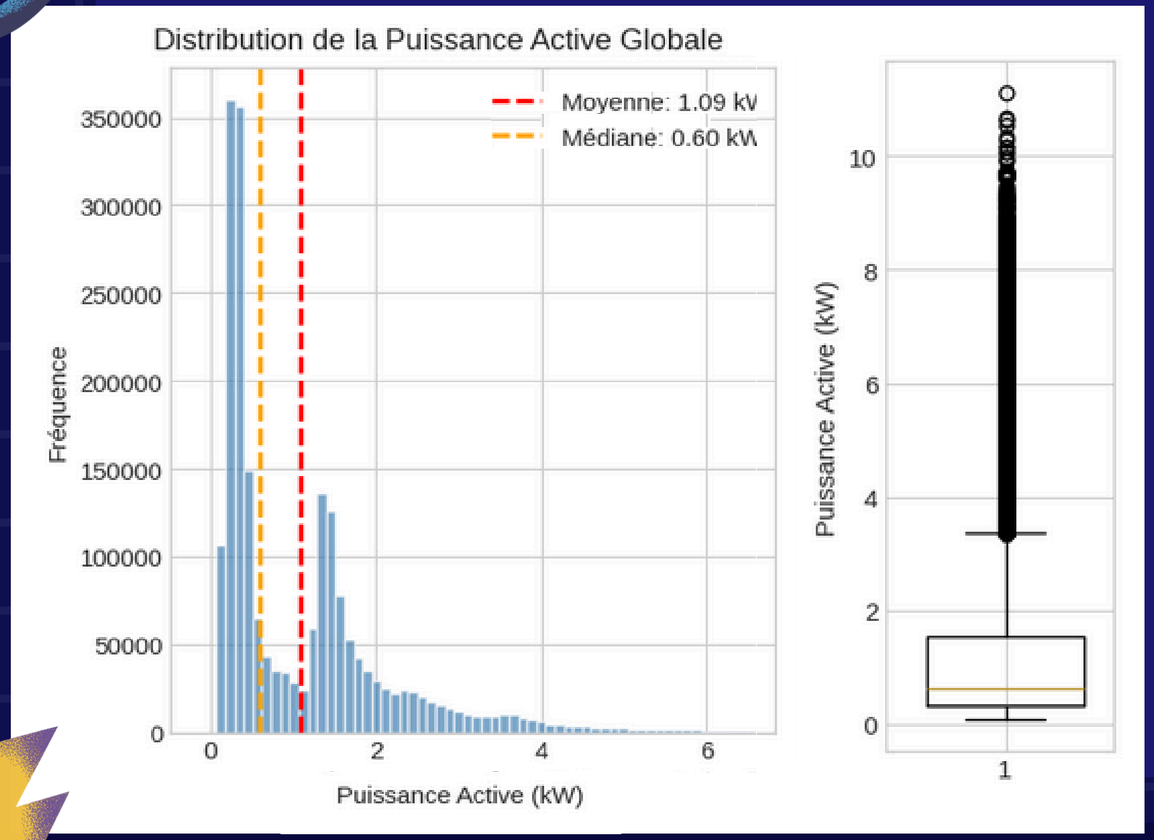
📊 Percentiles:

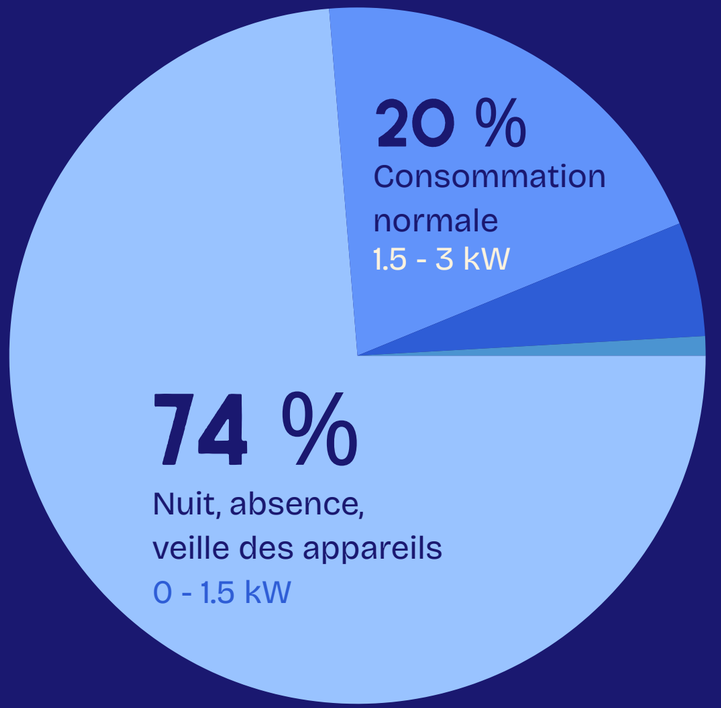
25%:	0.308 kW
50%:	0.602 kW
75%:	1.528 kW
95%:	3.264 kW
99%:	4.850 kW

Implication : Les modèles vont avoir tendance à prédire ~1.09 kW (la moyenne) et sous-estimer les pics.

DISTRIBUTION DE LA CONSOMMATION

Une distribution fortement déséquilibrée



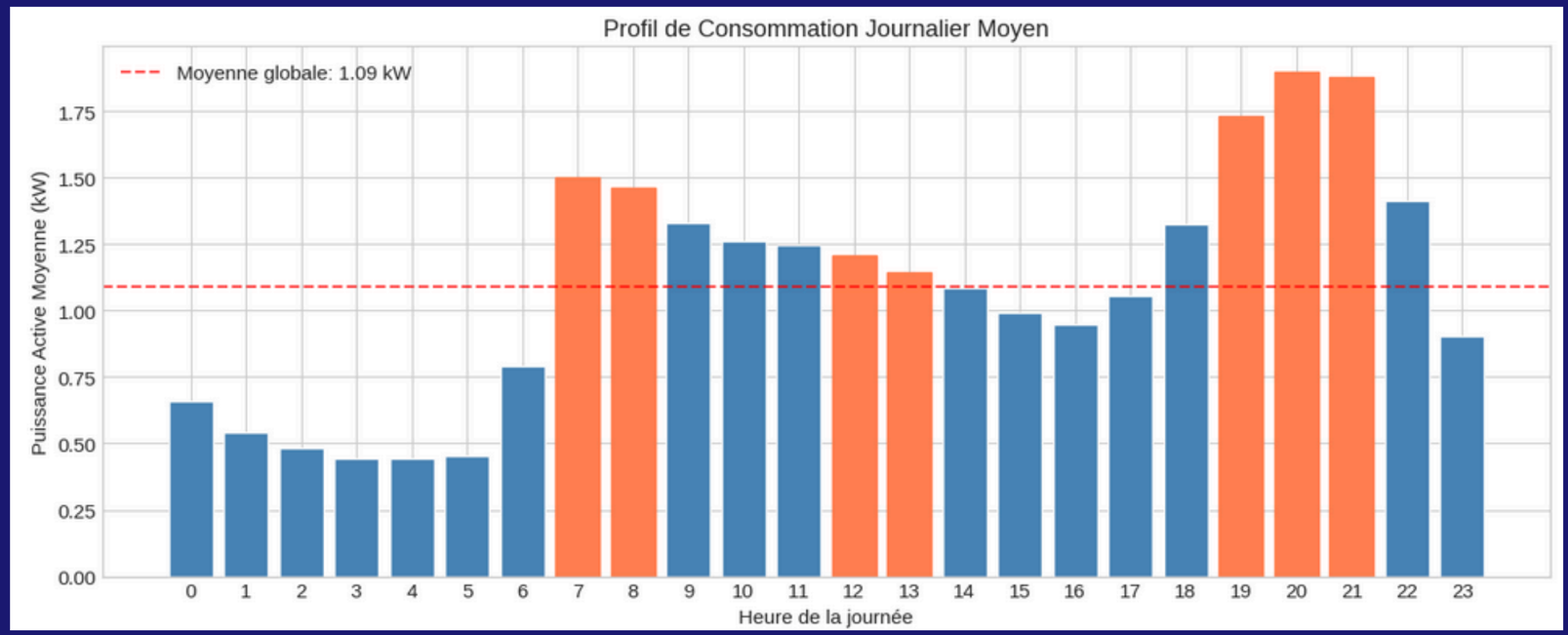


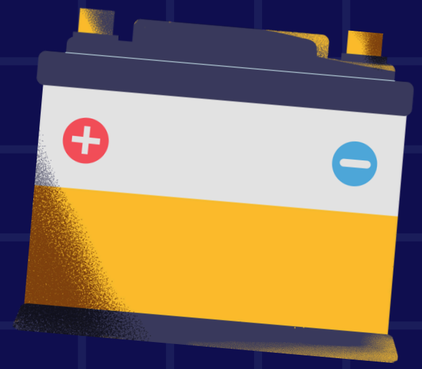
5%
Pics modérés
3 - 5 kW

1%
Pics extrêmes
> 5 kW

CONSOMMATION JOURNALIER

Les habitudes des Français révélées par les données



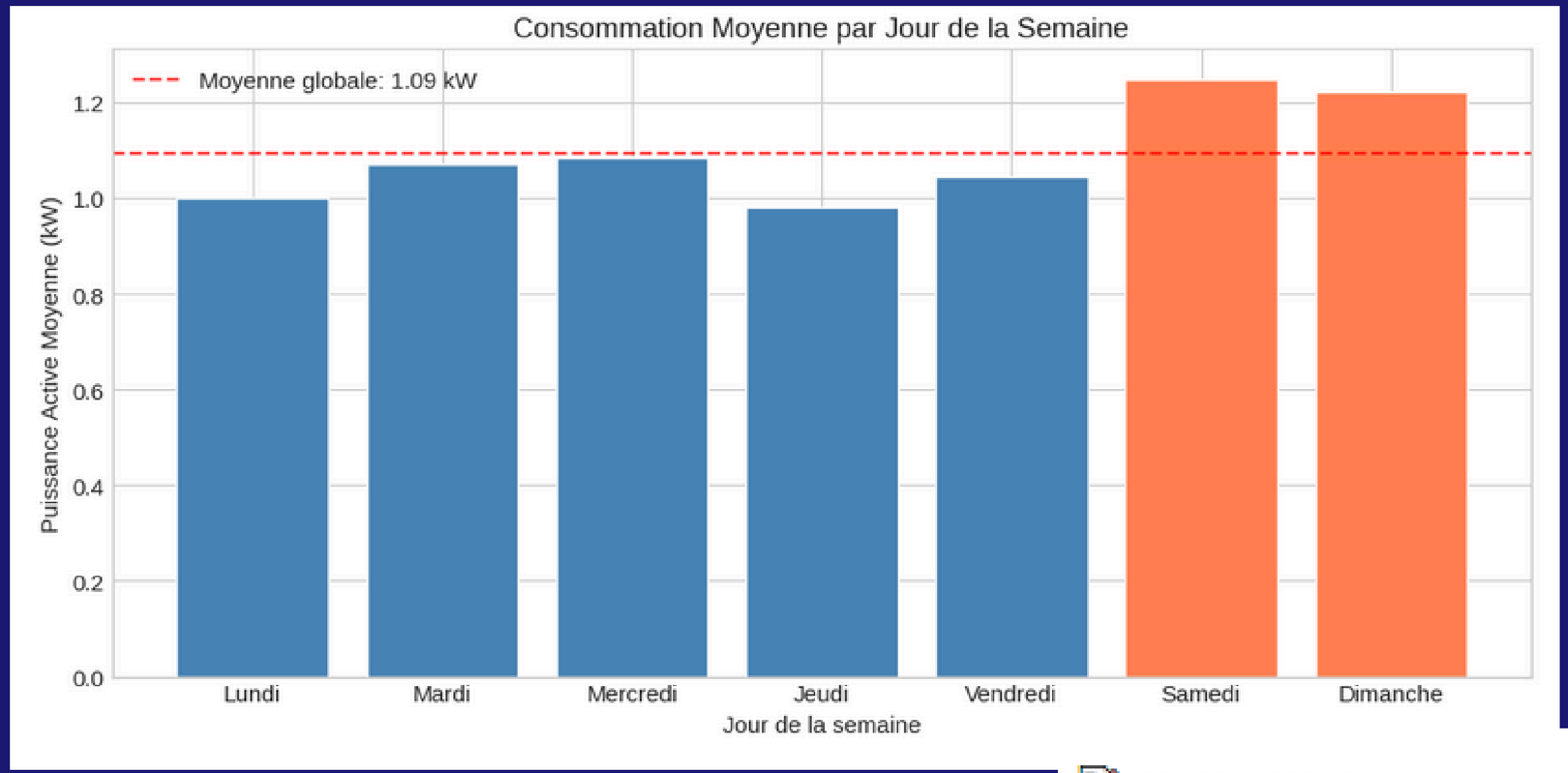


Ces patterns réguliers sont exactement ce que le Deep Learning peut apprendre !

📝 OBSERVATIONS:
- Pic matinal: 7h-9h (réveil, petit-déjeuner, préparation)
- Creux en milieu de journée: 10h-16h (absence du foyer)
- Pic du soir: 19h-21h (retour maison, cuisine, TV, chauffage)
- Minimum nocturne: 1h-5h (veille des appareils uniquement)

LE WEEKEND

Consommation Moyenne par Jour de la Semaine



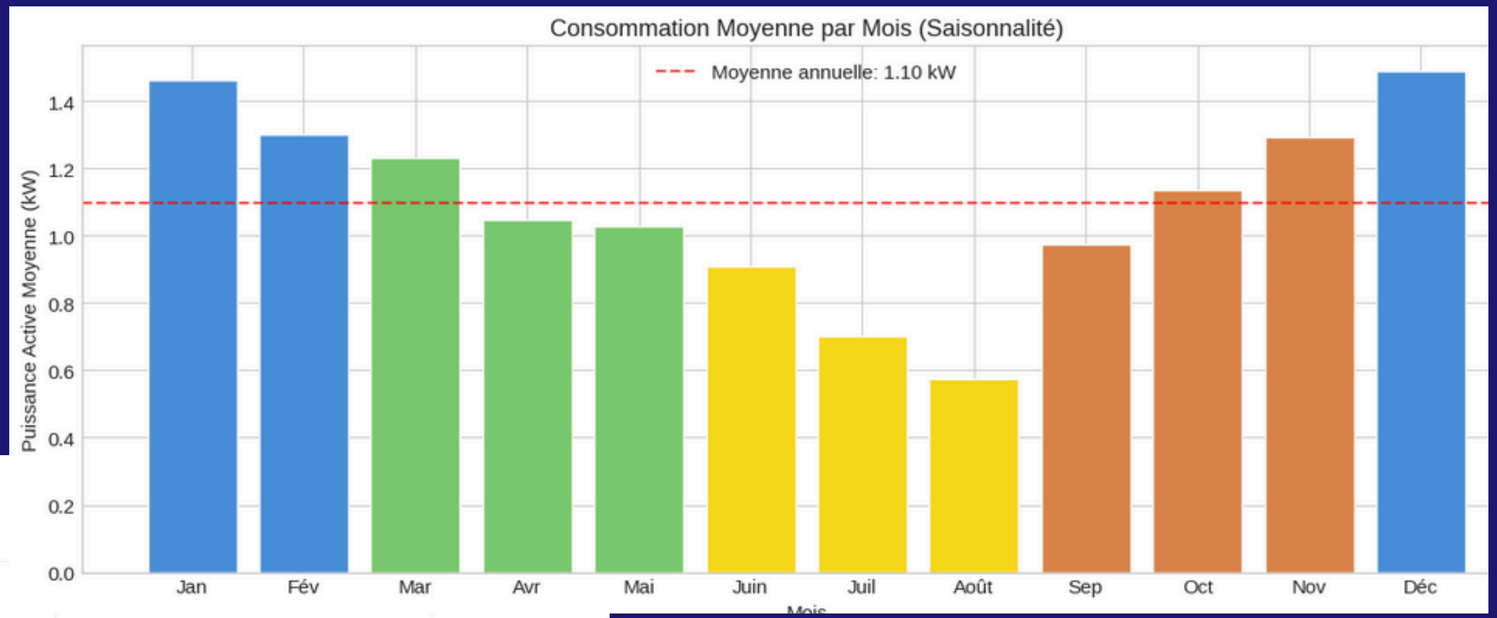
OBSERVATIONS:
- Moyenne semaine (Lun-Ven): 1.035 kW
- Moyenne weekend (Sam-Dim): 1.234 kW
- Différence: +19.2%

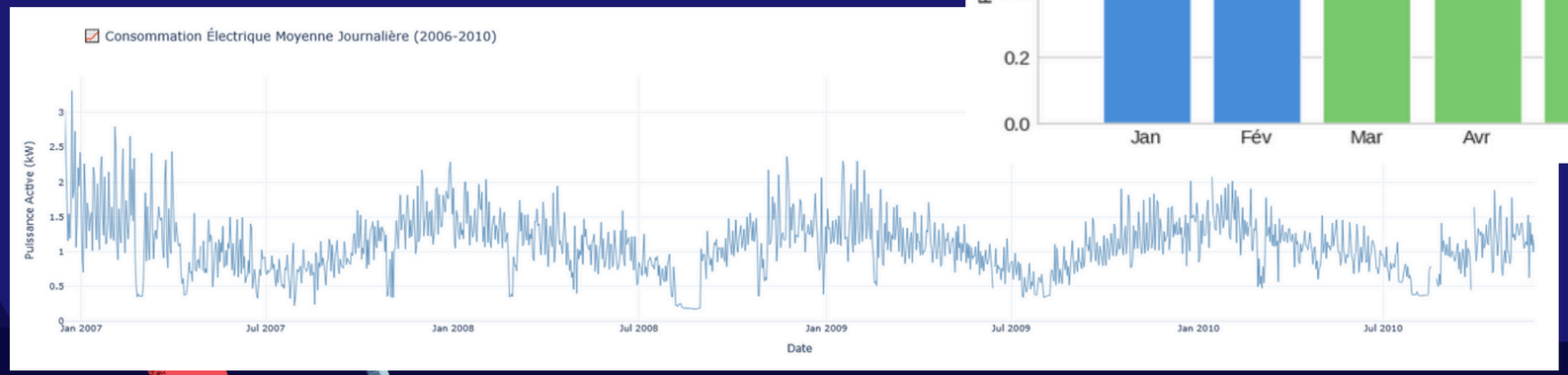
SAISONNALITÉ ANNUELLE

L'impact majeur du chauffage électrique

Observations clés :

- Consommation hivernale = 1.9× consommation estivale
- Le chauffage électrique domine la consommation
- Pattern très régulier d'une année à l'autre





- Moyenne hiver (Déc-Fév): 1.417 kW
- Moyenne été (Juin-Août): 0.727 kW
- Différence: +94.9%

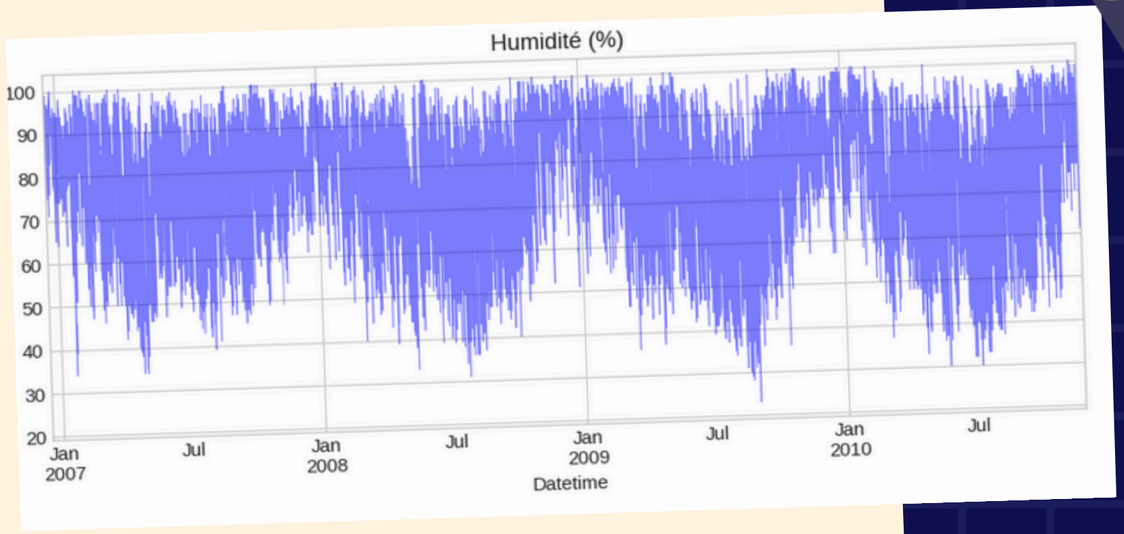
Implication : La météo sera une feature cruciale pour notre modèle.

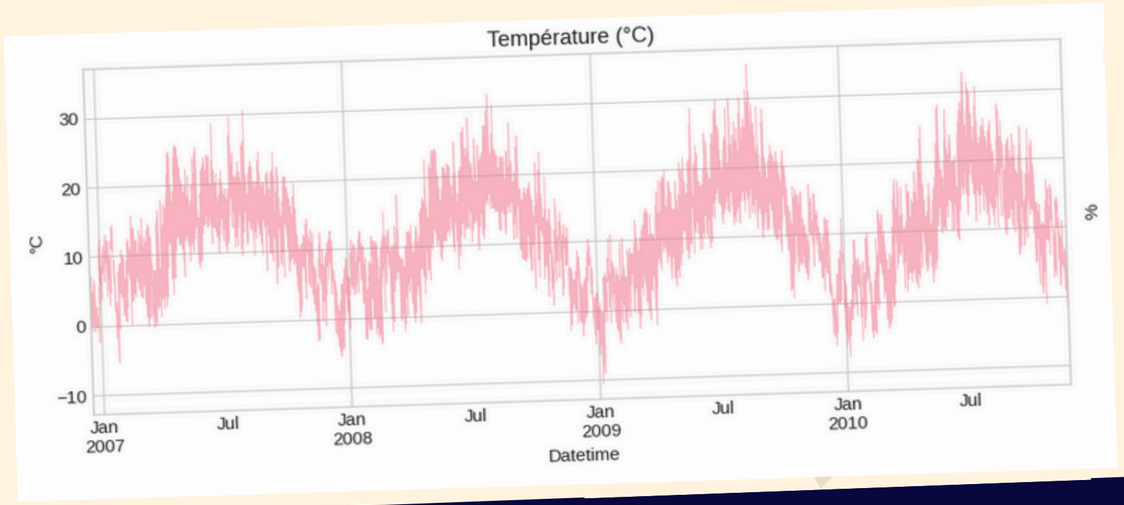
INTÉGRATION DES DONNÉES MÉTÉO

Ajouter le contexte climatique

J'ai enrichi le dataset avec des données météorologiques historiques via l'API Open-Meteo :

humidity	Humidité relative en %
temperature	Température horaire en °C





FEATURE ENGINEERING: L'ART CACHÉ DU ML

Pourquoi c'est si important ?

Le Feature Engineering consiste à créer de nouvelles variables à partir des données brutes.
C'est souvent ce qui fait la différence entre un modèle médiocre et un excellent modèle.

Mon approche :

34 features crées à partir de 9 variables brutes

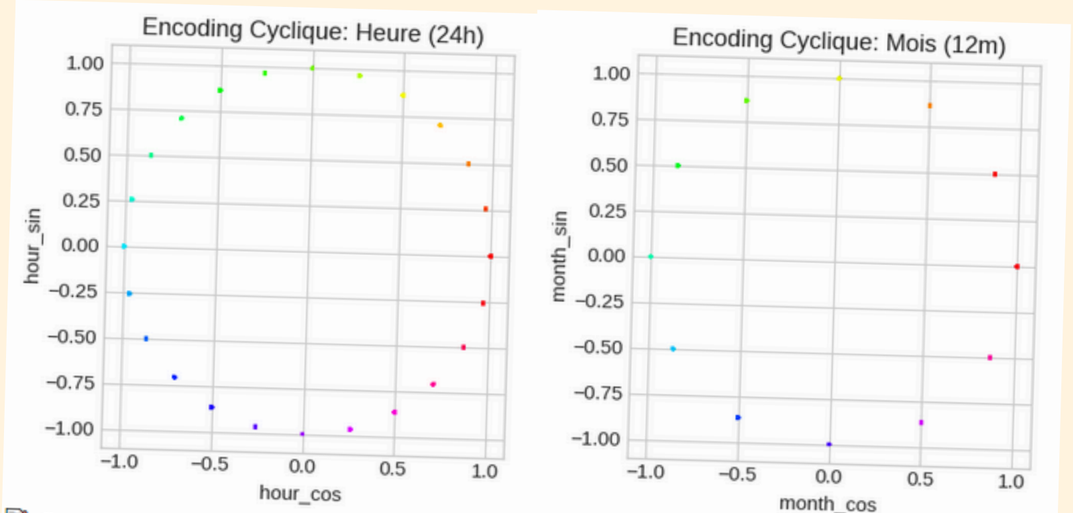
❌ SANS FEATURE ENGINEERING	✅ AVEC FEATURE ENGINEERING
Le modèle voit : 'heure = 19'	Le modèle voit : 'pic du soir probable'
Le modèle voit : 'température = 5°C'	Le modèle voit : 'besoin chauffage = 15 degrés'
Le modèle voit : 'lundi'	Le modèle voit : 'jour ouvré, même pattern que mardi-vendredi'

ENCODING CYCLIQUE : UNE ASTUCE ESSENTIELLE

Comment représenter le temps pour un réseau de neurones ?

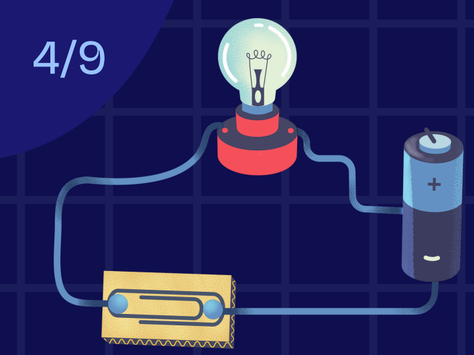
Problème : Pour un humain, 23h est proche de 0h. Mais pour un modèle, 23 et 0 sont très éloignés !

❌ ENCODING LINÉAIRE	✅ ENCODING CYCLIQUE
heure = 0, 1, 2, ... 23	hour_sin = sin(2π × heure/24)
Distance 23→0 = 23	hour_cos = cos(2π × heure/24)
Le modèle ne voit pas la continuité	Distance 23→0 = très proche !



Chaque point sur le cercle représente une valeur temporelle.
Les valeurs proches temporellement sont proches sur le cercle.

J'ai appliqué cet encoding à :
heure, jour de la semaine, mois



LAG FEATURES : LA MÉMOIRE DU PASSÉ

La consommation passée prédit la consommation future

Idée simple : si je connais ma consommation il y a 1 heure, 24 heures, ou 1 semaine, je peux mieux prédire maintenant.

LAG	VALEUR	CAPTURE
lag_1h, lag_2h, lag_3h	Consommation h-1, h-2, h-3	Tendance immédiate
lag_24h, lag_48h	Même heure hier, avant-hier	Pattern journalier
lag_168h, lag_336h	Même heure il y a 1-2 semaines	Pattern hebdomadaire

⚠ Attention au Data Leakage !

Pour prédire l'heure H, je ne peux utiliser que les données disponibles avant H.
Le lag_1h correspond donc à la consommation de l'heure H - 1, calculé avec shift(1).
Sans cette précaution, le modèle utiliserait des informations du futur — et les résultats seraient artificiellement trop bons.

ROLLING STATISTICS : CAPTURER LES TENDANCES

Moyennes et écarts-types glissants

Au lieu de regarder des points isolés, on regarde des fenêtres temporelles :

rolling_mean_6h	Moyenne des 6 dernières heures (tendance court terme)
rolling_mean_24h	Moyenne journalière (niveau de base)
rolling_mean_168h	Moyenne hebdomadaire (tendance long terme)
rolling_std_24h	Variabilité des 24 dernières heures (élevé = comportement irrégulier)
rolling_min/max_24h	Extrems : pics et creux récents

⚠ Attention au Data Leakage !

Encore une fois, j'applique shift(1) pour éviter le data leakage.

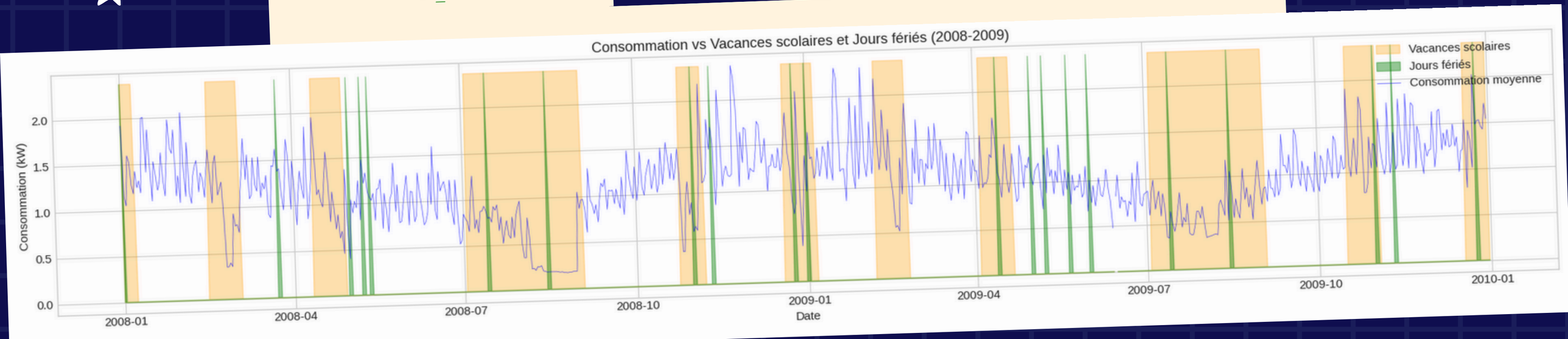
FEATURES CALENDAIRES : LE RYTHME DE VIE

Vacances, jours fériés et weekends influencent la consommation

Un mardi en août ne ressemble pas à un mardi en novembre.
Ces features permettent au modèle de distinguer les périodes "normales" des périodes "exceptionnelles".

Période	Observation	Explication
Vacances scolaires (zones orange)	Consommation souvent plus basse	Famille absente ou horaires décalés
Été 2008 (juillet-août)	Chute drastique (~0.3 kW)	Départ en vacances probable
Jours fériés (lignes vertes)	Pics ou creux ponctuels	Comportement type weekend

Feature	Description	Source
`is_weekend`	1 si samedi/dimanche, 0 sinon	DatetimeIndex
`is_holiday`	1 si jour férié français	Librairie holidays
`is_vacation`	1 si vacances scolaires (zone C)	Calendrier académique

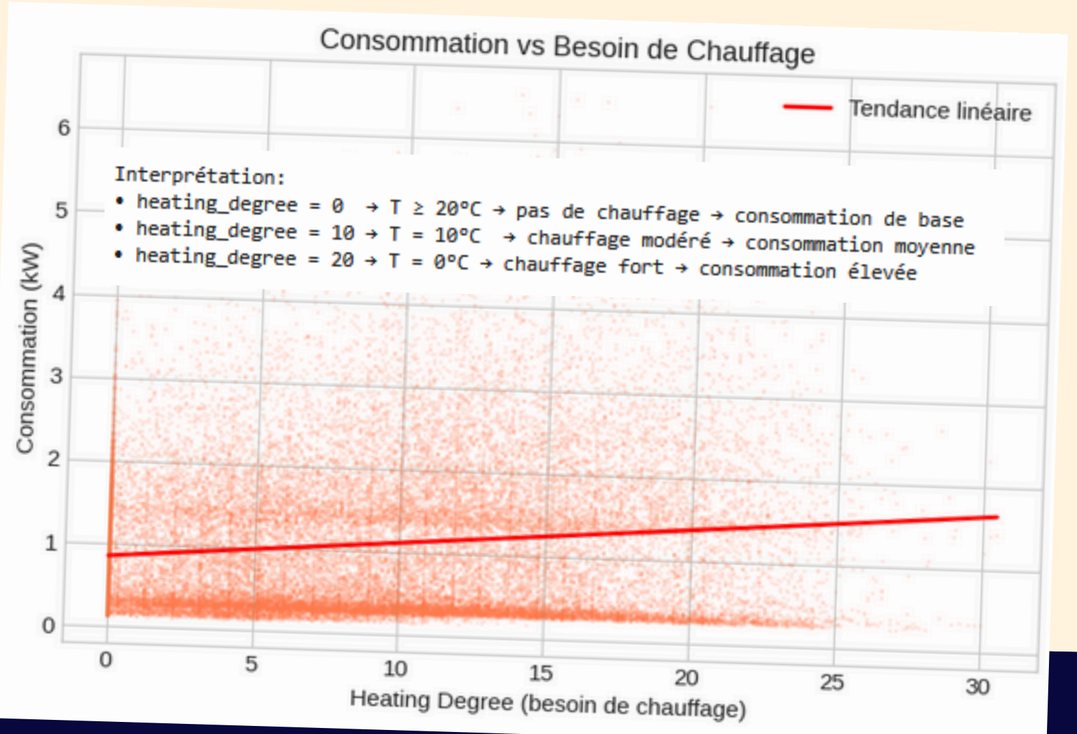


FEATURES MÉTÉO DÉRIVÉES

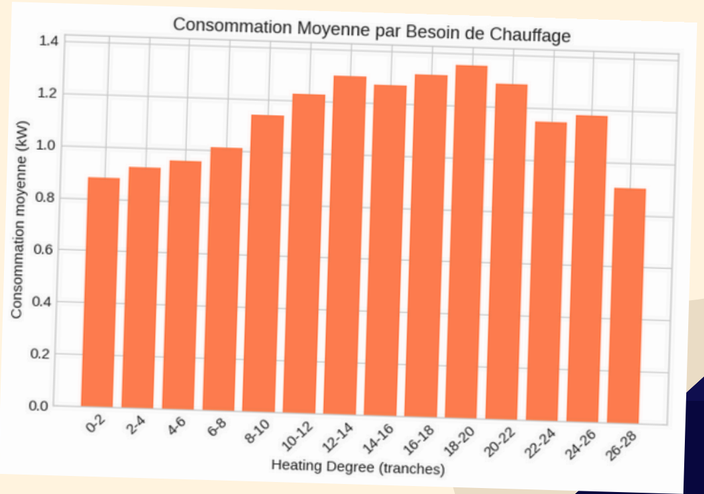
Traduire la météo en besoin énergétique

La température brute n'est pas optimale. Ce qui compte, c'est le besoin de chauffage :

heating_degree	max(0, 20 - température) → Plus il fait froid, plus c'est élevé
cold_humidity_index	heating_degree × humidité/100 → Froid humide = ressenti pire



Corrélation heating_degree - consommation: 0.187

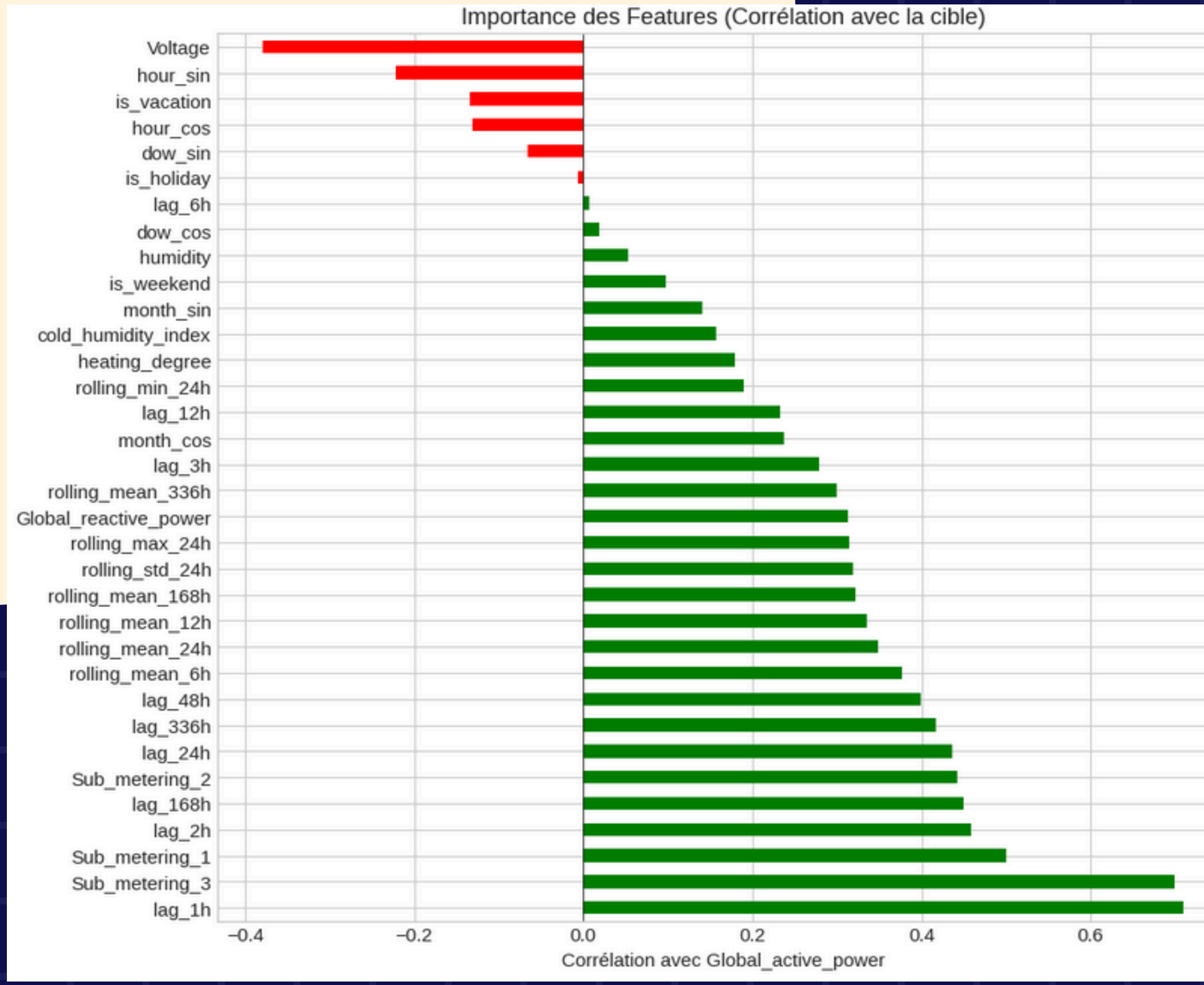


IMPORTANCE DES FEATURES

Quelles variables comptent vraiment ?

Top 5 des features les plus corrélées

Rang	Feature	Corrélation	Interprétation
1	lag_1h	0.71	Inertie immédiate
2	Sub_metering_3	0.70	Chauffage/clim
3	Sub_metering_1	0.50	Cuisine
4	lag_2h	0.49	Tendance récente
5	lag_168h	0.45	Pattern hebdo



PRÉPARATION DES SÉQUENCES

Transformer les données en entrées pour le Deep Learning

Les modèles LSTM et Transformer ont besoin de séquences structurées :

PARAMÈTRE	VALEUR	SIGNIFICATION
LOOKBACK	336 h	14 jours d'historique
HORIZON	24 h	1 jour de prévision
n_features	34	Variables d'entrée

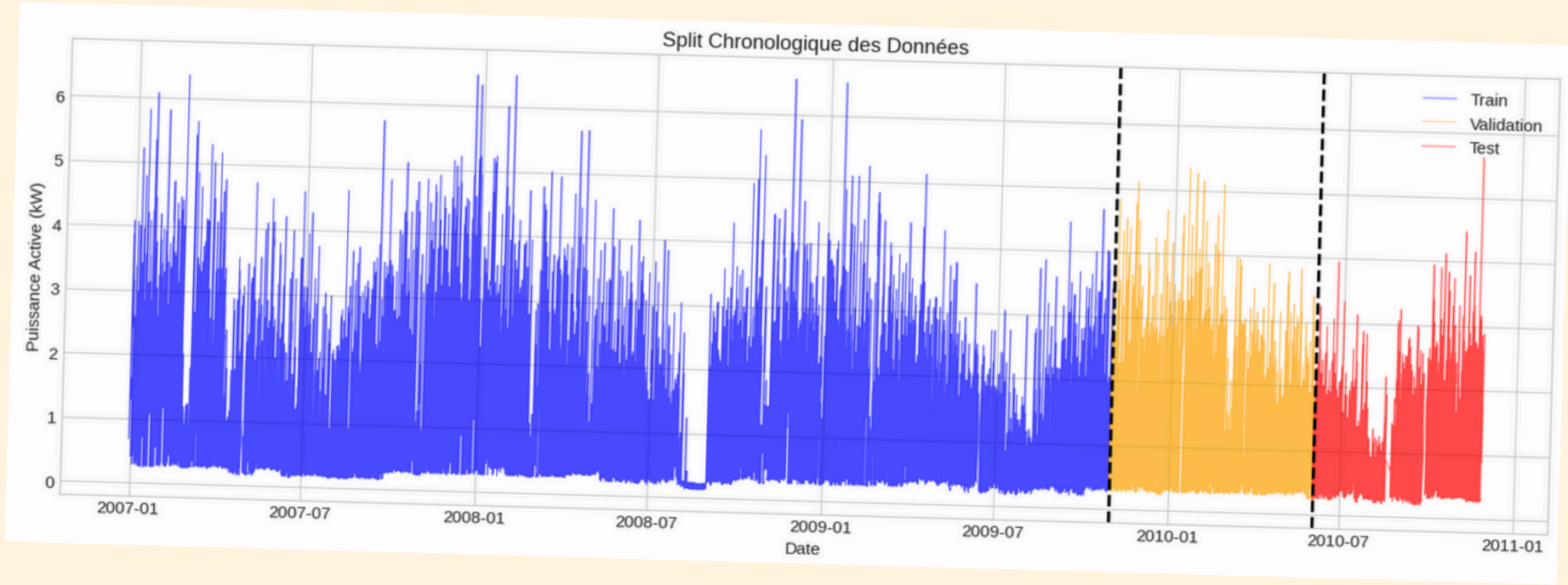
Forme des données

X (input) : (n_séquences, 336, 34) → 'Je regarde 14 jours de 34 variables'
y (output) : (n_séquences, 24) → 'Je prédis les 24 prochaines heures'

Pourquoi 14 jours ? Pour capturer 2 cycles hebdomadaires complets.

SPLIT TEMPOREL : RESPECTER LE TEMPS

Ne jamais mélanger passé et futur



Pourquoi pas de shuffle ?

En séries temporelles, on ne mélange **JAMAIS** les données. Le modèle doit apprendre sur le passé et être évalué sur le futur, comme en production réelle.

Set	Ratio	Séquences	Période
Train	73%	24 338	2006 - fin 2009
Validation	15%	4 716	fin 2009 - mi-2010
Test	12%	3 701	mi-2010 - fin 2010

NORMALISATION : MISE À L'ÉCHELLE

Pourquoi et comment normaliser ?

Les réseaux de neurones fonctionnent mieux quand toutes les variables sont sur la même échelle.

Sans normalisation	Avec MinMaxScaler (0-1)
température : -5 à 35	température : 0 à 1
consommation : 0 à 11	consommation : 0 à 1
humidity : 20 à 100	humidity : 0 à 1

⚠️ Règle d'or :

Le scaler est ajusté (fit) **UNIQUEMENT** sur le train set, puis appliqué (transform) sur validation set et test set.

Sinon = **data leakage** (on utilise des informations du futur pour calibrer).

LSTM : LE MODÈLE BASELINE

Long Short-Term Memory : Comprendre le concept

Le LSTM est un type de réseau de neurones récurrent (RNN) conçu pour apprendre les dépendances à court et long terme dans les séquences.

Analogie simple :

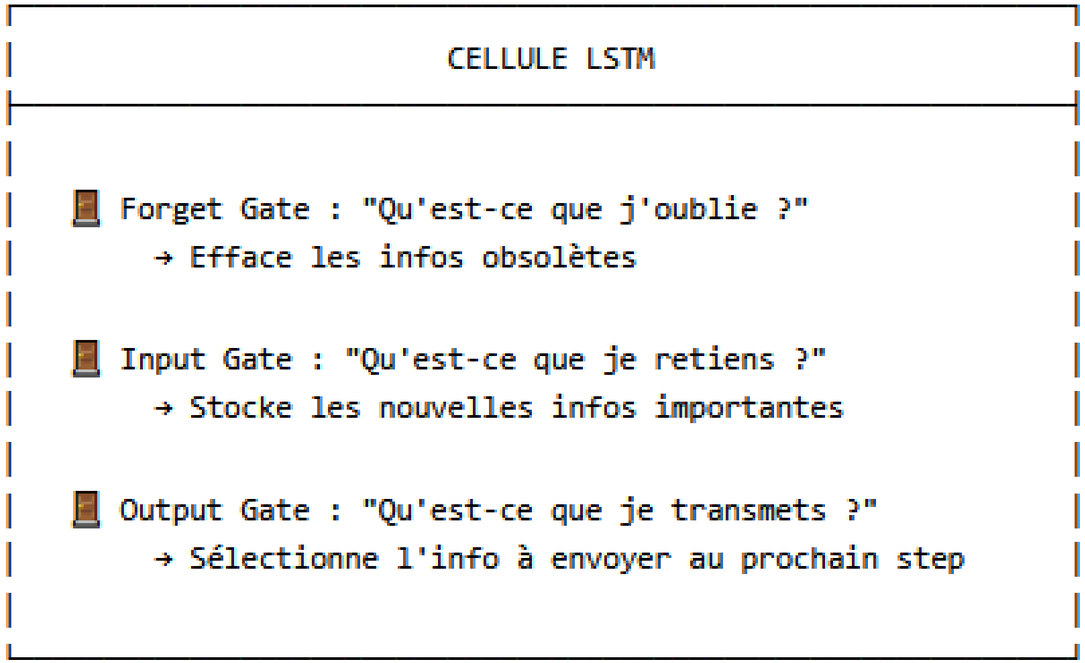
Imaginez lire un livre page par page en essayant de vous souvenir du début.

Le LSTM a une 'mémoire' qui lui permet de retenir les informations importantes.

Pourquoi LSTM pour les séries temporelles ?

- Conçu spécifiquement pour les séquences
- Standard industriel depuis 1997
- Capture les dépendances temporelles

Le LSTM possède des **portes** (gates) qui contrôlent le flux d'information :



ARCHITECTURE DE MON LSTM

Choix et justifications

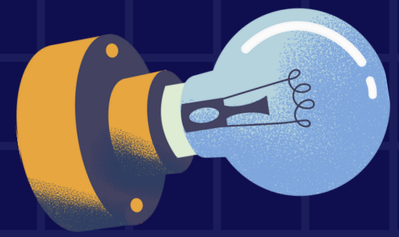
Couche	Configuration	Rôle
LSTM 1	64 unités, return_seq=True	Extraction patterns
Dropout 1	33%	Anti-overfitting
LSTM 2	32 unités, return_seq=False	Condensation info
Dropout 2	33%	Anti-overfitting
Dense	24 neurones	Output (24h)

Régularisation L2 = 0.004 sur chaque couche LSTM

```
============================================================
ARCHITECTURE DU MODÈLE LSTM
============================================================
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 336, 64)	25,344
dropout (Dropout)	(None, 336, 64)	0
lstm_1 (LSTM)	(None, 32)	12,416
dropout_1 (Dropout)	(None, 32)	0
dense (Dense)	(None, 24)	792

```
Total params: 38,552 (150.59 KB)
Trainable params: 38,552 (150.59 KB)
Non-trainable params: 0 (0.00 B)
```



RÉGULARISATION : LUTTER CONTRE L'OVERFITTING

Trois techniques combinées

L'overfitting, c'est quand le modèle 'apprend par cœur' le train set au lieu de généraliser.

TECHNIQUE	EXPLICATION SIMPLE
Dropout	Éteindre aléatoirement 33% des neurones à chaque batch → force le réseau à ne pas dépendre d'un seul chemin
L2	Pénaliser les gros poids → empêche le modèle de trop s'ajuster aux détails
Early Stopping	Arrêter l'entraînement quand la validation stagne → évite de trop apprendre

⚠ Le Dropout rend les résultats légèrement aléatoires

Mon approche

J'ai combiné les trois techniques. C'est ce qui m'a permis d'obtenir un overfitting de seulement 1.63% avec le LSTM.

CALLBACKS : AUTOMATISER L'ENTRAÎNEMENT

Trois gardiens pour un entraînement optimal

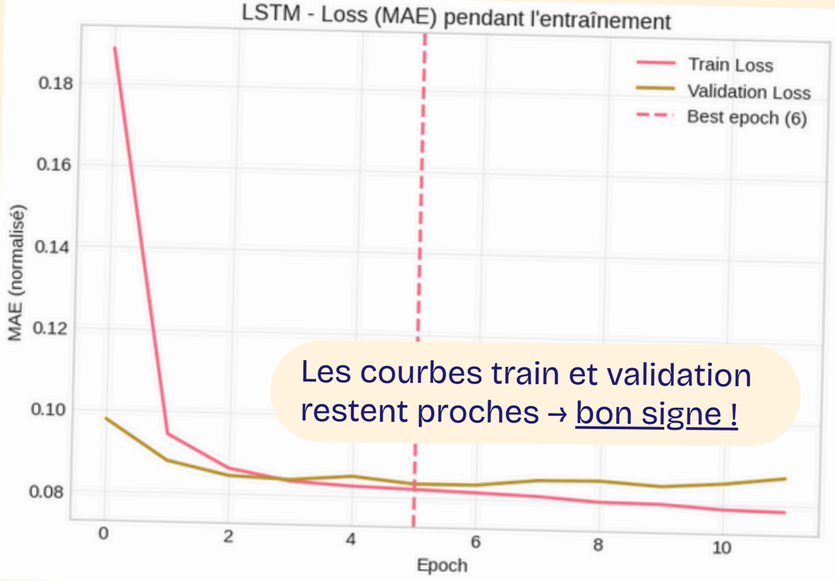
CALLBACK	PARAMÈTRES	EFFET
EarlyStopping	patience=6	Arrête si val_loss ne s'améliore pas pendant 6 epochs
ReduceLROnPlateau	factor=0.5, patience=2	Divise le learning rate par 2 si stagnation
ModelCheckpoint	save_best_only=True	Sauvegarde le meilleur modèle automatiquement

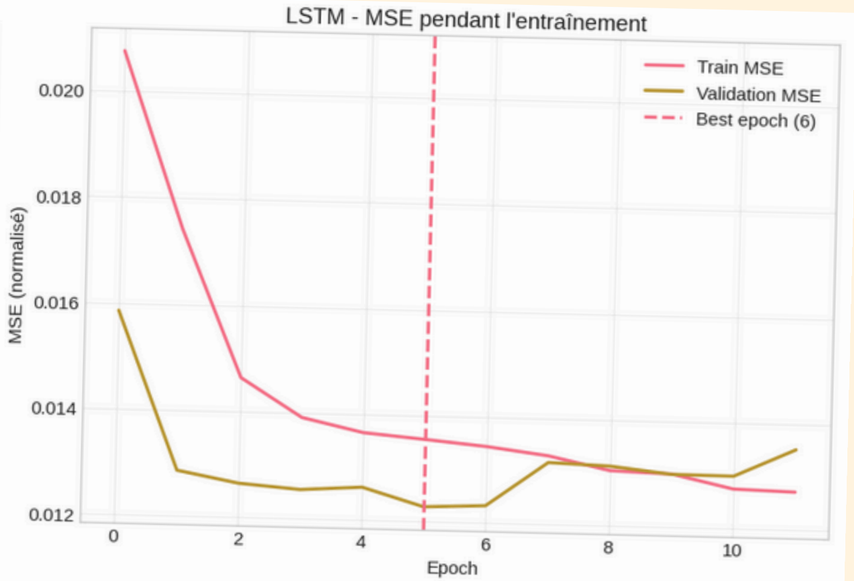
Pourquoi c'est important ?

Sans callbacks, il faudrait surveiller manuellement l'entraînement et deviner quand arrêter.
Avec callbacks, l'entraînement s'optimise tout seul.

ENTRAÎNEMENT DU LSTM

Résultats de l'entraînement





Les courbes train et validation restent proches → bon signe !

Métrique	Valeur
Epochs effectuées	12 (early stopping à epoch 6)
Temps d'entraînement	1.8 minutes
Meilleure val_loss	0.0835 (normalisé)
Train loss (best epoch)	0.0821 (normalisé)
Écart absolu	0.0013
Overfitting	1.63%

TRANSFORMER : L'ARCHITECTURE RÉVOLUTIONNAIRE

L'innovation qui a changé l'IA

Le Transformer (2017) est l'architecture derrière ChatGPT, BERT, et la plupart des IA modernes.
Son innovation clé : le mécanisme d'attention.

Analogie : LSTM vs Transformer

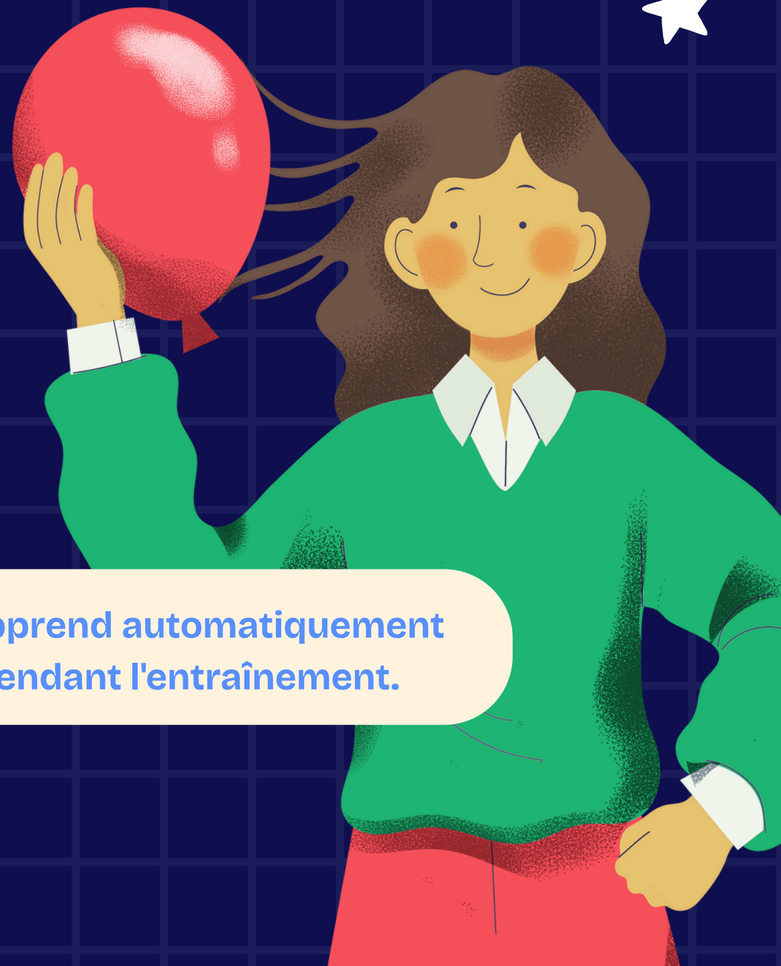
LSTM	Transformer
Lit le livre page par page	Toutes les pages étalées sur une table
Doit mémoriser pour se souvenir	Peut regarder n'importe quelle page instantanément
h1 → h2 → h3 → ... → h336 (séquentiel)	h1 ↔ h336 directement (parallèle)

LE MÉCANISME D'ATTENTION

'À quoi dois-je faire attention ?'

Pour prédire mardi 19h, le Transformer calcule un score d'attention avec toutes les heures passées :

Heure passée	Score	Pourquoi ?
Lundi 19h (hier)	0.25	Même heure, très pertinent
Mardi dernier 19h	0.30	Même jour/heure semaine passée
Mardi 18h (1h avant)	0.15	Tendance immédiate
Samedi 14h (random)	0.02	Peu pertinent



Le modèle apprend automatiquement ces scores pendant l'entraînement.

ARCHITECTURE DE MON TRANSFORMER

Adapté aux séries temporelles

Composant	Configuration	Rôle
Positional Encoding	Sin/Cos	Position temporelle
Multi-Head Attention	4 têtes	4 perspectives différentes
Encoder Blocks	2 blocs	Traitement en profondeur
Feed-Forward	128 neurones	Transformation non-linéaire
Dropout	35%	Régularisation

Régularisation L2 = 0.0055

```
================================================================
ARCHITECTURE DU MODÈLE TRANSFORMER
================================================================
Model: "functional_14"
```

Layer (type)	Output Shape	Param #
input_layer_7 (InputLayer)	(None, 336, 34)	0
dense_13 (Dense)	(None, 336, 64)	2,240
positional_encoding_2 (PositionalEncoding)	(None, 336, 64)	0
transformer_encoder_block_4 (TransformerEncoderBlock)	(None, 336, 64)	33,472
transformer_encoder_block_5 (TransformerEncoderBlock)	(None, 336, 64)	33,472
get_item_2 (GetItem)	(None, 64)	0
dense_18 (Dense)	(None, 24)	1,560

```
Total params: 70,744 (276.34 KB)
Trainable params: 70,744 (276.34 KB)
Non-trainable params: 0 (0.00 B)
```

2× plus que LSTM

= plus de risque d'**overfitting** sur un dataset de taille modeste

(voir expérimentation avec Transformer allégé)

J'ai voulu tester le Transformer dans une configuration "standard" comparable à la littérature. Réduire trop les paramètres risquait de créer un Transformer "bridé" qui ne représente pas vraiment l'architecture. La data augmentation permet de garder l'architecture intacte tout en luttant contre l'overfitting.

DATA AUGMENTATION : PLUS DE DONNÉES

Technique appliquée uniquement au Transformer

Le Transformer a plus de paramètres → besoin de plus de données pour bien généraliser.
J'ai doublé le training set avec des transformations légères :

TRANSFORMATION	PROBABILITÉ	DESCRIPTION
Scaling	50%	Multiplier X par un facteur entre 0.93 et 1.07
Time Shift	20%	Décaler la séquence de ±1 heure
Jitter	30%	Ajouter un bruit gaussien léger (std=0.003)

⚠️ Règle importante

Le target Y n'est JAMAIS modifié, seulement dupliqué.
On ne veut pas apprendre à prédire des valeurs fausses !

UN TRANSFORMER ENCODER-ONLY

Un choix adapté à notre problème

Le Transformer original (2017) utilise un **Encoder + Decoder.**
Pour la prévision énergétique, j'ai utilisé uniquement l'Encoder.

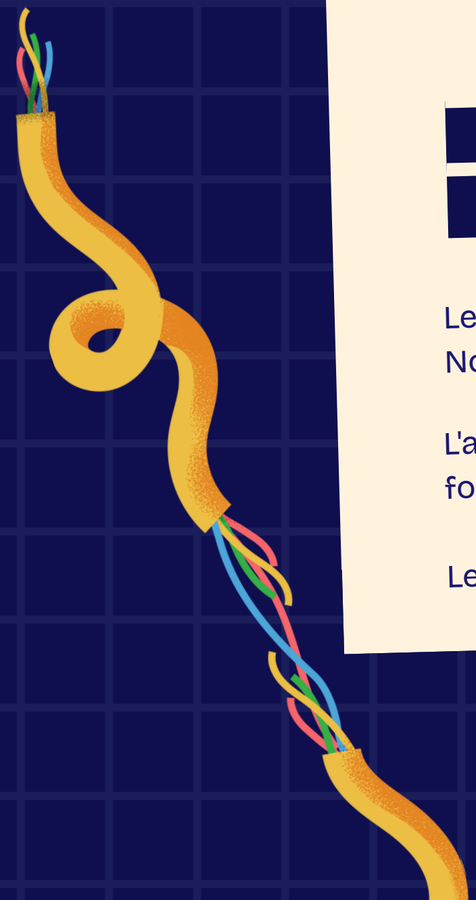
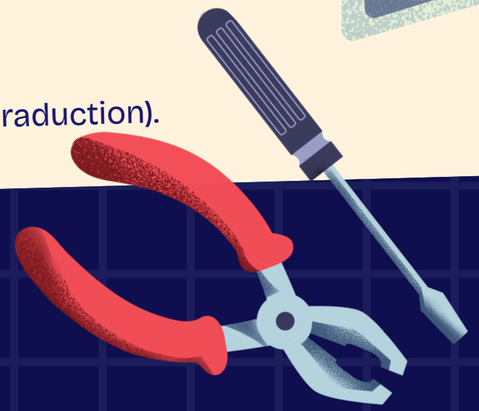
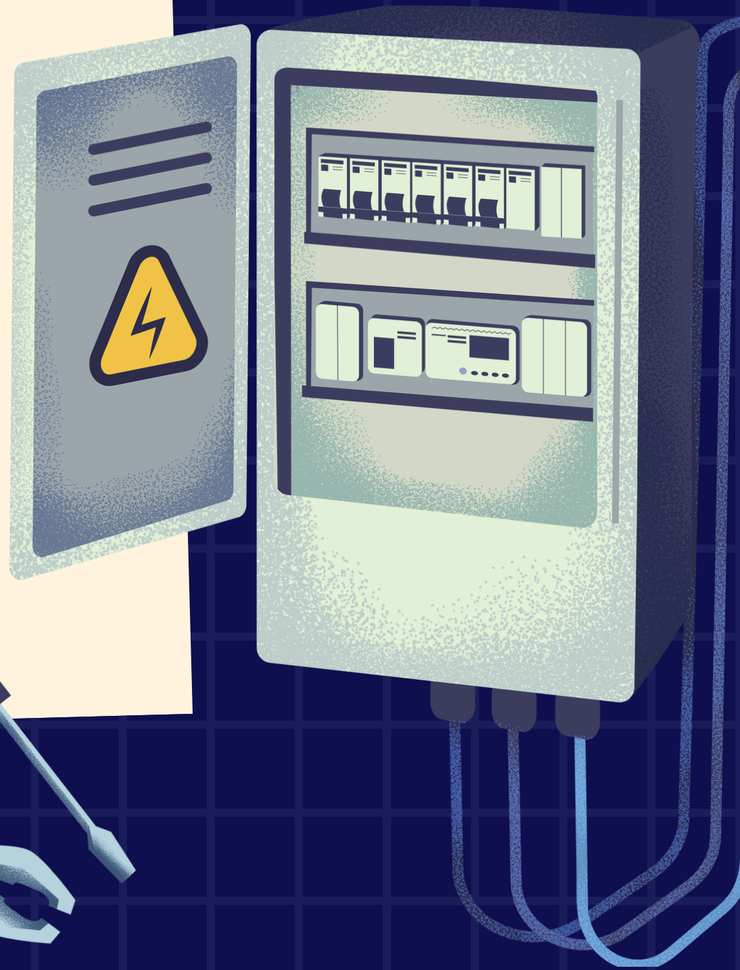
À quoi servent Encoder et Decoder ?

Composant	Rôle	Exemple
Encoder	Comprendre l'entrée	Lire et analyser 336h d'historique
Decoder	Générer une sortie mot par mot	Écrire une phrase, un mot à la fois

Le Decoder est utile quand on génère une séquence élément par élément.
Nous, on calcule 24 valeurs directement à partir de l'historique. **L'Encoder suffit.**

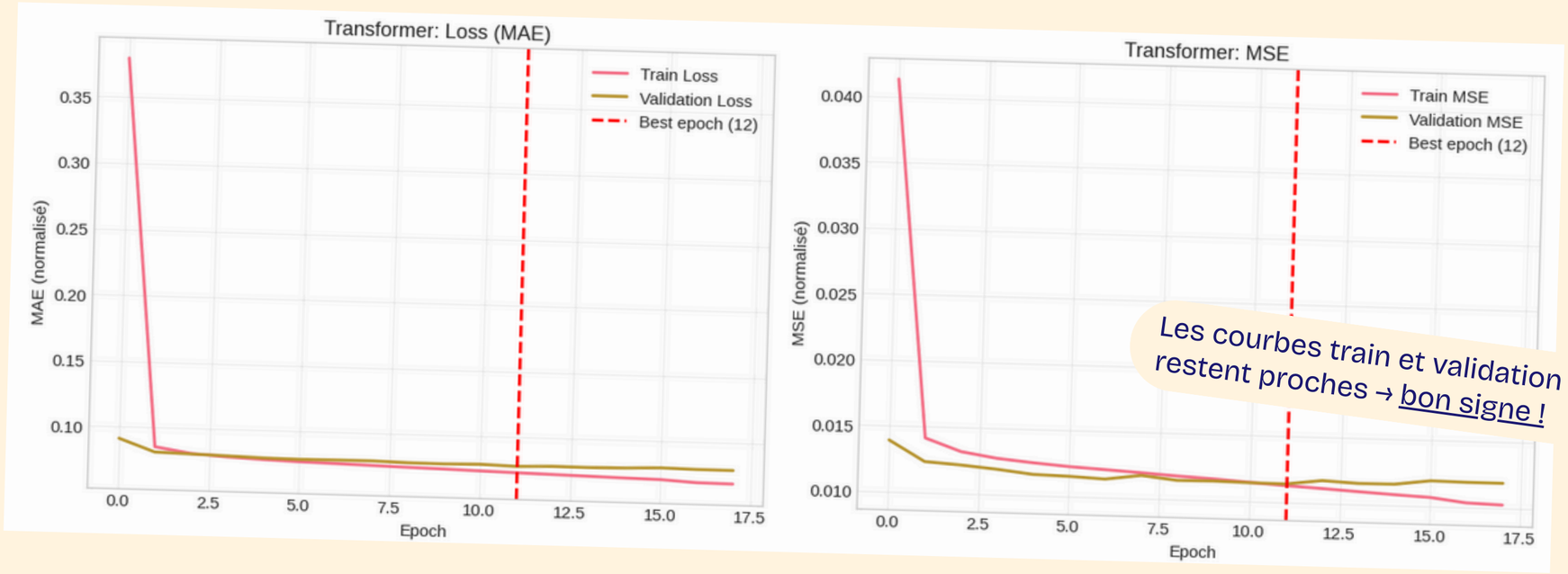
L'approche **encoder-only** est courante dans la littérature scientifique pour le forecasting de séries temporelles.

Le decoder est surtout utile pour la génération auto-régressive (texte, traduction).

ENTRAÎNEMENT DU TRANSFORMER

Résultats de l'entraînement



Les courbes train et validation restent proches → bon signe !

Métrique	Valeur
Epochs effectuées	18 (early stopping à epoch 12)
Temps d'entraînement	7.3 minutes
Meilleure val_loss	0.0781 (normalisé)
Train loss (best epoch)	0.0732 (normalisé)
Écart absolu	0.0048
Overfitting	6.62%

Le Transformer est plus long à entraîner et montre plus d'overfitting que le LSTM.

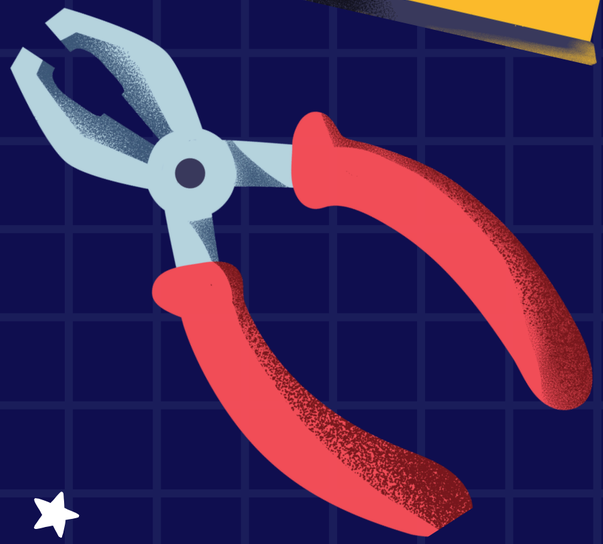
MÉTRIQUES D'ÉVALUATION

Comment mesurer la performance ?

MÉTRIQUE	FORMULE SIMPLIFIÉE	INTERPRÉTATION
MAE	Moyenne(|réel - prédit|)	Erreur moyenne en kW - MÉTRIQUE PRINCIPALE
RMSE	√Moyenne((réel - prédit)²)	Pénalise plus les grosses erreurs
R^2	1 - (erreur/variance)	Qualité globale (1.0 = parfait)
MAPE	Moyenne(|erreur|/réel) × 100	Erreur en % - sensible aux petites valeurs

Pourquoi MAE comme métrique principale ?

- Directement interprétable en kW
- Correspondance directe avec l'objectif business
- Moins sensible aux outliers que RMSE



⚠️ MAPE : quand la consommation réelle est faible (ex: 0.1 kW la nuit), même une petite erreur absolue donne un pourcentage énorme.

RÉSULTATS : LSTM VS TRANSFORMER

La bataille finale

Métrique	LSTM	Transformer	Gagnant
MAE	0.4145 kW	0.4086 kW	Transformer (+1.4%)
RMSE	0.5997 kW	0.5850 kW	Transformer
R^2	0.2941	0.3285	Transformer
MAPE	52.94%	54.52%	LSTM
Overfitting	1.63%	6.62%	LSTM (4× moins)
Paramètres	38 552	70 744	LSTM (2× moins)
Training time	1.8 min	7.3 min	LSTM (4× plus rapide)

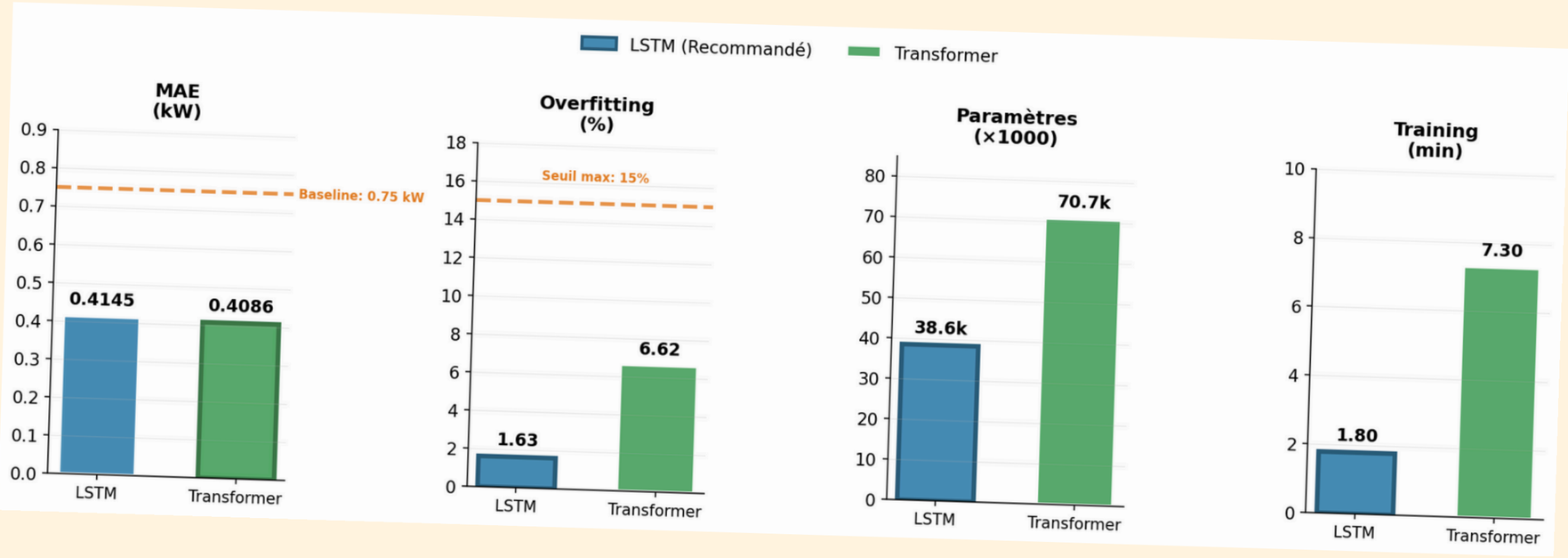
Score final :

LSTM 4/7

Transformer 3/7

COMPARAISON VISUELLE

Vue d'ensemble des performances



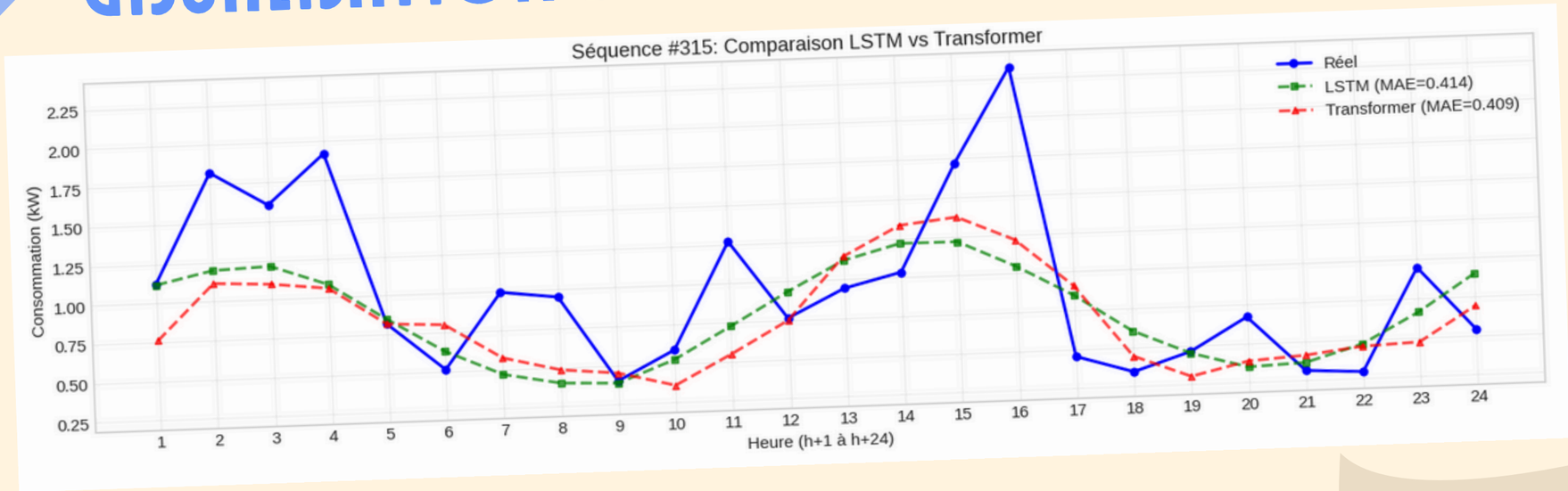
MAE	Les deux modèles battent largement la baseline de 0.75 kW (ligne orange)
Overfitting	Le LSTM reste très loin du seuil critique de 15% — le Transformer est 4× plus proche
Paramètres	Le LSTM est 2× plus léger → **(voir expérimentation avec Transformer allégé plus avant).**
Training	1.8 min vs 7.3 min → réentraînement hebdomadaire facilement envisageable

Conclusion visuelle

Le Transformer gagne sur la précision brute (+1.4%), mais le LSTM domine sur tous les critères opérationnels.

VISUALISATION DES PRÉDICTIONS

Prédictions vs Réalité



Observations clés :

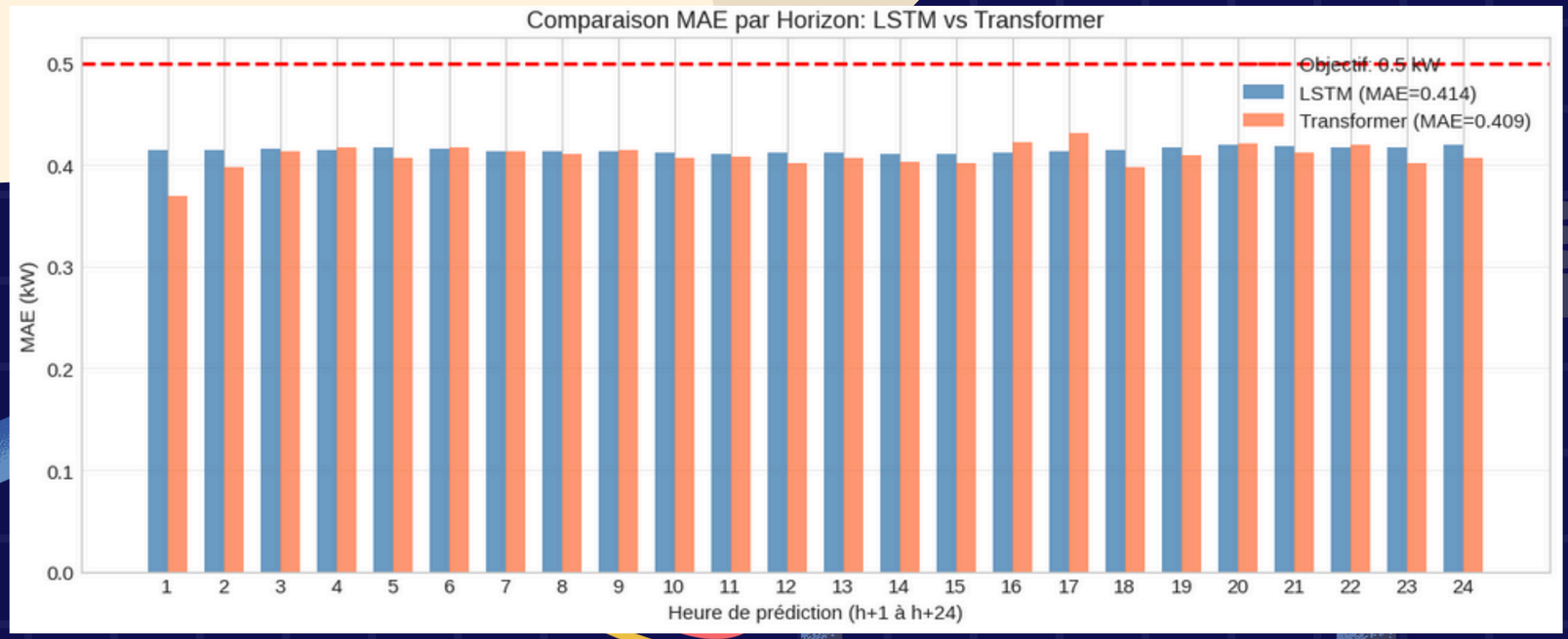
- Les deux modèles suivent bien les tendances générales
- Les pics sont systématiquement sous-estimés
- Les très faibles consommations nocturnes restent difficiles à prédire

PERFORMANCE PAR HORIZON TEMPOREL

Comment évolue l'erreur de h+1 à h+24 ?

HORIZON	MAE LSTM	MAE TRANSFORMER
h+1 (1ère heure)	~0.42 kW	~0.37 kW
h+12 (mi-journée)	~0.41 kW	~0.40 kW
h+24 (fin)	~0.43 kW	~0.41 kW
Dégradation	+2.4%	+11.0%

Le Transformer démarre mieux (h+1) mais se dégrade significativement vers h+24. Le LSTM reste remarquablement stable sur tout l'horizon. C'est un argument de plus pour recommander le LSTM en production : sa fiabilité est constante.



LIMITES IDENTIFIÉES

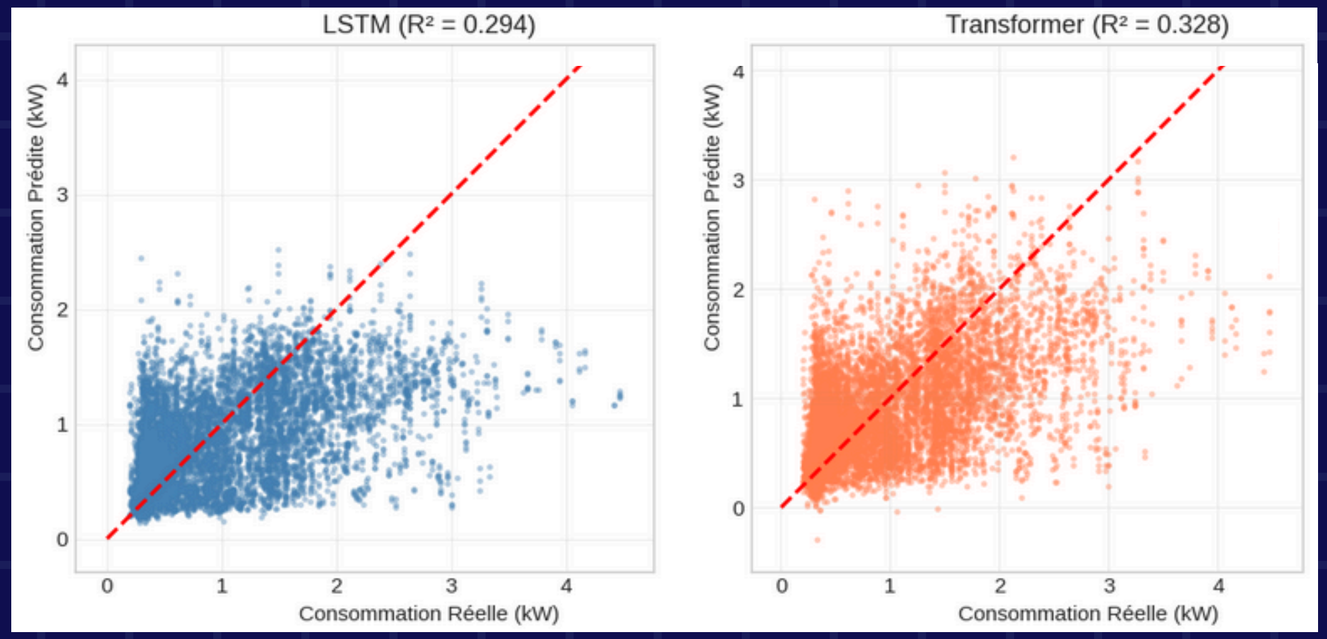
Ce que les modèles ne font pas bien

1. Sous-estimation des pics :

Les deux modèles peinent à prédire les consommations > 2.5 kW
Raison : 70% des données sont < 1.5 kW → le modèle optimise pour la majorité

2. Autres limites :

- Données d'un seul foyer → généralisation à 50 000 foyers à valider
- Période 2006-2010 → patterns de consommation ont évolué
- Météo reconstituée via API, non mesurée sur site

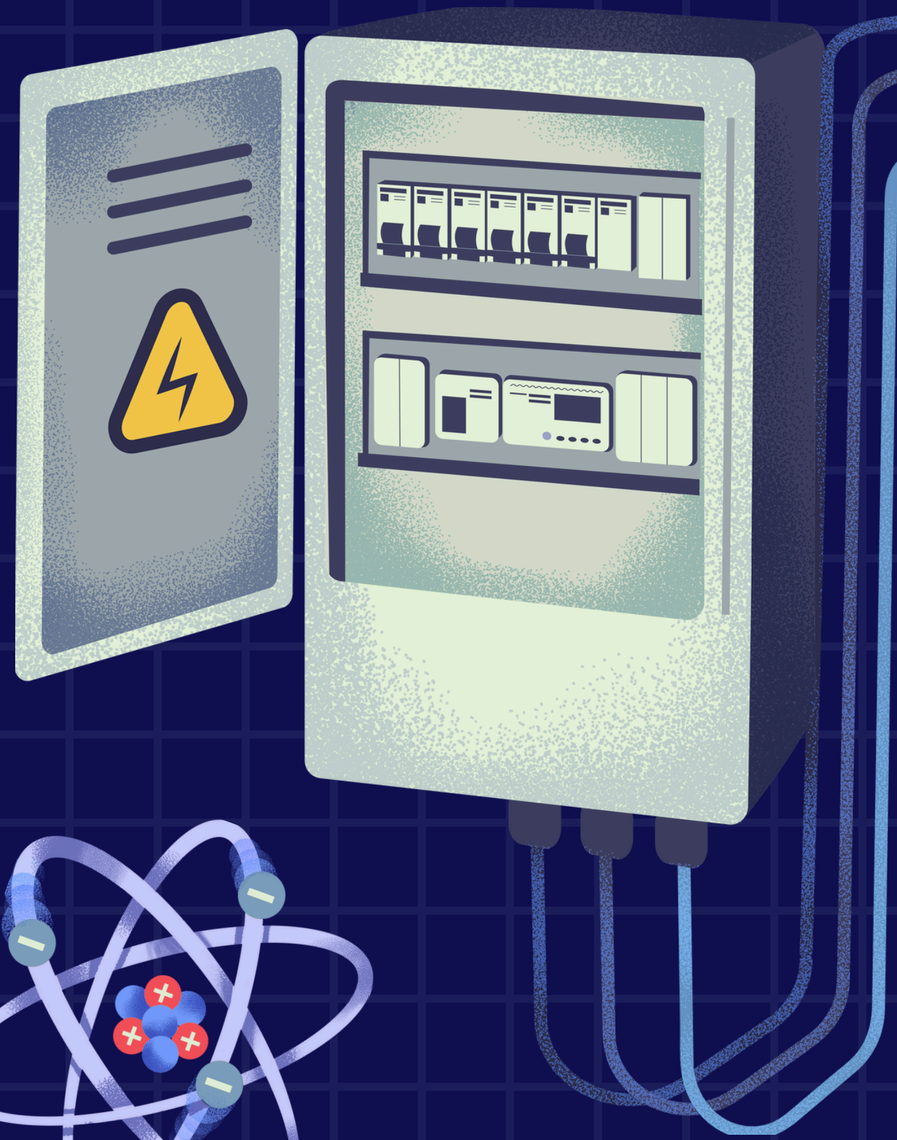


RECOMMANDATION : DÉPLOYER LE LSTM

Pourquoi ?

Critère	Argument
Robustesse	Overfitting 4× inférieur → meilleure performance en production
Rapidité	4× plus rapide → réentraînement fréquent possible
Performance	MAE 0.4145 kW < 0.5 kW → objectif atteint ✓

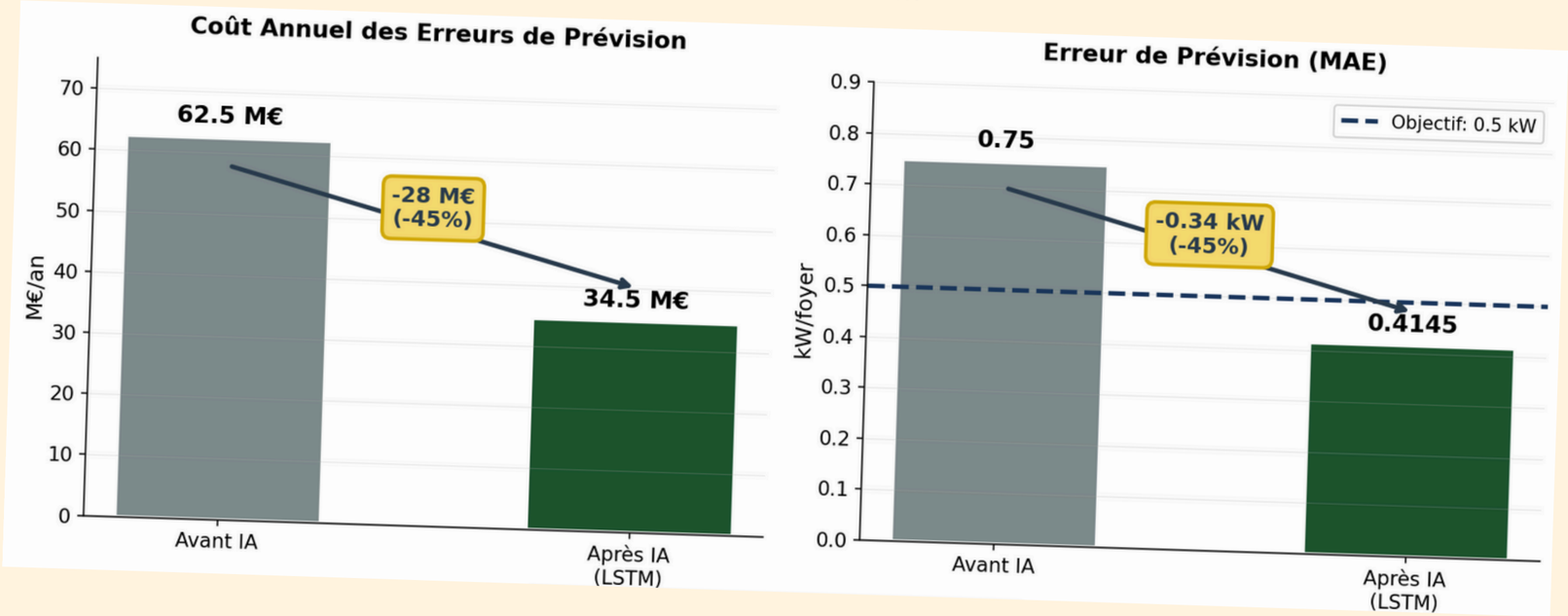
Le gain marginal du Transformer (+1.4%) ne justifie pas sa complexité et son risque d'overfitting.



IMPACT BUSINESS

Traduction en euros

Le graphique montre la réduction obtenue grâce au modèle LSTM :



La ligne bleue pointillée sur le graphique MAE indique l'objectif de 0.5 kW

—> notre modèle passe largement en dessous.

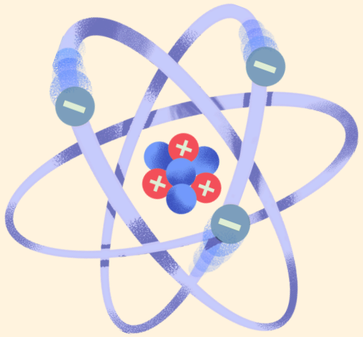
Indicateur	Avant IA	Après IA (LSTM)
Erreur de prévision (MAE)	0.75 kW	0.4145 kW
Réduction de l'erreur	-	-45%
Coût annuel des erreurs	62.5 M€	~34.5 M€
Économies annuelles	-	~28 M€

Objectifs atteints :

- Objectif technique : MAE 0.41 kW < 0.5 kW ✓
- Objectif business : dans la fourchette 30-40 M€ ✓

CE QUE J'AI APPRIS

Leçons de mon premier projet Deep Learning



Leçons techniques :

- Le data leakage est le piège n°1 en séries temporelles → shift(1) partout !
- L'overfitting se combat avec Dropout + L2 + Early Stopping combinés
- Le feature engineering apporte autant que l'architecture du modèle
- Le modèle le plus récent n'est pas toujours la meilleur choix

Surprise !

Le LSTM de 1997 bat le Transformer de 2017 sur ce dataset ! La simplicité paie.

Leçons méthodologiques :

- Comprendre les données avant de modéliser (EDA approfondie)
- Commencer simple, complexifier si nécessaire
- Documenter chaque choix et sa justification
- Être honnête sur les limites

BONUS : TRANSFORMER ALLÉGÉ

Et si on réduisait les paramètres du Transformer ?

Après avoir terminé le projet, une question me restait : le Transformer avait-il trop de paramètres pour notre dataset ? J'ai donc testé une version allégée.

Configuration testée :

Paramètre	Version originale	Version allégée
d_model	64	32
ff_dim	128	64
num_blocks	2	1
Total paramètres	70 744	**10 456**

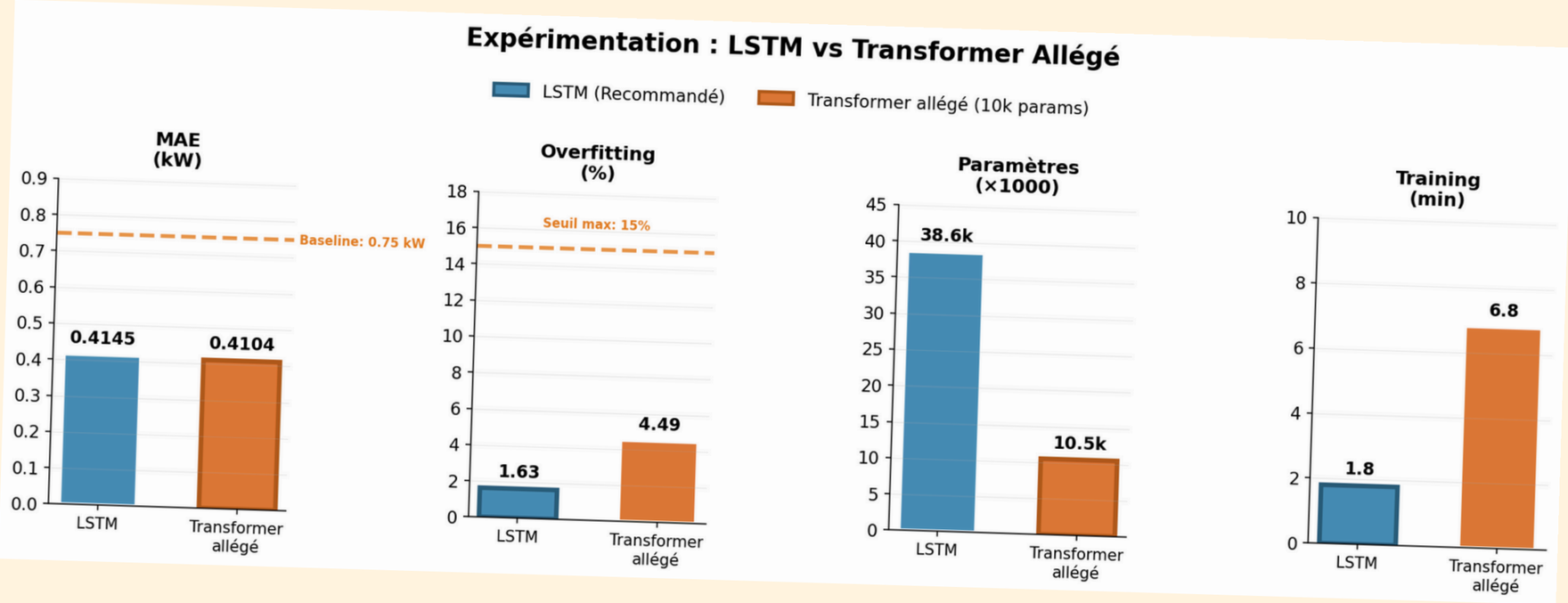
```
============================================================
ARCHITECTURE DU MODÈLE TRANSFORMER
============================================================
Model: "functional_11"
```

Layer (type)	Output Shape	Param #
input_layer_5 (InputLayer)	(None, 336, 34)	0
dense_9 (Dense)	(None, 336, 32)	1,120
positional_encoding_2 (PositionalEncoding)	(None, 336, 32)	0
transformer_encoder_block_2 (TransformerEncoderBlock)	(None, 336, 32)	8,544
get_item_2 (GetItem)	(None, 32)	0
dense_12 (Dense)	(None, 24)	792

```
Total params: 10,456 (40.84 KB)
Trainable params: 10,456 (40.84 KB)
Non-trainable params: 0 (0.00 B)
```

moins paramètres

COMPARAISON VISUELLE avec le Transformer allégé



- ✅ L'overfitting diminue (6.62% → 4.49%) — **le modèle original était bien surdimensionné**
- ❌ Le MAE reste excellente, mais équivalent à celui du LSTM
- ❌ Le LSTM reste plus robuste (overfitting 3× inférieur)...
- ❌ ...et plus rapide (4× plus rapide)
- ❌ Le temps de training ne baisse pas significativement (6.8 min vs 7.3 min)

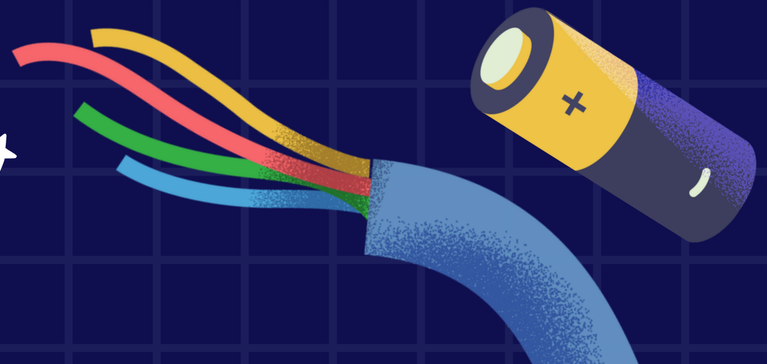
Le verdict final reste inchangé :

le LSTM est le meilleur choix pour ce cas d'usage.

ET APRÈS ? MES PERSPECTIVES

Comment je compte faire évoluer ce projet

Horizon	Recommandations
Court terme	• Intégrer météo temps réel (API Météo France) • Tester sur données multi-foyer • Optimiser hyperparamètres avec Optuna
Moyen terme	• Prévision probabiliste avec intervalles de confiance (Quantile LSTM) • Segmentation des foyers par profil de consommation
Long terme	• Explorer **Informer et Autoformer** (architectures spécialisées séries temporelles) • Créer des modèles par segment de foyers • Ajouter détection d'anomalies



Pourquoi Informer/Autoformer ?

Ce sont des Transformers optimisés pour les séries temporelles longues (2020-2021).

Ils pourraient combiner la précision du Transformer avec la robustesse du LSTM.

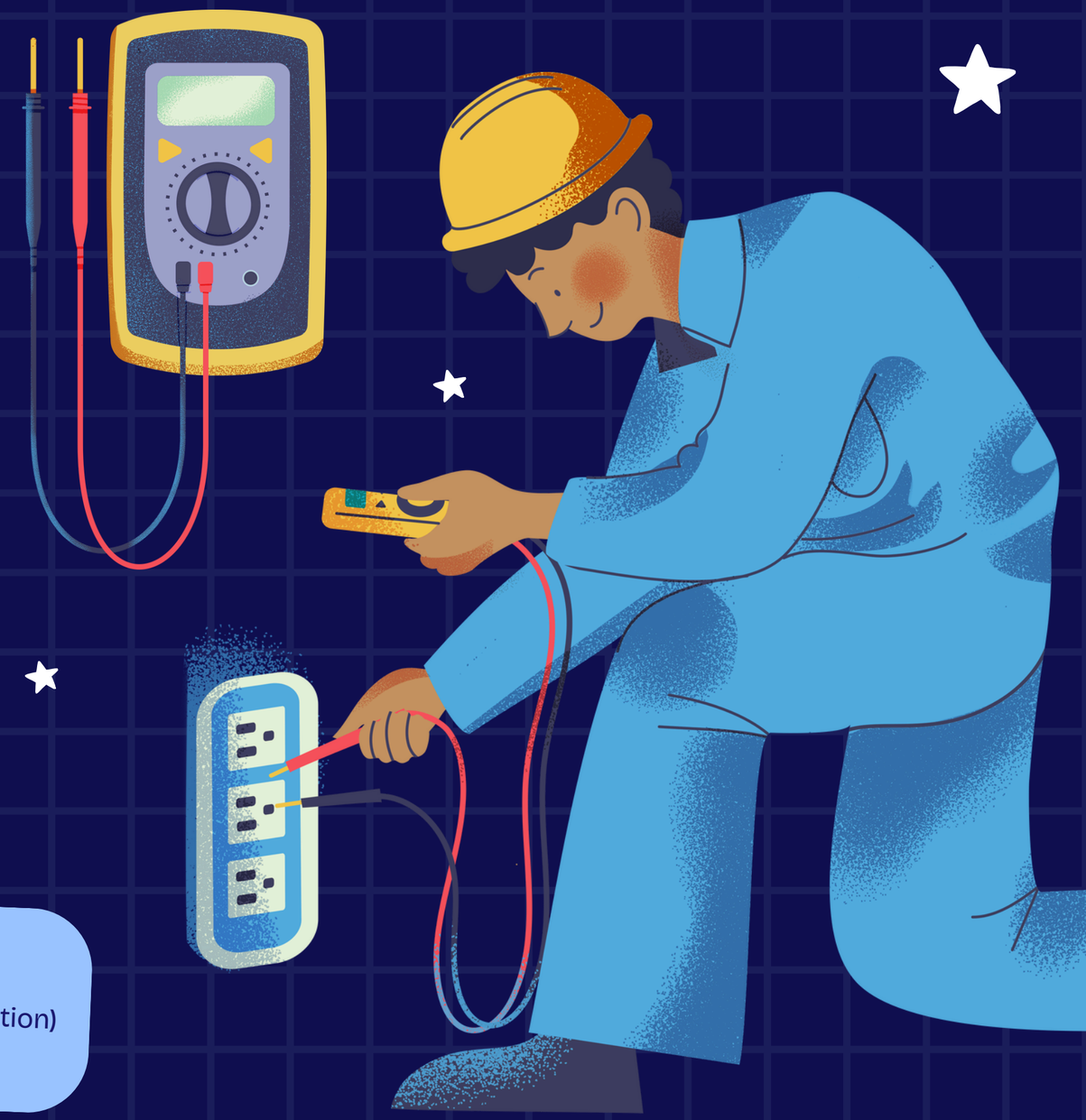
RECOMMANDATIONS POUR AI ENERGY SOLUTIONS

Plan d'action proposé

Priorité	Action
1	Phase pilote : Déployer le LSTM sur 5 000 foyers pendant 3 mois
2	Intégration : Connecter l'API Météo France pour données temps réel
3	Monitoring : Dashboard de suivi MAE quotidien avec alertes si > 0.5 kW
4	Pipeline : Automatiser le réentraînement hebdomadaire
5	Évolution : Tester Informer/Autoformer sur données réelles

Infrastructure requise :

CPU suffisant pour l'inférence (pas besoin de GPU en production)
GPU T4 pour le réentraînement (2 min/semaine)



CONCLUSION

La simplicité paie. Un LSTM bien conçu et régularisé peut surpasser un Transformer sur des données réelles.
Le feature engineering et la rigueur méthodologique comptent autant que l'architecture.

MERCI DE VOTRE ATTENTION

Giulia Governatori