D'accord ! Puisque tu as déjà les bases de données et que tu veux juste apprendre à utiliser **ConvLSTM** avec ces données, voici un guide simple et pratique. 🚀

**1. Charger les données de la base de données**

Puisque tu as des **images satellites** et des **données météorologiques**, il faut les récupérer et les préparer.

**a) Charger les images depuis la base de données**

Si tes images sont stockées dans une base **MySQL**, voici comment les extraire en Python :

import pymysql

import numpy as np

import cv2

# Connexion à la base de données

conn = pymysql.connect(host="localhost", user="root", password="", database="prediction\_db")

cursor = conn.cursor()

# Récupération des images triées par date

cursor.execute("SELECT image FROM images\_satellite ORDER BY timestamp ASC")

images = []

for row in cursor.fetchall():

img = np.asarray(bytearray(row[0]), dtype=np.uint8)

img = cv2.imdecode(img, cv2.IMREAD\_GRAYSCALE) # Convertir en niveaux de gris

img = cv2.resize(img, (64, 64)) # Redimensionner

images.append(img)

cursor.close()

conn.close()

# Conversion en tenseur pour ConvLSTM

images = np.array(images).reshape(-1, 10, 64, 64, 1) # (Batch, Time, Height, Width, Channels)

👉 **Explication** : On extrait 10 images consécutives pour chaque séquence temporelle.

**b) Charger les données météorologiques**

Si tes données météo sont dans **PostgreSQL** :

import psycopg2

import pandas as pd

conn = psycopg2.connect("dbname=prediction\_db user=postgres password=admin")

df = pd.read\_sql("SELECT \* FROM meteo ORDER BY timestamp ASC", conn)

conn.close()

# Sélection des variables météo et mise en forme

meteo\_data = df[['temperature', 'humidite', 'vent']].values.reshape(-1, 10, 3) # (Batch, Time, Features)

👉 **Explication** : On prend les 3 variables météo et on regroupe 10 enregistrements successifs.

**2. Construire le modèle ConvLSTM**

On va maintenant créer un modèle qui prend **les images et les données météo en entrée** et prédit une catastrophe.

from keras.models import Model

from keras.layers import ConvLSTM2D, BatchNormalization, Dropout, Dense, Flatten, Conv3D, Input, LSTM

# Entrée des images

input\_images = Input(shape=(10, 64, 64, 1))

x = ConvLSTM2D(filters=64, kernel\_size=(3, 3), activation='relu', padding='same', return\_sequences=True)(input\_images)

x = BatchNormalization()(x)

x = Dropout(0.2)(x)

x = ConvLSTM2D(filters=64, kernel\_size=(3, 3), activation='relu', padding='same', return\_sequences=False)(x)

x = Flatten()(x)

# Entrée des données météo

input\_meteo = Input(shape=(10, 3))

y = LSTM(32, return\_sequences=False)(input\_meteo)

# Fusion des deux entrées

z = Dense(64, activation='relu')((x + y))

z = Dense(1, activation='sigmoid')(z) # Prédiction binaire (catastrophe ou non)

# Définition du modèle

model = Model(inputs=[input\_images, input\_meteo], outputs=z)

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

👉 **Explication** :

* Le modèle prend **des images satellites et des données météo** comme entrées.
* Les **ConvLSTM2D** traitent les images, et les **LSTM** traitent les données météo.
* La sortie est une **prédiction de catastrophe (1 = oui, 0 = non)**.

**3. Entraîner le modèle**

model.fit([images, meteo\_data], labels, epochs=20, batch\_size=8, validation\_split=0.2)

👉 **Explication** :

* images et meteo\_data sont nos **données d'entrée**.
* labels est un tableau [0, 1, 0, 1, ...] indiquant si une catastrophe s'est produite après ces observations.
* epochs=20 signifie que l'on entraîne le modèle pendant 20 cycles.

**4. Faire une prédiction**

prediction = model.predict([images[:1], meteo\_data[:1]])

print("Probabilité de catastrophe :", prediction[0][0])

👉 **Explication** : On fait une prédiction pour une seule séquence.

**Conclusion**

✅ Maintenant, tu sais **comment utiliser ConvLSTM avec ta base de données** ! 🚀  
Si tu veux que j’adapte le code selon tes données, donne-moi plus de détails sur ta base. 😊