MULTIMODAL PNEUMONIA DETECTION SYSTEM

Complete Technical Report — Dual-Branch Fusion Pipeline

CSV Clinical Data · Chest X-Ray Images · Fusion Architecture · Full ML Pipeline

University of Saida, Algeria · AI & Medical Imaging · 2025–2026

Abstract. This report describes the complete design and implementation plan for a multimodal AI system that combines two independent data sources — a structured clinical CSV dataset (patient vitals and symptoms) and a chest X-ray image dataset — to detect and classify pediatric pneumonia. The report covers: (1) both dataset descriptions and download links, (2) the pediatric filtering problem in the CSV data and its solution, (3) the full dual-branch parallel training pipeline, (4) input/output specifications at every stage, (5) all fusion strategies with code, (6) the complete tabular ML pipeline with 7 models and cross-validation, and (7) implementation roadmap. Current project status: literature review and preprocessing complete — model training is the next step.

Branch	Dataset Name	Type	Kaggle Link
1 — Clinical	essienmary/pneumonia-dataset	CSV — vitals, symptoms, age, label	kaggle.com/datasets/essienmary/pneumonia-dataset
2 — X-Ray	paultimothymooney/chest-xray-pneumonia	JPEG chest X-rays — Normal / Pneumonia	kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

PROJECT TEAM — University of Saida, Algeria

Role	Name	Contact / Institution
Data Engineer	**Bouhmidi Amina Maroua**	University of Saida
ML Engineer	**Labani Nabila Nour Elhouda**	University of Saida
Project Manager	**Kassouar Fatima**	kassouarfatima48@gmail.com
Business Model	**Miloudi Meroua Amira**	University of Saida
Academic Supervisor	**Dr. Abderrahmane Khiat**	University of Saida, Algeria
Medical Advisor	**Dr. Aimer Mohammed Djamel Eddine**	

TABLE OF CONTENTS

SECTION 1 — DATASET DESCRIPTIONS

1.1 Branch A — Clinical CSV Dataset

Name: Pneumonia Patient Dataset | **Author:** essienmary
Link: https://www.kaggle.com/datasets/essienmary/pneumonia-dataset
Type: Structured tabular CSV — patient vitals, clinical symptoms, lab indicators, diagnosis label

This dataset contains structured clinical records for patients with and without pneumonia. Each row represents one patient with measurements of vital signs and symptoms collected at the time of hospital visit. The dataset includes patients of **all age groups** — from infants to elderly — which creates a critical problem for this project (see Section 2).

Column Name	Data Type	Description	Clinical Role
patient_id	Integer / ID	Unique patient identifier	Drop — not a predictive feature
age	Numeric (years)	Patient age	Critical for pediatric filter
sex	Categorical M/F	Biological sex	Encode: M=0, F=1
temperature	Numeric (°C)	Body temperature	Fever >38.5°C = active infection
spo2	Numeric (%)	Blood oxygen saturation	<95% = respiratory compromise
heart_rate	Numeric (bpm)	Heart beats per minute	Tachycardia common in pneumonia
respiratory_rate	Numeric (/min)	Breathing rate	>40/min in children = tachypnea
cough	Binary 0/1	Cough present or absent	Primary pneumonia symptom
label	Binary 0/1	0=Normal, 1=Pneumonia	Main classification target
pneumonia_severity	Ordinal 4-class	none / mild / moderate / severe	Secondary target for severity scoring

1.2 Branch B — Chest X-Ray Image Dataset

Name: Chest X-Ray Images (Pneumonia) — Kermany et al., 2018
Link: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia
Type: JPEG chest X-ray images — 5,856 images — **Pediatric patients aged 1–5 years only**

This dataset contains frontal chest X-rays collected from the Guangzhou Women and Children's Medical Center, China. All images were graded by two expert physicians and a third for verification. Images are stored in train/val/test folders with NORMAL and PNEUMONIA subfolders.

Split	NORMAL	PNEUMONIA	Total	Problem / Note
Train	1,341	3,875	**5,216**	Class imbalance 1:2.89 — needs class weighting
Validation	8	8	**16**	TOO SMALL — must be replaced with proper stratified split
Test	234	390	**624**	Final held-out evaluation set — do not touch during training
TOTAL	**1,583**	**4,273**	**5,856**	Stratified 70/15/15 split recommended (team fix)

SECTION 2 — THE PROBLEM: CSV IS NOT PEDIATRIC-ONLY + SOLUTION

2.1 The Core Problem

CRITICAL MISMATCH: The X-ray dataset (Branch B) contains ONLY pediatric patients aged 1–5 years. The CSV dataset (Branch A) contains patients of ALL ages — adults, elderly, and children mixed together. Using the full CSV with pediatric X-rays makes the fusion model clinically inconsistent and scientifically invalid.

Pneumonia has completely different clinical presentations depending on age. Vital sign ranges, typical pathogens, and X-ray patterns differ fundamentally between a 2-year-old child and a 70-year-old adult. A model trained on mixed-age clinical data and pediatric X-rays would learn incorrect cross-modal relationships.

Age Group	Normal SpO2	Resp. Rate (/min)	Typical Pathogen	X-Ray Pattern
Infant 0–1 yr	95–100%	30–60	RSV, viral	Bilateral haziness
Child 1–5 yr ★	97–100%	24–40	Viral, S. pneumoniae	Lobar / perihilar
Adult 18–60 yr	95–99%	12–20	S. pneumoniae, atypical	Lobar consolidation
Elderly 60+ yr	92–97%	15–25	Multi-resistant strains	Diffuse bilateral

★ Only this age group exists in the X-ray dataset — the two branches must match this population.

2.2 The Solution — Filter CSV to Pediatric Ages 1–5

SOLUTION: Before any training, filter the CSV dataset to keep ONLY rows where age is between 1 and 5 years. This aligns both branches on the same patient population and makes the fusion clinically valid. If too few samples remain after filtering, expand to ages 0–12.

Step-by-Step Filtering Code:

```
import pandas as pd # Step 1 — Load the full CSV df = pd.read_csv('pneumonia_dataset.csv') print(f'Full
dataset shape: {df.shape}') print(f'Age range: {df["age"].min():.0f} to {df["age"].max():.0f} years')
print(f'Age distribution:\n{df["age"].describe()}') # Step 2 — Check class balance BEFORE filter
print(f'\nLabel balance (full):') print(df['label'].value_counts()) # Step 3 — Apply pediatric filter
(ages 1–5, same as X-ray dataset) df_kids = df[(df['age'] >= 1) & (df['age'] <= 5)].copy()
print(f'\nPediatric dataset (ages 1-5): {len(df_kids)} rows') print(f'Label balance (pediatric):')
print(df_kids['label'].value_counts()) # Step 4 — If sample size is too small, expand to all pediatric
(0–12) if len(df_kids) < 200: print('WARNING: too few samples. Expanding to ages 0–12.') df_kids =
df[(df['age'] >= 0) & (df['age'] <= 12)].copy() print(f'Expanded pediatric dataset: {len(df_kids)} rows')
# Step 5 — Save filtered version for use in Branch A df_kids.to_csv('pneumonia_dataset_pediatric.csv',
index=False) print('Saved: pneumonia_dataset_pediatric.csv')
```

Ste p	Action	Why
1	**Load full CSV and check age distribution**	Understand the age spread in the dataset
2	**Filter: age >= 1 AND age <= 5**	Match the X-ray dataset population exactly
3	**Check class balance after filter**	Ensure enough positive and negative cases remain
4	**Expand to 0–12 if fewer than 200 rows remain**	Statistical minimum for reliable model training
5	**Save as pneumonia_dataset_pediatric.csv**	Use this file exclusively for Branch A training

SECTION 3 — FULL MULTIMODAL PIPELINE OVERVIEW

3.1 The Big Picture — Mimicking a Real Doctor

A real physician diagnoses pneumonia by consulting two sources simultaneously: the **patient chart** (vitals, symptoms, history) AND the **chest X-ray**. This system mirrors that workflow exactly. Two AI models are trained in parallel, each on one data source, and their outputs are fused for a final prediction. This is called a **Multimodal Fusion Architecture**.

INPUT A — Clinical CSV (after pediatric filter: ages 1–5) age, spo2, temperature, heart_rate, cough, sex...

INPUT B — Chest X-Ray Images JPEG 224×224 px NORMAL vs PNEUMONIA Pediatric patients ages 1–5

PREPROCESSING A • Filter age 1–5 • Encode: sex, severity • Impute missing values • StandardScaler • SMOTE (train fold only) • 80/20 stratified split

PREPROCESSING B • Resize to 224×224 • Normalize ÷255 • Augmentation (flip/rotate/zoom) • Fix validation: 70/15/15 split • Class weights 1:2.89

TRAIN IN PARALLEL (independent datasets)

BRANCH A — Tabular Models (trained in parallel) 1. Logistic Regression 2. Random Forest 3. XGBoost 4. LightGBM 5. SVM (RBF) 6. KNN 7. MLP Neural Net → Best model by AUC selected

BRANCH B — CNN Models (8 architectures) 1. MobileNetV2 2. AlexNet 3. ResNet50V2 4. InceptionV3 5. DenseNet121 6. EfficientNetB3 7. ViT 8. VGG16 → Best CNN by AUC selected

OUTPUT A p1 = P(pneumonia) from best tabular model or embedding vector [64d]

OUTPUT B p2 = P(pneumonia) from best CNN or embedding vector [64d]

FUSION LAYER Late Fusion: p_final = (w1×p1 + w2×p2) / (w1+w2) or Intermediate Fusion: Concatenate([embedding_A, embedding_B]) → Dense head

FINAL OUTPUT Pneumonia: YES / NO · Severity: none / mild / moderate / severe · Confidence score [0–1]

Figure 1: Complete multimodal pipeline. Both branches trained in parallel on independent datasets, fused for final prediction.

3.2 Input/Output Specification at Each Stage

Stage	Branch	Input	Output	Shape
1. Raw CSV	A	pneumonia_dataset.csv	Full DataFrame — all ages	N rows × 10 cols
2. Filter	A	All ages mixed	Pediatric only (ages 1–5)	n rows × 10 cols
3. Preprocess A	A	Raw pediatric rows	Normalized + encoded feature matrix	[n, 16] float32
4. Raw X-Ray	B	JPEG files in folders	Pixel arrays	[H, W, 3] uint8
5. Preprocess B	B	Raw JPEG	Normalized float image	[224, 224, 3] float32
6. Train Tabular	A	Feature matrix [n, 16]	7 trained models + CV metrics	prob p1 ∈ [0,1]
7. Train CNN	B	Image batch [B, 224, 224, 3]	8 trained CNNs + metrics	prob p2 ∈ [0,1]
8. Select Best	A+B	CV and val AUC scores	1 tabular model + 1 CNN selected	Two models
9. Late Fusion	A+B	p1 (tabular) + p2 (CNN)	Weighted average p_final	scalar ∈ [0,1]
10. Decision	A+B	p_final	PNEUMONIA/NORMAL + severity	Label + score

SECTION 4 — PREPROCESSING PIPELINES

4.1 Branch A — Clinical CSV Preprocessing

All preprocessing steps are applied **inside a scikit-learn Pipeline** to prevent data leakage. SMOTE is applied only inside each cross-validation fold on the training portion. The scaler is fit on training data only and applied to validation/test.

Step	Operation	Detail / Reason
1	**Load pediatric CSV**	Use pneumonia_dataset_pediatric.csv (ages 1–5 only)
2	**Drop patient_id**	Not a predictive feature — unique identifier only
3	**Encode sex**	LabelEncoder: Male=0, Female=1
4	**Encode severity**	OrdinalEncoder: none=0, mild=1, moderate=2, severe=3
5	**Median imputation**	SimpleImputer(strategy=median) for missing vitals
6	**Feature engineering**	Create: danger_score = (fever) + (low_spo2) + (tachypnea)
7	**Stratified 80/20 split**	Preserve class ratio in train and test
8	**StandardScaler**	Fit on X_train only, transform X_train and X_test
9	**SMOTE (train fold only)**	Synthetic oversampling — applied inside CV folds only to prevent leakage

Full Preprocessing Code — Branch A:

```
import pandas as pd import numpy as np from sklearn.preprocessing import LabelEncoder, OrdinalEncoder,
StandardScaler from sklearn.impute import SimpleImputer from sklearn.model_selection import
train_test_split from imblearn.over_sampling import SMOTE from imblearn.pipeline import Pipeline as
ImbPipeline # Load filtered pediatric CSV df = pd.read_csv('pneumonia_dataset_pediatric.csv') # Encode
categorical columns le = LabelEncoder() df['sex'] = le.fit_transform(df['sex']) oe =
OrdinalEncoder(categories=[['none','mild','moderate','severe']]) df['pneumonia_severity'] =
oe.fit_transform(df[['pneumonia_severity']]) # Feature engineering df['high_fever'] = (df['temperature']
> 38.5).astype(int) df['low_spo2'] = (df['spo2'] < 95).astype(int) df['tachypnea'] =
(df['respiratory_rate'] > 40).astype(int) df['danger_score'] = df['high_fever'] + df['low_spo2'] +
df['tachypnea'] # Drop ID and prepare X, y df.drop('patient_id', axis=1, inplace=True, errors='ignore') X
= df.drop('label', axis=1) y = df['label'] # Stratified 80/20 split X_train, X_test, y_train, y_test =
train_test_split( X, y, test_size=0.2, random_state=42, stratify=y ) print(f'Train: {len(X_train)} | Test:
{len(X_test)}') print(f'Features: {X_train.shape[1]}') print(f'Class balance (train):
{y_train.value_counts().to_dict()}')
```

4.2 Branch B — X-Ray Image Preprocessing

Step	Operation	Value / Detail
1	**Fix validation split**	Replace 16-image val set with stratified 70/15/15 (4099/878/879)
2	**Resize**	224×224 px — standard ImageNet input size
3	**Convert to float32**	Required — float16 causes sigmoid overflow (NaN loss)
4	**Normalize**	÷255 → [0,1] for all models except MobileNetV2 (÷127.5-1 → [-1,1])
5	**Augmentation (train only)**	Horizontal flip, rotate ±36°, zoom ±15%, brightness ±15%
6	**Class weighting**	w_normal=2.89, w_pneumonia=1.0 to compensate 1:2.89 imbalance
7	**Batching**	Batch size 32 (GPU) or 16 (CPU). Prefetch for speed.

CRITICAL: MobileNetV2 requires pixel values in [-1, 1] (divide by 127.5 then subtract 1). All other models use [0, 1] (divide by 255). Mixing these reduces accuracy by 5–15% silently. Also: the final Dense and Sigmoid layers MUST use dtype=float32 to avoid NaN loss under FP16.

SECTION 5 — BRANCH A: 7 TABULAR MODELS WITH CROSS-VALIDATION

5.1 Model Overview

Each of the 7 models is wrapped in a scikit-learn Pipeline (Imputer → Scaler → SMOTE → Model) and evaluated with **5-Fold Stratified Cross-Validation**. Reported metrics for each fold: Accuracy, F1, Precision, Recall, ROC-AUC. Best model selected by AUC for the fusion step.

No.	Model	Type	Key Strength	Extra Output
1	**Logistic Regression**	Linear	Interpretable baseline — coefficient per feature	Feature coefficients
2	**Random Forest**	Ensemble trees	Robust to outliers — feature importance ranking	Feature importance plot
3	**XGBoost**	Gradient boosting	Best accuracy on tabular medical data — handles imbalance	SHAP values
4	**LightGBM**	Gradient boosting	Faster than XGBoost — good for larger filtered datasets	Gain importance
5	**SVM (RBF kernel)**	Kernel method	Strong on small datasets — effective with scaled features	Support vectors
6	**KNN**	Distance-based	Non-parametric — useful as reference baseline	Decision boundary
7	**MLP Neural Net**	Neural network	3 hidden layers (128 → 64 → 32) — learns non-linear patterns	Loss curves

5.2 Cross-Validation Framework

Strategy: 5-Fold Stratified Cross-Validation — each fold preserves the class ratio.
Pipeline order: SimpleImputer → StandardScaler → SMOTE → Model (SMOTE only on train fold)
Metrics per fold: Accuracy, F1 (macro), Precision (macro), Recall (macro), ROC-AUC
Final report: Mean ± Std across 5 folds + test set metrics for the best model

Cross-Validation Code — All 7 Models:

```
import numpy as np from sklearn.linear_model import LogisticRegression from sklearn.ensemble import
RandomForestClassifier from sklearn.svm import SVC from sklearn.neighbors import KNeighborsClassifier from
sklearn.neural_network import MLPClassifier from xgboost import XGBClassifier from lightgbm import
LGBMClassifier from sklearn.impute import SimpleImputer from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold, cross_validate from imblearn.over_sampling import
SMOTE from imblearn.pipeline import Pipeline as ImbPipeline # Define all 7 models MODELS = { 'Logistic
Regression': LogisticRegression(max_iter=1000, random_state=42), 'Random Forest':
RandomForestClassifier(n_estimators=200, random_state=42), 'XGBoost': XGBClassifier(n_estimators=200,
eval_metric='logloss', random_state=42), 'LightGBM': LGBMClassifier(n_estimators=200, random_state=42),
'SVM (RBF)': SVC(kernel='rbf', probability=True, random_state=42), 'KNN':
KNeighborsClassifier(n_neighbors=5), 'MLP': MLPClassifier(hidden_layer_sizes=(128,64,32), max_iter=500,
random_state=42), } SCORING = ['accuracy','f1_macro','precision_macro','recall_macro','roc_auc'] cv =
StratifiedKFold(n_splits=5, shuffle=True, random_state=42) results = {} for name, model in MODELS.items():
pipe = ImbPipeline([ ('imputer', SimpleImputer(strategy='median')), ('scaler', StandardScaler()),
('smote', SMOTE(random_state=42)), ('model', model), ]) scores = cross_validate(pipe, X_train, y_train,
cv=cv, scoring=SCORING, return_train_score=True) results[name] = { 'Accuracy':
f"{scores['test_accuracy'].mean():.3f} ± {scores['test_accuracy'].std():.3f}", 'F1':
f"{scores['test_f1_macro'].mean():.3f} ± {scores['test_f1_macro'].std():.3f}", 'AUC':
f"{scores['test_roc_auc'].mean():.3f} ± {scores['test_roc_auc'].std():.3f}", } print(f'{name}: AUC =
{scores["test_roc_auc"].mean():.3f}') # Select best model by AUC best_name = max(results, key=lambda k:
float(results[k]['AUC'].split(' ')[0])) print(f'\nBest model: {best_name}')
```

5.3 Per-Model Evaluation — Additional Outputs

Model	Confusion Matrix	ROC Curve	Learning Curve	Feature Importance
Logistic Regression	**Yes**	**Yes**	**Yes**	Coefficients
Random Forest	**Yes**	**Yes**	**Yes**	Gini importance bar chart
XGBoost	**Yes**	**Yes**	**Yes**	SHAP summary plot
LightGBM	**Yes**	**Yes**	**Yes**	Gain importance
SVM (RBF)	**Yes**	**Yes**	**Yes**	Support vectors only
KNN	**Yes**	**Yes**	**Yes**	K-value sensitivity plot
MLP Neural Net	**Yes**	**Yes**	**Yes**	Train/val loss curves

SECTION 6 — BRANCH B: 8 CNN ARCHITECTURES + TRANSFER LEARNING

6.1 Architecture Comparison

Model	Params	Top-1	Speed	Key Innovation	Best Use
MobileNetV2	3.4 M	72.0%	Fastest	Inverted residuals, depthwise conv	CPU/mobile deployment
AlexNet*	60 M	—	Slow	First deep CNN — trained from scratch	Historical baseline
ResNet50V2	25 M	75.6%	Medium	Skip connections: output = F(x)+x	Robust baseline
InceptionV3	23 M	77.9%	Medium	Parallel 1×1, 3×3, 5×5 convolutions	Multi-scale CXR features
DenseNet121	8 M	74.7%	Fast	All layers connect to all prior layers	CheXNet backbone — recommended
EfficientNetB3	12 M	81.6%	Medium	Compound depth/width/resolution scaling	Best accuracy/size ratio
ViT	86 M	81.8%	Slow	Image patches as transformer tokens	Global bilateral patterns
VGG16	138 M	71.5%	Slowest	Deep uniform 3×3 conv stacks	Heavy reference baseline

** AlexNet has no ImageNet pre-training — trained from scratch on X-ray data only.*

6.2 Transfer Learning Strategy — 2-Phase Training

Parameter	Phase 1 — Head Training	Phase 2 — Fine-Tuning
Backbone	ALL layers frozen	Last 20 layers trainable
Learning Rate	1e-3 (higher — head not pre-trained)	1e-5 (very small — prevent forgetting)
Epochs	25 max (early stopping patience=8)	15 max (early stopping patience=6)
Batch Size	32 (GPU) / 16 (CPU)	32 (GPU) / 8 (CPU)
Risk	LR too small → slow convergence	LR too large → catastrophic forgetting

Classification Head — same for all 8 architectures:

```
from tensorflow.keras import Model from tensorflow.keras.layers import GlobalAveragePooling2D,
BatchNormalization from tensorflow.keras.layers import Dense, Dropout, Activation from
tensorflow.keras.applications import DenseNet121 # example def build_model(base_app, input_shape=(224,
224, 3)): base = base_app(include_top=False, weights='imagenet', input_shape=input_shape) base.trainable =
False # Phase 1: freeze all x = GlobalAveragePooling2D()(base.output) x = BatchNormalization()(x) x =
Dense(512, activation='relu')(x) x = Dropout(0.5)(x) x = Dense(256, activation='relu')(x) x =
Dropout(0.3)(x) # MUST be float32 — float16 causes NaN loss x = Dense(1, dtype='float32')(x) output =
Activation('sigmoid', dtype='float32')(x) return Model(inputs=base.input, outputs=output)
```

SECTION 7 — FUSION STRATEGIES

7.1 The Core Challenge — Datasets Are Not Paired

Key constraint: The two datasets do NOT share the same patients. There is no row in the CSV that corresponds to a specific X-ray image. This means end-to-end joint training is not possible with these datasets. The correct approach is Late Fusion (train separately, combine outputs). Intermediate Fusion requires a paired dataset like MIMIC-CXR.

7.2 Strategy 1 — Late Fusion (Recommended for this project)

Each model is trained completely independently on its own dataset. At inference time, both models produce a pneumonia probability and the outputs are combined with a weighted average. Weights are set to the validation AUC of each model.

```
# Both models are already trained and saved import numpy as np def late_fusion_predict(patient_clinical,
patient_xray, tabular_model, cnn_model, w1, w2): # w1, w2 = validation AUC of each model # Branch A —
tabular prediction p1 = tabular_model.predict_proba(patient_clinical)[0][1] # Branch B — CNN prediction p2
= float(cnn_model.predict(patient_xray)[0][0]) # Weighted average fusion p_final = (w1 * p1 + w2 * p2) /
(w1 + w2) # Decision label = 'PNEUMONIA' if p_final > 0.5 else 'NORMAL' print(f'Clinical: {p1:.3f} |
X-Ray: {p2:.3f} | Fused: {p_final:.3f} → {label}') return p_final, label # Example usage after training:
# w1 = 0.87 (tabular model validation AUC) # w2 = 0.97 (CNN validation AUC) # result =
late_fusion_predict(X_new, img_new, tab_model, cnn_model, w1, w2)
```

7.3 Strategy 2 — Intermediate Fusion (Advanced — needs paired data)

Instead of combining probabilities, extract the internal embeddings from each model, concatenate them into a joint vector, and train a new classification head on top. The fusion layer learns which cross-modal combinations are most predictive.

```
from tensorflow.keras import Input, Model from tensorflow.keras.layers import Dense, Dropout, Concatenate,
GlobalAveragePooling2D from tensorflow.keras.applications import EfficientNetB3 # BRANCH A — Tabular MLP
producing 64-dim embedding tabular_input = Input(shape=(16,), name='clinical_input') x1 = Dense(128,
activation='relu')(tabular_input) x1 = Dropout(0.3)(x1) tabular_emb = Dense(64,
name='clinical_embedding')(x1) # [64] # BRANCH B — CNN producing 64-dim embedding base_cnn =
EfficientNetB3(include_top=False, weights='imagenet', input_shape=(224, 224, 3)) x2 =
GlobalAveragePooling2D()(base_cnn.output) x2 = Dense(128, activation='relu')(x2) image_emb = Dense(64,
name='image_embedding')(x2) # [64] # FUSION — concatenate: [64] + [64] = [128] merged =
Concatenate()([tabular_emb, image_emb]) # [128] merged = Dense(64, activation='relu')(merged) merged =
Dropout(0.4)(merged) output = Dense(1, activation='sigmoid', dtype='float32')(merged) # Joint model —
needs PAIRED data (same patient in both modalities) fusion_model = Model( inputs=[tabular_input,
base_cnn.input], outputs=output, name='intermediate_fusion' ) # For unpaired data: use Late Fusion above.
# For paired data (MIMIC-CXR): use this intermediate fusion.
```

Property	Late Fusion	Intermediate Fusion
What is combined	Output probabilities p1 and p2	Internal embedding vectors (64-dim each)
Data required	Independent datasets — no pairing needed	Paired: same patient in both CSV and X-ray
Learns cross-modal	No — static weighted average	Yes — fusion layer learns interactions
Complexity	Low — 1 line of code	High — joint training required
For this project	**RECOMMENDED — use now**	Future work with MIMIC-CXR

SECTION 8 — EVALUATION METRICS & VALIDATION

8.1 Why Accuracy Alone Is Not Enough

In pneumonia diagnosis, a missed case (false negative) is far more dangerous than a false alarm (false positive). Standard accuracy is misleading with class imbalance and does not distinguish these clinically different errors. A full set of metrics is required.

Metric	Formula	Clinical Meaning	Target
Accuracy	(TP+TN)/(TP+TN+FP+FN)	Overall correct classification rate	>90%
Sensitivity/Recall	TP/(TP+FN)	Rate of real pneumonia caught — MOST CRITICAL	>95%
Specificity	TN/(TN+FP)	Rate of normal cases correctly identified	>85%
Precision	TP/(TP+FP)	% of predicted pneumonia that are truly pneumonia	>85%
F1 Score	2×(P×R)/(P+R)	Harmonic balance of precision and recall	>90%
ROC-AUC	Area under ROC curve	Overall discriminative power — primary metric	>0.95
PR-AUC	Area under PR curve	More reliable than ROC under class imbalance	>0.90

8.2 12 Plots Generated Per Model

No.	Plot	What It Shows
1	**Histograms + KDE**	Distribution of each clinical feature — skew, outliers, normality
2	**Boxplots by label**	Feature values split by Normal vs Pneumonia — t-test p-value shown
3	**Violin plots**	Full distribution shape and density by diagnosis label
4	**Target balance pie + bar**	Label balance, severity breakdown, sex distribution
5	**Binary signs bar**	Prevalence of each binary clinical sign (cough, fever, etc.)
6	**Correlation heatmap**	Full Pearson correlation matrix — feature interdependencies
7	**Missing values heatmap**	Pattern and percentage of missing values per feature
8	**Age analysis KDE + box**	Age distribution colored by severity level
9	**Pairplot (key vitals)**	Scatter matrix of spo2, temperature, heart_rate colored by label
10	**Confusion matrix**	TP/TN/FP/FN per model on test set
11	**ROC + PR curves (all 7)**	Combined ROC and Precision-Recall curves for all models on one plot
12	**Feature importance / SHAP**	Which clinical features most influence each model's prediction

SECTION 9 — IMPLEMENTATION ROADMAP & CURRENT STATUS

Phase	Name	Status	Key Deliverables
1	**Literature Review**	**COMPLETE**	CNN architectures, tabular ML, fusion strategies, medical AI papers studied
2	**EDA — Both Datasets**	**COMPLETE**	Age distributions, class imbalance identified, validation set problem found
3	**CSV Pediatric Filter**	**COMPLETE**	Filter to ages 1–5 coded and verified. pediatric_pneumonia.csv created.
4	**Preprocessing — Both Branches**	**COMPLETE**	Branch A: encode+scale+SMOTE pipeline. Branch B: resize+normalize+augment+fix split.
5	**Branch A Training (7 models)**	**NOT STARTED**	Run 5-fold CV for all 7 tabular models. Select best by AUC.
6	**Branch B Training (8 CNNs)**	**NOT STARTED**	Train 8 CNN architectures with 2-phase transfer learning. Select best by val AUC.
7	**Late Fusion Implementation**	DESIGNED	Code ready — awaits trained models. Weight by AUC of each branch.
8	**Full Evaluation + Report**	PLANNED	Confusion matrices, ROC curves, SHAP, final comparison of all models.
9	**Multi-Dataset Expansion**	FUTURE	Add NIH ChestX-ray14 + CheXpert for generalised model beyond pediatric.
10	**Intermediate Fusion (paired data)**	FUTURE	MIMIC-CXR paired data. Joint embedding training.

SECTION 10 — CONCLUSION & RECOMMENDATIONS

This report has described the complete design of a multimodal AI pneumonia detection system combining clinical tabular data and chest X-ray images. The dual-branch architecture mirrors real clinical practice and addresses a critical diagnostic need in under-resourced settings like Algeria, where specialist radiologists are scarce and diagnostic delays are long.

Topic	Conclusion
CSV age problem	Filter to ages 1–5 before training. Use pneumonia_dataset_pediatric.csv for Branch A only.
Best fusion now	Late Fusion is the correct strategy for these unpaired datasets. Simple and effective.
Best tabular model (expected)	XGBoost or LightGBM based on clinical tabular literature. Confirmed after 5-fold CV.
Best CNN (expected)	DenseNet121 (CheXNet backbone) or EfficientNetB3. Confirmed after training.
Current status	Literature review and preprocessing complete. Branch A and B training is the next step.
Future novelty	Confidence-aware rejection (uncertain cases sent to radiologist), severity scoring (0–3), multi-dataset expansion.
Publication potential	The Algerian-context multimodal system with local EMR integration is a genuine contribution. Publishable in IEEE Access or Frontiers in Digital Health after training results.

Contact for more information, collaboration, or questions:
Kassouar Fatima (Project Manager) — kassouarfatima48@gmail.com
GitHub: github.com/AminaMar/pediatric-pneumonia-detection
University of Saida, Algeria · 2025–2026